## Spring JDBC

**Spring Java Database Connectivity** 

### Introduction to Spring JDBC

#### What is Spring JDBC?

- Spring JDBC is a part of the Spring Framework that simplifies the use of JDBC (Java Database Connectivity) to interact with relational databases.
- It provides a consistent way to access databases and helps to avoid common errors and boilerplate code associated with traditional JDBC.

### Why use Spring JDBC?

- Reduces Boilerplate Code: Spring JDBC eliminates the need for repetitive code, such as opening and closing connections, handling exceptions, and managing transactions.
- Enhances Productivity: Developers can focus on writing business logic rather than dealing with low-level database operations.
- Improves Maintainability: The code is cleaner and easier to maintain.

#### Key Components of Spring JDBC

#### JdbcTemplate

- Core class for database operations.
- Simplifies the use of JDBC and helps to avoid common errors.

#### DataSource

- Interface for getting database connections.
- Supports connection pooling.

#### RowMapper

- Interface for mapping rows of a ResultSet to Java objects.
- NamedParameterJdbcTemplate
  - Extension of JdbcTemplate.
  - Supports named parameters for queries.

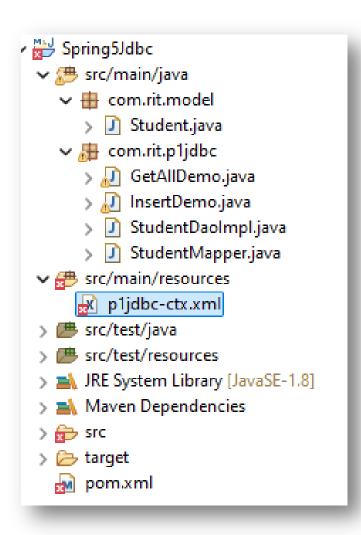
#### Setting Up Spring JDBC

- Dependencies
  - Maven/Gradle dependencies for Spring JDBC.
- Configuration
  - XML-based configuration.
  - Java-based configuration using @Configuration and @Bean.

#### Basic CRUD Operations

- Create
  - Using JdbcTemplate.update() for insert operations.
- Read
  - Using JdbcTemplate.query() and RowMapper for select operations.
- Update
  - Using JdbcTemplate.update() for update operations.
- Delete
  - Using JdbcTemplate.update() for delete operations.

### Step by step



- Create a maven project
- Include dependencies
- Create DB Configuration in XML
- Create a Model (Student) Class
- Create a StudentMapper Class
- Create a StudentDaoImpl Class
  - Implement CRUD
- Create a main class

### Include Dependencies

```
<dependency>
                                       <dependency>
                                       <groupId>org.springframework
<groupId>org.springframework
<artifactId>spring-jdbc</artifactId>
                                       <artifactId>spring-beans</artifactId>
<version>5.3.31
                                       <version>5.3.31
</dependency>
                                       </dependency>
<dependency>
                                       <dependency>
<groupId>org.springframework
                                       <groupId>org.springframework
<artifactId>spring-
                                       <artifactId>spring-tx</artifactId>
context</artifactId>
                                       <version>5.3.31
<version>5.3.31
                                       </dependency>
</dependency>
                                       <dependency>
<dependency>
                                       <groupId>mysql
<groupId>org.springframework
                                       <artifactId>mysql-connector-java</artifactId>
<artifactId>spring-core</artifactId>
                                       <version>8.0.33</version>
<version>5.3.31
                                       <scope>runtime</scope>
                                      </dependency>
</dependency>
```

### DB Configuration in XML

```
√ Æ com.rit.p1jdbc

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

✓ 

Æ src/main/resources

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
                                                                                  p1jdbc-ctx.xml
    <bean id="dataSource"</pre>
                                                                               > # src/test/java
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        cproperty name="url" value="jdbc:mysql://localhost:3306/spring db" />
        cproperty name="username" value="root" />
        cproperty name="password" value="root" />
    </bean>
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        cproperty name="dataSource" ref="dataSource" />
    </bean>
    <bean id="studentDao" class="com.rit.p1jdbc.StudentDaoImpl">
        cproperty name="jdbcTemplate" ref="jdbcTemplate" />
    </bean>
</beans>
```

