Spring JDBC

1. Write a program to insert, update and delete records from the given table.

2. Write a program to demonstrate PreparedStatement in Spring JdbcTemplate.

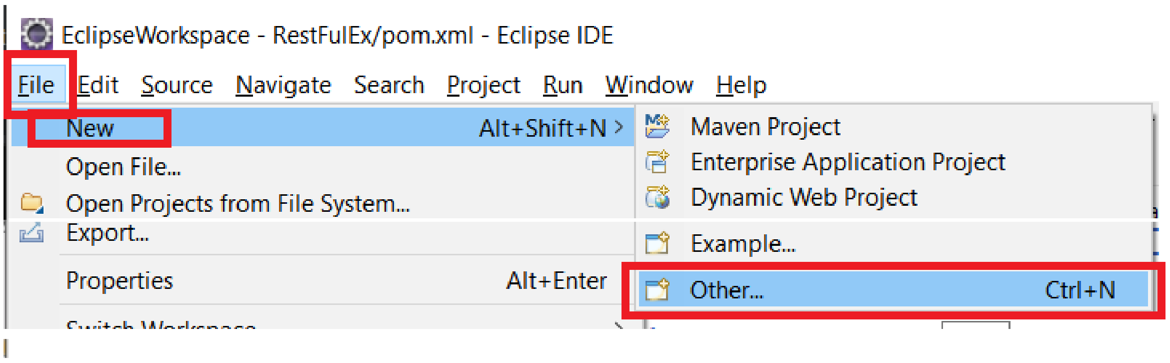
3. Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface.

4. Write a program to demonstrate RowMapper interface to fetch the records from the database.

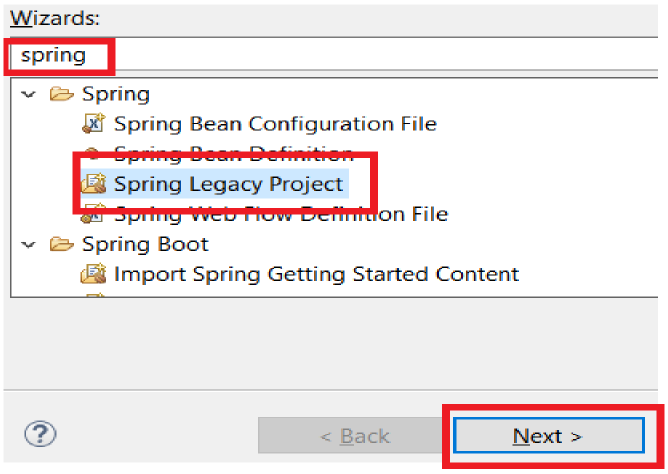
Steps to Create Spring Legacy Project

Step 1 : Creating Spring Legacy Project.

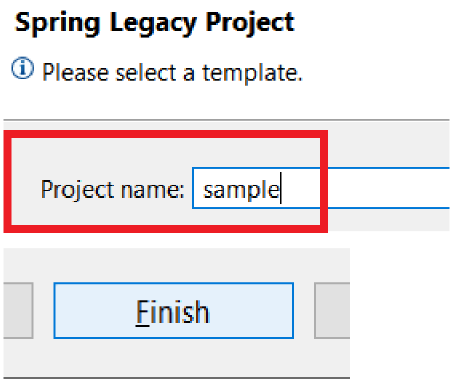
1.1 : Open Eclipse. Go To File > New > Other.


EclipseWorkspace - RestFulEx/pom.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Export...
Properties Alt+Enter
Maven Project
Enterprise Application Project
Dynamic Web Project
Example...
Other... Ctrl+N

1.2 : Search for ‘spring’ and Select ‘Spring Legacy Project’. Then Click on Next.

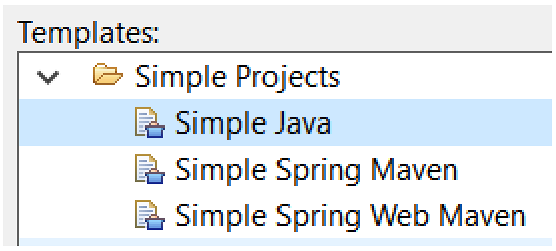

Wizards:
spring
Spring
Spring Bean Configuration File
Spring Bean Definition
Spring Legacy Project
Spring Web Flow Definition File
Spring Boot
Import Spring Getting Started Content
< Back
Next >

1.3 : ChooseProject Name of your wish, below there select **Simple Java**& simply Finish.

