Spring\_JDBCTemplate

**1. 配置数据库连接：**

**1.1 使用：mysql-connector-java-5.x.xx.jar, 即5.开头的连接器**

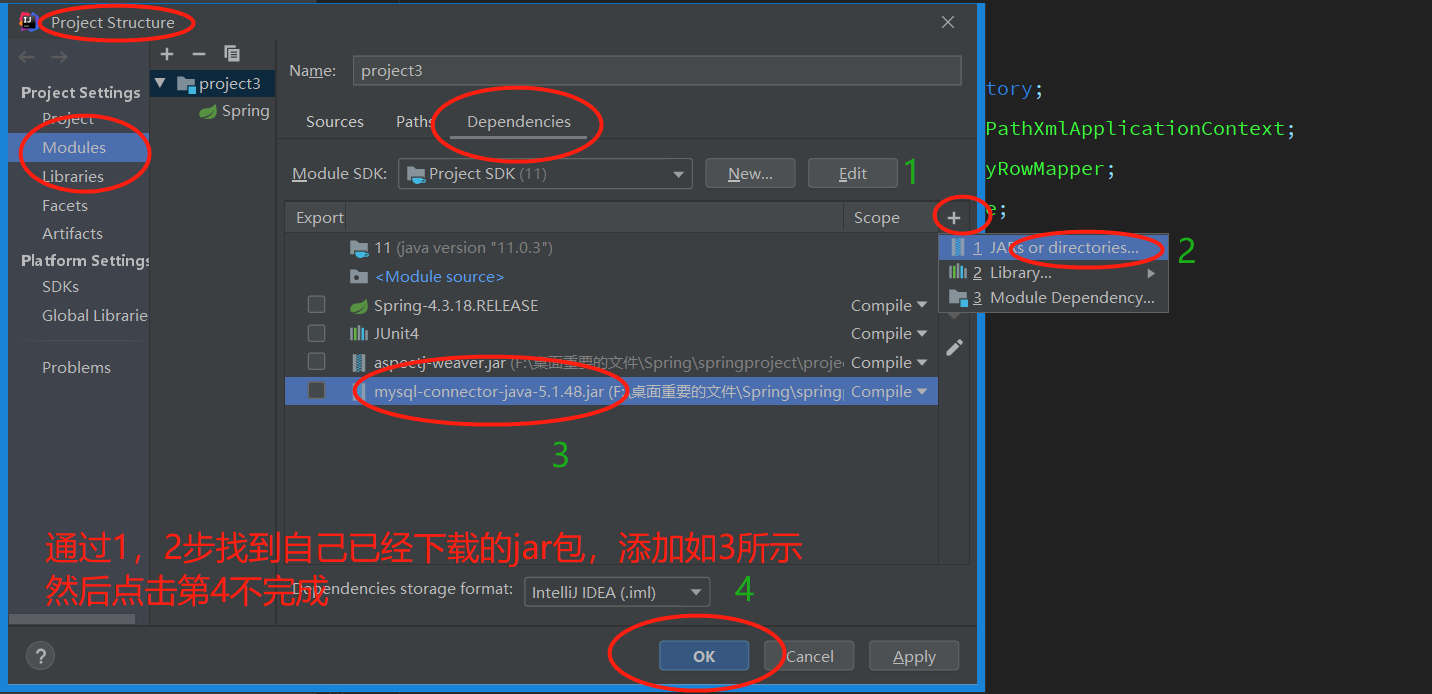
**1.1.1 xml配置代码框架：**

<context:annotation-config/>

<bean *id*="dataSource" *class*="org.springframework.jdbc.datasource.DriverManagerDataSource">  
 <property *name*="driverClassName" *value*="com.mysql.jdbc.Driver"/>  
 <property *name*="url" *value*="jdbc:mysql://localhost/jdbc"/>  
 <property *name*="username" *value*="jjf"/>  
 <property *name*="password" *value*="mysqlroot"/>  
</bean>  
  
<bean *id*="jdbcTemplate" *class*="org.springframework.jdbc.core.JdbcTemplate">  
 <property *name*="dataSource" *ref*="dataSource"/>  
</bean>

注：上面代码中只有红色字体下划线部分的内容需要改变，依此表示选择数据库名称，用户名，登陆密码，上面是数据库连接的搭建，但com.mysql.jdbc.Driver资源还没有导入，即mysql-connector-java-5.x.xx.jar包还没有导入.

