🔰 Spring JDBC

1. Understand JDBC Basics

What is JDBC?

- JDBC stands for **Java Database Connectivity**.
- It allows Java programs to **connect** to a database, **send SQL queries**, and **process the results**.
- Think of JDBC as a bridge between your Java application and a relational database (like MySQL, Oracle, etc.).

Key Components of JDBC:

- **Connection** – Connects Java code to the database.
- **Statement** – Used to run **static SQL** queries (without parameters).
- **PreparedStatement** – Used to run **parameterized SQL** queries (more secure and reusable).
- **ResultSet** – Holds the **data returned** from the database after executing a SELECT query.

Problems with raw JDBC:

- A lot of **boilerplate code** – opening connections, creating statements, closing everything manually.
- **Repetitive error handling** – try-catch-finally blocks everywhere.
- Hard to maintain, test, and debug.
- Doesn't handle **connection pooling** or **exception translation** well.

---

2. Why Spring JDBC?

What is Spring JDBC?

- A module in the Spring Framework that **simplifies working with JDBC**.
- It mainly uses **JdbcTemplate**, a helper class to handle common JDBC operations.

Advantages of Spring JDBC:

✅ Simplifies JDBC code
You don't need to write code for opening/closing connections or handling exceptions every time.

✅ Less boilerplate
Just focus on the SQL and data — Spring does the rest.

✅ Better error handling
Translates SQL exceptions into more readable **Spring exceptions** (DataAccessException and its subclasses).

✅ Cleaner and more readable
Code becomes shorter, cleaner, and easier to maintain.

Step-by-Step Spring JDBC Setup and Small Project

🚀 **Project Objective:** Build a simple Spring JDBC application to **insert** and **update** employee data in a MySQL database.

1. Environment Setup

Tools Required:

- Java JDK 11+ (LTS recommended)
- Maven
- MySQL (or any RDBMS)
- IntelliJ IDEA / Eclipse

---

2. Create Maven Project

pom.xml dependencies:

```
        <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-core</artifactId>
          <version>7.0.0-M7</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
```

```
            <version>7.0.0-M7</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>7.0.0-M7</version>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
        </dependency>
```

---

3. Create MySQL Table

```
        CREATE DATABASE jdbcstudent;
        USE jdbcstudent;

        CREATE TABLE student (
        rollNo INT PRIMARY KEY,
        name VARCHAR(100),
        course VARCHAR(100)
        );
```

🛠 4. Spring Configuration

config.xml (in src/main/java -> package (com.bytexl.jdbc))

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

```
		<bean name="driverManager"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
			<property name="driverClassName"
value="com.mysql.cj.jdbc.Driver" />
			<property name="url"
value="jdbc:mysql://localhost:3306/jdbcstudent" />
			<property name="username" value="root" />
			<property name="password" value="Rishabh@124"
/>
		</bean>

		<bean name="jdbcTemplate"
class="org.springframework.jdbc.core.JdbcTemplate">
			<property name="dataSource">
				<ref bean="driverManager" />
			</property>
		</bean>
</beans>
```

---

5. Model + DAO Layer

Student.java

```
package com.bytexl.jdbc;

public class Students {
	private int rollNo;
	private String name;
	private String course;

	public int getRollNo() {
		return rollNo;
	}
	public void setRollNo(int rollNo) {
		this.rollNo = rollNo;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getCourse() {
		return course;
	}
```

```
	public void setCourse(String course) {
		this.course = course;
	}

	public Students(int rollNo, String name, String course) {
		super();
		this.rollNo = rollNo;
		this.name = name;
		this.course = course;
	}
	public Students() {
		super();
		// TODO Auto-generated constructor stub
	}

	@Override
	public String toString() {
		return "Students [rollNo=" + rollNo + ", name=" + name
+ ", course=" + course + "]";
	}
}
```

