Practical 1

PROGRAM ON JAVA GENERICS

Program 1
AIM: WRITE A JAVA PROGRAM TO DEMONSTRATE GENERIC CLASS.
Steps:
D DRIVE>JAVA FOLDER>ECLIPSE>NEW>JAVA PROJECT>PACKAGE NAME:genericprogram>Click on Don't create>CLICK ON YOUR PACKAGE NAME i.e genericprogram > click on src > new > class name it as Box

CODE:

```
package genericprogram;
import java.io.*;
import java.util.*;
public class Box <T>{
private T t;
public void add(T t)
{
this.t=t;
}
public T get()
{
return t;
}
public static void main(String[]args) {
Box<Integer>integerBox = new Box<Integer>();
Box<String>stringBox = new Box<String>();
integerBox.add(new Integer(10));
stringBox.add(new String("HELLO WORLD"));
System.out.println("Integer value:%d\n\n"+integerBox.get());
System.out.println("String value:%s\n"+stringBox.get());
}
}
```

Program 2

AIM: Write a JAVA program to demonstrate Generic Method.
Steps:

Click on your package name()>click on src>new>class>name:Genemethod
Code:

```
package genericprogram;
```

```
import java.io.*;
import java.util.*;
public class Genemethod {
public static<E> void printArray(E[]arr){
for(E element:arr) {
System.out.printf("%S",element);

}
System.out.println();
}
public static void main(String[]args)
{
Integer[]intarr = {1,2,3,4,5};
Double[]darr = {1.1,2.2,3.3,4.4,5.5};
Character[]carr = {'H','E','L','L','O'};
System.out.println("INTEGER ARRAY:");
printArray(intarr);
System.out.println("DOUBLE ARRAY:");
printArray(darr);
System.out.println("CHARACTER ARRAY:");
printArray(carr);
}
}
```

Output:

Program 3

AIM: Write a JAVA program to demonstrate WildCrad of Unknown type.
Code:

```
package genericprogram;
import java.io.*;
import java.util.*;
public class Unknown {
static void processElements(ArrayList<?>a)
{
for(Object element:a)
{
System.out.println(element);
}
}
public static void main(String[]args) {
```

```
ArrayList<Integer>a1=new ArrayList<>();
a1.add(10);
a1.add(20);
a1.add(30);
processElements(a1);
ArrayList<String>a2=new ArrayList<>();
a2.add("ONE");
a2.add("TWO");
a2.add("THREE");
processElements(a2);
}
}
```

Output:

Programs based on List Interface

Practical No: 2

Program No:1
Aim: Write a JAVA program to Create List and demonstrate all Operation of List.
a. Add element
b. Appending list elements.
c. clear/empty the list.
d. Size of List
e. Updating elements in a List using set
f. Extracting a portion of a list
g. Removing elements from a list
h. Searching for an element in a list
i. Sorting a list
j. Copying elements from one list into another
k. Shuffling elements in a list
l. Reversing elements in a list.

Code:-

```
import java.util.*;
public class List_interface {
public static void main(String args[])
{
List<String> vowels = new ArrayList<String>(25);
//add example
vowels.add("A");
vowels.add("I");
//lets insert E between A and I
vowels.add(1,"E");
List<String>list = new ArrayList<String>();
list.add("O");
list.add("U");
//appennding list elements to letters
vowels.addAll(list);
System.out.println("Element in vowels list After using addAll()="+vowels);
//clear example to empty the list
System.out.println("Before clear method the list object elment="+list);
list.clear();
System.out.println("After clear method the list object element="+list);
System.out.println("Vowels list size="+vowels.size());
//updating elements in a list using set
vowels.set(2, "X");
```

```
System.out.println("Elements in vowels list after using set()"+vowels);
//Extracting a portion of a list
/* The sublist(fromIndex,toIndex) allows us to get a portion of the list between
the specified fromIndex(inclusive) and toIndex(exclusive)*/
list = vowels.subList(2,4);
System.out.println("Element in vowels list ="+vowels+", Element in list="+list);
System.out.println();
vowels.set(0, "A");
System.out.println("Element in vowels list ="+vowels+", Element in list="+list);
list.add("U");
System.out.println("Element in vowels list ="+vowels+", Element in list="+list);
System.out.println();
list.add("A");
//Removing elements from a list
System.out.println("Element in list before remove()="+list);
if(list.remove("A")) {
System.out.println("Element is removed");
}else {
System.out.println("There is no such element");
}
System.out.println("Element in list after remove()="+list);
System.out.println();
vowels.add("O");
vowels.add("U");
vowels.add("A");
vowels.add("U");
System.out.println();
System.out.println("Element in vowel list="+vowels);
System.out.println();
//searching for an element in a list
if(vowels.contains("U"))
{
System.out.println("Found the element");
}
else {
System.out.println("There is non such element");
}
System.out.println();
int firstIndex=vowels.indexOf("A");
System.out.println("first index of A is:"+firstIndex);
System.out.println();
int lastIndex = vowels.lastIndexOf("U");
System.out.println("Last index of U is:"+lastIndex);
//sorting a list
```

```
System.out.println();
System.out.println("listStrings before sorting:"+vowels);
Collections.sort(vowels);
System.out.println("listStrings after sorting:"+vowels);
System.out.println();
//Copying elements from one list into another
List<String> sourceList = new ArrayList<String>();
sourceList.add("A");
sourceList.add("B");
sourceList.add("C");
sourceList.add("D");
List<String> destList = new ArrayList<String>();
destList.add("V");
destList.add("W");
destList.add("X");
destList.add("Y");
destList.add("Z");
System.out.println("destList before copy:"+destList);
Collections.copy(destList, sourceList);
System.out.println("destList after copy:"+destList);
//shuffling elements in a list
System.out.println("Vowels List before shuffling:"+vowels);
Collections.shuffle(vowels);
System.out.println("vowels list after shuffling:"+vowels);
System.out.println();
//Reversing elements in a list
System.out.println("vowels List before reversing:"+vowels);
Collections.reverse(vowels);
System.out.println("vowels List after reversing:"+vowels);
System.out.println();

}
}
```

Output:

Program No : 2

Aim: Write a JAVA program to Create List containing list of items and use ListIteration interface to print
items present in the list.
Code:-

```
import java.util.*;
public class ListIteratorExample {
public static void main(String args[]) {
List <Integer> list = new ArrayList<>();
for(int i=0;i<5;i++)
list.add(i);
Iterator<Integer> iterator = list.iterator();
//simple iteration
while(iterator.hasNext())
{
int i = (int) iterator.next();
System.out.println(i+",");
}
System.out.println("\n"+list);
//modification of list using iterator
iterator=list.iterator();
while(iterator.hasNext())
{
int x = (int) iterator.next();
if(x%2==0)
iterator.remove();

}
System.out.println(list);
//changing list structure while iterating
iterator = list.iterator();
/*
while(iterator.hasNext())
{
int x = (int) iterator.next(); ////concurrenModificationException here
if(x==1)
list.add(10);
}*/
}
}
```