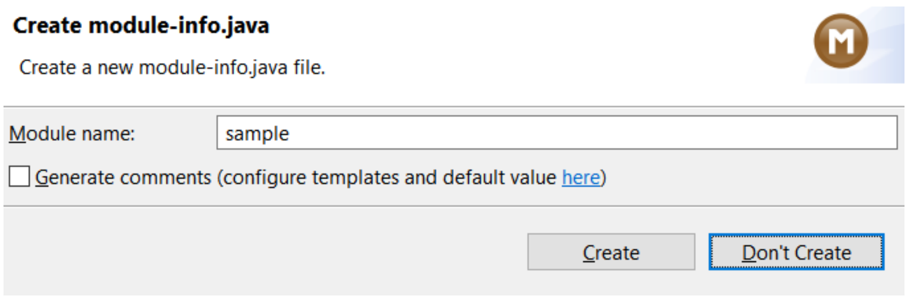

Spring Legacy Project

Please select a template.

Project name: sample


Templates:

1.4 : If asked for Creating module-info.java file, click on **Don't Create**.


Create module-info.java

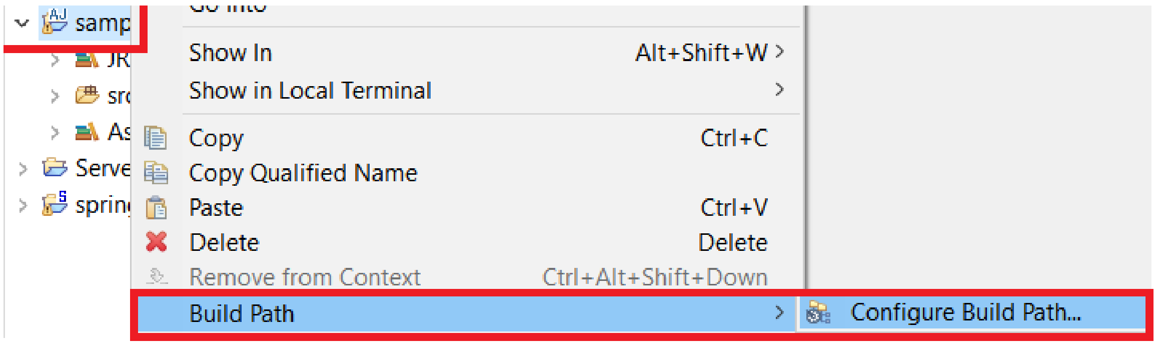
Create a new module-info.java file.

Module name: sample

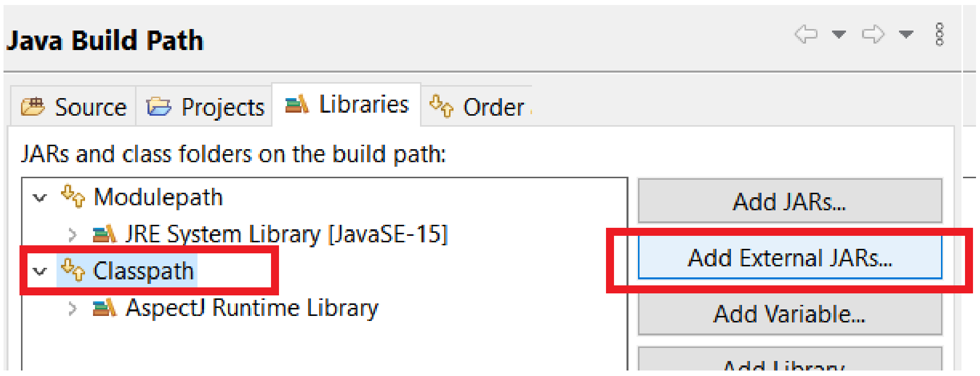
Generate comments (configure templates and default value here)

Step 2 : Adding the Spring Libraries.

2.1 : Right click on your Newly created Spring Legacy project, Choose Build Path > Configure Build Path.

