

实验2 Spring 事务实验

(一) 实验目的

1. 通过本实验了解Spring JDBC模块和Spring事务的作用
2. 通过本实验掌握Spring JDBC模块的配置和Spring事务模块的配置
3. 通过本实验掌握通过JdbcTemplate对数据进行增加
4. 通过service层控制事务边界

(二) 实验内容

1. 新建Web工程
2. 创建applicationContext.xml 文件，引入Spring的约束
3. 分层实现service层和dao层bean
4. 再service的实现层调用dao层，并实现事务的控制
5. 调用JdbcTemplate的数据操作方法
6. 在applicationContext.xml配置数据源和JdbcTemplate，
7. 在applicationContext.xml中配置Dao层组件
8. 在applicationContext.xml中配置Service层组件
9. 访问service层的bean对象

(三) 实验要求

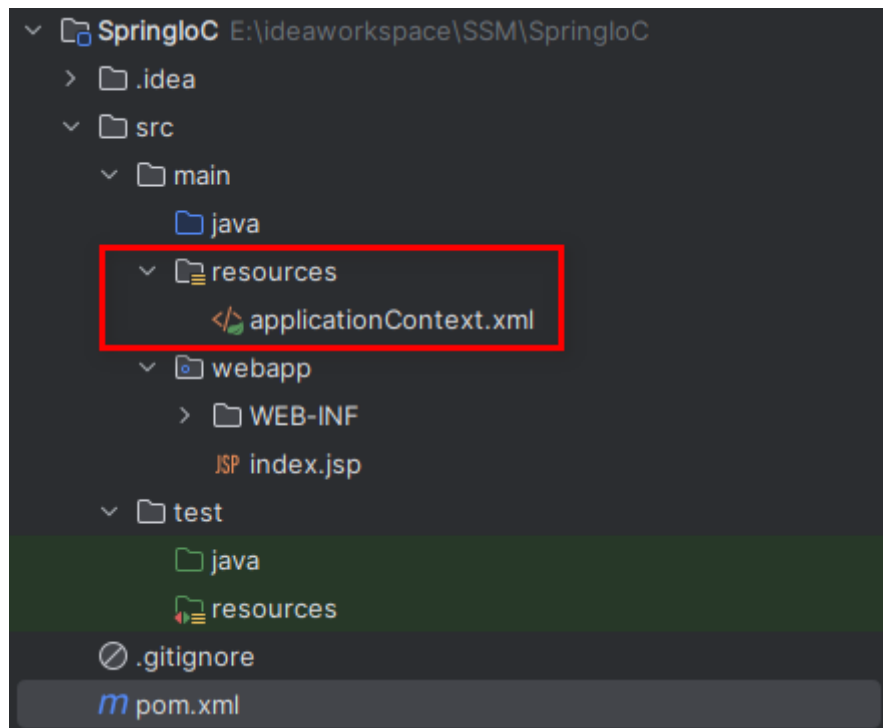
1. 实验过程要求每个同学根据实验指导书认真完成

(四) 实验步骤

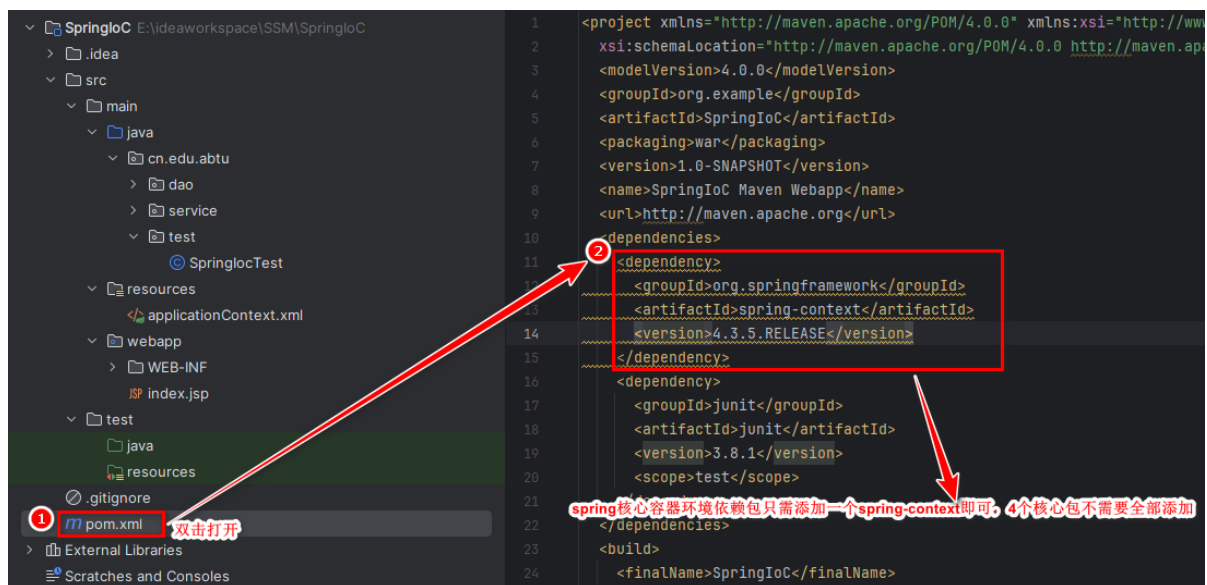
1. 在IDEA中创建一个Maven项目（项目名：spring_trans）

此步骤略，请参考 [IDEA新建Maven项目实验](#)。

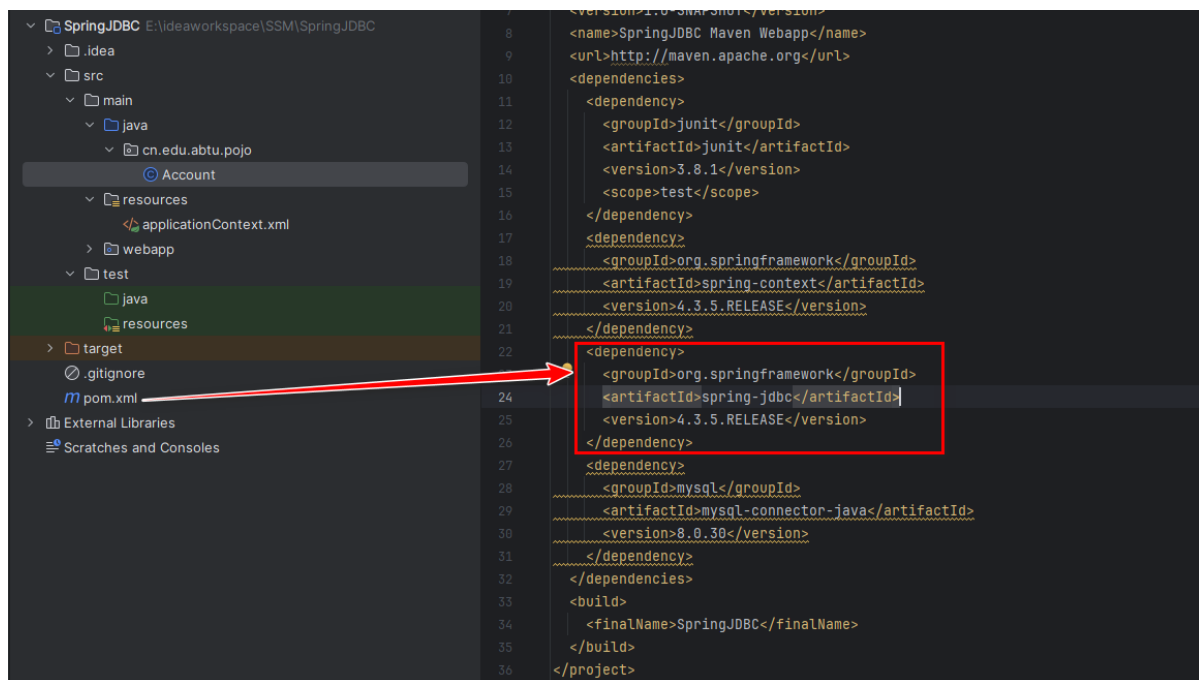
2. 将applicationContext.xml文件引入工程src->main->resources目录下