🪠 6. Main Application

App.java

```
package com.bytexl.jdbc;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;

public class App {
	public static void main(String[] args) {
		System.out.println("Jdbc Connectivity..!");
		ApplicationContext context = new
ClassPathXmlApplicationContext("com/bytexl/jdbc/config.xml");
		JdbcTemplate template =
context.getBean("jdbcTemplate", JdbcTemplate.class);

		System.out.println(template);

		String query = "insert into student(rollNo, name,
course) values(?,?,?)";
```

```
                int result = template.update(query,102, "Aman Singh",
"ADSA");
                System.out.println("Record Inserted.." + result);

        }
}
```

☑ Output

When you run the app:

Insert Status: 1

Additional Learning Resources

- Spring JDBC Official Docs
- Baeldung: Spring JDBC Tutorial

Summary for Your Notes

Topic	Simple Explanation
JDBC	Java API to connect and work with databases.
Connection	Establishes the link between Java and DB.
Statement	Executes simple SQL queries.
PreparedStatement	Executes SQL with input values (safer).
ResultSet	Holds query results like a table.
Raw JDBC Problems	Too much code, hard to handle errors, manual connection management.
Spring JDBC	Simplifies JDBC using JdbcTemplate.
Why Spring JDBC?	Less code, automatic connection handling, better error handling, cleaner code.

You can also follow this screen shorts to setup your projects:

```xml
<!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>


<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.30</version>
</dependency>
```

