Spring JDBC

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

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;
	}
}
```

MovieDAO.java

```
package org.me;
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);
	}
}
```

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) {
		appCon = new ClassPathXmlApplicationContext("appctx.xml");
		MovieDAO m1 = (MovieDAO) appCon.getBean("mymovie");
		Movie1 t1 = new Movie1(11, "Wolverine", "Hugh Jackman");
		System.out.println(m1.insMovie(t1));
		int status = m1.updateMovie(new Movie1(10, "The Dark Knight",
"Christian Bale"));
		System.out.println(status);
		Movie1 t2 = new Movie1();
		t2.setMid(5);
		status = m1.deleteMovie(t2);
		System.out.println(status);

	}
}
```

Database table

```
CREATE TABLE mymovies1

(

mid int,

title varchar(50),
```

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

select * from mymovies1;
```

Output :

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> MovieTest [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe (12-Dec-2024, 3:03:54 pm)
1
1
0
```

Data Output Messages Notifications

SQL

	mid [PK] integer	title character varying (50)	actor character varying (50)
1	11	Wolverine	Hugh Jackman
2	10	The Dark Knight	Christian Bale

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

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

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) {
		appCon = new ClassPathXmlApplicationContext("appctx1.xml");
		MovieDAO1 m1 = (MovieDAO1) appCon.getBean("mymovie");
		m1.saveMovieByPreparedStatement(new Movie1(5, "Ford vs Ferrari",
"Christian Bale"));
	}
}
```

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/demo" />
		<property name="username" value="postgres" />
		<property name="password" value="root123" />
	</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>
```

Database table

CREATE TABLE movies

(

mid int,

title varchar(50),

actor varchar(50),

PRIMARY KEY (mid)

);

select * from movies;

Output :

Data Output Messages Notifications

	mid [PK] integer	title character varying (50)	actor character varying (50)
1	5	Ford vs Ferrari	Christian Bale

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

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;
			}
		});
	}
}
```

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);
	}
}
```

Appctx2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
	<bean id="ds"
		class="org.springframework.jdbc.datasource.DriverManagerDataSource">
		<property name="driverClassName"
			value="org.postgresql.Driver" />
		<property name="url"
			value="jdbc:postgresql://localhost:5432/demo" />
		<property name="username" value="postgres" />
		<property name="password" value="root123" />
```

```
        </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>
```

Database table

CREATE TABLE mymovies1

(

mid int,

title varchar(50),

actor varchar(50),

PRIMARY KEY (mid)

);

select * from mymovies1;

Output :

Data Output Messages Notifications

	mid [PK] integer	title character varying (50)	actor character varying (50)
1	11	Wolverine	Hugh Jackman
2	10	The Dark Knight	Christian Bale

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> MovieTest2 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe (12-Dec-2024, 3:27:30 pm)
11 Wolverine Hugh Jackman
10 The Dark Knight Christian Bale
```

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

Movie3.java

```
package org.me;
public class Movie3 {
    int mid;
    String title;
    String actor;

    public Movie3(int mid, String title, String actor) {
        super();
        this.mid = mid;
        this.title = title;
        this.actor = actor;
    }

    public Movie3() {
        super();
    }

    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;
    }

    // Override the toString() method
    @Override
    public String toString() {
        return "Movie ID: " + mid + ", Title: " + title + ", Actor: " + actor;
    }
}
```

MovieDAO3.java

```
package org.me;
```

```java
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<Movie3> getAllEmployeesRowMapper() {
		return jdbcTemplate.query("select * from mymovies1", new
RowMapper<Movie3>() {
			@Override
			public Movie3 mapRow(ResultSet rs, int rownumber) throws
SQLException {
				Movie3 e = new Movie3();
				e.setMid(rs.getInt(1));
				e.setTitle(rs.getString(2));
				e.setActor(rs.getString(3));
				return e;
			}
		});
	}
}
```

MovieTest3.java

```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<Movie3> list = m1.getAllEmployeesRowMapper();
		for (Movie3 e : list)
			System.out.println(e);
	}
}
```

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/demo" />
                <property name="username" value="postgres" />
                <property name="password" value="root123" />
        </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>
```

Database table

CREATE TABLE mymovies1

(

mid int,

title varchar(50),

actor varchar(50),

PRIMARY KEY (mid)

);

Select * from mymovies1;

Output :

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Console
<terminated> MovieTest3 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe (12-Dec-2024, 3:40:10 pm)
Movie ID: 11, Title: Wolverine, Actor: Hugh Jackman
Movie ID: 10, Title: The Dark Knight, Actor: Christian Bale
```

Data Output Messages Notifications

	mid [PK] integer	title character varying (50)	actor character varying (50)
1	11	Wolverine	Hugh Jackman
2	10	The Dark Knight	Christian Bale