▼ 

Spring5Jdbc

com.rit.model

Student.java

> III GetAllDemo.java

> InsertDemo.java

> J StudentDaolmpl.java

StudentMapper.java

### Create a Model (Student) Class

```
public class Student {

    // Member variables
    private int id;
    private String name;
    private String department;

    //constructors, Getters, Setters and toString() method
```

```
v 👑 Spring5Jdbc

✓ № com.rit.p1jdbc

      > 🔃 GetAllDemo.java
       InsertDemo.java
       StudentDaolmpl.java
      > I StudentMapper.java

✓ 
## src/main/resources

     p1jdbc-ctx.xml
```

#### Create a StudentMapper Class

```
public class StudentMapper implements RowMapper<Student> {
    @Override
    public Student mapRow(ResultSet rs, int rowNum)
    throws SQLException {
        Student student = new Student();
        student.setId(rs.getInt("id"));
        student.setName(rs.getString("name"));
        student.setDepartment(rs.getString("department"));
        return student;
    }
}
```

```
Spring5Jdbc

Spring5Jdbc

Student.java

Student.java

Com.rit.p1jdbc

GetAllDemo.java

InsertDemo.java

StudentDaolmpl.java

StudentMapper.java

StudentMapper.java

Figure StudentMapper.java

InsertDemo.java

StudentMapper.java

StudentMapper.java

StudentMapper.java

Figure StudentMapper.java

StudentMapper.java

StudentMapper.java

StudentMapper.java
```

#### Create a StudentDoalmpl Class

```
> 🔎 GetAllDemo.java
public class StudentDaoImpl {
                                                                     InsertDemo.java
                                                                     private JdbcTemplate;
                                                                    StudentMapper.java

✓ ## src/main/resources

public JdbcTemplate getJdbcTemplate() {
                                                                   p1jdbc-ctx.xml
   return jdbcTemplate;
                                                                 src/test/java
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
       this.jdbcTemplate = jdbcTemplate;
//Insert a Student Object
public int saveStudent(Student student) {
   String sql = "INSERT INTO student (id, name, department) VALUES (?, ?, ?)";
   return jdbcTemplate.update(sql, student.getId(), student.getName(),
   student.getDepartment());
```

✓ 

Spring5Jdbc

Spring5Dac

S

▼ 

## src/main/java

→ 

⊕ com.rit.model

✓ M com.rit.p1jdbc

Student.java

```
public List<Student> getAllStudents() {
   String sql = "SELECT * FROM student";
   List<Student> list = jdbcTemplate.query(sql, new StudentMapper());
   return list;
public int updateStudent(Student student) {
   String sql = "UPDATE student SET name = ?, department = ? WHERE id = ?";
   return jdbcTemplate.update(sql, student.getName(), student.getDepartment(),
   student.getId());
public int deleteStudent(int id) {
   String sql = "DELETE FROM student WHERE id = ?";
   return jdbcTemplate.update(sql, id);
public Student getStudentById(int id) {
   NamedParameterJdbcTemplate namedParam = new
   NamedParameterJdbcTemplate(jdbcTemplate);
   String sql = "SELECT * FROM student WHERE id = :id";
   Map<String, Object> params = new HashMap<>();
   params.put("id", id);
   Student student = namedParam.queryForObject(sql, params, new
   StudentMapper());
   return student;
```

#### Main class

```
public class GetAllDemo {
   public static void main(String[] args) {
       AbstractApplicationContext context = new
       ClassPathXmlApplicationContext("p1jdbc-ctx.xml");
       StudentDaoImpl studentDaoImpl =
       (StudentDaoImpl)context.getBean("studentDao");
       List<Student> studentList = studentDaoImpl.getAllStudents();
       for(Student student : studentList) {
              System.out.println(student);
```

### Main class (Insert, Update, Delete & GetByld)

```
public class InsertDemo {
   public static void main(String[] args) {
       AbstractApplicationContext context = new
       ClassPathXmlApplicationContext("p1jdbc-ctx.xml");
       StudentDaoImpl studentDaoImpl =
       (StudentDaoImpl)context.getBean("studentDao");
       Student student = new Student(202, "Anand", "Computers");
       int res = studentDaoImpl.saveStudent(student);
       if(res == 0)
           System.out.println("Error Saving Student");
       else
           Svstem.out.println(studentDaoImpl.getStudentById(202));
           student.setDepartment("Science");
           studentDaoImpl.updateStudent(student);
           System.out.println(studentDaoImpl.getStudentById(202));
           studentDaoImpl.deleteStudent(202);
           System.out.println("Record Deleted");
```

# Thank you