**1.1.2 导入mysql-connector-java-5.x.xx.jar包**



**这样便可以开始数据库编程，然而，如果要选择更高版本的jdbc。上面步骤替换为：**

**1.2 使用mysql-connector-java-8.x.xx.jar，即8版本开头的数据库连接器**

**1.2.1 xml配置代码框架：**

<context:annotation-config/>

<bean *id*="dataSource" *class*="org.springframework.jdbc.datasource.DriverManagerDataSource">  
 <property *name*="driverClassName" *value*="com.mysql.cj.jdbc.Driver"/>  
 <property *name*="url" *value*="jdbc:mysql://localhost:3306/jdbc?serverTimezone=GMT%2B8&amp;useSSL=false"/>  
 <property *name*="username" *value*="jjf"/>  
 <property *name*="password" *value*="mysqlroot"/>  
</bean>  
  
<bean *id*="jdbcTemplate" *class*="org.springframework.jdbc.core.JdbcTemplate">  
 <property *name*="dataSource" *ref*="dataSource"/>  
</bean>

**注：显然合1.1.1代码框架稍有不同，如：**com.mysql.cj.jdbc.Driver**中多了cj包，且GMT%2B8表示时区**

**1.2.1 导入mysql-connector-java-8.0.13.jar包，步骤和上面一样；**

**2. 操作数据库实例：**

**2.1 增**

BeanFactory factory = *new ClassPathXmlApplicationContext*("JDBC/Spring\_JDBC/jdbc\_configure/JDBC\_Configure.xml");

JdbcTemplate jdbcTemplate = factory.getBean("jdbcTemplate", JdbcTemplate.*class*);  
String sqlInsert = "insert into student values(null ,'jjf',22)";   
*int* count = jdbcTemplate.update(sqlInsert);  
System.out.println(count);

**注：sqlInset 表示mysql插入语句，其中如果值是字符串，则需用单引号，如’jjf’.**

jdbcTemplate.update(sqlInsert);**相当于执行相应的mysql语句，返回值表示数据库操作的行数，这里增加一条记录，所以返回值是1**

**2.2 删**

BeanFactory factory = *new ClassPathXmlApplicationContext*("JDBC/Spring\_JDBC/jdbc\_configure/JDBC\_Configure.xml");

JdbcTemplate jdbcTemplate = factory.getBean("jdbcTemplate", JdbcTemplate.*class*);  
String sqlDelete = "delete from student where id > 3";  
System.out.println(jdbcTemplate.update(sqlDelete));

**2.3 改**

BeanFactory factory = *new ClassPathXmlApplicationContext*("JDBC/Spring\_JDBC/jdbc\_configure/JDBC\_Configure.xml");

JdbcTemplate jdbcTemplate = factory.getBean("jdbcTemplate", JdbcTemplate.*class*);  
System.out.println(jdbcTemplate.update("update student set " + "name='jjf' where id = 1"));

**注：**"update student set " + "name='jjf' where id = 1" **表明mysql语句可以通过字符串拼接合成**

**2.4 查：**

BeanFactory factory = *new ClassPathXmlApplicationContext*("JDBC/Spring\_JDBC/jdbc\_configure/JDBC\_Configure.xml");

JdbcTemplate template = factory.getBean("jdbcTemplate", JdbcTemplate.*class*);  
RowMapper<Student> mapper = *new BeanPropertyRowMapper*<>(Student.*class*); //语句A  
List<Student> student = template.query("select \* from student", mapper); //语句B  
*for* (Student stu : student) {  
 System.out.println(stu);  
}

**注；语句A和语句B运行的结果是，查询到的每一条记录都会被依此被Student对象中，query能查询多条记录，所以返回Student的List链表。**

**原理：对于每一条记录，首先会通过mapper建立一个Student对象，通过该记录的每个字段，会查找Student对象中的所有setter方法，只要字段名称能够和setter方法对应上，则用该记录的字段的值作为实参调用对于的setter方法，如：**

**字段为：myName，则Student如果有setMyname()方法，且参数类型能够对应上，则调用这个方法，但如果找不到方法，不会调用也不会报错。最后将所有记录产生的Student对象以List<Student>的数据结构组合返回。**

**所以，如果要保存记录，就应该在设计类时与数据库表的定义相互吻合；**

**如果只返回一条记录，可以使用如下语句：**

Student student1 = template.queryForObject("select \* from student where id = 2", mapper);  
System.out.println(student);