

Practical No.6

Prac 6.1

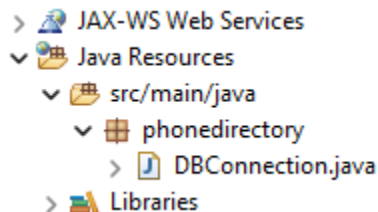
Aim: Create a Telephone directory using JSP and store all the information within a database, so that later could be retrieved as per the requirement.

Code:

Create a database and create table

```
CREATE DATABASE PhoneDirectory;  
USE PhoneDirectory;  
CREATE TABLE Contacts (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    phone VARCHAR(20),  
    email VARCHAR(255)  
);
```

Create a java class in



DBConnection.java:

```
package phonedirectory;
```

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;
```

```
public class DBConnection {  
    public static Connection getConnection() throws SQLException {  
        String url = "jdbc:mysql://localhost:3306/telephone_directory";  
        String username = "root";  
        String password = "password";  
  
        Connection connection = null;  
  
        try {  
            connection = DriverManager.getConnection(url, username, password);  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```

        throw e; // You may want to handle this exception more gracefully in a real application
    }

    return connection;
}
}

```

Add_contact.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.*"%>
<%@ page import="phonedirectory.DBConnection"%><%
Connection connection = DBConnection.getConnection();
if (request.getMethod().equals("POST")) {
String name = request.getParameter("name");
String phone = request.getParameter("phone");
String email = request.getParameter("email");
if (name != null && phone != null && email != null) {
try {
// Insert the new contact into the database
String insertQuery = "INSERT INTO contacts (name, phone, email) VALUES (?, ?, ?)";
PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);
preparedStatement.setString(1, name);
preparedStatement.setString(2, phone);
preparedStatement.setString(3, email);
preparedStatement.executeUpdate();
preparedStatement.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
}
// Close the database connection
if (connection != null) {
connection.close();
}
%>
<html>
<body>
<h1>Add a New Contact</h1>
<form action="add_contact.jsp" method="post">
<label>Name: <input type="text" name="name"></label><br>
<label>Phone: <input type="text" name="phone"></label><br>
<label>Email: <input type="text" name="email"></label><br>
<input type="submit" value="Add Contact">

```

```

<a href="display_contacts.jsp">show data</a>
</form>
</body>
</html>

```

Display_contact.jsp:

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="phonedirectory.DBConnection" %>
<!DOCTYPE html>
<html>
<head>
<title>Display Users</title>
</head>
<body>
<h1>Registered Users</h1>
<table border="1">
<tr>
<th>User ID</th>
<th>name</th>
<th>phone</th>
<th>Email</th>
<th>Update</th>
</tr>
<%
try {
// Establish a database connection
Connection connection = DBConnection.getConnection();
// Create and execute an SQL SELECT statement
String selectQuery = "SELECT * FROM contacts";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(selectQuery);
while (resultSet.next()) {
int userId = resultSet.getInt("id");
String username = resultSet.getString("name");
String phone = resultSet.getString("phone");
String email = resultSet.getString("email");
%>
<tr>
<td><%= userId %></td>
<td><%= username %></td>
<td><%= phone %></td>
<td><%= email %></td>
<td>
<a href="edit_contact.jsp?id=<%= userId %>">Edit</a>
<a href="delete_contact.jsp?id=<%= userId %>">Delete</a>

```

```

</td>
</tr>
<%
}
resultSet.close();
statement.close();
connection.close();
} catch (SQLException e) {
// Handle database errors here
out.println("Database error: " + e.getMessage());
}
%>
</table>
</body>
</html>

```

Edit_contacts.jsp:

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="phonedirectory.DBConnection" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Edit Contact</title>
</head>
<body>
<h1>Edit Contact</h1>
<%
Connection connection = null;
PreparedStatement preparedStatement = null;
ResultSet resultSet = null;
try {
// Establish a database connection
connection = DBConnection.getConnection();
// Check if an ID parameter is provided in the URL
String idParam = request.getParameter("id");
int contactId = -1;
if (idParam != null) {
contactId = Integer.parseInt(idParam);
// Retrieve the contact's current details
String selectQuery = "SELECT * FROM contacts WHERE id=?";
preparedStatement = connection.prepareStatement(selectQuery);
preparedStatement.setInt(1, contactId);
resultSet = preparedStatement.executeQuery();

```

```

if (resultSet.next()) {
    String currentName = resultSet.getString("name");
    String currentPhone = resultSet.getString("phone");
    String currentEmail = resultSet.getString("email");
    %>
    <form action="update_contact.jsp" method="post">
    <input type="hidden" name="id" value="<%= contactId %>">
    Name: <input type="text" name="name" value="<%= currentName %>"><br>
    Phone: <input type="text" name="phone" value="<%= currentPhone %>"><br>
    Email: <input type="text" name="email" value="<%= currentEmail %>"><br>
    <input type="submit" value="Update Contact">
    </form>
    <%
    }
    }
    } catch (SQLException e) {
    e.printStackTrace();
    } finally {
    // Close database resources individually
    if (resultSet != null) resultSet.close();
    if (preparedStatement != null) preparedStatement.close();
    if (connection != null) connection.close();
    }
    %>
</body>
</html>

