**Common configuration for Any Spring Jdbc Approach:**

In case of Java JDBC Connection which is the 1st and fore most thing bcz without connection we cannot do anything. Similarly in spring we have a class called JdbcTemplate which acts as centric for all the classes and to interact with DB. That it is going to get the connection from the DB using Connection class internally to interact with DB.

class StudentDao {

}

🡪For example if we wanted to execute the SQL select query then we need to talk to the jdbcTemplate bcz it can talk to the DB so we need to create the obj of jdbcTemplate obj so that if we give SQL query to the jdbcTemplate it establishes the connection, creates the Statement, executes the Statement and iterating over the resultset. Here we are not creating the connection so JdbcTemplate need to create the connection so in order to create the connection we need to create/configure dataSource to get the connection from that dataSource. That's why we need to configure the dataSouce so that jdbcTemplate will asks the dataSource to create the connection and holds with it.

🡪Configuring the dataSource or creating the Data Source means configuring it as bean in spring bean configuration file.

🡪In order to configure dataSource as bean we need dataSource impl class bcz javax.sql.DataSource is an interface so we cannot configure interfaces as beans bcz we/IOC cannot create obj for interfaces.

🡪javax.sql.DataSource (interface from java API)

|-For this interface there are application multiple server vendors provided impl so that impl classes are part of thier server and application sholud be up and running.

|-Spring JDBC will supports for Non-J2EE as well that means we need DataSource impl class in an Non-J2EE as well to work with Non-J2EE applications that's where there are multiple 3rd party libraries comes in to picture.

For Non-J2EE (outside J2EE container) Application Connection Pool libraries:

dbcp (no use bcz this it has been deprecated)

c3po

Proxool

Which provides Data Base Connection pool libraries by providing the Impl classes for DataSource.

Apart from these spring has provided one impl class for DataSource interface called as DriverManagerDataSource (internally uses DriverManager to create the connection) which will gets the Connection. That means using DriverManagerDataSource Spring Jdbc will gets the connection and gives to the jdbctemplate. But DriverManagerSource that is provided by the spring will not gives any connection pools and it is just for the sake of working easily it provided dummy Implementaion.

1. driverClass

2. url

3. username

4. password

So we need to configure thses in the spring bean configuration file.

🡪How to you work with Connection pools in an J2EE and Non-J2EE environment?

In J2EE environments the server vendor will provides the implementation for the DataSource interface so that we don't need to use any 3rd party libraries to configure the connection data source pool. But if we are working with Non-J2EE (outside the J2EE) then we need to use the any one of the 3rd party libraries like c3po libraries.

Java group web site deployed on the Tomcat server and tomcat will not provide the connection pools so he has used c3po libraries to configure the pool of connections in tomcat server bcz it has been deployed in the tomcat so that performance of the application will not go down.

application-context.xml

<bean id="dataSource" class="DriverManagerDataSource">

<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />

<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />

<property name="username" value="spring\_user" />

<property name="password" value="spring\_user" />

</bean>

🡪Once we configure the data source we need to give this dataSource obj to the Spring JdbcTemplate so that it can establishes the connection. So we need to inject the dataSource in to the JdbcTemplate by declaring JdbcTemplate as a bean and inject dataSource via setter or constructor that means JdbcTemplate will have Default constructor and parameterized constructor and parameterized constructor used to inject the constructor injection and setter for setter injection.

🡪StudentDao needs the connection to perform the operation so we need to inject the jdbcTemplate into the StudentDao otherwise StudentDao cannot perform the persistency operations. So we need to inject the jdbcTemplate via setter or contructor, but it is recomended to use constructor bcz without the jdbcTemplate we cannot perform the persistency operations so we need to use constructor injection.

class StudentDao {

private JdbcTemplate jdbcTemplate;

public StudentDao() {

}

//inject via constructor injection

public StudentDao(JdbcTemplate jdbcTemplate) {

this.jdbcTemplate=jdbcTemplate;

}

}

<bean id="jdbcTemplate" class="JdbcTemplate">

<constructor-arg ref="dataSource" />

</bean>

<bean id="studentDao" class="StudentDao">

<constructor-arg ref="jdbcTemplate" />

</bean>

🡪So this is the basic configuration logic we need to write for any Approach while working with Spring JDBC.

🡪What is the tool you are using while working with Database?

We are using Oracle SQL Developer. We can tell even we are using TOAD which is commercial which works with any DB. But Oracle SQL Developer and it works with only Oracle DB.

🡪What is the first-step code that you are going to write while you are working with Spring JDBC and can u plz write that snippet of code?

We wanted to perform the persistency logic 1st we need to writ the DAO class and then we need to configure the data source.