

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.

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

Create Movies Table : CREATE TABLE

mymovies1 (

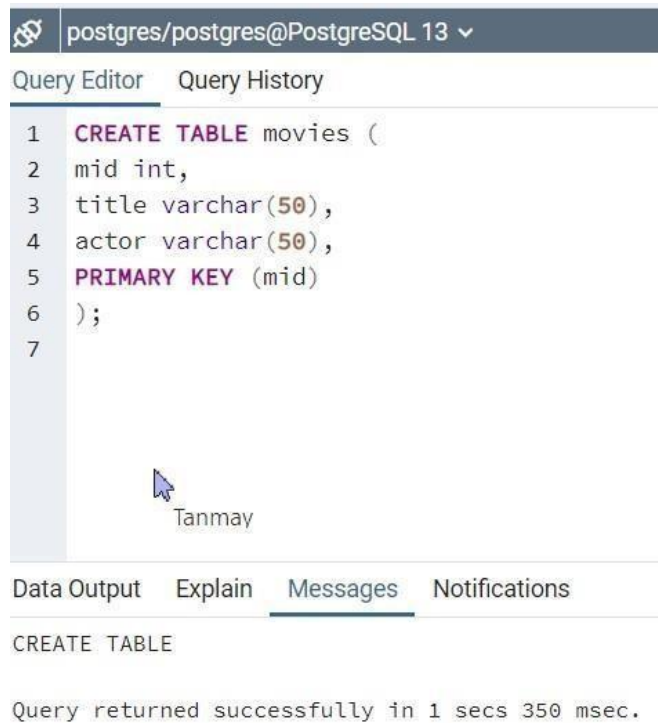
mid int, title

varchar(50), actor

varchar(50),

PRIMARY KEY (mid)

);



The screenshot shows a PostgreSQL query editor interface. At the top, the database connection is set to 'postgres/postgres@PostgreSQL 13'. Below this, there are tabs for 'Query Editor' and 'Query History'. The 'Query Editor' tab is active, displaying a SQL query to create a table named 'movies'. The query is as follows:

```
1 CREATE TABLE movies (  
2   mid int,  
3   title varchar(50),  
4   actor varchar(50),  
5   PRIMARY KEY (mid)  
6 );  
7
```

Below the query editor, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the result of the query execution:

```
CREATE TABLE
```

At the bottom, a status message indicates: 'Query returned successfully in 1 secs 350 msec.'

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

Solution :

Solution:

How to generate getter and setter methods

Right click on file-> source-> Generate getters and setters methods.

Movie1.java

```
Package com.spring;
```

```
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();  
        // 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; }  
  
}
```

MovieDAO.java

```
Package com.spring;

import org.springframework.jdbc.core.*;
public class MovieDAO { JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) { this.jdbcTemplate
    = jdbcTemplate;
}
public int insMovie(Movie1 m1)
{
    String insSql="insert into mymovies1
values("+m1.getMid()+",""+m1.getTitle()+",""+m1.getActor()+")"; return
jdbcTemplate.update(insSql);
} public int updateMovie(Movie1

m1){

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

public int 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="password" />
</bean>

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

<bean id="mymovie" class="com.springMovieDAO">
<property name="jdbcTemplate" ref="jdbcTemplate"></property> </bean>
</beans>

```

Create Main java File package com.spring; 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(8, "17 raj", "Zac");

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

Movie1 t = **new** Movie1(9, "shree", "Christopher");

