



**PSGR  
Krishnammal College for Women**



**College of Excellence, Autonomous and Affiliated to Bharathiar University  
Accredited with A++ grade by NAAC, An ISO 9001: 2015 Certified Institution  
Peelamedu, Coimbatore-641004**

**DEPARTMENT OF COMPUTER SCIENCE (PG)**

**ADVANCED JAVA PROGRAMMING LAB (MCS23P5)**

**2024-2025**



**PSGR  
Krishnammal College for Women**



**College of Excellence, Autonomous and Affiliated to Bharathiar University  
Accredited with A++ grade by NAAC, An ISO 9001: 2015 Certified Institution  
Peelamedu, Coimbatore-641004**

**DEPARTMENT OF COMPUTER SCIENCE (PG)  
ADVANCED JAVA PROGRAMMING LAB (MCS23P5)**

**REGISTER NUMBER: \_\_\_\_\_**

Certified that this is a bonafide record work done by \_\_\_\_\_ of  
II M.Sc. Computer Science during the year 2024-2025.

**FACULTY INCHARGE**

**HEAD OF THE DEPARTMENT**

Submitted for the practical examination held on \_\_\_\_\_ at PSGR  
Krishnammal College for Women, Coimbatore.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## INDEX

S No.	Date	Topics	Page No	Sign
1		Server-Client Communication using socket		
2		Simple calculator using servlet		
3		Login validation using servlet		
4		Validation using request dispatcher		
5		Session tracking using servlet		
6		Exceptions in spring for invalid bank transaction		
7		Date injection in spring bean		
8		Employee details using hibernate		
9		JSP application with intrinsic objects		
10		Passing values with validation in JSP		
11		Expression language using JSP		
12		Login form validation using JavaBeans		
13		File uploading using JSP		
14		Electricity bill generation		
15		Resume builder servlet application		

<b>Ex No: 1</b>	<b>SERVER-CLIENT COMMUNICATION USING SOCKET</b>
<b>Date:</b>	

**AIM**

To develop Java socket programming to read and write on servers and clients.

**ALGORITHM**

**STEP 1:** Create a new project for the server.

**STEP 2:** Add the MyServer class to the server project by right-clicking on the MyServerProject and selecting New > Java Class.

**STEP 3:** Create a new project for the client.

**STEP 4:** Add the MyClient class to the client project by right-clicking on the MyClientProject and selecting New > Java Class.

**STEP 5:** Run the server and the client.

**STEP 6:** View the output.

**STEP 7:** Stop the process.

## PROGRAM

```
package myclient;

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

public class MyClient {
    public static void main(String[] args) {
        try{
            try (Socket s = new Socket("localhost",6666);
                DataOutputStream dout = new DataOutputStream(s.getOutputStream())) {
                dout.writeUTF("Hello Server");
                dout.flush();
            } catch(IOException e){System.out.println(e);}
        }
    }
}

package myserver;
import java.io.*;
import java.net.*;

public class MyServer {
    public static void main(String[] args) {
        try{

            try (ServerSocket ss = new ServerSocket(6666)) {
                Socket s=ss.accept();//establishes connection

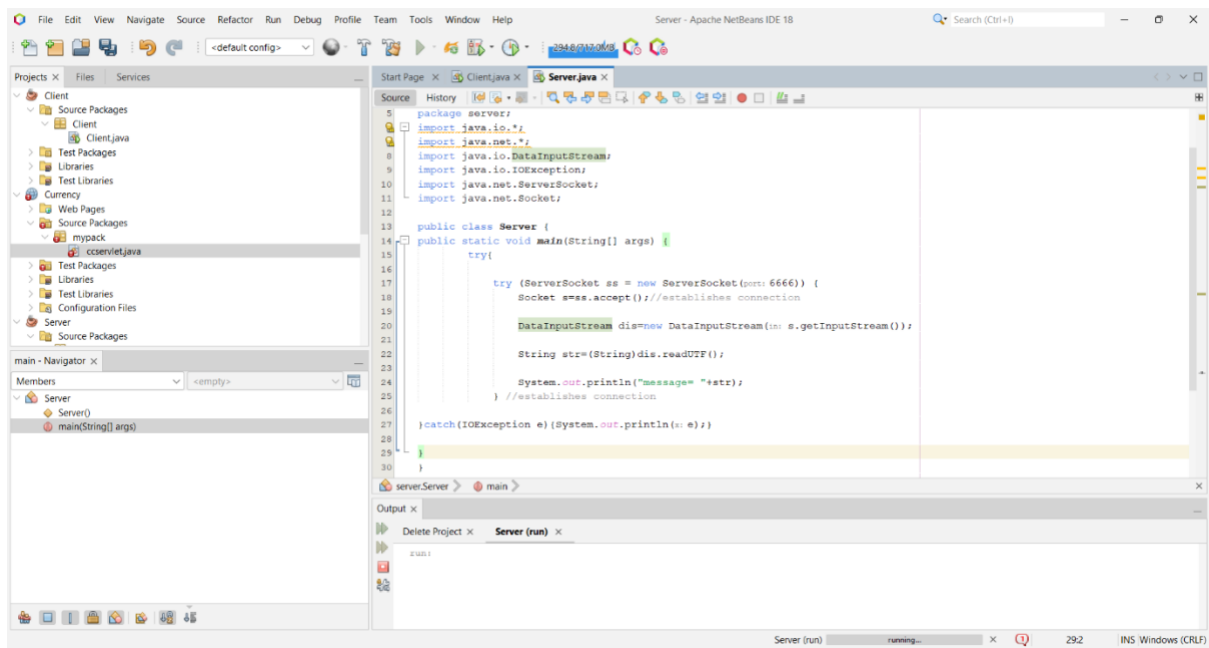
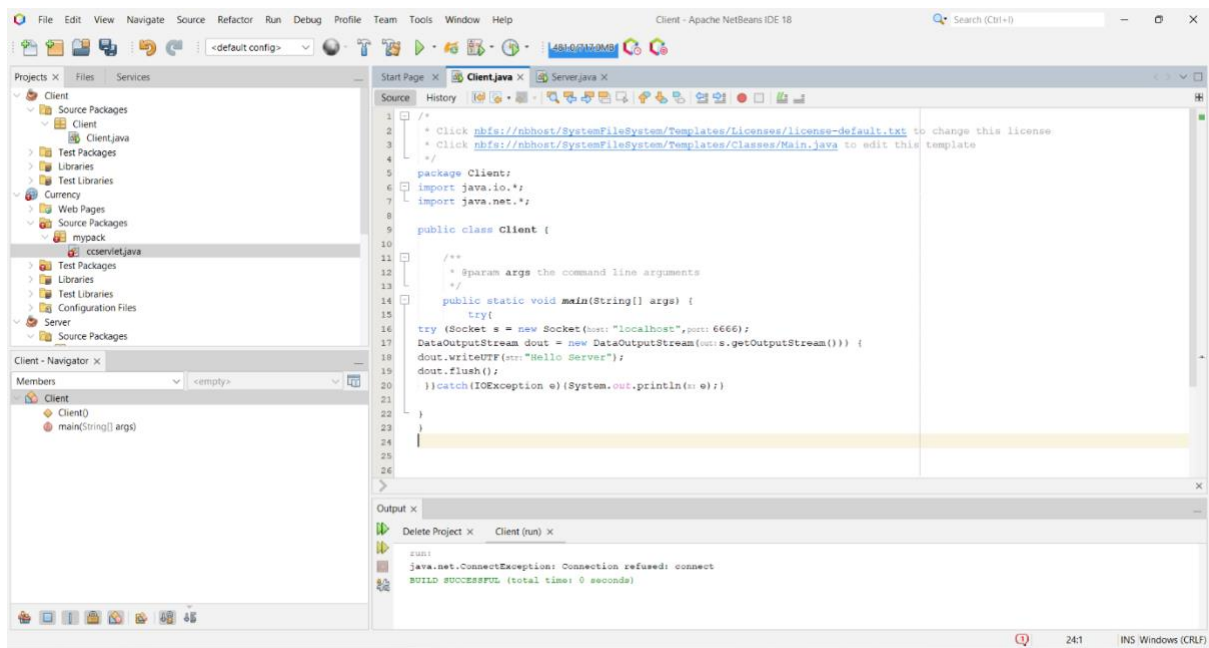
                DataInputStream dis=new DataInputStream(s.getInputStream());

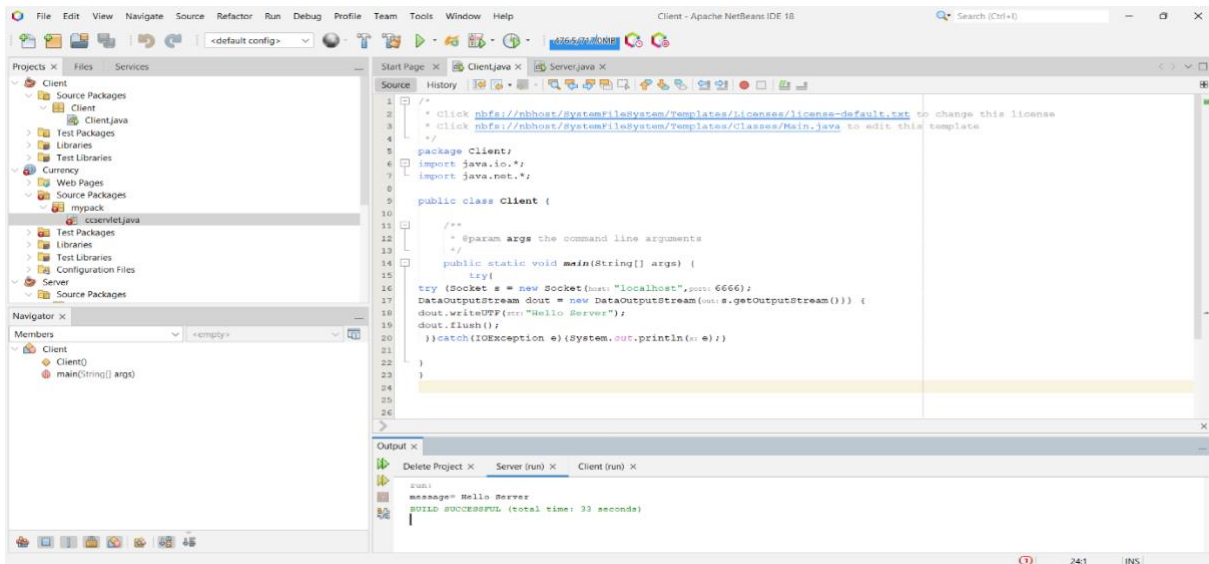
                String str=(String)dis.readUTF();

                System.out.println("message= "+str);
            } //establishes connection

        } catch(IOException e){System.out.println(e);}
    }
}
```

## OUTPUT





## RESULT

Thus, the program is executed successfully and output is verified.

<b>Ex no:2</b>	<b>SIMPLE CALCULATOR USING SERVLET</b>
<b>Date:</b>	

## **AIM**

To develop a web-based calculator application using Java Servlets that performs basic arithmetic operations (addition, subtraction, multiplication, and division) and displays the result to the user.

## **ALGORITHM**

**STEP 1:** Create an HTML form that allows users to input two numbers and select an arithmetic operation.

**STEP 2:** Develop a Java Servlet to handle HTTP POST requests from the HTML form.

**STEP 3:** Extract the input values (numbers and operation) from the request parameters.

**STEP 4:** Use a switch-case structure to perform the specified arithmetic operation based on user input.

**STEP 5:** Handle special cases, such as division by zero.

**STEP 6:** Generate an HTML response displaying the calculation result or an error message.

**STEP 7:** Provide a link to return to the input form.



## PROGRAM

### Index.html:

```
<html>

<head>

    <title>Calculator</title>

</head>

<body>

    <form action="Calculator1" method="post">

        <input type="number" name="num1" placeholder="Number 1">

        <input type="number" name="num2" placeholder="Number 2">

        <select name="operation">

            <option value="add">Add</option>

            <option value="subtract">Subtract</option>

            <option value="multiply">Multiply</option>

            <option value="divide">Divide</option>

        </select>

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

    </form>

</body>

</html>
```

