Practical No 9

Programs Based on Spring JDBC

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());
    }

}
```

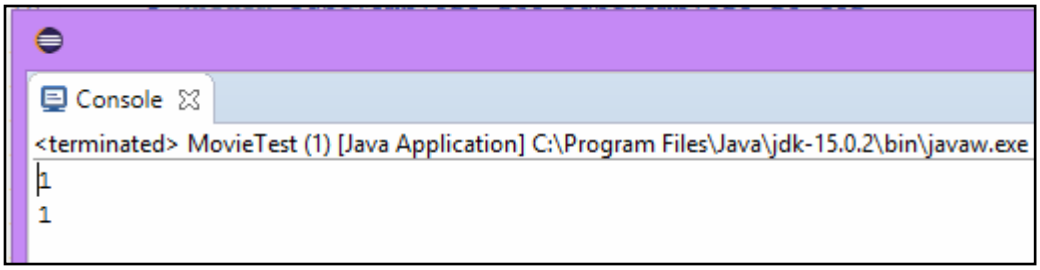
Appctx.xml

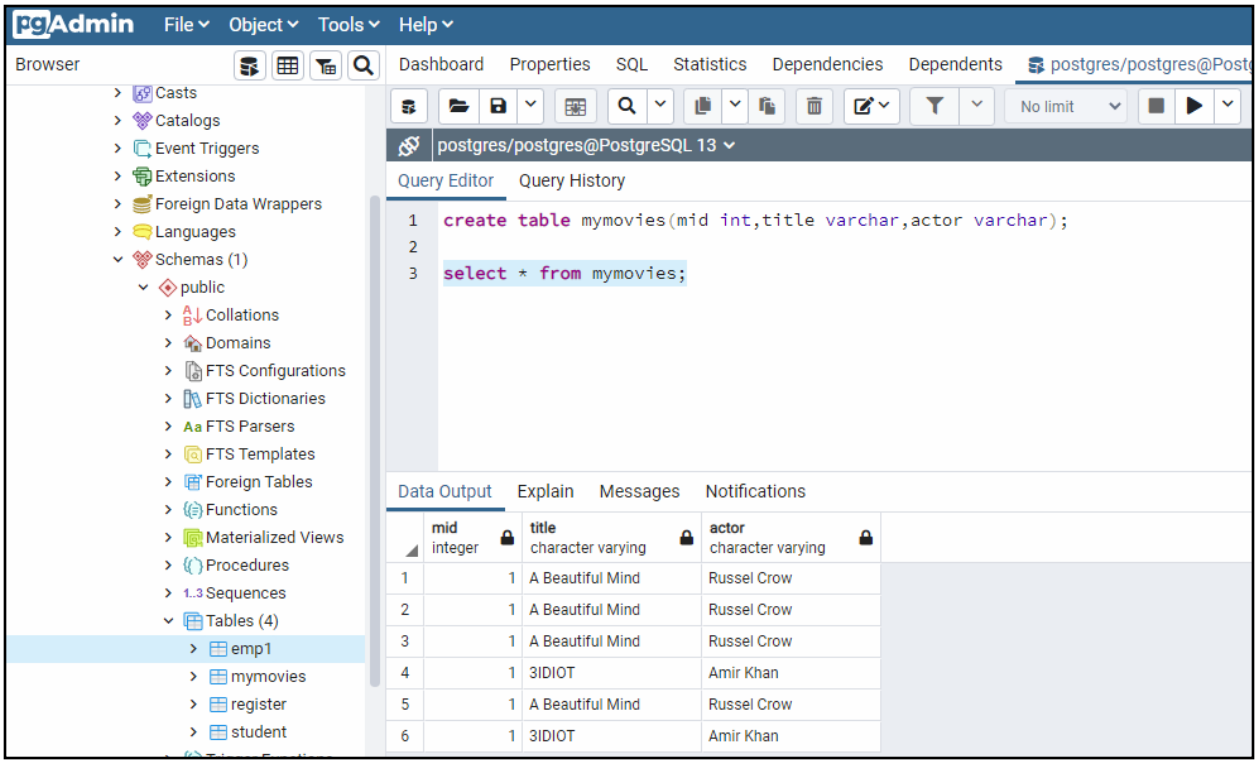
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">




<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

```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));


      }
}
```

Output:


Console
<terminated> MovieTest (1) [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe
1
1


pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
postgres/postgres@Post
No limit
postgres/postgres@PostgreSQL 13
Query Editor
Query History
1 create table mymovies(mid int,title varchar,actor varchar);
2
3 select * from mymovies;
Data Output
Explain
Messages
Notifications
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas (1)
public
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (4)
emp1
mymovies
register
student
mid integer
title character varying
actor character varying
1 1 A Beautiful Mind Russel Crow
2 1 A Beautiful Mind Russel Crow
3 1 A Beautiful Mind Russel Crow
4 1 3IDIOT Amir Khan
5 1 A Beautiful Mind Russel Crow
6 1 3IDIOT Amir Khan

Program no. 2Write 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());
		}
	}
}
```

