



**AMITY UNIVERSITY**  
UTTAR PRADESH

## **ADVANCED JAVA PROGRAMMING[IT404]**

### **PRACTICAL FILE**



**AMITY SCHOOL OF ENGINEERING  
AND TECHNOLOGY AMITY  
UNIVERSITY CAMPUS, SECTOR-  
125, NOIDA-201303**

**Submitted By-  
Anchal Kumari  
A12405218083 (6CSE-3X)**

**Submitted To-  
Dr. Shilpi Sharma**

## INDEX

S. No.	Name of Program	Date of Perform/Conduct	Date of Submission	Maximum Marks	Marks Obtained	Signature of Faculty
1.	Write a program in java to make a registration form in applet using exception handling.	5/12/2020	12/01/2021			Checked
2.	Write a java program to create three frames: home page, login page and catalogue page.	12/01/2021	19/01/2021			Checked
3.	Write a java program to create a table insert, update and delete record(s) from the table.	19/01/2021	2/02/2021			Checked
4.	Write a java program to insert record in the table using PreparedStatement. Also perform resultset	2/02/2021	9/02/2021			checked
5.	WAP using servlet to 1) Print hello world 2) To create a form and accept the form values 3) Display visitor count 4) Wap for validating userid and password	5/02/2021	16/02/2021			Checked
6.	Write a program to input your name and display it in the browser window using jsp.	15/02/2021	23/02/2021			Good checked

**ANCHAL KUMARI**  
**A12405218083**

7.	Write a program to make a loan calculator using jsp.	15/02/2021	23/02/2021			
8.	Write a java program to (a) take input from the user (b) display the total number of visitor using cookies.	15/02/2021	23/02/2021			Good Checked
9.	Write a program to jdbc_servlet connectivity	15/02/2021	23/02/2021			Good Checked
10	Write a program 1. To print 10 number using scriptlet 2. To accept form values and print the form values using jsp 3. To handle exception 4. To calculate salary 5. To handle error	02/03/2021	09/03/2021			Good Checked
11	Write a program to implement stateless session beans	09/03/2021	16/03/2021			Good Checked
12	Write a program to implement stateful session beans	09/03/2021	16/03/2021			Good Checked
13	Write a program to implement EJB	16/03/2021	23/03/2021			Good Checked

**ANCHAL KUMARI**  
**A12405218083**

14	Write a program to implement struts	18/03/2021	23/03/2021			Good Checked
15	Write a program to implement javaMail API	24/03/2021	05/03/2021			
16	Write a program to implement spring	24/03/2021	05/03/2021			
17	Open ended experiment (movie review website)	24/03/2021	31/03/2021			

**ANCHAL KUMARI**  
**A12405218083**

## EXPERIMENT-1

**AIM: Design a registration form using applet and exception handling.**

**THEORY:** An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.

Exception handling ensures that the flow of the program doesn't break when an exception occurs. For example, if a program has bunch of statements and an exception occurs mid-way after executing certain statements then the statements after the exception will not execute and the program will terminate abruptly.

By handling we make sure that all the statements execute and the flow of program doesn't break.

### CODE:

```
package registerform;
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.Exception;

public class RegisterForm extends Applet
{

    GridBagLayout grid=new GridBagLayout();
    GridBagConstraints c=new GridBagConstraints();
    TextField eno, name, email, cno;
    Choice section;
    CheckboxGroup elective;
    Button submit;
    String email1, cno1;

    public void init()
    {
```

```
setBackground(Color.cyan.brighter());
resize(700,650);
setLayout(grid);
setFont(new Font("New Times Roman",Font.BOLD,18));
setForeground(Color.blue.darker());
Font f1=new Font("New Times Roman",Font.BOLD,30);
Label main=new Label("Amity University, Uttar Pradesh");
main.setFont(f1);
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(main, c);
add(main);
```

```
Font f2=new Font("New Times Roman",Font.BOLD|Font.ITALIC,22);
Label sub=new Label("ASET CSE", Label.RIGHT);
sub.setFont(f2);
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(sub, c);
add(sub);
```

```
Label spacer=new Label("");
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(spacer, c);
add(spacer);
Label enolabel=new Label("Enrolment No. ");
Label namelabel=new Label("Name");
eno=new TextField(30);
name=new TextField(30);
eno.setForeground(Color.black);
name.setForeground(Color.black);
c.anchor=GridBagConstraints.WEST;
c.gridwidth=GridBagConstraints.RELATIVE;
grid.setConstraints(enolabel, c);
```

```
add(enolabel);
c.anchor=GridBagConstraints.WEST;
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(eno, c);
add(eno);
spacer=new Label("");
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(spacer, c);
add(spacer);
c.anchor=GridBagConstraints.WEST;
c.gridwidth=GridBagConstraints.RELATIVE;
grid.setConstraints(namelabel, c);
add(namelabel);
c.anchor=GridBagConstraints.WEST;
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(name, c);
add(name);
spacer=new Label("");
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(spacer, c);
add(spacer);
```

```
Label sectionlabel=new Label("Section");
c.anchor=GridBagConstraints.WEST;
c.gridwidth=GridBagConstraints.RELATIVE;
grid.setConstraints(sectionlabel, c);
add(sectionlabel);
```

```
section = new Choice();
section.addItem("6CSE3-X");
section.addItem("6CSE3-Y");
section.setForeground(Color.black);
```

```
c.anchor=GridBagConstraints.WEST;  
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(section, c);  
add(section);  
spacer=new Label("");  
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(spacer, c);  
add(spacer);
```

```
Label electivelabel=new Label("Elective:");  
c.anchor=GridBagConstraints.WEST;  
c.gridwidth=GridBagConstraints.RELATIVE;  
grid.setConstraints(electivelabel, c);  
add(electivelabel);  
elective=new CheckboxGroup();
```

```
Checkbox networking = new Checkbox("Networking", elective, false);  
Checkbox cloud = new Checkbox("Cloud", elective, false);  
Checkbox android = new Checkbox("Android App", elective, false);  
c.anchor=GridBagConstraints.WEST;  
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(networking, c);  
add(networking);  
c.anchor=GridBagConstraints.WEST;  
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(cloud, c);  
spacer=new Label("");  
c.gridwidth=GridBagConstraints.RELATIVE;  
grid.setConstraints(spacer, c);  
add(spacer);  
add(cloud);  
c.anchor=GridBagConstraints.WEST;
```



```
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(android, c);  
spacer=new Label("");  
c.gridwidth=GridBagConstraints.RELATIVE;  
grid.setConstraints(spacer, c);  
add(spacer);  
add(android);
```

```
spacer=new Label("");  
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(spacer, c);  
add(spacer);  
Label emaillabel=new Label("Email ID ");  
Label cnolabel=new Label("Contact no. ");  
email=new TextField(30);  
cno=new TextField(30);  
email.setForeground(Color.black);  
cno.setForeground(Color.black);  
c.anchor=GridBagConstraints.WEST;  
c.gridwidth=GridBagConstraints.RELATIVE;  
grid.setConstraints(emaillabel, c);  
add(emaillabel);  
c.anchor=GridBagConstraints.WEST;  
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(email, c);  
add(email);  
spacer=new Label("");  
c.gridwidth=GridBagConstraints.REMAINDER;  
grid.setConstraints(spacer, c);  
add(spacer);  
c.anchor=GridBagConstraints.WEST;  
c.gridwidth=GridBagConstraints.RELATIVE;
```

```
grid.setConstraints(cnolabel, c);
add(cnolabel);
c.anchor=GridBagConstraints.WEST;
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(cno, c);
add(cno);
spacer=new Label("");
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(spacer, c);
add(spacer);
```

```
submit=new Button("Save and Submit");
submit.setForeground(Color.red.darker());
c.anchor=GridBagConstraints.CENTER;
c.gridwidth=GridBagConstraints.REMAINDER;
c.fill = GridBagConstraints.HORIZONTAL;
grid.setConstraints(submit, c);
add(submit);
```

```
spacer = new Label("");
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(spacer, c);
add(spacer);
Label submittext=new Label();
submittext.setBackground(Color.pink.brighter());
submittext.setFont(new Font("New Times Roman",Font.BOLD, 13));
submittext.setForeground(Color.red.darker());
c.anchor=GridBagConstraints.CENTER;
c.gridwidth=GridBagConstraints.REMAINDER;
grid.setConstraints(submittext, c);
add(submittext);
```

```
submit.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        email1 = email.getText();
        cno1 = cno.getText();
        try
        {
            if(!(email1.contains("'"+"@" )    &&    email1.contains("'"+"." )    &&
cno1.length()==10))
            {
                throw(new Exception());
            }
            else
            {
                submittext.setText("Your form has been submitted successfully!");
            }
        }
        catch(Exception exc)
        {
            if(email1.contains("'"+"@" )    &&    !email1.contains("'"+"." )    &&
cno1.length()==10)
            {
                submittext.setText("WARNING: Your email must contain .");
            }

            if(!email1.contains("'"+"@" )    &&    email1.contains("'"+"." )    &&
cno1.length()==10)
            {
                submittext.setText("WARNING: Your email must contain @");
            }
        }
    }
}
```

```
        if(email1.contains("'"+"@" ) && !email1.contains("'"+".") && cno1.length()!=10)
        {
            submittext.setText("WARNING: Your email must contain . and number must
contain 10 digits.");
        }
        if(!email1.contains("'"+"@" ) && email1.contains("'"+".") && cno1.length()!=10)
        {
            submittext.setText("WARNING: Your email must contain @ and number must
contain 10 digits.");
        }
        if(!email1.contains("'"+"@" ) && !email1.contains("'"+".") &&
cno1.length()==10)
        {
            submittext.setText("WARNING: Your email must contain @ and .");
        }
        if(email1.contains("'"+"@" ) && email1.contains("'"+".") && cno1.length()!=10)
        {
            submittext.setText("WARNING: Your number must contain 10 digits.");
        }
        if(!email1.contains("'"+"@" ) && !email1.contains("'"+".") &&
cno1.length()!=10)
        {
            submittext.setText("WARNING: Check your email and contact number");
        }
    }
});
}
```

OUTPUT-

**Amity University, Uttar Pradesh**  
**ASET CSE**

Enrollment No.

A12405218083

Name

ANCHAL KUMARI

Section

6CSE3-X

Elective:

☒ Machine learning

☐ Cloud computing

☐ Android App

Email ID

anchal@gmail.com

Contact no.

6203871908

Save and Submit

Your form has been submitted successfully!

**Amity University, Uttar Pradesh**  
**ASET CSE**

Enrollment No.

A12405218083

Name

ANCHAL KUMARI

Section

6CSE3-X

Elective:

☒ Machine learning

☐ Cloud computing

☐ Android App

Email ID

anchal@gmail

Contact no.

6203

Save and Submit

WARNING: Your email must contain . and number must contain 10 digits.

## EXPERIMENT-2

**AIM: Write a program to create library information page.**

**1) HOME PAGE: The static home page must contain three frames.**

**2) LOGIN PAGE**

**3) CATALOGUE PAGE:**

**The catalogue page should contain the details of all the books available in the web site in a table"**

### **THEORY:**

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

**Java Swing** is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

### **CODE:**

#### **Home page**

```
package library1;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Library1 extends JFrame implements ActionListener{

    private JPanel contentPane;
    JLabel l1;
    JButton b1,b2,b3;

    public Library1() {
        setBounds(100, 150, 600, 600);

        //        setSize(700,500);
        setLayout(null);
        setLocation(300,300);

        l1 = new JLabel(" ");
        l1.setBounds(100, 46, 500, 35);
        l1.setForeground(new Color(204, 51, 102));
```

```
        l1.setFont(new Font("Segoe UI Semilight", Font.BOLD, 30));

//        l1.setBounds(268, 30, 701, 80);
        b1 = new JButton("Home");
        b2 = new JButton("Login/Signup");
        b3 = new JButton("Catalogue");

        ImageIcon i1 = new
        ImageIcon(ClassLoader.getResource("Library1/image/library.jpg"));
        Image i3 = i1.getImage().getScaledInstance(600,600,Image.SCALE_DEFAULT);
        ImageIcon i2 = new ImageIcon(i3);
        l1 = new JLabel(i2);

        b1.setBounds(0,5,200,30);
        b2.setBounds(200,5,200,30);
        b3.setBounds(400,5,200,30);
        l1.setBounds(0, 100, 600, 400);

        l1.add(b1);
        l1.add(b2);
        l1.add(b3);
        add(l1);

        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
    }

    public void actionPerformed(ActionEvent ae){

        if(ae.getSource() == b1){
            this.setVisible(false);
            new Library1().setVisible(true);
        }
        if(ae.getSource() == b2){
            this.setVisible(false);
            new Login().setVisible(true);
        }
        if(ae.getSource() == b3){
            this.setVisible(false);
            new Catalogue().setVisible(true);
        }
    }

    public static void main(String[] args) {
        Library1 window = new Library1();
    }
```

```
        window.setVisible(true);  
    }  
}
```

### **Login page**

```
import java.awt.*;  
import javax.swing.*;  
import java.awt.event.*;  
import java.sql.*;  
  
public class Login_User extends JFrame implements ActionListener{  
  
    private JPanel panel;  
    private JTextField textField;  
    private JPasswordField passwordField;  
    private JButton b1,b2,b3;  
  
    public Login_User() {  
  
        setBackground(new Color(169, 169, 169));  
        setBounds(600, 300, 600, 400);  
  
        panel = new JPanel();  
        panel.setBackground(new Color(176, 224, 230));  
        setContentPane(panel);  
        panel.setLayout(null);  
  
        JLabel l1 = new JLabel("Username : ");  
        l1.setBounds(124, 89, 95, 24);  
        panel.add(l1);  
  
        JLabel l2 = new JLabel("Password : ");  
        l2.setBounds(124, 124, 95, 24);  
        panel.add(l2);  
  
        textField = new JTextField();  
        textField.setBounds(210, 93, 157, 20);  
        panel.add(textField);  
  
        passwordField = new JPasswordField();  
        passwordField.setBounds(210, 128, 157, 20);  
        panel.add(passwordField);  
  
        JLabel l3 = new JLabel("");  
        l3.setBounds(377, 79, 46, 34);  
        panel.add(l3);  
    }  
}
```



```
JLabel l4 = new JLabel("");
l4.setBounds(377, 124, 46, 34);
panel.add(l3);

b1 = new JButton("Login");
b1.addActionListener(this);

b1.setForeground(new Color(46, 139, 87));
b1.setBackground(new Color(250, 250, 210));
b1.setBounds(149, 181, 113, 39);
panel.add(b1);

b2 = new JButton("SignUp");
b2.addActionListener(this);
b2.setForeground(new Color(139, 69, 19));
b2.setBackground(new Color(255, 235, 205));
b2.setBounds(289, 181, 113, 39);
panel.add(b2);

}
public void actionPerformed(ActionEvent ae){

}

public static void main(String[] args) {
    new Login_User().setVisible(true);
}

}
```

### **Catalogue page:**

```
import java.applet.Applet;
import javax.swing.*;
import java.awt.*;
public class CataloguePage extends Applet {
    JTable Book_Cat ;
    public void init() {
        setBackground(Color.CYAN);
        String data[][] = { {"AD102","COA","Morris maano","317.00"},
                             {"DA12","Word Power Made Easy","Norman Lewis","151.00"} };
        String column[] = {"ID","Book Name","Author","Price"};
        JTable Book_Cat = new JTable(data , column);
        Book_Cat.setBounds(20,40,500,700);
        JScrollPane sp = new JScrollPane(Book_Cat);
        this.add(sp);
    }
}
```

```
this.setSize(400,500);  
this.setVisible(true);  
}  
}
```

## OUTPUT:



Username :

Password :

**DATE: 12/01/2021**

ID	Book Name	Author	Price
AD102	COA	Morris maano	317.00
DA12	Word Power Made...	Norman Lewis	151.00

**ANCHAL KUMARI**  
**A12405218083**

## EXPERIMENT-3

### AIM:

1. To create a Employee table
2. Write a program to store emp records and fetch the emp records
3. To delete an emp record
4. To update an emp record

### THEORY:

JDBC stands for **Java Database Connectivity**, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.
- **Execute a query:** Requires using an object of type Statement for building and submitting an SQL statement to delete records from a table. This Query makes use of the **WHERE** clause to delete conditional records.
- **Execute a query:** Requires using an object of type Statement for building and submitting an SQL statement to update records in a table. This Query makes use of **IN** and **WHERE** clause to update conditional records.
- **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.
- **Connection:** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
- **Statement:** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
- **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
- **SQLException:** This class handles any errors that occur in a database application.

**To create a Employee table**

#	EMP_ID	EMPNAME	EMAIL	DEPARTMENT
1		1001 ANCHAL	anchal@yahoo.com	manager
2		1002 Kashish Tinguria	k35@yahoo.com	HR
3		1003 Sheetal	s35@yahoo.com	sales
4		1004 Divya	d35@yahoo.com	manager
5		1005 Swati	s35@yahoo.com	sales
6		1006 Aniket	a35@yahoo.com	HR
7		1007 Aman	aman@yahoo.com	manager
8		1008 Ananya	ananya@yahoo.com	sales
9		1009 Simran	si@yahoo.com	HR
10		1010 Rohit	r@yahoo.com	sales

**Write a program to store emp records and fetch the emp records**

- 1. To delete an emp record**
- 2. To update an emp record**

**CODE:**

```

package jdbcexp1;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class Jdbcexp1 {
    public static final String DBURL="jdbc:derby://localhost:1527/EmployeeRecord";
    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Connection con=null;
        Statement stmt=null;
        try
        {
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            con=DriverManager.getConnection(DBURL);
            stmt=con.createStatement();
            //updating record
            String sql="UPDATE EMPRECORD set email='ktinguria35@yahoo.com' WHERE
EMP_ID=1001 ";
            int count= stmt.executeUpdate(sql);
            System.out.println("Updated Record"+count);
            //deleting record
            String sql2="DELETE from EMPRECORD where emp_id=1009";

```

**ANCHAL KUMARI**  
**A12405218083**

```
stmt.executeUpdate(sql2);

System.out.println("Deleted Record successfully!");
ResultSet rs=stmt.executeQuery("Select * from EMPRECORD");
while(rs.next())
{
    int id=rs.getInt("EMP_ID");
    String name=rs.getString("EMPNAME");
    String email=rs.getString("EMAIL");
    String dept=rs.getString("DEPARTMENT");
    System.out.println(id + "\t\t" + name + "\t\t" + email+ "\t\t"+dept);
}
}
catch(SQLException se)
{
    se.printStackTrace();
}
}
```

## OUTPUT:

```
run:
Updated Record1
Deleted Record successfully!
1001      Kashish Tinguria      ktinguria35@yahoo.com      Manager
1002      Ankit Singh          asingh23@yahoo.com         HR
1003      Ankur Tripathi       ankur45t@gmail.com         Sales
1004      Divya Jain           divyaj@gmail.com           Manager
1005      Niharika Malhotra     malhotran@hotmail.com      Sales
1006      Deepak Singh         ds@yahoo.com               HR
1007      Ankita Tripathi       at456l@gmail.com           Accountant
1008      R.S. Sharma          rs23@gmail.com             HR
1010      Ankit Singh          ankitsingh@hotmail.com     Sales
1011      Prashant Saxena       prasaxena45@gmail.com      Manager
BUILD SUCCESSFUL (total time: 0 seconds)
```

## EXPERIMENT-4

**AIM:** Write a java program to insert record in the table using PreparedStatement also perform resultset.

**THEORY:** A Java JDBC PreparedStatement is a special kind of Java JDBC Statement object with some useful additional features. Remember, you need a `Statement` in order to execute either a query or an update. You can use a Java JDBC PreparedStatement instead of a `Statement` and benefit from the features of the PreparedStatement. The Java JDBC PreparedStatement primary features are:

- Easy to insert parameters into the SQL statement.
- Easy to reuse the PreparedStatement with new parameter values.
- May increase performance of executed statements.
- Enables easier batch updates.

### ResultSet interface

The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row. By default, ResultSet object can be moved forward only and it is not updatable. But we can make this object to move forward and backward direction by passing either `TYPE_SCROLL_INSENSITIVE` or `TYPE_SCROLL_SENSITIVE` in `createStatement(int,int)` method as well as we can make this object as updatable by:

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE  
ResultSet.CONCUR_UPDATABLE);
```

The SQL statements that read data from a database query, return the data in a result set. The `SELECT` statement is the standard way to select rows from a database and view them in a result set. The `java.sql.ResultSet` interface represents the result set of a database query.

A `ResultSet` object maintains a cursor that points to the current row in the result set. The term "result set" refers to the row and column data contained in a `ResultSet` object.

The methods of the `ResultSet` interface can be broken down into three categories –

- **Navigational methods:** Used to move the cursor around.
- **Get methods:** Used to view the data in the columns of the current row being pointed by the cursor.
- **Update methods:** Used to update the data in the columns of the current row. The updates can then be updated in the underlying database as well.

### Type of ResultSet

The possible `RSType` are given below. If you do not specify any `ResultSet` type, you will automatically get one that is `TYPE_FORWARD_ONLY`.

Type	Description
ResultSet.TYPE_FORWARD_ONLY	The cursor can only move forward in the result set.
ResultSet.TYPE_SCROLL_INSENSITIVE	The cursor can scroll forward and backward, and the result set is not sensitive to changes made by others to the database that occur after the result set was created.
ResultSet.TYPE_SCROLL_SENSITIVE.	The cursor can scroll forward and backward, and the result set is sensitive to changes made by others to the database that occur after the result set was created.

**CODE:**

```
import java.io.*;
import java.sql.*;
import java.lang.Exception;
public class emp{
public static void main(String args[])throws Exception{
    Class.forName("org.apache.derby.jdbc.ClientDriver");
    Connection
con=DriverManager.getConnection("jdbc:derby://localhost:1527/employee_records");

    PreparedStatement ps=con.prepareStatement("insert into empl values(?,?,?,?)");

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

    do{
        System.out.println("enter id:");
        int id=Integer.parseInt(br.readLine());
        System.out.println("enter firstname:");
        String firstname=br.readLine();
        System.out.println("enter lastname:");
```



```
String lastname=br.readLine();
System.out.println("enter salary:");
int salary=Integer.parseInt(br.readLine());
System.out.println("enter hire_date:");
String hire_date=br.readLine();

ps.setInt(1,id);
ps.setString(2,firstname);
ps.setString(3,lastname);
ps.setInt(4,salary);
ps.setString(5,hire_date);
int i=ps.executeUpdate();
System.out.println(i+" records affected");

System.out.println("Do you want to continue: yes/no");
String s=br.readLine();
if(s.startsWith("n")){
    break;
}
}while(true);
PreparedStatement pst = con.prepareStatement(
"SELECT * from empl");
ResultSet rs;
rs=pst.executeQuery();
System.out.print("EMP_ID\tFirst Name\tLast Name\t SALARY\n");
while(rs.next())
{
    int id=rs.getInt("EMPLOYEE_id");
    String FNAME=rs.getString("FIRSTNAME");
    String LNAME=rs.getString("LASTNAME");
    int SAL=rs.getInt("SALARY");

    String DATE=rs.getString("HIRE_DATE");

    System.out.println(id+"\t"+FNAME +"\t"+LNAME+"\t"+SAL);

}

System.out.println("RESULT SET UPDATED SALARY BY 500\n");
String sql3 = "SELECT * FROM EMPL";
PreparedStatement pst3;
pst3 = con.prepareStatement(sql3, ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
ResultSet rs1= null;
rs1 = pst3.executeQuery();
while(rs1.next())
{
```

```
int sal_u = rs1.getInt("SALARY") + 500;
rs1.updateDouble( "SALARY", sal_u );
rs1.updateRow();
```

```
System.out.println(rs1.getInt("EMPLOYEE_id")+"\t"+rs1.getString("FIRSTNAME")+rs1.ge
tString("LASTNAME")+"\t"+rs1.getInt("SALARY"));
```

```
}
```

```
con.close();
```

```
}}
```

## OUTPUT:

```
Output X
EmpDb (run) X Java DB Database Process X S
run:
enter id:
15
enter firstname:
Priya
enter lastname:
Chauhan
enter salary:
23000
enter hire_date:
2016-09-08
1 records affected
Do you want to continue: yes/no
yes

enter id:
16
enter firstname:
Ritu
enter lastname:
Singh
enter salary:
10000
enter hire_date:
2016-03-05
1 records affected
Do you want to continue: yes/no
no

EMP_ID      First Name    Last Name    SALARY
1           Sheetal      K            3000
2           Nikhil       B            36000
4           Swati        T            6000
5           Divya        S            35500
6           Dhruv        R            8000
7           Anchal       K            9000
8           Shreya       K            10000
9           Kashish      T            11000
10          Shivam       P            12000
15          Priya        Chauhan      23000
16          Ritu         Singh        10000

1           Sheetal K            3500
2           Nikhil B            36500
4           Swati T            6500
5           Divya S            36000
6           Dhruv R            8500
7           Anchal K            9500
8           Shreya K            10500
9           Kashish T            11500
10          Shivam P            12500
15          Priya Chauhan 23500
16          Ritu Singh 10500
RESULT SET UPDATED SALARY BY 500

BUILD SUCCESSFUL (total time: 57 seconds)
```

## EXPERIMENT-5

**AIM:** WAP using servlet to

- 1) Print hello world
- 2) To create a form and accept the form values
- 3) Display visitor count
- 4) validating userid and password

**THEORY:** Servlet can be described in many ways, depending on the context.

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.

There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

**There are many problems in CGI technology:**

1. If the number of clients increases, it takes more time for sending the response.
2. For each request, it starts a process, and the web server is limited to start processes.
3. It uses platform dependent language e.g. C, C++.

**Advantage of using servlet over CGI technology:**

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because it uses java language.

**CODE:**

**FirstPage.java**

```
package FirstPage;

import java.io.IOException;

import java.io.PrintWriter;

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(name = "FirstPage", urlPatterns = { "/FirstPage" })

public class FirstPage extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

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

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

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

            out.println("<html>");

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

```
        out.println("<title>Welcome Page</title>");

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

        out.println("<body><h1 align = \"center\">Hello World<h1>");

        out.println("    <div    style=\"text-align:center;\">        <form    action=\"SecondPage\"
        method=\"get\">

                +            "<input                type=\"submit\"                name=\"btnadd\"
        value=\"LOGIN\"></form></div>");

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

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

    }

}

}
```