ctrl + shift + T

Search For JDBC Template

Goto Top Copy the package
org.springframework.jdbc.core.JdbcTemplate

paste this in `config.xml file` inside class attribute

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

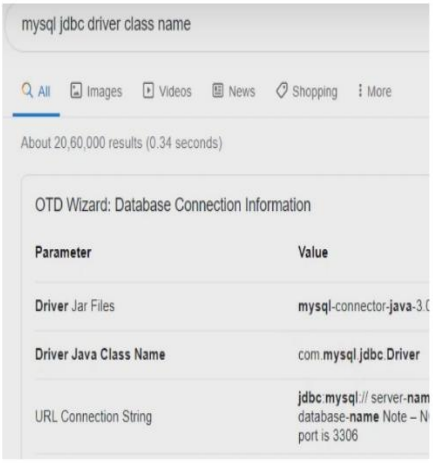
<bean class="" name="ds"></bean>

Goto main exe class and type ctrl + shift + T

Search For -> Driver Manager DataSource

Copy The Package From The Top:
org.springframework.jdbc.datasource

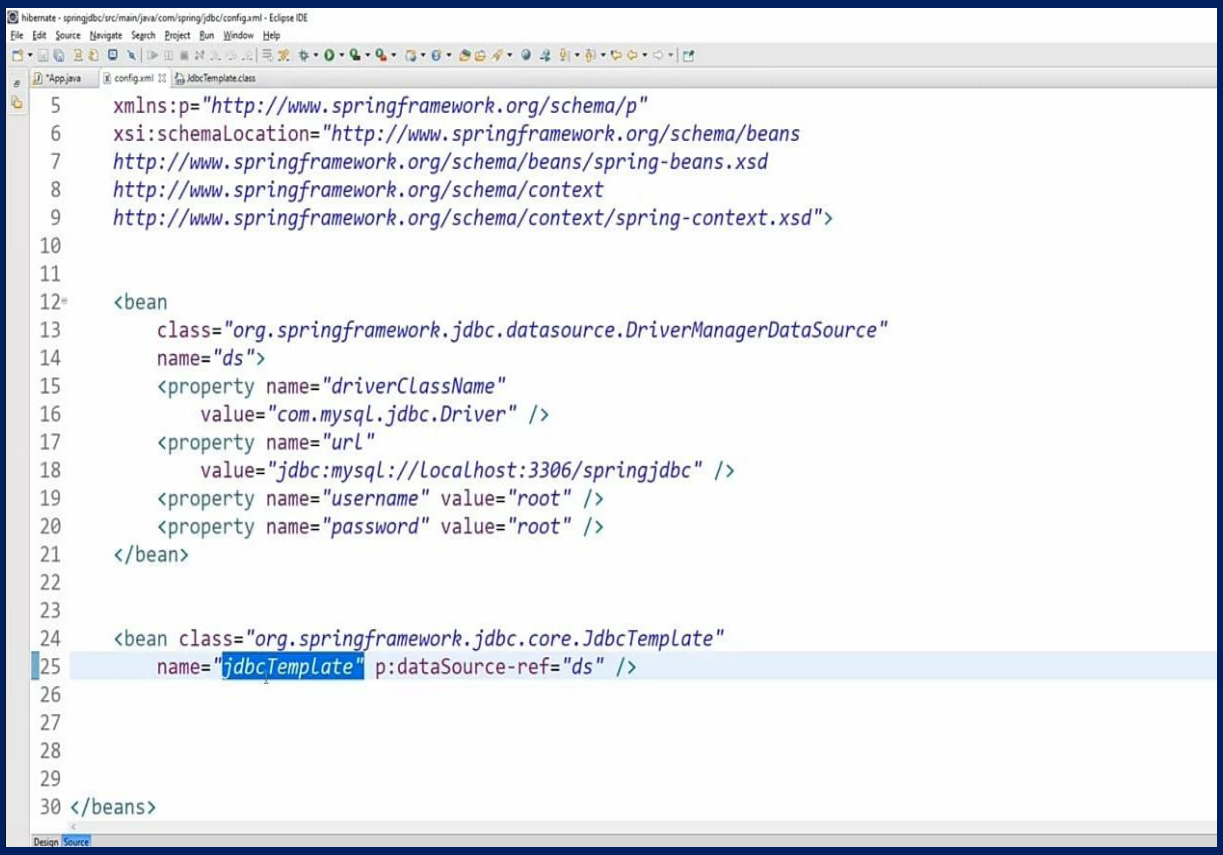
Then go & copy the class name:
DriverManagerDataSource



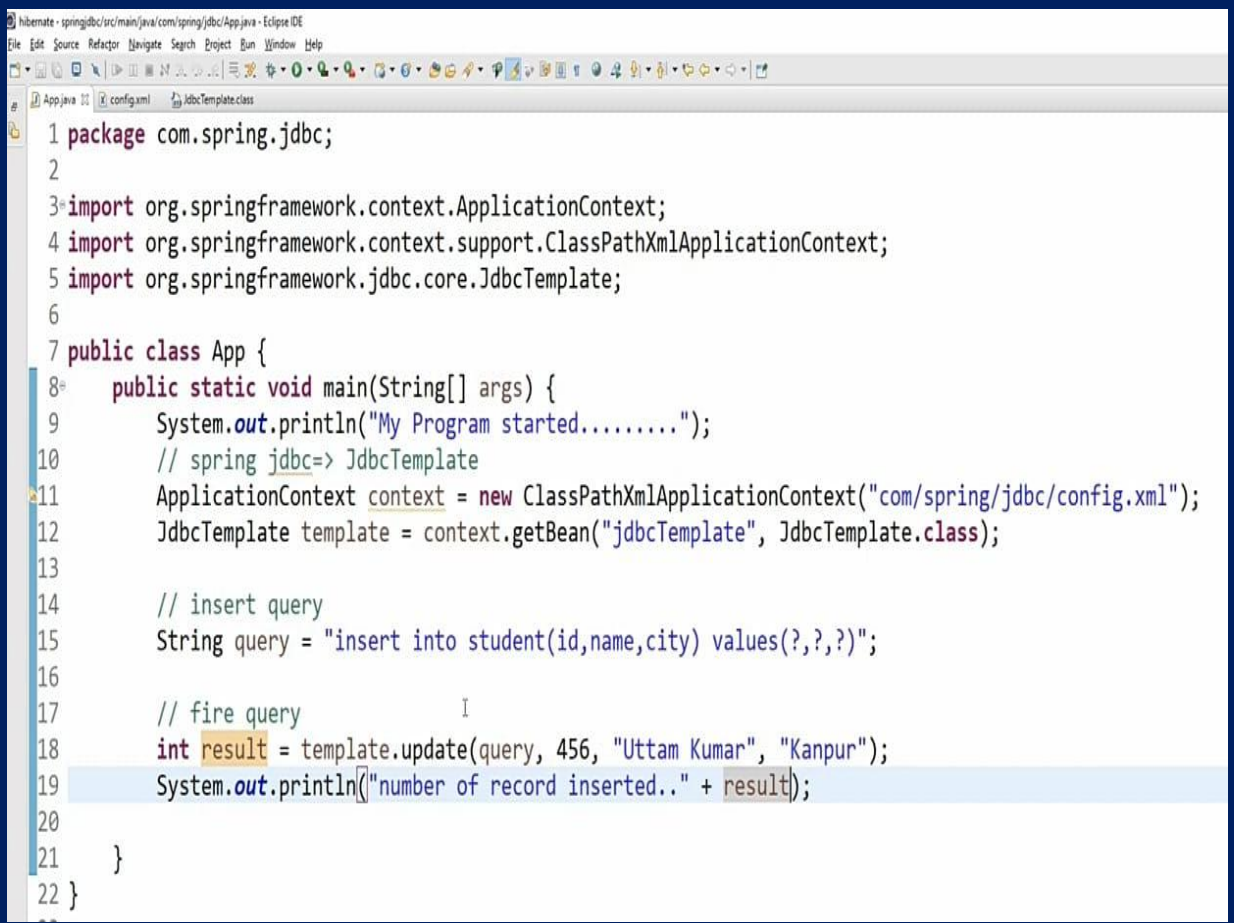

```
<bean class="org.springframework.jdbc.datasource.DriverManagerDataSource" name="ds" >
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/springjdbc" />
    <property name="username" value="root" />
    <property name="password"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

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


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