### Calculator1.java:

```
import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;
```

```

@WebServlet("/Calculator1")

public class Calculator1 extends HttpServlet {

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

        int num1 = Integer.parseInt(request.getParameter("num1"));

        int num2 = Integer.parseInt(request.getParameter("num2"));

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

        int result = 0;

        switch (operation) {

            case "add":

                result = num1 + num2;

                break;

            case "subtract":

                result = num1 - num2;

                break;

            case "multiply":

                result = num1 * num2;

                break;

            case "divide":

                result = num1 / num2;

                break;

        }

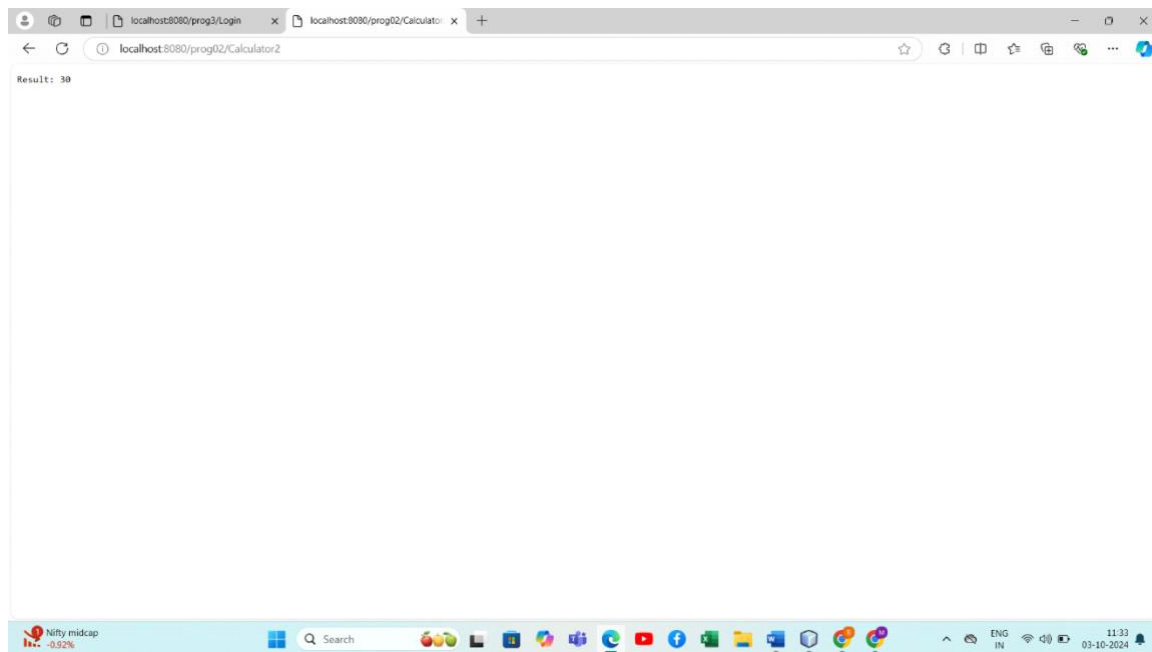
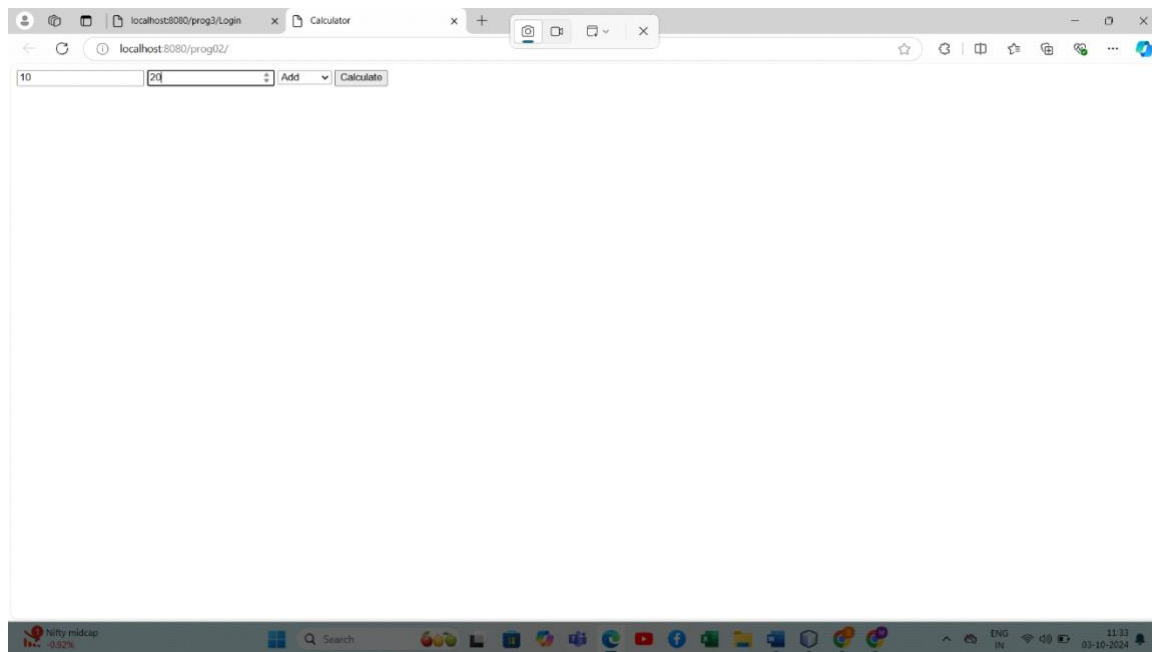
        response.getWriter().println("Result: " + result);

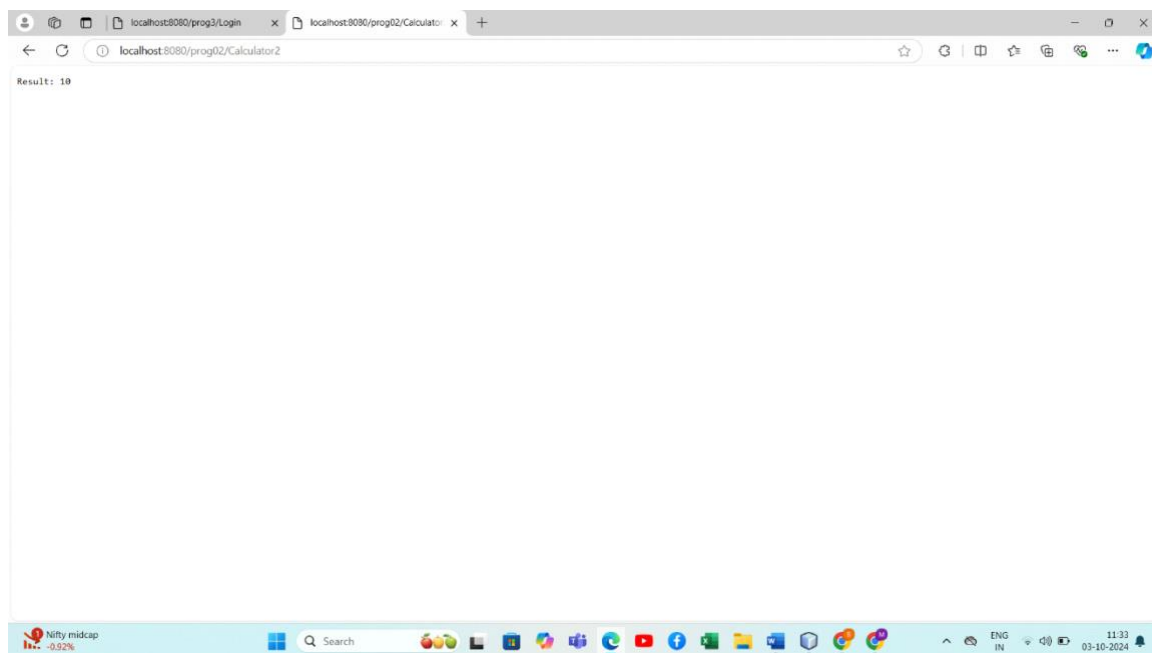
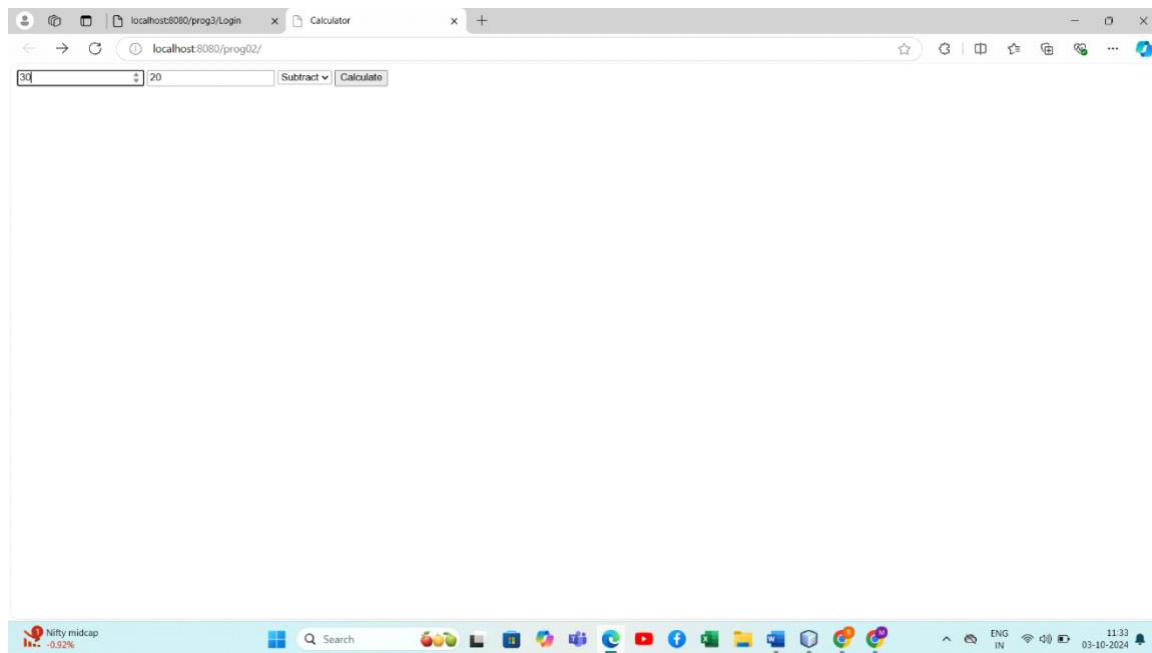
    }

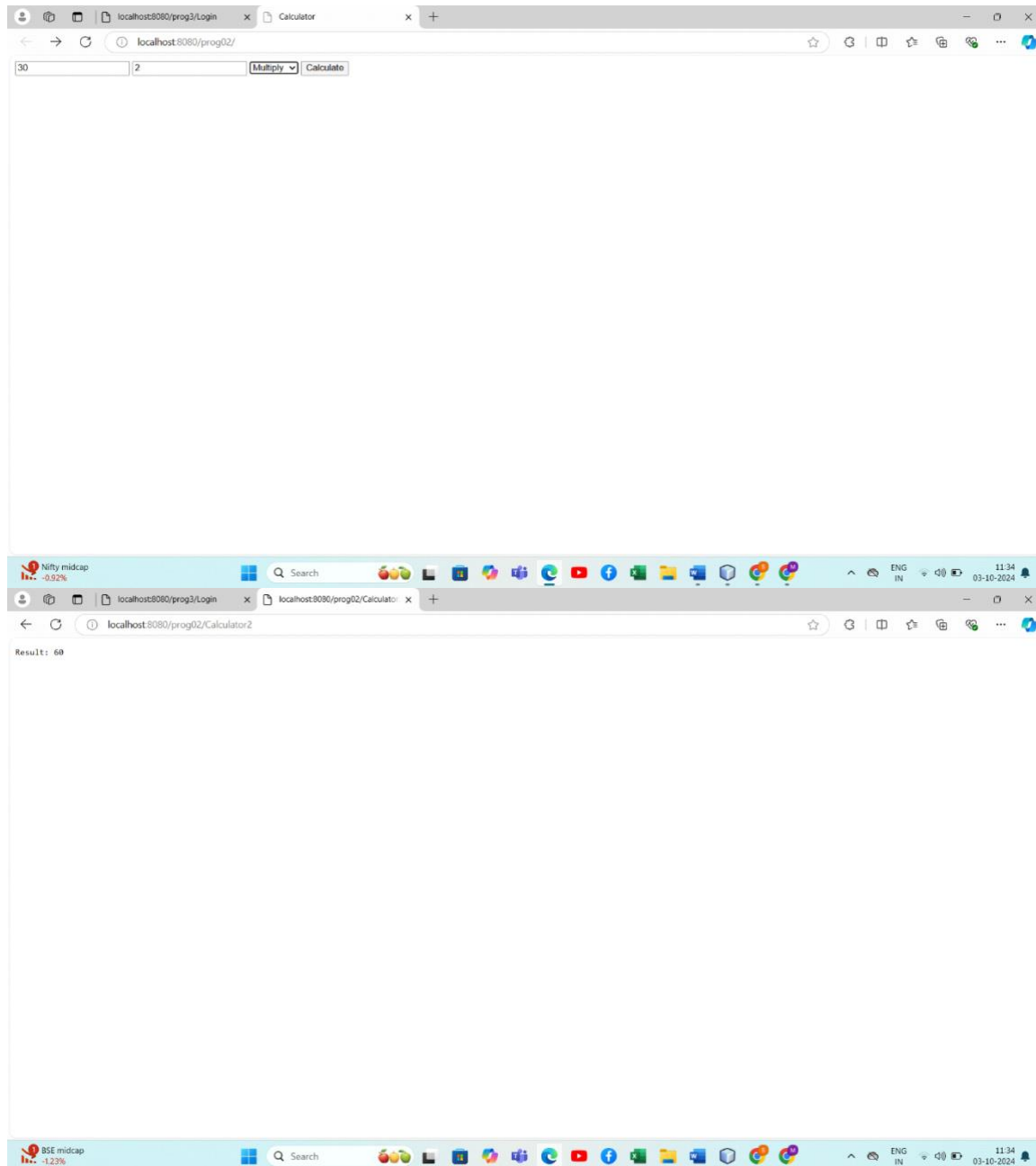
}

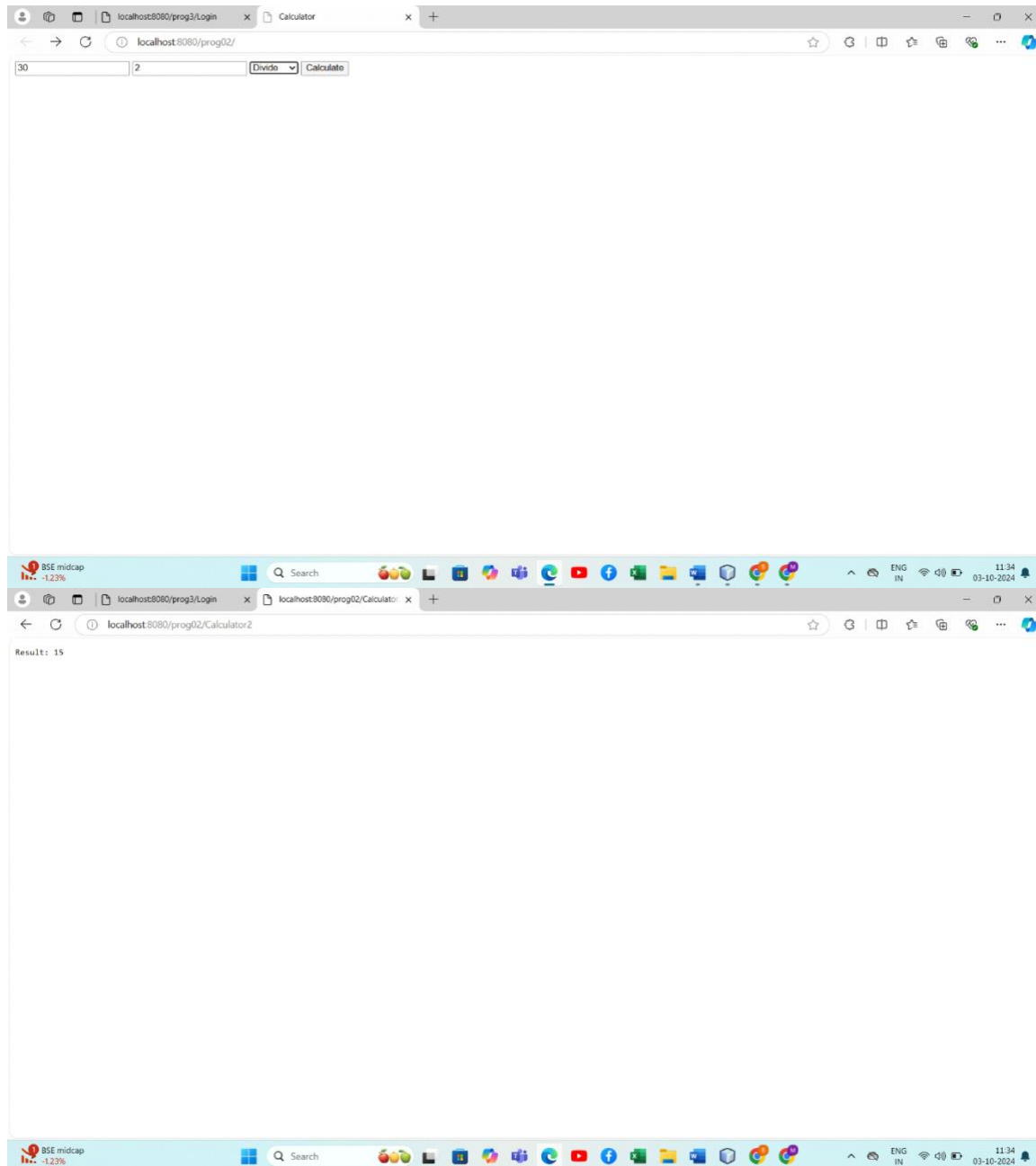
```