Practical no: 3

Programs based on Set Interface

Program no: 1
Aim:Write a program to Create a set containing list of items al type String and print the items in
the List using Iterator interface also print the list in Reverse/backward direction.
Code:

```
import java.util.*;
public class setExample {
public static void main(String[] args) {
//create a set containing String items
Set<String> itemset = new HashSet<>();
itemset.add("Apple");
itemset.add("Banana");
itemset.add("Cherry");
itemset.add("Date");
itemset.add("Elderberry");
//print items using Iterator
System.out.println("Item in the set:");
Iterator<String> iterator = itemset.iterator();
while(iterator.hasNext()) {
System.out.println(iterator.next());
}
//convert the set to a list to allow reverse iteration
List<String> itemList = new ArrayList<>(itemset);

//print items inb reverse order
System.out.println("\nItems in reverse order:");
for(int i = itemList.size() -1; i>= 0; i--) {
System.out.println(itemList.get(i));
}
}
}
```

Program no: 2
Aim: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 into other set
c) Remove items from the set
d) Search the specified item in the set
Code:

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;
public class setoperatorExample {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
//create a set to store items
Set<String> itemset = new HashSet<>();
// a. Add items to the set
System.out.println("Adding items to the set. Type 'exit' to stop addding.");
while(true) {
System.out.println("Enter items:");
String item = scanner.nextLine();
if(item.equalsIgnoreCase("exit")) {
break;
}
itemset.add(item);
}
//b. Insert items of one set into other set
Set<String> anotherSet = new HashSet<>();
anotherSet.add("Grapes");
anotherSet.add("orange");
anotherSet.add("pineapple");
itemset.addAll(anotherSet);
System.out.println("\nItems after inserting another set:" + itemset);
//c. Remove items from the set
System.out.println("\nEnter item to remove from the set:");
String itemToRemove = scanner.nextLine();
if(itemset.remove(itemToRemove)) {
System.out.println(itemToRemove +"removed from the set.");
}
else {
System.out.println(itemToRemove + "not found in the set.");
}
System.out.println("current items in the set: " + itemset);
```

```
//d. Search for a specified item in the set
System.out.println("\nEnter item to search in the set: ");
String itemToSearch = scanner.nextLine();
if(itemset.contains(itemToSearch)) {
System.out.println(itemToSearch + " is found in the set.");
}else {
System.out.println(itemToSearch + " is not found in the set.");
}
//close the scanner
scanner.close();
}
}
```

Practical no :4
Programs based Map Interface

Program no :1
Aim: Write down 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 for a specific key from the Map.
d) Get values of the Specified key.
e) Insert map elements of one map into another map.
f) Print all keys and values of the map.
Code:

```
import java.util.*;
public class MapOperationsExample {
public static void main(String[]args) {
Scanner scanner = new Scanner(System.in);
// Create a Map to store items
Map<String, String> itemMap = new HashMap<>();
//a. Add items in the map
System.out.println("Adding items to the map.Type 'exit' to stop adding.");
while(true) {
System.out.println("Enter key: ");
String key = scanner.nextLine();
if(key.equalsIgnoreCase("exit")) {
break;
}
System.out.println("Enter value: ");
String value = scanner.nextLine();
itemMap.put(key,value);
}
//b. Remove items from the Map
System.out.print("\nEnter key to remove from the map: ");
String keyToRemove = scanner.nextLine();
if(itemMap.remove(keyToRemove)!=null) {
System.out.println(keyToRemove + " removed from the map.");
}else {
System.out.println(keyToRemove + " is not found in the map.");
}
//c. Search specific key from the Map
System.out.print("\nEnter key to search in the map: ");
String keyToSearch = scanner.nextLine();
if(itemMap.containsKey(keyToSearch)) {
```

```
System.out.println(keyToSearch + " is found in the map.");
}else {
System.out.println(keyToSearch + " is not found in the map.");
}
//d. Get value of the specified key.
System.out.print("\nEnter key to get its value: ");
String keyToGetValue = scanner.nextLine();
String value = itemMap.get(keyToGetValue);
if( value!=null){
System.out.println("Value for key '"+ keyToGetValue + "' is: "+ value);
}else {
System.out.println("Key '" + keyToGetValue + "' not found.");
}
//e. Insert map elements of one map into another map.
Map<String, String> anotherMap = new HashMap<>();
anotherMap.put("Orange","Fruit");
anotherMap.put("Carrot","Vegetable");
itemMap.putAll(anotherMap);
System.out.println("\nItems after inserting another map: "+itemMap);
//f. Print all keys and values of the map
System.out.println("\nAll keys and values in the map: ");
for(Map.Entry<String, String>entry : itemMap.entrySet()) {
System.out.println(" key: "+entry.getKey()+", Value: "+entry.getValue());
}
//Close the scanner
scanner.close();
}
}
```

.Practical No: 05
Programs based on Lambda Expression

Program No: 01
AIM: WAP using Lambda Expression to print “hello World”
Code:

```
interface Hello{
void sayHello();
}
public class HelloWorld {
public static void main(String[] args) {
Hello hello=()->System.out.println("Hello World");
hello.sayHello();
```

```
}
}
```

Output:

Program No: 02
AIM: WAP using Lambda Expression to concatenate two strings.
Code:

```
interface Concatenate{
String join(String s1,String s2) ;
}
public class Concatenation {
public static void main(String[] args) {
Concatenate concat=(s1,s2)->s1+s2;
System.out.println(concat.join("Hello","World"));
}
}
```

Output:

Program No: 03
AIM: WAP using Lambda Expression with single parameter.
Code:

```
interface SingleParameter{
void display(String message);
}
public class SingleParameterExample {
public static void main(String[] args) {
SingleParameter displayMessage=message-> System.out.println("Message:"+message);
displayMessage.display("Lambda with single parameter");
}
}
```

Output:

Program No :04
AIM: WAP using Lambda Expression with multiple parameters to add two numbers.
Code:

```
interface Add{
int sum(int a, int b);
}
public class MultipleParameter {
public static void main(String[] args) {
Add addition=(a,b)->a+b;
System.out.println("Sum:"+addition.sum(19, 10));
```

```
}
}
```

Program no 05:
AIM: WAP using lambda expression to calculate following
a) Convert fahrenheit to celsius.
Code:

```
interface FahrenheitToCelsius{
double convert(double fahrenheit);
}
public class TemperatureConversion {
public static void main(String[] args) {
FahrenheitToCelsius convert=f->(5.0/9)*(f-32);
System.out.println("Celsius:"+convert.convert(98.6));
}
}
```

Output:

b) Convert kilometer to miles.
Code:

```
interface KilometersToMiles{
double convert(double km);
}
public class DistanceConversion {
public static void main(String[] args) {
KilometersToMiles convert=km->km*0.621371;
System.out.println("Miles:"+ convert.convert(10));
}
}
```

Output:

Program no : 06
AIM: WAP using lambda exp with and without return keyword.
A] With Return
Code:

```
interface Multiply{
int product(int a, int b);
}
public class WithReturn {
public static void main(String[] args) {
Multiply multiply=(a,b)-> {
return a*b;
```

```
};
System.out.println("Product of a and b :"+multiply.product(5, 10));
}
}
```

Output:

B] Without Return

Code:

