

Bharati Vidyapeeth’s

**Institute of Management & Information Technology**

C.B.D. Belapur, Navi Mumbai 400614

**Vision:**

Providing high quality, innovative and value-based education in information technology to build competent professionals.

**Mission**

M1. Technical Skills:-To provide solid technical foundation theoretically as well as practically capable of providing quality services to industry.

M2. Development: -Department caters to the needs of students through comprehensive educational programs and promotes lifelong learning in the field of computer Applications.

M3. Ethical leadership:-Department develops ethical leadership insight in the students to succeed in industry, government and academia.

**CERTIFICATE**

This is to certify that the journal is the work of Mr.

\_\_\_Suraj Sampat Kadam\_\_ Roll No. 26 of MCA (Sem- 1 Div: A ) for the academic year 2022 - 2024

Subject Code: MCAL12

Subject Name: Advanced Java Lab

|  |  |
| --- | --- |
| Subject-in-charge Date: | Principal |

External Examiner Date:

# Bharati Vidyapeeth's Institute of Managment & Information Technology

**MCA Semester I AY 2022-24**

**MCAL12: Advanced Java Lab**

# INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Date** | **Topic** | **Sign** |
| **1** |  | **Java Generics** |  |
| 1.1 |  | Write a Java Program to demonstrate a Generic Class. |  |
| 1.2 |  | Write a Java Program to demonstrate Generic Methods. |  |
| 1.3 |  | Write a Java Program to demonstrate Wildcards in Java Generics. |  |
| **2** |  | **List Interface** |  |
| 2.1 |  | Write a Java program to create List containing list of items of type String and use for- --each loop to print the items of the list. |  |
| 2.2 |  | Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backward direction. |  |
| **3** |  | **Set Interface** |  |
| 3.1 |  | Write a Java program to create a Set containing list of items of type String and print the items in the list using Iterator interface. Also print the list in reverse/ backward direction. |  |
| 3.2 |  | Write a Java program using Set interface containing list of items and perform the following operations: a. Add items in the set.   1. Insert items of one set in to other set. 2. Remove items from the set 3. Search the specified item in the set |  |
| **4** |  | **Map Interface** |  |
| 4.1 |  | Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations:   1. Add items in the map. 2. Remove items from the map 3. Search specific key from the map 4. Get value of the specified key 5. Insert map elements of one map in to other map. 6. Print all keys and values of the map. |  |
| **5** |  | **Lambda Expression** |  |
| 5.1 |  | Write a Java program using Lambda Expression to print ”Hello World”. |  |
| 5.2 |  | Write a Java program using Lambda Expression with single parameters. |  |
| 5.3 |  | Write a Java program using Lambda Expression with multiple parameters to add two numbers. |  |
| 5.4 |  | Write a Java program using Lambda Expression to calculate the following:   1. Convert Fahrenheit to Celcius 2. Convert Kilometers to Miles. |  |
| 5.5 |  | Write a Java program using Lambda Expression with or without return |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | keyword. |  |
| 5.6 |  | Write a Java program using Lambda Expression to concatenate two strings. |  |
| **6** |  | **Web application development using JSP** |  |
| 6.1 |  | Create a Telephone directory using JSP and store all the information within a database, so that later could be retrieved as per the requirement. Make your own assumptions. |  |
| 6.2 |  | Write a JSP page to display the Registration form (Make your own assumptions). |  |
| 6.3 |  | Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table. |  |
| 6.4 |  | 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:   1. 1 to 7 year at 5.35% 2. 8 to 15 year at 5.5% 3. 16 to 30 year at 5.75% |  |
| 6.5 |  | 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. |  |
| 6.6 |  | Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table. |  |
| 6.7 |  | Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer. |  |
| 6.8 |  | Write a JSP program that demonstrates the use of Cookies and session tracking in java. |  |
| 6.9 |  | Write a JSP program that demonstrates the use of custom tags |  |
| **7** |  | **Spring Framework** |  |
| 7.1 |  | Write a program to print “Hello World” using spring framework. |  |
| 7.2 |  | Write a program to demonstrate dependency injection via setter method. |  |
| 7.3 |  | Write a program to demonstrate dependency injection via Constructor. |  |
| **8** |  | **Aspect Oriented Programming** |  |
| 8.1 |  | Write a program to demonstrate Spring AOP – before advice. |  |
| 8.2 |  | Write a program to demonstrate Spring AOP – after advice. |  |
| 8.3 |  | Write a program to demonstrate Spring AOP – around advice. |  |
| 8.4 |  | Write a program to demonstrate Spring AOP – after returning advice. |  |
| 8.5 |  | Write a program to demonstrate Spring AOP – after throwing advice. |  |
|  |  |  |  |
| **9** |  | **Spring JDBC** |  |
| 9.1 |  | Write a program to insert, update and delete records from the given table. |  |
| 9.2 |  | Write a program to demonstrate PreparedStatement in Spring JdbcTemplate. |  |
| 9.3 |  | Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface. |  |
| 9.4 |  | Write a program to demonstrate RowMapper interface to fetch the records from the database. |  |
| **10** |  | **Spring Boot and RESTful Web Services** |  |
| 10.1 |  | Write a program to create a simple Spring Boot application that prints a message. |  |
| 10.2 |  | Write a program to demonstrate RESTful Web Services with spring boot. |  |

1. **Java Generics**
2. **Write a Java Program to demonstrate a Generic Class.**

package examples.lambda;

class Geg<T>

{

T obj1;

Geg(T obj1)

{ this.obj1=obj1; }

public T getobj1()

{ return this.obj1; }

}

class G1

{

public static void main(String[] args)

{

Geg<Integer>i=new Geg<Integer>(26);

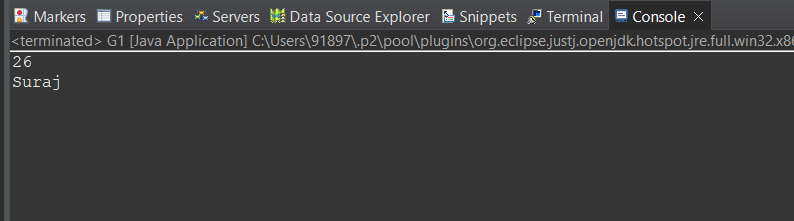
System.out.println(i.getobj1());

Geg<String>s=new Geg<String>("Suraj");

System.out.println(s.getobj1());

}

}



1.2) **Write a Java Program to demonstrate Generic Methods.**

package examples.lambda;

public class GenericExample {

public static void main(String[] args) {

System.out.println(getObj(1));

System.out.println(getObj(1.5));

System.out.println(getObj(true));

System.out.println(getObj("Hello, World"));

System.out.println(getClassValue(1));

System.out.println(getClassValue(1.5));

System.out.println(getClassValue(true));

System.out.println(getClassValue("Hello, World"));

}

public static <T> String getClassValue(T obj) {

return obj.getClass().getName() + " = " + obj;

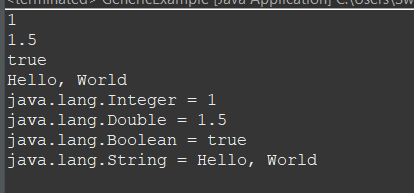
}

public static <T> T getObj(T obj) {

return obj;

}

}



1.3) **Write a Java Program to demonstrate Wildcards in Java Generics.**

//UpperBounded Wildcard

package examples.lambda;

import java.util.List;

import java.util.Arrays;

public class UpperWildCardExample {

public static void main(String[] args) {

List<Integer> intList = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

List<Double> doubleList = Arrays.asList(66.6, 77.2, 88.2, 92.5, 62.8, 88.1, 99.9);

System.out.println("Sum of intList: " + sum(intList));

System.out.println("Sum of doubleList: " + sum(doubleList));

}

public static double sum(List<? extends Number> list) {

double sum = 0;

for (Number n : list) {

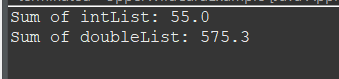
sum += n.doubleValue();

}

return sum;

}

}



//LowerBounded Wildcard

package examples.lambda;

import java.util.List;

import java.util.Arrays;

public class LowerWildCardExample {

public static void main(String[] args) {

List<Integer> intList =

Arrays.asList(1,2,3,4,5,6,7,8,9,10);

List<Double> doubleList =

Arrays.asList(66.6,77.2,88.2,92.5,62.8,88.1,99.9);

List<Number> nList = Arrays.asList(1,2,3,4,5,6,6);

printl(intList);

printl(nList);

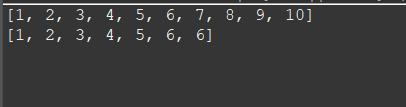
}

public static void printl(List<? super Integer> list) {

System.out.println(list);

}

}



1. **List Interface**

**2.1)** **Write a Java program to create List containing list of items of type String and use for- --each loop to print the items of the list.**

**package example.lambda;**

import java.util.\*;

public class ListOfItems {

public static void main(String[] args) {

int a;

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

list.add("basketball");

list.add("rugby");

list.add("football");

list.add("cricket");

System.out.println(list);

System.out.println("Traversing list through for each loop ");

for(String sport:list)

System.out.println(sport);

//Traversing list through Iterator

System.out.println("Traversing list through Iterator ");

Iterator itr=list.iterator();

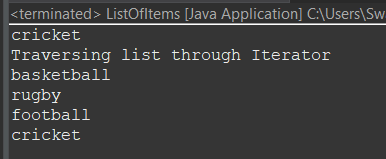
while(itr.hasNext()){

System.out.println(itr.next());

}

}

}

****

**2.2**) **Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backward direction.**

package example.lambda;

import java.util.\*;

public class ListIteratorInterface {

public static void main(String[] args) {

// Let us create a list of strings

List<String> mylist = new ArrayList<String>();

mylist.add("suraj");

mylist.add("mayur");

mylist.add("anirudh");

mylist.add("prasad");

System.out.println("Original list ");

Iterator itr=mylist.iterator();

while(itr.hasNext()){

System.out.println(itr.next());

}Collections.reverse(mylist);

System.out.println("reversed list ");

Iterator itr1=mylist.iterator();

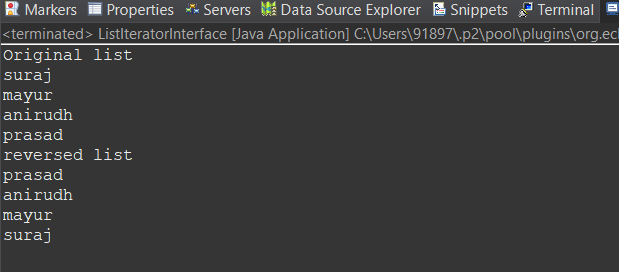
while(itr1.hasNext()){

System.out.println(itr1.next());