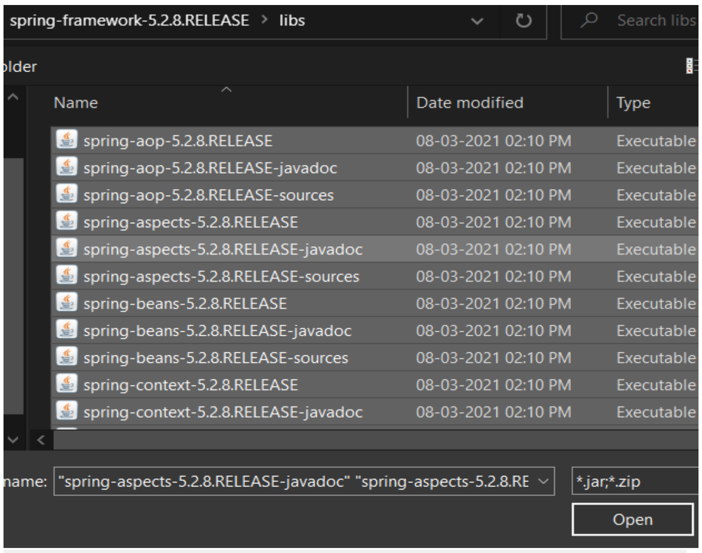


2.2 On Java Build Path wizard, Choose **Classpath** and then select **Add External JARs.**


Java Build Path

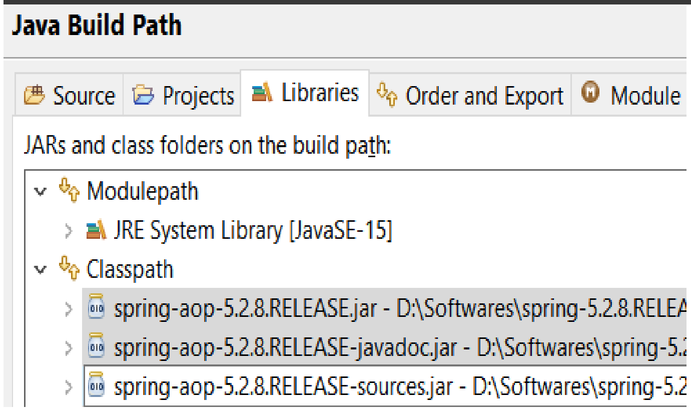
JARs and class folders on the build path:

2.3 : Choose all the Spring Libraries you've downloaded, and click on OPEN. This will add all libraries to Classpath.



Name	Date modified	Type
spring-aop-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-aop-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable
spring-aop-5.2.8.RELEASE-sources	08-03-2021 02:10 PM	Executable
spring-aspects-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-aspects-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable
spring-aspects-5.2.8.RELEASE-sources	08-03-2021 02:10 PM	Executable
spring-beans-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-beans-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable
spring-beans-5.2.8.RELEASE-sources	08-03-2021 02:10 PM	Executable
spring-context-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-context-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable

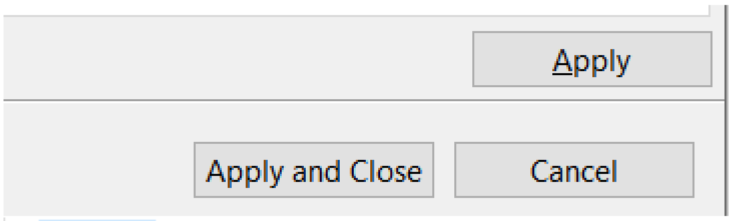
name: "spring-aspects-5.2.8.RELEASE-javadoc" "spring-aspects-5.2.8.RE
.jar;.zip

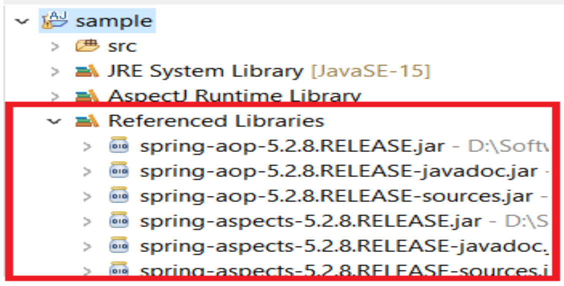


JARs and class folders on the build path:

```
Modulepath
  JRE System Library [JavaSE-15]
Classpath
  spring-aop-5.2.8.RELEASE.jar - D:\Softwares\spring-5.2.8.RELEA
  spring-aop-5.2.8.RELEASE-javadoc.jar - D:\Softwares\spring-5.2
  spring-aop-5.2.8.RELEASE-sources.jar - D:\Softwares\spring-5.2
```