```
interface Diffrence{
int Subtract(int a, int b);
}
public class WithoutReturn {
public static void main(String[] args) {
Diffrence subtract=(a,b)->a-b;
System.out.println("multiplicaton of three numbers is:"+subtract.Subtract(20,3));
}
}
```

Output:

Practical no. 6

Steps: file-> new->dynamic webpage->select the target runtime (Apache Tomcat version) ->create project-> right click on project -> new-> class-> then run the code-> while running the program select tomcat if required.

Programs based on web application development using JSP

Program no. 1:- jspDemo1
AIM : Write a jsp program to demonstrate directives declaration on jsp page.

Index.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-Tyoe" content="text/html; charset=UTF-8">
<title>Page Attribute </title>
</head>
<body>
<form action="directive.jsp">
<h1> Enter the value of n1 and n2: </h1>
Number1: <input type="number" name="n1"/><br/>
Number2: <input type="number" name="n2"/><br/>
<input type="submit"/>
</form>
</body>
</html>
```

error.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page isErrorPage="true" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-Type" content="text/html; charset=UTF-8">
<title>Page Attributes</title>
</head>
<body>

<h2> Value of n2 variable of zero(n/0 is infinity)</h2>
<h3> Sorry an exception occurred!</h3> <br/>
```

```
<h3> The Exception is: <%= exception %></h3>
</body>
</html>
```

directive.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ page info="composed by DSATM" %>
<%@ page language="java" %>
<%@ page buffer="16kb" %>
<%@ page autoFlush="true" %>
<%@ page isThreadSafe="true" %>
<%@ page errorPage="error.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-Type" content="text/html; charset=UTF-8">
<title>Page Attributes</title>
</head>
<body bgcolor="orange">
<h2> Usage of import attributes</h2>
<h2> Todays Date is: <%=new Date() %></h2>
<h2> To see the use of error page enter n2 value zero and click submit</h2>
<%
int n1=Integer.parseInt(request.getParameter("n1"));
int n2=Integer.parseInt(request.getParameter("n2"));
%>
<h2> Value of n1/n2 :<%=n1/n2 %></h2>
</body>
</html>
```

output:

Program no. 2:- jspDemo2

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

Sonia.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"
%>
<%@ include file="header.html" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Demo Page</title>
</head>
<body>
<h1>Enter the Message</h1>
<form action="Response.jsp" method="post">
Enter Message: <input type="text" value="" name="text1">
<input type="submit" value="Submit" />
</form>
</body>
</html>
<%@ include file="footer.html" %>
```

Response.jsp

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

<%@ include file="header.html" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
```

```
<body>
<% String s1=request.getParameter("text1"); %>
<%=s1%>
</body>
</html>
<%@include file="footer.html" %>
```

header.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1> hello welcome back to jsp</h1>
</body>
</html>
```

footer.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h3>visit again</h3>
</body>
</html>
```

Program no. 3 calculator
AIM: - Design Grade calculator using JSP which accepts Marks of subjects. Calculate average of marks
and Display the grade of student based on average marks.

```
<%@ 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>
<h2>Grade Calculator-------- Main content of the page</h2>
<form action="" method="post">
<table>
<tr><td>Program: MCA</td>
<td></td></tr>
<tr><td>JAVA (out of 100):</td>
<td><input type="text" value="" name="text1"></td></tr>
<tr><td>ADBMS (out of 100):</td>
<td><input type="text" value="" name="text2"></td></tr>
<tr><td>SPM (out of 100):</td>
<td><input type="text" value="" name="text3"></td></tr>
<tr><td>MATHS (out of 100):</td>
<td><input type="text" value="" name="text4"></td></tr>
<tr><td></td>
<td><input type="submit" value="Submit" name="Submit" /></td></tr>
<tr>
</table>
</form>
<% try {
String res="";
int t1=Integer.parseInt(request.getParameter("text1"));
int t2=Integer.parseInt(request.getParameter("text2"));
int t3=Integer.parseInt(request.getParameter("text3"));
int t4=Integer.parseInt(request.getParameter("text4"));
float r, av;
```

```
r=t1+t2+t3+t4;
av=r/4;
if(av>75 && av<=100){
res="You have passed with O Grade."; }
else if(av>60 && av<=75){
res="You have passed with A Grade."; }
else if(av>50 && av<=60){
res="You have passed with B Grade."; }
else if(av>40 && av<=50){
res="You have passed with C Grade."; }
else{
res="You have Failed."; }
%>
<%= av%>
<br>
<%= res %>
<% } catch(Exception e) { } %>
</body>
</html>
```

Program no. 4 Session
AIM: Write a program in JSP to demonstrate session tracking technique.

Session.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>
<%@ page import = "java.io.*,java.util.*" %>
<%
// Get session creation time.
Date createTime = new Date(session.getCreationTime());
// Get last access time of this Webpage.
Date lastAccessTime = new Date(session.getLastAccessedTime());
String title = "Welcome Back to my website";
Integer visitCount = new Integer(0);
String visitCountKey = new String("visitCount");
String userIDKey = new String("userID");
String userID = new String("ABCD");
```

```
// Check if this is new comer on your Webpage
if (session.isNew() ){
title = "Welcome to my website";
session.setAttribute(userIDKey, userID);
session.setAttribute(visitCountKey, visitCount);
}
visitCount = (Integer)session.getAttribute(visitCountKey);
visitCount = visitCount + 1;
userID = (String)session.getAttribute(userIDKey);
session.setAttribute(visitCountKey, visitCount);
%>
<body>
<center>
<h1>Session Tracking</h1>
</center>
<table border = "1" align = "center">
<tr bgcolor = "#949494">
<th>Session info</th>
<th>Value</th>
</tr>
<tr>
<td>id</td>
<td><% out.print( session.getId()); %></td>
</tr>
<tr>
<td>Creation Time</td>
<td><% out.print(createTime); %></td>
</tr>
<tr>
<td>Time of Last Access</td>
<td><% out.print(lastAccessTime); %></td>
</tr>
<tr>
<td>User ID</td>
<td><% out.print(userID); %></td>
</tr>
<tr>
<td>Number of visits</td>
<td><% out.print(visitCount); %></td>
</tr>
</table>
</body>
</html>
```

output:

Program no. 5 Student Record
AIM : WAP to Insert records in Student Master.
PgADMIN

```
CREATE TABLE student(
sname varchar(10),
srollno varchar(50),
scoll varchar (50),
sadd varchar(50)
);
select * from student;
```

student.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>
<h1>student master</h1>
<form action="response.jsp" method="post">
<table>
<tr><td>Student Name :</td>
<td><input type="text" name="text1" /></td></tr>
<tr><td>Roll No : </td>
<td><input type="text" name="text2" /></td></tr>
<tr><td>Name of the College:</td>
<td><input type="text" name="text3" /></td></tr>
<tr><td>Address :</td>
<td><input type="text" name="text4" /></td></tr>
</table>
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

response.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>
<%@page import="java.sql.*" %>
<% try {
Class.forName("org.postgresql.Driver");
System.out.println("Drivers Registered");
Connection con=DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/postgres","postgres","abc");
String t1, t2, t3, t4;
t1=request.getParameter("text1");
t2=request.getParameter("text2");
t3=request.getParameter("text3");
t4=request.getParameter("text4");
PreparedStatement ps;
ps=con.prepareStatement("insert into student(sname,srollno,scoll,sadd) values(?, ?, ?, ?);");
ps.setString(1, t1);
ps.setString(2, t2);
ps.setString(3, t3);
ps.setString(4, t4);
ps.executeUpdate();
out.println("Record inserted successfully.");
Statement st=con.createStatement();
String sql="select * from student";
ResultSet rs=st.executeQuery(sql);
while(rs.next()) {
String n1=rs.getString(1);
String n2=rs.getString(2);
String n3=rs.getString(3);
String n4=rs.getString(4);
%> <br>
<% out.println(n1+" "+n2+" "+n3+" "+n4);}
} catch(Exception e){ out.println(e); } %>
</body>
</html>
```

Program no. 6

Delete Student Record
AIM WAP to Delete records in Student Master.