}

}

}



**3. Set Interface**

**3.1)** **Write a Java program to create a Set containing list of items of type String and print the items in the list using Iterator interface. Also print the list in reverse/ backward direction**

package examples.lambda;

import java.util.\*;

public class Reverse {

public static void main(String[] args) {

List<String> mylist = new ArrayList<String>();

mylist.add("suraj");

mylist.add("mayur");

mylist.add("anirudh");

mylist.add("prasad");

System.out.println("Original list ");

Iterator<String> itr=mylist.iterator();//getting the Iterator

while(itr.hasNext()){//check if iterator has the elements

System.out.println(itr.next());

}

Collections.reverse(mylist);

System.out.println("");

System.out.println("reversed list");

Iterator<String>itrl=mylist.iterator();

while(itrl.hasNext())

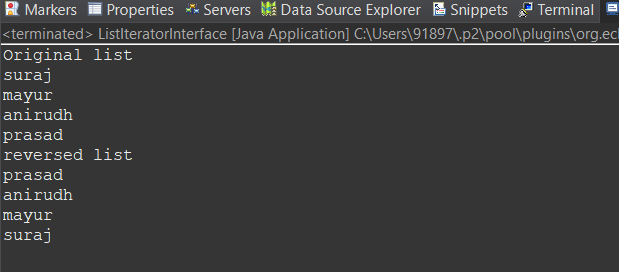
{

System.out.println(itrl.next());

}

}

}



3.2) **Write a Java program using Set interface containing list of items and perform the following operations: a. Add items in the set. b. Insert items of one set in to other set. c. Remove items from the set d. Search the specified item in the set**

**package examples.lambda;**

import java.util.\*;

public class set1{ public static void main(String[] args) {

// TODO Auto-generated method stub

Set<Integer> s = new LinkedHashSet<Integer>(); s.add(69);

s.add(57);

s.add(10);

s.add(18);

s.add(90);

s.add(151);

Set<Integer> s1 = new LinkedHashSet<Integer>(); s1.add(70); s1.add(35);

s.addAll(s1);

System.out.println("Set1: " + s);

System.out.println("Set2: " + s1);

System.out.println();

System.out.println("After Adding set2 into set1: " + s); s.remove(10);

s.remove(18);

System.out.println("Set after removing elements: " + s);

System.out.println();

System.out.println("Does the Set contains: 57? "

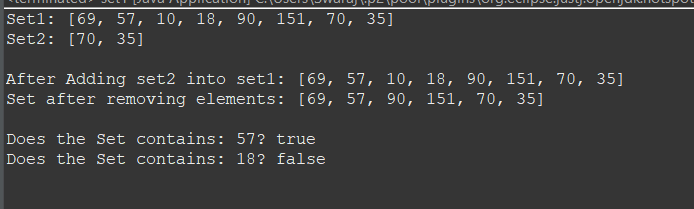
+ s.contains(57));

System.out.println("Does the Set contains: 18? "

+ s.contains(18));

}

}



**4.Map Interface**

4.1)**Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations:**

**a. Add items in the map.**

**b. Remove items from the map**

**c. Search specific key from the map**

**d. Get value of the specified key**

**e. Insert map elements of one map in to other map.**

**f. Print all keys and values of the map**

package map;

import java.util.\*;

public class mapdemo {

public static void main(String[] args) {

Map<String, String> hmap=new HashMap<>();

hmap.put("USA", "Washington DC");

hmap.put("Germany", "Berlin");

hmap.put("France", "Paris");

hmap.put("Spain", "Madrid");

hmap.put("UK", "London");

for(Map.Entry m:hmap.entrySet()){

System.out.println("capital of "+m.getKey()+" is "+m.getValue());

}

System.out.println("----------");

hmap.remove("UK");

for(Map.Entry m:hmap.entrySet()){

System.out.println("capital of "+m.getKey()+" is "+m.getValue());

}

System.out.println("----------");

System.out.println("capital of USA is "+hmap.get("USA"));

System.out.println("----------");

Map<String, String> hmap2=new HashMap<>();

hmap2.put("India", "New Delhi");

hmap2.put("Georgia", "Tbilisi");

hmap.putAll(hmap2);

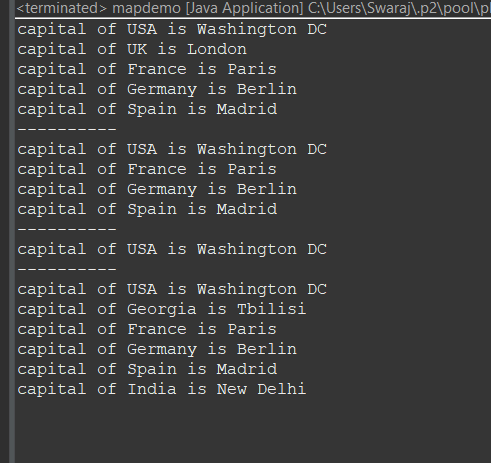
for(Map.Entry m:hmap.entrySet()) {

System.out.println("capital of "+m.getKey()+" is "+m.getValue());

}

}

}

****

**5) Lambda Expression**

**5.1)** **Write a Java program using Lambda Expression to print ”Hello World”.**

package examples.lambda;

interface HelloWorld {

public void sayhello();

}

public class helloWorldLambda {

public static void main(String[] args) {

HelloWorld h1 = () -> {

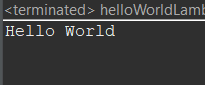
System.out.println("Hello World");

};

h1.sayhello();

}

}



5.2) **Write a Java program using Lambda Expression with single parameters**

package examples.lambda;

interface SqArea {

public double area(double a);

}

public class SingleParameterLambda {

public static void main(String[] args) {

SqArea sq = (double a)-> {

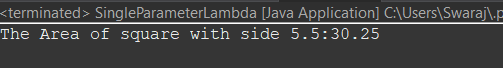
return a\*a;

};

System.*out*.println("The Area of square with side 5.5:"+sq.area(5.5));

}

}

****

5.3) **Write a Java program using Lambda Expression with multiple parameters to add two numbers.**

package examples.lambda;

interface Sum {

public int sumint(int a , int b);

}

public class MultipleParameterLambda {

public static void main(String[] args) {

Sum s1 = (int a,int b)-> {

return a+b;

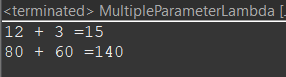
};

System.out.println("12 + 3 ="+s1.sumint(12, 3));

System.out.println("80 + 60 ="+s1.sumint(80 , 60));

}

}

****

**5.4) Write a Java program using Lambda Expression to calculate the following:**

**a. Convert Fahrenheit to Celcius**

**b. Convert Kilometers to Miles.**

package examples.lambda;

interface TempConversion {

public double convert(double f);

}

interface DistanceConversion {

public double convert(double km);

}

public class ConversionLambda {

public static void main(String[] args) {

TempConversion tc = (double f)->(f-32)\*5/9;

System.out.println("-8.4 Farenheit is"+ String.format("%.2f",tc.convert(-8.4))+"Celcius");

System.out.println("32 Farenheit is " + String.format("%.2f",tc.convert(32))+"Celcius");

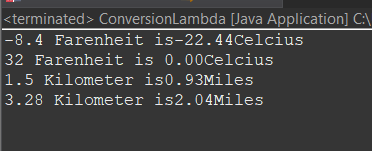
DistanceConversion dc = (double km)-> km/1.609344;

System.out.println("1.5 Kilometer is"+String.format("%.2f",dc.convert(1.5), args)+"Miles");

System.out.println("3.28 Kilometer is"+String.format("%.2f",dc.convert(3.28), args)+"Miles");

}

}



**5.5) Write a Java program using Lambda Expression with or without return keyword.**

package examples.lambda;

public class lambdaExample {

public static void main(String[] args) {

// Lambda expression without return keyword

MyFunction add = (a, b) -> a + b;

int sum = add.apply(2, 3);

System.out.println("Sum without return keyword: " + sum);

// Lambda expression with return keyword

MyFunction subtract = (a, b) -> {

int result = a - b;

return result;

};

int difference = subtract.apply(5, 2);

System.out.println("Difference with return keyword: " + difference);

}

// Functional interface with a single abstract method

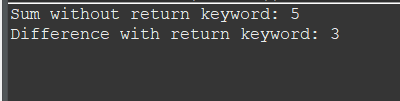
@FunctionalInterface

interface MyFunction {

int apply(int a, int b);

}

}



**5.6) Write a Java program using Lambda Expression to concatenate two strings.**

package examples.lambda;

interface concatstr {

public String concat(String s1 , String s2);

}

public class ComcatenationLambda {

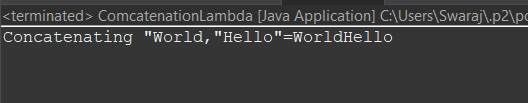
public static void main(String[] args) {

concatstr cs = (String s1,String s2)-> s1+s2;

System.out.println("Concatenating \"World,\"Hello\"=" + cs.concat("World","Hello"));

}

}

****

**6. Web application development using JSP**

**6.1) Create a Telephone directory using JSP and store all the information within a database, so that later could be retrieved as per the requirement. Make your own assumptions.**

**//index.jsp :**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("delete")!=null)

{

int id=Integer.parseInt(request.getParameter("delete"));

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("delete from phone2 where id=? "); // delete query

pstmt.setInt(1,id);

pstmt.executeUpdate(); //execute query

con.close(); //close connection

} } catch(Exception e)

{

out.println(e);

}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete </title>

</head>

<body>

<br>

<br>

<center>

<h1><a href="add.jsp">CLICK HERE TO ADD A NEW MOBILE NUMBER</a></h1>

</center>

<br>

<center>

<table border='1' cellpadding='23'>

</center>

<tr>

<th>ID</th>

<th>NAME</th>

<th>MOBILE NUMBER</th>

<th>UPDATE</th>

<th>DELETE</th>

</tr>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password); PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("select \* from phone2"); //select query

ResultSet rs=pstmt.executeQuery(); //execute query and set in resultset object rs.

while(rs.next())

{

%>

<tr>

<td><%=rs.getInt(1)%></td>

<td><%=rs.getString(2)%></td>

<td><%=rs.getString(3)%></td>

<td><a href="update.jsp?edit=<%=rs.getInt(1)%> ">Edit</a></td>

<td><a href="?delete=<%=rs.getInt(1)%> ">Delete</a></td>

</tr>

<%

}

} catch(Exception e)

{

out.println(e); }

%>

</table>

</body>

</html>

**//add.jsp:**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("btn\_add")!=null) //check button click event not null