2.4 Finally click on Apply & Close, now you are ready to work with Spring Legacy Project.





```
sample
  src
  JRE System Library [JavaSE-15]
  AspectJ Runtime Library
  Referenced Libraries
    spring-aop-5.2.8.RELEASE.jar - D:\Softw
    spring-aop-5.2.8.RELEASE-javadoc.jar -
    spring-aop-5.2.8.RELEASE-sources.jar -
    spring-aspects-5.2.8.RELEASE.jar - D:\S
    spring-aspects-5.2.8.RELEASE-javadoc.
    spring-aspects-5.2.8.RELEASE-sources.j
```

Problem Statement 1 : Write a program to insert, update and delete records from the given table.

Solution :

Movie1.java

```
package org.me;

publicclass Movie1 {

	int mid;
	String title;
	String actor;
	public Movie1(int mid, String title, String actor) {
		super();
		this.mid = mid;
		this.title = title;
		this.actor = actor;
	}
	public Movie1() {
		super();
		// TODO Auto-generated constructor stub
	}
	publicint getMid() {
		return mid;
	}
	publicvoid setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
		return title;
	}
	publicvoid setTitle(String title) {
		this.title = title;
	}
	public String getActor() {
		return actor;
	}
	publicvoid setActor(String actor) {
		this.actor = actor;
	}


}
```

MovieDAO.java

```
package org.me;

import org.springframework.jdbc.core.*;
publicclass MovieDAO {
```

```
JdbcTemplate jdbcTemplate;

publicvoid setJdbcTemplate(JdbcTemplate jdbcTemplate) {
	this.jdbcTemplate = jdbcTemplate;
}
publicint insMovie(Movie1 m1)
{
	String insSql="insert into
mymovies1	values("+m1.getMid()+",'"+m1.getTitle()+"','"+m1.getActor()+"')";

return jdbcTemplate.update(insSql);
}

publicint updateMovie(Movie1 m1){
   String query="update mymovies1 set title='"+m1.getTitle()+"',actor='"+m1.getActor()+"'
where mid='"+m1.getMid()+"' ";
return jdbcTemplate.update(query);
}

publicint deleteMovie(Movie1 m1){
   String query="delete from mymovies1 where mid='"+m1.getMid()+"' ";
return jdbcTemplate.update(query);
}
}
```

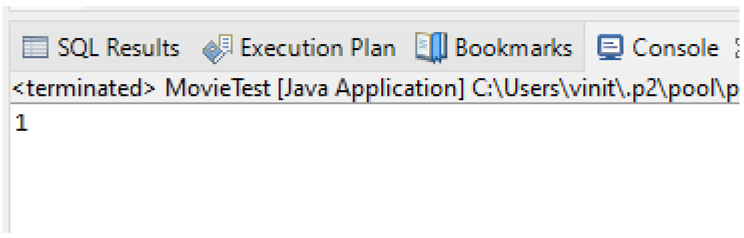
appctx.xml

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

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName"value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="admin" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="mymovie" class="org.me.MovieDAO">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean></beans>
```

MovieTest.java

```
package org.me;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;


public class MovieTest {
        private static ApplicationContext appCon;


        public static void main(String[] args) {
                // TODO Auto-generated method stub

                appCon = new ClassPathXmlApplicationContext("appctx.xml");
                MovieDAO m1=(MovieDAO)appCon.getBean("mymovie");

                //insert query
                Movie1 t1=new Movie1(10,"Mirzapur","P");
                System.out.println(m1.insMovie(t1));

                //update query

                //int status=m1.updateMovie(new Movie1(10,"war","hritik"));
        // System.out.println(status);


                //delete

                // Movie1 t2=new Movie1();
          //t2.setMid(5);
          //int status=m1.deleteMovie(t2);
         // System.out.println(status);