## OUTPUT









## RESULT

Thus, the program is executed successfully and the output is verified.

<b>Ex no: 3</b>	<b>LOGIN VALIDATION USING SERVLET</b>
<b>Date:</b>	

### **AIM**

To Create a servlet for a login page. If the username and password are correct then it says message "Hello" else a message "login failed".

### **ALGORITHM**

**Step 1:** Show the login form for username and password.

**Step 2:** User submits the form.

**Step 3:** Retrieve username and password from the form.

**Step 4:** Check if any field is empty If yes, display "Please enter both username and password."

**Step 5:** Check if the username is "divya" and the password is "123".If yes, display "Hello, divya! Welcome to Java Servlet!". If no, display "Login Failed. Please check your username and password."

**Step-6:** Successful login message displayed.

**Step-7:** End the process.

## PROGRAM

### 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>
    <form action="login" method="post" name="frm">
      Enter Username:
      <input name="uname" type="text"/>
      Enter Password:
      <input name="pass" type="password"/>
      <input type="submit" value="login"/>
      <input type="reset" value="reset"/>
    </form>
  </body>
</html>
```

### login.java

```
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;

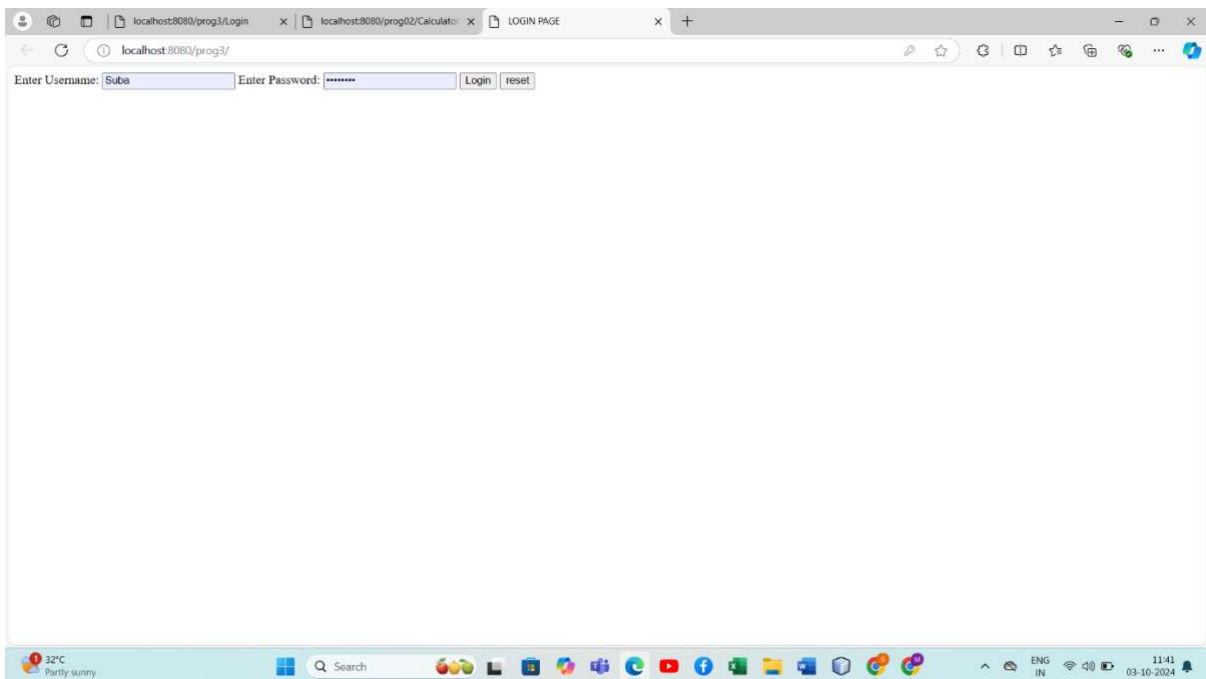
public class login extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

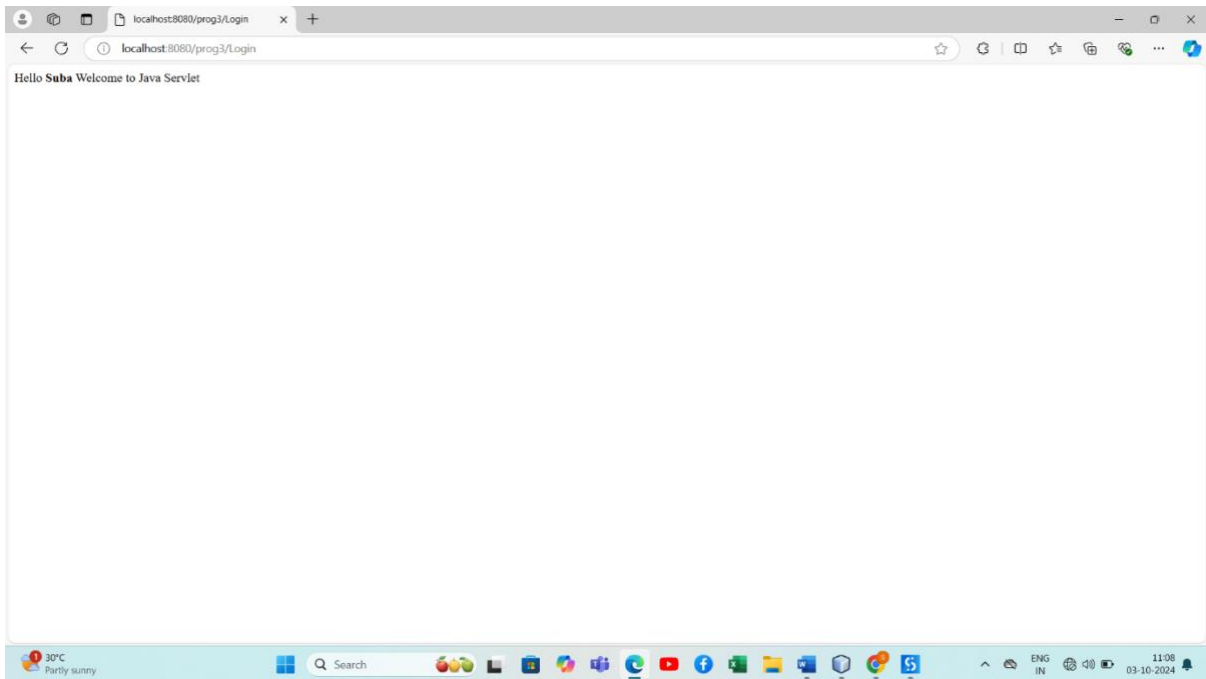


```
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
    String uname = request.getParameter("uname");
    String pass = request.getParameter("pass");

    if(uname.equals("divya")&& pass.equals("123"))
    {
        out.println("Hello <b>" +(uname) + "</b> Welcome to Java Servlet");
    }
    else
    {
        out.println("Login Failed");
    }
}
}
```

## OUTPUT





## RESULT

Thus, the program is executed successfully and the output is verified.

<b>Ex no:4</b>	<b>VALIDATION USING REQUEST DISPATCHER</b>
<b>Date:</b>	

### **AIM**

To create a servlet which will validate the password entered by the user using request dispatcher.

### **ALGORITHM**

**STEP 1:** Start the process.

**STEP 2:** Create a java web application.

**STEP 3:** In Index.html page design the page for username and password.

**STEP 4:** In LoginServlet page create two variables “uname” and “upass”.

**STEP 5:** Using requestDispatcher method move forward to Welcome page else Include index.html page with login failed message is included.

**STEP-6:** Stop the process.

## PROGRAM

### index.html

```
<html>
<body>
<form action="LoginServlet" method="post">
Name:<input type="text" name="userName"/><br/>
Password:<input type="password" name="userPass"/><br/>
<input type="submit" value="login"/>
</form>
</body>
</html>
```

### LoginServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        String name=request.getParameter("userName");
        String pwd=request.getParameter("userPass");
        if(pwd.equals("servlet"))
        {

            RequestDispatcher rd=request.getRequestDispatcher("WelcomeServlet");
            rd.forward(request, response);