{

String name,no; name=request.getParameter("txt\_name"); //txt\_name

no=request.getParameter("txt\_no"); //txt\_owner

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("insert into phone2(name,no)values(?,?)"); // insert query

pstmt.setString(1,name);

pstmt.setString(2,no);

pstmt.executeUpdate(); //execute query

con.close(); //close connection

out.println("Insert Successfully...! Click Home page.");// after insert record successfully message

}

} catch(Exception e)

{

out.println(e); }

%>

<html>

<head>

<title>JSP:Insert, Update, Delete using MySQL</title>

<!-- javascript for form validation-->

<script>

function validate()

{ var name = document.myform.txt\_name;

var no = document.myform.txt\_no;

if (name.value == "")

{

window.alert("please enter a name ?");

name.focus();

return false;

}

if (no.value == "")

{

window.alert("please enter a mobile number ?");

name.focus();

return false;

}

}

</script>

</head>

<body>

<form method="post" name="myform" onsubmit="return validate();">

<center>

<h1>Insert Record</h1>

</center>

<table>

<tr>

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

<td><input type="text" name="txt\_name"></td>

</tr>

<tr>

<td><b>Phone number:</b></td></b>

<td><input type="text" name="txt\_no"></td>

</tr>

<tr>

<td><input type="submit" name="btn\_add" value="Insert"></td>

</tr>

</table>

<center>

<h1><a href="Index.jsp">Home page</a></h1>

</center>

</form>

</body>

</html>

//update.jsp :

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("btn\_update")!=null) //check button click event not null

{

int hide,name,no;; String name\_up,no\_up;

hide=Integer.parseInt(request.getParameter("txt\_hide")); //it is hidden id get for update record

name\_up=request.getParameter("txt\_name");

no\_up=request.getParameter("txt\_no"); //txt\_name

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("update phone2 set name=?,no=? where id=?"); // update query

pstmt.setString(1,name\_up);

pstmt.setString(2,no\_up);

pstmt.setInt(3,hide);

pstmt.executeUpdate(); //execute query

con.close(); //connection close

out.println("Update Successfully...! Click Back link."); //after update record successfully message

}

} catch(Exception e)

{

out.println(e);

}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete using MySQL</title>

<!-- javascript for form validation-->

<script>

function validate()

{

var name = document.myform.txt\_name;

var no = document.myform.txt\_no;

if (rno.value == "")

{

window.alert("please enter name ?");

name.focus();

return false;

}

if (name.value == "")

{

window.alert("please enter number ?");

name.focus();

return false;

}

}

</script>

</head>

<body>

<form method="post" name="myform" onsubmit="return validate();">

<center>

<h1>Update Record</h1> </center>

<table>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("edit")!=null)

{

int id=Integer.parseInt(request.getParameter("edit"));

String name,no;

PreparedStatement pstmt=null; // create statement

pstmt=con.prepareStatement("select \* from phone2 where id=?"); // sql select query

pstmt.setInt(1,id);

ResultSet rs=pstmt.executeQuery(); // execute query store in resultset object rs.

while(rs.next()) {

id=rs.getInt(1); name=rs.getString(2); no=rs.getString(3);

%>

<tr>

<td>Name</td>

<td><input type="text" name="txt\_name" value="<%=name%>"></td>

</tr>

<tr>

<td>Mobile Number</td>

<td><input type="text" name="txt\_no" value="<%=no%>"></td>

</tr>

<tr>

<td><input type="submit" name="btn\_update" value="Update"></td>

</tr>

<input type="hidden" name="txt\_hide" value="<%=id%>">

<%

}

} }

catch(Exception e)

{

out.println(e);

}

%>

</table>

<center>

<h1><a href="Index.jsp">Back</a></h1>

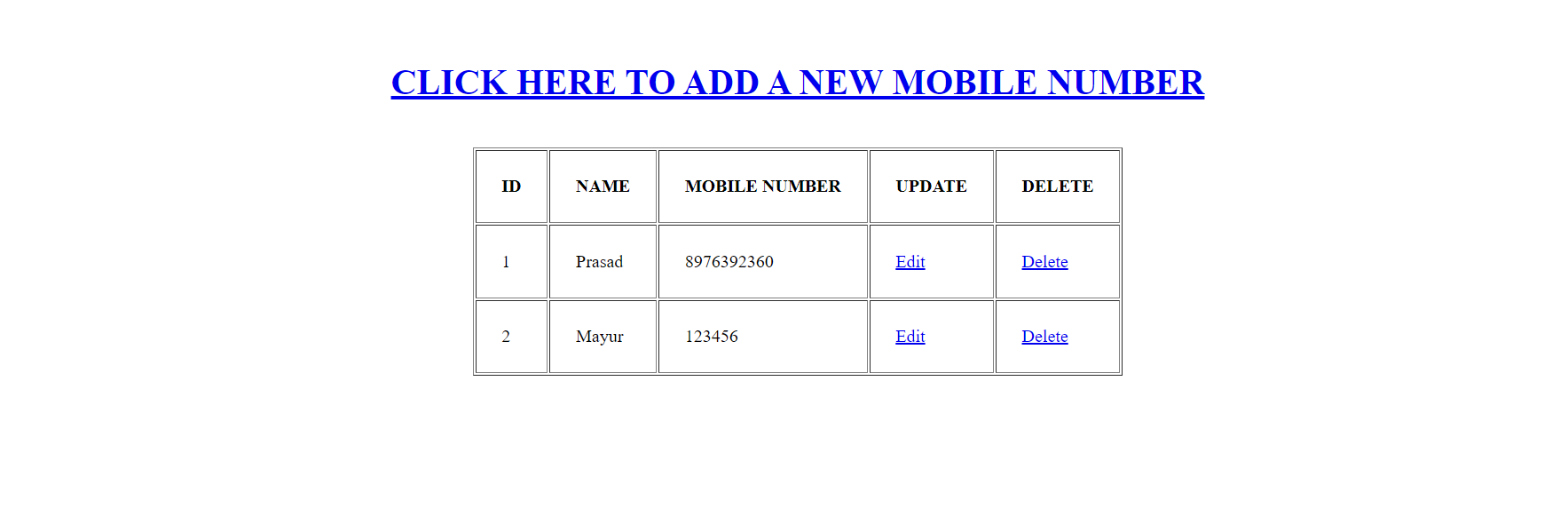
</center>

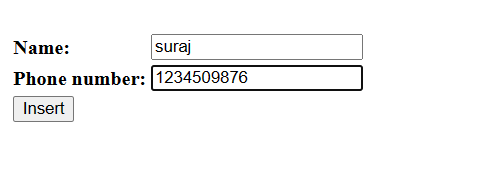
</form>

</body>

</html>

Output:









**6.2) Write a JSP page to display the Registration form (Make your own assumptions).**

**//index.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("delete")!=null)

{

int id=Integer.parseInt(request.getParameter("delete"));

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("delete from studentreg1 where id=? "); // delete query

pstmt.setInt(1,id);

pstmt.executeUpdate(); //execute query

con.close(); //close connection

}

}

catch(Exception e) {

out.println(e);

}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete </title> </head>

<body bgcolor="F9CDAD">

<br>

<br>

<br>

<center>

<h1><u>DETAILS OF REGISTERED STUDENTS</u></h1>

</center>

<br><br>

<center>

<table border='1' cellpadding='23'>

</center>

<tr>

<th>ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Mobile Number</th>

<th>Address</th>

<th>Course</th>

<th>College Name</th>

</tr>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("select \* from studentreg1"); //select query

ResultSet rs=pstmt.executeQuery(); //execute query and set in resultset object rs.

while(rs.next())

{

%>

<tr>

<td><%=rs.getInt(1)%></td>

<td><%=rs.getString(2)%></td>

<td><%=rs.getString(3)%></td>

<td><%=rs.getString(4)%></td>

<td><%=rs.getString(5)%></td>

<td><%=rs.getString(6)%></td>

<td><%=rs.getString(7)%></td>

</tr>

<%

}

}

catch(Exception e)

{

out.println(e);

}

%>

</table>

</body>

</html>

//add.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("btn\_add")!=null) //check button click event not null

{

String first\_name,last\_name,phn\_number,address,course,college\_name; first\_name=request.getParameter("txt\_first\_name"); //txt\_name

last\_name=request.getParameter("txt\_last\_name"); //txt\_owner

phn\_number=request.getParameter("txt\_phn\_number");

address=request.getParameter("txt\_address");

course=request.getParameter("txt\_course");

college\_name=request.getParameter("txt\_college\_name")

; PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("insert into studentreg1(first\_name,last\_name,phn\_number,address,course,college\_name)values(?,?,?,?,?,?)"); // insert query

pstmt.setString(1,first\_name);

pstmt.setString(2,last\_name);

pstmt.setString(3,phn\_number);

pstmt.setString(4,address);

pstmt.setString(5,course);

pstmt.setString(6,college\_name);

pstmt.executeUpdate(); //execute query

con.close(); //close connection

out.println("Insert Successfully...!");// after insert record successfully message

}

}

catch(Exception e)

{

out.println(e); }

%>

<html>

<head>

<!-- javascript for form validation-->

<script>

function validate()

{ var first\_name = document.myform.txt\_first\_name;

var last\_name = document.myform.txt\_last\_name;

var phn\_number = document.myform.txt\_phn\_number;

var address = document.myform.txt\_address;

var course = document.myform.txt\_course;

var college\_name = document.myform.txt\_college\_name;

if (first\_name.value == "")

{

window.alert("please enter a first name ?");

name.focus();

return false;

} if (last\_name.value == "")

{

window.alert("please enter a last name ?");

name.focus();

return false;

} if (phn\_number.value == "")

{

window.alert("please enter a mobile number ?");

name.focus();

return false;

}

if (address.value == "")

{ window.alert("please enter address ?");

name.focus();

return false;

}

if (course.value == "")

{

window.alert("please enter course ?");

name.focus();

return false;

} if (college\_name.value == "")

{

window.alert("please enter college name ?");

name.focus();

return false;

}

}

</script>

</head>

<body bgcolor="deea94">

<div align="center">

<form method="post" name="myform" onsubmit="return validate();">

<center>

<h1><u>STUDENT REGISTRATION FORM</u></h1>

</center>

<br>

<table>

<tr>

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

<td><input type="text" name="txt\_first\_name"></td>

</tr>

<tr>

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

<td><input type="text" name="txt\_last\_name"></td>

</tr>

<tr>

<td><b>Phone number:</b></td></b>

<td><input type="text" name="txt\_phn\_number"></td>

</tr>

<tr>

<td><b>Address:</b></td></b>

<td><input type="text" name="txt\_address"></td>

</tr>

<tr>

<td><b>Course:</b></td></b>

<td><input type="text" name="txt\_course"></td>

</tr>

<tr>

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

<td><input type="text" name="txt\_college\_name"></td>

</tr>

<tr>

<td><br><input type="submit" name="btn\_add" value="Submit"></br></td> </tr>

</table>

<center>

<p><a><span>&#8595;</span> <u>Click Below to list all the</u> <span>&#8595;</span></a></p>