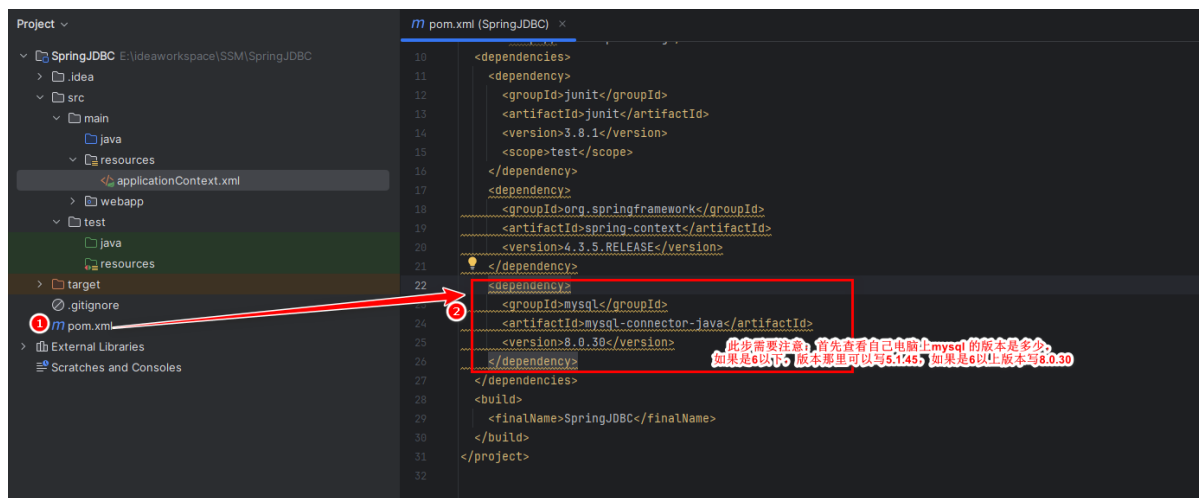
3. 在项目pom.xml文件中引用Spring的核心jar包



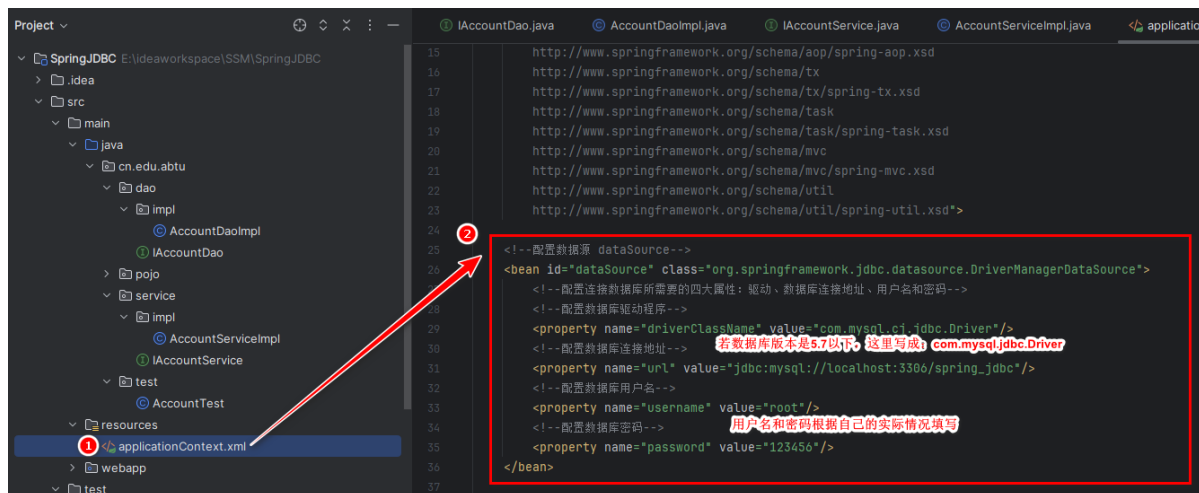
4. 在项目pom.xml文件中引入Spring JDBC模块包



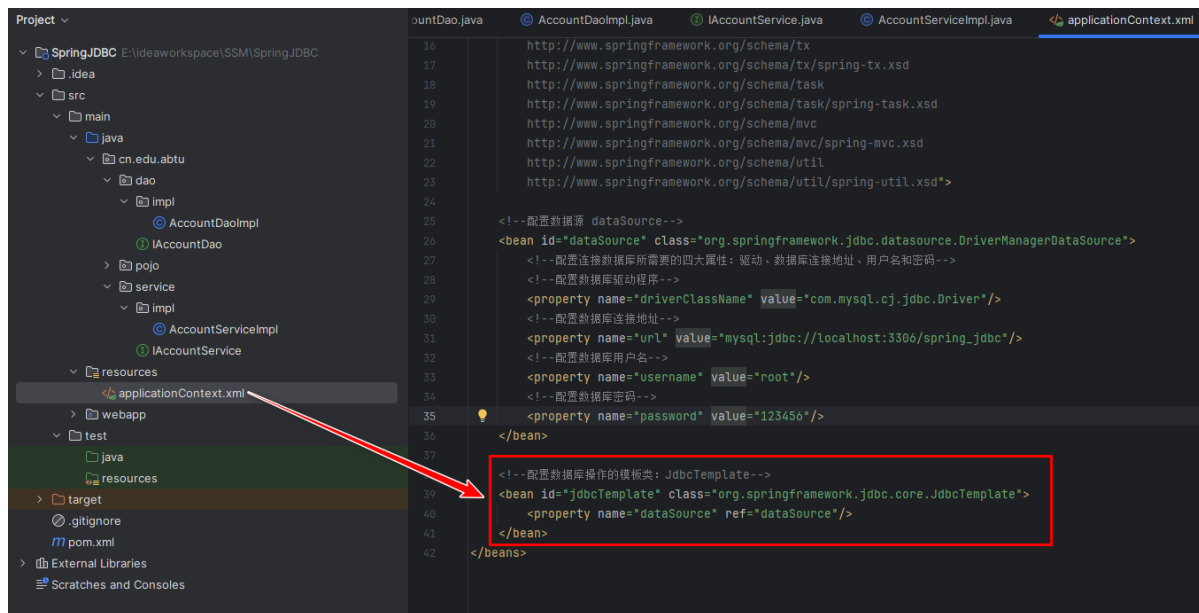