        }

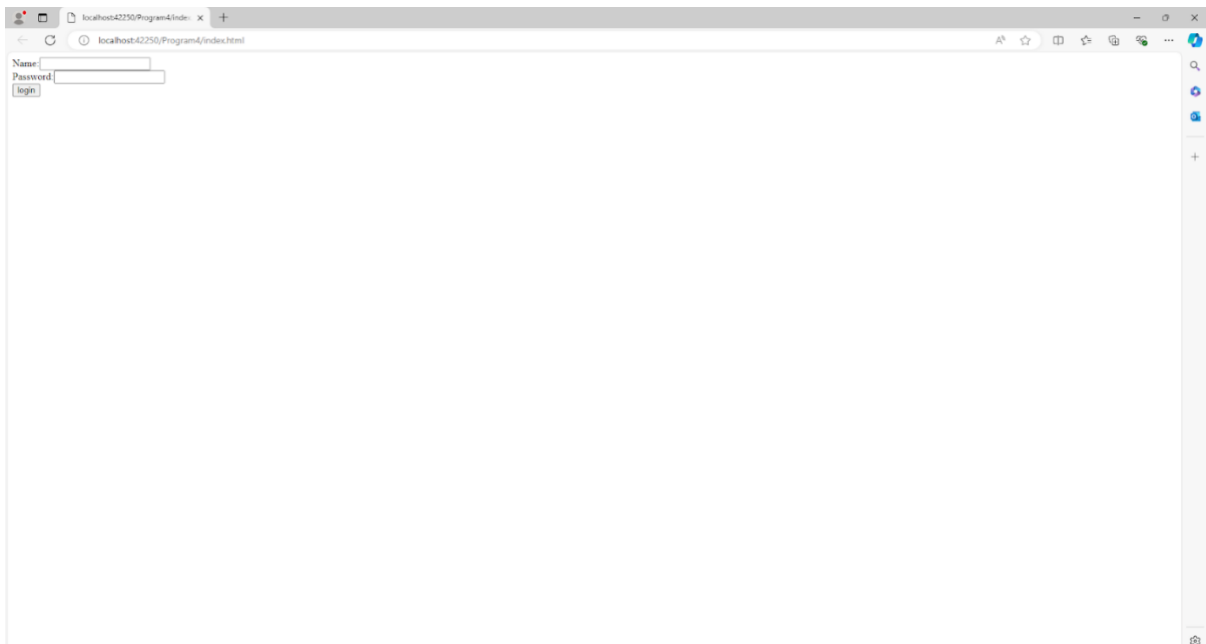
        else
        {
            out.print("Sorry UserName or Password Error!");
            RequestDispatcher rd=request.getRequestDispatcher("/index.html");
            rd.include(request, response);
        } } }
```

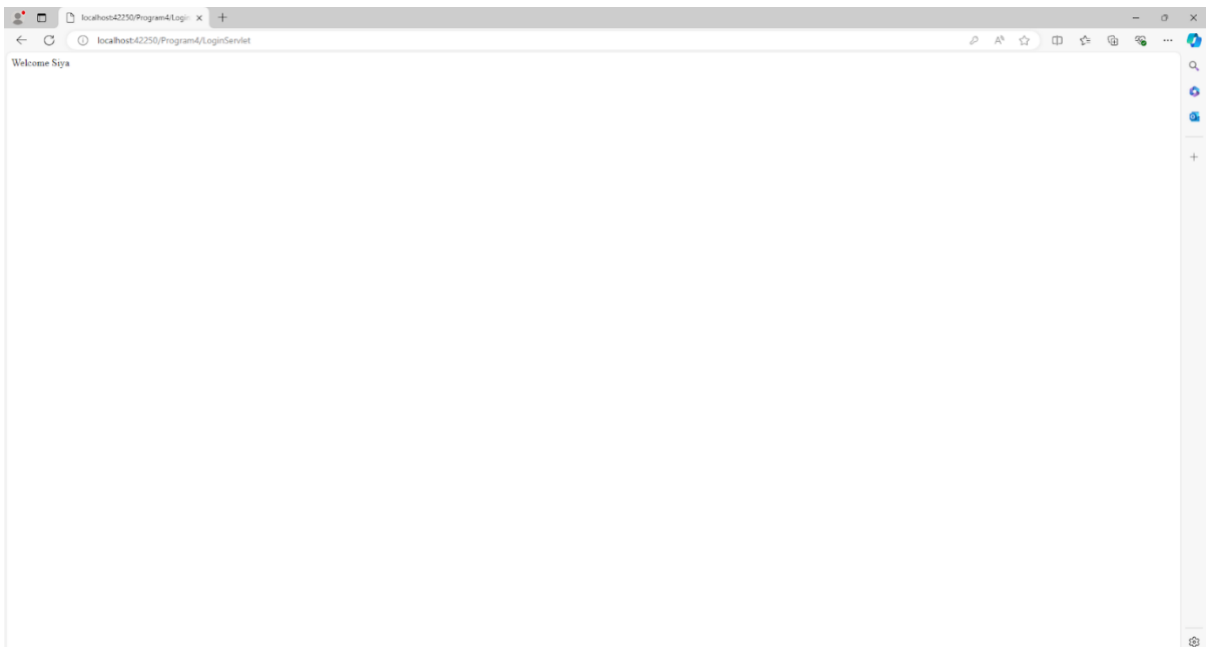
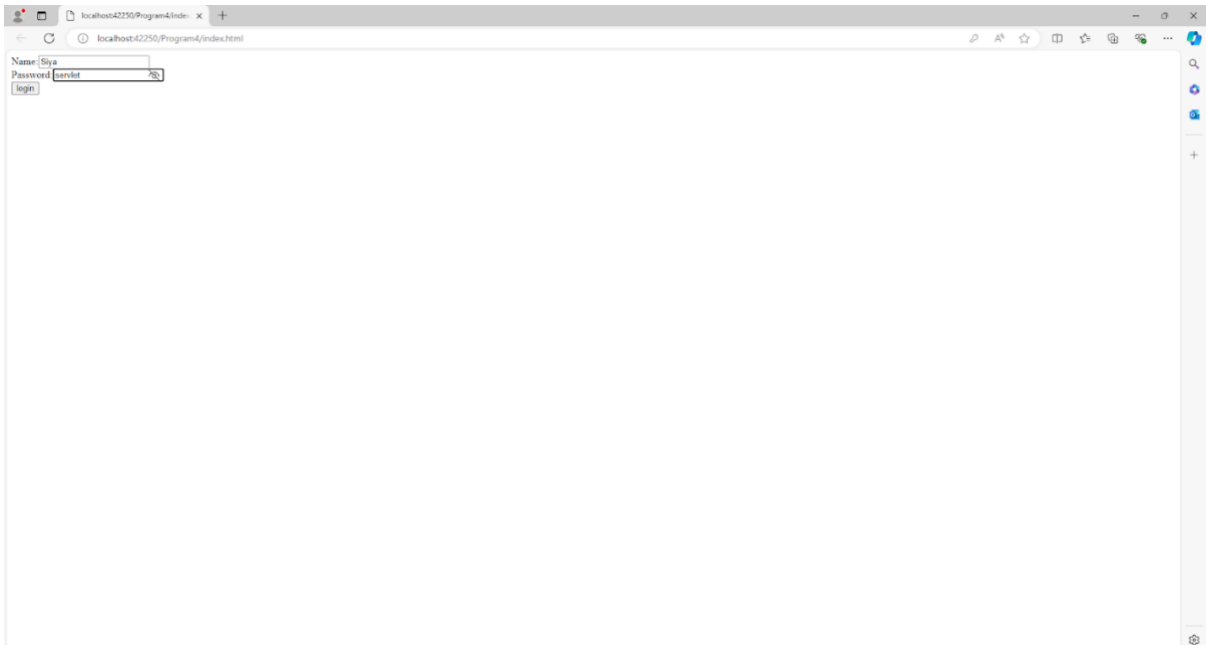
## WELCOMESERVLET.JAVA

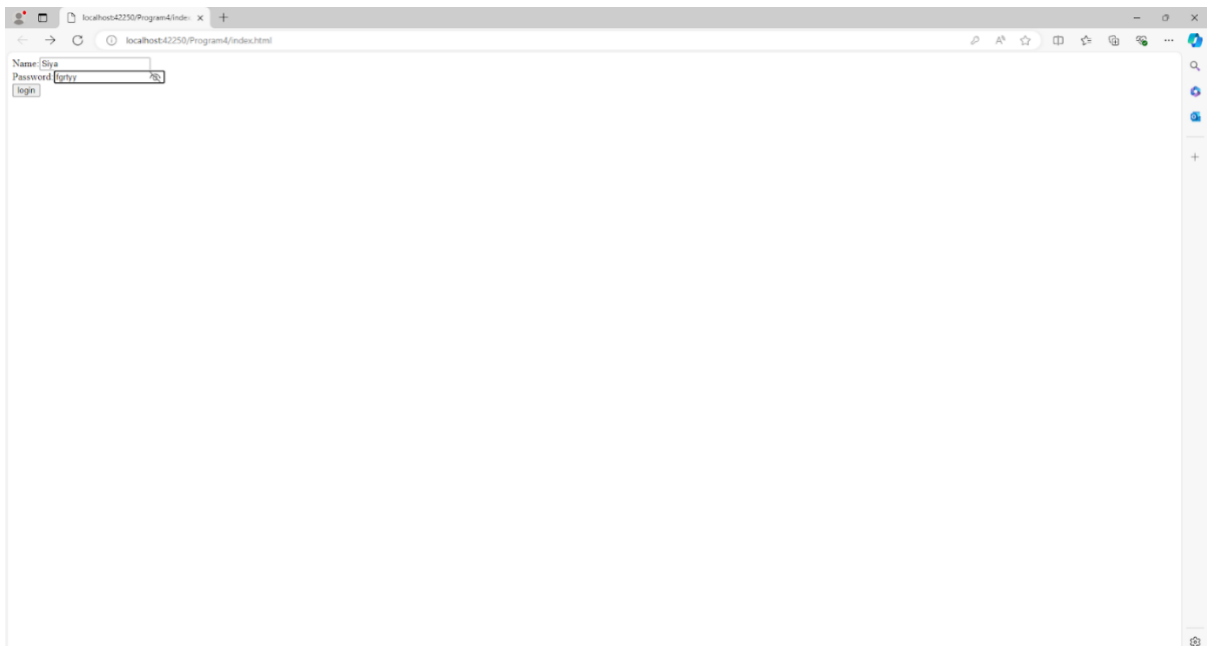
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class WelcomeServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String name=request.getParameter("userName");
        out.print("Welcome "+name);
    }
}
```

## OUTPUT







## RESULT

Thus, the program is executed successfully and output is verified.

<b>Ex no: 5</b>	<b>SESSION TRACKING USING SERVLET</b>
<b>Date:</b>	

## **AIM**

To create a servlet, demonstrating the use of session creation and destruction and also check whether the user has already visited the page.

## **ALGORITHM**

**STEP 1:** Open NetBeans and navigate to File -> New Project.

**STEP 2:** In the "New Project" window, choose Java Web from the categories and Web Application from the projects.

**STEP 3:** Right-click on the Source Packages folder in your project, go to New -> Servlet.

**STEP 4:** Name the servlet as SessionDemoServlet

**STEP 5:** Write the Servlet Code handling session creation, checking visit counts, and session destruction. The code demonstrates the session handling by checking if it's the first visit or not and displays the appropriate message

**STEP 6:** Run the Project. This will deploy your web application to the selected server and open the browser window pointing to the project URL.

**STEP 7:** Refresh the page multiple times to see the visit count increment each time, as the session persists and increments the visitCount

**STEP 8:** Click the link labeled Destroy Session on the servlet page.

**STEP 9:** This will trigger the session.invalidate() method, which will destroy the current session

**STEP 10:** After destroying the session, refresh the page to confirm that the session is reset and it again shows Welcome, first-time visitor!.

**STEP 11:** Stop the Process.



## PROGRAM

### index.html

```
<html>
<head>
<title>Welcome Page</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<form action="sesCount" method="get">
<h1>Welcome to Home Page</h1>
Enter your username:<input type="text" name="username">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

### sesCount.java

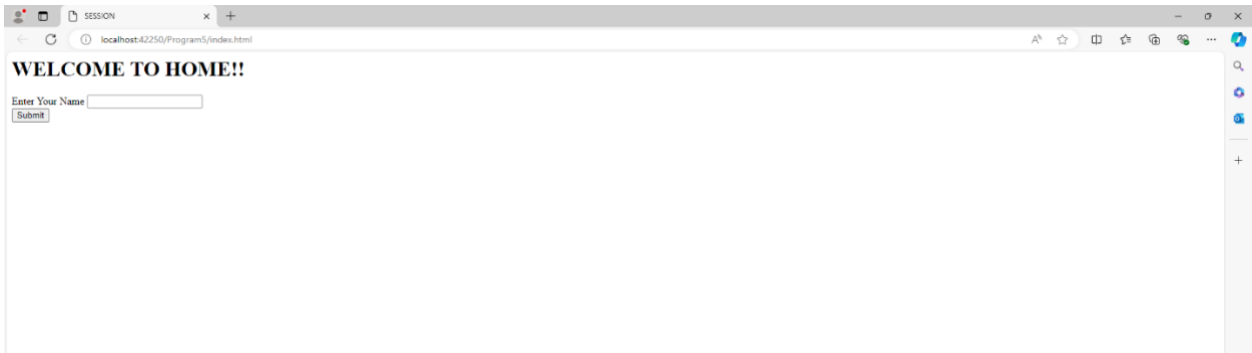
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

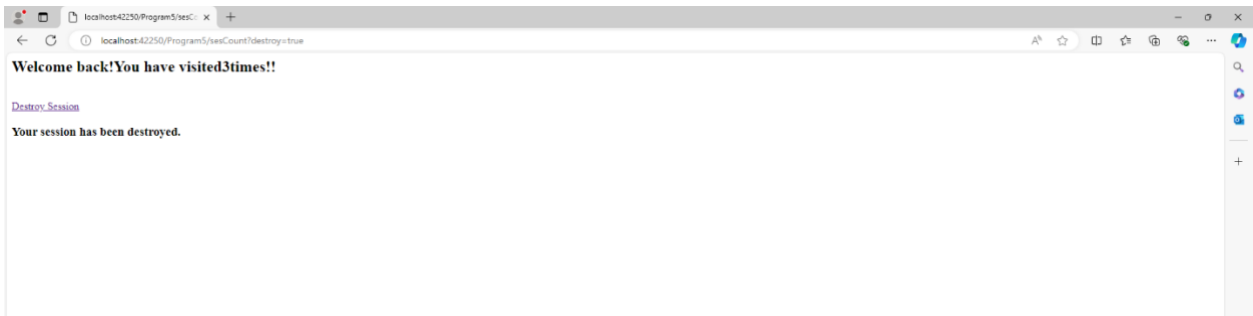
public class sesCount extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response
        throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter())
        {
            HttpSession session = request.getSession();
            String uname = request.getParameter("txtName");
            if(session.isNew())
            {
                out.println("WELCOME!!"+uname);
                out.println("<h2>Welcome First Time Visitor</h2>");
                session.setAttribute("visitCount",1);
            }
            else
            {
                Integer visitCount1= (Integer)session.getAttribute("visitCount");
                visitCount1++;
                session.setAttribute("visitCount",visitCount1);
                out.println("<h2> Welcome back!You have visited" +visitCount1+ "times!!</h2>");
            }
            out.println("<br><a href='sesCount?destroy=true'>Destroy Session</a>");
            String destroy = request.getParameter("destroy");
        }
    }
}
```

```
    if (destroy != null && destroy.equals("true"))
    {
        session.invalidate();
        out.println("<h3>Your session has been destroyed.</h3>");
    }
    out.close();
}
```

## OUTPUT





## RESULT

The program is successfully executed, and the output is verified.

<b>Ex no:6</b>	<b>EXCEPTIONS IN SPRING FOR INVALID BANK TRANSACTION</b>
<b>Date:</b>	

### **AIM**

To create a java spring program to raise exceptions for invalid bank transactions.