<p><a href="index.jsp">Registered Students Details</a></p>

</center>

</form>

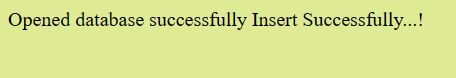
</div>

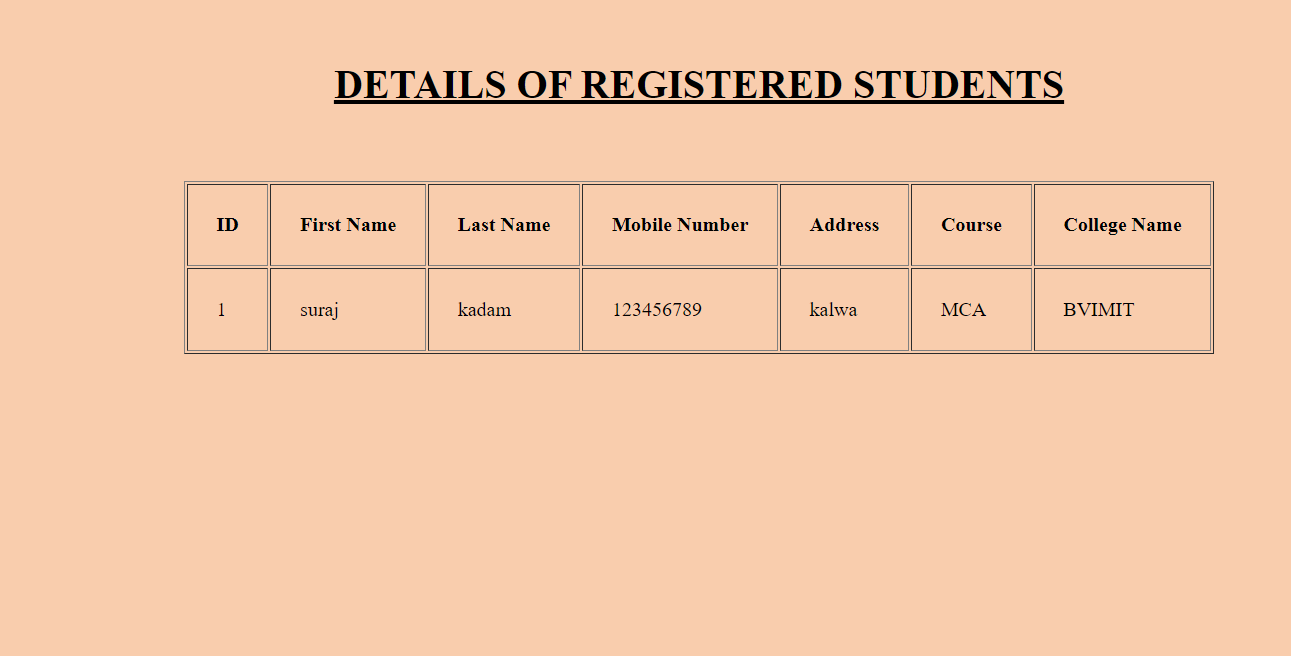
</body>

</html>

Output:







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

//index.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance(); con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("delete")!=null)

{

int id=Integer.parseInt(request.getParameter("delete"));

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("delete from student1 where id=? "); // delete query

pstmt.setInt(1,id); pstmt.executeUpdate(); //execute query

con.close(); //close connection

} } catch(Exception e)

{

out.println(e);

}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete </title>

</head>

<body>

<center>

<h1><a href="add.jsp">Add Record</a></h1>

</center>

<table>

<tr>

<th>ID</th>

<th>Roll No</th>

<th>Name</th>

<th>Sem</th>

<th>Course</th>

<th>Update</th>

<th>Delete</th>

</tr>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("select \* from student1"); //select query

ResultSet rs=pstmt.executeQuery(); //execute query and set in resultset object rs.

while(rs.next())

{

%>

<tr>

<td><%=rs.getInt(1)%></td>

<td><%=rs.getString(2)%></td>

<td><%=rs.getString(3)%></td>

<td><%=rs.getString(4)%></td>

<td><%=rs.getString(5)%></td>

<td><a href="update.jsp?edit=<%=rs.getInt(1)%> ">Edit</a></td>

<td><a href="?delete=<%=rs.getInt(1)%> ">Delete</a></td>

</tr>

<%

} } catch(Exception e)

{

out.println(e);

}

%>

</table>

</body>

</html>

**//add.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("btn\_add")!=null) //check button click event not null

{

String rno,name,semester,course;

rno=request.getParameter("txt\_rno");

name=request.getParameter("txt\_name"); //txt\_name

semester=request.getParameter("txt\_sem"); //txt\_owner

course=request.getParameter("txt\_course"); //txt\_owner

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("insert into student1(rno,name,semester,course)values(?,?,?,?)"); // insert query

pstmt.setString(1,rno);

pstmt.setString(2,name);

pstmt.setString(3,semester);

pstmt.setString(4,course);

pstmt.executeUpdate(); //execute query

con.close(); //close connection

out.println("Insert Successfully...! Click Back link.");// after insert record successfully message

} } catch(Exception e)

{

out.println(e);

}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete using MySQL</title>

<!-- javascript for form validation-->

<script>

function validate()

{

var rno = document.myform.txt\_rno;

var name = document.myform.txt\_name;

var semester = document.myform.txt\_sem;

var course = document.myform.txt\_course;

if (rno.value == "")

{

window.alert("please enter rno ?");

name.focus();

return false;

} if (name.value == "")

{

window.alert("please enter name ?");

name.focus();

return false;

}

if (semester.value == "")

{ window.alert("please enter sem ?");

owner.focus();

return false;

}

if (course.value == "")

{

window.alert("please enter course ?");

owner.focus();

return false;

}

}

</script>

</head>

<body>

<form method="post" name="myform" onsubmit="return validate();">

<center>

<h1>Insert Record</h1>

</center>

<table>

<tr>

<td>Roll No</td>

<td><input type="text" name="txt\_rno"></td>

</tr>

<tr>

<td>Name</td>

<td><input type="text" name="txt\_name"></td>

</tr>

<tr>

<td>Sem</td>

<td><input type="text" name="txt\_sem"></td>

</tr>

<tr>

<td>Course</td>

<td><input type="text" name="txt\_course"></td>

</tr>

<tr>

<td><input type="submit" name="btn\_add" value="Insert"></td>

</tr>

</table>

<center>

<h1><a href="index.jsp">Back</a></h1>

</center>

</form>

</body>

</html>

**//update.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ page import="java.sql.\*" %>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password); System.out.println("Opened database successfully"); if(request.getParameter("btn\_update")!=null) //check button click event not null

{

int hide,rno,name,semester,course;

String rno\_up,name\_up,semester\_up,course\_up;

hide=Integer.parseInt(request.getParameter("txt\_hide")); //it is hidden id get for update record

rno\_up=request.getParameter("txt\_rno");

name\_up=request.getParameter("txt\_name"); //txt\_name

semester\_up=request.getParameter("txt\_semester");

course\_up=request.getParameter("txt\_course");

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("update student1 set rno=?,name=?, semester=?, course=? where id=?"); // update query

pstmt.setString(1,rno\_up); pstmt.setString(2,name\_up);

pstmt.setString(3,semester\_up);

pstmt.setString(4,course\_up);

pstmt.setInt(5,hide); pstmt.executeUpdate(); //execute query

con.close(); //connection close

out.println("Update Successfully...! Click Back link."); //after update record successfully message

} } catch(Exception e)

{

out.println(e);

}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete using MySQL</title>

<!-- javascript for form validation-->

<script>

function validate()

{ var rno = document.myform.txt\_rno;

var name = document.myform.txt\_name;

var semester = document.myform.txt\_semester;

var course = document.myform.txt\_course;

if (rno.value == "")

{

window.alert("please enter rno ?");

name.focus();

return false;

} if (name.value == "")

{

window.alert("please enter name ?");

name.focus();

return false;

}

if (semester.value == "")

{

window.alert("please enter sem ?");

owner.focus();

return false;

}

if (course.value == "")

{

window.alert("please enter course ?");

owner.focus();

return false;

}}

</script>

</head> <body>

<form method="post" name="myform" onsubmit="return validate();">

<center>

<h1>Update Record</h1>

</center>

<table>

<%

try

{

String driver ="org.postgresql.Driver";

String url ="jdbc:postgresql://localhost:5432/postgres";

String username ="postgres";

String password ="admin";

Connection con =null;

Class.forName(driver).newInstance();

con = DriverManager.getConnection(url,username,password);

out.println("Opened database successfully");

if(request.getParameter("edit")!=null)

{

int id=Integer.parseInt(request.getParameter("edit"));

String rno,name,semester,course;

PreparedStatement pstmt=null; // create statement

pstmt=con.prepareStatement("select \* from student1 where id=?"); // sql select query

pstmt.setInt(1,id);

ResultSet rs=pstmt.executeQuery(); // execute query store in resultset object

while(rs.next()) {

id=rs.getInt(1);

rno=rs.getString(2);

name=rs.getString(3);

semester=rs.getString(4);

course=rs.getString(5);

%>

<tr>

<td>Roll NO</td>

<td><input type="text" name="txt\_rno" value="<%=rno%>"></td>

</tr>

<tr>

<td>Name</td>

<td><input type="text" name="txt\_name" value="<%=name%>"></td>

</tr>

<tr>

<td>Sem</td>

<td><input type="text" name="txt\_semester" value="<%=semester%>"></td>

</tr>

<tr>

<td>Course</td>

<td><input type="text"name="txt\_course="<%=course%>"></td>

</tr>

<tr>

<td><input type="submit"name-="btn\_update"value="Update"></td>

</tr>

<input type="hidden"name="txt\_hide"value="<%=id%>">

<%

}

}

}

catch(Exception e)

{

out.println(e);

}

%>

</table>

</center>

</form>

</body>

</html>

Output:







**6.4) 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%**

**//Cal.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

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

<title>JSP Page</title>

</head>

<body><br><br><center>

<form action="test.jsp "><pre>

<h1>Principle :: <input type=text name=principle value=0 " ><br>

No. of Years :: <input type=text name=year value=0 " ><br>

Rate of Interest :: <input type=text name=interest value=0 " > %<br> <br>

<input type=submit value="Submit"></h1>

</pre></form></center>

</body> </html>

**//test.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

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

<title>JSP Page</title>

</head>

<body><br><br><center><pre><H1>

<%

String ns= request.getParameter("principle");