Delete.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>
<h1>student master</h1>
<form action="response1.jsp" method="post">
<table> <tr><td>Student Name :</td>
<td><input type="text" name="text1" /></td></tr>
<tr><td>Roll No : </td>
<td><input type="text" name="text2" /></td></tr>
<tr><td>Name of the College:</td>
<td><input type="text" name="text3" /></td></tr>
<tr><td>Address :</td>
<td><input type="text" name="text4" /></td></tr>
</table>
<input type="submit" value="Submit" />
</form> </body> </html>
```

response1.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>
<%@page import="java.sql.*" %>
<%
try {
Class.forName("org.postgresql.Driver");
System.out.println("Drivers Registered");
Connection con=DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/postgres","postgres","");
String t1, t2, t3, t4;
t1=request.getParameter("text1");
```

```
t2=request.getParameter("text2");
t3=request.getParameter("text3");
t4=request.getParameter("text4");
PreparedStatement ps;
//insert into student(sname,srollno,scoll,sadd) values(?, ?, ?, ?);
ps=con.prepareStatement("DELETE FROM student where srollno= ?;");
ps.setString(1, t2);
/*ps.setString(1, t1);
ps.setString(3, t3);
ps.setString(4, t4);*/
ps.executeUpdate();
//out.println("Record inserted successfully.");
Statement st=con.createStatement();
String sql="select * from student";
ResultSet rs=st.executeQuery(sql);
while(rs.next()) {
String n1=rs.getString(1);
String n2=rs.getString(2);
String n3=rs.getString(3);
String n4=rs.getString(4);
%> <br>
<% out.println(n1+" "+n2+" "+n3+" "+n4 ); }
} catch(Exception e){ out.println(e); } %>
</body> </html>
```

Program no. 7

Registration form
AIM Write a program to create registration form[JSP to Database
pgADMIN

```
CREATE TABLE register(
uid varchar(20),
pass varchar(10),
fname varchar(50),
lname varchar (50),
email varchar(50)
);
Select * from register
```

register.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>

<h1>Registration Form</h1>
<form action="response.jsp" method="post">
<table> <tr><td>UserID:</td>
<td><input type="text" name="text1" /></td></tr>
<tr><td>Password:</td>
<td><input type="password" name="text2" /></td></tr>
<tr><td>First Name:</td>
<td><input type="text" name="text3" /></td></tr>
<tr><td>Last Name:</td>
<td><input type="text" name="text4" /></td></tr>
<tr><td>Email-id:</td>
<td><input type="email" name="text5" /></td></tr>
</table>
<input type="submit" value="Submit" />
</form> </body> </html>
```

response.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>
<%@page import="java.sql.*" %>
<%
try {
Class.forName("org.postgresql.Driver");
System.out.println("Registered");
Connection con=DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/postgres","postgres","abc");
String t1, t2, t3, t4, t5;
t1=request.getParameter("text1");
t2=request.getParameter("text2");
t3=request.getParameter("text3");
t4=request.getParameter("text4");
t5=request.getParameter("text5");
PreparedStatement ps;
```

```
ps=con.prepareStatement("insert into register(uid, pass, fname, lname, email) values(?, ?, ?, ?,
?);");
ps.setString(1, t1);
ps.setString(2, t2);
ps.setString(3, t3);
ps.setString(4, t4);
ps.setString(5, t5);
ps.executeUpdate();
out.println("Record inserted successfully.");
Statement st=con.createStatement();
String sql="select * from register";
ResultSet rs=st.executeQuery(sql);
while(rs.next()) {
String n1=rs.getString(1);
String n2=rs.getString(2);
String n3=rs.getString(3);
String n4=rs.getString(4);
String n5=rs.getString(5);
%>
<br>
<% out.println(n1+" "+n2+" "+n3+" "+n4 ); }
} catch(Exception e){ out.println(e); } %>
</body> </html>
```

output:

Practical no. 7

Programs based on Assignment based Spring Framework

Program no. 1
AIM:
Write a program to print “Hello World” using spring framework.
Steps to Create a Spring "Hello World" Application
Step 1: Create a New Maven Project

1. Open Eclipse.(2024)
2. Go to File > New > Other....
3. In the wizard, select Maven > Maven Project and click Next.
4. Select Create a simple project (skip archetype selection) and click Next.
5. Fill in the Group Id (e.g., com.example) and Artifact Id (e.g., hello-spring) and click Finish.

Step 2: Update pom.xml
Open the pom.xml file in your project and add the Spring dependencies. Here’s a basic example:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>hello-spring</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.3.10</version> <!-- Check for the latest version -->
</dependency>
</dependencies>
```

```
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.1</version>
<configuration>
<source>1.8</source>
<target>1.8</target>
</configuration>
</plugin>
</plugins>
</build>
</project>
```

Step 3: Create the Application Class

1. Right-click on the src/main/java directory, select New > Package, and name it Com.example.hellospring.

2. Right-click on the newly created package, select New > Class, and name it HelloWorldApp. Here's a simple example of what the class might look like:

```
package com.example.hellospring;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class HelloWorldApp {
public static void main(String[] args) {
ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);
HelloWorld helloWorld = context.getBean(HelloWorld.class);
helloWorld.sayHello();
}
}
```

Step 4: Create the Configuration Class
1. Right-click on the same package, select New > Class, and name it AppConfig. Here's an example of what the configuration class might look like:

```
package com.example.hellospring;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
@Configuration
public class AppConfig {
@Bean
```

```
public HelloWorld helloWorld() {
return new HelloWorld();
}
}
```

Step 5: Create the HelloWorld Class

1. Right-click on the same package, select New > Class, and name it HelloWorld.

Here's how it might look:

```
package com.example.hellospring;
public class HelloWorld {
public void sayHello() {
System.out.println("Hello, World!");
}
}
```

Step 6: Run the Application

1. Right-click on the HelloWorldApp.java file and select Run As > Java Application.

2. You should see Hello, World! printed in the console.

Program no. 2

AIM

Write a program to demonstrate dependency injection via setter method.

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>SpringDIExample</artifactId>
<version>1.0-SNAPSHOT</version>
<properties>
<spring.version>5.3.22</spring.version>
<java.version>1.8</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
```

```
<version>${spring.version}</version>
</dependency>
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-api</artifactId>
<version>1.7.30</version>
</dependency>
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-simple</artifactId>
<version>1.7.30</version>
</dependency>
</dependencies>
</project>
```

