Class Examples Git Repo : https://github.com/ashokitschool/sbms-63.git

============================
Spring JDBC / Spring DAO
============================

=> Spring JDBC is part of the Spring Framework

=> It is used to simplify DB operations in our java applications.

=> Spring JDBC handles resource management, exception handling, and reduces boilerplate code.

=> Spring JDBC provided predefined components and methods to simplify Db operations

Ex : JdbcTemplate, RowMapper, DataSource....

---------- Normal JDBC Code----------

```
Connection conn = DriverManager.getConnection(url, user, pass);

PreparedStatement stmt = conn.prepareStatement("SELECT * FROM students");

ResultSet rs = stmt.executeQuery();

while(rs.next()) {
    Student s = new Student();
    s.setId(rs.getInt("id"));
    s.setName(rs.getString("name"));
}

rs.close();
stmt.close();
conn.close();
```

---------- Spring JDBC ----------

```
String query = "SELECT * FROM students";

List<Student> students = jdbcTemplate.query(query, new StudentRowMapper());
```

==============================
Key Components of Spring JDBC
==============================

1) DataSource (I)

2) JdbcTemplate

3) RowMapper (I)

=> DataSource is an interface it is used to represent Database Connection Pooling.

=> JdbcTemplate is a core class of spring jdbc, which is used to execute SQL queries.

=> RowMapper is used to map resultset data to java objects.

==========

Developing First Spring JDBC Application
==========

@@@@ Project Lombok Tutorial By Mr. Ashok : https://youtu.be/8tDym-FxU0A?si=8M8PClnyApWpFZKV

Step-1 : Setup MySQL DB Server and Workbench(DB Client)

@@ Reference Video : https://www.youtube.com/watch?v=EsAIXPIsyQg

Step-2 : Create Database table

```
CREATE DATABASE sbms63;
USE sbms63;

CREATE TABLE student (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    city VARCHAR(100)
);
```

Step-3 : Create Maven Project with required dependencies

a) spring-context
b) spring-jdbc
c) mysql-driver

Step-4 : Create DTO class to represent data in object format.

```
public class Student {

    private int id;
    private String name;
    private String city;

    //setters and getters
}
```

Step-4 : Create DAO class to perform DB operations using JdbcTemplate class.

```
public class StudentDao {

    private JdbcTemplate jdbcTemplate;

    public StudentDao(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public int insert(Student s) {

        String query = "INSERT INTO student (id, name, city) VALUES (?, ?, ?)";

        int cnt = jdbcTemplate.update(query, s.getId(), s.getName(), s.getCity());

        return cnt;
    }
}
```

Step-5 : Create Beans config file and represent java classes as spring beans.

```
<bean id="ds"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/sbms63" />
    <property name="username" value="root" />
    <property name="password" value="root" />
</bean>

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

<bean id="sdao" class="in.ashokit.dao.StudentDao">
    <constructor-arg ref="jt" />
</bean>
```

Step-6 : Create main class and test the application.

```
public class App {

    public static void main(String[] args) {

        ApplicationContext ctxt = new ClassPathXmlApplicationContext("beans.xml");

        StudentDao sdao = ctxt.getBean(StudentDao.class);

        Student student = new Student();
        student.setId(2);
        student.setName("John");
        student.setCity("Vizag");

        int cnt = sdao.insert(student);
        System.out.println("Record Inserted : " + cnt);

    }
}
```

==========
what is row mapper in spring jdbc
==========

=> RowMapper is an interface provided by Spring JDBC that helps you to map rows from a ResultSet to Java objects.

=> When you run a SQL query to retrieve records from table , the results will come in the form of a ResultSet. But our java application usually works with objects. RowMapper bridges that gap.

```
public interface RowMapper<T> {
    T mapRow(ResultSet rs, int rowNum) throws SQLException;
}
```

Note: You implement the mapRow() method to convert each row of the result set into a Java object.

```
public class StudentRowMapper implements RowMapper<Student> {
```

```
    @Override
    public Student mapRow(ResultSet rs, int rowNum) throws SQLException {

        Student s = new Student();
        s.setId(rs.getInt("id"));
        s.setName(rs.getString("name"));
        s.setCity(rs.getString("city"));

        return s;
    }
}
```

```
public List<Student> getStudents() {

    String sql = "select * from student";

    List<Student> students = jdbcTemplate.query(sql, new StudentRowMapper());

    return students;
}
```

==========