        }

}
```

Output :



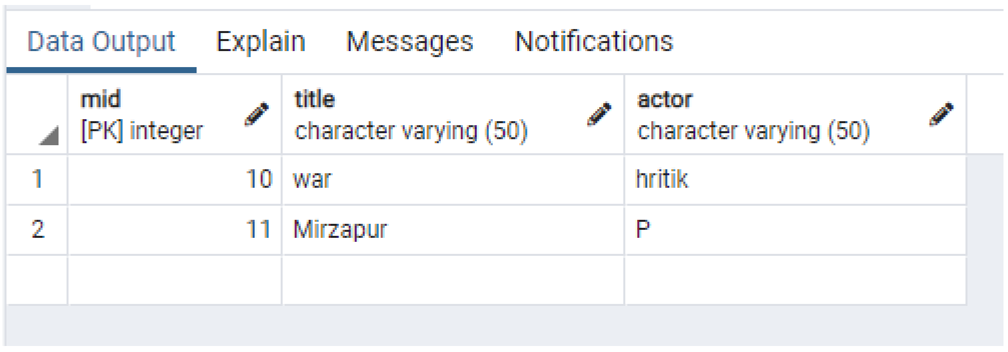
```
<terminated> MovieTest [Java Application] C:\Users\vinit\.p2\pool\p
1
```

Database :

CREATE TABLE mymovies1

```
(
mid int,
title varchar(50),
actor varchar(50),
PRIMARY KEY (mid)
);
```

Final Table After Execution :


Data Output | Explain | Messages | Notifications

	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P

Problem Statement 2 : Write a program to demonstrate PreparedStatement in Spring JdbcTemplate.

Solution :

Movie1.java

```
package org.me;

public class Movie1 {

	int mid;
	String title;
	String actor;
	public Movie1(int mid, String title, String actor) {
		super();
		this.mid = mid;
		this.title = title;
		this.actor = actor;
	}
	public Movie1() {
		super();
	}
	public int getMid() {
		return mid;
	}
	public void setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getActor() {
        return actor;
    }
    public void setActor(String actor) {
        this.actor = actor;
    }

}
```

MovieDAO1.java

```
package org.me;

import java.sql.PreparedStatement;

import java.sql.SQLException;


import org.springframework.dao.DataAccessException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.PreparedStatementCallback;

public class MovieDAO1 {

    JdbcTemplate jdbcTemplate;


    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {

        this.jdbcTemplate = jdbcTemplate;

    }

    public Boolean saveMovieByPreparedStatement(final Movie1 e){

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

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

       @Override

       public Boolean doInPreparedStatement(PreparedStatement ps)

             throws SQLException, DataAccessException {

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

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

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

          return ps.execute();
```

```
            }
            });
        }
}
```

appctx1.java

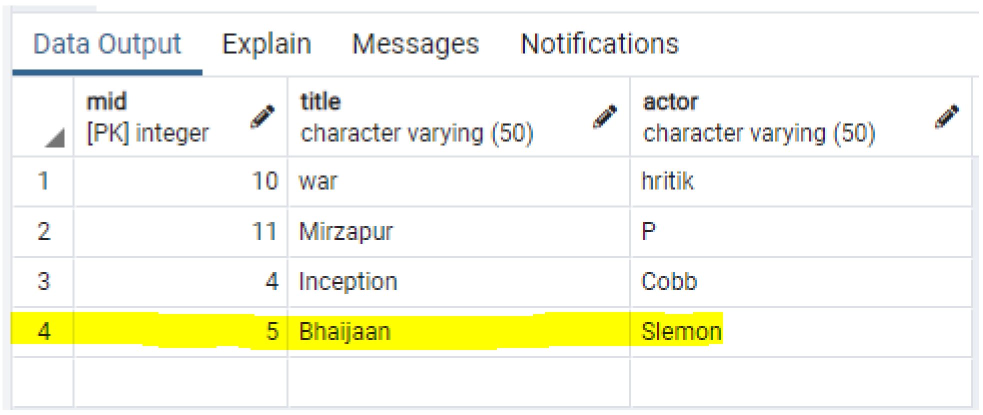
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<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="pass" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="mymovie" class="org.me.MovieDAO1">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

MovieTest1.java

```
package org.me;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest1 {


        private static ApplicationContext appCon;

        public static void main(String[] args) {

                // TODO Auto-generated method stub

                appCon = new ClassPathXmlApplicationContext("appctx1.xml");

                MovieDAO1 m1=(MovieDAO1)appCon.getBean("mymovie");

           m1.saveMovieByPreparedStatement(new Movie1(5,"Bhaijaan","Slemon"));
```

```
	}
}
```

Output :



	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P
3	4	Inception	Cobb
4	5	Bhaijaan	Slemon

Problem Statement 3 : Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface.

Solution :

Movie2.java