```

update_contact.jsp:

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="phonedirectory.DBConnection" %>
<!DOCTYPE html>
<html>
<head>
<title>Update Contact</title>
</head>
<body>
<h1>Update Contact</h1>
<%
Connection connection = null;
PreparedStatement preparedStatement = null;
try {
    // Establish a database connection
    connection = DBConnection.getConnection();
    // Get the data submitted from the form
    int contactId = Integer.parseInt(request.getParameter("id"));

```

```

String newName = request.getParameter("name");
String newPhone = request.getParameter("phone");
String newEmail = request.getParameter("email");
// Update the contact's information in the database
String updateQuery = "UPDATE contacts SET name=?, phone=?, email=? WHERE id=?";
preparedStatement = connection.prepareStatement(updateQuery);
preparedStatement.setString(1, newName);
preparedStatement.setString(2, newPhone);
preparedStatement.setString(3, newEmail);
preparedStatement.setInt(4, contactId);
preparedStatement.executeUpdate();
%>
<p>Contact updated successfully.</p>
<a href="display_contacts.jsp">Back to Contacts</a>
<%
} catch (SQLException e) {
e.printStackTrace();
%>
<p>An error occurred while updating the contact: <%= e.getMessage() %></p>
<%
} finally {
// Close database resources individually
if (preparedStatement != null) preparedStatement.close();
if (connection != null) connection.close();
}
%>
</body>
</html>

```

Delete_contact.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.*"%>
<%@ page import="phonedirectory.DBConnection"%>
<%
Connection connection = DBConnection.getConnection();
// Check if an ID parameter is provided in the URL
String idParam = request.getParameter("id");
if (idParam != null) {
int contactId = Integer.parseInt(idParam);
try {
// Delete the contact from the database
String deleteQuery = "DELETE FROM contacts WHERE id=?";
PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);
preparedStatement.setInt(1, contactId);
preparedStatement.executeUpdate();

```

```

preparedStatement.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
// Close the database connection
if (connection != null) {
connection.close();
}
%>
<html>
<body>
<h1>Contact Deleted</h1>
<p>The contact has been deleted successfully.</p>
<a href="display_contacts.jsp">Back to Contacts</a>
</body>
</html>

```

Output:

Add a New Contact

Name:

Phone:

Email:

[show data](#)

Edit Contact

Name:

Phone:

Email:

Registered Users

User ID	name	phone	Email	Update
7	manoranjana	9892600158	m@gmail.com	Edit Delete
9	manoranjana	9892600158	m@gmail.com	Edit Delete

[Back to register](#)

Update Contact

Contact updated successfully.

[Back to Contacts](#)

Contact Deleted

The contact has been deleted successfully.

[Back to Contacts](#)

Registered Users

User ID	name	phone	Email	Update
7	manoranjana baral	9892600158	m@gmail.com	Edit Delete

[Back to register](#)

Prac 6.2

Aim: Write a JSP page to display the Registration form.

Code:

Register.jsp:

```
<!DOCTYPE html>
<html>
<head>
<title>Registration Form</title>
</head>
<body>
<h1>Registration Form</h1>
<form action="conn.jsp" method="post">
<label for="username">Username:</label>
<input type="text" id="username" name="username" required><br><br>
<label for="email">Email:</label>
<input type="email" id="email" name="email" required><br><br>
<label for="password">Password:</label>
<input type="password" id="password" name="password" required><br><br>
<label for="confirmPassword">Confirm Password:</label>
<input type="password" id="confirmPassword" name="confirmPassword" required><br><br>
<input type="submit" value="Register" name="btn_add">
</form>
</body>
</html>
```

Conn.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.sql.*" %>
<!DOCTYPE html>
<html>
<head>
<title>Registration Result</title>
</head>
<body>
<h1>Registration Result</h1>
<%
String username = request.getParameter("username");
String email = request.getParameter("email");
String password = request.getParameter("password");
String confirmPassword = request.getParameter("confirmPassword");
```

```
if (request.getParameter("btn_add")!=null) {
    try {
        // Establish a database connection
        String dbUrl = "jdbc:mysql://localhost:3306/penta";
        String dbUser = "root";
        String dbPassword = "password";
        Connection connection = DriverManager.getConnection(dbUrl, dbUser, dbPassword);
        // Create and execute an SQL INSERT statement
        String insertQuery = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
        PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);
        preparedStatement.setString(1, username);
        preparedStatement.setString(2, email);
        preparedStatement.setString(3, password);
        preparedStatement.executeUpdate();
        connection.close();
    } catch (SQLException e) {
        out.println("Database error: " + e.getMessage());
    }
} else if(request.getParameter("btn_show")!=null){
    out.println("Passwords do not match. Please try again.");
}

```

Output:

Registration Form

Username:

Email:

Password:

Confirm Password:

Registration Result

Registration successful!

Prac 6.3

Aim: Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table.

Code:

```
DBConnection.java:
package studentmaster;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    public static Connection getConnection() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/studentmaster";
        String username = "root";
        String password = "password";

        Connection connection = null;

        try {
            connection = DriverManager.getConnection(url, username, password);
        } catch (SQLException e) {
            e.printStackTrace();
            throw e; // You may want to handle this exception more gracefully in a real application
        }

        return connection;
    }
}
```

Add_student.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.*"%>
<%@ page import="studentmaster.DBConnection"%>
<%
    Connection connection = DBConnection.getConnection();
    boolean ia = false;
    if (request.getMethod().equals("POST")) {
        String RollNo =request.getParameter("RollNo");
```

```

String Name = request.getParameter("Name");
String Semester = request.getParameter("Semester");
String Course = request.getParameter("Course");
if (RollNo != null && Name != null && Semester != null && Course != null) {
    try {
        // Insert the new contact into the database
        String insertQuery = "INSERT INTO students (RollNo,Name,Semester, Course) VALUES (?, ?, ?,?)";
        PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);
        preparedStatement.setString(1, RollNo);
        preparedStatement.setString(2, Name);
        preparedStatement.setString(3, Semester);
        preparedStatement.setString(4, Course);
        ia = preparedStatement.executeUpdate() > 0;
        preparedStatement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
// Close the database connection
if (connection != null) {
    connection.close();
}
%>
<html>
<body>
<h1>Add a New student</h1>
<form action="add_student.jsp" method="post">
    <label>RollNO: <input type="number" name="RollNo"></label><br><br>
    <label>Name: <input type="text" name="Name"></label><br><br>
    <label>Semester: <input type="text" name="Semester"></label><br><br>
    <label>Course: <input type="text" name="Course"></label><br><br>
    <input type="submit" value="Add Student">
    <a href="display_student.jsp">show data</a>
</form>
<div>
<% if (ia) { %>
<p>Data added successfully!</p>
<% } %>
</div>
</body>
</html>

```

Display_student.jsp

```

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

```

```

<%@ page import="java.sql.*" %>
<%@ page import="studentmaster.DBConnection" %>
<!DOCTYPE html>
<html>
<head>
<title>Display Student Info</title>
</head>
<body>
<h1>Registered Students </h1>
<table border="1">
<tr>
<th>User ID</th>
<th>RollNo</th>
<th>Name</th>
<th>Semester</th>
<th>Course</th>
<th>Update</th>
</tr>
<%
try {
// Establish a database connection
Connection connection = DBConnection.getConnection();
// Create and execute an SQL SELECT statement
String selectQuery = "SELECT * FROM students";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(selectQuery);
while (resultSet.next()) {
int userId = resultSet.getInt("id");
Integer RollNo=resultSet.getInt("RollNo");
String username = resultSet.getString("Name");
String semester = resultSet.getString("Semester");
String course = resultSet.getString("Course");
%>
<tr>
<td><%= userId %></td>
<td><%= RollNo %></td>
<td><%= username %></td>
<td><%= semester %></td>
<td><%= course %></td>
<td>
<a href="edit_student.jsp?id=<%= userId %>">Edit</a>
<a href="delete_student.jsp?id=<%= userId %>">Delete</a>
</td>
</tr>
<%
}
resultSet.close();

```

```

statement.close();
connection.close();
} catch (SQLException e) {
// Handle database errors here
out.println("Database error: " + e.getMessage());
}
}%>
</table><br><br>
Back to register:<a href="add_student.jsp">link</a>
</body>
</html>

```

Edit_student.jsp:

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="studentmaster.DBConnection" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Edit Student Info</title>
</head>
<body>
<h1>Edit Student Info</h1>
<%
Connection connection = null;
PreparedStatement preparedStatement = null;
ResultSet resultSet = null;
try {
// Establish a database connection
connection = DBConnection.getConnection();
// Check if an ID parameter is provided in the URL
String idParam = request.getParameter("id");
int contactId = -1;
if (idParam != null) {
contactId = Integer.parseInt(idParam);
// Retrieve the contact's current details
String selectQuery = "SELECT * FROM students WHERE id=?";
preparedStatement = connection.prepareStatement(selectQuery);
preparedStatement.setInt(1, contactId);
resultSet = preparedStatement.executeQuery();
if (resultSet.next()) {
Integer RollNo=resultSet.getInt("RollNo");
String username = resultSet.getString("Name");
String semester = resultSet.getString("Semester");

```