### **ALGORITHM**

**STEP 1:** Start the process

**STEP 2:** Open NetBeans and navigate to File -> New Project.

**STEP 3:** In the New Project-window, choose Java from the categories and Java Application from the projects.

**STEP 4:** Add spring JARs from the library by right clicking on the project->libraries->import spring framework 4.0.1 and add library

**STEP 5:** Create a package named Bankapp.exception and create exception classes for Insufficient Balance Exception and Invalid Account Exception

**STEP 6:** Create a class BankService under Bankapp.service to include various services provided by the bank.

**STEP 7:** Create a class Bank config,Bank account and Springbank app under the package Bankapp

**STEP 8:** Run the program and validate the process.

**STEP 9:** Stop the process.

## PROGRAM

### Custom Exceptions:

#### InsufficientBalanceException.java

```
package bankapp.exceptions;

    public class InsufficientBalanceException extends RuntimeException {
        public InsufficientBalanceException(String message) {
            super(message);
        }
    }
```

#### InvalidAccountException.java

```
package bankapp.exceptions;

    public class InvalidAccountException extends RuntimeException {
        public InvalidAccountException(String message) {
            super(message);
        }
    }
```

#### BankAccount.java

```
package bankapp;

public class BankAccount {
    private String accountNumber;
    private double balance;
    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }
    public String getAccountNumber() {
        return accountNumber;
    }
    public double getBalance() {
        return balance;
    }
    public void withdraw(double amount) {
        this.balance -= amount;
    }
}
```

#### BankService.Class

```
package bankapp.service;
import bankapp.BankAccount;
import bankapp.exceptions.InsufficientBalanceException;
import bankapp.exceptions.InvalidAccountException;
import org.springframework.stereotype.Service;
import java.util.HashMap;
import java.util.Map;
@Service
public class BankService {
    private Map<String, BankAccount> accounts = new HashMap<>();
    public BankService() {
        accounts.put("12345", new BankAccount("12345", 1000.00));
        accounts.put("67890", new BankAccount("67890", 500.00));
    }
    public void validateAccount(String accountNumber) {
```

```

        if (!accounts.containsKey(accountNumber)) {
            throw new InvalidAccountException("Invalid account number: " + accountNumber);
        }
    }
    public void withdraw(String accountNumber, double amount) {
        BankAccount account = accounts.get(accountNumber);
        if (account.getBalance() < amount) {
            throw new InsufficientBalanceException("Insufficient balance to withdraw " + amount);
        }
        account.withdraw(amount);
    }
    public double getBalance(String accountNumber) {
        return accounts.get(accountNumber).getBalance();
    }
}

```

### **SpringBankApp.java**

```

package bankapp;
import bankapp.service.BankService;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class SpringBankApp {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(BankConfig.class);
        BankService bankService = context.getBean(BankService.class);
        try {
            bankService.validateAccount("12345");
            bankService.withdraw("12345", 200);
            System.out.println("Balance after withdrawal: " + bankService.getBalance("12345"));
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        context.close();
    }
}

```

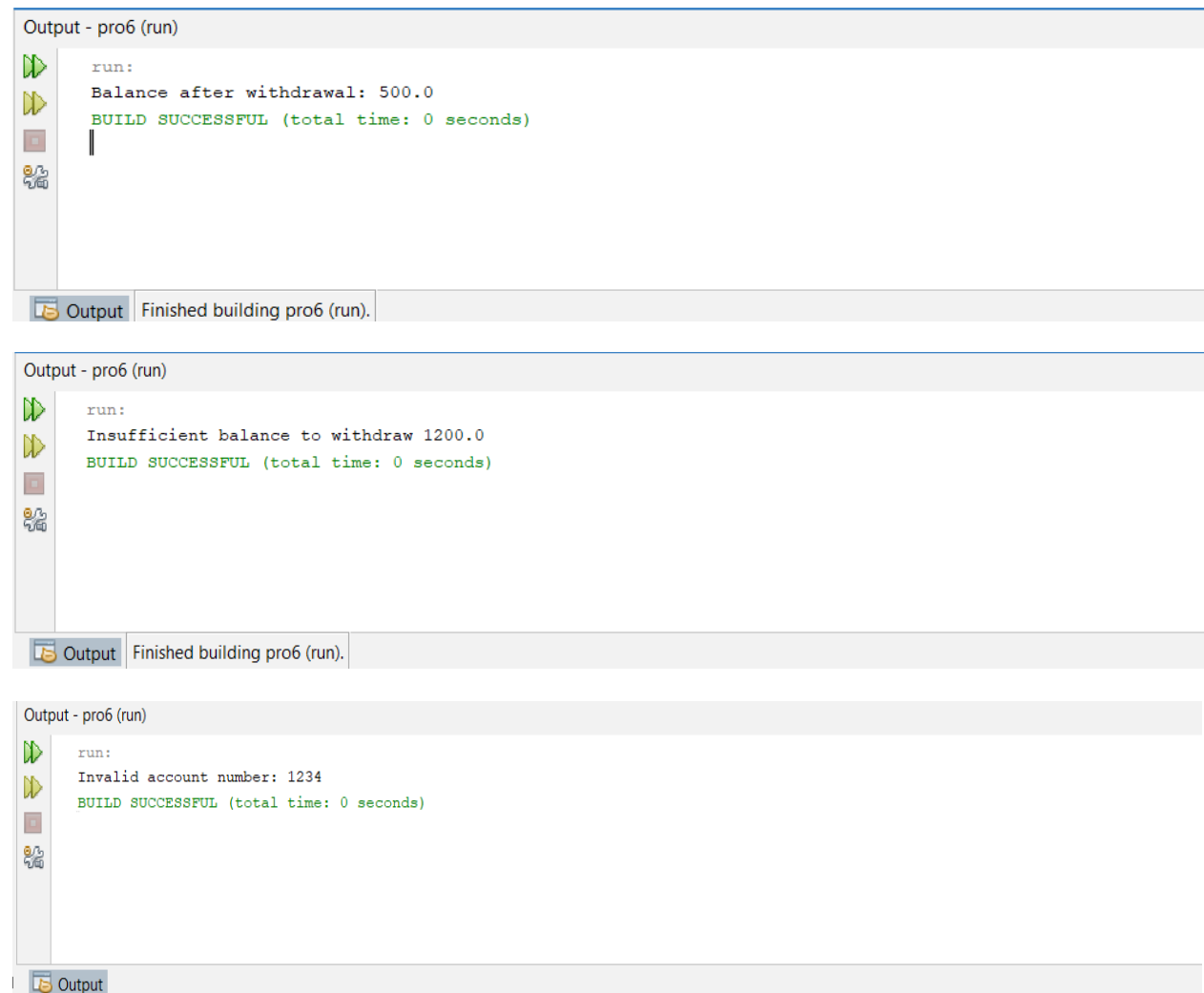
### **BankConfig.java**

```

package bankapp;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import bankapp.service.BankService;
@Configuration
public class BankConfig {
    @Bean
    public BankService bankService() {
        return new BankService();
    }
}

```

## OUTPUT



The image displays three sequential screenshots of an IDE's output window, titled "Output - pro6 (run)". Each screenshot shows the output of a program execution, with a status bar at the bottom indicating "Finished building pro6 (run)".

**First Screenshot:**

```
run:
Balance after withdrawal: 500.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Second Screenshot:**

```
run:
Insufficient balance to withdraw 1200.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Third Screenshot:**

```
run:
Invalid account number: 1234
BUILD SUCCESSFUL (total time: 0 seconds)
```

## RESULT

Thus, the program is executed successfully and the output is verified.

<b>Ex no:7</b>	<b>DATE INJECTION IN SPRING BEAN</b>
<b>Date:</b>	

### **AIM**

To write a program to inject date in to spring Bean property.

### **ALGORITHM**

**STEP 1:** Create a new project Mainapp.java right click to create myBean class and AppConfig class.

**STEP 2:** Create a class AppConfig to define configuration and manage Spring beans.

**STEP 3:** In AppConfig create two bean that returns the current date and time and MyBean bean that depends on the Date bean to set the currentdate.

**STEP 4:** MyBean object is created, Spring will automatically inject the Date object into MyBean by passing it as an argument.

**STEP 5:** Add library to spring framework Set up the Spring Application Context using AnnotationConfigApplicationContext to read the configuration from AppConfig.

**STEP 6:** Print the MyBean object, which internally calls its toString() method to display the current date.



## PROGRAM

### MyBean.java

```
package SpringDate;
import java.util.Date;
public class MyBean {
    private Date currentDate;
    // Getter and Setter
    public Date getCurrentDate() {
        return currentDate;
    }
    public void setCurrentDate(Date currentDate) {
        this.currentDate = currentDate;
    }

    @Override
    public String toString() {
        return "MyBean{" + "currentDate=" + currentDate + '}';
    }
}
```

### MainApp.java

```
package SpringDate;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);
        MyBean myBean = context.getBean(MyBean.class);

        System.out.println(myBean);
    }
}
```

### AppConfig.java

```
package SpringDate;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.Date;

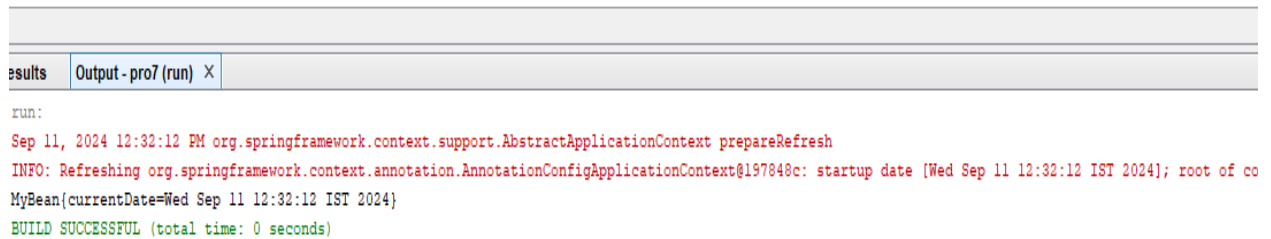
@Configuration
public class AppConfig {

    @Bean
    public Date currentDate() {
        return new Date(); // This returns the current date and time
    }

    @Bean
    public MyBean myBean(Date currentDate) {
```

```
MyBean myBean = new MyBean();  
myBean.setCurrentDate(currentDate);  
return myBean;  
}  
}
```

## OUTPUT



The screenshot shows an IDE output window with a tab labeled "Output - pro7 (run) X". The output text is as follows:

```
run:  
Sep 11, 2024 12:32:12 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh  
INFO: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@197848c: startup date [Wed Sep 11 12:32:12 IST 2024]; root of co  
MyBean{currentDate=Wed Sep 11 12:32:12 IST 2024}  
BUILD SUCCESSFUL (total time: 0 seconds)
```

## RESULT:

Thus, the program is executed successfully and the output is verified.

<b>Ex no:8</b>	<b>EMPLOYEE DETAILS USING HIBERNATE</b>
<b>Date:</b>	

### **AIM**

To Create an application for employee details using Hibernate.

### **ALGORITHM**

**STEP 1:** Open NetBeans and Set up the Project Environment.

**STEP 2:** Create the Hibernate Configuration File (hibernate.cfg.xml).

**STEP 3:** Create an Employee class annotated with @Entity, having fields: id, name, department, and salary.

**STEP 4:** Create HibernateUtil class to provide a SessionFactory object for managing Hibernate sessions.

**STEP 5:** Create the DAO Class for performing CRUD operations (EmployeeDAO.java).

**STEP 6:** Create the Main Class to Test the Application (Main.java).

**STEP 7:** Run the Main class, and it will interact with the database using Hibernate.

## PROGRAM

### hibernate.cfg.xml:

```
<?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
```

### Employee.java:

```
import javax.persistence.*;

@Entity
@Table(name = "employees")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name")
    private String name;

    @Column(name = "email")
    private String email;

    public Employee() {}

    public Employee(String name, String email) {
        this.name = name;
        this.email = email;
    }

    // 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 getEmail() {
    return email;
}

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

```

### **HibernateUtil.java:**

```

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

public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}

```

### **Main.java:**

```

import org.hibernate.Session;
import org.hibernate.Transaction;

public class Main {
    public static void main(String[] args) {
        // Create an employee
        Employee newEmployee = new Employee("John Doe", "john.doe@example.com");

        // Save the employee to the database
        Transaction transaction = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            session.save(newEmployee);
            transaction.commit();
            System.out.println("Employee saved: " + newEmployee.getId());
        } catch (Exception e) {
            if (transaction != null) transaction.rollback();
            e.printStackTrace();
        }

        // Fetch employee details
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {

```

```

    Employee employee = session.get(Employee.class, newEmployee.getId());
    System.out.println("Employee ID: " + employee.getId());
    System.out.println("Employee Name: " + employee.getName());
    System.out.println("Employee Email: " + employee.getEmail());
}
}
}

```

## OUTPUT

The screenshot shows the phpMyAdmin interface for a MySQL database. The 'register' table is selected, and the SQL query 'SELECT \* FROM register' is executed. The results show 4 rows of data. The table structure is as follows:

id	first_name	last_name	password	date_of_birth	email	phone_number	address	created_at
1	NITHYA	DEVI S	@Nithya	2017-08-21	snithya55521@gmail.com	2147483647	kutibollg	2024-06-10 13:19:57
2	akila	v	@akila	2006-08-21	akila@gmail.com	987654321	it,qjepr	2024-06-11 09:50:07
3	surya	v	surya@	2002-11-11	surya@gmail.com	987654321	nouyfv	2024-06-11 10:52:02
4	kaviya	b	@kaviya	2002-08-21	kaviya@gmail.com	2147483647	abcd	2024-06-12 10:20:49

## RESULT

Thus, the program executed and run successfully.

<b>Ex No:9</b>	<b>JSP APPLICATION WITH INTRINSIC OBJECTS</b>
<b>Date:</b>	

### **AIM**

To Develop a simple JSP application to display values obtained from the use of intrinsic objects of various types.