String ns1= request.getParameter("year");

String ns2= request.getParameter("interest");

int n1=Integer.parseInt(ns);

int n2=Integer.parseInt(ns1);

float n3 = Float.valueOf(ns2);

double si=((n1\*n2\*n3)/100);

double x; x=n1+si;

double r = (n3)/(12\*100);

int mon; mon=((n2)\*12);

double emi= (n1\*r\*Math.pow(1+r,mon))/(Math.pow(1+r,mon)-1);

%>

<%

out.println("Principle = "+n1);

out.println(" Years = "+n2);

out.println(" Rate of Interest = "+n3);

out.println("<br> ");

out.println("Loan Amount = "+n1);

out.println(" Interest Paid = "+si);

out.println(" Total Loan Amount = "+x);

out.println("<br> ");

out.print(" Loan Tenure in months= " +mon);

out.println("<br> ");

out.print(" EMI is= "+emi+"\n");

%>

</H1>

</pre>

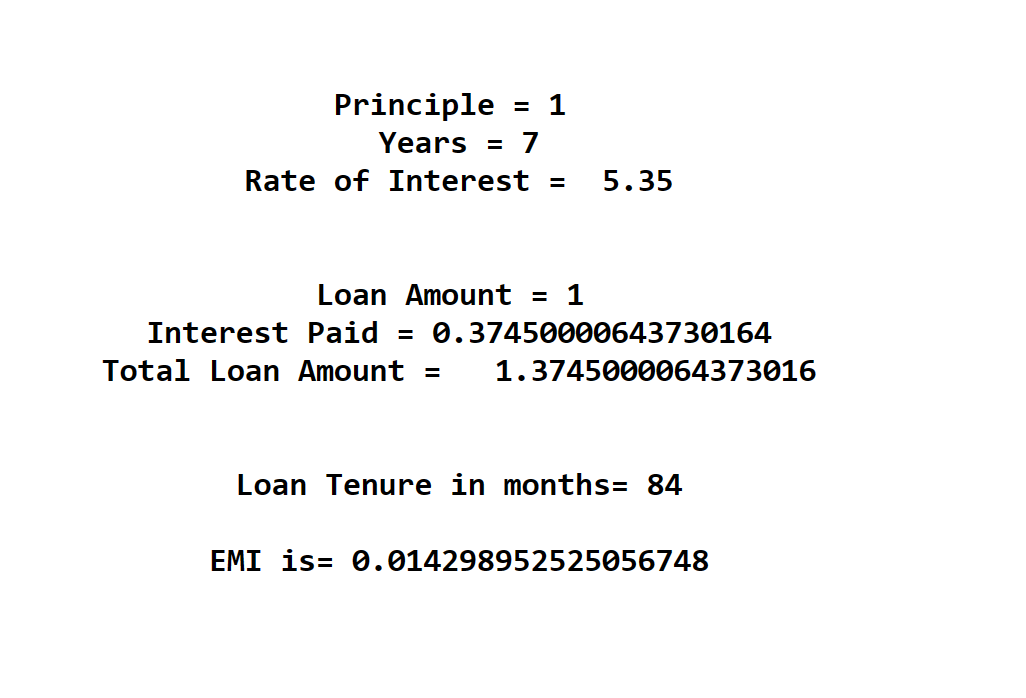
</center>

</body> </body>

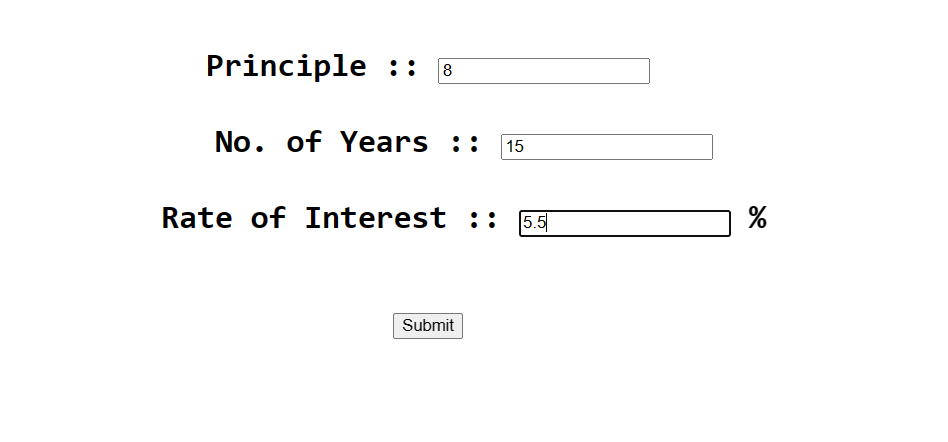
</html>

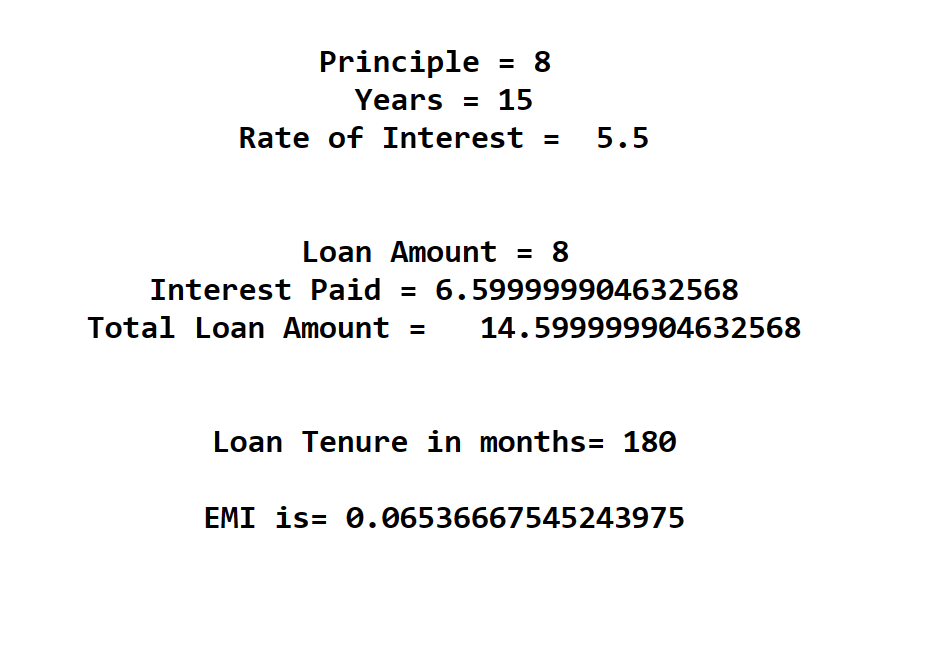
a)1 to 7 at 5.35%



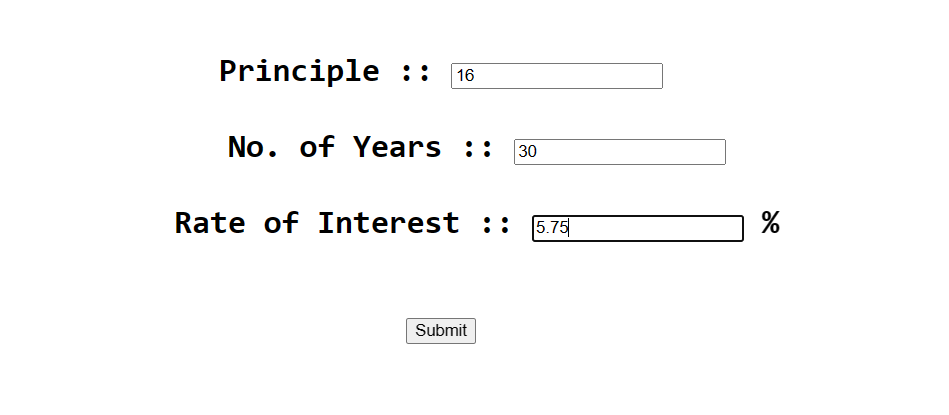


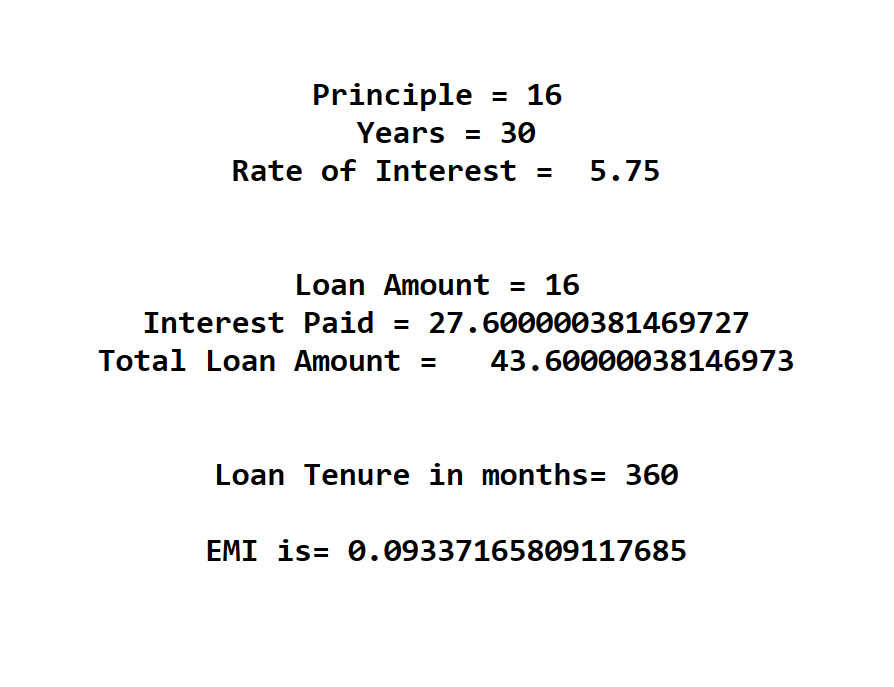
b) 8 to 15 at 5.5%





c)16 to 30 at 5.75%





**6.5) 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.**

**//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>Change of Study Center</title>

<link rel="stylesheet"href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.0/font/bootstrap-icons.css">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-BmbxuPwQa21c/FVzBcNJ7UAyJxM6wuqIj61tLre4wSX0szH/Ev+nYRRuWlolflfl"

crossorigin="anonymous">

</head>

<body>

<center>

<h1>Confirmation</h1>

<%

// get the form data

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

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

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

// validate the form data

if(uid == null || uid.trim().isEmpty() || currentCenter == null || currentCenter.trim().isEmpty() || newCenter == null || newCenter.trim().isEmpty()) {

out.println("<p>Invalid form data. Please fill all the required fields.</p>");

} else {

// send an email to the administrator

String adminEmail = "admin@example.com";