V8Engine.java

```
package di_program;
public class V8Engine implements Engine {
@Override
public void start() {
System.out.println("V8 Engine is starting...");
}
}
```

Interface

```
package di_program;
public interface Engine {
void start();
}
```

Car.java

```
package di_program;
public class Car {
private Engine engine;
// Setter method for dependency injection
public void setEngine(Engine engine) {
this.engine = engine;
}
public void startCar() {
if (engine != null) {
engine.start();
System.out.println("Car is ready to go!");
} else {
System.out.println("Engine is not set. Cannot start the car.");
```

```
}
}
}
```

Appconfig.java

```
package di_program;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
@Configuration
public class AppConfig {
@Bean
public Engine engine() {
return new V8Engine(); // Create and return the V8Engine bean
}
@Bean
public Car car() {
Car car = new Car();
car.setEngine(engine()); // Inject the engine using the setter method
return car;
}
}
```

Main.java

```
package di_program;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class Main {
public static void main(String[] args) {
// Create the application context from the configuration class
ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);
// Retrieve the car bean
Car car = context.getBean(Car.class);
// Start the car
car.startCar();
}
}
```

Output

Program no. 3
AIM : Write a program to demonstrate dependency injection via Constructor.

Step 1: Create a New Maven Project
1. Open Eclipse.(from D drive)

2. Go to File > New > Spring Legacy Project
3. Give project name org.viva ,select simple java-> finish.
4. Expand Project->select src->create a package-> org.viva(if name is coming by default just click on finish).
5. Right click on package-> new->Class->Account.java
6. Again-> Right click on package-> new->Class->AccountTest.java
7. Select Src->Right Click->new->other->Bean Configuration File ->give name.
8. Select project-> right click->build path>configure build path->classpath-> add external jar->from d drive->open spring RELEASE->libs->select all jar files->apply->apply and close.
9. Run as JAVA Application-> Account.Test file.
10. Follow same steps for other program too.

Account.java

```
package org.viva;
public class Account {

int acNo;
String acName;
double acbalance;
/**
* @return the acNo
*/
public int getAcNo() {
return acNo;
}
/**
* @param acNo the acNo to set
*/
public void setAcNo(int acNo) {
this.acNo = acNo;
}
/**
* @return the acName
*/
public String getAcName() {
return acName;
}
/**
* @param acName the acName to set
*/
public void setAcName(String acName) {
this.acName = acName;
}
```

```
/**
* @return the acbalance
*/
public double getAcbalance() {
return acbalance;
}
/**
* @param acbalance the acbalance to set
*/
public void setAcbalance(double acbalance) {
this.acbalance = acbalance;
}
public Account(int acNo, String acName, double acbalance) {
super();
this.acNo = acNo;
this.acName = acName;
this.acbalance = acbalance;
}
public Account() {
super();
}
}
```

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="Account" class="org.viva.Account">

<constructor-arg type="int" value="000001" >
</constructor-arg>
<constructor-arg type="String" value="Priya">
</constructor-arg>
<constructor-arg type="double" value="2300">
</constructor-arg>
</bean>
</beans>
```

AcoountTest.java

```
package org.viva;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
import org.springframework.context.ApplicationContext;

public class AccountTest {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext ctx=new ClassPathXmlApplicationContext("appctx.xml");
Account a1=(Account) ctx.getBean("Account");
System.out.println("Ac NO:"+a1.getAcNo());
System.out.println("Ac Name:"+a1.getAcName());
System.out.println("Ac Balance:"+a1.getAcbalance());
}
}
```

Program no. 4
AIM : Write a program to demonstrate Autowiring.

Engine.java

```
package myspring.viva;
import org.springframework.stereotype.Component;
@Component
public class Engine {
public void start() {
System.out.println("Engine started!");
}
}
```

Car.java

```
package myspring.viva;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component
public class Car {
private Engine engine;
// Autowire the Engine class into the Car class
@Autowired
public Car(Engine engine) {
this.engine = engine;
}
public void drive() {
engine.start();
System.out.println("Car is moving!");
}
```

```
}
```
Spring-config.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"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
<!-- Enable component scanning for the 'myspring.viva' package -->
<context:component-scan base-package="myspring.viva" />
</beans>
```

MainApp.java

```
package myspring.viva;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {
public static void main(String[] args) {
ApplicationContext context = new ClassPathXmlApplicationContext("spring-config.xml");
// Get the Car bean from the Spring container
Car car1 = context.getBean(Car.class);
car1.drive();
((ClassPathXmlApplicationContext) context).close();
}
}
```

Practical No:08

Spring AOP based program

Steps: file-> new-> maven project-> create simple project -> next
Give the Group id and Artifact id-> finish .
Go to src/main/java->new->create all classes.
Go to src/main/resources->new->spring bean configuration file-> create (appctx.java) change file.-> make changes in pom.xml file.

Program no. 1 Write a program to demonstrate Spring AOP – before advice..

Logging.java

```
package org.mca1;
public class Logging {
public void beforeAdvice() {
System.out.println("Going to setup student profile.");
}
}
```

Student.java

```
package org.mca1;
public class Student {
int age;
String name;
public int getAge() {
return age;
}
public void setAge(int age) {
this.age = age;
}
public String getName() {
return name;
}
```

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

StudentTest.java

```
package org.mca1;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.io.*;
public class StudentTest {
private static ApplicationContext context;
public static void main(String[] args) {
// TODO Auto-generated method stub
context = new ClassPathXmlApplicationContext("appctx.xml");
Student student = (Student) context.getBean("student");
String abc=student.getName();
System.out.println("Name is:"+abc);
int ag=student.getAge();
System.out.println("Age is:"+ag);
}
}
```

In src/main/resources

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"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:context="http://www.springframework.org/schema/context"
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-4.3.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">
<aop:config>
<aop:aspect id = "log" ref = "logging">
<aop:pointcut id = "selectAll"
expression = "execution(* org.mca1.Student.getName(..))"/>
<aop:before pointcut-ref="selectAll" method="beforeAdvice"/>
</aop:aspect>
</aop:config>
```

```
<bean id = "student" class = "org.mca1.Student">
<property name = "name" value = "Zara" />
<property name = "age" value = "11"/>
</bean>
<!-- Definition for logging aspect -->
<bean id = "logging" class = "org.mca1.Logging"/>
</beans>
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.mca1</groupId>
<artifactId>org.mca1</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>AopStudent</name>
<url>http://maven.apache.org</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
<groupId>org.postgresql</groupId>
```

<artifactId>postgresql</artifactId>
<version>42.2.19</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjrt</artifactId>
<version>1.9.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.9.6</version>
</dependency>

</dependencies>
</project>

Program no.2 Write a program to demonstrate Spring AOP – after advice.

Logging.java

```
package org.mca2;
public class Logging {
/**
* This is the method which I would like to execute
* after a selected method execution.
*/
public void afterAdvice(){
System.out.println("Student profile setup complete.");
}
}
```

Student.java

```
package org.mca2;
public class Student {
int age;
String name;
public int getAge() {
return age;
}
public void setAge(int age) {
this.age = age;
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}}
```

StudentTest.java

```
package org.mca2;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.io.*;
public class StudentTest {
private static ApplicationContext context;
public static void main(String[] args) {
```