### **ALGORITHM**

**STEP 1:** Start the process.

**STEP 2:** Open NetBeans and create a java web application.

**STEP 3:** In index.html Design a page for name and submit button.

**STEP 4:** Write the coding in the Web page to enter the username and submit button.

**STEP 5:** Run the file.

**STEP 6:** Stop the process.

## PROGRAM

### index.html

```
<form action="example.jsp" method="get">
<label for="username">Enter your username:</label>
<input type="text" id="username" name="username">
<input type="submit" value="Submit">
</form>
```

### example.jsp

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>EXAMPLE JSP PAGE</title>
  </head>
  <body>
    <!-- Request Object: Reading request parameters --%>
    <%
      String username = request.getParameter("username");
      if (username == null || username.isEmpty()) {
        username = "Guest";
      }
    %>

    <!-- Session Object: Setting a session attribute --%>
    <%
      session.setAttribute("username", username);
    %>

    <!-- Application Object: Accessing application context --%>
    <%
      String appName = application.getInitParameter("applicationName");
    %>

    <!-- Out Object: Writing output to the client --%>
    <%
      out.println("<h1>Welcome, " + username + "!</h1>");
      out.println("<p>Application Name: " + appName + "</p>");
    %>

    <!-- Config Object: Accessing servlet configuration --%>
    <%
      ServletConfig config1 = getServletConfig();
      String servletName = config.getServletName();
    %>
    <p>Servlet Name: <%= servletName %></p>

    <!-- PageContext Object: Setting and getting page attributes --%>
    <%
      pageContext.setAttribute("greeting", "Hello from PageContext!");
      String greeting = (String) pageContext.getAttribute("greeting");
    %>
```



```
<p><%= greeting %></p>
```

```
<%-- Page Object: Referring to the current instance of the JSP --%>
```

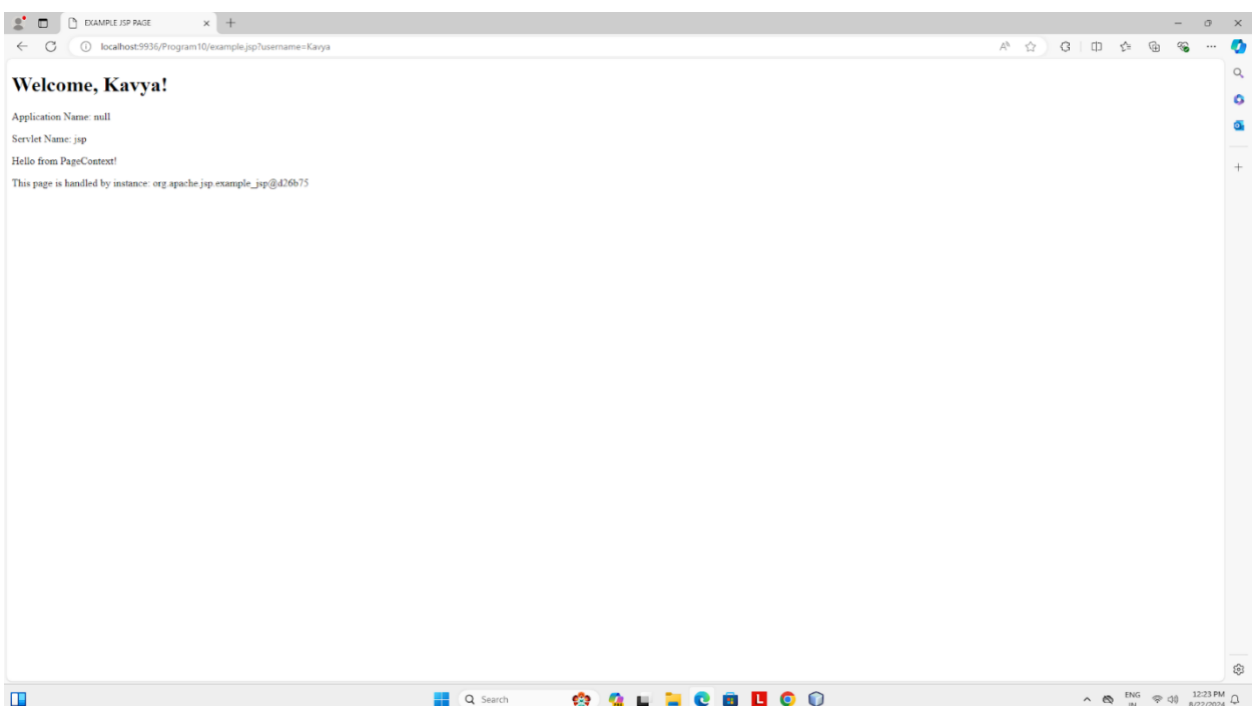
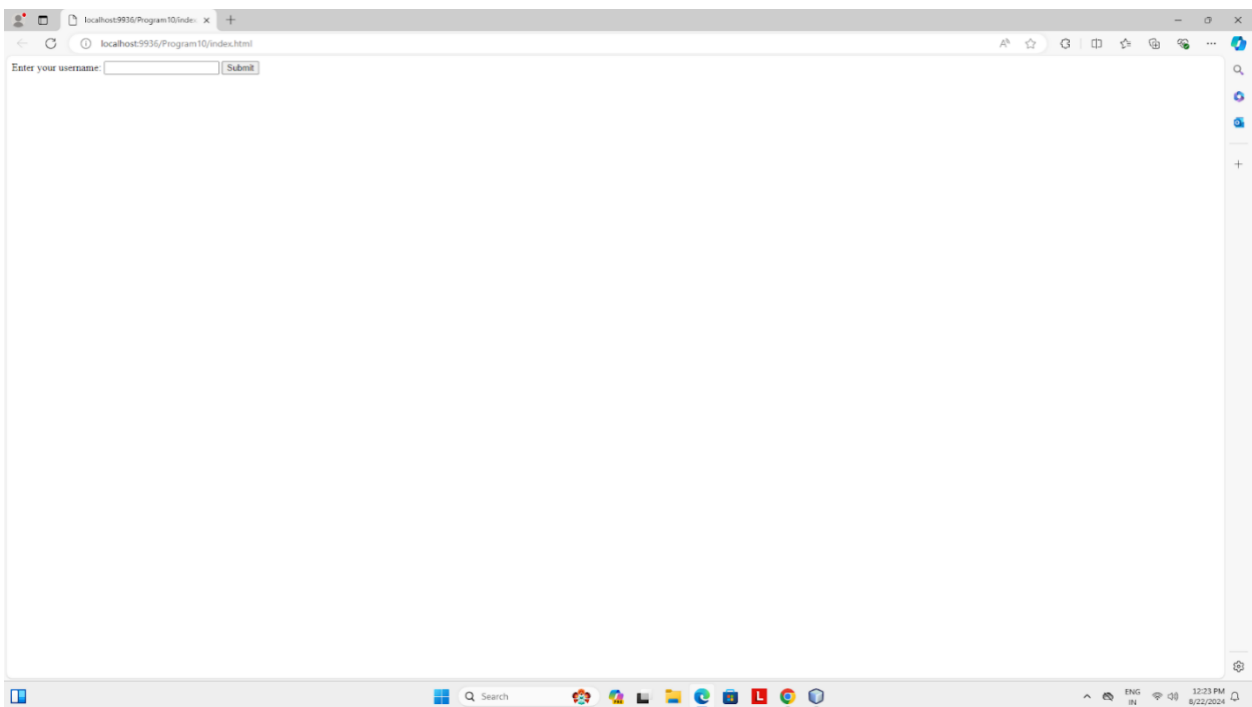
```
<%
```

```
    out.println("<p>This page is handled by instance: " + page.toString() + "</p>");  
%>
```

```
</body>
```

```
</html>
```

## OUTPUT



**RESULT:**

Thus, the program is executed successfully and output is verified.

<b>Ex no:10</b>	<b>PASSING VALUES WITH VALIDATION IN JSP</b>
<b>Date:</b>	

**AIM**

To Develop a simple JSP application to pass values from one page to another with validation.

**ALGORITHM**

**STEP 1:** Start the process

**STEP 2:** Create new web application.

**STEP 3:** In index.html create input fields for name ,age, hobby, email, gender.

**STEP 4:** Create a new JSP page named submit.jsp.

**STEP 5:** In JSP page using request.getParameter get values of the attributes.

**STEP 6:** Check whether the fields are non-empty and also validate.

**STEP 7:** Display the validated results.

**STEP 8:** Stop the process.

## PROGRAM

### Index.html

```
<html>
<head>
<title>Simple JSP Form</title>
</head>
<body>
<h2>Enter your details</h2>
<form action="submit.jsp" method="post">
Name: <input type="text" name="name" required /><br/><br/>
Age: <input type="text" name="age" required /><br/><br/>
Hobbies: <br/>
<input type="checkbox" name="hobbies" value="Reading" /> Reading<br/>
<input type="checkbox" name="hobbies" value="Traveling" /> Traveling<br/>
<input type="checkbox" name="hobbies" value="Gaming" /> Gaming<br/><br/>
Email: <input type="email" name="email" required /><br/><br/>
Gender: <br/>
<input type="radio" name="gender" value="Male" required /> Male<br/>
<input type="radio" name="gender" value="Female" required /> Female<br/><br/>
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

### submit.jsp

```
<html>
<head>
<title>Submitted Details</title>
</head>
<body>
<h2>Submitted Details</h2>
<%
String name = request.getParameter("name");
String age = request.getParameter("age");
String[] hobbies = request.getParameterValues("hobbies");
String email = request.getParameter("email");
String gender = request.getParameter("gender");
boolean valid = true;
StringBuilder validationMessages = new StringBuilder();
if (name == null || name.isEmpty()) {
valid = false;
validationMessages.append("Name is required.<br/>");
}
if (age == null || age.isEmpty()) {
valid = false;
validationMessages.append("Age is required.<br/>");
} else
{
try {
int ageInt = Integer.parseInt(age);
if (ageInt <= 0) {
valid = false;
```

```

validationMessages.append("Age must be a positive number.<br/>");
}
} catch (NumberFormatException e) {
valid = false;
validationMessages.append("Age must be a number.<br/>");
}
}
if (email == null || email.isEmpty()) {
valid = false;
validationMessages.append("Email is required.<br/>");
}
if (gender == null || gender.isEmpty()) {
valid = false;
validationMessages.append("Gender is required.<br/>");
}
if (valid)
{
%>
<p>Name: <%= name %></p>
<p>Age: <%= age %></p>
<p>Hobbies:
<%
if (hobbies != null) {
for (String hobby : hobbies) {
out.print(hobby + " ");
}
}
else
{
out.print("None");
}
%>
</p>
<p>Email: <%= email %></p>
<p>Gender: <%= gender %></p>
<%
} else {
%>
<h3>Validation Errors</h3>
<p><%= validationMessages.toString() %></p>
<a href="index.jsp">Go back to the form</a>
<%
}
%>
</body>
</html>

```

## OUTPUT

The first screenshot shows a web browser window with the title 'Simple JSP Form'. The address bar shows 'localhost:9936/Program11/index.html'. The form is titled 'Enter your details' and contains the following fields:

- Name:
- Age:
- Hobbies: ☒ Reading, ☒ Traveling, ☐ Gaming
- Email:
- Gender: ☐ Male, ☒ Female
- 

The second screenshot shows the same browser window with the title 'Submitted Details'. The address bar shows 'localhost:9936/Program11/submit.jsp'. The form displays the submitted details:

- Name: Harsha
- Age: 22
- Hobbies: Reading Traveling
- Email: 23mcs014@psgkew.ac.in
- Gender: Female

## RESULT:

Thus, the program is executed successfully and the output is verified.

<b>Ex no: 11</b>	<b>EXPRESSION LANGUAGE USING JSP</b>
<b>Date:</b>	

### **AIM**

To Create a JSP page to demonstrate the use of Expression Language

### **ALGORITHM**

**STEP 1:** Start the process.

**STEP 2:** Create java web application.

**STEP 3:** In index.html create input fields for username,age and salary.

**STEP 4:** Create JSP page submit.jsp and using getParameter get the values.

**STEP 5:** Perform arithmetic operations, check logic condition and string concatenation.

**STEP 6:** Stop the process.

## PROGRAM

### 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>
    <form action="new.jsp" method="get">
      <label for="username">Enter your username:</label>
      <input type="text" id="username" name="username">
      <label for="age">Enter your Age:</label>
      <input type="number" id="age" name="age">
      <label for="salary">Enter your Salary:</label>
      <input type="number" id="salary" name="salary" >
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