String subject = "Change of Study Center Request";

String message = "UID: " + uid + "\nCurrent Center: " + currentCenter + "\nNew Center: " + newCenter;

// code to send email

// display the confirmation message

out.println("<p>Your request to change Study Center from " + currentCenter + " to " + newCenter + " has been sent to the Administrator.</p>");

}

%>

<a href="Center.jsp">Go back</a>

</center>

</body>

</html>

**//Center.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>Study Center</title>

<link rel="stylesheet"href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.0/font/bootstrap-icons.css">

<link href="href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css"rel="stylesheet"

integrity="sha384-BmbxuPwQa21c/FVzBcNJ7UAyJxM6wuqIj61tLre4wSX0szH/Ev+nYRRuWlolflfl"

crossorigin="anonymous">

</head>

<body>

<center>

<h1> Change of Study Center</h1>

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

<table>

<tr>

<td>UID No.</td>

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

</tr>

<tr>

<td>

CurrentCenter

</td>

<td>

<select name="currentCenter"required>

<option value="Mumbai">Mumbai</option>

<option value="Pune">pune</option>

<option value="Kolhapur">Kolhapur</option>

</select>

</td>

</tr>

<tr>

<td>

New Center

</td>

<td>

<select name="newCenter"required>

<option value="Mumbai">Mumbai</option>

<option value="Pune">pune</option>

<option value="Kolhapur">Kolhapur</option>

</select>

</td>

</tr>

</table>

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

</form>

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

</form>

</center>

<%

if(request.getParameter("uid") != null&& request.getParameter("currentCenter") != null&& request.getParameter("newCenter") != null)

{

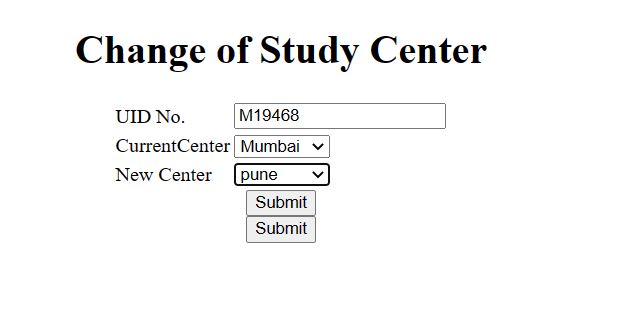
out.println("<center><br>Your request to change Study Center from <br>" +request.getParameter("currentCenter") + " to " + request.getParameter("newCenter") + "<br> has been sent to the Administrator.</center>");

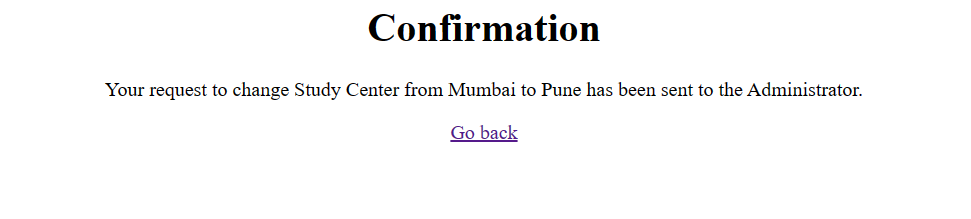
}

%>

</body>

</html>





**6.7) Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer.**

**//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 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(3) %></br>

<%!

int rectangle(int l,int b){ return l\*b;}

%>

<%= "Area of rectangle is:"+rectangle(3,4

) %></br>

<%!

int perimeter(int x,int y){

int peri=2\*(x+y);

return peri;}

%>

<%= "Perimeter of rectanlge:"+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>

<%!

int pageCount = 0;

void addCount() {

pageCount++;

}

%>

<% addCount(); %>

<html>

<head>

<title>JSP declaration, scriptlet, directives, expression, header and footer Example</title>

</head>

<body>

<center>

<h2>The include Directive Example</h2>

<p>This site has been visited <%= pageCount %> times.</p>

</center>

<br/><br/>

</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>

<br/><br/>

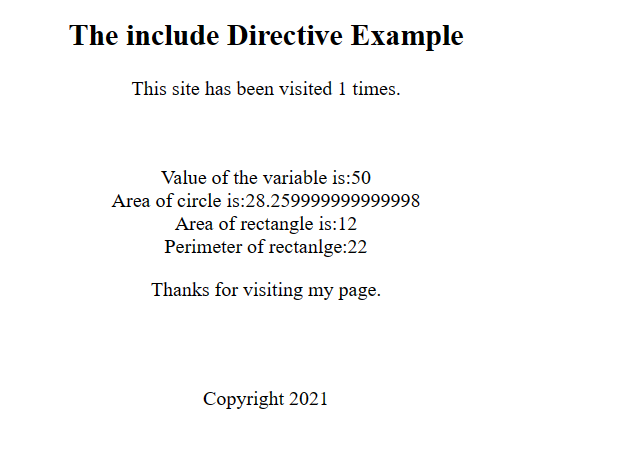
<center>

<p>Copyright 2021</p>

</center>

</body>

</html>



**6.7) Write a JSP program that demonstrates the use of Cookies and session tracking in java.**

**//Cookies.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<body>

<form action = "Main.jsp" method = "GET">

First Name: <input type = "text" name = "first\_name">

<br />

Last Name: <input type = "text" name = "last\_name" />

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

</form>

</body>

</html>

**//Main.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<%

Cookie firstName = new Cookie("first\_name", request.getParameter("first\_name"));

Cookie lastName = new Cookie("last\_name", request.getParameter("last\_name"));

firstName.setMaxAge(60\*60\*24);

lastName.setMaxAge(60\*60\*24);

response.addCookie( firstName );

response.addCookie( lastName );

%>

<html>

<head>

<title>Setting Cookies</title>

</head>

<body>

<center>

<h1>Setting Cookies</h1>

</center>

<ul>

<li><p><b>First Name:</b>

<%= request.getParameter("first\_name")%>

</p></li>

<li><p><b>Last Name:</b>

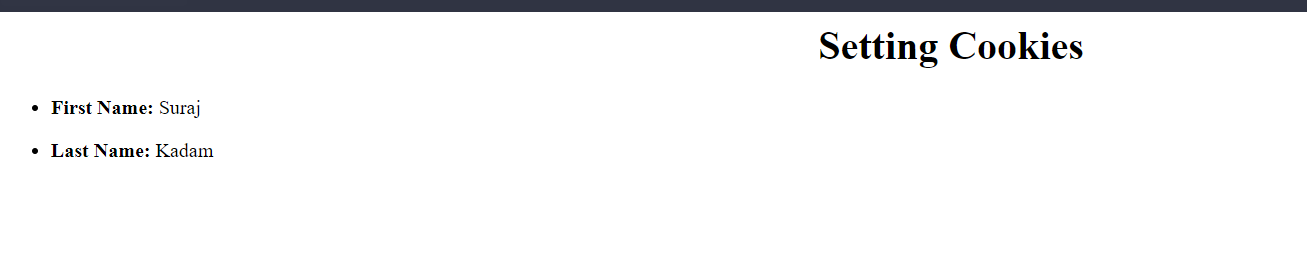
<%= request.getParameter("last\_name")%>

</p></li>

</ul>

</body>

</html>



**7. Spring Framework**

**7.1) Write a program to print “Hello World” using spring framework.**

**//HelloWorld.java**

package Spring1;

public class HelloWorld { String name;

public String getName() { return name;

}

public void setName(String name) { this.name = name;

}

@Override

public String toString() { return "Hello World, I'm " + name + ".";

}

}

**//appctx.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<bean id="hw" class="Spring1.HelloWorld">

<property name="name" value="Suraj"/>

</bean>

</beans>

**//Test.java**

package Spring1;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) {

ClassPathXmlApplicationContext app = new ClassPathXmlApplicationContext("appctx.xml");

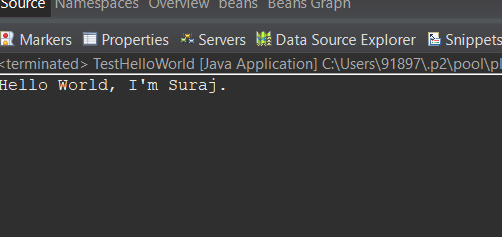
HelloWorld hw = (HelloWorld) app.getBean("hw");

System.out.println(hw.toString());

app.close();

}

}



**7.2) : Write a program to demonstrate dependency injection via setter method.**

**//Account1.java**

package Spring1;

public class Account1 {

int id;

String name; int balance;

public Account1(int id, String name, int balance) { super();

this.id = id; this.name = name;

this.balance = balance;

}

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 int getBalance() { return balance;

}

public void setBalance(int balance) { this.balance = balance;

}

@Override

public String toString() { return "Account [id=" + id + ", name=" + name + ", balance=" + balance + "]";

}

}

**//Test.java**

package Spring1;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test

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

ApplicationContext con = new ClassPathXmlApplicationContext("appctx.xml");

Account1 acc = (Account1) con.getBean("Account1");

System.out.println(acc.toString());

}

}

**//appctx.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<bean id="Account1" class="Spring1.Account1">

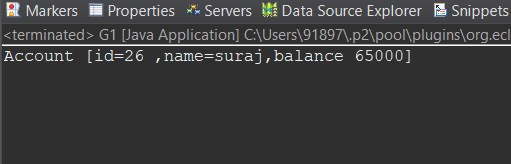
<constructor-arg name="id" value="26"></constructor-arg>

<constructor-arg name="name" value="suraj"></constructor-arg>

<constructor-arg name="balance" value="65000"></constructor-arg>

</bean>

</beans>



**7.3) Write a program to demonstrate dependency injection via Constructor.**

**//Singer.java**

package Spring1;

public class Singer { String name; int age;

public String getName() { return name;

}

public void setName(String name) { this.name = name;

}

public int getAge() { return age;

}

public void setAge(int age) { this.age = age;

} void displayInfo()

{

System.out.println("Name:" +name+" Age:" +age);

} }

**//SingerTest.java**

package Spring1;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SingerTest {

private static ApplicationContext ctx;

public static void main(String[] args) {

ctx=new ClassPathXmlApplicationContext("appctx.xml");

Singer singer=(Singer)ctx.getBean("Singer");

singer.displayInfo();

}

}

**//appctx.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans

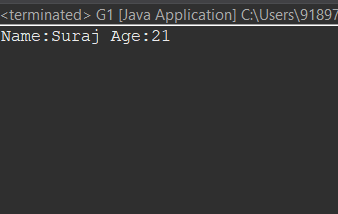
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="Singer" class="Spring1.Singer">