### **SecondPage.java**

```
package FirstPage;

import java.io.IOException;

import java.io.PrintWriter;

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(name = "SecondPage", urlPatterns = {"/SecondPage"})

public class SecondPage extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

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

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

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

            out.println("<html>");

            out.println("<head>");

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

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

            out.println("<body>");

            out.println("<p>Enter login details</p>\n" +

"    <form action=\"ThirdPage\" method=\"post\">\n" +

"        Username: <input type=\"text\" name=\"username\"><br><br>\n" +

"        Password: <input type=\"password\" name=\"password\"><br><br>\n" +

"        <input type=\"submit\" name=\"btnadd\" value=\"Login\">\n" +
```

```
"    </form>");

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

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

### **ThirdPage.java**

```
package FirstPage;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;

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(name = "ThirdPage", urlPatterns = { "/ThirdPage" })

public class ThirdPage 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 user = request.getParameter("username");

String pass = request.getParameter("password");

if(user.equals("anchal") && pass.equals("anchal"))

{

    RequestDispatcher rs = request.getRequestDispatcher("FourthPage");

    rs.forward(request, response);

}

else

{

    RequestDispatcher rs = request.getRequestDispatcher("SecondPage");

    out.println("Username or Password incorrect");

rs.include(request, response);

}

}

}}
```

**FourthPage.java**

```
package FirstPage;

import java.io.IOException;

import java.io.PrintWriter;
```



```
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(name = "FourthPage", urlPatterns = {"/FourthPage"})

public class FourthPage extends HttpServlet {

    private int hitCount;

    public void init(){

        hitCount = 0;

    }

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

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

        hitCount++;

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

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

            out.println("<body style=\"background-color:Thistle\">");

            out.println(" <h1>Hi "+n+"</h1>\n");

        }

    }

}
```

```
out.println("<p>Logged in successfully</p>\n");
```

```
out.println("<p>VISITOR COUNT="+ hitCount + "</p>\n");
```

```
}
```

```
}
```

```
}
```

OUTPUT:

# Hello World

LOGIN

**Username or Password incorrect**

Enter login details

Username:

Password:

Login

Enter login details

Username:

Password:

Login

# Hi anchal

Logged in successfully

VISITOR COUNT=2

## EXPERIMENT-6

**AIM: Write a program to input your name and display it in the browser window using jsp.**

### **THEORY:**

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. This tutorial will teach you how to use Java Server Pages to develop your web applications in simple and easy steps.

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offers several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
- JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP, etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

### **CODE:**

#### **first.jsp**

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html><head><title>
Name Input Form
</title> </head>
<body>
<form method = "post" action ="second.jsp">
Enter your name
<input type= "text" name= "username"/>
<input type="submit" value= "submit"/>
</form>
</body>
</html>
```

#### **second.jsp**

```
<% String name =request.getParameter ("username");
```

```
    if (name == null)
        name = " ";
    session.setAttribute("username", name);
%>
<body>
<a href = "third.jsp">
Next page to view </a> </body></html>
```

### **third.jsp**

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@page language="Java"%>
<% String uname=(String)session.getAttribute("username");
if(uname==null)
uname=" ";
%>
<html>
<body>
Welcome <%= uname %>
</body></html>
```

### **index.html**

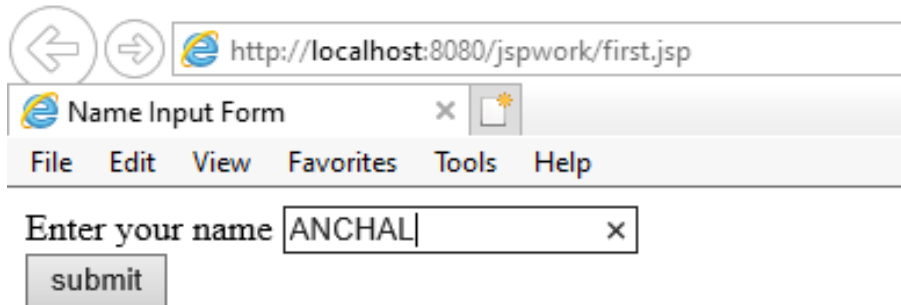
```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<div>TODO write content</div>
</body>
</html>
```

### **context.xml**

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

<Context path="/WebApplication1"/>

## OUTPUT:



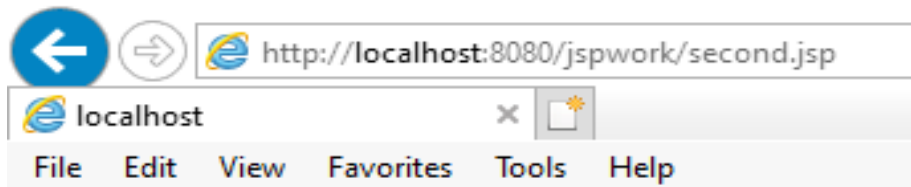
http://localhost:8080/jspwork/first.jsp

Name Input Form

File Edit View Favorites Tools Help

Enter your name

submit

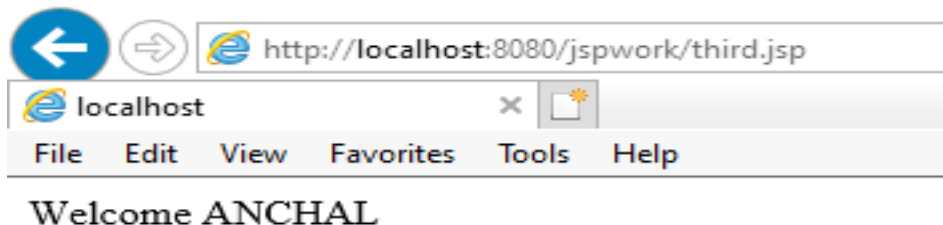


http://localhost:8080/jspwork/second.jsp

localhost

File Edit View Favorites Tools Help

[Next page to view](#)



http://localhost:8080/jspwork/third.jsp

localhost

File Edit View Favorites Tools Help

Welcome ANCHAL

## EXPERIMENT-7

**AIM: Write a program to make a loan calculator using jsp.**

### THEORY:

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. This tutorial will teach you how to use Java Server Pages to develop your web applications in simple and easy steps.

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offers several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
- JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP, etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

### CODE:

#### compound.jsp

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

<!DOCTYPE html>
<% @ page errorPage="error.jsp" %>
<%!
public double calculate(double amount, double interest, int period) {
    if(amount <= 0) {
        throw new IllegalArgumentException("Amount should be greater than 0: " + amount);
    }
    if(interest <= 0) {
        throw new IllegalArgumentException("Interest should be greater than 0: " + interest);
    }
    if(period <= 0) {
        throw new IllegalArgumentException("Period should be greater than 0: " + period);
    }
    return amount*Math.pow(1 + interest/100, period);
}
```

```

    }
    %>

<html>
<head>
    <title>Compound</title>
</head>
<body style="font-family:verdana;font-size:10pt;">
    <% @ include file="header.jsp" %>
    <%
        double amount = Double.parseDouble(request.getParameter("amount"));
        double interest = Double.parseDouble(request.getParameter("interest"));
        int period = Integer.parseInt(request.getParameter("period"));
    %>
    <b>Principal using compound interest:</b>
    <%= calculate(amount, interest, period) %>
    <br/><br/>
    <jsp:include page="footer.jsp"/>
</body>
</html>

```

### simple.jsp

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<% @ page errorPage="error.jsp" %>
<%!
    public double calculate(double amount, double interest, int period) {
        if(amount <= 0) {
            throw new IllegalArgumentException("Amount should be greater than 0: " + amount);
        }
        if(interest <= 0) {
            throw new IllegalArgumentException("Interest should be greater than 0: " + interest);
        }
        if(period <= 0) {
            throw new IllegalArgumentException("Period should be greater than 0: " + period);
        }
        return amount*(1 + period*interest/100);
    }
    %>

<html>
<head>
    <title>Simple</title>
</head>
<body style="font-family:verdana;font-size:10pt;">
    <% @ include file="header.jsp" %>

```



```

<%
    double amount = Double.parseDouble(request.getParameter("amount"));
    double interest = Double.parseDouble(request.getParameter("interest"));
    int period = Integer.parseInt(request.getParameter("period"));
%>
<b>Principal using simple interest:</b>
<%= calculate(amount, interest, period) %>
<br/><br/>
<jsp:include page="footer.jsp"/>
</body>
</html>

```

### **controller.jsp**

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%
    String type = request.getParameter("type");
    if(type.equals("S")) {
%>
<jsp:forward page="/simple.jsp"/>
<%
    } else {
%>
<jsp:forward page="/compound.jsp"/>
<%
    }
%>

```

### **error.jsp**

```

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

<!DOCTYPE html>
<% @ page isErrorPage="true" %>
<html>
<head>
<title>Simple</title>
</head>
<body style="font-family:verdana;font-size:10pt;">
<% @ include file="header.jsp" %>
<p style="color:#ff0000"><b><%= exception.getMessage() %></b></p>
<jsp:include page="footer.jsp"/>
</body>
</html>

```

**footer.jsp**

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%= new java.util.Date() %>
```

**header.jsp**

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<h3>Loan Calculator</h3>
```

**index.jsp**

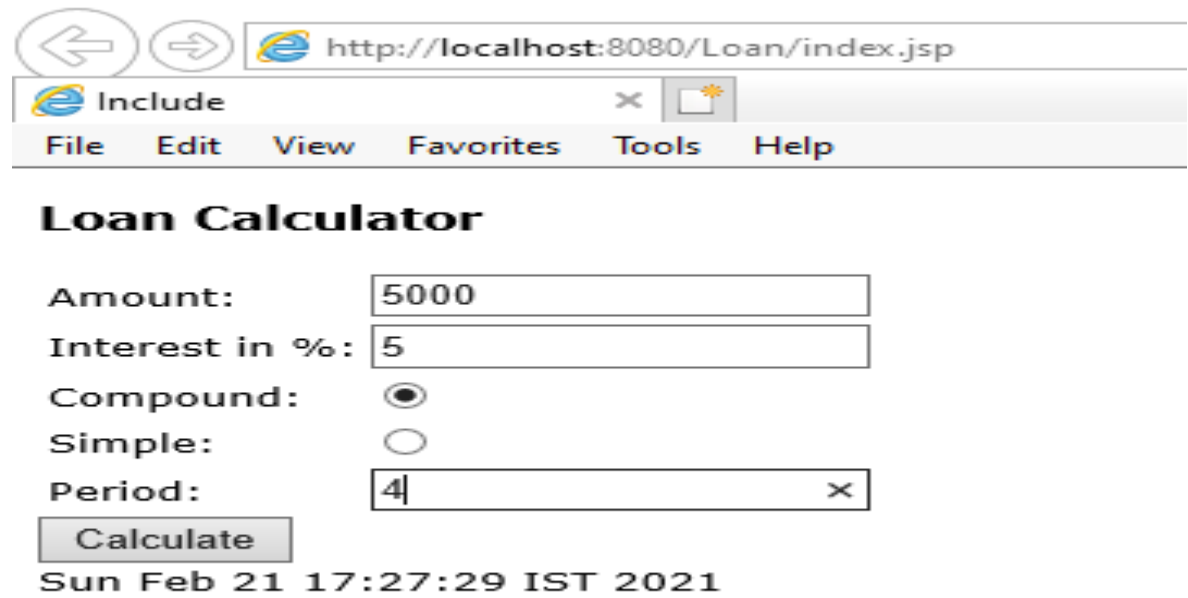
```
<% @page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
  <head>
    <title>Include</title>
  </head>
  <body style="font-family:verdana;font-size:10pt;">
    <% @ include file="header.jsp" %>
    <form action="controller.jsp">
      <table border="0" style="font-family:verdana;font-size:10pt;">
        <tr>
          <td>Amount:</td>
          <td><input type="text" name="amount" />
        </tr>
        <tr>
          <td>Interest in %:</td>
          <td><input type="text" name="interest"/></td>
        </tr>
        <tr>
          <td>Compound:</td>
          <td><input type="radio" name="type" value="C" checked/></td>
        </tr>
        <tr>
          <td>Simple:</td>
          <td><input type="radio" name="type" value="S" /></td>
        </tr>
        <tr>
          <td>Period:</td>
          <td><input type="text" name="period"/></td>
        </tr>
      </table>
      <input type="submit" value="Calculate"/>
    </form>
```

```
<jsp:include page="footer.jsp"/>
</body>
</html>
```

## OUTPUT:

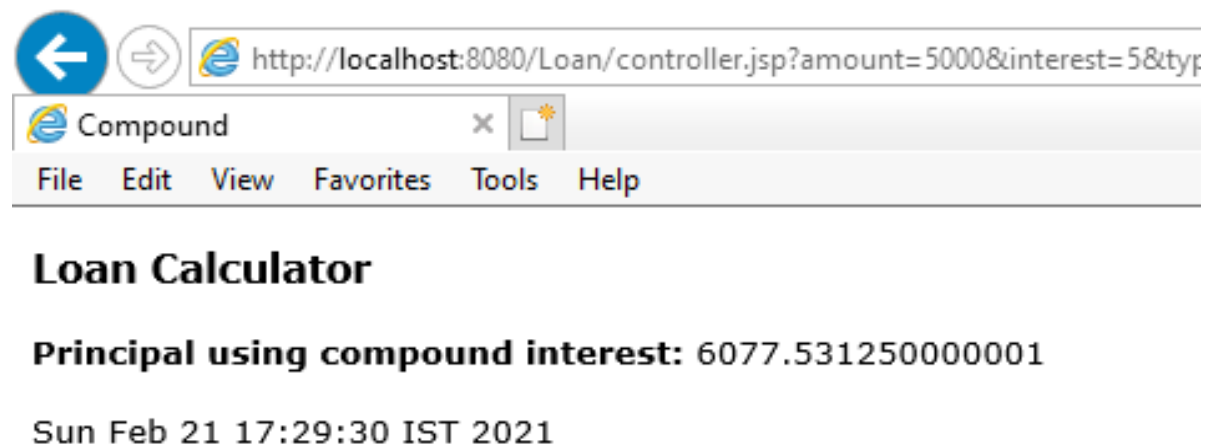
**Compound interest:**



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Loan/index.jsp`. The browser has a menu bar with `File`, `Edit`, `View`, `Favorites`, `Tools`, and `Help`. The page title is `Include`. The main content area displays the **Loan Calculator** form. The form includes the following fields and controls:

- Amount:** A text input field containing the value `5000`.
- Interest in %:** A text input field containing the value `5`.
- Compound:** A radio button that is selected.
- Simple:** An unselected radio button.
- Period:** A text input field containing the value `4`, with a close button (`x`) to its right.
- Calculate:** A button to submit the form.

Below the form, the timestamp `Sun Feb 21 17:27:29 IST 2021` is displayed.

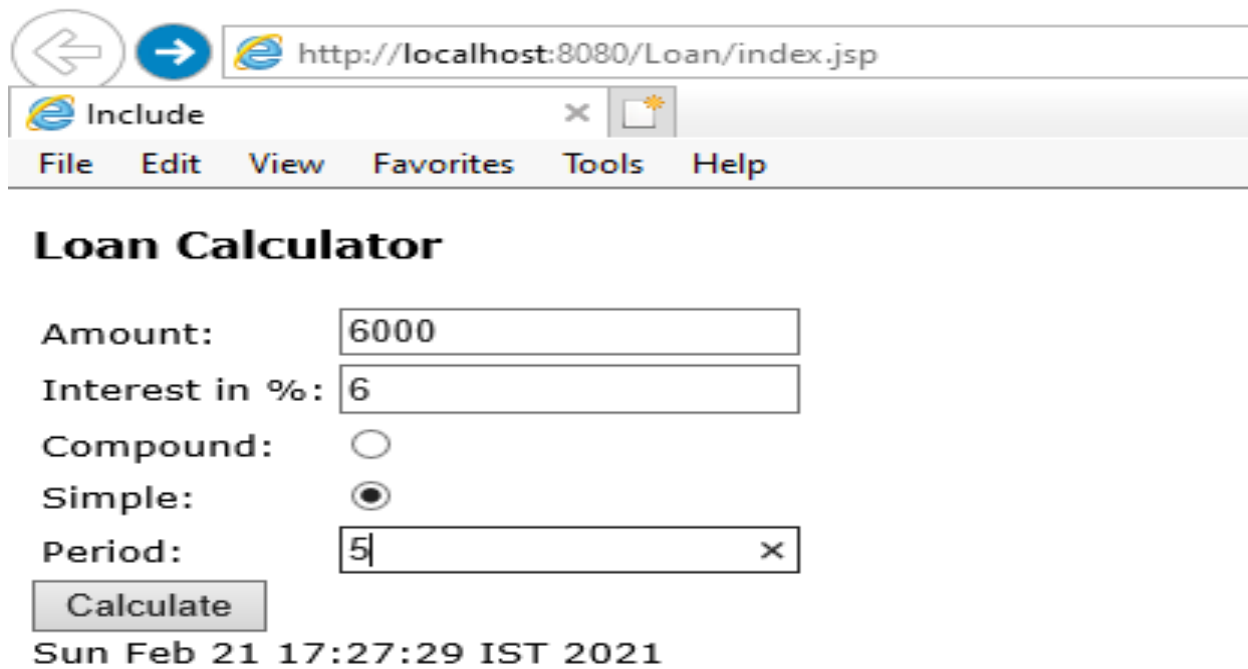


The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Loan/controller.jsp?amount=5000&interest=5&type=compound`. The browser has a menu bar with `File`, `Edit`, `View`, `Favorites`, `Tools`, and `Help`. The page title is `Compound`. The main content area displays the **Loan Calculator** result. The result is:

**Principal using compound interest: 6077.531250000001**

Below the result, the timestamp `Sun Feb 21 17:29:30 IST 2021` is displayed.

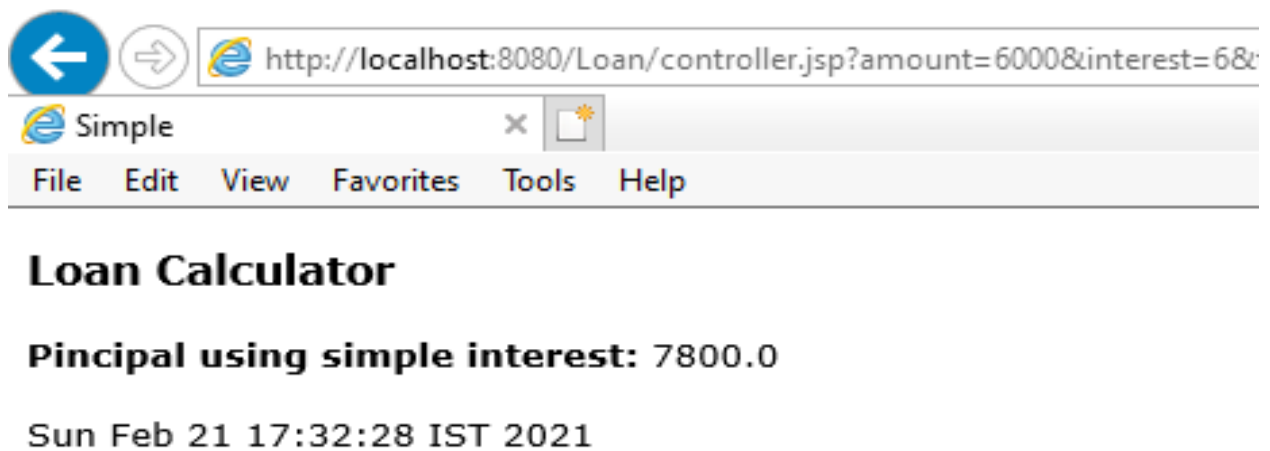
Simple interest:



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Loan/index.jsp`. The browser has a menu bar with 'File', 'Edit', 'View', 'Favorites', 'Tools', and 'Help'. Below the menu bar, there is a tab labeled 'Include'. The main content area displays the 'Loan Calculator' form. The form includes the following fields and controls:

- Amount:** A text input field containing the value '6000'.
- Interest in %:** A text input field containing the value '6'.
- Compound:** A radio button that is currently unselected.
- Simple:** A radio button that is currently selected.
- Period:** A text input field containing the value '5'.
- Calculate:** A button with the text 'Calculate'.

Below the form, the text 'Sun Feb 21 17:27:29 IST 2021' is displayed.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Loan/controller.jsp?amount=6000&interest=6&period=5`. The browser has a menu bar with 'File', 'Edit', 'View', 'Favorites', 'Tools', and 'Help'. Below the menu bar, there is a tab labeled 'Simple'. The main content area displays the 'Loan Calculator' results. The results are as follows:

- Pincipal using simple interest:** 7800.0

Below the results, the text 'Sun Feb 21 17:32:28 IST 2021' is displayed.

## EXPERIMENT-8

**AIM: Write a java program to**

**(a) take input from the user**

**(b) display the total number of visitor using cookies.**

**THEORY:** Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies. There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

### **Non-persistent cookie**

It is **valid for single session** only. It is removed each time when user closes the browser.

### **Persistent cookie**

It is **valid for multiple session** It is not removed each time when user closes the browser. It is removed only if user logout or signout.

### **Setting cookies with servlet involves three steps:**

(1) Creating a Cookie object- You call the Cookie constructor with a cookie name and a cookie value, both of which are strings.

```
Cookie cookie = new Cookie("key","value");
```

Keep in mind, neither the name nor the value should contain white space or any of the following characters: [ ] ( ) = , " / ? @ : ;

(2) Setting the maximum age - You use setMaxAge to specify how long (in seconds) the cookie should be valid. Following would set up a cookie for 24 hours.

```
cookie.setMaxAge(60 * 60 * 24);
```

(3) Sending the Cookie into the HTTP response headers- You use response.addCookie to add cookies in the HTTP response header as follows :

```
response.addCookie(cookie);
```

**CODE:****welcome.java**

```
package programs;
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class welcome extends HttpServlet {
    private String message;
    public void init() throws ServletException {
        message = "WELCOME TO HOME PAGE";
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        try(PrintWriter out = response.getWriter())
        {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Welcome Page</title>");
            out.println("<body style='background-color:cyan;'>");
            out.println("<center>");
            out.println("<h3><font size=50px color=blue>"+message+"</font></h3></br></br>");
            out.println("<div      style=\"text-align:      center;\"><form      action=\"Cookies\"
method=\"get\">"+<input      type=\"submit\"      name=\"btnadd\"
value=\"NEXT\"></form></div>");
            out.println("</center>");
            out.println("</body>");
            out.println("</html>");
        }
    }
    public void destroy() {
        // do nothing.
    }
}
```

**Cookies.java**

```
package programs;
```

```
import java.io.*;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class Cookies extends HttpServlet {
    public void init()
    {
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-8");
        try(PrintWriter out=response.getWriter())
        {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>LOGIN FORM</title>");
            out.println("<body style='background-color:magenta;'>");
            out.println("<center>");
            out.println("<form action = 'Cookies' method = 'POST'>");
            out.println("<h3><font                color=cyan                size=50px>LOGIN
FORM</font></h3></br></br>");
            out.println("<font size= 20px color=blue>Username </font>");
            out.println("<input type = 'text' id = 'username' name = 'username'></br></br>");
            out.println("<font size=20px color=blue>Password </font>");
            out.println("<input type = 'password' id = 'password' name = 'password'></br></br>");
            out.println("<input type = 'submit' value = 'LOGIN'></br></br>");
            out.println("</form>");
            out.println("</center>");
            out.println("</body>");
            out.println("</html>");
        }
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try(PrintWriter out=response.getWriter())
        {
```

```

String username= request.getParameter("username");
String password=request.getParameter("password");
if(username.equals("anchal")&&password.equals("123"))
{
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Login Validation</title>");
out.println("</head>");
out.println("<body style='background-color:green;'>");
out.println("<center><h3><font size=50px color=white> Login ID and password is
valid</font></h3>");
out.println("<div style='text-align: center;'><form action='visitors'
method='get'>"+<input type='submit' name='btnadd'
value='NEXT'></form></div>");
}
else if(username.isEmpty()){
out.println("<body style='background-color:black;'>");
out.println("<center>");
out.print("<h4><font size=50px color=red> Username or password field cannot be
null!</font></h4>");
out.println("<form action='Cookies' method='GET'");
out.println("<input type='submit' value='Try again'></br></br>");
out.println("</form>");
out.println("</center>");
out.println("</body>");
}
else if(password.isEmpty())
{
out.println("<body style='background-color:black;'>");
out.println("<center>");
out.print("<h4><font size=50px color=red>Username or password field cannot be
null!</font></h4>");
out.println("<form action='Cookies' method='GET'");
out.println("<input type='submit' value='Try again'></br></br>");
out.println("</form>");
out.println("</center>");
out.println("</body>");
}
else
{
out.println("<body style='background-color:red;'>");
out.println("<center>");
out.print("<h4><font size=50px color=black>Invalid,please try again</font></h4>");
out.println("<form action='Cookies' method='GET'");
out.println("<input type='submit' value='Try again'></br></br>");
out.println("</form>");
}

```



```

        out.println("</center>");
        out.println("</body>");
    }
    out.close();
}
}
}

```

### **visitors.java**

```

package programs;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class visitors extends HttpServlet {
    private int count;
    public void init() throws ServletException
    {
        count=0;
    }
    public void doGet(HttpServletRequest request,HttpServletResponse response)throws
    ServletException,IOException
    {
        PrintWriter out=response.getWriter();
        String initialCount=String.valueOf(count);
        Cookie ck=new Cookie("noOfVisit",initialCount);
        response.addCookie(ck);
        int cookieVal=Integer.parseInt(ck.getValue());
        if(cookieVal==1)
        {
            out.println("Welcome");
        }
        else
        {
            out.println("<h2>The servlet has been accessed "+ count +" times ! ");
        }
        count++; } }

```

OUTPUT:

# WELCOME TO HOME PAGE

NEXT

## LOGIN FORM

Username

Password

LOGIN

**Username or password field cannot be null!**

## **LOGIN FORM**

**Username**

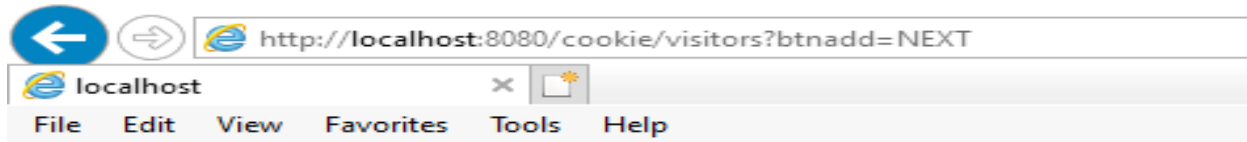
 x

**Password**

LOGIN

**Login ID and password is valid**

NEXT



## LOGIN FORM

Username

Password

LOGIN

Invalid, please try again

## EXPERIMENT-9

**AIM: Write a program to make a jdbc servlet connectivity.**

### THEORY:

Servlet with JDBC, JDBC is an application programming interface (JDBC API) that characterizes an arrangement of standard operations for connecting with DBMS. The DBMSs might be situated on a remote machine associated with the Internet. Keeping in mind the end goal to get to a database under a particular DBMS, for instance, PostgreSQL, one must have a driver for that DBMS and the driver must actualize JDBC API.