### new.jsp

```
<% @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>Processed User Information</h1>

    <!-- Retrieve and display parameters from the request -->
    <%
      String username = request.getParameter("username");
      String ageParam = request.getParameter("age");
      String salaryParam = request.getParameter("salary");

      int age = 0;
      double salary = 0.0;

      // Convert parameters to appropriate types
      try {
        age = Integer.parseInt(ageParam);
      } catch (NumberFormatException e) {
        // Handle invalid number format for age
      }

      try {
        salary = Double.parseDouble(salaryParam);
      } catch (NumberFormatException e) {
        // Handle invalid number format for salary
      }
    %>
```



```

    }
    %>

<!-- Display values -->
<p>Username: <%= username %></p>
<p>Age: <%= age %></p>
<p>Salary: <%= salary %></p>

<!-- Performing arithmetic operations -->
<h2>Arithmetic Operations:</h2>
<p>Age in 5 years: <%= age + 5 %></p>
<p>Half of Salary: <%= salary / 2 %></p>
<p>Salary with 10% bonus: <%= salary * 1.10 %></p>

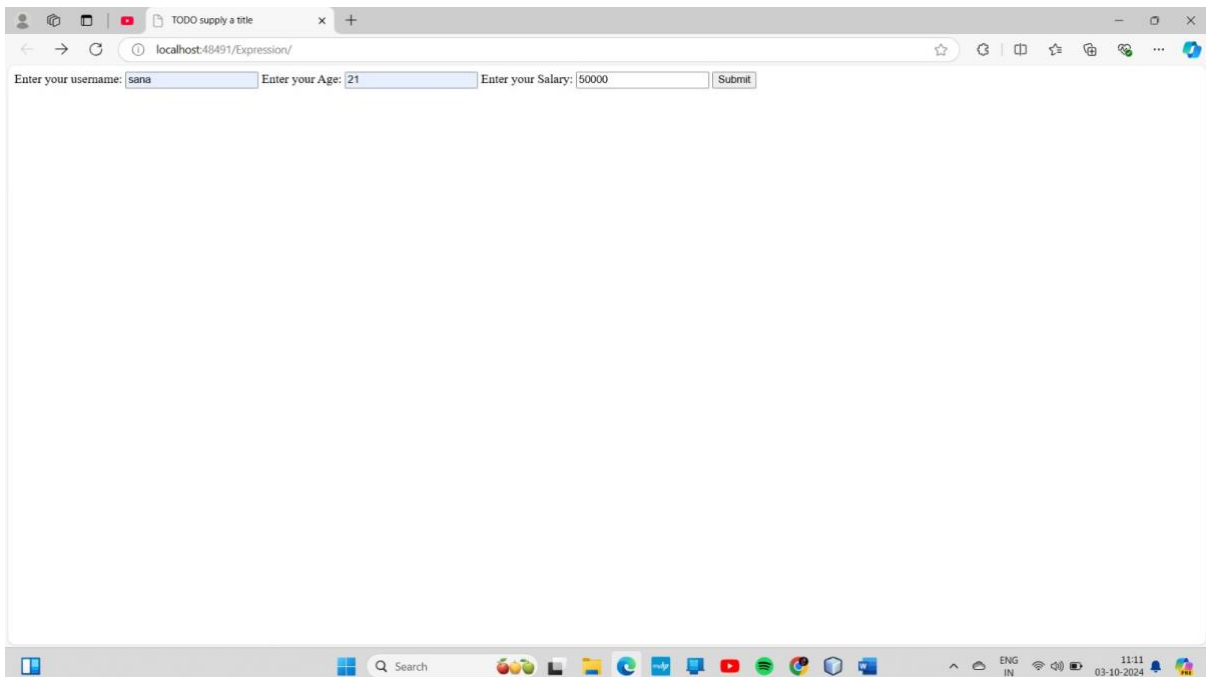
<!-- Conditional logic -->
<h2>Conditional Logic:</h2>
<p>Is user an adult? <%= age >= 18 ? "Yes" : "No" %></p>
<p>Salary status: <%= salary > 40000 ? "High" : "Low" %></p>

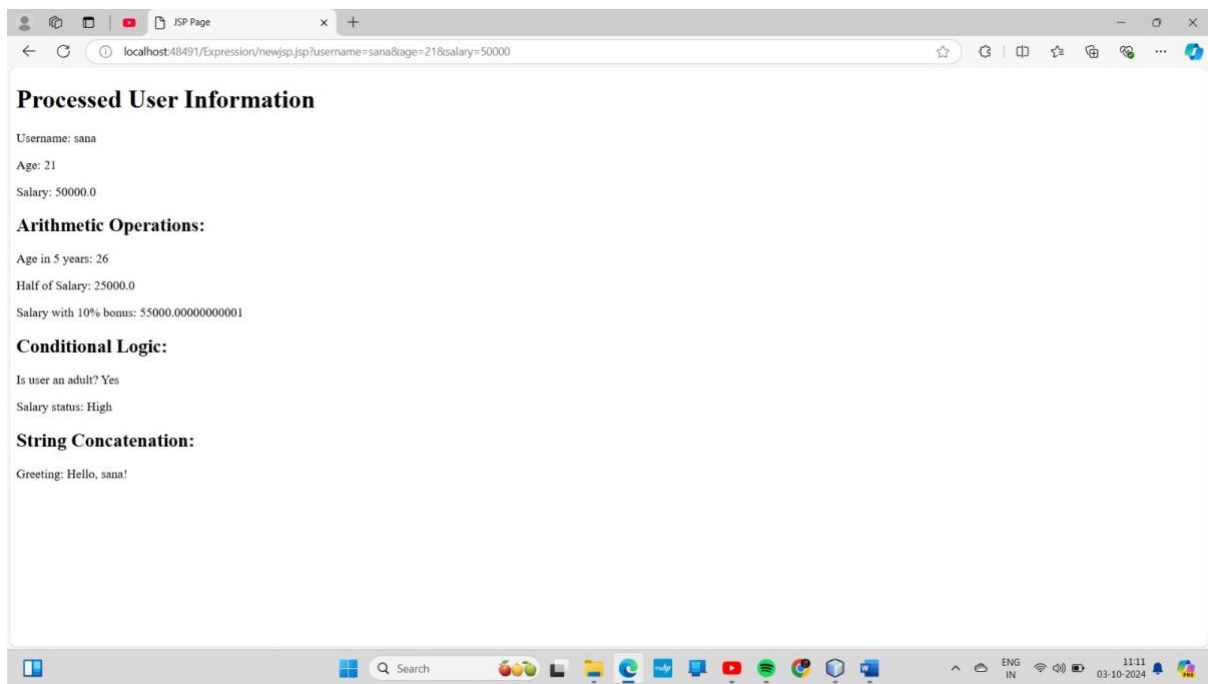
<!-- String concatenation -->
<h2>String Concatenation:</h2>
<p>Greeting: <%= "Hello, " + username + "!" %></p>

</body>
</html>

```

## OUTPUT





## RESULT

Thus, the program is executed successfully and the output is verified.

<b>Ex no:12</b>	<b>LOGIN FORM VALIDATION USING JAVABEANS</b>
<b>Date:</b>	

### **AIM**

To create a login form validation system using Javabeans, ensuring secure and efficient user authentication.

### **ALGORITHM**

**STEP 1:** Start the Process.

**STEP 2:** Create a Javabean and define private field for username and password.

**STEP 3:** Add Validation logic to check if the username and password are non empty.

**STEP 4:** Create JSP form for user input.

**STEP 5:** Include fields for username and password with a POST request to a servlet.

**STEP 6:** Instantiate userbean , set values and call the validate() method.

**STEP 7:** Redirect to success or failure pages based on validation result.

**STEP 8:** Design success and failure (login.jsp) pages to display appropriate message based on validation output.

**STEP 9:** Stop the process.

## PROGRAM

### index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Welcome to Login Validation Demo</title>
</head>
<body>
  <h1>Welcome to the Login Validation Demo</h1>

  <p>Click the link below to access the login form:</p>

  <!-- Link to the login.jsp page -->
  <a href="login.jsp">Go to Login Form</a>
</-->
```

### login.jsp

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
</head>
<body>
  <h2>Login Form</h2>

  <!-- Form submission will go to the same page (login.jsp) -->
  <form method="post" action="login.jsp">
    Username: <input type="text" name="username" /><br><br>
    Password: <input type="password" name="password" /><br><br>
    <input type="submit" value="Login" />
  </form>

  <%
    // Get the username and password from the form
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    if (username != null && password != null) {
      // Create an instance of LoginBean and set the form values
      LoginBean loginBean = new LoginBean();
      loginBean.setUsername(username);
      loginBean.setPassword(password);

      // Validate the login
      if (loginBean.validate()) {
        out.println("<p>Login Successful! Welcome, " + username + ".</p>");
      } else {
        out.println("<p style='color:red;'>Invalid username or password!</p>");
      }
    }
  %>
</body>
</html>
```

### LoginBean.java

```
package com.login;

public class LoginBean {
    private String username;
    private String password;

    // Getters and setters
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    // A method to validate username and password
    public boolean validate() {
        // Simple validation (hardcoded for demonstration purposes)
        if (username.equals("priyanka") && password.equals("priya123")) {
            return true;
        }
        return false;
    }
}
```

### OUTPUT

#### Login Form

Username:

Password:

Login

Login Successful! Welcome, priyanka.

## Login Form

Username:

Password:

Login

Invalid username or password!



## RESULT

Thus, the program executed successfully and output is verified.

<b>Ex no:13</b>	<b>FILE UPLOADING USING JSP</b>
<b>Date:</b>	

### **AIM**

To write a JSP program to select a file and then upload the file to the server.

### **ALGORITHM**

**Step 1:** Start the process.

**Step 2:** Create a new project in NetBeans.

**Step 3:** Click on File New project.

**Step 4:** Design the form using html code, with browser and submit button.

**Step 5:** Select the file by clicking on the browser button and click on submit button to upload the file.

**Step 6:** Add fileupload,io commons jar file to the library.

**Step 7:** First we have to import com.reilly.servlet.MultipartRequest for uploading file in drive, we are using multipartRequest.

**Step 8:** MultipartRequest is used for uploading file in drive up to 1mb.

**Step 9:** Then, we have to create an object of MultipartRequest and specify parameter with drive name in which we need to upload file.

**Step 10:** At last, add jar file cos.jar. It is used for uploading file.

**Step 11:** Run the Program and stop the process.

## PROGRAM

### index.html

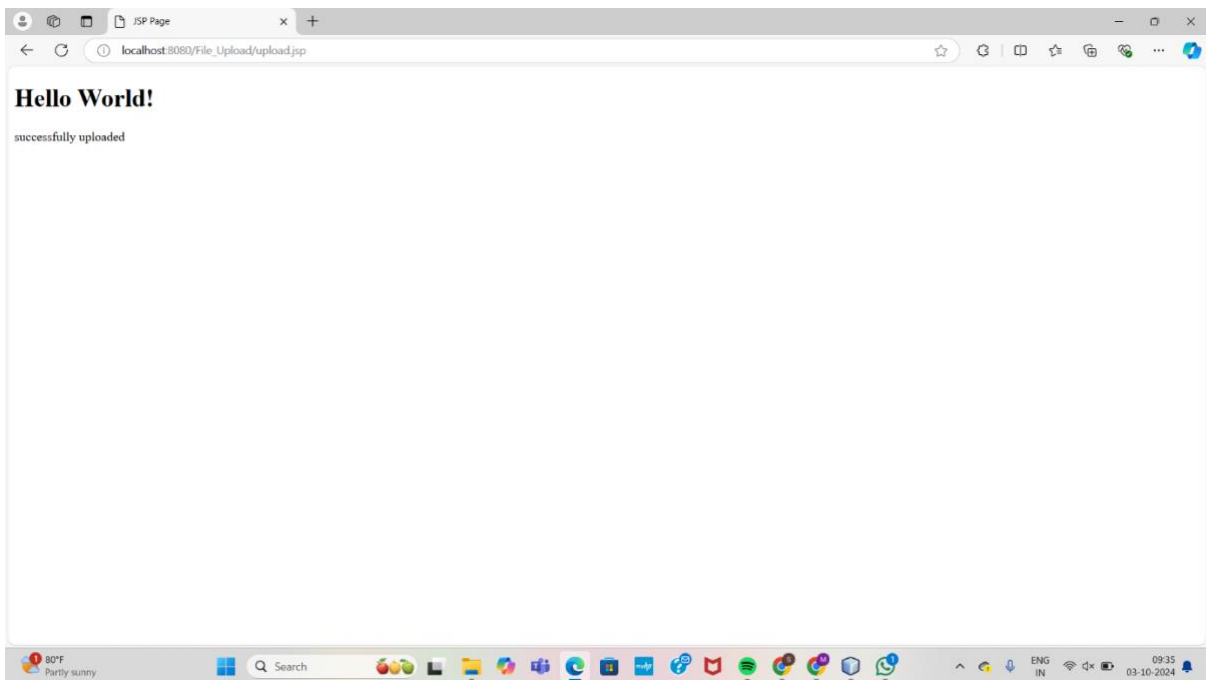
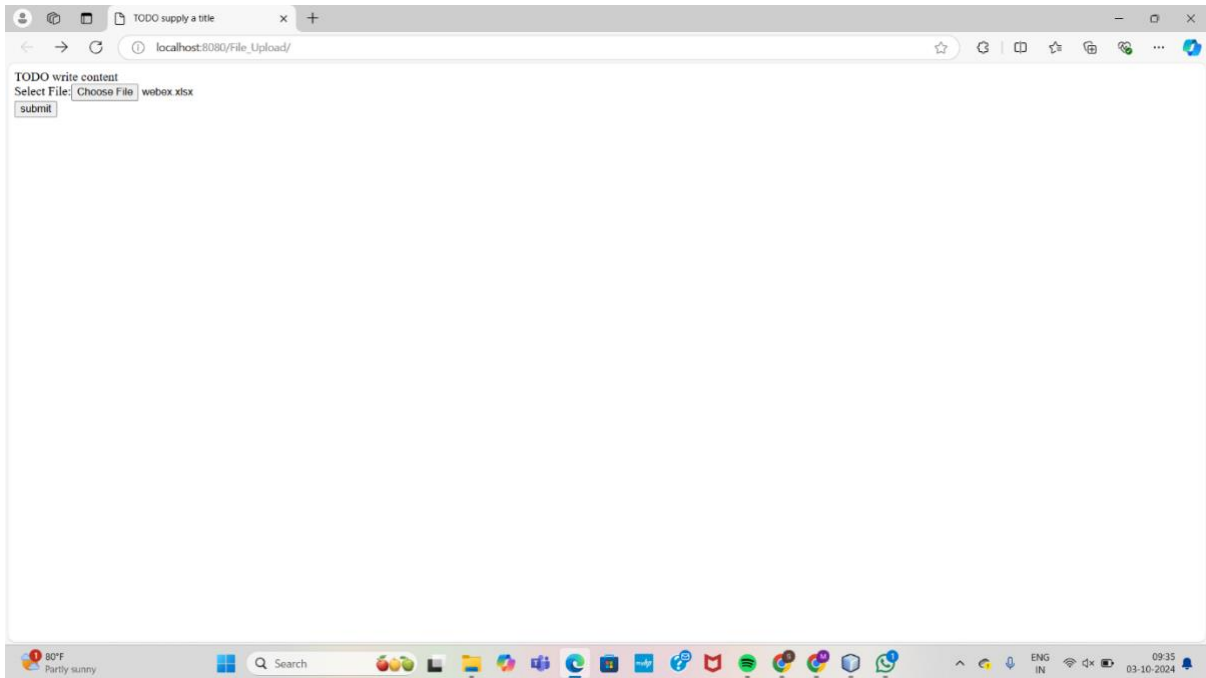
```
<!DOCTYPE html>
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<div>TODO write content</div>
<form action="upload.jsp" method="post" enctype="multipart/form-data">
Select File:<input type="file" name="fname"/><br/>
<input type="submit" value="submit"/>
</form> </body>
</html>
```