```
hibernate - springjdbc/src/main/java/com/spring/jdbc/config.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
*App.java  config.xml  JdbcTemplate.class
 5     xmlns:p="http://www.springframework.org/schema/p"
 6     xsi:schemaLocation="http://www.springframework.org/schema/beans
 7     http://www.springframework.org/schema/beans/spring-beans.xsd
 8     http://www.springframework.org/schema/context
 9     http://www.springframework.org/schema/context/spring-context.xsd">
10
11
12     <bean
13         class="org.springframework.jdbc.datasource.DriverManagerDataSource"
14         name="ds">
15         <property name="driverClassName"
16             value="com.mysql.jdbc.Driver" />
17         <property name="url"
18             value="jdbc:mysql://localhost:3306/springjdbc" />
19         <property name="username" value="root" />
20         <property name="password" value="root" />
21     </bean>
22
23
24     <bean class="org.springframework.jdbc.core.JdbcTemplate"
25         name="jdbcTemplate" p:dataSource-ref="ds" />
26
27
28
29
30 </beans>
Design Source
```



```
hibernate - springjdbc/src/main/java/com/spring/jdbc/App.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
App.java  config.xml  JdbcTemplate.class
 1 package com.spring.jdbc;
 2
 3 import org.springframework.context.ApplicationContext;
 4 import org.springframework.context.support.ClassPathXmlApplicationContext;
 5 import org.springframework.jdbc.core.JdbcTemplate;
 6
 7 public class App {
 8     public static void main(String[] args) {
 9         System.out.println("My Program started.....");
10         // spring jdbc=> JdbcTemplate
11         ApplicationContext context = new ClassPathXmlApplicationContext("com/spring/jdbc/config.xml");
12         JdbcTemplate template = context.getBean("jdbcTemplate", JdbcTemplate.class);
13
14         // insert query
15         String query = "insert into student(id,name,city) values(?,?,?)";
16
17         // fire query
18         int result = template.update(query, 456, "Uttam Kumar", "Kanpur");
19         System.out.println("number of record inserted.." + result);
20
21     }
22 }
```