The term JDBC stands for **Java Database Connectivity**. JDBC is a standard Java API for database and used for connecting the wide range database and the Java programming language. In servlet connecting to the database is an important task why because while dealing with big projects more databases will be used. All the databases will support, but the class should be mentioned in the code and username and password of the database are needed. The following task can be done using the JDBC library class.

- By using JDBC, one can connect to a database.
- One can create SQL or MYSQL statements using JDBC.
- The records can be viewed and modified using JDBC.
- The SQL or MYSQL queries can be executed in the database using JDBC.

**Servlet** is a Java program which exists and executes in the J2EE servers and is used to receive the `HTTP` protocol request, process it and send back the response to the client.

Servlets make use of the Java standard extension classes in the packages `javax.servlet` and `javax.servlet.http`. Since Servlets are written in the highly portable Java language and follow a standard framework, they provide a means to create the sophisticated server extensions in a server and operating system in an independent way.

### CODE:

#### Register.java

```
import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;
```

```
import java.sql.SQLException;

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(urlPatterns = {"/Register"})

public class Register extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

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

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

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

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

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

            Class.forName("org.apache.derby.jdbc.ClientDriver");

            Connection conn =

            DriverManager.getConnection("jdbc:derby://localhost:1527/Student","root","root");

            PreparedStatement stat = conn.prepareCall("insert into jstudent(name,email,pass)

            values(?,?,?)");

            stat.setString(1,name);
```

```
stat.setString(2,email);

stat.setString(3,password);

int val = stat.executeUpdate();

if(val>0)

    out.println("<h3 style=\"color: green; margin-left: 25px;\">User Registered
Successfully</h3>");

else

    out.println("<h3 style=\"color: coral; margin-left: 25px;\">Registration
Failed</h3>");

request.getRequestDispatcher("index.html").include(request, response);

} catch (ClassNotFoundException | SQLException ex) {

    ex.printStackTrace();

}

}

@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";

    }

}
```

## **index.html**

```
<!DOCTYPE html>

<html>

    <head>

        <title>Registration Form </title>

        <meta charset="UTF-8">

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

    </head>

    <body style="margin: 0; padding: 0; font-family: verdana; background: powderblue; font-size: 10pt;">

        <form action="Register" method = "POST" style="text-align: left">

            <table cellpadding="5" style="font-family: verdana; font-size: 10pt;">
```



```
<tr>

<td><b>Name : </b></td>

<td><input name="name" type="text" style="width: 300px;" required/></td>

</tr>

<tr>

<td><b>Email ID : </b></td>

<td><input name="email" type="text" style="width: 300px;" required/></td>

</tr>

<tr>

<td><b>Password : </b></td>

<td><input name="password" type="password" style="width: 300px;"
required/></td>

</tr>

<tr>

<td></td>

<td><input type="submit" value="REGISTER"/></td>

</tr>

</table>

</form>

</body></html>
```

**OUTPUT:**

**User Registered Successfully**

**Name :**

**Email ID :**

**Password :**

#	NAME	EMAIL	PASS
1	anchal	anchal@gmail.com	1234
2	sheetal	sheetal@gmail.com	5678
3	rohit	rohit@gmail.com	4455
4	dabbu	dabbu@gmail.com	7766
5	abhi	abhi@gmail.com	7895

## EXPERIMENT-10

**AIM: Write a program to**

1. To print 10 number using scriptlet
2. To accept form values and print the form values using jsp
3. To handle exception
4. To calculate salary
5. To handle error

### THEORY:

**JSP** technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc. A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

### Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

#### 1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

#### 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

#### 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

#### 4) Less code than Servlet

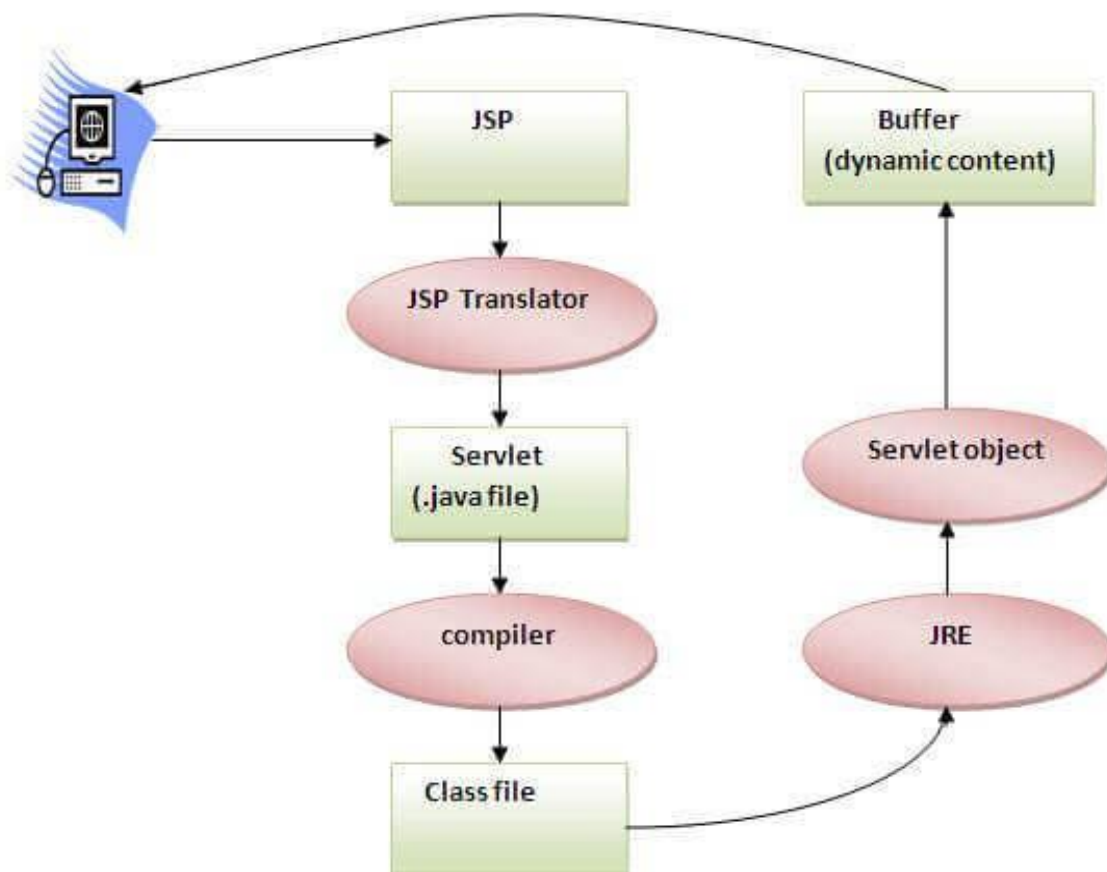
In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

## The Lifecycle of a JSP Page

The JSP pages follow these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization ( the container invokes jspInit() method).
- Request processing ( the container invokes \_jspService() method).
- Destroy ( the container invokes jspDestroy() method).

Note: jspInit(), \_jspService() and jspDestroy() are the life cycle methods of JSP.



## CODE:

### index.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>

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

            <%

                out.print("<b>First 10 numbers are:</><br>");

                for(int i=1;i<=10;i++)

                {

                    out.print("<br><b>"+i+"</b>");

                }

            %>

            <br><br>

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

        </form>

    </body></html>
```

## Calculate\_Sal.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>
    <form method="post" action="other.jsp">
      <p>Employee ID</p>
      <input type="text" name="empid" required/>
      <br>
      <p>First Name</p>
      <input type="text" name="firstname" required/>
      <br>
      <p>Last Name</p>
      <input type="text" name="lastname" required/>
      <br>
      <p>Designation</p>
      <input type="text" name="designation" required/>
      <br>
      <p>CTC</p>
      <input type="text" name="CTC" required/>
      <br>
      <p>Tax</p>
      <input type="text" name="Tax" required/>
      <br>
      <p>Other Charges</p>
      <input type="text" name="OC" required/>
      <br><br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

## other.jsp

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

<% @page errorPage="error.jsp" %>

<%!
```

```
public int calculate(int sal,int tax,int oc)

{

if(sal<=0)

{

throw new IllegalArgumentException("Salary should be greater than zero!");

}

return (sal-tax-oc);

}

%>

<!DOCTYPE html>

<html>

    <head>

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

        <title>Salary</title>

    </head>

    <body>

        <%

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

String fname=request.getParameter("firstname");

String lname=request.getParameter("lastname");
```

```
String desig=request.getParameter("designation");

String salary=request.getParameter("CTC");

String tax=request.getParameter("Tax");

String oc=request.getParameter("OC");

int i=Integer.parseInt(salary);

int j=Integer.parseInt(tax);

int k=Integer.parseInt(oc);

calculate(i,j,k);

out.println("Welcome "+fname+" "+lname);

out.println("<br><br>Designation:          "+desig+"<br><br>Tax          Added:
"+tax+"<br><br>Other Charges: "+oc);

out.println("<br><br>Your total salary is: "+calculate(i,j,k));

%>

</body>

</html>
```

### **error.jsp**

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

<% @page language="Java" %>

<!DOCTYPE html>

<% @ page isErrorPage="true" %>
```



```
<html>

<head>

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

    <title>JSP Page</title>

</head>

<body style="font-family:verdana;font-size:30px;">

    <p style="color: red">

        <b><%=exception.getMessage()%></b>

    </p>

</body>

</html>
```

## OUTPUT:

To print 10 number using scriptlet

---

**First 10 numbers are:**

**1  
2  
3  
4  
5  
6  
7  
8  
9  
10**

**Submit**

To accept form values and print the form values using jsp

Employee ID

101

First Name

ANCHAL

Last Name

SINGH

Designation

Software engineer

CTC

2000000

Tax

10

Other Charges

5

Submit

To calculate salary

Welcome ANCHAL SINGH

Designation: Software engineer

Tax Added: 10

Other Charges: 5

Your total salary is: 1999985

To handle exception

---

Employee ID

101

First Name

ANCHAL

Last Name

|

This is a required field

Software engineer

CTC

2000000

Tax

10

Other Charges

5

Submit

**Employee ID**

|

This is a required field

**Last Name**

**Designation**

**CTC**

**Tax**

**Other Charges**

**Submit**

### Error handling

Employee ID

First Name

Last Name

Designation

CTC

Tax

Other Charges

**Salary should be greater than zero!**

## EXPERIMENT-11

**AIM: write a program to implement stateless session beans**

### Theory:

**Stateless Session bean** is a business object that represents business logic only. It doesn't have state (data).

In other words, conversational state between multiple method calls is not maintained by the container in case of stateless session bean.

The stateless bean objects are pooled by the EJB container to service the request on demand.

It can be accessed by one client at a time. In case of concurrent access, EJB container routes each request to different instance.

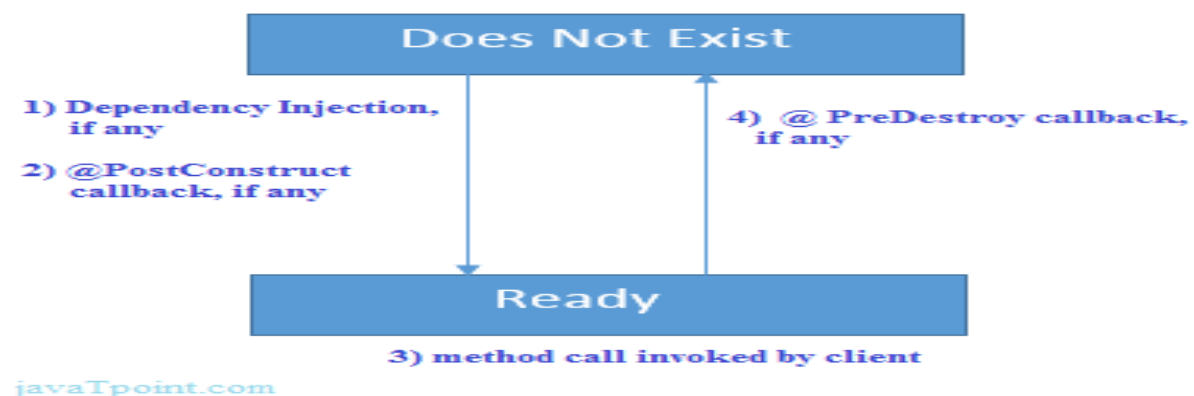
### Annotations used in Stateless Session Bean

There are 3 important annotations used in stateless session bean:

1. @Stateless
2. @PostConstruct
3. @PreDestroy

### Life cycle of Stateless Session Bean

There is only two states of stateless session bean: does not exist and ready. It is explained by the figure given below.



EJB Container creates and maintains a pool of session bean first. It injects the dependency if then calls the @PostConstruct method if any. Now actual business logic method is invoked by the client. Then, container calls @PreDestory method if any. Now bean is ready for garbage collection.

## **Steps to Create a Stateless EJB**

Following are the steps required to create a stateless EJB –

- Create a remote/local interface exposing the business methods.
- This interface will be used by the EJB client application.
- Use @Local annotation, if EJB client is in same environment where EJB session bean is to be deployed.
- Use @Remote annotation, if EJB client is in different environment where EJB session bean is to be deployed.
- Create a stateless session bean, implementing the above interface.
- Use @Stateless annotation to signify it a stateless bean. EJB Container automatically creates the relevant configurations or interfaces required by reading this annotation during deployment.

## **CODE:**

### **NewSessionBean.java**

```
package working;
import javax.ejb.Stateless;
@Stateless
public class NewSessionBean implements NewSessionBeanLocal {
    @Override
    public float add(float x, float y) {
        return (x+y);
    }
    @Override
    public float subtract(float x, float y) {
        return (x-y);
    }
    @Override
    public float multiply(float x, float y) {
        return (x*y);
    }
    @Override
    public float division(float x, float y) {
```

```

        return (x/y);
    }
}

```

### NewSessionBeanLocal.java

```

package working;
import javax.ejb.Local;
@Local
public interface NewSessionBeanLocal {
    float add(float x, float y);
    float subtract(float x, float y);
    float multiply(float x, float y);
    float division(float x, float y);
}

```

### index.html

```

<!DOCTYPE html>
<html>
<head>
    <title>Stateless Session Bean</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body style="background-color: pink; font-family: verdana; font-size: 12px">
    <h3>Welcome to the Stateless Session Bean Calculator</h3>
    <br><br>
    <form action="calculator" style="text-align: left;">
        <table>
            <tr>
                <th>Enter first number : </th>
                <td><input name="num1" required="required" type="text"/></td>
            </tr>
            <tr>
                <td></td><td></td></tr>
            <tr>
                <th>Enter second number : </th>
                <td><input name="num2" required="required" type="text"/></td>
            </tr>
            <tr>
                <td></td><td></td></tr>
            <tr>
                <th>Choose Operation : </th>
                <td><input type="radio" name="group1" value ="add">Addition<br>
                    <input type="radio" name="group1" value ="sub">Subtraction<br>
                    <input type="radio" name="group1" value ="multi">Multiplication<br>
                    <input type="radio" name="group1" value ="div">Division<br>

```



```

        </td>
    </tr>
    <tr><td></td><td></td></tr>
    <tr>
        <td></td>
        <td><input type="submit" name="submit" value ="Submit"></td>
    </tr>
</table>
</form>
</body>
</html>

```

### calculator.java

```

package working;
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;
@WebServlet(name = "calculator", urlPatterns = {"/calculator"})
public class calculator extends HttpServlet {
    @EJB
    private NewSessionBeanLocal newSessionBean;
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            String opr = request.getParameter("group1");
            int num1 = Integer.parseInt(request.getParameter("num1"));
            int num2 = Integer.parseInt(request.getParameter("num2"));
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet calculator</title>");
            out.println("</head>");
            out.println("<body>");
            if(opr.equals("add"))
            {
                float ans = newSessionBean.add(num1, num2);
                out.println("<h3> Result Addition : " + ans + "</h1>");
            }
        }
    }
}

```

```
    }
    else if(opr.equals("sub"))
    {
        float ans = newSessionBean.subtract(num1, num2);
        out.println("<h3> Result Subtraction : " + ans + "</h1>");
    }
    else if(opr.equals("multi"))
    {
        float ans = newSessionBean.multiply(num1, num2);
        out.println("<h3> Result Multiply : " + ans + "</h1>");
    }
    else if(opr.equals("div"))
    {
        float ans = newSessionBean.division(num1, num2);
        out.println("<h3> Result Division : " + ans + "</h1>");
    }
    out.println("</body>");
    out.println("</html>");
}

}

@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:

### Addition

**Welcome to the Stateless Session Bean Calculator**

Enter first number :

Enter second number :

Choose Operation :   
☒ Addition  
☐ Subtraction  
☐ Multiplication  
☐ Division

**Result Addition : 20.0**

### Subtraction

**Welcome to the Stateless Session Bean Calculator**

Enter first number :

Enter second number :

Choose Operation :   
☐ Addition  
☒ Subtraction  
☐ Multiplication  
☐ Division

**Result Subtraction : 10.0**

## Multiplication

**Welcome to the Stateless Session Bean Calculator**

Enter first number :

Enter second number :

Choose Operation :   
☐ Addition  
☐ Subtraction  
☒ Multiplication  
☐ Division

**Result Multiply : 75.0**

## Division

**Welcome to the Stateless Session Bean Calculator**

Enter first number :

Enter second number :

Choose Operation :   
☐ Addition  
☐ Subtraction  
☐ Multiplication  
☒ Division

**Result Division : 3.0**

## EXPERIMENT-12

### AIM: Write a program to implement stateful session beans

#### Theory:

**Stateful Session bean** is a business object that represents business logic like stateless session bean. But, it maintains state (data).

In other words, conversational state between multiple method calls is maintained by the container in stateful session bean.

A stateful session bean is a type of enterprise bean, which preserve the conversational state with client. A stateful session bean as per its name keeps associated client state in its instance variables. EJB Container creates a separate stateful session bean to process client's each request. As soon as request scope is over, stateful session bean is destroyed.

#### Annotations used in Stateful Session Bean

There are 5 important annotations used in stateful session bean:

1. @Stateful
2. @PostConstruct
3. @PreDestroy
4. @PrePassivate
5. @PostActivate

#### Steps to Create Stateful EJB

Following are the steps required to create a stateful EJB –

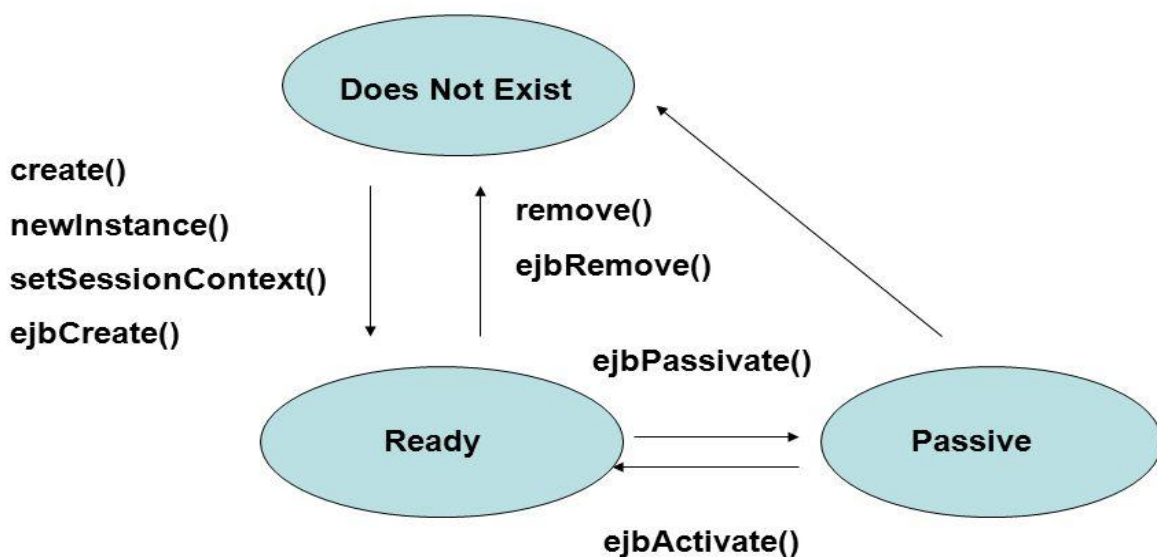
- Create a remote/local interface exposing the business methods.
- This interface will be used by the EJB client application.
- Use @Local annotation if EJB client is in same environment where EJB session bean need to be deployed.
- Use @Remote annotation if EJB client is in different environment where EJB session bean need to be deployed.
- Create a stateful session bean, implementing the above interface.

- Use @Stateful annotation to signify it a stateful bean. EJB Container automatically creates the relevant configurations or interfaces required by reading this annotation during deployment.

## Life cycle of Stateless Session Bean

# Stateful Session Bean

- Life Cycle of Stateful Session Bean



## CODE:

### BankTransaction.java

```
package working;
```

```
import javax.ejb.Stateful;
```

```
@Stateful
```

```
public class BankTransaction implements BankTransactionLocal {
```

```
    int bal = 500;
```

```
    @Override
```

```
    public int deposit(int amount) {
```

```
        bal = bal + amount;
```

```
        return bal;
```

```
    }
```

```
    @Override
```

```
    public int withdraw(int amount) {
```

```
        bal = bal - amount;
```

```
        return bal;
```

```
    }
```

```
    @Override
```

```
    public int getbal() {
```

```
        return bal;
```

```
    }
```

```
}
```

**BankTransactionLocal.java**

```
package working;

import javax.ejb.Local;

@Local

public interface BankTransactionLocal {

    int deposit(int amount);

    int withdraw(int amount);

    int getbal();

}
```

### **index.html**

```
<!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>Stateful Bank</title>
```

```
        <meta charset="UTF-8">
```

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



</head>

<body style="background-color:yellow; font-family: verdana; font-size: 12px">

<h3>Welcome to the Stateful Session Bean Bank Application</h3>

<br><br>

<form action="sbank">

<table>

<tr>

<th>Enter Amount : </th>

<td><input name="amount" required="required" type="text"/></td>

</tr>

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

<tr>

<th>Choose Operation : </th>

<td><input type="radio" name="group1" value ="deposit">Deposit<br>

<input type="radio" name="group1" value ="withdraw">Withdraw<br>

</td>

</tr>

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

<tr>

<td></td>

```
<td><input type="submit" name="submit" value ="Submit"></td>

</tr>

</table>

</form>

</body>

</html>
```

### **sbank.java**

```
package working;

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;

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

public class sbank extends HttpServlet {

    BankTransactionLocal bankTransaction = lookupBankTransactionLocal();

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

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

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

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

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

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

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

            out.println("<html>");

            out.println("<head>");

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

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

            out.println("<body>");

            if(opr.equals("deposit"))
```

```
{

    int finalbal = bankTransaction.deposit(amount);

    out.println("<h3> Amount Rs " + amount + " deposited successfully.\n Available
Balance - Rs " + finalbal + "</h3>");

}

else if(opr.equals("withdraw"))

{

    int availbal = bankTransaction.getbal();

    if(availbal>amount)

    {

        int finalbal = bankTransaction.withdraw(amount);

        out.println("<h3> Amount Rs " + amount + " withdrawn successfully!\n Available
Balance - Rs " + finalbal + "</h3>");

    }

    else

    {

        out.println("<h3> Amount Rs " + amount + " cannot be withdrawn as the available
Balance - Rs " + availbal + "</h3>");

    }

}

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 `POST` 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>

    private BankTransactionLocal lookupBankTransactionLocal() {

    try {

        Context c = new InitialContext();

        return (BankTransactionLocal) c.lookup("java:global/StatefulBank/StatefulBank-
ejb/BankTransaction!working.BankTransactionLocal");

    } catch (NamingException ne) {

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

        throw new RuntimeException(ne);

    }

    }

    }
```

## OUTPUT:

### For deposit



**Welcome to the Stateful Session Bean Bank Application**

Enter Amount :

Choose Operation : ☒ Deposit ☐ Withdraw

---

**Amount Rs 3000 deposited successfully. Available Balance - Rs 4500**

**For withdraw**

**Welcome to the Stateful Session Bean Bank Application**

**Enter Amount :**

**Choose Operation :** ☐ Deposit  
☒ Withdraw

**Amount Rs 4000 withdrawn successfully! Available Balance - Rs 500**



## EXPERIMENT-13

### AIM: Write a program to implement EJB

**Theory:** EJB is an acronym for enterprise java bean. It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.

To get information about distributed applications, visit RMI Tutorial first.

To run EJB application, you need an application server (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc.

It performs:

- a. life cycle management,
- b. security,
- c. transaction management, and
- d. object pooling.

EJB application is deployed on the server, so it is called server side component also.

EJB is like COM (Component Object Model) provided by Microsoft. But, it is different from Java Bean, RMI and Web Services.

### When use Enterprise Java Bean?

1. **Application needs Remote Access.** In other words, it is distributed.
2. **Application needs to be scalable.** EJB applications supports load balancing, clustering and fail-over.
3. **Application needs encapsulated business logic.** EJB application is separated from presentation and persistent layer

### Types of Enterprise Java Bean

There are 3 types of enterprise bean in java.

#### Session Bean

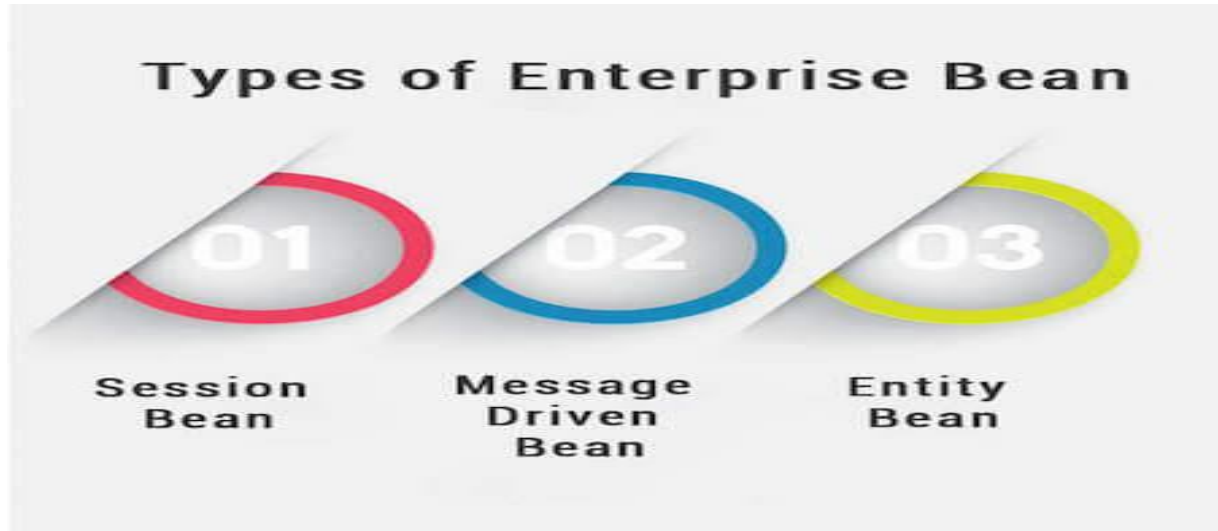
Session bean contains business logic that can be invoked by local, remote or webservice client.