<property name="name" value="Suraj"></property>

<property name="age" value="21"></property> </bean>

</beans>



**8. Aspect Oriented Programming**

**8.1)** **Write a program to demonstrate Spring AOP – before advice.**

**//beforeaop.java**

package aspectj;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.Pointcut;

@Aspect

public class beforeaop {

@Pointcut("execution(int beforeoperation.\*(..))") public void p(){}

@Before("p()") public void myadvice(JoinPoint jp)

{

System.out.println("before advice"); } }

**//beforeaopOperation.java**

package aspectj;

public class beforeaopOperation {

public void msg() {System.out.println("method 1");

}

public int m(){System.out.println("method 2 with return");

return 2;

} public int k(){System.out.println("method 3 with return");

return 3;}

}

**//beforetest.java**

package aspectj;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class beforetest {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("appctx.xml");

beforeaopOperation e = (beforeaopOperation) context.getBean("opBean");

System.out.println("calling m1......");

e.msg();

System.out.println("calling m2......");

e.m();

System.out.println("calling m3......");

e.k();

}

}

**//appctx.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans

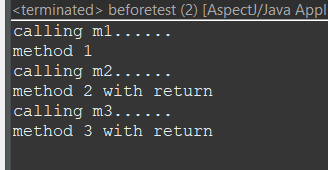
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="opBean" class="aspectj.beforeaopOperation"> </bean> <bean id="trackMyBean" class="aspectj.beforeaop"></bean>

<bean

class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean>

</beans>



**8.2) Write a program to demonstrate Spring AOP – after advice.**

**//afteraop.java**

package aspectj;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Pointcut;

@Aspect

public class afteraop {

@Pointcut("execution(int afteroperation.\*(..))") public void p(){}

@After("p()") public void myadvice(JoinPoint jp)

{

System.out.println("after advice"); } }

**//afteeroperation.java**

package aspectj;

public class afteeraopOperation {

public void msg() {System.out.println("method 1");

}

public int m(){System.out.println("method 2 with return");return 2;}

public int k(){System.out.println("method 3 with return");return 3;}

}

**//AfterTest.java**

package aspectj;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class afterTest {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("appctx.xml");

afteeraopOperation e = (afteeraopOperation) context.getBean("opBean");

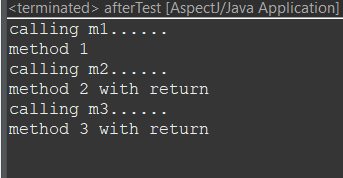
System.out.println("calling m1......"); e.msg();

System.out.println("calling m2......"); e.m();

System.out.println("calling m3......"); e.k();

}

}



**8.3) Write a program to demonstrate Spring AOP – around advice.**

**//bankaopdata.java**

package aspectj;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Pointcut;

@Aspect

public class BankaopData {

@Pointcut("execution(\* Bank.\*(..))") public void a() {}

@Around("a()") public Object myadvice(ProceedingJoinPoint p)throws Throwable

{

System.out.println("Around concern Before calling actual method");

Object obj=p.proceed();

System.out.println("Around Concern After calling actual method"); return obj;

}

}

**//Bank.java**

package aspectj;

public class Bank { public void welcome() {

System.out.println("welcome to bank");

}

public int icici()

{System.out.println("icici bank interest rate");

return 7;

}

public int pnb() {

System.out.println("pnb bank interest rate");

return 6;

}

}

**//BankTest.java**

package aspectj;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class BankTest {

public static ApplicationContext context;

public static void main(String[] arrgs) {

context = new ClassPathXmlApplicationContext("BankaopData.xml");

Bank e =(Bank) context.getBean("opBean");

System.out.println("Calling welcome method..");

e.welcome();

System.out.println("Calling icici method...");

e.icici();

System.out.println("Calling pnb method...");

e.pnb();

}

}

**//BankaopData.xml**

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

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

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

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

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

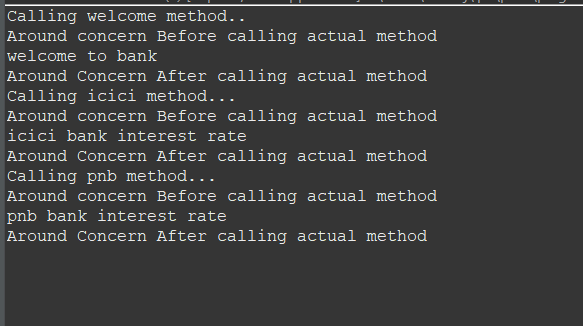
http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">

<bean id="opBean" class="aspectj.Bank" />

<bean id="trackMyBean" class="aspectj.BankaopData" />

<aop:aspectj-autoproxy />

</beans>



**8.4) : Write a program to demonstrate Spring AOP – after returning advice.**

**//Bank.java**

package aspectj;

public class Bank {

public void welcome() {

System.out.println("welcome to bank");

}

public int icici() {

System.out.println("icici bank interest rate");

return 7;} public int pnb() {

System.out.println("pnb bank interest rate");return 6;}

}

**//BankaopData.java**

package aspectj;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Pointcut;

@Aspect

public class BankaopData {

@AfterReturning( pointcut ="execution(\* Bank.\*(..))", returning="result")

public void myadvice(JoinPoint jp,Object result)

{

System.out.println("AfterReturning concern");

System.out.println("Result in advice" +result); }

}

**//BankTest**

package aspectj;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class BankTest {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("BankaopData.xml");

Bank e = (Bank) context.getBean("opBean");

System.out.println("Calling welcome method...");

e.welcome();

System.out.println("Calling icici method...");

e.icici();

System.out.println("Calling pnb method...");

e.pnb();

}

}

**//BankaopData.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans

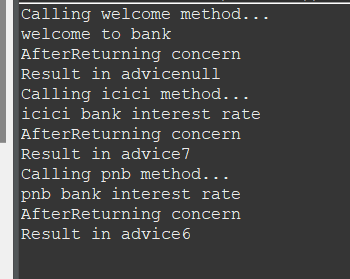
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="opBean" class="aspectj.Bank"> </bean>

<bean id="trackMyBean" class="aspectj.BankaopData"></bean>

<bean

class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean> </beans>



**8.5) Write a program to demonstrate Spring AOP – after throwing advice.**

**//Operationaop.java**

package aspectj;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class Operationaop {

@AfterThrowing( pointcut = "execution(\* Operation\_at.\*(..))", throwing = "error")

public void myadvice(JoinPoint jp, Throwable error)

{

System.out.println("AfterThrowing concern");

System.out.println("Exception is: "+error);

System.out.println("end of after throwing advice....");

}

}

**//Operation\_at.java**

package aspectj;

class Operation\_at {

public void validate(int att)throws Exception{

if(att<75) { throw new ArithmeticException("Not eligible for exam");

}

else {

System.out.println("Eligible for exam"); }

}

}

**//TestValidaation.java**

package aspectj;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class TestValidaation {

private static ApplicationContext context;

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("validctx.xml");

Operation\_at op = (Operation\_at) context.getBean("opBean");

System.out.println("calling validate....");

try { op.validate(85);

}catch(Exception e){

System.out.println(e);}

System.out.println("calling validate again....");

try { op.validate(25);

}catch(Exception e){System.out.println(e);}

}

}

**//validctx.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans

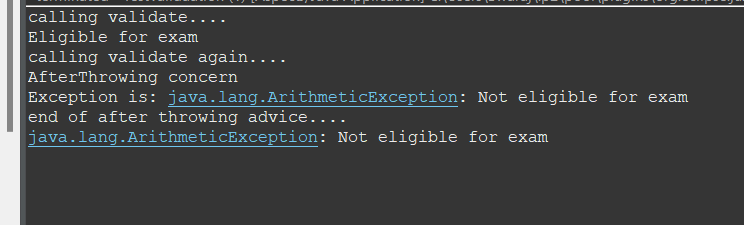
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="opBean" class="aspectj.Operation\_at"></bean>

<bean id="trackMyBean" class="aspectj.Operationaop"></bean>

<bean

class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean></beans>



**8.6) Write a program to demonstrate Spring AOP –pointcuts.**

**//AopData\_pc.java**

package aspectj;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Pointcut;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

@Aspect

public class AopData\_pc {

@Pointcut("execution(int Operation.\*(..))") public void p(){}

@After("p()") public void myadvice(JoinPoint jp)

{

System.out.println("After advice");

}

@Pointcut("execution(\* Operation.\*(..))") public void i(){} @Before("i()")

public void myadvice1(JoinPoint jp)

{

System.out.println("Before advice");

}

}

**//Operation\_pc.java**

package aspectj;

public class Operation\_pc {

public void msg() {

System.out.println("method 1");

}

public int m() {

System.out.println("method 2 with return");

return 2;}

public int k() {

System.out.println("method 3 with return");

return 3;

} }

**//Test\_pc.java**

package aspectj;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test\_pc {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("appctx.xml");

Operation\_pc e=(Operation\_pc)context.getBean("opBean");

System.out.println("calling m1...");

e.msg();

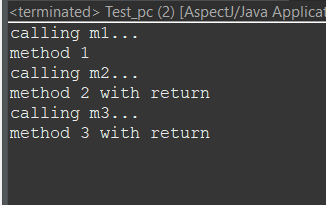
System.out.println("calling m2...");

e.m();

System.out.println("calling m3...");

e.k();

} }



**9. Spring JDBC**

**9.1) Write a program to insert, update and delete records from the given table.**

**//Movie.java**

package Spring1;

public class Movie {

int mid;

String title;

String actor;

public Movie(int mid, String title, String actor) {

super();

this.mid = mid;

this.title = title;

this.actor = actor;

}

public Movie() {

super();

// TODO Auto-generated constructor stub

}

public int getMid() {

return mid;

}

public void setMid(int mid) {

this.mid = mid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getActor() {

return actor;

}

public void setActor(String actor) {

this.actor = actor;

}

}

**//MoviesDao.java**

package Spring1;

import org.springframework.jdbc.core.\*;

public class MovieDao {

JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {

this.jdbcTemplate = jdbcTemplate;

}

public int insMovie(Movie m1)

{

String insSql="insert into mymovies1 values("+m1.getMid()+",'"+m1.getTitle()+"','"+m1.getActor()+"')";

return jdbcTemplate.update(insSql);

}

public int updateMovie(Movie m1){

String query = "update mymovies1 set title='" + m1.getTitle() + "', actor='" + m1.getActor() + "' where mid='" + m1.getMid() + "'";

return jdbcTemplate.update(query);

}

public int deleteMovie(Movie m1){

String query="delete from mymovies1 where mid='"+m1.getMid()+"' ";

return jdbcTemplate.update(query);

}

}

**//MoviesTest.java**

package Spring1;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Movietest {

private static ApplicationContext appCon;

public static void main(String[] args) {

// TODO Auto-generated method stub

appCon = new ClassPathXmlApplicationContext("appctx.xml");