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

```
// update query
int status = m1.updateMovie(new Movie1(5, "ashish", "Zac"));

System.out.println(status);

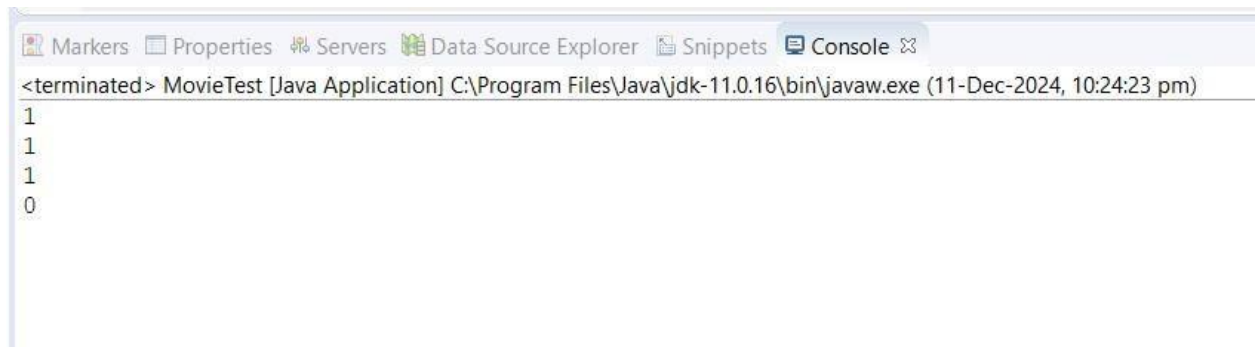
// delete

Movie1 t2=new Movie1(); t2.setMid(3); int s=m1.deleteMovie(t2);

System.out.println(s);

}

}
```



The screenshot shows an IDE console window with the following tabs: Markers, Properties, Servers, Data Source Explorer, Snippets, and Console. The console output is as follows:

```
<terminated> MovieTest [Java Application] C:\Program Files\Java\jdk-11.0.16\bin\javaw.exe (11-Dec-2024, 10:24:23 pm)
1
1
1
0
```

Update:

We update row 1

Data Output	Explain	Messages	Notifications
	mid [PK] integer	title character varying (50)	actor character varying (50)
1	1	18 Again	Zac
2	2	23 Again	Zac
3	4	17 Again	Zac
4	5	Interstellar	Christopher

Delete:

We deleted row no 3 So, After deleted row

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	mid [PK] integer	title character varying (50)	actor character varying (50)
1	4	17 Again	Zac
2	5	ashish	Zac
3	8	17 raj	Zac
4	9	shree	Christopher

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

Solution :

Movie1.java package

```
com.spring; 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 com.spring; import
java.sql.PreparedStatement; import java.sql.SQLException;
import org.springframework.dao.DataAccessException; import
org.springframework.jdbc.core.JdbcTemplate; import
org.springframework.jdbc.core.PreparedStatementCallback;
public class MovieDAO { JdbcTemplate jdbcTemplate; public
void setJdbcTemplate(JdbcTemplate jdbcTemplate)
{ this.jdbcTemplate =
jdbcTemplate;
}
public Boolean saveMovieByPreparedStatement(final Movie1 e)
{
String query="insert into mymovies1 values(?,?,?)";

return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
@Override      public Boolean
doInPreparedStatement(PreparedStatement ps)
throws SQLException, DataAccessException
{
ps.setInt(1,e.getMid());
ps.setString(2,e.getTitle());
ps.setString(3,e.getActor());      return
ps.execute();
}
});
}

```

} appctx.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/springbeans.xsd"
"> <bean id="ds"
class="org.springframework.jdbc.datasource.DriverManagerDat
aSource">
<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="com.spring.MovieDAO">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>

</beans>
```

MovieTest1.java

```
package com.spring; import
org.springframework.context.ApplicationContext;
```

```

import
org.springframework.context.support.ClassPathXmlApplic
ationContext; public class MovieTest { private static
ApplicationContext appCon; public static void
main(String[] args) { // TODO Autogenerated method stub

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

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

    m1.saveMovieByPreparedStatement(new Movie1(80,"Bhaijaan","Slemon"));

}

}

```

Output :

Data Output Messages Notifications			
	mid [PK] integer	title character varying (50)	actor character varying (50)
1	4	17 Again	Zac
2	5	ashish	Zac
3	8	17 raj	Zac
4	9	shree	Christopher
5	80	Bhaijaan	Slemon

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

Solution :

Movie2.java package

com.spring;