#### Message Driven Bean

Like Session Bean, it contains the business logic but it is invoked by passing message.

## Entity Bean

It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).



## Difference between RMI and EJB

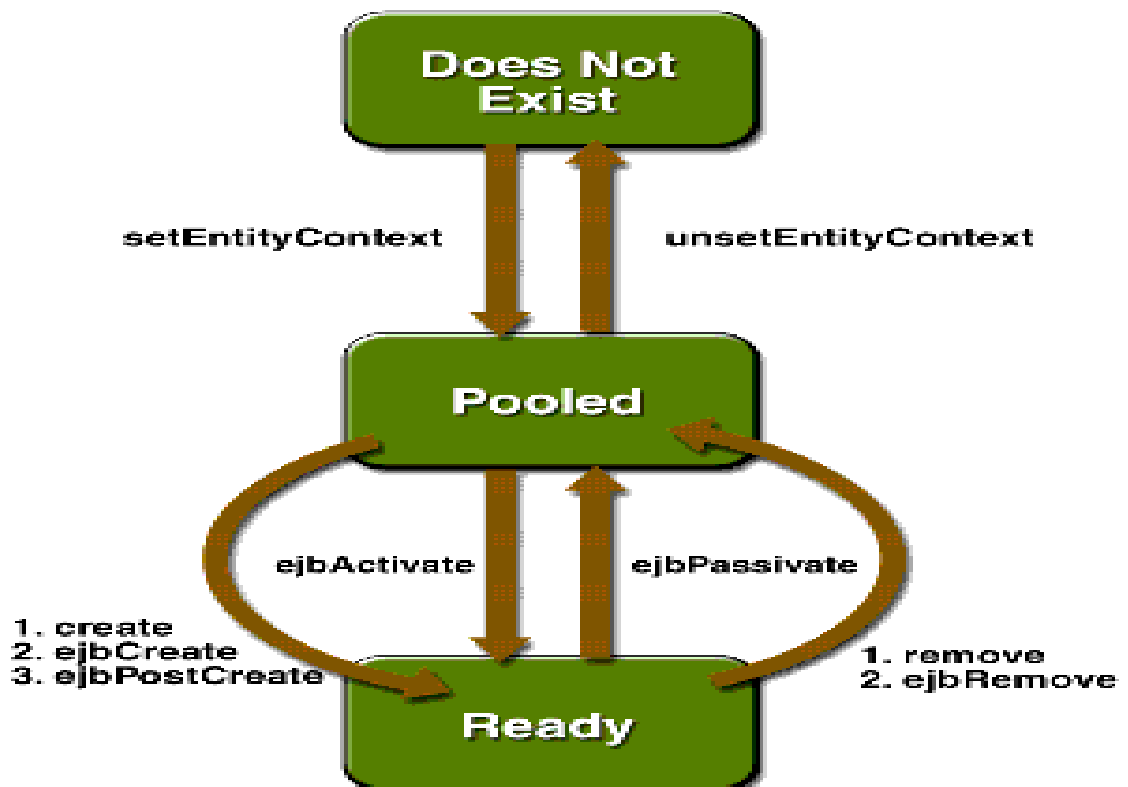
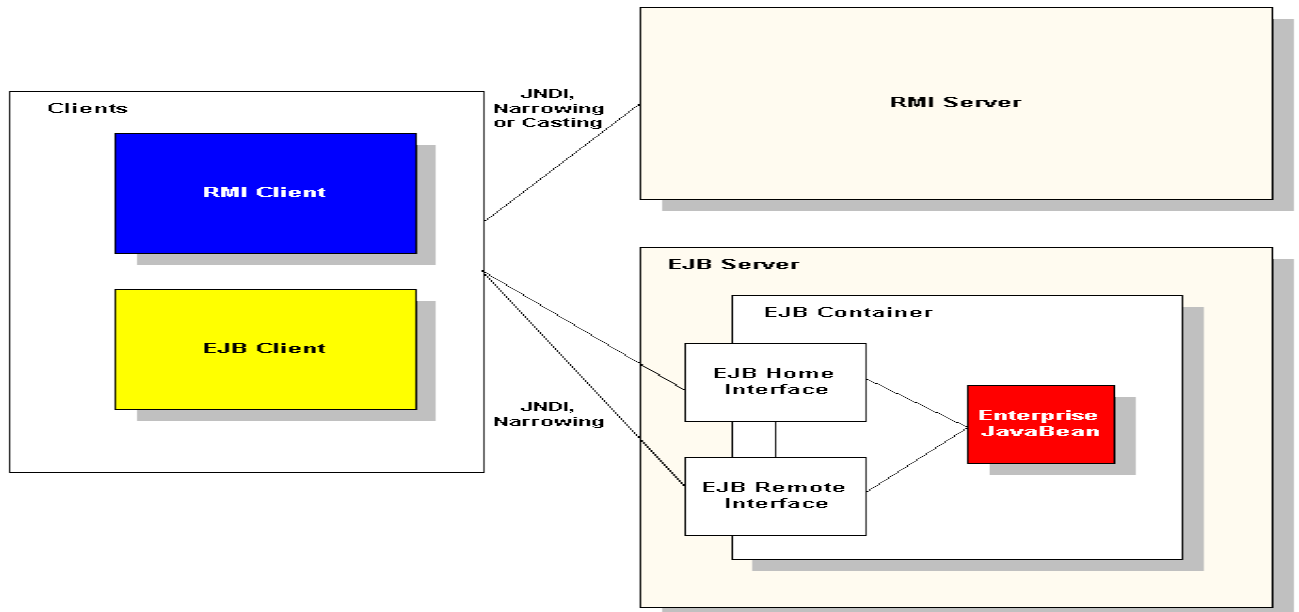
Both RMI and EJB, provides services to access an object running in another JVM (known as remote object) from another JVM. The differences between RMI and EJB are given below:

RMI	EJB
In RMI, middleware services such as security, transaction management, object pooling etc. need to be done by the java programmer.	In EJB, middleware services are provided by EJB Container automatically.
RMI is not a server-side component. It is not required to be deployed on the server.	EJB is a server-side component, it is required to be deployed on the server.
RMI is built on the top of socket programming.	EJB technology is built on the top of RMI.

## Disadvantages of EJB

1. Requires application server

2. Requires only java client. For other language client, you need to go for webservice.
3. Complex to understand and develop ejb applications.



CODE:

## index.jsp

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

<!DOCTYPE html>

<html>

  <head>

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

    <title>Entity Bean Example</title>

  </head>

  <body bgcolor="pink">

    <center>

      <h1>Entity Bean</h1><br>

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

        Type name:-<input type="text" name="name" value="" size="40"/><br>

        salary:-<input type="text" name="salary" value="" size="20"/><br>

        <input type="submit" value="Add Fields" name="add"/><br>

      </form>

    </center>

  </body>

</html>
```

## Employee.java

```
package EntityBeanEx;

import java.io.Serializable;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Employee implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private int salary;

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getSalary() {

        return salary;

    }

}
```

```
public void setSalary(int salary) {  
    this.salary = salary;  
}  
  
public Long getId() {  
    return id;  
}  
  
public void setId(Long id) {  
    this.id = id;  
}  
  
@Override  
public int hashCode() {  
    int hash = 0;  
    hash += (id != null ? id.hashCode() : 0);  
    return hash;  
}  
  
@Override  
public boolean equals(Object object) {  
    if (!(object instanceof Employee)) {  
        return false;  
    }  
  
    Employee other = (Employee) object;
```

```
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {  
            return false;  
        }  
        return true;  
    }  
}
```

**@Override**

```
public String toString() {  
    return "EntityBeanEx.Employee[ id=" + id + " ]";  
}
```

```
}
```

### **EmployeeFacade.java**

```
package EntityBeanEx;
```

```
import javax.ejb.Stateless;
```

```
import javax.persistence.EntityManager;
```

```
import javax.persistence.PersistenceContext;
```

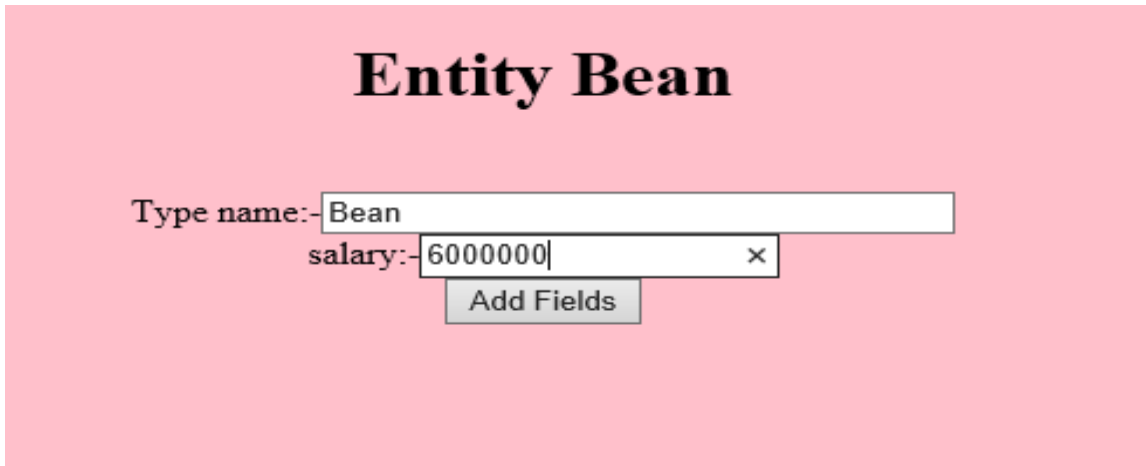
**@Stateless**

```
public class EmployeeFacade extends AbstractFacade<Employee> implements  
EmployeeFacadeLocal {
```

```
    @PersistenceContext(unitName = "EntityBeanEx-ejbPU")
```

```
private EntityManager em;  
  
@Override  
  
protected EntityManager getEntityManager() {  
  
    return em;  
  
}  
  
public EmployeeFacade() {  
  
    super(Employee.class);  
  
}  
  
}
```

## OUTPUT



The screenshot shows a web form titled "Entity Bean" on a pink background. It contains two input fields: "Type name:-" with the value "Bean" and "salary:-" with the value "6000000". There is a small "x" icon in the salary field. Below the salary field is a button labeled "Add Fields".

**You Entered your detail Mr. Beansuccessfully.**



## Entity Bean

Type name:-

salary:-

**You Entered your detail Mr. Singh successfully.**

#	ID	NAME	SALARY
1		1 anchal	300000
2		2 anchal	300000
3		3 sheetal	300000
4		4 Rohit	600000
5		5 shiddhart	600000
6		6 abhi	70000
7		7 Anchal	70000
8		8 Bean	700000
9		9 Anchal	700000
10		10 ak	5000000
11		11 Bean	6000000
12		12 Singh	7000000

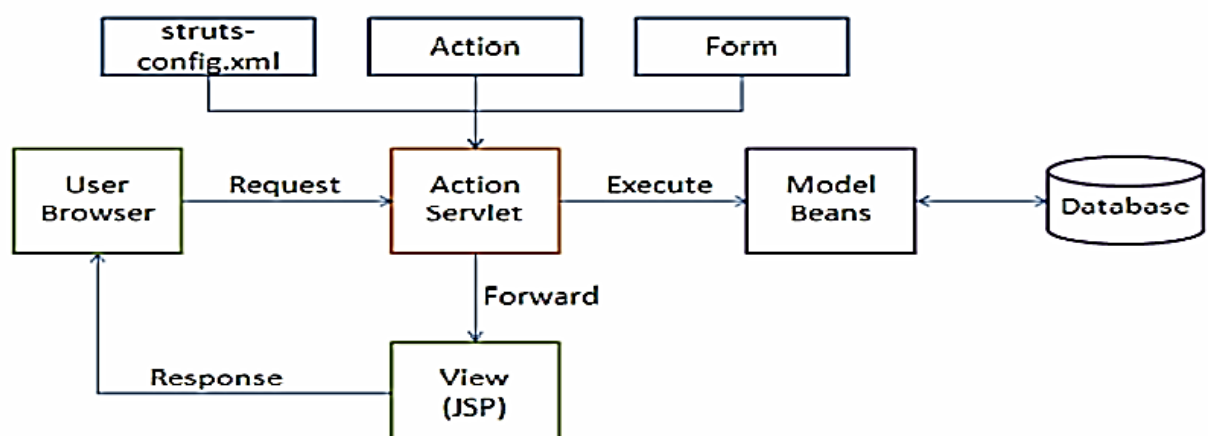
## EXPERIMENT-14

**AIM: Write a program to implement struts**

### Theory:

- ☐ Apache Struts is a free open-source framework for creating Java web applications.
- ☐ Web applications differ from conventional websites in that web applications can create a dynamic response.
- ☐ Many websites deliver only static pages. A web application can interact with databases and business logic engines to customize a response.
- ☐ Web applications based on JavaServer Pages sometimes can mingle database code, page design code, and control flow code.
- ☐ In practice, these concerns are to be separated, larger applications become difficult to maintain.
- ☐ One way to separate concerns in a software application is to use a Model-ViewController (MVC) architecture. The Model represents the business or database code, the View represents the page design code, and the Controller represents the navigational code.

### STRUTS ARCHITECTURE



❖ **STRUTS CLASSES/FILES Configuration Files:**

**1. web.xml:** Servlet mapping has to be configured in web.xml. Mapping between URLPattern and Action Servlet

**2. struts.config.xml:** Struts related configuration is done through struts-config.xml

### ❖ **Controller in Struts Framework:**

**ActionServlet:** Receives the request

- ☐ Struts framework provides a built-in base servlet org.apache.struts.action.ActionServlet
- ☐ It extends HttpServlet
- ☐ RequestProcessor: Helper Class (Actual Processing)
- ☐ Action Mapping: It defines the mapping between request URI of incoming requests to specific Action class-struts.cnfig file
- ☐ Display Pages : Success & Failure Pages

### **ActionForm**

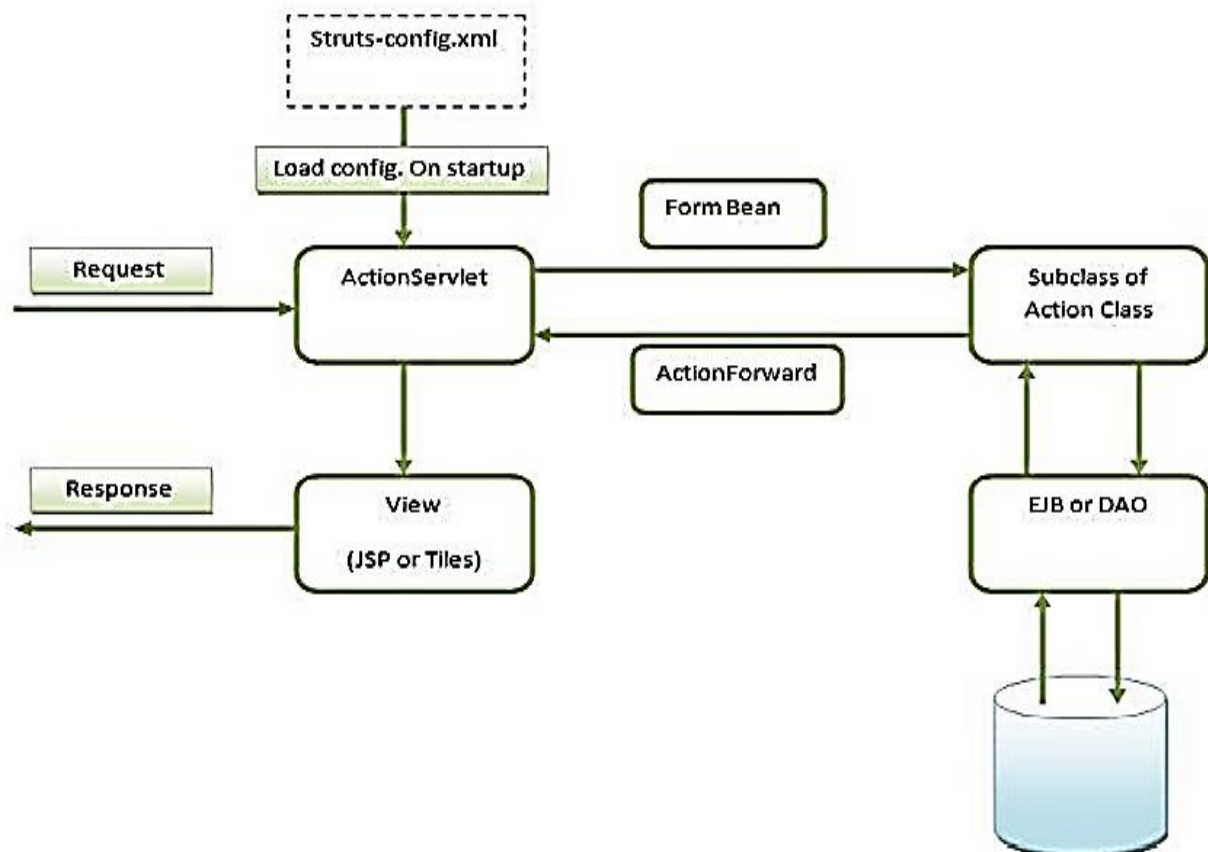
This is Java class which receives and validate input data from a HTML form before sending to Action class.

- ☐ All ActionForm class are also extended from org.apache.struts.action.ActionForm.
- ☐ ActionForm is like Javabean, its scopes are only session and request .
- ☐ ActionForm is classified 2 types, which are StaticForm (extends form ActionForm) and DynamicForm (existed Form)

## Action

- ❑ Action is Java class processing request and response by calling JavaObject or Model.
- ❑ Each action need to be mapped in struts-config.xml. One of necessary method needed to define in Action class is execute() method.
- ❑ The method have responsibility for processing business logic.
- ❑ The method will return a ActionForward instance which determines next processing component for request.

## Struts-Flow of Exceution



**CODE:**

**login.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 bgcolor="pink">

        <h1>Struts Login Form!!!!</h1>

        <form action="loginform.do" method="post">

            Username <input type="text" name="uname"/><br/>

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

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

        </form>

    </body>

</html>
```

## **failure.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 bgcolor="lightgreen">

        <h1 style="color:red"> Uppss..... YOU Enter Wrong Credentials!!!</h1>

    </body>

</html>
```

## **success.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 bgcolor="yellow">
```

```
<h1 style="color:magenta"> YOU HAVE SUCCESSFULLY LOGGED IN!!!!</h1>
```

```
</body>
```

```
</html>
```

## **loginbean.java**

```
package com.myapp.struts;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import org.apache.struts.action.ActionErrors;
```

```
import org.apache.struts.action.ActionMapping;
```

```
import org.apache.struts.action.ActionMessage;
```

```
public class loginbean extends org.apache.struts.action.ActionForm {
```

```
    private String  uname;
```

```
    private String upass;
```

```
    public String getUname() {
```

```
        return uname;
```

```
    }
```

```
    public void setUname(String uname) {
```

```
        this.uname = uname;
```

```
    public String getUpass() {
```

```
return upass;
```

```
}
```

```
public void setUpass(String upass) {
```

```
    this.upass = upass;
```

```
}
```

```
public loginbean() {
```

```
    super();
```

```
}
```

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
```

```
    ActionErrors errors = new ActionErrors();
```

```
    return errors;
```

```
}
```

```
}
```

### **loginform.java**

```
package com.myapp.struts;
```

```
import javax.servlet.http.HttpServletRequest;
```

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

```
import org.apache.struts.action.ActionForm;
```



```
import org.apache.struts.action.ActionForward;

import org.apache.struts.action.ActionMapping;

public class loginform extends org.apache.struts.action.Action {

    private static final String SUCCESS = "success";

    private static final String FAILURE = "failure";

    @Override

    public ActionForward execute(ActionMapping mapping, ActionForm form,

        HttpServletRequest request, HttpServletResponse response)

        throws Exception {

        loginbean lb = (loginbean)form;

        if(lb.getUserName().equals("Anchal")&&lb.getUpass().equals("anchal"))

            return mapping.findForward(SUCCESS);

        else

            return mapping.findForward(FAILURE);

    }}

```

### **struts-config.xml**

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

<!DOCTYPE struts-config PUBLIC

    "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"

    "http://jakarta.apache.org/struts/dtds/struts-config_1_3.dtd">

<struts-config>

```

```
<form-beans>

    <form-bean name="loginbean" type="com.myapp.struts.loginbean"/>

</form-beans>

<global-exceptions>

</global-exceptions>

<global-forwards>

    <forward name="failure" path="/failure.jsp"/>

    <forward name="success" path="/success.jsp"/>

    <forward name="welcome" path="/Welcome.do"/>

</global-forwards>

<action-mappings>

    <action          name="loginbean"          path="/loginform"          scope="request"
type="com.myapp.struts.loginform" validate="false"/>

    <action path="/Welcome" forward="/welcomeStruts.jsp"/>

</action-mappings>

<controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>

<message-resources parameter="com/myapp/struts/ApplicationResource"/>

<plug-in className="org.apache.struts.tiles.TilesPlugin" >

    <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />

    <set-property property="moduleAware" value="true" />

</plug-in>
```

```
<!-- ===== Validator plugin
===== -->

<plug-in className="org.apache.struts.validator.ValidatorPlugIn">

    <set-property

        property="pathnames"

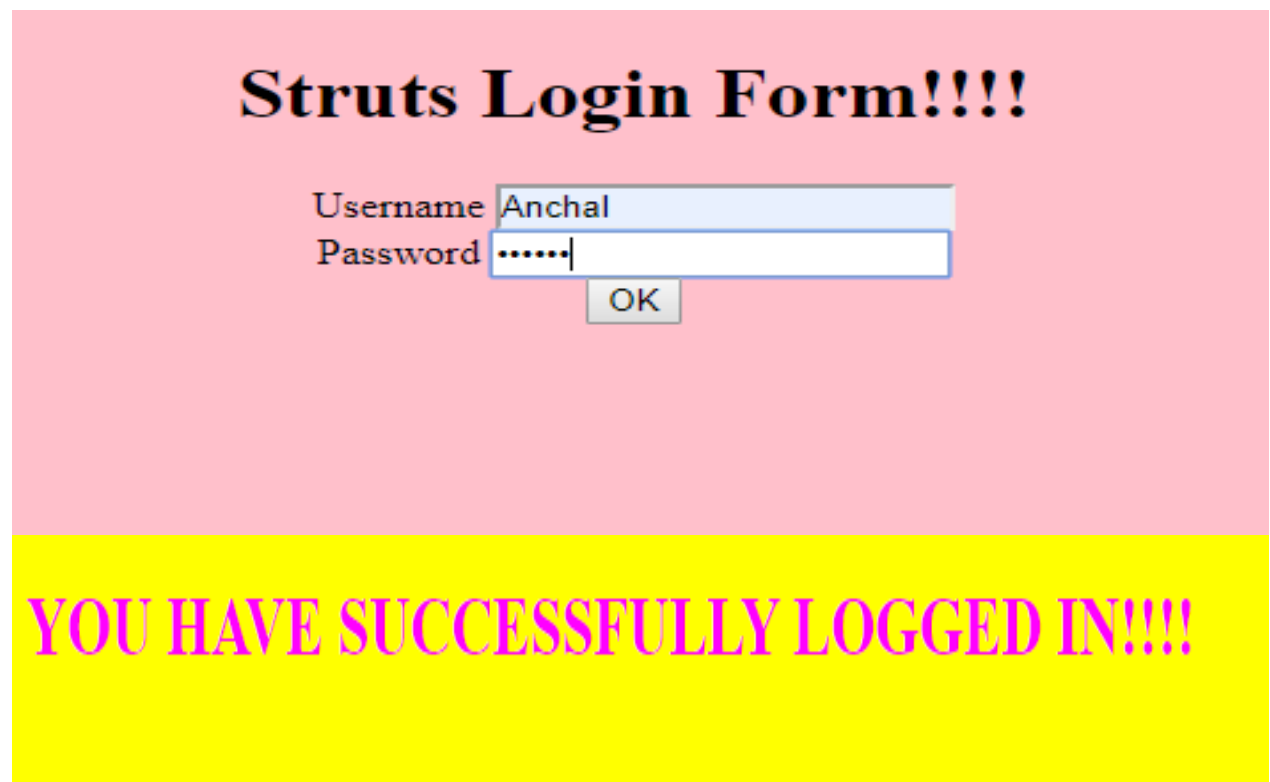
        value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>

</plug-in>

</struts-config>
```

## OUTPUT:

### SUCCESS PAGE



The screenshot displays a web application interface. At the top, a pink banner contains the text "Struts Login Form!!!!" in a large, bold, black serif font. Below this, there is a login form with two input fields: "Username" containing the text "Anchal" and "Password" containing six dots. A grey "OK" button is positioned below the password field. At the bottom of the screenshot, a yellow banner displays the text "YOU HAVE SUCCESSFULLY LOGGED IN!!!!" in a large, bold, magenta serif font.

**FAILUARE PAGE**

# Struts Login Form!!!!

Username	<input type="text" value="nchal"/>
Password	<input type="password" value="..."/>
<input type="button" value="OK"/>	

**Uppss..... YOU Enter Wrong Credentials!!!**

## EXPERIMENT-15

**AIM: Write a program to implement javaMail API**

### Theory:

The **JavaMail** is an API that is used to compose, write and read electronic messages (emails). The JavaMail API provides protocol-independent and platform-independent framework for sending and receiving mails.

The **javax.mail** and **javax.mail.activation** packages contains the core classes of JavaMail API. The JavaMail facility can be applied to many events. It can be used at the time of registering the user (sending notification such as thanks for your interest to my site), forgot password (sending password to the users email id), sending notifications for important updates etc. So there can be various usage of java mail api.

### Protocols used in JavaMail API

There are some protocols that are used in JavaMail API.