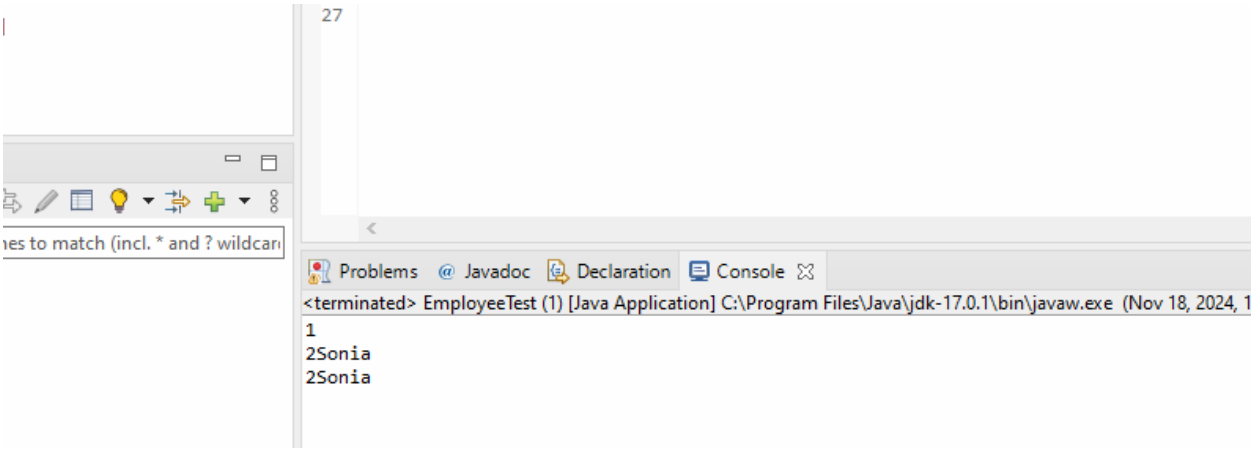
EmpRowMapper.java

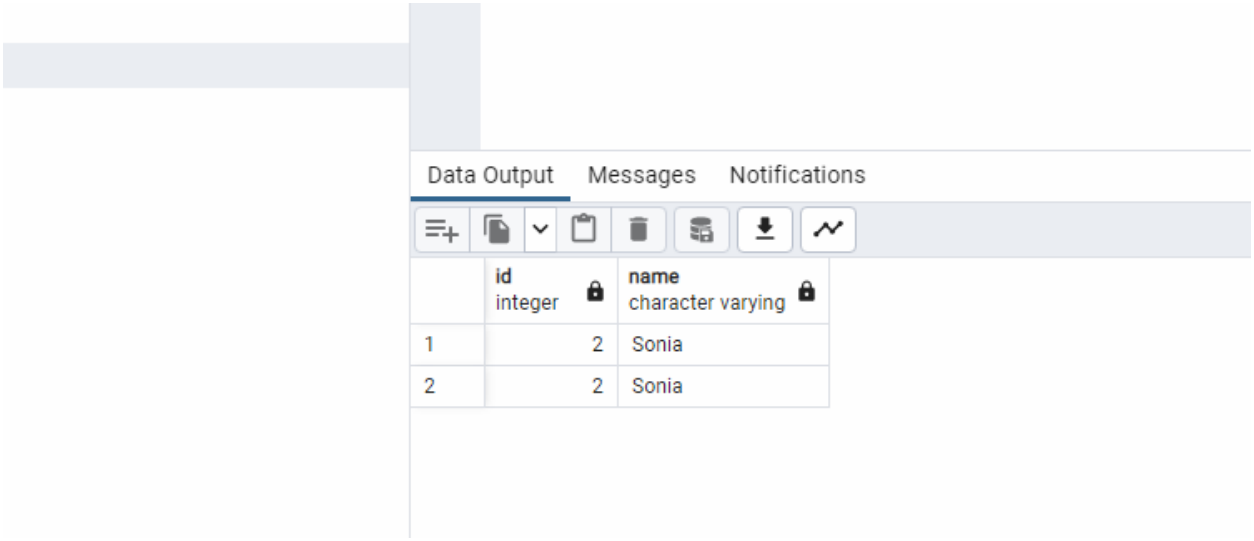
```
package com.viva;

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>
```

Output:


27
es to match (incl. * and ? wildcar
Problems
Javadoc
Declaration
Console
<terminated> EmployeeTest (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Nov 18, 2024, 1
1
2Sonia
2Sonia


Data Output
Messages
Notifications
id
integer
name
character varying
1
2
Sonia
2
2
Sonia

Program no. 4Write 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);


	}

}
```

Query Query History

```
1
2
3  insert into employee23 values(102,'sonu');
4  select * from employee23;
```

Data Output Messages Notifications

	id integer	name character varying	salary integer
1	101	sonia	[null]
2	101	sonia	[null]
3	102	sonu	[null]

avaSE-15]

set names to match (incl. * and ? wildcar

Problems Javadoc Declaration Console

<terminated> EmployeeTest (2) [Java Application] C:\Program Files\Java\jd

```
Employee [id=101, name=sonia, salary=0]
Employee [id=101, name=sonia, salary=0]
Employee [id=102, name=sonu, salary=0]
```