### upload.jsp

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<% @ page import="com.oreilly.servlet.MultipartRequest" %>
<!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>
<% MultipartRequest m = new MultipartRequest(request, "d:/");
out.print("successfully uploaded"); %>
</body>
</html>
```



## OUTPUT



## RESULT

Thus, the program is executed successfully and the output is verified.

<b>Ex no:14</b>	<b>ELECTRICITY BILL GENERATION</b>
<b>Date:</b>	

## **AIM**

To develop an application using JSP to generate electricity bill.

## **ALGORITHM**

**Step 1:** Start the Process.

**Step 2:** Open Netbeans file new project java web web application.

**Step 3:** In Netbeans create two JSP pages, Index.jsp and bill.jsp

**Step 4:** Design part is done in Index.jsp

**Step 5:** Designing page to have the customer name, connection number, date, zone name, previous meter reading, current meter reading.

**Step 6:** In Bill.jsp check whether the consumed unit, If it exceeds 100 then generate the bill amount.

**Step 7:** Run the program and display the output.

**Step 8:** Stop the process.

## PROGRAM

### ebbill. html

```
<html>
<head>
<title></title>
</head>
<body>
<form action="eb.jsp">
<h2><div style="text-align:center">TAMILNADU ELECTRICITY BOARD</div></h2>
<table BORDER="1" >
<CAPTION><EM> ELECTRICITY BILL</EM></CAPTION>
<tr rowspan="2">
<th> Consumer Name:</th>
<td><input type="text" name="conname"></td>
</tr>
<tr>
<th> Connection no:</th>
<td><input type="text" name="conno"></td>
</tr>
<tr>
<th> Date:</th>
<td><input type="text" name="condate"></td>
</tr>
<tr>
<th> Zone Name:</th>
<td><input type="text" name="conzn"></td>
</tr>
<tr>
<th>Previous meter reading :</th>
<td><input type="text" name="conpmr"></td>
</tr>
<tr>
<th> Current meter reading:</th>
<td><input type="text" name="concmr"></td>
</tr>
</table>
<br><br><br>
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

### eb.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
try{
String cname=request.getParameter("conname");
String cno=request.getParameter("conno");
String cndate=request.getParameter("condate");
```

```

String cnzn=request.getParameter("conzn");
String cnpmr=request.getParameter("conpmr");
String cncmr=request.getParameter("concmr");
int cnpmr1=Integer.parseInt(cnpmr);
int cncmr1=Integer.parseInt(cncmr);
int units=cncmr1-cnpmr1;
int runits=units-100;
double tamt=0.0;
if (runits > 100){
if (runits >= 200){
tamt=tamt+(100.0*2.00);
}else{
tamt=tamt+(runits*2.00);
}}
if (runits > 201){
if (runits > 400){
tamt=tamt+(200.0*4.00);
}else{
runits=runits-100;
tamt=tamt+(runits*4.00);
}
}
if (runits > 400){
if(runits > 600){
tamt=tamt+(runits*6.00);
} else{
runits=runits-400;
tamt=tamt+(runits*6.00);
}
}
%>
<form target="_blank">
<h2><div style="text-align:center">TAMILNADUELECTRICITY BOARD</div></h2>
<table BORDER="1" >
<CAPTION><EM> ELECTRICITY BILL</EM></CAPTION>
<tr rowspan="2">
<th> Consumer Name:</th>
<td><input type="text" name="conname" value=<%=cname%>></td>
</tr>
<tr>
<th> Connection no:</th>
<td><input type="text" name="conno" value=<%=cno%>></td>
</tr>
<tr>
<th> Date:</th>
<td><input type="text" name="condate" value=<%=cndate%>></td>
</tr>
<tr>
<th> Zone Name:</th>
<td><input type="text" name="conzn" value=<%=cnzn%>></td>
</tr>
<tr>
<th>Previous meter reading :</th>
<td><input type="text" name="conpmr" value=<%=cnpmr%>></td>
</tr>
<tr>

```

```

<th> Current meter reading:</th>
<td><input type="text" name="concmr" value=<%=cncmr%>></td>
</tr>
<tr>
<th> Units consumed:</th>
<td><input type="text" name="conunits" value=<%=units%>></td>
</tr>
<tr>
<th> Amount to be paid:</th>
<td><input type="text" name="conamt" value=<%=tamt%>></td>
</tr>
</table>
</form>
<%
}
catch(NumberFormatException exp){
out.println("There is something wrong:"+exp);
}
%>
</body>
</html>

```

## OUTPUT

<i>ELECTRICITY BILL</i>	
<b>Consumer Name:</b>	priyu
<b>Connection no:</b>	21
<b>Date:</b>	02/01/2024
<b>Zone Name:</b>	cbe
<b>Previous meter reading :</b>	100
<b>Current meter reading:</b>	200

Submit

## TAMILNADUELECTRICITY BOARD

### *ELECTRICITY BILL*

<b>Consumer Name:</b>	priyu
<b>Connection no:</b>	21
<b>Date:</b>	02/01/2024
<b>Zone Name:</b>	cbe
<b>Previous meter reading :</b>	100
<b>Current meter reading:</b>	200
<b>Units consumed:</b>	100
<b>Amount to be paid:</b>	0.0

### RESULT

Thus, the above program is successfully executed and output is verified.

<b>Ex no:15</b>	<b>RESUME BUILDER SERVLET APPLICATION</b>
<b>Date:</b>	

### **AIM**

To develop a java application program to create a resume using servlet.

### **ALGORITHM**

**STEP 1:** Start the program.

**STEP 2:** Click the Start > All Programs > Netbeans

**STEP 3:** Click the File > New Project > Java Web application in the Netbeans platform.

**STEP 4:** Create a java servlet to perform their respective process.

**STEP 5:** The HTML file consists of the input fields needed for resume and a submit button.

**STEP 6:** The java servlet file consists of the process request in which the input given to HTML files is processed.

**STEP 7:** Declare the inputs in the servlet file to identify and display the inputs.

**STEP 8:** Save and run the program.

**STEP 9:** Stop the program.

## PROGRAM

### 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>
<form action="ResumeServlet">
<h1>RESUME</h1>
<hr>
Name:<br><input type="text" name="nme"/><br>
Email Id:<br><input type="email" name="mail"/><br>
Date of Birth:<br><input type="date" name="dte"/><br>
Gender:<br><input type="radio" name="gender" value="male" />Male<br>
<br><input type="radio" name="Gender" value="female" />Female<br>
Age:<br><input type="text" name="age"/><br>
Qualification:<br><input type="text" name="qualify"/><br>
Address:<br>
<textarea rows="5" cols="10" name="address">
</textarea>
<br>
Hobbies:<br>
<input type="text" name="hobby"/><br>
<input type="submit" name="submit"/><br>
</form>
</body>
</html>
```

### ResumeServlet.java

```
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 ResumeServlet extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
String Name=request.getParameter("nme");
String Email=request.getParameter("mail");
String Date=request.getParameter("dte");
String Gender=request.getParameter("gender");
String Age=request.getParameter("age");
String Qual=request.getParameter("qualify");
String Address=request.getParameter("address");
String Hobbies=request.getParameter("hobby");
out.println("Hi "+Name+"Welcome to Servlet page"+"<br>");
out.println("Name:"+Name+"<br>");
out.println("Email-Id:"+Email+"<br>");
out.println("Date of Birth:"+Date+"<br>");
out.println("Gender"+Gender+"<br>");
```



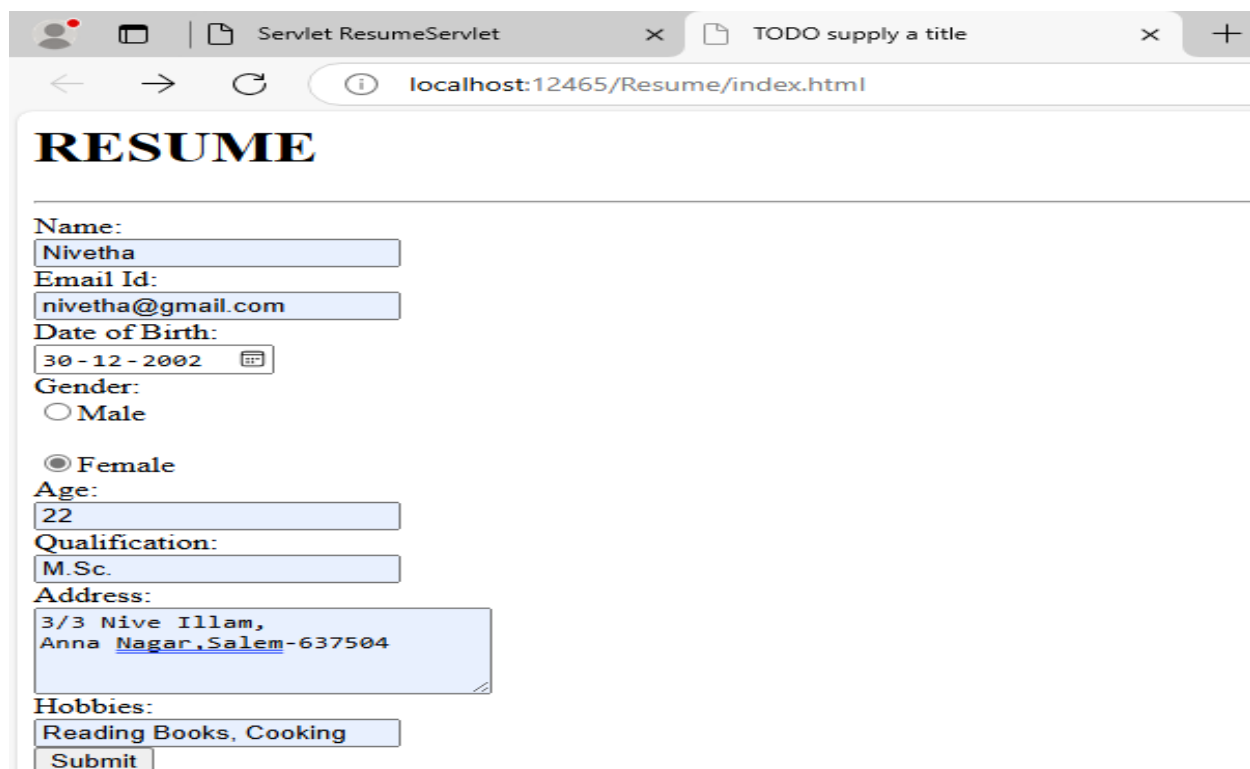
```

out.println("Age:"+Age+"<br>");
out.println("Qualification:"+Qual+"<br>");
out.println("Address:"+Address+"<br>"); out.println("Hobbies"+Hobbies+"<br>"); }
}
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
@Override
public String getServletInfo() {
return "Short description";
} }

```

## OUTPUT

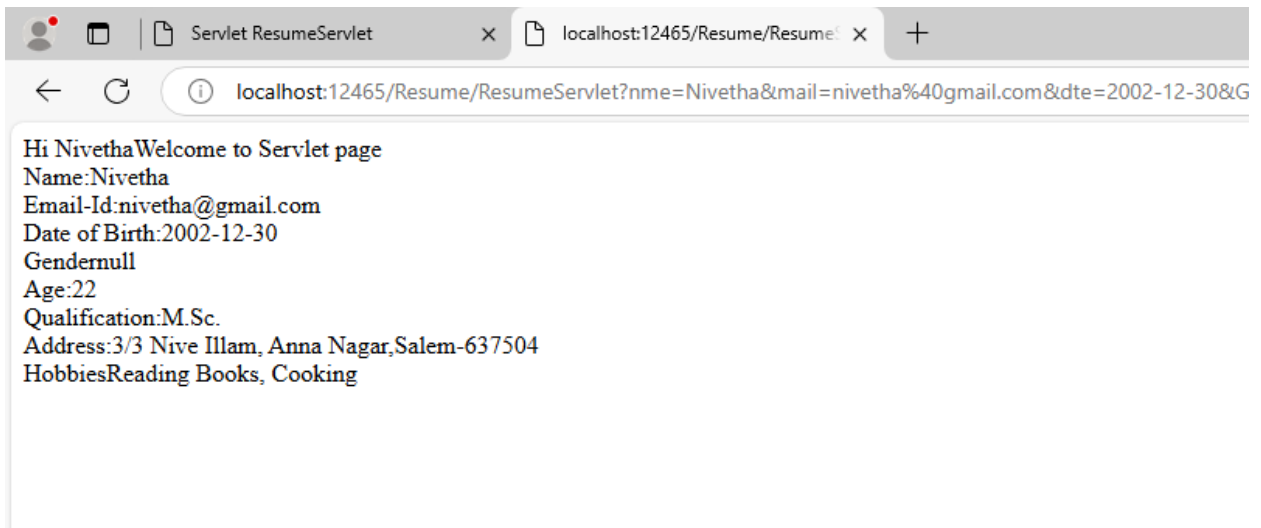
### index.html



The screenshot shows a web browser window with two tabs: 'Servlet ResumeServlet' and 'TODO supply a title'. The address bar shows 'localhost:12465/Resume/index.html'. The page content is a resume form titled 'RESUME' in a large, bold, serif font. Below the title is a horizontal line. The form contains the following fields and values:

- Name:** Nivetha
- Email Id:** nivetha@gmail.com
- Date of Birth:** 30-12-2002 (with a calendar icon)
- Gender:**
  - ☐ Male
  - ☒ Female
- Age:** 22
- Qualification:** M.Sc.
- Address:** 3/3 Nive Illam, Anna Nagar, Salem-637504
- Hobbies:** Reading Books, Cooking
- Submit** button

## ResumeServlet.java



## RESULT

Thus, the program is executed successfully and the output is verified.