MovieDao m1 = (MovieDao) appCon.getBean("mymovie");

// insert query

Movie t1 = new Movie(4, "17 Again", "Zac");

System.out.println(m1.insMovie(t1));

Movie t = new Movie(5, "Interstellar", "Christopher");

System.out.println(m1.insMovie(t));

// update query

int status = m1.updateMovie(new Movie(1, "18 Again", "Zac"));

System.out.println(status);

// delete

Movie t2=new Movie();

t2.setMid(3);

int s=m1.deleteMovie(t2);

System.out.println(s);

} }

**//appctx.xml**

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

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

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

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

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

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="org.postgresql.Driver" />

<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />

<property name="username" value="postgres" />

<property name="password" value="admin" />

</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">

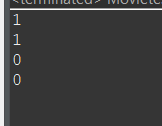
<property name="dataSource" ref="ds"></property>

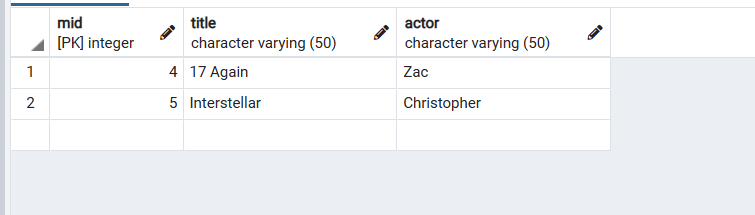
</bean>

<bean id="mymovie" class="Spring1.MovieDao">

<property name="jdbcTemplate" ref="jdbcTemplate"></property>

</bean> </beans>





**9.2) Write a program to demonstrate PreparedStatement in Spring JdbcTemplate.**

**//MovieDao.java**

package Spring1;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import org.springframework.dao.DataAccessException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.PreparedStatementCallback;

public class MovieDao {

JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {

this.jdbcTemplate = jdbcTemplate;

}

public Boolean saveMovieByPreparedStatement(final Movie e){

String query="insert into mymovies1 values(?,?,?)";

return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){

@Override

public Boolean doInPreparedStatement(PreparedStatement ps)

throws SQLException, DataAccessException {

ps.setInt(1,e.getMid());

ps.setString(2,e.getTitle());

ps.setString(3,e.getActor());

return ps.execute();

}

});

}

}

**//Movie.java**

package Spring1;

public class Movie {

int mid;

String title;

String actor;

public Movie(int mid, String title, String actor) {

super();

this.mid = mid;

this.title = title;

this.actor = actor;

}

public Movie() {

super();

}

public int getMid() {

return mid;

}

public void setMid(int mid) {

this.mid = mid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getActor() {

return actor;

}

public void setActor(String actor) {

this.actor = actor;

}

}

**//MovieTest.java**

package Spring1;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Movietest {

private static ApplicationContext appCon;

public static void main(String[] args) {

// TODO Auto-generated method stub

appCon = new ClassPathXmlApplicationContext("appctx.xml");

MovieDao m1=(MovieDao)appCon.getBean("mymovie");

m1.saveMovieByPreparedStatement(new Movie(2,"Happy New Year","Shahrukh"));

}

}

**//appctx.xml**

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

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

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

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

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

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="org.postgresql.Driver" />

<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />

<property name="username" value="postgres" />

<property name="password" value="admin" />

</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">

<property name="dataSource" ref="ds"></property>

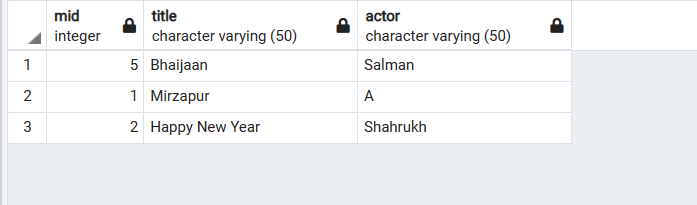
</bean>

<bean id="mymovie" class="Spring1.MovieDao">

<property name="jdbcTemplate" ref="jdbcTemplate"></property>

</bean>

</beans>

****

**9.3) Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface.**

**//Movie2.java**

package Spring1;

public class Movie2 {

int mid;

String title;

String actor;

public int getMid() {

return mid;

}

public void setMid(int mid) {

this.mid = mid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getActor() {

return actor;

}

public void setActor(String actor) {

this.actor = actor;

}

public String toString(){

return mid+" "+title+" "+actor;

}

}

**//MovieDAO2.java**

package Spring1;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import org.springframework.dao.DataAccessException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.ResultSetExtractor;

public class MovieDAO2 {

JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {

this.jdbcTemplate = jdbcTemplate;

}

public List<Movie2> getAllMovie(){

return jdbcTemplate.query("select \* from mymovies1",new

ResultSetExtractor<List<Movie2>>(){

@Override

public List<Movie2> extractData(ResultSet rs) throws SQLException,

DataAccessException {

List<Movie2> list=new ArrayList<Movie2>();

while(rs.next()){

Movie2 e=new Movie2();

e.setMid(rs.getInt(1));

e.setTitle(rs.getString(2));

e.setActor(rs.getString(3));

list.add(e);

}

return list;

}

});

}

}

**//MovieTest2.java**

package Spring1;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest2 {

private static ApplicationContext appCon;

public static void main(String[] args) {

appCon = new ClassPathXmlApplicationContext("appctx2.xml");

MovieDAO2 m1=(MovieDAO2)appCon.getBean("mymovie");

List<Movie2> list=m1.getAllMovie();

for(Movie2 e:list)

System.out.println(e);

}

}

**//appctx2.xml**

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

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

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

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

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

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="org.postgresql.Driver" />

<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />

<property name="username" value="postgres" />

<property name="password" value="admin" />

</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">

<property name="dataSource" ref="ds"></property>

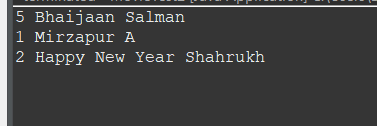
</bean>

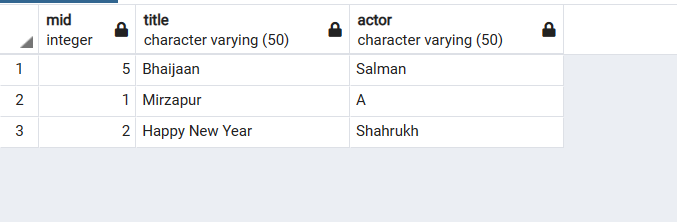
<bean id="mymovie" class="Spring1.MovieDAO2">

<property name="jdbcTemplate" ref="jdbcTemplate"></property>

</bean>

</beans>





**9.4) Write a program to demonstrate RowMapper interface to fetch the**

**records from the database.**

**//Movie3.java**

package org.me;

public class Movie3 {

int mid;

String title;

String actor;

public Movie3(int mid, String title, String actor) {

super();

this.mid = mid;

this.title = title;

this.actor = actor;

}

public Movie3() {

super();

// TODO Auto-generated constructor stub

}

public int getMid() {

return mid;

}

public void setMid(int mid) {

this.mid = mid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getActor() {

return actor;

}

public void setActor(String actor) {

this.actor = actor;

}

}

**//MovieDAO3.java**

package org.me;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.RowMapper;

public class MovieDAO3 {

JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {

this.jdbcTemplate = jdbcTemplate;

}

public List<Movie2> getAllEmployeesRowMapper(){

return jdbcTemplate.query("select \* from mymovies1",new RowMapper<Movie2>(){

@Override

public Movie2 mapRow(ResultSet rs, int rownumber) throws SQLException {

Movie2 e=new Movie2();

e.setMid(rs.getInt(1));

e.setTitle(rs.getString(2));

e.setActor(rs.getString(3));

return e;

}

});

}

}

**//appxtx3.xml**

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

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

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

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"ds"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"org.postgresql.Driver"* />

<property name=*"url"* value=*"jdbc:postgresql://localhost:5432/postgres"* />

<property name=*"username"* value=*"postgres"* />

<property name=*"password"* value=*"password"* />

</bean>

<bean id=*"jdbcTemplate"* class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<property name=*"dataSource"* ref=*"ds"*></property>

</bean>

<bean id=*"mymovie"* class=*"org.me.MovieDAO3"*>

<property name=*"jdbcTemplate"* ref=*"jdbcTemplate"*></property>

</bean>

</beans>

**//MovieTest3.java**

package org.me;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest3 {

private static ApplicationContext appCon;

public static void main(String[] args) {

appCon = new ClassPathXmlApplicationContext("appctx3.xml");

MovieDAO3 m1=(MovieDAO3)appCon.getBean("mymovie");

List<Movie2> list=m1.getAllEmployeesRowMapper();

for(Movie2 e:list)

System.out.println(e);

}

}

**Output :**

Graphical user interface, text, application, Word

Description automatically generated

Table

Description automatically generated

**10. Spring Boot and RESTful Web Services**

**10.1)** **Write a program to create a simple Spring Boot application that prints a message.**

**//BoothelloApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class BoothelloApplication {

public static void main(String[] args) {

SpringApplication.run(BoothelloApplication.class, args);

}

}

**//HelloWorldController.java**

package com.example.demo;

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

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

@RestController

public class HelloWorldController {

@RequestMapping("/")

public String hello()

{

return "Vinit is here !";

}

}

Graphical user interface, text, application, chat or text message

Description automatically generated

**10.2)** **Write a program to demonstrate RESTful Web Services with spring boot**

**//HelloWorldBean.java**

package com.example.demo;

public class HelloWorldBean {

public String message;

//constructor of HelloWorldBean

public HelloWorldBean(String message)

{

this.message=message;

}

//generating getters and setters

public String getMessage()

{

return message;

}

public void setMessage(String message)

{

this.message = message;

}

@Override

//generate toString

public String toString()

{

return String.*format* ("HelloWorldBean [message=%s]", message);

}

}

**//HelloWorldController.java**

package com.example.demo;

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

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

//Controller

@RestController

public class HelloWorldController

{

//using get method and hello-world as URI

@GetMapping(path="/hello-world")

public String helloWorld()

{

return "Vinit is here!";

}

@GetMapping(path="/hello-world-bean")

public HelloWorldBean helloWorldBean()

{

return new HelloWorldBean("Kaise ho? xD"); //constructor of HelloWorldBean } }

**//RestfulwebserviceApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestfulwebserviceApplication {

public static void main(String[] args) {

SpringApplication.run(RestfulwebserviceApplication.class, args);

}

}

**Output :**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Testing API with PostMan.

A screenshot of a computer screen

Description automatically generated with low confidenceEndPoint : <http://localhost:9999/hello-world-bean>

Graphical user interface, application

Description automatically generated