```
String course = resultSet.getString("Course");
%>
<form action="update_student.jsp" method="post">
<input type="hidden" name="id" value="<%= contactId %>">
RollNo: <input type="number" name="RollNo" value="<%= RollNo %>"><br><br>
Name: <input type="text" name="Name" value="<%= username %>"><br><br>
Semester: <input type="text" name="Semester" value="<%= semester %>"><br><br>
Course: <input type="text" name="Course" value="<%= course %>"><br><br>
<input type="submit" value="Update Student">
</form>
<%
}
}
} catch (SQLException e) {
e.printStackTrace();
} finally {
// Close database resources individually
if (resultSet != null) resultSet.close();
if (preparedStatement != null) preparedStatement.close();
if (connection != null) connection.close();
}
%>
</body>
</html>
```

Update_student.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ page import="studentmaster.DBConnection" %>
<!DOCTYPE html>
<html>
<head>
<title>Update Student Info</title>
</head>
<body>
<h1>Update Student Info</h1>
<%
Connection connection = null;
PreparedStatement preparedStatement = null;
try {
// Establish a database connection
connection = DBConnection.getConnection();
// Get the data submitted from the form
int contactId = Integer.parseInt(request.getParameter("id"));
String newRollNo = request.getParameter("RollNo");
```



```

String newName = request.getParameter("Name");
String newsemester = request.getParameter("Semester");
String newcourse = request.getParameter("Course");
// Update the contact's information in the database
String updateQuery = "UPDATE students SET RollNo=?,Name=?, Semester=?, Course=? WHERE id=?";
preparedStatement = connection.prepareStatement(updateQuery);
preparedStatement.setString(1, newRollNo);
preparedStatement.setString(2, newName);
preparedStatement.setString(3, newsemester);
preparedStatement.setString(4, newcourse);
preparedStatement.setInt(5, contactId);
preparedStatement.executeUpdate();
%>
<p>Student Info updated successfully.</p>
Back to Registered Students:<a href="display_student.jsp">link</a>
<%
} catch (SQLException e) {
e.printStackTrace();
%>
<p>An error occurred while updating the contact: <%= e.getMessage() %></p>
<%
} finally {
// Close database resources individually
if (preparedStatement != null) preparedStatement.close();
if (connection != null) connection.close();
}
%>
</body>
</html>

```

Delete_student.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.*"%>
<%@ page import="studentmaster.DBConnection"%>
<%
Connection connection = DBConnection.getConnection();
// Check if an ID parameter is provided in the URL
String idParam = request.getParameter("id");
if (idParam != null) {
int contactId = Integer.parseInt(idParam);
try {
// Delete the contact from the database
String deleteQuery = "DELETE FROM students WHERE id=?";
PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);

```

```
preparedStatement.setInt(1, contactId);
preparedStatement.executeUpdate();
preparedStatement.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
// Close the database connection
if (connection != null) {
connection.close();
}
%>
<html>
<body>
<h1>Contact Deleted</h1>
<p>The Student data has been deleted successfully.</p>
Back to registered students:<a href="display_student.jsp">link</a>
</body>
</html>
```

Output:

Add a New student

RollNO:

Name:

Semester:

Course:

[show data](#)

Registered Students

User ID	RollNo	Name	Semester	Course	Update
6	1	Manoranjan	sem1	mca	Edit Delete
7	2	Monalisa	sem1	mca	Edit Delete

Back to register:[link](#)

Edit Student Info

RollNo:

Name:

Semester:

Course:

Update Student Info

Student Info updated successfully.

Back to Registered Students:[link](#)

Registered Students

User ID	RollNo	Name	Semester	Course	Update
6	1	Manoranjan Baral	sem1	mca	Edit Delete
7	2	Monalisa	sem1	mca	Edit Delete

Back to register:[link](#)

Contact Deleted

The Student data has been deleted successfully.

Back to registered students:[link](#)

Registered Students

User ID	RollNo	Name	Semester	Course	Update
6	1	Manoranjana Baral	sem1	mca	Edit Delete

Back to register:[link](#)

Prac 6.4

Aim: Design loan calculator using JSP which accepts Period of Time (in years) and Principal Loan Amount. Display the payment amount for each loan and then list the

loan balance and interest paid for each payment over the term of the loan for the following time period and interest rate:

- a. 1 to 7 year at 5.35%
- b. 8 to 15 year at 5.5%
- c. 16 to 30 year at 5.75%

Code:

```
<!DOCTYPE html>
<html>
<head>
<title>Simple Interest Calculator</title>
</head>
<body>
<h1>Simple Interest Calculator</h1>
<form>
Principal Amount: <input type="text" id="principal" required><br>
Rate of Interest (per annum): <input type="text" id="rate" required><br>
Time (in years): <input type="text" id="time" required><br>
<input type="button" value="Calculate" onclick="calculateSimpleInterest()">
</form>
<p id="result"></p>
<script>
function calculateSimpleInterest() {
var principal = parseFloat(document.getElementById("principal").value);
var rate = parseFloat(document.getElementById("rate").value);
var time = parseFloat(document.getElementById("time").value);
var simpleInterest = (principal * rate * time) / 100;
// Display the result in a traditional way
var resultMessage = "Simple Interest: " + simpleInterest;
var resultElement = document.getElementById("result");
resultElement.innerHTML = resultMessage;
}
</script>
</body>
</html>
```

Output:

Simple Interest Calculator

Principal Amount:

Rate of Interest (per annum):

Time (in years):

Simple Interest: 3640

Prac 6.5

Aim: Write a program using JSP that displays a webpage consisting Application form for change of Study Center which can be filled by any student who wants to change his/her study center. Make necessary assumptions

Code:

student_exchange.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Change Study Center Application</title>
</head>
<body>
<h1>Change Study Center Application</h1>
<form action="exchange.jsp" method="post">
<label for="studentName">Student Name:</label>
<input type="text" id="studentName" name="studentName" required><br><br>
<label for="currentCenter">Current Study Center:</label>
<input type="text" id="currentCenter" name="currentCenter" required><br><br>
<label for="newCenter">New Study Center:</label>
<input type="text" id="newCenter" name="newCenter" required><br><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Exchange.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.io.*, java.util.*" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Application Submitted</title>
</head>
<body>
<h1>Application Submitted</h1>
```

```
<p>Thank you for submitting your study center change application.</p>
<p>Student Name: <%= request.getParameter("studentName") %></p>
<p>Current Center: <%= request.getParameter("currentCenter") %></p>
<p>New Center: <%= request.getParameter("newCenter") %></p>
<!-- Perform database updates or other necessary actions here -->
</body>
</html>
```

Output:

Change Study Center Application

Student Name:

Current Study Center:

New Study Center:

Application Submitted

Thank you for submitting your study center change application.

Student Name: Manoranjan

Current Center: mumbai

New Center: pune

Prac 6.6

Aim: Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer.

Code:

Main.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Declaration, Scriptlet, Directive, Expression, Header, and Footer Example</title>
</head>
<body>
<%@ include file="header.jsp" %>
<center>
<%! int data=50; %>
<%= "Value of the variable is: " + data %>
<%! double circle(int n) { return 3.14 * n * n; } %>
<br>
<%= "Area of Circle is: " + circle(5) %>
<br>
<%! int rectangle(int l, int b) { return l * b; } %>
<%= "Area of rectangle is: " + rectangle(4, 6) %>
<br>
<%! int perimeter(int x, int y) {
int peri = 2 * (x + y);
return peri;
} %>
<br>
<%= "Perimeter of rectangle: " + perimeter(5, 6) %>
<br>
<p>Thanks for visiting my page</p>
</center>
<%@ include file="footer.jsp" %>
</body>
</html>
```

Header.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<%! int pageCount=0;
void addcount(){pageCount++;}%>
<% addcount();%>
<title>Jsp Declaration, Scriptlet,directive,expression,header and footer example</title>
</head>
<body>
<center>
<h2>
    The include Directive Example
</h2>
<p><b>This site has been visited <%= pageCount %>.</b></p>
</center>
</body>
</html>
```

Footer.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<center>
    <b>Manoranjan baral </b><br>
    <b>23-Mca-05</b>
</center>
</body>
</html>
```

Output:

The include Directive Example

This site been visited 1.

Value of the variable is: 50

Area of Circle is: 78.5

Area of rectangle is: 24

Perimeter of rectangle: 22

Thanks for visiting my page

Manoranjan baral

23-Mca-05

Practical No.7**Prac 7.1**

Aim: Write a program to print “Hello World” using spring framework.

Code:

Prac71new2Application.java:

```
package com.example.demo;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

@SpringBootApplication
public class Prac71new2Application {

    public static void main(String[] args) {
        try (AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(HelloWorldConfig.class)) {
            HelloWorldService helloWorldService = context.getBean(HelloWorldService.class);
            helloWorldService.sayHello();
        }
    }
}
```

HelloWorldConfig.java:

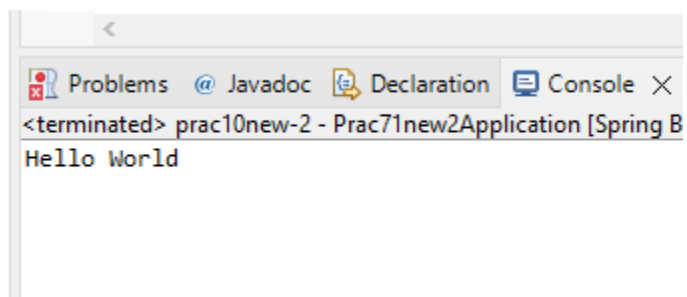
```
package com.example.demo;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class HelloWorldConfig {
    @Bean
    public HelloWorldService helloWorldService() {
        return new HelloWorldService();
    }
}
```

HelloWorldService.java:

```
package com.example.demo;  
public class HelloWorldService {  
    public void sayHello() {  
        System.out.println("Hello World");  
    }  
}
```

Output:



Prac 7.2

Aim: Write a program to demonstrate dependency injection via setter method

Code:

Helloapplication.java:

```
package com.example.demo;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@ComponentScan(basePackages = "com.example.demo")
public class Hello1Application {

    public static void main(String[] args) {
        SpringApplication.run(Hello1Application.class, args);
    }
    @Bean
    public CommandLineRunner demo(Car car) {
return args -> {
    // Start the Car
    car.setModel("Honda");
    Driver dr=new Driver();
    dr.setName("Manoranjan");

    car.setDriver(dr);
    car.start();
};
    }

}
@RestController
class HelloController {

    private final Car car;
```

```
public HelloController(Car car) {
    this.car = car;
}

@GetMapping("/hello")
public String hello() {
    car.start();
    return car.getModel();
}
}
```

Driver.java:

```
package com.example.demo;
import org.springframework.stereotype.Component;
@Component
public class Driver {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Car.java:

```
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component
public class Car {
    private String model;
    private Driver driver;
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public Driver getDriver() {
        return driver;
    }
}
```

```
}  
@Autowired  
public void setDriver(Driver driver) {  
    this.driver = driver;  
}  
public void start() {  
    System.out.println("Car model: " + model);  
    System.out.println("Driver: " + driver.getName());  
    System.out.println("Car is starting...");  
}  
}
```

Output:

```
2023-11-29T17:07:58.062+05:30 IN  
Car model: Honda  
Driver: Manoranjan  
Car is starting...
```


Prac 7.3

Aim: Write a program to demonstrate dependency injection via Constructor

Code:

Helloapplication.java:

```
package com.example.demo;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@ComponentScan(basePackages = "com.example.demo")
public class Hello1Application {

    public static void main(String[] args) {
        SpringApplication.run(Hello1Application.class, args);
    }
    @Bean
    public CommandLineRunner demo(Car car) {
return args -> {
    // Start the Car
    car.setModel("Honda");
    Driver dr=new Driver();
    dr.setName("Manoranjan");

    car.setDriver(dr);
    car.start();
};
    }

}
@RestController
class HelloController {

    private final Car car;
```

```
public HelloController(Car car) {
    this.car = car;
}

@GetMapping("/hello")
public String hello() {
    car.start();
    return car.getModel();
}
}
```

Driver.java:

```
package com.example.demo;
import org.springframework.stereotype.Component;
@Component
public class Driver {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Car.java:

```
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component
public class Car {
    private String model;
    private Driver driver;
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public Driver getDriver() {
        return driver;
    }
}
```

```
}  
@Autowired  
public void setDriver(Driver driver) {  
    this.driver = driver;  
}  
public void start() {  
    System.out.println("Car model: " + model);  
    System.out.println("Driver: " + driver.getName());  
    System.out.println("Car is starting...");  
}  
}
```

Output:

```
2023-11-29T17:07:58.062+05:30 IN  
Car model: Honda  
Driver: Manoranjan  
Car is starting...
```

Practical No.8**Prac 8.1**

Aim: Write a program to demonstrate Spring AOP – before advice.

Code:

Application.java:

```
package com.example;  
  
import com.example.service.YourService;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ConfigurableApplicationContext;  
  
@SpringBootApplication  
public class YourApplication {  
    public static void main(String[] args) {  
        ConfigurableApplicationContext context = SpringApplication.run(YourApplication.class, args);  
        YourService yourService = context.getBean(YourService.class);  
        yourService.yourmethod();  
    }  
}
```

Aspect.java:

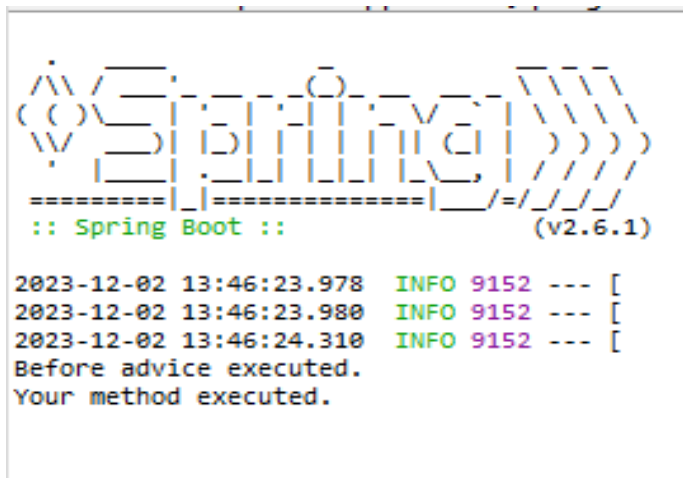
```
package com.example.aspect;  
import org.aspectj.lang.annotation.Aspect;  
import org.aspectj.lang.annotation.Before;  
import org.springframework.stereotype.Component;  
@Aspect  
@Component  
public class MyAspect {  
    @Before("execution(* com.example.service.YourService.yourmethod())")  
    public void beforeAdvice() {  
        System.out.println("Before advice executed.");  
    }  
}
```

Service.java:

```
package com.example.service;  
import org.springframework.stereotype.Service;
```

```
@Service
public class YourService {
    public void yourmethod() {
        System.out.println("Your method executed.");
    }
}
```

Output:



```
:: Spring Boot :: (v2.6.1)

2023-12-02 13:46:23.978 INFO 9152 --- [
2023-12-02 13:46:23.980 INFO 9152 --- [
2023-12-02 13:46:24.310 INFO 9152 --- [
Before advice executed.
Your method executed.
```

Prac 8.2

Aim: Write a program to demonstrate Spring AOP – After advice.

Code:

Application.java:

```
package com.example;
import com.example.service.YourService;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class YourApplication {
    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(YourApplication.class, args);
        YourService yourService = context.getBean(YourService.class);
        yourService.yourmethod();
    }
}
```

Aspect.java:

```
package com.example.aspect;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class MyAspect {
    @After("execution(* com.example.service.YourService.yourmethod())")
    public void beforeAdvice() {
        System.out.println("After advice executed.");
    }
}
```

Service.java:

```
package com.example.service;
import org.springframework.stereotype.Service;

@Service
public class serv1 {
```

```
public void service2() {
    System.out.println("my method");
}
```

Output:

[illegible]

Prac 8.3

Aim: Write a program to demonstrate Spring AOP – Around advice.

Code:

Application.java

```
package com.example;

import com.example.service.YourService;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class YourApplication {
    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(YourApplication.class, args);
        YourService yourService = context.getBean(YourService.class);
        yourService.yourmethod();
    }
}
```

Aspect.java:

```
package com.example.aspect;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class MyAspect {

    @Around("execution(* com.example.service.serv1.service2())")
    public void aspect(ProceedingJoinPoint joinPoint) {
        try {
            System.out.println("Before Advice");
            joinPoint.proceed(); // Proceed with the original method
            System.out.println("After Advice");
        } catch (Throwable e) {
            System.err.println("Exception in advice: " + e.getMessage());
        }
    }
}
```



```

    }
}

```

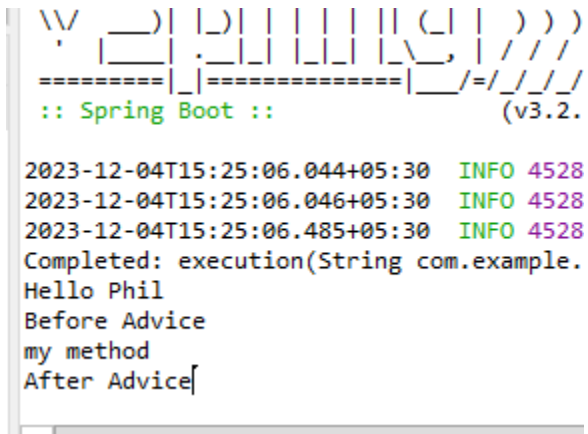
Service.java:

```

package com.example.service;
import org.springframework.stereotype.Service;
@Service
public class serv1 {
    public void service2() {
        System.out.println("my method");
    }
}

```

Output:



```

\W _ _ ) | | | | | | | | ( | | ) ) )
| _ _ | . _ | | | | | \ , | / / /
=====|_|=====|_|/=//_/_/_/
:: Spring Boot ::                (v3.2.0)

2023-12-04T15:25:06.044+05:30 INFO 4528
2023-12-04T15:25:06.046+05:30 INFO 4528
2023-12-04T15:25:06.485+05:30 INFO 4528
Completed: execution(String com.example.
Hello Phil
Before Advice
my method
After Advice[

```

Prac 8.4

Aim: Write a program to demonstrate Spring AOP – after returning advice..

Code:

Application.java

```
package com.example;

import com.example.service.YourService;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class YourApplication {
    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(YourApplication.class, args);
        YourService yourService = context.getBean(YourService.class);
        yourService.yourmethod();
    }
}
```

Aspect.java:

```
package com.example.aspect;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class MyAspect {

    @AfterReturning(
        pointcut = "execution(* com.example.service.serv1.service2())",
        returning = "result"
    )
    public void afterReturningAdvice() {
        System.out.println("After Returning Advice: Aspect executed after service2() returns");
    }
}
```


Prac 8.5

Aim: Write a program to demonstrate Spring AOP – after throwing advice.

Code:

Application.java

```
package com.example;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;
import com.example.service.serv1;
@SpringBootApplication
public class application {
    public static void main(String[] args) {
        ConfigurableApplicationContext context=SpringApplication.run(application.class, args);
        serv1 mservice=context.getBean(serv1.class);
        try {
            mservice.service2();

        }catch(Exception e) {

        }
    }
}
```

Aspect.java:

```
package com.example.aspect;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class myaspect {

    @AfterThrowing(
        pointcut = "execution(* com.example.service.serv1.service2())",
        throwing= "exception"
    )
    public void afterThrowingAdvice(Exception exception) {
        System.out.println("After Throwing Advice: Exception caught - " + exception.getMessage());
    }
}
```


Prac 8.6

Aim: Write a program to demonstrate Spring AOP – pointcuts.

Code:

Application.java:

```
package com.example;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;
import com.example.service.serv1;
@SpringBootApplication
public class application {
    public static void main(String[] args) {
        ConfigurableApplicationContext context=SpringApplication.run(application.class, args);
        serv1 mservice=context.getBean(serv1.class);

        mservice.method1();
        mservice.method2();
    }
}
```

Aspect.java:

```
package com.example.aspect;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;
@Aspect
@Component
public class myaspect {

    @Pointcut("execution(* com.example.service.serv1.method1())")
    public void pointcutMethod1() {
    }

    @Pointcut("execution(* com.example.service.serv1.method2())")
    public void pointcutMethod2() {
    }

    @Before("pointcutMethod1()")
    public void beforeMethod1() {
        System.out.println("Before advice for method1");
    }
}
```

```

@Before("pointcutMethod2()")
public void beforeMethod2() {
    System.out.println("Before advice for method2");
}
}

```

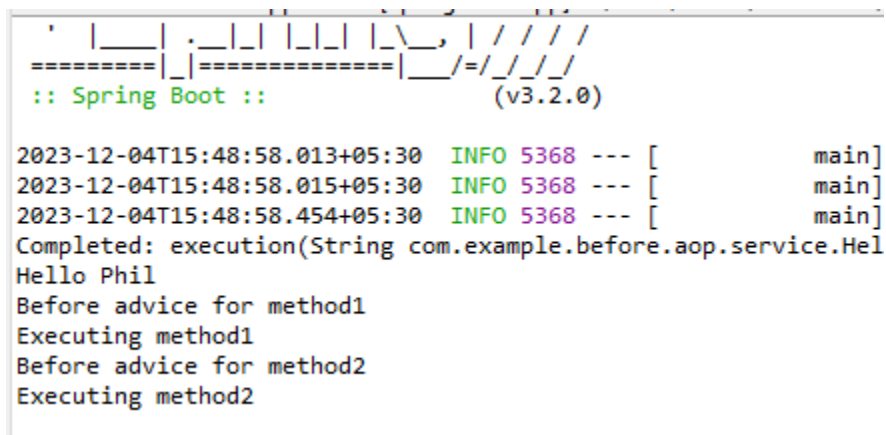
Service.java:

```

package com.example.service;
import org.springframework.stereotype.Service;
@Service
public class serv1 {
    public void method1() {
        System.out.println("Executing method1");
    }
    public void method2() {
        System.out.println("Executing method2");
    }
}

```

Output:



```

:: Spring Boot :: (v3.2.0)

2023-12-04T15:48:58.013+05:30 INFO 5368 --- [main]
2023-12-04T15:48:58.015+05:30 INFO 5368 --- [main]
2023-12-04T15:48:58.454+05:30 INFO 5368 --- [main]
Completed: execution(String com.example.before.aop.service.Hel
Hello Phil
Before advice for method1
Executing method1
Before advice for method2
Executing method2

```

Practical No.10**Prac 10.1**

Aim: Write a program to create a simple Spring Boot application that prints a message.

Code:

Greet.java:

```
package com.example.demo;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class GreetingController {
    @GetMapping("/greet")
    public String greet() {
        return "Hello, Manoranjan";
    }
}
```

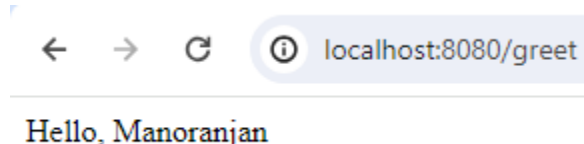
Application.java

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class Prac10newApplication {

    public static void main(String[] args) {
        SpringApplication.run(Prac10newApplication.class, args);
    }

}
```

Output:



Prac 10.2

Aim: Write a program to demonstrate RESTful Web Services with spring boot.

Code:

Controller.java:

```
package com.example.demo;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class controller {
    @GetMapping("/hello-world")
    public String helloWorld() {
        return "Hey Mr";
    }
    @GetMapping("/hello-world-bean")
    public HelloWorldBean helloWorldBean() {
        return new HelloWorldBean("Hey");
    }
}
```

HelloWorldBean.java:

```
package com.example.demo;
public class HelloWorldBean {
    private String message;
    public HelloWorldBean(String message) {
        this.message = message;
    }
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
}
```

Application.java:

```
package com.example.demo;

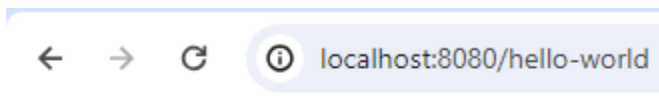
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Prac10newApplication {

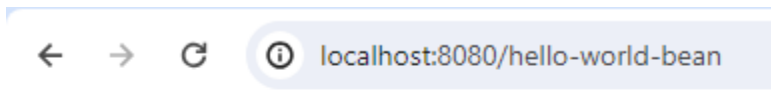
    public static void main(String[] args) {
        SpringApplication.run(Prac10newApplication.class, args);
    }

}
```

Output:



Hey Mr



```
{"message": "Hey"}
```