```
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 com.spring;  
import      java.sql.ResultSet;  
import java.sql.SQLException;
```

```

import    java.util.ArrayList;
import java.util.List;    import
org.springframework.dao.Data
AccessExcepion;    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,
        DataAccessExcepion {

            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");

```

```

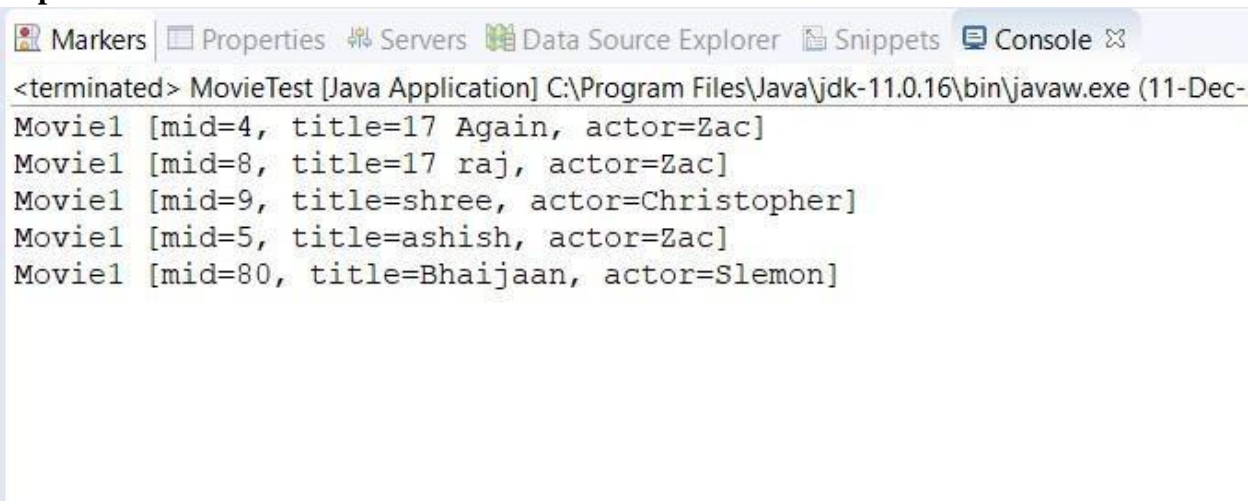
        List<Movie2> list=m1.getAllMovie();
        for(Movie2 e:list)

            System.out.println(e);

    }
}

```

Output :



```

<terminated> MovieTest [Java Application] C:\Program Files\Java\jdk-11.0.16\bin\javaw.exe (11-Dec-2020)
Movie1 [mid=4, title=17 Again, actor=Zac]
Movie1 [mid=8, title=17 raj, actor=Zac]
Movie1 [mid=9, title=shree, actor=Christopher]
Movie1 [mid=5, title=ashish, actor=Zac]
Movie1 [mid=80, title=Bhaijaan, actor=Slemon]

```

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

Solution :

Filename-Movie.java

Package com.spring;


```
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;
        }
    }
}

```

Filename- MovieDAO3.java

```

Package com.spring; 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 movies",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;

        }

    });    }

}

```

Filename- appctx3.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:5434/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="MovieDAO3">

```

```
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

Filename- MovieTest3.java

```
Package com.spring
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) { // TODO Auto-generated method
        stub 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-

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> MovieTest [Java Application] C:\Program Files\Java\jdk-11.0.16\bin\javaw.exe (11-Dec-2024, 11:33:35 pm)

com.spring.Movie1@1f53a5dc

com.spring.Movie1@1b75c2e3

com.spring.Movie1@1984b1f

com.spring.Movie1@3bd323e9

com.spring.Movie1@39ac0c0a