```
package org.me;

public class Movie2 {

	int mid;
	String title;
	String actor;
	public int getMid() {
		return mid;
	}
	public void setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getActor() {
		return actor;
	}
	public void setActor(String actor) {
		this.actor = actor;
	}
	public String toString(){
	   return mid+" "+title+" "+actor;
}
}
```

MovieDAO2.java

```
package org.me;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.ResultSetExtractor;
public class MovieDAO2 {
	JdbcTemplate jdbcTemplate;

	public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
		this.jdbcTemplate = jdbcTemplate;
	}
	public List<Movie2> getAllMovie(){
		 return jdbcTemplate.query("select * from mymovies1",new
ResultSetExtractor<List<Movie2>>(){
			  @Override
			   public List<Movie2> extractData(ResultSet rs) throws SQLException,
			        DataAccessException {


			     List<Movie2> list=new ArrayList<Movie2>();
			     while(rs.next()){
			        Movie2 e=new Movie2();
			        e.setMid(rs.getInt(1));
			        e.setTitle(rs.getString(2));
			        e.setActor(rs.getString(3));
			        list.add(e);
			     }
```

```
                    return list;
                    }
                  });
                }
}
```

appctx2.java

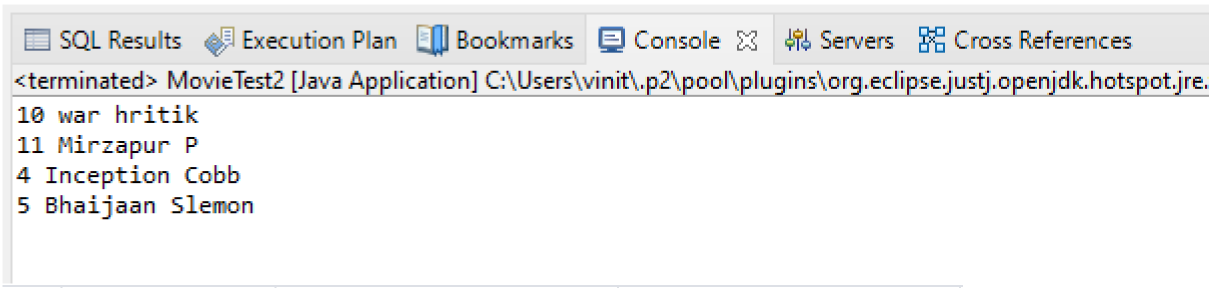
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="password" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="mymovie" class="org.me.MovieDAO2">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

MovieTest2.java

```
package org.me;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest2 {

        private static ApplicationContext appCon;

        public static void main(String[] args) {

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

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

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

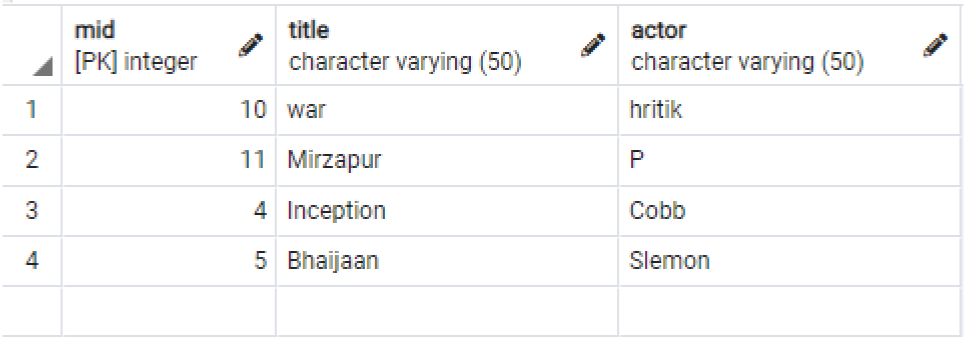
```
              for(Movie2 e:list)
                  System.out.println(e);
            }
}
```

Output :



SQL Results   Execution Plan   Bookmarks   Console   Servers   Cross References

```
<terminated> MovieTest2 [Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.
10 war hritik
11 Mirzapur P
4 Inception Cobb
5 Bhaijaan Slemon
```



	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P
3	4	Inception	Cobb
4	5	Bhaijaan	Slemon

Problem Statement 4 : Write a program to demonstrate RowMapper interface to fetch the records from the database.