- SMTP
- POP
- IMAP
- MIME
- NNTP and others

### SMTP

SMTP is an acronym for Simple Mail Transfer Protocol. It provides a mechanism to deliver the email. We can use Apache James server, Postcast server, cmail server etc. as an SMTP server. But if we purchase the host space, an SMTP server is by default provided by the host provider. For example, my smtp server is mail.javatpoint.com. If we use the SMTP server provided by the host provider, authentication is required for sending and receiving emails.

### POP

POP is an acronym for Post Office Protocol, also known as POP3. It provides a mechanism to receive the email. It provides support for single mail box for each user. We can use Apache James server, cmail server etc. as an POP server. But if we purchase the host space, an POP server is by default provided by the host provider. For example, the pop server provided by the host provider for my site is mail.javatpoint.com. This protocol is defined in RFC 1939.

## IMAP

IMAP is an acronym for Internet Message Access Protocol. IMAP is an advanced protocol for receiving messages. It provides support for multiple mail box for each user ,in addition to, mailbox can be shared by multiple users. It is defined in RFC 2060.

## MIME

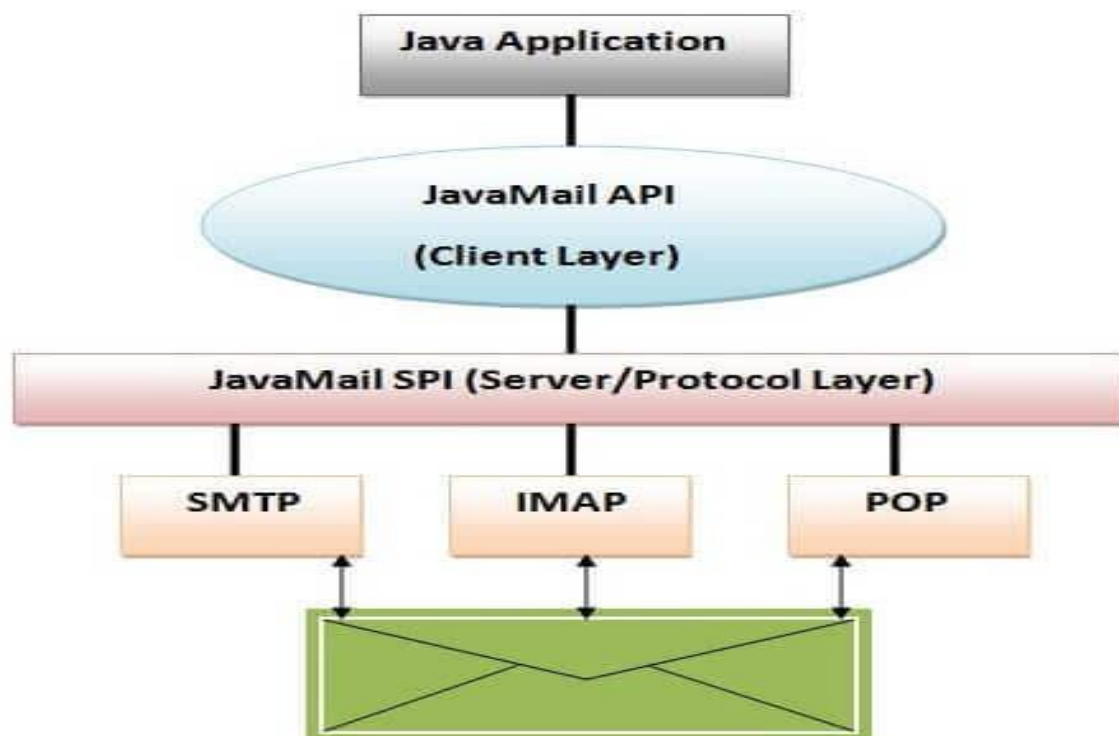
Multiple Internet Mail Extension (MIME) tells the browser what is being sent e.g. attachment, format of the messages etc. It is not known as mail transfer protocol but it is used by your mail program.

## NNTP and Others

There are many protocols that are provided by third-party providers. Some of them are Network News Transfer Protocol (NNTP), Secure Multipurpose Internet Mail Extensions (S/MIME)

## JavaMail Architecture

The java application uses JavaMail API to compose, send and receive emails. The JavaMail API uses SPI (Service Provider Interfaces) that provides the intermediary services to the java application to deal with the different protocols. Let's understand it with the figure given below:



## JavaMail API Core Classes

There are two packages that are used in Java Mail API: javax.mail and javax.mail.internet package. These packages contains many classes for Java Mail API. They are:

- javax.mail.Session class
- javax.mail.Message class
- javax.mail.internet.MimeMessage class
- javax.mail.Address class
- javax.mail.internet.InternetAddress class
- javax.mail.Authenticator class
- javax.mail.PasswordAuthentication class
- javax.mail.Transport class
- javax.mail.Store class
- javax.mail.Folder class etc.

## CODE:

### JavaMail.java

```
package mail;
import mail.JavaMailUtil;
public class JavaMail {
    public static void main(String[] args) throws Exception
    { JavaMailUtil.sendMail("anchalgoldie@gmail.com");
    }
}
```

## **JavaMailUtil.java**

```
package mail;

import com.sun.istack.internal.logging.Logger;

import javax.mail.Session;

import javax.mail.Authenticator;

import java.util.Properties;

import java.util.logging.Level;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Transport;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class JavaMailUtil {

    public static void sendMail(String recipient) throws Exception

    {

        System.out.println("Preparing to send email");

        Properties properties = new Properties();

        properties.put("mail.smtp.auth", "true");
```



```
properties.put("mail.smtp.starttls.enable", "true");
```

```
properties.put("mail.smtp.host", "smtp.gmail.com");
```

```
properties.put("mail.smtp.port", "587");
```

```
String myAccountEmail = "anchalgoldie@gmail.com";
```

```
String password = "Nevergiveup@00";
```

```
Session session = Session.getInstance(properties, new Authenticator()
```

```
{
```

```
protected PasswordAuthentication getPasswordAuthentication() {
```

```
return new PasswordAuthentication(myAccountEmail,password);
```

```
}
```

```
});
```

```
Message message = prepareMessage(session, myAccountEmail, receipient);
```

```
Transport.send(message);
```

```
System.out.println("Message sent successfully");
```

```
}
```

```
private static Message prepareMessage(Session session, String session1, String session2)
throws MessagingException, MessagingException {

    try {

        Message message = new MimeMessage(session);

        message.setFrom(new InternetAddress(session1));

        message.setRecipient(Message.RecipientType.TO, new InternetAddress(session2));

        message.setSubject("My First Email from Java Mail API");

        String htmlcode = "<h1 style='color: #663300'> WELCOME ANCHAL!</h1>";

        message.setContent(htmlcode,"text/html");

        return message;

    }

    catch (AddressException ex) {

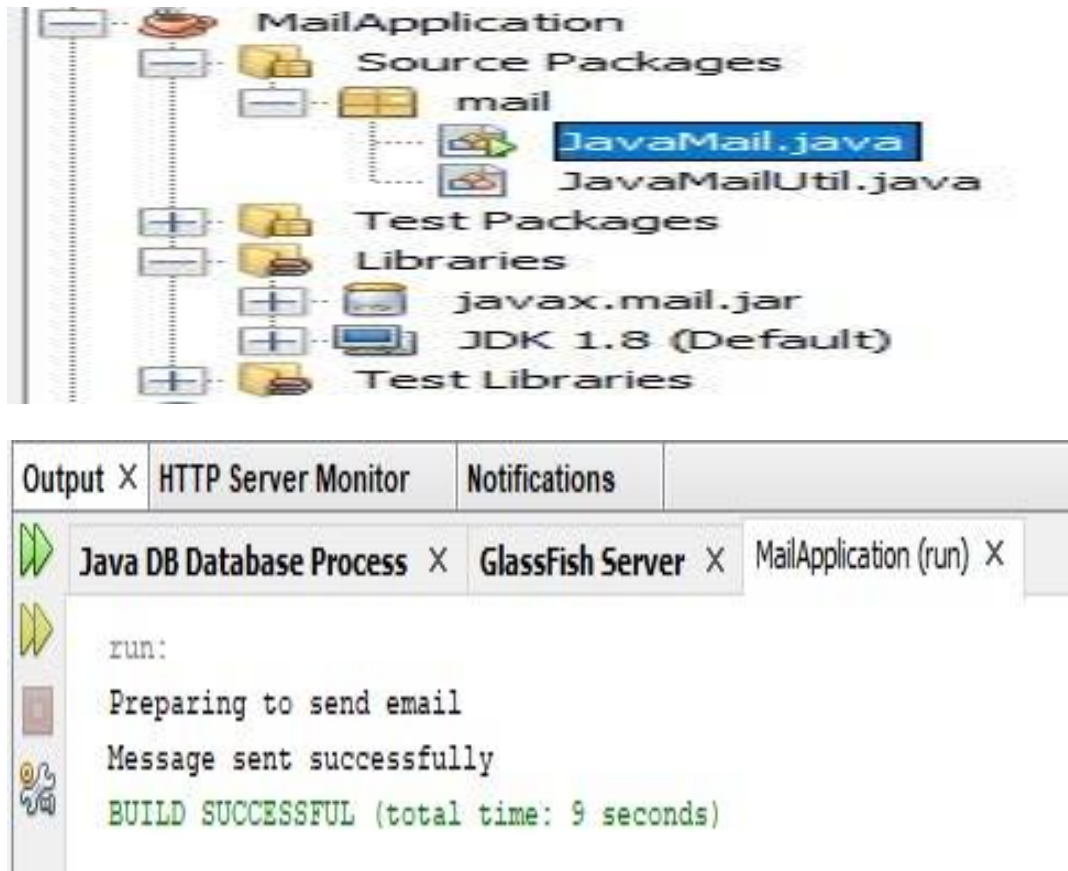
        java.util.logging.Logger.getLogger(JavaMailUtil.class.getName()).log(Level.SEVERE, null,
        ex);

    }

    return null;

}

}
```

**OUTPUT:**

## My First Email from Java Mail API Inbox x



**anchalgoldie@gmail.com**

to me ▾

# WELCOME ANCHAL!

↩ Reply

➡ Forward

**ANCHAL KUMARI**  
A12405218083

## EXPERIMENT-16

### AIM: Write a program to implement Spring

#### Theory:

- The core container is the fundamental layer of the framework that provides Inversion of Control functionality.
- Application context module extends the core modules by supplying many enterprise services such as JNDI access, remote access, multi-language support and scheduling.
- AOP module provides rich support for aspect-oriented programming
- JDBC layer simplifies the process of database connection.
- ORM module provides compatibility to existing framework such as Hibernate, JDO and TopLink.
- Web module provides a context for web-based applications.
- Spring comes with a full-featured MVC framework for building web applications.

#### What is Spring good for?

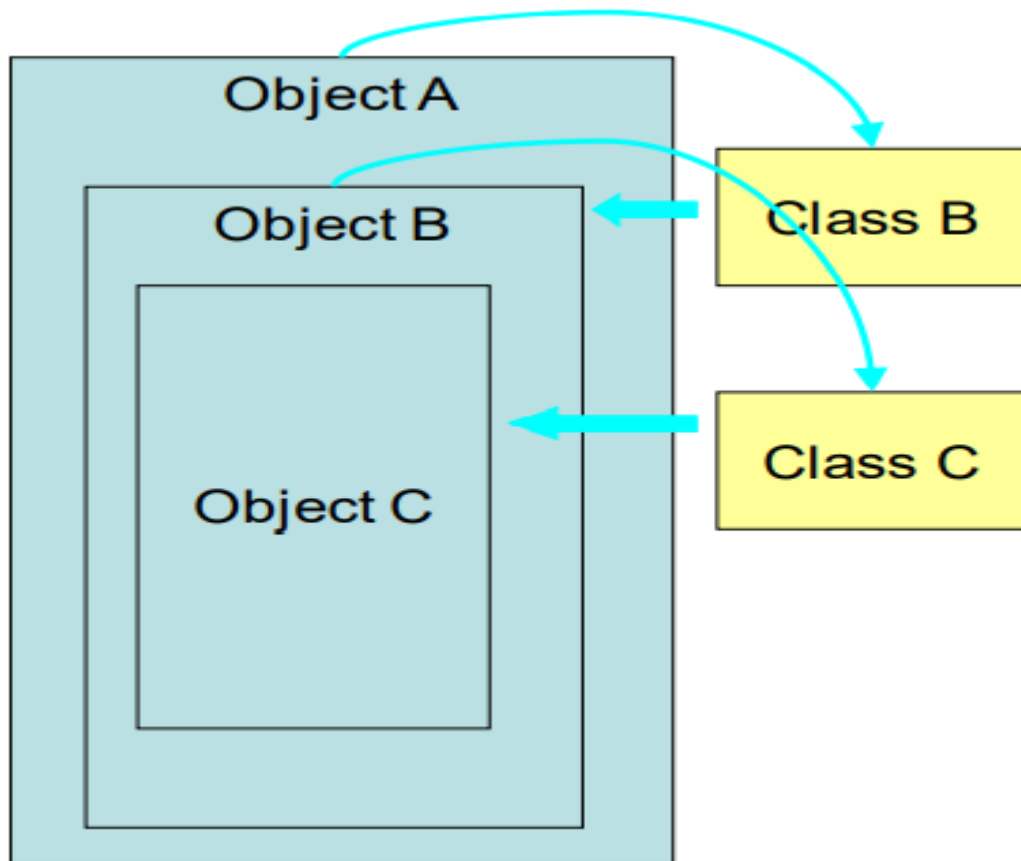
- Allows you to build applications with Plain Old Java Objects (POJOs) containing only the business logic. vs EJB
- The famous “Inversion of Control”.
- Compatible with various ORM technology, such as TopLink and Hibernate.
- Provides a testing framework that you can perform unit testing on.
- Supports Aspect Oriented Programming.
- Supports Model-View-Controller framework.

#### Spring IoC

- IoC stands for Inversion of Control
- Hollywood Principle: “Don’t call me, I’ll call you.”
- In a nutshell ... inversion of control is about: the responsibility of coordinating collaboration between dependent objects is transferred away from the objects themselves. – Spring in Action

#### Traditional approach

- In order to use object B, object A has to instantiate from Class B.
- In order to use object C, object B has to instantiate from Class C.



### Problem with traditional approach

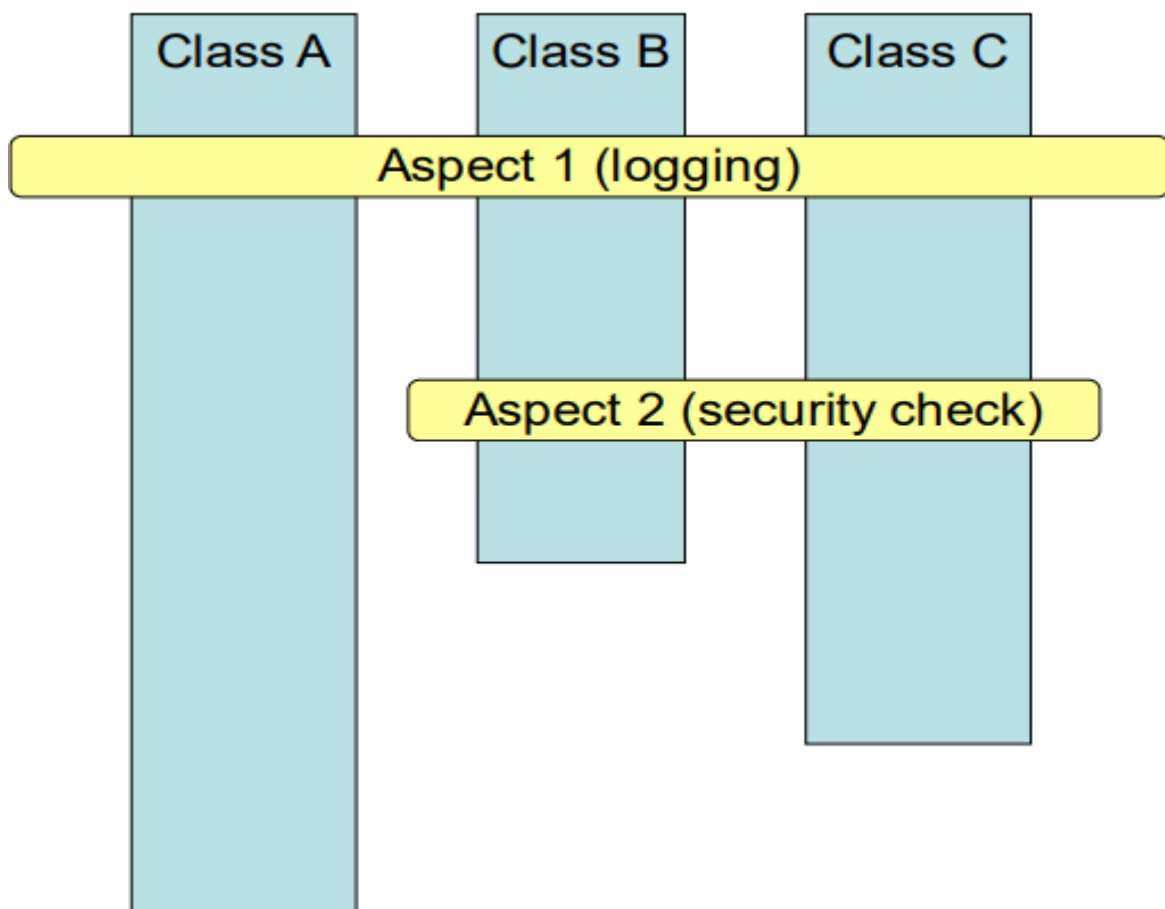
- Classes are tightly coupled
- Hard to test:  $A \leftarrow B$ ,  $B \leftarrow C$
- Hard to reuse
- Hard to understand: cannot see the main flow
- Hard to maintain: fixing one bug may result creating more new bugs
- Utilizing interface will not solve the problem

### Spring AOP

- Aspect-Oriented Programming
- AOP complements OOP
- Decompose programs into aspects (concerns)
- An aspect crosscuts multiple programs
- Common aspects:
  - Exception handling
  - Transaction management
  - Logging – Security checking

## Aspect-Oriented Programming

- Classes handle functional requirements
  - Use cases
  - Functions
- Aspects handle non-functional requirements
  - Security
  - Logging
- One aspect usually impacts multiple classes. It's tedious to implement the aspect in each class.



### AOP concepts

- Aspect
  - Action to take at a join point
- Join point
  - A method invocation
- Advice
  - Action to take at a join point
- Point cut
  - A set of join points
- Target
  - Object containing the join point

- AOP proxy and advisor – Objects for weaving the above components together

## CODE:

### Alien.java

```
package springdemo;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class SpringDemo {
    public static void main(String[] args) {
        ApplicationContext context= new
        AnnotationConfigApplicationContext(AppConfig.class);
        Alien a1= context.getBean(Alien.class);
        a1.show();
    }
}
```

### AppConfig.java

```
package springdemo;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@Configuration
@ComponentScan("springdemo")
public class AppConfig
{
    public Alien getAlien()
    {
        return new Alien();
    }
}
```

## Desktop.java

```
package springdemo;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;
@Component
@Primary
public class Desktop implements Computer{

    @Override
    public void features() {
        System.out.println("i7preprocessor 16gbram 2tbharddrive");
    }

}
```

## Laptop.java

```
package springdemo;
import org.springframework.stereotype.Component;
@Component
public class Laptop implements Computer{

    @Override
    public void features() {
        System.out.println("i5preprocessor 2gbRAM 500gbharddrive");
    }

}
```

## SpringDemo.java

```
package springdemo;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class SpringDemo {
    public static void main(String[] args) {
        ApplicationContext context= new
        AnnotationConfigApplicationContext(AppConfig.class);
        Alien a1= context.getBean(Alien.class);
        a1.show();
    }
}
```



## OUTPUT:

### Feature of Desktop

```
Hello World  
i7preprocessor 16gbram 2tbharddrive  
BUILD SUCCESSFUL (total time: 2 seconds)
```

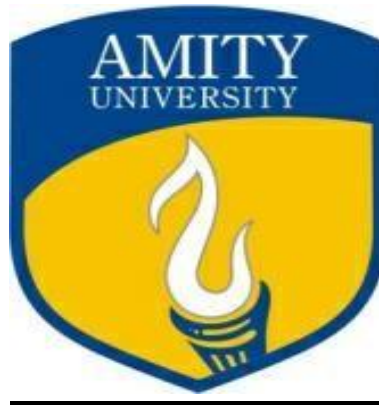
### Feature of Laptop

```
Hello World  
i5preprocessor 2gbRAM 500gbharddrive  
BUILD SUCCESSFUL (total time: 1 second)
```



**AMITY UNIVERSITY**  
UTTAR PRADESH

**ADVANCED JAVA PROGRAMMING [IT404]  
MOVIE REVIEW WEBSITE**



**AMITY SCHOOL OF  
ENGINEERING AND  
TECHNOLOGY  
AMITY UNIVERSITY CAMPUS,  
SECTOR-125, NOIDA-201303**

**Submitted By  
Anchal Kumari  
A12405218083 (6CSE-3X)  
Sheetal Kaniganti  
A2305218161 (6CSE-3X)  
Divya Shakya  
A2305218192 (6CSE-3Y)**

**Submitted To-  
Dr. Shilpi Sharma**

**Aim:** To make a Movie Review Web Application showing the use of login and register pages and show all the posts done by users till date in different genres.

**Contribution:**

Anchal Kumari- JSP+Servlet+JDBC+Front End (Login, Register)

Sheetal Kaniganti- JSP+Servlet+JDBC+ Front End (Category, Show Posts)

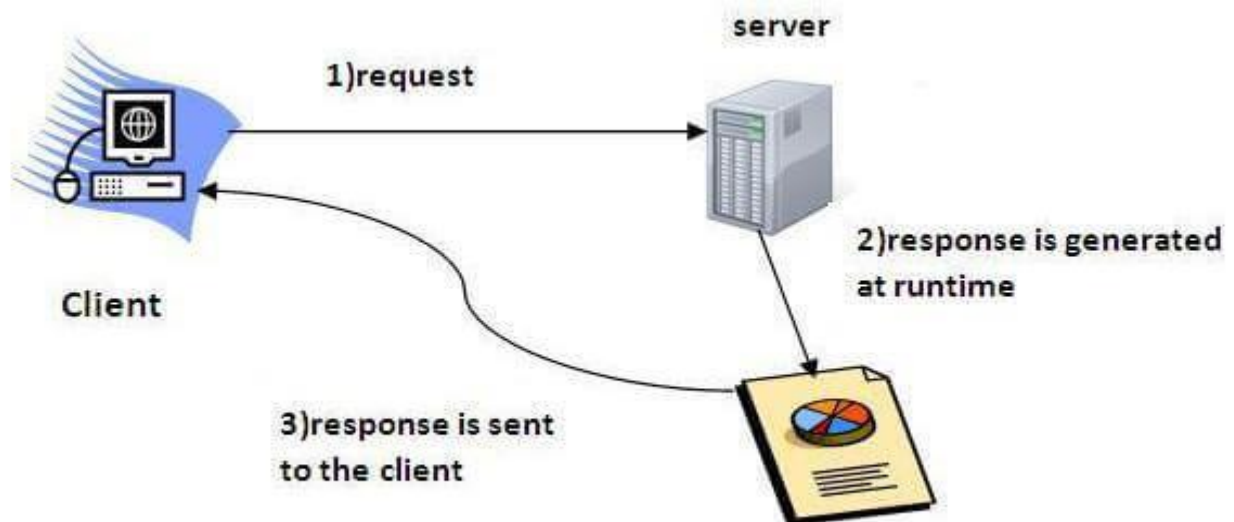
Divya Shakya- JSP+Servlet+JDBC+ Front End (Do Post, Profile)

**Theory:**

**SERVLET:**

Servlet can be described in many ways, depending on the context.

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.



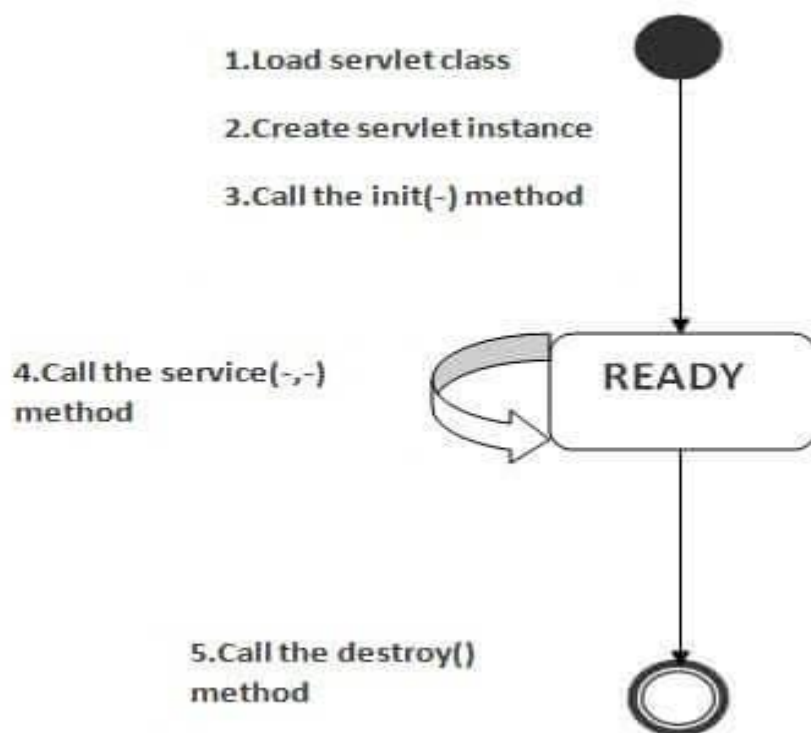
**Advantages of Servlet**

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because it uses java language.

## Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



### 1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

### 2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle

### 3) init method is invoked

- The web container calls the init method only once after creating the servlet instance.
- The init method is used to initialize the servlet.
- It is the life cycle method of the javax.servlet.Servlet interface.
- Syntax of the init method is given below:

**public void** init(ServletConfig config) **throws** ServletException

### 4) service method is invoked

- The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method.
- If servlet is initialized, it calls the service method.
- Notice that servlet is initialized only once.
- The syntax of the service method of the Servlet interface is given below:

**public void** service(ServletRequest request, ServletResponse response) **throws** ServletException, IOException

### 5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

**public void** destroy()

## JSP

- **JSP** technology is used to create web application just like Servlet technology.
- It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.
- A JSP page consists of HTML tags and JSP tags.
- The JSP pages are easier to maintain than Servlet because we can separate designing and development.
- It provides some additional features such as Expression Language, Custom Tags, etc.

### Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

#### 1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

## 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

## 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

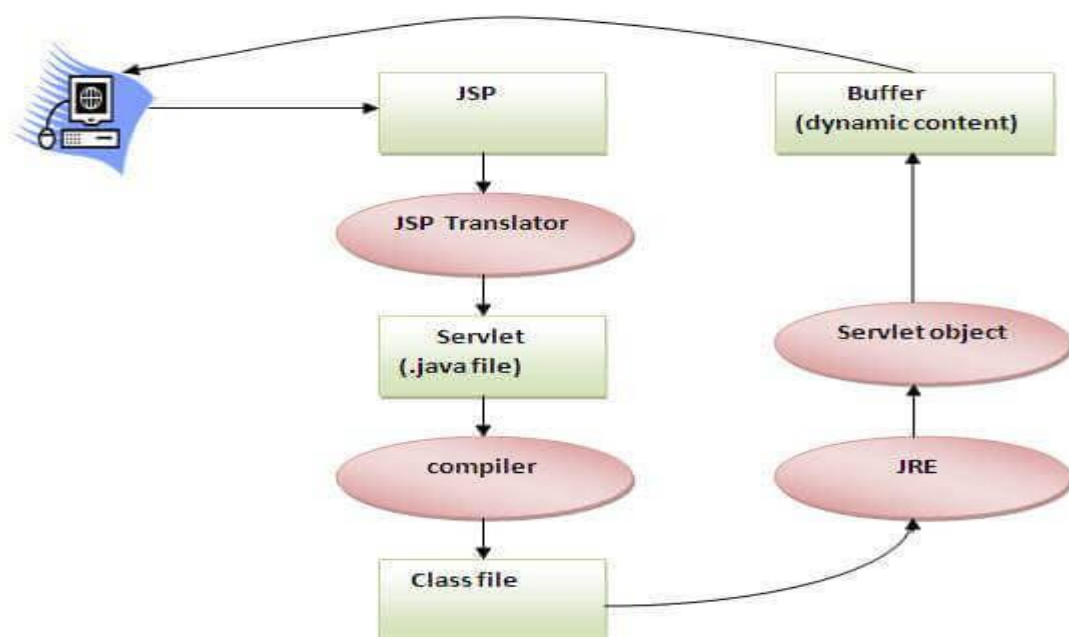
## 4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

## The Lifecycle of a JSP Page

The JSP pages follow these phases:

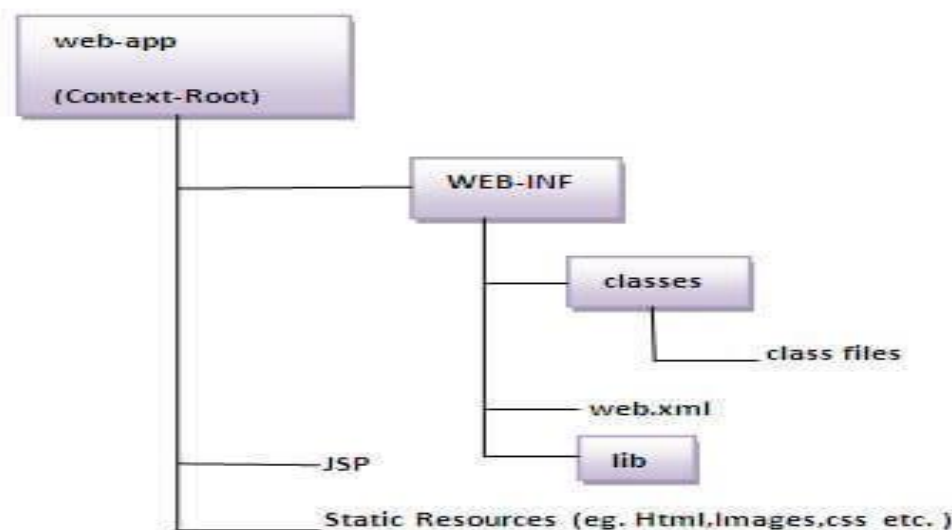
- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization ( the container invokes `jspInit()` method).
- Request processing ( the container invokes `_jspService()` method).
- Destroy ( the container invokes `jspDestroy()` method).
- **Note: `jspInit()`, `_jspService()` and `jspDestroy()` are the life cycle methods of JSP.**



As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

### The Directory structure of JSP

The directory structure of JSP page is same as Servlet. We contain the JSP page outside the WEB-INF folder or in any directory.



### JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

#### JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP.

Syntax is as follows:

<% java source code %>

## JSP expression tag

The code placed within **JSP expression tag** is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

```
<%= statement %>
```

**Note: Do not end your statement with semicolon in case of expression tag.**

## JSP Declaration Tag

The **JSP declaration tag** is used to declare fields and methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

```
<%! field or method declaration %>
```

## Difference between Servlet and JSP

Servlet	JSP
<ul style="list-style-type: none"><li>• Servlet is a java code.</li></ul>	<ul style="list-style-type: none"><li>• JSP is a html based code.</li></ul>
<ul style="list-style-type: none"><li>• Writing code for servlet is harder than JSP as it is html in java.</li></ul>	<ul style="list-style-type: none"><li>• JSP is easy to code as it is java in html.</li></ul>
<ul style="list-style-type: none"><li>• Servlet plays a controller role in MVC approach.</li></ul>	<ul style="list-style-type: none"><li>• JSP is the view in MVC approach for showing output.</li></ul>
<ul style="list-style-type: none"><li>• Servlet is faster than JSP.</li></ul>	<ul style="list-style-type: none"><li>• JSP is slower than Servlet because the first step in JSP lifecycle is the</li></ul>



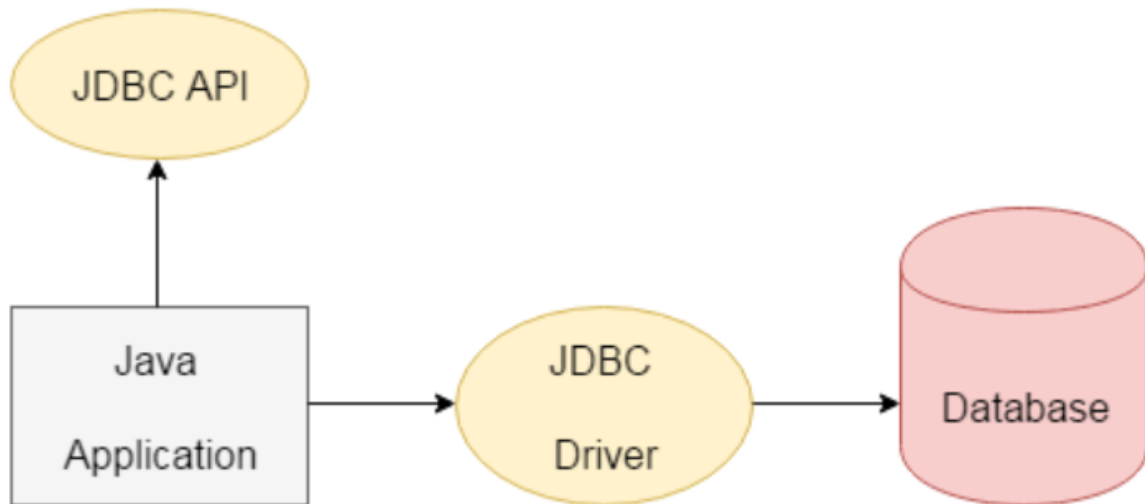
	translation of JSP to java code and then compile.
<ul style="list-style-type: none"> <li>Servlet can accept all protocol requests.</li> </ul>	<ul style="list-style-type: none"> <li>JSP only accept http requests.</li> </ul>
<ul style="list-style-type: none"> <li>In Servlet, we can override the service() method.</li> </ul>	<ul style="list-style-type: none"> <li>In JSP, we cannot override its service() method.</li> </ul>
<ul style="list-style-type: none"> <li>In Servlet by default session management is not enabled, user have to enable it explicitly.</li> </ul>	<ul style="list-style-type: none"> <li>In JSP session management is automatically enabled.</li> </ul>
<ul style="list-style-type: none"> <li>In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.</li> </ul>	<ul style="list-style-type: none"> <li>In JSP business logic is separated from presentation logic by using javaBeans.</li> </ul>
<ul style="list-style-type: none"> <li>Modification in Servlet is a time consuming task because it includes reloading, recompiling and restarting the server.</li> </ul>	<ul style="list-style-type: none"> <li>JSP modification is fast, just need to click the refresh button.</li> </ul>

### **JDBC stands for Java Database Connectivity**

JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database.



The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

A list of popular *classes* of JDBC API are given below:

- DriverManager class
- Blob class
- Clob class
- Types class

### Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database

2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

## **Code:**

### **index.jsp**

```
<% @page import="com.tech.blog.helper.ConnectionProvider"%>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<% @page import="java.sql.*" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>

    <!--css-->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
    <link href="css/mystyle.css" rel="stylesheet" type="text/css"/>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

    <style>

  </style>

</head>
<body>

    <!--navbar-->
    <% @include file="normal_navbar.jsp" %>
```

<!--/banner-->

<div class="container-fluid p-0 m-0">

<div class="jumbotron primary-background text-white banner-background">

<div class="container">

<h3 class="display-3">Movie Geeks </h3>



<p>We Are Movie Geeks began as a simple project for fun, an online way for us to share our love of movies in our spare time.

</p>

<p>

We're happy to be here, we thank our readers for being a major factor in our success and allowing us this ongoing opportunity for sharing our passion with you and talking all things movies... from the minds of the Movie Geeks.

</p>

<button class="btn btn-outline-light btn-lg"> <span ></span> Movie lovers, Share your thoughts here!</button>

<a href="login\_page.jsp" class="btn btn-outline-light btn-lg"> <span class="fa fa-user-circle fa-spin"></span> Login Here</a>

</div>

</div>

</div>

```
<!--//cards-->
```

```
<div class="container">
```

```
<div class="row mb-2">
```

```
<div class="col-md-4">
```

```
<div class="card" >
```

```
<div class="card-body ">
```

```

```

```
<h5 class="card-title">Watch A Movie</h5>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="col-md-4">
```

```
<div class="card" >
```

```
<div class="card-body ">
```

```

```

```
<h5 class="card-title">Post Your Review</h5>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="col-md-4">
```

```
<div class="card" >
```

```
<div class="card-body ">
    
    <h5 class="card-title ">See Other's Reviews</h5>
```

```
</div>
</div>
</div>
```

```
</div>
```

```
<!--javascripts-->
<script
    src="https://code.jquery.com/jquery-3.4.1.min.js"
    integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
    crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
<script src="js/myjs.js" type="text/javascript"></script>
```

```
</body>
</html>
```

### **load\_posts.jsp**

```
<% @page import="com.tech.blog.entities.User"%>
```

```
<% @page import="com.tech.blog.dao.LikeDao"%>
<% @page import="com.tech.blog.entities.Post"%>
<% @page import="java.util.List"%>
<% @page import="com.tech.blog.helper.ConnectionProvider"%>
<% @page import="com.tech.blog.dao.PostDao"%>

<div class="row">

    <%

        User uu=(User)session.getAttribute("currentUser");

        PostDao d = new PostDao(ConnectionProvider.getConnection());

        int cid = Integer.parseInt(request.getParameter("cid"));
        List<Post> posts = null;
        if (cid == 0) {
            posts = d.getAllPosts();
        } else {
            posts = d.getPostByCatId(cid);
        }

        if (posts.size() == 0) {
            out.println("<h3 class='display-4 text-white text-center'>Oops! No Posts in this
category..</h3>");
            return;
        }

        for (Post p : posts) {
    %>

    <div class="col-md-6 mt-2">
```

```

<div class="card">
    
    <div class="card-body">
        <b><%= p.getpTitle()%></b>
        <p><%= p.getpContent()%></p>

    </div>
    <div class="card-footer primary-background text-center">
        <%
            LikeDao ld = new LikeDao(ConnectionProvider.getConnection());
        %>

        <a href="#" onclick="doLike(<%= p.getPid()%>,<%= uu.getId()%>)" class="btn
btn-outline-light btn-sm"> <i class="fa fa-thumbs-o-up"></i> <span class="like-
counter"><%= ld.countLikeOnPost(p.getPid())%></span> </a>

        <a href="show_blog_page.jsp?post_id=<%= p.getPid()%>" class="btn btn-outline-
light btn-sm">Read More...</a>

        <a href="#" class="btn btn-outline-light btn-sm"> <i class="fa fa-commenting-
o"></i> <span>20</span> </a>

    </div>

</div>

</div>

<%
    }

%>

```



</div>

## **login\_page.jsp**

```
<% @page import="com.tech.blog.entities.Message"%>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Login Page</title>

        <!--css-->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
        <link href="css/mystyle.css" rel="stylesheet" type="text/css"/>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
        <style>

    </style>

    </head>
    <body>

        <!--navbar-->
        <% @include file="normal_navbar.jsp" %>

        <main class="d-flex align-items-center primary-background banner-background"
style="height: 70vh">
```

```

<div class="container">
  <div class="row">
    <div class="col-md-4 offset-md-4">

      <div class="card">
        <div class="card-header primary-background text-white text-center">
          <span class="fa fa-user-plus fa-3x"></span>
          <br>
          <p>Login here</p>
        </div>

        <%
          Message m = (Message) session.getAttribute("msg");
          if (m != null) {
        %>
        <div class="alert <%= m.getCssClass() %>" role="alert">
          <%= m.getContent() %>
        </div>

        <%
          session.removeAttribute("msg");
        %>

        <div class="card-body">
          <form action="LoginServlet" method="post">
            <div class="form-group">
              <label for="exampleInputEmail1">Email address</label>

```

```
        <input name="email" required type="email" class="form-control"
id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email">
```

```
        <small id="emailHelp" class="form-text text-muted">We'll never
share your email with anyone else.</small>
```

```
    </div>
```

```
    <div class="form-group">
```

```
        <label for="exampleInputPassword1">Password</label>
```

```
        <input name="password" required type="password" class="form-
control" id="exampleInputPassword1" placeholder="Password">
```

```
    </div>
```

```
    <div class="container text-center">
```

```
        <button type="submit" class="btn btn-primary">Submit</button>
```

```
    </div>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</main>
```

```
<!--javascripts-->
```

```

<script
  src="https://code.jquery.com/jquery-3.4.1.min.js"
  integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
  crossorigin="anonymous"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
  integrity="sha384-
  ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
  crossorigin="anonymous"></script>

  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
  integrity="sha384-
  JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI"
  crossorigin="anonymous"></script>

  <script src="js/myjs.js" type="text/javascript"></script>

</body>
</html>

```

### **normal navbar.jsp**

```

<nav class="navbar navbar-expand-lg navbar-dark primary-background">
  <div>
    
  </div>
  <a class="navbar-brand" href="index.jsp"> <span> </span> Movie Bloggers</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
  target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
  expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">

      <li class="nav-item">
        <a class="nav-link" href="login_page.jsp"> <span class="fa fa-user-circle
        "></span> Login</a>

```

```

        </li>

        <li class="nav-item">
            <a class="nav-link" href="register_page.jsp"> <span class="fa fa-user-plus
"></span> Sign up</a>
        </li>

    </ul>

</div>
</nav>

```

### **profile.jsp**

```

<% @page import="com.tech.blog.entities.Category"%>
<% @page import="java.util.ArrayList"%>
<% @page import="com.tech.blog.helper.ConnectionProvider"%>
<% @page import="com.tech.blog.dao.PostDao"%>
<% @page import="com.tech.blog.entities.Message"%>
<% @page import="com.tech.blog.entities.User"%>
<% @page errorPage="error_page.jsp" %>
<%

```

```

    User user = (User) session.getAttribute("currentUser");
    if (user == null) {
        response.sendRedirect("login_page.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>
```

```
<!--css-->
```

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
```

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

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
```

```
<style>
```

```
.banner-background{
    clip-path: polygon(30% 0%, 70% 0%, 100% 0, 100% 91%, 63% 100%, 22% 91%,
0 99%, 0 0);
}
```

```
body{
    background:url(img/bg.jpg);
    background-size: cover;
    background-attachment: fixed;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<!--navbar-->
```

```
<nav class="navbar navbar-expand-lg navbar-dark primary-background">
```

```

```

```

        <a class="navbar-brand" href="index.jsp"> <span ></span>Movie Bloggers</a>

        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">

            <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">

            <ul class="navbar-nav mr-auto">

                <li class="nav-item">

                    <a class="nav-link" href="#" data-toggle="modal" data-target="#add-post-
modal" > <span class="fa fa-asterisk"></span> Do Post</a>

                </li>

            </ul>

            <ul class="navbar-nav mr-right">

                <li class="nav-item">

                    <a class="nav-link" href="#" data-toggle="modal" data-target="#profile-
modal"> <span class="fa fa-user-circle "></span> <%= user.getName()%> </a>

                </li>

                <li class="nav-item">

                    <a class="nav-link" href="LogoutServlet"> <span class="fa fa-user-plus
"></span> Logout</a>

                </li>

            </ul>

        </div>

```

```
</nav>
```

```
<!--end of navbar-->
```

```
<%
```

```
    Message m = (Message) session.getAttribute("msg");
```

```
    if (m != null) {
```

```
%>
```

```
<div class="alert <%= m.getCssClass()%>" role="alert">
```

```
    <%= m.getContent()%>
```

```
</div>
```

```
<%
```

```
    session.removeAttribute("msg");
```

```
}
```

```
%>
```

```
<!--main body of the page-->
```

```
<main>
```

```
    <div class="container">
```

```
        <div class="row mt-4">
```

```
            <!--first col-->
```

```
            <div class="col-md-4">
```

```
                <!--categories-->
```

```
                <div class="list-group">
```



```
<a href="#" onclick="getPosts(0, this)" class="c-link list-group-item list-group-item-action active">
```

```
    All Posts
```

```
</a>
```

```
<!--categories-->
```

```
        <%
            PostDao d = new
            PostDao(ConnectionProvider.getConnection());
```

```
        ArrayList<Category> list1 = d.getAllCategories();
```

```
        for (Category cc : list1) {
```

```
        %>
```

```
        <a href="#" onclick="getPosts(<%= cc.getCid()%>, this)" class="c-link
list-group-item list-group-item-action"><%= cc.getName()%></a>
```

```
        <%
            }
        %>
```

```
    %>
```

```
</div>
```

```
</div>
```

```
<!--second col-->
```

```
<div class="col-md-8" >
```

```
    <!--posts-->
```

```
    <div class="container text-center" id="loader">
```

```
        <i class="fa fa-refresh fa-4x fa-spin"></i>
```

```
        <h3 class="mt-2">Loading...</h3>
```

```
    </div>
```

```
<div class="container-fluid" id="post-container">
```

```
</div>
```

</div>

</div>

</div>

</main>

<!--end main body of the page-->

<!--profile modal-->

<!-- Modal -->

<div class="modal fade" id="profile-modal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header primary-background text-white text-center">

<h5 class="modal-title" id="exampleModalLabel"> Movie Review </h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<div class="container text-center">



<br>

<h5 class="modal-title mt-3" id="exampleModalLabel"> <%= user.getName()%> </h5>

<!--//details-->

<div id="profile-details">

```
<table class="table">
```

```
<tbody>
```

```
<tr>
```

```
<th scope="row"> ID :</th>
```

```
<td><%= user.getId()%></td>
```

```
</tr>
```

```
<tr>
```

```
<th scope="row"> Email : </th>
```

```
<td><%= user.getEmail()%></td>
```

```
</tr>
```

```
<tr>
```

```
<th scope="row">Gender :</th>
```

```
<td><%= user.getGender()%></td>
```

```
</tr>
```

```
<tr>
```

```
<th scope="row">Status :</th>
```

```
<td><%= user.getAbout()%></td>
```

```
</tr>
```

```
<tr>
```

```
<th scope="row">Registered on :</th>
```

```
<td><%= user.getDateTime().toString()%></td>
```

```
</tr>
```

```
</tbody>
```

```
</table>
```

```
</div>
```

```
<!--profile edit-->
```

```

<div id="profile-edit" style="display: none;">
    <h3 class="mt-2">Please Edit Carefully</h3>
    <form action="EditServlet" method="post" enctype="multipart/form-
data">

        <table class="table">
            <tr>
                <td>ID :</td>
                <td><%= user.getId()%></td>
            </tr>
            <tr>
                <td>Email :</td>
                <td><input type="email" class="form-control"
name="user_email" value="<%= user.getEmail()%>" > </td>
            </tr>
            <tr>
                <td>Name :</td>
                <td><input type="text" class="form-control" name="user_name"
value="<%= user.getName()%>" > </td>
            </tr>
            <tr>
                <td>Password :</td>
                <td><input type="password" class="form-control"
name="user_password" value="<%= user.getPassword()%>" > </td>
            </tr>
            <tr>
                <td>Gender :</td>
                <td><%= user.getGender().toUpperCase()%> </td>
            </tr>
            <tr>
                <td>About :</td>
                <td>
                    <textarea rows="3" class="form-control" name="user_about"
><%= user.getAbout()%>

```

```
        </textarea>

    </td>

</tr>

<tr>

    <td>New Profile:</td>

    <td>

        <input type="file" name="image" class="form-control" >

    </td>

</tr>

</table>

<div class="container">

    <button type="submit" class="btn btn-outline-
primary">Save</button>

</div>

</form>

</div>

</div>

</div>

<div class="modal-footer">

    <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>

    <button id="edit-profile-button" type="button" class="btn btn-
primary">EDIT</button>

</div>

</div>

</div>

</div>
```

```
<!--end of profile modal-->
```

```
<!--add post modal-->
```

```
<!-- Modal -->
```

```
<div class="modal fade" id="add-post-modal" tabindex="-1" role="dialog" aria-  
labelledby="exampleModalLabel" aria-hidden="true">
```

```
<div class="modal-dialog" role="document">
```

```
<div class="modal-content">
```

```
<div class="modal-header">
```

```
<h5 class="modal-title" id="exampleModalLabel">Provide the post  
details..</h5>
```

```
<button type="button" class="close" data-dismiss="modal" aria-  
label="Close">
```

```
<span aria-hidden="true">&times;</span>
```

```
</button>
```

```
</div>
```

```
<div class="modal-body">
```

```
<form id="add-post-form" action="AddPostServlet1" method="post">
```

```
<div class="form-group">
```

```
<select class="form-control" name="cid">
```

```
<option selected disabled>---Select Category---</option>
```

```
<%
```

```
PostDao postd = new  
PostDao(ConnectionProvider.getConnection());
```

```
ArrayList<Category> list = postd.getAllCategories();
```

```
for (Category c : list) {
```

```
%>
```

```
<option value="<%= c.getCid()%>"><%= c.getName()%></option>
```

```
<%
```

```
}
```

```
%>
```

```
</select>
```

```
</div>
```

```
<div class="form-group">
```

```
<input name="pTitle" type="text" placeholder="Enter Movie Title"
class="form-control"/>
```

```
</div>
```

```
<div class="form-group">
```

```
<textarea name="pContent" class="form-control" style="height: 200px;"
placeholder="Enter your review"></textarea>
```

```
</div>
```

```
<div class="form-group">
```

```
<textarea name="pCode" class="form-control" style="height: 200px;"
placeholder="Summarize your review"></textarea>
```

```
</div>
```

```
<div class="form-group">
```

```
<label>Select your pic.</label>
```

```
<br>
```

```
<input type="file" name="pic" >
```

```
</div>
```

```
<div class="container text-center">
```

```
<button type="submit" class="btn btn-outline-primary">Post </button>
```

```
</div>
```

```
</form>
```

</div>

</div>

</div>

</div>

<!--END add post modal-->

</body>

</html>

### **register\_page.jsp**

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

<!DOCTYPE html>

<html>

<head>

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

<title>Register Page</title>

<!--css-->

<link rel="stylesheet"  
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"  
integrity="sha384-  
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"  
crossorigin="anonymous">

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

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-  
awesome/4.7.0/css/font-awesome.min.css">

<style>

.banner-background{  
clip-path: polygon(30% 0%, 70% 0%, 100% 0, 100% 91%, 63% 100%, 22% 91%,  
0 99%, 0 0);  
}



</style>

</head>

<body>

<% @include file="normal\_navbar.jsp" %>

<main class="primary-background banner-background" style="padding-bottom: 80px;">

<div class="container">

<div class="col-md-6 offset-md-3">

<div class="card">

<div class="card-header text-center primary-background text-white">

<span class="fa fa-3x fa-user-circle"></span>

<br>

Register here

</div>

<div class="card-body">

<form id="reg-form" action="RegisterServlet" method="POST">

<div class="form-group">

<label for="user\_name">User Name</label>

<input name="user\_name" type="text" class="form-control" id="user\_name" aria-describedby="emailHelp" placeholder="Enter name">

</div>

<div class="form-group">

<label for="exampleInputEmail1">Email address</label>

<input name="user\_email" type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email">

```
        <small id="emailHelp" class="form-text text-muted">We'll never  
share your email with anyone else.</small>  
    </div>
```

```
    <div class="form-group">  
        <label for="exampleInputPassword1">Password</label>  
        <input name="user_password" type="password" class="form-control"  
id="exampleInputPassword1" placeholder="Password">  
    </div>
```

```
    <div class="form-group">  
        <label for="gender">Select Gender</label>  
        <br>  
        <input type="radio" id="gender" name="gender" value="male" >Male  
        <input type="radio" id="gender" name="gender"  
value="female">Female  
    </div>
```

```
    <div class="form-group">  
        <textarea name="about" class="form-control" id="" rows="5"  
placeholder="Enter something about yourself"></textarea>  
    </div>
```

```
    <div class="form-check">  
        <input name="check" type="checkbox" class="form-check-input"  
id="exampleCheck1">  
        <label class="form-check-label" for="exampleCheck1">agree terms  
and conditions</label>  
    </div>
```

```

        <br>

        <div class="container text-center" id="loader" style="display: none;">
            <span class="fa fa-refresh fa-spin fa-4x"></span>
            <h4>Please wait..</h4>
        </div>

        <button id="sumbimt-btn" type="submit" class="btn btn-
dark">Submit</button>
    </form>

</div>

</div>

</div>

</div>

</main>
</body>
</html>

```

### **show\_blog\_page.jsp**

```

<% @page import="com.tech.blog.dao.LikeDao"%>
<% @page import="java.text.DateFormat"%>
<% @page import="com.tech.blog.dao.UserDao"%>
<% @page import="java.util.ArrayList"%>
<% @page import="com.tech.blog.entities.Category"%>
<% @page import="com.tech.blog.entities.Category"%>
<% @page import="com.tech.blog.helper.ConnectionProvider"%>
<% @page import="com.tech.blog.dao.PostDao"%>