```
// TODO Auto-generated method stub
context = new ClassPathXmlApplicationContext("appctx.xml");
Student student = (Student) context.getBean("student");
String abc=student.getName();
System.out.println("Name is:"+abc);
int ag=student.getAge();
System.out.println("Age is:"+ag);
}
}
```

In src/main/resources

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"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:context="http://www.springframework.org/schema/context"
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-4.3.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">
<aop:config>
<aop:aspect id = "log" ref = "logging">
<aop:pointcut id = "selectAll"
expression = "execution(* org.mca2.Student.getName(..))"/>
<aop:after pointcut-ref="selectAll" method="afterAdvice"/>
</aop:aspect>
</aop:config>
<bean id = "student" class = "org.mca2.Student">
<property name = "name" value = "Zara" />
<property name = "age" value = "11"/>
</bean>
<!-- Definition for logging aspect -->
<bean id = "logging" class = "org.mca2.Logging"/>
</beans>
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
```

```
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.mca2</groupId>
<artifactId>org.mca2</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>AopStudent</name>
<url>http://maven.apache.org</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>42.2.19</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
<dependency>
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-aop</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjrt</artifactId>
<version>1.9.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.9.6</version>
</dependency>
</dependencies>
</project>
```

Program no. 3 Write a program to demonstrate Spring AOP – after returning advice.

Student.java:

```
package org.mca;
public class Student {
private Integer age;
private String name;
public void setAge(Integer age) {
this.age = age;
}
public Integer getAge() {
System.out.println("Age : " + age );
return age;
}
public void setName(String name) {
this.name = name;
}
public String getName() {
System.out.println("Name : " + name );
return name;
}
public void printThrowException(){
System.out.println("Exception raised");
throw new IllegalArgumentException();
}
```

```
}
```

Logging.java:

```
package org.mca;
public class Logging {
public void afterReturningAdvice(Object retVal){
System.out.println("Returning:" + retVal.toString() );
}
}
```

Beans.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"
xmlns:aop = "http://www.springframework.org/schema/aop"
xsi:schemaLocation = "http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">
<aop:config>
<aop:aspect id = "log" ref = "logging">
<aop:pointcut id = "selectAll"
expression = "execution(* org.mca.*.*(..))"/>
<aop:after-returning pointcut-ref = "selectAll"
method = "afterReturningAdvice" returning = "retVal"/>
</aop:aspect>
</aop:config>
<!-- Definition for student bean -->
<bean id = "student" class = " org.mca.Student">
<property name = "name" value = "Zara" />
<property name = "age" value = "11"/>
</bean>
<!-- Definition for logging aspect -->
<bean id = "logging" class = " org.mca.Logging"/>
</beans>
```

MainApp.java:

```
package org.mca;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

```
// TODO Auto-generated method stub
ApplicationContext context = new
ClassPathXmlApplicationContext("Beans.xml");
Student student = (Student) context.getBean("student");
student.getName();
student.getAge();
}
}
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.tutorialspoint</groupId>
<artifactId>AfterReturningAdvise</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>AopStudent</name>
<url>http://maven.apache.org</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
```

```
<dependency>
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>42.2.19</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjrt</artifactId>
<version>1.9.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.9.6</version>
</dependency>
</dependencies>
</project>
```

Program no 4 Write a program to demonstrate Spring AOP –around advice.

Student.java

```
package org.viva2;
public class Student {
private Integer age;
private String name;
public void setAge(Integer age) {
this.age = age;
}
```

```
public Integer getAge() {
System.out.println("Age : " + age );
return age;
}
public void setName(String name) {
this.name = name;
}
public String getName() {
System.out.println("Name : " + name );
return name;
}
public void printThrowException(){
System.out.println("Exception raised");
throw new IllegalArgumentException();
}
}
```

Logging.java

```
package org.viva2;
import org.aspectj.lang.ProceedingJoinPoint;
Name : Hetal Rajbhar Roll no: 36
public class Logging {
/**
* This is the method which I would like to execute
* around a selected method execution.
*/
public String aroundAdvice(ProceedingJoinPoint jp) throws Throwable{
System.out.println("Around advice");
Object[] args = jp.getArgs();
if(args.length>0){
System.out.print("Arguments passed: " );
for (int i = 0; i < args.length; i++) {
System.out.print("arg "+(i+1)+": "+args[i]);
}
}
Object result = jp.proceed(args);
System.out.println("Returning " + result);
return result.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"
xmlns:aop = "http://www.springframework.org/schema/aop"
xsi:schemaLocation = "http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">
<aop:config>
<aop:aspect id = "log" ref = "logging">
<aop:pointcut id = "selectName"
expression = "execution(* org.viva2.Student.getName(..))"/>
<aop:around pointcut-ref = "selectName" method = "aroundAdvice"/>
</aop:aspect>
</aop:config>
<!-- Definition for student bean -->
<bean id = "student" class = "org.viva2.Student">
<property name = "name" value = "Sanjana" />
<property name = "age" value = "21"/>
</bean>
<!-- Definition for logging aspect -->
<bean id = "logging" class = "org.viva2.Logging"/>
</beans>
```

StudentTest.java

```
package org.viva2;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class StudentTest {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext context = new
ClassPathXmlApplicationContext("Appctx.xml");
Student student = (Student) context.getBean("student");
student.getName();
}}
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.viva2</groupId>
<artifactId>org.viva2</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

```
<packaging>jar</packaging>
<name>AopStudent</name>
<url>http://maven.apache.org</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>42.2.19</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
<dependency>
```

```
<groupId>org.aspectj</groupId>
<artifactId>aspectjrt</artifactId>
<version>1.9.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.9.6</version>
</dependency>

</dependencies>
</project>
```

Program no. 5 Write a program to demonstrate Spring AOP –After Throwing advice.

Student.java:

```
package org.viva4;
public class Student {
private Integer age;
private String name;
public Integer getAge() {
return age;
}
public void setAge(Integer age) {
this.age = age;
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public void printThrowException(){
System.out.println("Exception raised");
throw new IllegalArgumentException();
}
}
```

Logging.java:

```
package org.viva4;
```

```
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterThrowing;
@Aspect
public void afterThrowingAdvice(JoinPoint jp, Throwable error){
System.out.println("Method Signature: " + jp.getSignature());
System.out.println("Exception: "+error);
}
}
```

Beans.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"
xmlns:aop = "http://www.springframework.org/schema/aop"
xsi:schemaLocation = "http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">
<aop:aspectj-autoproxy/>
<!-- Definition for student bean -->
<bean id = "student" class = " org.viva4.Student">
<property name = "name" value = "Zara" />
<property name = "age" value = "11"/>
</bean>
<!-- Definition for logging aspect -->
<bean id = "logging" class = " org.viva4.Logging"/>
</beans>
```

MainApp.java:

```
package org.viva4;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {

public static void main(String[] args) {
ApplicationContext context = new
ClassPathXmlApplicationContext("Beans.xml");

Student student = (Student) context.getBean("student");
student.printThrowException();
}
}
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.tutorialspoint</groupId>
<artifactId>AfterThrowing</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>AopStudent</name>
<url>http://maven.apache.org</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>42.2.19</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>5.2.8.RELEASE</version>
```

```
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
<version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjrt</artifactId>
<version>1.9.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.9.6</version>
</dependency>

</dependencies>
</project>
```

Output:

Program 6 Write a program to demonstrate Spring AOP – pointcuts.