Solution :

Movie3.java

```
package org.me;

public class Movie3 {

        int mid;
        String title;
        String actor;
        public Movie3(int mid, String title, String actor) {
                super();
```

```
		this.mid = mid;
		this.title = title;
		this.actor = actor;
	}

	public Movie3() {
		super();
		// TODO Auto-generated constructor stub
	}
	public int getMid() {
		return mid;
	}
	public void setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getActor() {
		return actor;
	}
	public void setActor(String actor) {
		this.actor = actor;
	}
}
```

MovieDAO3.java

```
package org.me;


import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;


import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.RowMapper;
```

```
public class MovieDAO3 {
	JdbcTemplate jdbcTemplate;
	public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
		this.jdbcTemplate = jdbcTemplate;
	}
	public List<Movie2> getAllEmployeesRowMapper(){
		 return jdbcTemplate.query("select * from mymovies1",new
RowMapper<Movie2>(){
		    @Override
		    public Movie2 mapRow(ResultSet rs, int rownumber) throws
SQLException {
		          Movie2 e=new Movie2();
		        e.setMid(rs.getInt(1));
		        e.setTitle(rs.getString(2));
		        e.setActor(rs.getString(3));
		        return e;
		    }
		    });
		  }
}
```

appxtx3.xml

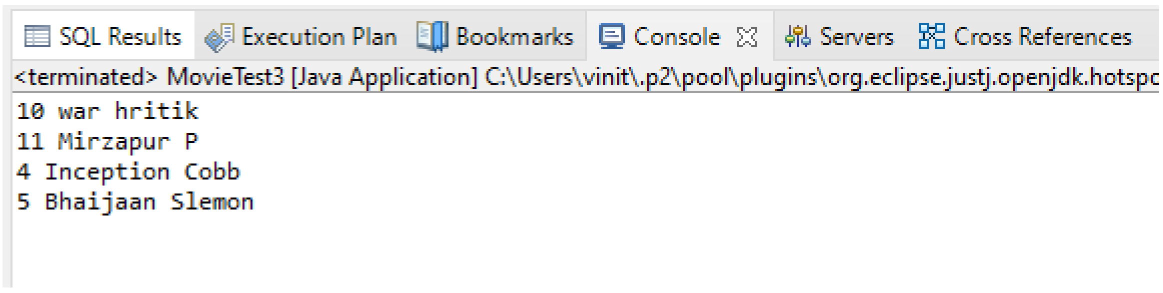
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="password" />
</bean>
```

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

<bean id="mymovie" class="org.me.MovieDAO3">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

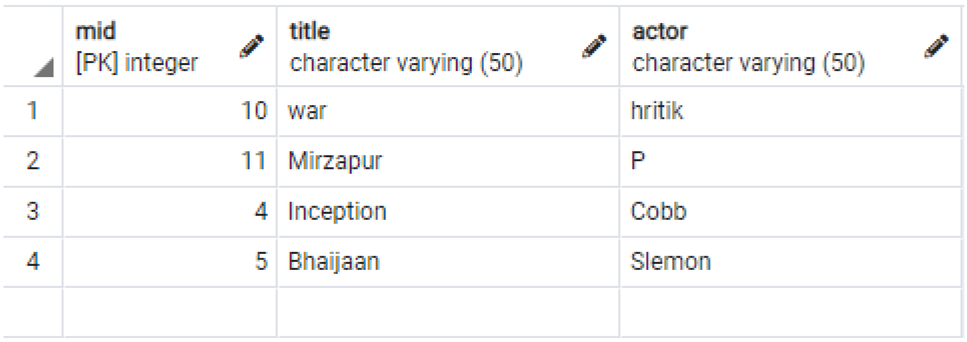
MovieTest3.java

```
package org.me;


import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest3 {

        private static ApplicationContext appCon;

        public static void main(String[] args) {

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

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

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


                   for(Movie2 e:list)

                       System.out.println(e);

        }

}
```

Output :



```
<terminated> MovieTest3 [Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspc
10 war hritik
11 Mirzapur P
4 Inception Cobb
5 Bhaijaan Slemon
```



	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P
3	4	Inception	Cobb
4	5	Bhaijaan	Slemon