```

```
<% @page import="com.tech.blog.entities.Post"%>
<% @page import="com.tech.blog.entities.User"%>
<% @page errorPage="error_page.jsp" %>

<%

    User user = (User) session.getAttribute("currentUser");
    if (user == null) {
        response.sendRedirect("login_page.jsp");
    }

%>

<%    int postId = Integer.parseInt(request.getParameter("post_id"));
    PostDao d = new PostDao(ConnectionProvider.getConnection());

    Post p = d.getPostByPostId(postId);
%>

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title><%= p.getpTitle()%> || Give Movie Reviews here! </title>

        <!--css-->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
```

```
<link href="css/mystyle.css" rel="stylesheet" type="text/css"/>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
```

```
<style>
```

```
.banner-background{
    clip-path: polygon(30% 0%, 70% 0%, 100% 0, 100% 91%, 63% 100%, 22% 91%,
0 99%, 0 0);
}
.post-title{
    font-weight: 100;
    font-size: 30px;
}
.post-content{
    font-weight: 100;
    font-size: 25px;

}
.post-date{
    font-style: italic;
    font-weight: bold;
}
.post-user-info{
    font-size: 20px;

}

.row-user{
    border:1px solid #e2e2e2;
    padding-top: 15px;
```

```
}
```

```
body{  
    background:url(img/bg.jpeg);  
    background-size: cover;  
    background-attachment: fixed;  
}
```

```
</style>
```

```
<div id="fb-root"></div>
```

```
<script async defer crossorigin="anonymous"  
src="https://connect.facebook.net/en_GB/sdk.js#xfbml=1&version=v10.0"  
nonce="KP0VM3rh"></script>
```

```
</head>
```

```
<body>
```

```
<!--navbar-->
```

```
<nav class="navbar navbar-expand-lg navbar-dark primary-background">
```

```
    
```

```
    <a class="navbar-brand" href="index.jsp"> <span>Movie Bloggers</span></a>
```

```
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-  
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-  
expanded="false" aria-label="Toggle navigation">
```

```
        <span class="navbar-toggler-icon"></span>
```

```
    </button>
```

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
```

```
    <ul class="navbar-nav mr-auto">
```

```
        <li class="nav-item">
```

```
            <a class="nav-link" href="#" data-toggle="modal" data-target="#add-post-  
modal" > <span class="          fa fa-asterisk"></span> Do Post</a>
```

</li>

</ul>

<ul class="navbar-nav mr-right">

<li class="nav-item">

<a class="nav-link" href="#" data-toggle="modal" data-target="#profile-modal"> <span class="fa fa-user-circle "></span> <%= user.getName()%> </a>

</li>

<li class="nav-item">

<a class="nav-link" href="LogoutServlet"> <span class="fa fa-user-plus "></span> Logout</a>

</li>

</ul>

</div>

</nav>

<!--end of navbar-->

<!--main content of body-->

<div class="container">

<div class="row my-4">

<div class="col-md-8 offset-md-2">

```
<div class="card">
```

```
<div class="card-header primary-background text-white">
```

```
<h4 class="post-title"><%= p.getpTitle()%></h4>
```

```
</div>
```

```
<div class="card-body">
```

```

```

```
<div class="row my-3 row-user">
```

```
<div class="col-md-8">
```

```
<% UserDao ud = new  
UserDao(ConnectionProvider.getConnection());%>
```

```
<p class="post-user-info"> <a href="#"> <%=  
ud.getUserByUserId(p.getUserId()).getName()%></a> has posted : </p>
```

```
</div>
```

```
<div class="col-md-4">
```

```
<p class="post-date"> <%=  
DateFormat.getDateTimeInstance().format(p.getpDate()%> </p>
```

```
</div>
```

```
</div>
```

```
<p class="post-content"><%= p.getpContent()%></p>
```



<br>

<br>

<div class="post-code">

<pre><%= p.getpCode()%></pre>

</div>

</div>

<div class="card-footer primary-background">

<%

LikeDao ld = new LikeDao(ConnectionProvider.getConnection());

%>

<a href="#" onclick="doLike(<%= p.getPid()%>,<%= user.getId()%>)"  
class="btn btn-outline-light btn-sm"> <i class="fa fa-thumbs-o-up"></i> <span class="like-  
counter"><%= ld.countLikeOnPost(p.getPid())%></span> </a>

<a href="#" class="btn btn-outline-light btn-sm"> <i class="fa fa-  
commenting-o"></i> <span>20</span> </a>

</div>

<div class="card-footer">

<div class="fb-comments" data-  
href="http://localhost:9494/TechBlog/show\_blog\_page.jsp?post\_id=<%=p.getPid() %>"  
data-width="" data-numposts="5"></div>

</div>

</div>

</div>

</div>

</div>

<!--end of main content of body-->

<!--profile modal-->

<!-- Modal -->

<div class="modal fade" id="profile-modal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header primary-background text-white text-center">

<h5 class="modal-title" id="exampleModalLabel"> Movie Bloggers</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<div class="container text-center">



<br>

<h5 class="modal-title mt-3" id="exampleModalLabel"> <%= user.getName()%> </h5>

<!--//details-->

<div id="profile-details">

<table class="table">

```
<tbody>
  <tr>
    <th scope="row"> ID :</th>
    <td> <%= user.getId()%></td>

  </tr>
  <tr>
    <th scope="row"> Email : </th>
    <td><%= user.getEmail()%></td>

  </tr>
  <tr>
    <th scope="row">Gender :</th>
    <td><%= user.getGender()%></td>

  </tr>
  <tr>
    <th scope="row">Status :</th>
    <td><%= user.getAbout()%></td>

  </tr>
  <tr>
    <th scope="row">Registered on :</th>
    <td><%= user.getDateTime().toString()%></td>

  </tr>
</tbody>
</table>
</div>

<!--profile edit-->
```

```
<div id="profile-edit" style="display: none;">
  <h3 class="mt-2">Please Edit Carefully</h3>
  <form action="EditServlet" method="post" enctype="multipart/form-data">
    <table class="table">
      <tr>
        <td>ID :</td>
        <td><%= user.getId()%></td>
      </tr>
      <tr>
        <td>Email :</td>
        <td><input type="email" class="form-control" name="user_email"
value="<%= user.getEmail()%>" > </td>
      </tr>
      <tr>
        <td>Name :</td>
        <td><input type="text" class="form-control" name="user_name"
value="<%= user.getName()%>" > </td>
      </tr>
      <tr>
        <td>Password :</td>
        <td><input type="password" class="form-control"
name="user_password" value="<%= user.getPassword()%>" > </td>
      </tr>
      <tr>
        <td>Gender :</td>
        <td><%= user.getGender().toUpperCase()%> </td>
      </tr>
      <tr>
        <td>About :</td>
        <td>
          <textarea rows="3" class="form-control" name="user_about"
><%= user.getAbout()%>
          </textarea>
        </td>
      </tr>
    </table>
  </form>
</div>
```

```

        </td>
    </tr>
    <tr>
        <td>New Profile:</td>
        <td>
            <input type="file" name="image" class="form-control" >
        </td>
    </tr>

</table>

<div class="container">
    <button type="submit" class="btn btn-outline-
primary">Save</button>
</div>

</form>

</div>

</div>
</div>
<div class="modal-footer">
    <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
    <button id="edit-profile-button" type="button" class="btn btn-
primary">EDIT</button>
</div>
</div>
</div>
</div>

<!--end of profile modal-->

```

```
<!--add post modal-->
```

```
<!-- Modal -->
```

```
<div class="modal fade" id="add-post-modal" tabindex="-1" role="dialog" aria-  
labelledby="exampleModalLabel" aria-hidden="true">
```

```
<div class="modal-dialog" role="document">
```

```
<div class="modal-content">
```

```
<div class="modal-header">
```

```
<h5 class="modal-title" id="exampleModalLabel">Provide the post  
details..</h5>
```

```
<button type="button" class="close" data-dismiss="modal" aria-label="Close">
```

```
<span aria-hidden="true">&times;</span>
```

```
</button>
```

```
</div>
```

```
<div class="modal-body">
```

```
<form id="add-post-form" action="AddPostServlet" method="post">
```

```
<div class="form-group">
```

```
<select class="form-control" name="cid">
```

```
<option selected disabled>---Select Category---</option>
```

```
<%
```

```
PostDao postd = new PostDao(ConnectionProvider.getConnection());
```

```
ArrayList<Category> list = postd.getAllCategories();
```

```
for (Category c : list) {
```

```
%>
```

```
<option value="<%= c.getCid()%>"><%= c.getName()%></option>
```

```
<%
```

```

        }
    %>
</select>
</div>

<div class="form-group">
    <input name="pTitle" type="text" placeholder="Enter post Title"
class="form-control"/>
</div>

<div class="form-group">
    <textarea name="pContent" class="form-control" style="height: 200px;"
placeholder="Enter your content"></textarea>
</div>
<div class="form-group">
    <textarea name="pCode" class="form-control" style="height: 200px;"
placeholder="Enter your program (if any)"></textarea>
</div>
<div class="form-group">
    <label>Select your pic.</label>
    <br>
    <input type="file" name="pic" >
</div>

<div class="container text-center">
    <button type="submit" class="btn btn-outline-primary">Post </button>
</div>

</form>

</div>

</div>

```

```
</div>
</div>
```

```
<!--END add post modal-->
```

```
</body>
</html>
```

### **com.tech.blog.dao**

#### **UserDao.java**

```
package com.tech.blog.dao;
```

```
import com.tech.blog.entities.User;
import java.sql.*;
```

```
public class UserDao {
```

```
    private Connection con;
```

```
    public UserDao(Connection con) {
        this.con = con;
    }
```

```
    //method to insert user to data base:
```

```
    public boolean saveUser(User user) {
        boolean f = false;
        try {
            //user --> database
```

```
            String query = "insert into user(name,email,password,gender,about) values
            (?, ?, ?, ?, ?)";
```



```

        PreparedStatement pstmt = this.con.prepareStatement(query);
        pstmt.setString(1, user.getName());
        pstmt.setString(2, user.getEmail());
        pstmt.setString(3, user.getPassword());
        pstmt.setString(4, user.getGender());
        pstmt.setString(5, user.getAbout());

        pstmt.executeUpdate();
        f = true;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return f;
}

//get user by useremail and userpassword:
public User getUserByEmailAndPassword(String email, String password) {
    User user = null;

    try {

        String query = "select * from user where email =? and password=?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, email);
        pstmt.setString(2, password);

        ResultSet set = pstmt.executeQuery();

        if (set.next()) {
            user = new User();

            //      data from db

```

```

        String name = set.getString("name");
//      set to user object
        user.setName(name);

        user.setId(set.getInt("id"));
        user.setEmail(set.getString("email"));
        user.setPassword(set.getString("password"));
        user.setGender(set.getString("gender"));
        user.setAbout(set.getString("about"));
        user.setDateTime(set.getTimestamp("rdate"));
        user.setProfile(set.getString("profile"));

    }

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

    return user;
}

public boolean updateUser(User user) {

    boolean f = false;
    try {

        String query = "update user set name=? , email=? , password=? , gender=? ,about=? ,
profile=? where id =?";

        PreparedStatement p = con.prepareStatement(query);
        p.setString(1, user.getName());
        p.setString(2, user.getEmail());
        p.setString(3, user.getPassword());
        p.setString(4, user.getGender());

```

```

        p.setString(5, user.getAbout());
        p.setString(6, user.getProfile());
        p.setInt(7, user.getId());

        p.executeUpdate();
        f = true;

    } catch (Exception e) {
        e.printStackTrace();
    }
    return f;
}

public User getUserById(int userId) {
    User user = null;
    try {
        String q = "select * from user where id=?";
        PreparedStatement ps = this.con.prepareStatement(q);
        ps.setInt(1, userId);
        ResultSet set = ps.executeQuery();
        if (set.next()) {
            user = new User();

//            data from db
            String name = set.getString("name");
//            set to user object
            user.setName(name);

            user.setId(set.getInt("id"));
            user.setEmail(set.getString("email"));
            user.setPassword(set.getString("password"));
            user.setGender(set.getString("gender"));
            user.setAbout(set.getString("about"));

```

```

        user.setDateTime(set.getTimestamp("rdate"));
        user.setProfile(set.getString("profile"));
    }
} catch (Exception e) {
    e.printStackTrace();
}

return user;
}
}

```

### **PostDao.java**

```

package com.tech.blog.dao;

import com.tech.blog.entities.Category;
import com.tech.blog.entities.Post;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PostDao {

    Connection con;

    public PostDao(Connection con) {
        this.con = con;
    }

    public ArrayList<Category> getAllCategories() {
        ArrayList<Category> list = new ArrayList<>();

        try {

            String q = "select * from categories";

```

```

Statement st = this.con.createStatement();
ResultSet set = st.executeQuery(q);
while (set.next()) {
    int cid = set.getInt("cid");
    String name = set.getString("name");
    String description = set.getString("description");
    Category c = new Category(cid, name, description);
    list.add(c);
}

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

return list;
}

public boolean savePost(Post p) {
    boolean f = false;
    try {

        String q = "insert into posts(pTitle,pContent,pCode,pPic,catId,userId)
values(?,?,?,?,?,?)";

        PreparedStatement pstmt = con.prepareStatement(q);
        pstmt.setString(1, p.getpTitle());
        pstmt.setString(2, p.getpContent());
        pstmt.setString(3, p.getpCode());
        pstmt.setString(4, p.getpPic());
        pstmt.setInt(5, p.getCatId());
        pstmt.setInt(6, p.getUserId());
        pstmt.executeUpdate();
        f = true;
    }
}

```

```
    } catch (Exception e) {
        e.printStackTrace();
    }

    return f;
}

// get all the posts
public List<Post> getAllPosts() {

    List<Post> list = new ArrayList<>();
    //fetch all the post
    try {

        PreparedStatement p = con.prepareStatement("select * from posts order by pid desc");

        ResultSet set = p.executeQuery();

        while (set.next()) {

            int pid = set.getInt("pid");
            String pTitle = set.getString("pTitle");
            String pContent = set.getString("pContent");
            String pCode = set.getString("pCode");
            String pPic = set.getString("pPic");
            Timestamp date = set.getTimestamp("pDate");
            int catId = set.getInt("catId");
            int userId = set.getInt("userId");
            Post post = new Post(pid, pTitle, pContent, pCode, pPic, date, catId, userId);

            list.add(post);
        }
    }
```

```

    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;
}

public List<Post> getPostByCatId(int catId) {
    List<Post> list = new ArrayList<>();
    //fetch all post by id
    //fetch all the post
    try {

        PreparedStatement p = con.prepareStatement("select * from posts where catId=?");
        p.setInt(1, catId);
        ResultSet set = p.executeQuery();

        while (set.next()) {

            int pid = set.getInt("pid");
            String pTitle = set.getString("pTitle");
            String pContent = set.getString("pContent");
            String pCode = set.getString("pCode");
            String pPic = set.getString("pPic");
            Timestamp date = set.getTimestamp("pDate");

            int userId = set.getInt("userId");
            Post post = new Post(pid, pTitle, pContent, pCode, pPic, date, catId, userId);

            list.add(post);
        }

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

```

```

    }
    return list;
}

public Post getPostByPostId(int postId) {
    Post post = null;
    String q = "select * from posts where pid=?";
    try {
        PreparedStatement p = this.con.prepareStatement(q);
        p.setInt(1, postId);

        ResultSet set = p.executeQuery();

        if (set.next()) {

            int pid = set.getInt("pid");
            String pTitle = set.getString("pTitle");
            String pContent = set.getString("pContent");
            String pCode = set.getString("pCode");
            String pPic = set.getString("pPic");
            Timestamp date = set.getTimestamp("pDate");
            int cid = set.getInt("catId");
            int userId = set.getInt("userId");
            post = new Post(pid, pTitle, pContent, pCode, pPic, date, cid, userId);

        }

    } catch (Exception e) {
        e.printStackTrace();
    }
    return post;
}
}

```



## **LikeDao**

```
package com.tech.blog.dao;
```

```
import java.sql.*;
```

```
public class LikeDao {
```

```
    Connection con;
```

```
    public LikeDao(Connection con) {
```

```
        this.con = con;
```

```
    }
```

```
    public boolean insertLike(int pid, int uid) {
```

```
        boolean f = false;
```

```
        try {
```

```
            String q = "insert into liked(pid,uid)values(?,?)";
```

```
            PreparedStatement p = this.con.prepareStatement(q);
```

```
            //values set...
```

```
            p.setInt(1, pid);
```

```
            p.setInt(2, uid);
```

```
            p.executeUpdate();
```

```
            f = true;
```

```
        } catch (Exception e) {
```

```
            e.printStackTrace();
```

```
        }
```

```
        return f;
```

```
    }
```

```
    public int countLikeOnPost(int pid) {
```

```
int count = 0;
```

```
String q = "select count(*) from liked where pid=?";
```

```
try {
```

```
    PreparedStatement p = this.con.prepareStatement(q);
```

```
    p.setInt(1, pid);
```

```
    ResultSet set = p.executeQuery();
```

```
    if (set.next()) {
```

```
        count = set.getInt("count(*)");
```

```
    }
```

```
} catch (Exception e) {
```

```
    e.printStackTrace();
```

```
}
```

```
return count;
```

```
}
```

```
public boolean isLikedByUser(int pid, int uid) {
```

```
    boolean f = false;
```

```
    try {
```

```
        PreparedStatement p = this.con.prepareStatement("select * from liked where pid=?  
and uid=?");
```

```
        p.setInt(1, pid);
```

```
        p.setInt(2, uid);
```

```
        ResultSet set = p.executeQuery();
```

```
        if (set.next()) {
```

```
            f = true;
```

```
        }
```

```
} catch (Exception e) {
```

```
}
```

```
return f;
```

```

    }

    public boolean deleteLike(int pid, int uid) {
        boolean f = false;
        try {
            PreparedStatement p = this.con.prepareStatement("delete from liked where pid=? and uid=? ");
            p.setInt(1, pid);
            p.setInt(2, uid);
            p.executeUpdate();
            f = true;
        } catch (Exception e) {
            e.printStackTrace();
        }

        return f;
    }
}

```

### **com.tech.blog.entities**

#### **Category.java**

```

package com.tech.blog.entities;

public class Category {
    private int cid;
    private String name;
    private String description;

    public Category(int cid, String name, String description) {
        this.cid = cid;
        this.name = name;
        this.description = description;
    }
}

```

```
public Category() {  
    }
```

```
public Category(String name, String description) {  
    this.name = name;  
    this.description = description;  
}
```

```
public int getCid() {  
    return cid;  
}
```

```
public void setCid(int cid) {  
    this.cid = cid;  
}
```

```
public String getName() {  
    return name;  
}
```

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

```
public String getDescription() {  
    return description;  
}
```

```
public void setDescription(String description) {  
    this.description = description;  
}  
}
```

### **Message.java**

```
package com.tech.blog.entities;
```

```
public class Message {
```

```
    private String content;
```

```
    private String type;
```

```
    private String cssClass;
```

```
    public Message(String content, String type, String cssClass) {
```

```
        this.content = content;
```

```
        this.type = type;
```

```
        this.cssClass = cssClass;
```

```
    }
```

```
// getters and setters
```

```
    public String getContent() {
```

```
        return content;
```

```
    }
```

```
    public void setContent(String content) {
```

```
        this.content = content;
```

```
    }
```

```
    public String getType() {
```

```
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getCssClass() {
        return cssClass;
    }

    public void setCssClass(String cssClass) {
        this.cssClass = cssClass;
    }
}
```

### **Post.java**

```
package com.tech.blog.entities;

import java.sql.*;

public class Post {

    private int pid;
    private String pTitle;
    private String pContent;
```

```
private String pCode;  
private String pPic;  
private Timestamp pDate;  
private int catId;  
private int userId;
```

```
public Post() {  
}
```

```
public Post(int pid, String pTitle, String pContent, String pCode, String pPic, Timestamp  
pDate, int catId, int userId) {  
    this.pid = pid;  
    this.pTitle = pTitle;  
    this.pContent = pContent;  
    this.pCode = pCode;  
    this.pPic = pPic;  
    this.pDate = pDate;  
    this.catId = catId;  
    this.userId = userId;  
}
```

```
public Post(String pTitle, String pContent, String pCode, String pPic, Timestamp pDate,  
int catId, int userId) {  
    this.pTitle = pTitle;  
    this.pContent = pContent;  
    this.pCode = pCode;  
    this.pPic = pPic;  
    this.pDate = pDate;  
    this.catId = catId;  
    this.userId = userId;
```

```
}
```

```
public int getPid() {  
    return pid;  
}
```

```
public void setPid(int pid) {  
    this.pid = pid;  
}
```

```
public String getpTitle() {  
    return pTitle;  
}
```

```
public void setpTitle(String pTitle) {  
    this.pTitle = pTitle;  
}
```

```
public String getpContent() {  
    return pContent;  
}
```

```
public void setpContent(String pContent) {  
    this.pContent = pContent;  
}
```

```
public String getpCode() {  
    return pCode;  
}
```



```
public void setpCode(String pCode) {  
    this.pCode = pCode;  
}
```

```
public String getpPic() {  
    return pPic;  
}
```

```
public void setpPic(String pPic) {  
    this.pPic = pPic;  
}
```

```
public Timestamp getpDate() {  
    return pDate;  
}
```

```
public void setpDate(Timestamp pDate) {  
    this.pDate = pDate;  
}
```

```
public int getCatId() {  
    return catId;  
}
```

```
public void setCatId(int catId) {  
    this.catId = catId;  
}
```

```
    public int getUserId() {  
        return userId;  
    }  
  
    public void setUserId(int userId) {  
        this.userId = userId;  
    }  
  
}
```

### **User.java**

```
package com.tech.blog.entities;
```

```
import java.sql.*;
```

```
public class User {
```

```
    private int id;  
    private String name;  
    private String email;  
    private String password;  
    private String gender;  
    private Timestamp dateTime;  
    private String about;  
    private String profile;
```

```
    //constructor
```

```
    public User(int id, String name, String email, String password, String gender, Timestamp  
dateTime, String about) {
```

```
        this.id = id;  
        this.name = name;  
        this.email = email;  
        this.password = password;
```

```
        this.gender = gender;
        this.dateTime = dateTime;
        this.about = about;
    }
    //default constructor
    public User() {
    }
    //except id
    public User(String name, String email, String password, String gender, String about) {
        this.name = name;
        this.email = email;
        this.password = password;
        this.gender = gender;
        this.about = about;
    }

    // getters and setters

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

```
public String getEmail() {  
    return email;  
}
```

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

```
public String getPassword() {  
    return password;  
}
```

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

```
public String getGender() {  
    return gender;  
}
```

```
public void setGender(String gender) {  
    this.gender = gender;  
}
```

```
public Timestamp getDateTime() {  
    return dateTime;  
}
```

```
public void setDateTime(Timestamp dateTime) {  
    this.dateTime = dateTime;  
}
```

```
public String getAbout() {  
    return about;  
}  
  
public void setAbout(String about) {  
    this.about = about;  
}  
  
public String getProfile() {  
    return profile;  
}  
  
public void setProfile(String profile) {  
    this.profile = profile;  
}  
}
```

### **com.tech.blog.helper**

#### **ConnectionProvider.java**

```
package com.tech.blog.helper;  
import java.sql.*;  
public class ConnectionProvider {  
    private static Connection con;  
    public static Connection getConnection(){  
        try{  
            if(con==null){  
                //driver class load  
                Class.forName("com.mysql.jdbc.Driver");  
                //create a connection  
  
                con=DriverManager.getConnection("jdbc:mysql://localhost:3306/techblog","root","hmk506002");  
            }  
        }  
    }  
}
```

```
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
        return con;  
    }  
}
```

### **Helper.java**

```
package com.tech.blog.helper;
```

```
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.InputStream;
```

```
public class Helper {
```

```
    public static boolean deleteFile(String path) {
```

```
        boolean f = false;
```

```
        try {
```

```
            File file = new File(path);
```

```
            f = file.delete();
```

```
        } catch (Exception e) {
```

```
            e.printStackTrace();
```

```
        }
```

```

        return f;

    }

    public static boolean saveFile(InputStream is, String path) {
        boolean f = false;

        try {
            byte b[] = new byte[is.available()];

            is.read(b);

            FileOutputStream fos = new FileOutputStream(path);

            fos.write(b);

            fos.flush();
            fos.close();
            f = true;

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

        return f;
    }
}

mystyle.css
body{

}

.primary-background{

```

```
        background:black;

    }

    .s{
        max-width: 80%;
        height: auto;
    }

    .y{
        background-color: #FFD400;
    }
```

### **com.tech.blog.servlets;**

#### **AddPostServlet.java**

```
import com.tech.blog.dao.PostDao;
import com.tech.blog.entities.Post;
import com.tech.blog.entities.User;
import com.tech.blog.helper.ConnectionProvider;
import com.tech.blog.helper.Helper;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

/**
 *
 * @author Durgesh
 */
@MultipartConfig
public class AddPostServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP GET and POST
     * 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()) {
        /* TODO output your page here. You may use following sample code. */

        int cid = Integer.parseInt(request.getParameter("cid"));
        String pTitle = request.getParameter("pTitle");
        String pContent = request.getParameter("pContent");
        String pCode = request.getParameter("pCode");
        Part part = request.getPart("pic");
//        getting currentuser id
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("currentUser");

//        out.println("your post title is " + pTitle);
//        out.println(part.getSubmittedFileName());
        Post p = new Post(pTitle, pContent, pCode, part.getSubmittedFileName(), null, cid,
user.getId());
        PostDao dao = new PostDao(ConnectionProvider.getConnection());
        if (dao.savePost(p)) {

            String path = request.getRealPath("/") + "blog_pics" + File.separator +
part.getSubmittedFileName();
            Helper.saveFile(part.getInputStream(), path);
            out.println("done");
        } else {
            out.println("error");
        }

    }
}

// <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>
}

```

### **AddPostServlet1.java**

```

package com.tech.blog.servlets;
import com.tech.blog.dao.PostDao;
import com.tech.blog.entities.Post;
import com.tech.blog.entities.User;
import com.tech.blog.helper.ConnectionProvider;
import com.tech.blog.helper.Helper;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

```

```

@MultipartConfig
public class AddPostServlet1 extends HttpServlet {

```

```

/**

```

```

* 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()) {
        /* TODO output your page here. You may use following sample code. */

        int cid = Integer.parseInt(request.getParameter("cid"));
        String pTitle = request.getParameter("pTitle");
        String pContent = request.getParameter("pContent");
        String pCode = request.getParameter("pCode");
        Part part = request.getPart("pic");
//        getting currentuser id
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("currentUser");

//        out.println("your post title is " + pTitle);
//        out.println(part.getSubmittedFileName());
        Post p = new Post(pTitle, pContent, pCode, part.getSubmittedFileName(), null, cid,
user.getId());
        PostDao dao = new PostDao(ConnectionProvider.getConnection());
        if (dao.savePost(p)) {

            String path = request.getRealPath("/") + "blog_pics" + File.separator +
part.getSubmittedFileName();
            Helper.saveFile(part.getInputStream(), path);
            out.println("done");
        } else {
            out.println("error");
        }
    }
}

// <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>
}