MyService.java

```
package com.example.aopdemo;
public class MyService {
public void performTask() {
System.out.println("Executing the task in MyService.");
}
}
```

LoggingAspect.java

```
package com.example.aopdemo;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
@Aspect
public class LoggingAspect {
// Define a pointcut expression
@Before("execution(* com.example.aopdemo.MyService.performTask(..))")
public void logBefore() {
System.out.println("LoggingAspect: Before executing performTask.");
```

```
}
}
```

applicationContext.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"
xmlns:aop="http://www.springframework.org/schema/aop"
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">
<!-- Enable AspectJ auto proxy -->
<aop:aspectj-autoproxy/>
<!-- Register beans -->
<bean id="myService" class="com.example.aopdemo.MyService"/>
<bean id="loggingAspect" class="com.example.aopdemo.LoggingAspect"/>
</beans>
MainApp.java
package com.example.aopdemo;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {
public static void main(String[] args) {
ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
// Retrieve the service bean
MyService myService = context.getBean("myService", MyService.class);
// Call the method to see AOP in action
myService.performTask();
// Close the context
((ClassPathXmlApplicationContext) context).close();
}
}
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>SpringAOPDemo</artifactId>
```

```
<version>1.0-SNAPSHOT</version>
<dependencies>
<!-- Spring Context for AOP -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.3.30</version>
</dependency>
<!-- Spring AOP -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
<version>5.3.30</version>
</dependency>
<!-- AspectJ for annotations -->
<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.9.19</version>
</dependency>
</dependencies>
</project>
```

Practical No 9

Programs Based on Spring JDBC

Steps: file-> new-> maven project-> create simple project->group id (com.example) and artifact id(Spring-jdbc-demo) -> finish

Click on src/main/java-> new->package->org.viva-> finish.
Righ click on package->new->class-> create all classes-> finish.

Click on src/main/resources-> spring configuration file->create (appctx.xml)file -> finish.
Add the jar files -> right click project > build path -> configure build path-> libraries-> add external jars -> (spring framework release and posgresql)

Program no. 1 Write a program to demonstrate Spring JdbcTemplate class to store data in
database table.
Program:

On pgAdmin4
```
create table mymovies(mid int,title varchar,actor varchar);
select * from mymovies;
```

Movies.java
```
package org.viva;
public class Movies {
int mid;
String title;
String actor;
public Movies(int mid, String title, String actor) {
super();
this.mid = mid;
this.title = title;
this.actor = actor;
}
/**
* @return the mid
*/
public int getMid() {
return mid;
}
```

```
/**
* @param mid the mid to set
*/
public void setMid(int mid) {
this.mid = mid;
}
/**
* @return the title
*/
public String getTitle() {
return title;
}
/**
* @param title the title to set
*/
public void setTitle(String title) {
this.title = title;
}
/**
* @return the actor
*/
public String getActor() {
return actor;
}
/**
* @param actor the actor to set
*/
public void setActor(String actor) {
this.actor = actor;
} }
```

MoviesDao.java

```
package org.viva;
import org.springframework.jdbc.core.JdbcTemplate;
public class MoviesDao {
private JdbcTemplate jdbcTemplate;
// Getter method for jdbcTemplate
public JdbcTemplate getJdbcTemplate() {
return jdbcTemplate;
}
// Setter method for jdbcTemplate
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
this.jdbcTemplate = jdbcTemplate;
```

```
}
// Method to insert movie record into the database
public int insMovie(Movies m1) {
// Use parameterized SQL to avoid SQL injection
String insSql = "INSERT INTO mymovies (mid, title, actor) VALUES (?, ?, ?)";
// Use jdbcTemplate's update method with parameters to safely insert data
return jdbcTemplate.update(insSql, m1.getMid(), m1.getTitle(), m1.getActor());
}
}
```

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="abc" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="moviebean" class="org.viva.MovieDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

MovieTest.java

```
package org.viva;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MovieTest {

public static void main(String[] args) {
// TODO Auto-generated method stub
appCon = new ClassPathXmlApplicationContext("appctx.xml");
MovieDao m1=(MovieDao)appCon.getBean("moviebean");
Movies t1=new Movies(1,"A Beautiful Mind","Russel Crow");
System.out.println(m1.insMovie(t1));
```

```
}
}
```

Program no. 2 : Write a program to demonstrate Spring JdbcTemplate class to store data in database table Employee and also demonstrate update and delete.
Program:

On pgAdmin

```
create table employee(id int,name varchar, salary int );
select * from employee;
```

Employee.java

```
package org.viva;
public class Employee {
private int id;
private String name;
private int salary;
public Employee() {
super();
// TODO Auto-generated constructor stub
}
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 getSalary() {
return salary;
}
public void setSalary(int salary) {
this.salary = salary;
}
public Employee(int id, String name, int salary) {
```

```
super();
this.id = id;
this.name = name;
this.salary = salary;
}
}
```

EmployeeDao.java

```
package org.viva;
import org.springframework.jdbc.core.JdbcTemplate;
public class EmployeeDao {
private JdbcTemplate jdbcTemplate;
public EmployeeDao() {
super();
// TODO Auto-generated constructor stub
}
public EmployeeDao(JdbcTemplate jdbcTemplate) {
super();
this.jdbcTemplate = jdbcTemplate;
}
public JdbcTemplate getJdbcTemplate() {
return jdbcTemplate;
}
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
this.jdbcTemplate = jdbcTemplate;
}
public int saveEmployee(Employee e){
String query="insert into Employee

values('"+e.getId()+"','"+e.getName()+"','"+e.getSalary()+"')";
return jdbcTemplate.update(query); }

public int updateEmployee(Employee e){
String query="update employee set

name='"+e.getName()+"',salary='"+e.getSalary()+"' where id='"+e.getId()+"' ";

return jdbcTemplate.update(query);
}
public int deleteEmployee(Employee e){
String query="delete from employee where id='"+e.getId()+"' ";
return jdbcTemplate.update(query);
}
```

```
}
```

appctx1.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="abc" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="edao" class="org.viva.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

EmployeeTest.java

```
package org.viva;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class EmployeeTest {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext ctx=new ClassPathXmlApplicationContext("appctx1.xml");
EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
int status=dao.saveEmployee(new Employee(102,"Amit",350));
System.out.println(status);
/*int status=dao.updateEmployee(new

Employee(102,"Sonoo",15000));

System.out.println(status);
*/
/*Employee e=new Employee();
e.setId(102);
int status=dao.deleteEmployee(e);
System.out.println(status);*/
```

```
}
}
```

Program no. 3 Write a program to demonstrate RowMapper interface to fetch the records from the database.

Program:
On pgAdmin
create table emp1(id int,name varchar);
select *from emp1;

Employee.java

```
package com.viva;
public class Employee {
int id;
String name;
public Employee() {
super();
}
public Employee(int id, String name) {
super();
this.id = id;
this.name = name;
}
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;
}
}
```

EmployeeDao.java