```

### **EditServlet.java**

```

/*
 * 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.
 */
package com.tech.blog.servlets;

import com.tech.blog.dao.UserDao;
import com.tech.blog.entities.Message;
import com.tech.blog.entities.User;
import com.tech.blog.helper.ConnectionProvider;
import com.tech.blog.helper.Helper;

```

```

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

```

@MultipartConfig

```
public class EditServlet extends HttpServlet {
```

```
    /**
```

```
    * 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()) {
```

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

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

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

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

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

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

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

```
//        fetch all data
```

```
        String userEmail = request.getParameter("user_email");
```

```
        String userName = request.getParameter("user_name");
```

```
        String userPassword = request.getParameter("user_password");
```

```
        String userAbout = request.getParameter("user_about");
```

```
        Part part = request.getPart("image");
```

```
        //image's name with extension
```

```
        String imageName = part.getSubmittedFileName();
```

```
        //get the user from the session...
```

```
        HttpSession s = request.getSession();
```

```
        User user = (User) s.getAttribute("currentUser");
```

```
        user.setEmail(userEmail);
```

```
        user.setName(userName);
```

```
        user.setPassword(userPassword);
```

```

user.setAbout(userAbout);
String oldFile = user.getProfile();

user.setProfile(imageName);

//update database....
UserDao userDao = new UserDao(ConnectionProvider.getConnection());

boolean ans = userDao.updateUser(user);
if (ans) {

    String path = request.getRealPath("/") + "pics" + File.separator + user.getProfile();

    //start of photo work
    //delete code
    String pathOldFile = request.getRealPath("/") + "pics" + File.separator + oldFile;

    if (!oldFile.equals("default.png")) {
        Helper.deleteFile(pathOldFile);
    }

    if (Helper.saveFile(part.getInputStream(), path)) {
        out.println("Profile updated...");
        Message msg = new Message("Profile details updated...", "success", "alert-
success");
        s.setAttribute("msg", msg);

    } else {
        ////////////
        Message msg = new Message("Something went wrong..", "error", "alert-
danger");
        s.setAttribute("msg", msg);
    }

    //end of photo work
} else {
    out.println("not updated..");
    Message msg = new Message("Something went wrong..", "error", "alert-danger");
    s.setAttribute("msg", msg);
}

response.sendRedirect("profile.jsp");

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>

}
```

### **LikeServlet.java**

```
package com.tech.blog.servlets;

import com.tech.blog.dao.LikeDao;
import com.tech.blog.helper.ConnectionProvider;
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;

/**
 *
 * @author Durgesh
 */
public class LikeServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP GET and POST
     * 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()) {
            /* TODO output your page here. You may use following sample code. */
            String operation = request.getParameter("operation");
            int uid = Integer.parseInt(request.getParameter("uid"));
            int pid = Integer.parseInt(request.getParameter("pid"));

            //      out.println("data from server");
            //      out.println(operation);
            //      out.println(uid);
            //      out.println(pid);
            LikeDao ldao = new LikeDao(ConnectionProvider.getConnection());
            if (operation.equals("like")) {
                boolean f=ldao.insertLike(pid, uid);
                out.println(f);
            }
        }
    }

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

}

```

### **LoginServlet.java**

```

/*
 * 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.
 */
package com.tech.blog.servlets;

import com.tech.blog.dao.UserDao;
import com.tech.blog.entities.Message;
import com.tech.blog.entities.User;
import com.tech.blog.helper.ConnectionProvider;
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 LoginServlet extends HttpServlet {

    /**
     * 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()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet LoginServlet</title>");
            out.println("</head>");
            out.println("<body>");
//            login
//            fetch username and password from request
            String userEmail = request.getParameter("email");
            String userPassword = request.getParameter("password");

            UserDao dao = new UserDao(ConnectionProvider.getConnection());

            User u = dao.getUserByEmailAndPassword(userEmail, userPassword);

            if (u == null) {
                //login.....
                //            error//
                //            out.println("Invalid Details..try again");
                Message msg = new Message("Invalid Details ! try with another", "error", "alert-
danger");
                HttpSession s = request.getSession();
                s.setAttribute("msg", msg);

                response.sendRedirect("login_page.jsp");
            } else {
                //.....
                //            login success
                HttpSession s = request.getSession();
                s.setAttribute("currentUser", u);

```

```

        response.sendRedirect("profile.jsp");

    }

    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>
}

```

## **LogoutServlet.java**

```
/*
 * 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.
 */
package com.tech.blog.servlets;

import com.tech.blog.entities.Message;
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 LogoutServlet extends HttpServlet {

    /**
     * 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()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet LogoutServlet</title>");
            out.println("</head>");
            out.println("<body>");

            HttpSession s = request.getSession();

            s.removeAttribute("currentUser");

            Message m = new Message("Logout Successfully", "success", "alert-success");

            s.setAttribute("msg", m);
```

```

        response.sendRedirect("login_page.jsp");

        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>
}

```

### **RegisterServlet.java**

```

/*
 * 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.
 */
package com.tech.blog.servlets;

import com.tech.blog.dao.UserDao;
import com.tech.blog.entities.User;
import com.tech.blog.helper.ConnectionProvider;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@MultipartConfig
public class RegisterServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP GET and POST
     * 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()) {
            /* TODO output your page here. You may use following sample code. */

            //      fetch all form data
            String check = request.getParameter("check");
            if (check == null) {
                out.println("box not checked");
            } else {
                //baki ka data yaha nikalna..
                String name = request.getParameter("user_name");
                String email = request.getParameter("user_email");
                String password = request.getParameter("user_password");
                String gender = request.getParameter("gender");
                String about = request.getParameter("about");
            }
        }
    }
}

```

```

        //create user object and set all data to that object..
        User user = new User(name, email, password, gender, about);

        //create a user daao object..
        UserDao dao = new UserDao(ConnectionProvider.getConnection());
        if (dao.saveUser(user)) {
//            save..
            out.println("done");
        } else {
            out.println("error");
        }
    }
}

// <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

```

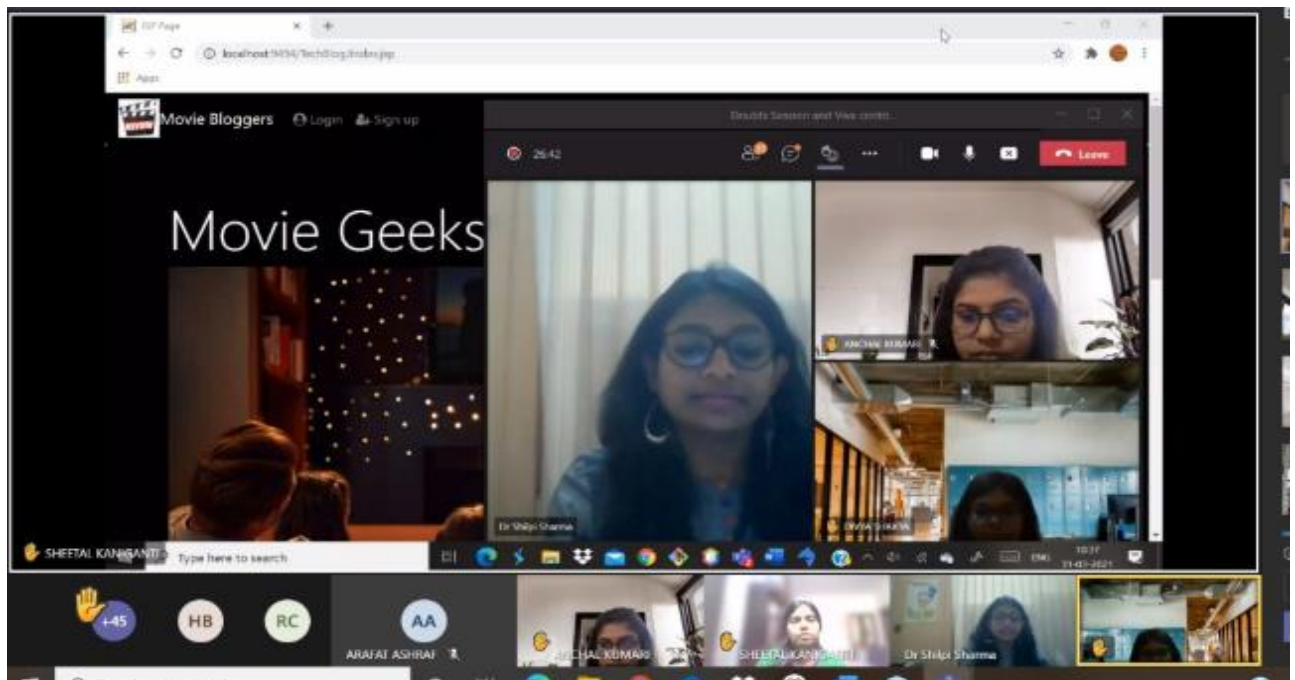
\*/

@Override

```
public String getServletInfo() {  
    return "Short description";  
}
```

}


**While presenting:**






## OUTPUT:

## HOME PAGE

 [Movie Bloggers](#) [Login](#) [Sign up](#)

# Movie Geeks



We Are Movie Geeks began as a simple project for fun, an online way for us to share our love of movies in our spare time.

We're happy to be here, we thank our readers for being a major factor in our success and allowing us this ongoing opportunity for sharing our passion with you and talking all things movies... from the minds of the Movie Geeks.

[Movie lovers, Share your thoughts here!](#) [Login Here](#)



[Watch A Movie](#)

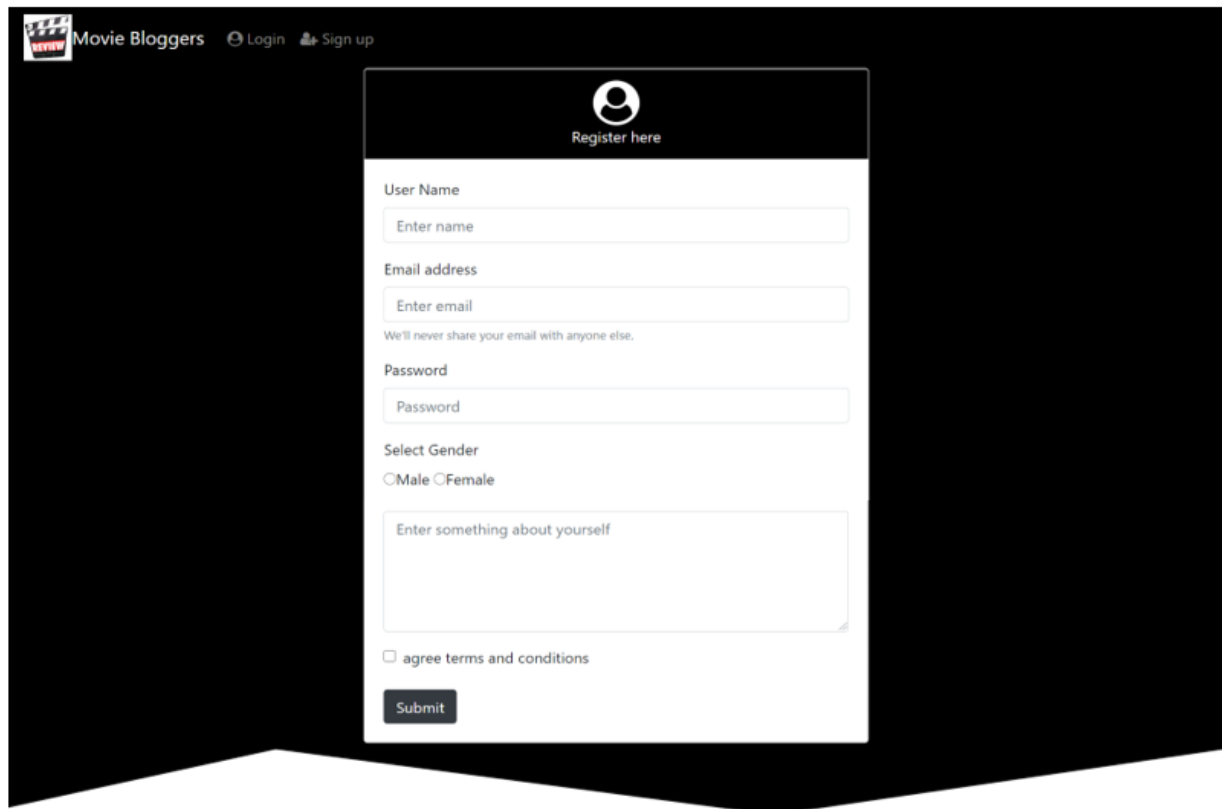


[Post Your Review](#)



[See Other's Reviews](#)

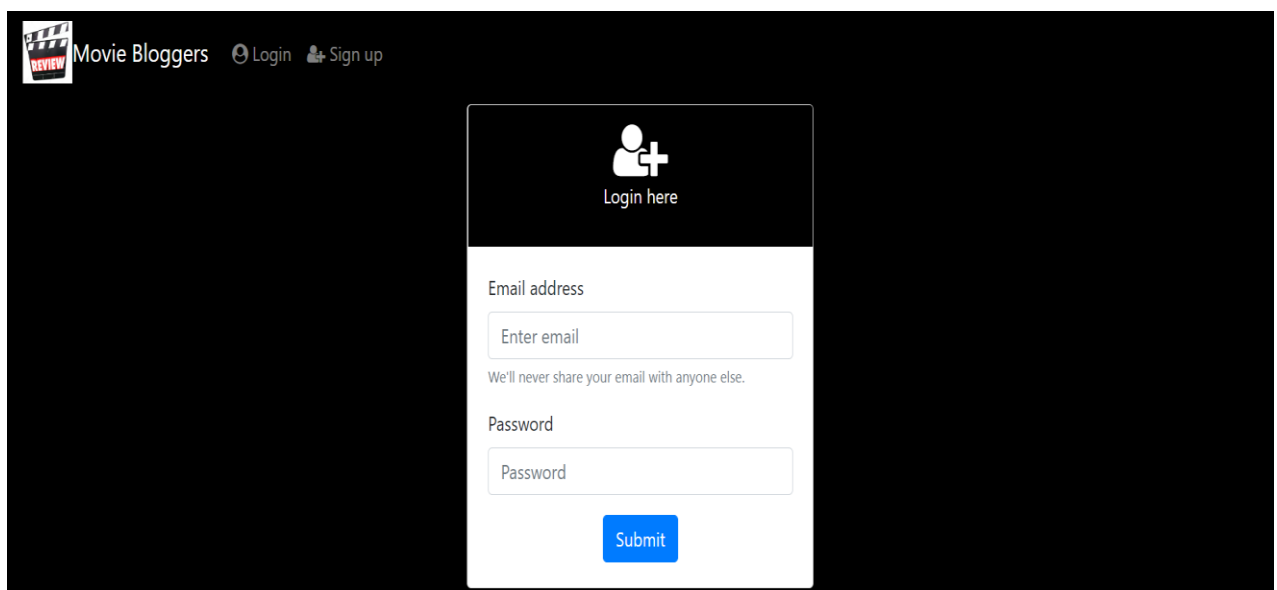
## REGISTER PAGE



The image shows a registration form for 'Movie Bloggers' on a dark background. The form is centered and contains the following elements:

- Header:** A logo with a clapperboard icon and the text 'Movie Bloggers', followed by 'Login' and 'Sign up' links.
- Form Header:** A dark box with a user icon and the text 'Register here'.
- Fields:**
  - User Name:** A text input field with the placeholder 'Enter name'.
  - Email address:** A text input field with the placeholder 'Enter email'. Below it is a note: 'We'll never share your email with anyone else.'
  - Password:** A text input field with the placeholder 'Password'.
  - Select Gender:** Radio buttons for 'Male' and 'Female'.
  - Text Area:** A larger text input field with the placeholder 'Enter something about yourself'.
- Footer:** A checkbox labeled 'agree terms and conditions' and a 'Submit' button.


## LOGIN PAGE




The image shows a login form for 'Movie Bloggers' on a dark background. The form is centered and contains the following elements:

- Header:** A logo with a clapperboard icon and the text 'Movie Bloggers', followed by 'Login' and 'Sign up' links.
- Form Header:** A dark box with a user icon and the text 'Login here'.
- Fields:**
  - Email address:** A text input field with the placeholder 'Enter email'. Below it is a note: 'We'll never share your email with anyone else.'
  - Password:** A text input field with the placeholder 'Password'.
- Footer:** A blue 'Submit' button.

## CONSTRAINTS APPLY IN LOGIN PAGE


 Movie Bloggers [Login](#) [Sign up](#)


  
Login here


Email address

We'll never share your email with anyone else.

Password

 Please fill in this field.

 Movie Bloggers [Login](#) [Sign up](#)

  
Login here

Invalid Details ! try with another

Email address

We'll never share your email with anyone else.

Password

## DOPOST PAGE

Movie Bloggers

Do Post

Anchal Logout

All Posts

Action

Fiction

Comedy

Drama

Fantasy

Biopic

Documentaries

Suspense

Horror

Documentaries

Suspense

Horror

Provide the post details..

Action

Thor

The Marvel Cinematic Universe boldly straddles sci-fi, comic-book action and fantasy – never more so than in the 'Thor' movies, with their Tolkein-influenced take on Norse mythology and outrageous 'Flash Gordon'-style fetish costumes. 'Thor' is essentially a reboot of 'Masters of the Universe' – bulging hero heads to Earth to battle skeletal psychopath – but with better special effects and more nod-wink humour.

Magic moment: The glistening CG cityscape of Asgard could've come straight from a mid-70s Rick Wakeman LP cover.

Select your pic..

Choose file thor.jfif

Post

Munna Bhai MBBS

URI

THE SURGICAL STRIKE

11th JAN 2019

YEH NAYA HINDUSTAN HAI

YEH SHAHI NAHI DINDINDA BHI AUR MAARDA BHI

URI

THE SURGICAL STRIKE

11th JAN 2019

YEH NAYA HINDUSTAN HAI

YEH SHAHI NAHI DINDINDA BHI AUR MAARDA BHI

Movie Bloggers

Do Post

Anchal Logout

All Posts

Action

Fiction

Comedy

Drama

Fantasy

Biopic

Documentaries

Suspense

Horror

Good job!

saved successfully

OK

The Marvel Cinematic Universe boldly straddles sci-fi, comic-book action and fantasy – never more so than in the 'Thor' movies, with their Tolkein-influenced take on Norse mythology and outrageous 'Flash Gordon'-style fetish costumes. 'Thor' is essentially a reboot of 'Masters of the Universe' – bulging hero heads to Earth to battle skeletal

Choose file thor.jfif

Post

Munna Bhai MBBS

URI

THE SURGICAL STRIKE

11th JAN 2019

YEH NAYA HINDUSTAN HAI

YEH SHAHI NAHI DINDINDA BHI AUR MAARDA BHI

URI

THE SURGICAL STRIKE

11th JAN 2019

YEH NAYA HINDUSTAN HAI

YEH SHAHI NAHI DINDINDA BHI AUR MAARDA BHI

## READ OTHERS POST

Movie Bloggers

Do Post

abc

Logout

All Posts

Action

Fiction

Comedy

Drama

Fantasy


Biopic

Documentaries

Suspense

Horror

Adventure




### Munna Bhai MBBS

Munna Bhai MBBS taught me MSL Movie Hai Sanjay Dutt as Munna Bhai Arshad Warsi as Circuit & Many More. Munna is a goon who sets out to fulfil his father's dream of becoming a doctor. With some help from his ally, Circuit, he enrolls himself in a medical college and drives Dr Asthana up the wall. Director by Rajkumar Hirani One year, however, Munna's plan goes awry when his father meets an old acquaintance, Dr. J. C. Asthana (Boman Irani) and the two older men decide to betroth Munna to Asthana's daughter, Chinku. At this point the truth about Munna is revealed. Asthana insults Munna's parents and calls them "fools" for being ignorant of Munna's real life. Munna's father and mother, aghast and later heartbroken, leave for their village.

4

Read More...

20



### URR: THE SURGICAL STRIKE MOVIE REVIEW

Aditya Charrat's unforgiving war drama incorporates the events that led to the surgical strikes as seen through the eyes of protagonist Major Vhaan Singh Shergill (Vicky Kaushal). To make things harder for him, he has personal battles to fight at home as well. First things first, Vicky Kaushal is on a roll. Interestingly, after playing a valiant Pakistani Army officer in Raazi, here he switches sides and plays an invincible Para (Special Forces) Commando, Indian Army. Justifying the hype around him, the actor continues to grow from strength to strength. His sincere and effortless presence adds depth to this film, that otherwise lacks the palpable tension you expect from a war drama. What makes it then engaging is not its execution, but the audacity of the mission it dramatically decodes and recreates. Despite knowing the result, you watch the events unfold with childlike intrigue as the complex operation plan was classified. The rigorous process of how 80 Indian Para SF commandos managed to infiltrate PoK and destroy the terror camps, makes for an instructive watch if not gripping.

4

Read More...

20

## EDIT PROFILE

Movie Bloggers

Do Post

Movie Review

abc Logout

Profile details updated...

All Posts

Action

Fiction

Comedy

Drama


Fantasy

Biopic

Documentaries

Suspense

Horror



abc

ID : 12

Email : abc@g.com

Gender : male

Status :

Registered on : 2021-03-28 19:06:37.0

Close

EDIT

URI

THE SURGICAL STRIKE

11th JAN 2019

YEH NAYA HINDUSTAN HAI

YEH KHAR MEH DHISEDA DHI AJN MAAREGA DHI

Munna Bhai MBBS Bought hi MST Movie

Movie Bloggers

Do Post

Movie Review

abc Logout

Profile details updated...

All Posts

Action

Fiction

Comedy

Drama

Fantasy

Biopic

Documentaries

Suspense

Action

Fiction

Comedy

Drama

Fantasy

Biopic

Documentaries

Suspense



abc

Please Edit Carefully

ID : 12

Email : abc@g.com

Name : abc

Password : ...

Gender : MALE

About :

New Profile: Choose file No file chosen

Save

Close

Back

URI

THE SURGICAL STRIKE

11th JAN 2019


YEH NAYA HINDUSTAN HAI

YEH KHAR MEH DHISEDA DHI AJN MAAREGA DHI


RECOMMENDATIONS



## LOGOUT PAGE

Movie Bloggers

[Login](#) [Sign up](#)

  
Login here

Logout Successfully

Email address

We'll never share your email with anyone else.

Password

## FACEBOOK PLUGIN PAGE

[illegible]

## DATABASE FOR SIGNUP PAGE

1 Messages 2 Table Data 3 Info

Limit rows First row 0 # of rows 1000

<input type="checkbox"/>	id	name	email	password	gender	about	rdate	profile
<input type="checkbox"/>	1	Anchal	a@gmail.com	anchal	female	I like watching action movies!	2021-03-26 23:08:47	a.JPG
<input type="checkbox"/>	2	Divya	divya@gmail.com	divya	female	I like Fantasy movies!	2021-03-26 23:47:35	d.JPG
<input type="checkbox"/>	3	Sheetal	sh@gmail.com	sheetal	female	I like horror movies!	2021-03-26 23:56:15	s.JPG
<input type="checkbox"/>	4	Rohit	r@gmail.com	rohit	male	I like biopic movies!	2021-03-27 00:04:09	
<input type="checkbox"/>	5	Ansh	an@gmail.com	ansh	male		2021-03-28 19:01:04	default.png
<input type="checkbox"/>	6	abc	abc@g.com	abc	male		2021-03-28 19:06:37	c.JPG
*	(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	'Hey! Providing my movie reviews.'	CURRENT_TIMESTAMP	default.png

## DATABASE FOR DOPOST

<input type="checkbox"/>	pid	pTitle	pContent	pCode	pPic	pDate	catId	userId	
<input type="checkbox"/>	1	URI: THE SURGICAL STRIKE MOVIE REVIEW	Aditya Thakre unforgiving war drama incorporates the even...	1K	The film is based on the surgical strikes com	uri.jpg	2021-03-28 20:02:10	3	2
<input type="checkbox"/>	2	Munna Bhai MBBS	Munna Bhai MBBS Bought hi MST Movie Hai Sanjay Dutt as Mu...	746B	Munna, in grief and despair, decides that the	munna-bhai-mbbs-306x393.jpg	2021-03-29 19:59:00	5	3
<input type="checkbox"/>	3	Thor	The Marvel Cinematic Universe boldly straddles sci-fi, co...	469B	Magic moment: The glistening CG cityscape of	thor.jfif	2021-03-29 20:37:20	3	1
<input type="checkbox"/>	4	Harry Potter And The Sorcerer's Stone	"Harry Potter and the Sorcerer's Stone" is a red-blooded...	780B	Scary, yes, but not too scary--just scary end	harryp.jfif	2021-03-29 20:45:15	12	3
*	(Auto)	(NULL)	(NULL)	(K)	(NULL)	(NULL)	CURRENT_TIMESTAMP	(NULL)	(NULL)

## DATABASE FOR LIKE

1 Messages 2 Table Data

<input type="checkbox"/>	lId	pId	uId
<input type="checkbox"/>	13	1	2
<input type="checkbox"/>	14	1	2
<input type="checkbox"/>	15	1	2
<input type="checkbox"/>	16	1	2
<input type="checkbox"/>	17	1	2
<input type="checkbox"/>	18	2	2
<input type="checkbox"/>	19	2	6
<input type="checkbox"/>	20	2	6
<input type="checkbox"/>	21	2	6
<input type="checkbox"/>	22	3	1
<input type="checkbox"/>	23	3	1
<input type="checkbox"/>	24	3	1
<input type="checkbox"/>	25	3	1
*	(Auto)	(NULL)	(NULL)

## DATABASE FOR MOVIES CATEGORIES

1 Messages 2 Table Data 3 Info

<input type="checkbox"/>	cid	name	description
<input type="checkbox"/>	3	Action	
<input type="checkbox"/>	4	Fiction	(NULL)
<input type="checkbox"/>	5	Comedy	(NULL)
<input type="checkbox"/>	6	Drama	(NULL)
<input type="checkbox"/>	7	Fantasy	(NULL)
<input type="checkbox"/>	8	Biopic	(NULL)
<input type="checkbox"/>	9	Documentaries	(NULL)
<input type="checkbox"/>	10	Suspense	(NULL)
<input type="checkbox"/>	11	Horror	(NULL)
<input type="checkbox"/>	12	Adventure	(NULL)
*	(Auto)	(NULL)	(NULL)