```
package com.viva;
import org.springframework.jdbc.core.*;
import java.util.*;
public class EmployeeDao {
JdbcTemplate jdbcTemplate;
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
this.jdbcTemplate = jdbcTemplate;
}
public int saveEmp(Employee e){
String query="insert into emp1 values("+e.getId()+",'"+e.getName()+"')";
return jdbcTemplate.update(query);
}
public List<Employee> findAll() {
String sql = "SELECT * FROM emp1";
List<Employee> obj = jdbcTemplate.query(sql,new EmpRowMapper());
return obj;
}
public int saveEmp1(Employee e1) {
// TODO Auto-generated method stub
return 0; }}
```

EmployeeTest.java

```
package com.viva;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.util.*;
public class EmployeeTest {
private static ApplicationContext appCon;
public static void main(String[] args) {
// TODO Auto-generated method stub
appCon = new ClassPathXmlApplicationContext("appctx.xml");
EmployeeDao fac=(EmployeeDao)appCon.getBean("Emp1");
Employee e1=new Employee(2,"Sonia");
System.out.println(fac.saveEmp(e1));
List<Employee> lstemp=fac.findAll();
for(Employee e2:lstemp)
{
System.out.print(e2.getId());
System.out.println(e2.getName());}}}
```

EmpRowMapper.java

```
package com.viva;
```

```
import org.springframework.jdbc.core.RowMapper;
import java.sql.ResultSet;
import java.sql.SQLException;
public class EmpRowMapper implements RowMapper<Employee>
{
@Override
public Employee mapRow(ResultSet arg0, int arg1) throws SQLException
{
Employee e1=new Employee();
e1.setId(arg0.getInt(1));
e1.setName(arg0.getString(2));
return e1;}}
```

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="abc" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="Emp1" class="com.viva.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

Program no. 4 Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface

On pgAdmin

```
create table employee23(id int,name varchar, salary int );
select * from employee23;
insert into employee23 values(102,'sonia');
insert into employee23 values(102,'sonu');
select * from employee23;
```

Employee.java

```
package org.viva23;
public class Employee {
private int id;
private String name;
private int salary;
public Employee() {
super();
// TODO Auto-generated constructor stub
}
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 getSalary() {
```

```
return salary;
}
public void setSalary(int salary) {
this.salary = salary;
}
public Employee(int id, String name, int salary) {
super();
this.id = id;
this.name = name;
this.salary = salary;
}
@Override
public String toString() {
return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";
}}
```

EmployeeDao.java

```
package org.viva23;
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 EmployeeDao {
private JdbcTemplate jdbcTemplate;
public EmployeeDao() {
super();
// TODO Auto-generated constructor stub
}
public EmployeeDao(JdbcTemplate jdbcTemplate) {
super();
this.jdbcTemplate = jdbcTemplate;
}
public JdbcTemplate getJdbcTemplate() {
return jdbcTemplate;
}
```

```
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
this.jdbcTemplate = jdbcTemplate;
}
public List<Employee> getAllEmployees(){
return jdbcTemplate.query("select * from employee23",new

ResultSetExtractor<List<Employee>>(){

@Override
public List<Employee> extractData(ResultSet rs) throws SQLException,
DataAccessException {
List<Employee> list=new ArrayList<Employee>();
while(rs.next()){
Employee e=new Employee();
e.setId(rs.getInt(1));
e.setName(rs.getString(2));
e.setSalary(rs.getInt(3));
list.add(e);
}
return list; } }); } }
```

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="abc" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="edao" class="org.viva23.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

EmployeeTest.java

```
package org.viva23;
```

```
import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class EmployeeTest {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext ctx=new ClassPathXmlApplicationContext("appctx.xml");
EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
List<Employee> list=dao.getAllEmployees();
for(Employee e:list)
System.out.println(e);
}}
```

Practical No: 10

Steps ;

1. Go to start.spring.io

Step 2: (CHECK maven and java as project and jar as projectmetadata)

Step 3: then add the dependencies:(Spring Web and Spring Data JPA)

Step 4: then click on generate . after downloading the demo6 zip file -> then extract

Step 5: go to eclipse -> file -> new-> import -> maven-> existing maven project -> browse -> demo6 file.

Step 6: then click on file-> new-> class (1. Person 2. PersonController)

Step file->new->interface(code)

Step7 : click on src/main/resources -> application properties-> code

Step 8: click on pom.xml modify code.
Then right click on default application.java file ->run as-> java application
Step 9: Go to PgADMIN - > Run as Administrator -> postgresql 15-> databases ->postgres-> Schemas-> tables -> check Person table is available.
Step 10 : then click on table person-> add code -> run on server http://localhost:8081/persons

Program no. 1 Write a program to create a simple Spring Boot application that prints a message.

```
package org.mca.Spring_Boot_Demo;
/**
* Hello world!
*
*/
public class App
{
public static void main( String[] args )
{
System.out.println( "Hello World!" );
}
}
```

Program no. 2 Write a program to demonstrate RESTful Web Services with spring boot.

WelcomeController.java

```
package com.springboot.app;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class WelcomeController {
@GetMapping("/welcome")
public String welcome()
```

```
{
return "welcome to springboot dear";
}
}
```

SpringBootApp.java

```
package com.springboot.app;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringBootApp {
public static void main(String[] args) {
SpringApplication.run(SpringBootApp.class, args);
}
}
```

Navigate to http://localhost:8080/welcome

Program no. 3 Write a program to demonstrate Database Connection with spring boot.

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>3.4.0</version>
<relativePath/> <!-- lookup parent from repository -->
```

```
</parent>
<groupId>com.example</groupId>
<artifactId>demo_6</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>demo_6</name>
<description>Demo project for Spring Boot</description>
<url/>
<licenses>
<license/>
</licenses>
<developers>
<developer/>
</developers>
<scm>
<connection/>
<developerConnection/>
<tag/>
<url/>
</scm>
<properties>
<java.version>17</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>42.6.0</version> <!-- Use the latest version -->
</dependency>
</dependencies>
<build>
<plugins>
```

```
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
</project>
```

In src/main/resources

Application.properties

```
spring.application.name=demo_6
# PostgreSQL Database Configuration
spring.datasource.url=jdbc:postgresql://localhost:5433/postgres
spring.datasource.username=postgres
spring.datasource.password=abc
server.port=8081
spring.datasource.driver-class-name=org.postgresql.Driver
# JPA Configuration
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Person.java

```
package com.example.demo_6;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class Person {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
private String name;
// Getters and Setters
public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}
public String getName() {
return name;
```

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

Interface PersonRepository

```
package com.example.demo_6;
import org.springframework.data.jpa.repository.JpaRepository;
public interface PersonRepository extends JpaRepository<Person, Long> {
}
```

PersonController.java

```
package com.example.demo_6;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.List;
@RestController
public class PersonController {
@Autowired
private PersonRepository personRepository;
@GetMapping("/persons")
public List<Person> getAllPersons() {
return personRepository.findAll();
}
}
```

Person query:

Go back to PgADMIN and execute following Queries

```
INSERT INTO person (name) VALUES ('sonia');
INSERT INTO person (name) VALUES ('jasmine');
INSERT INTO person (name) VALUES ('John Doe');
INSERT INTO person (name) VALUES ('Jane Smith');
```