5. 在项目pom.xml文件中引入mysql的驱动包



6. 在applicationContext中配置连接数据库的数据源

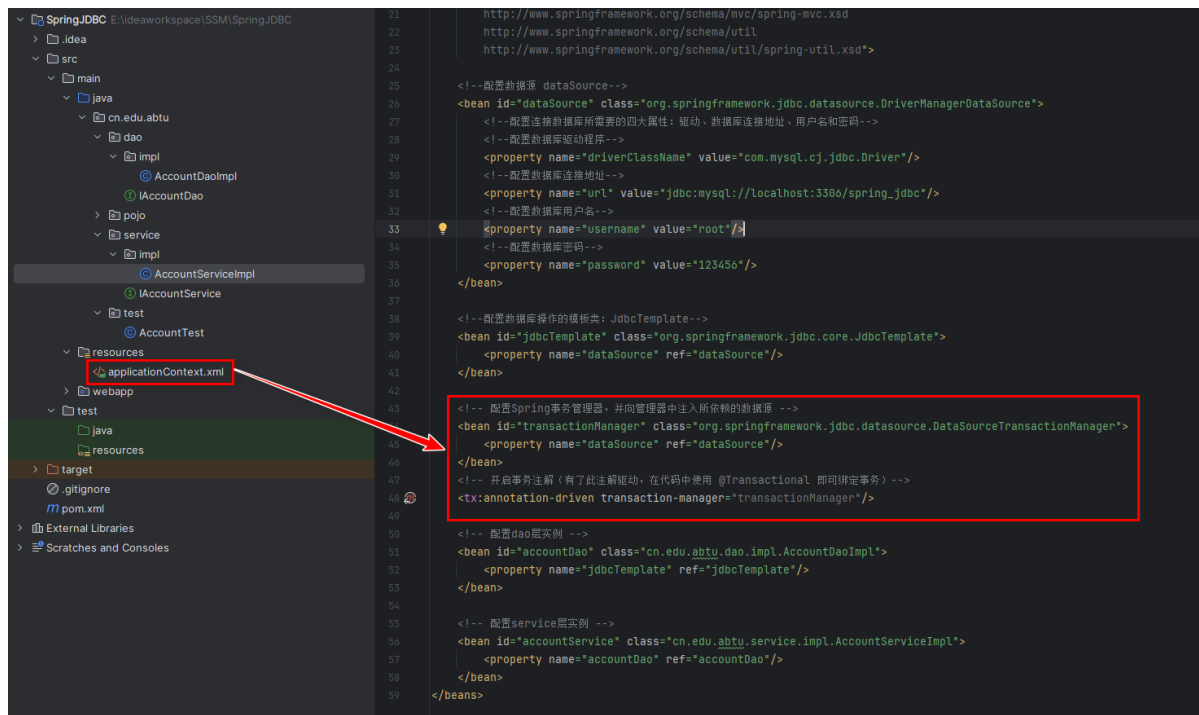


7. 在applicationContext.xml中配置数据库模板类JdbcTemplate



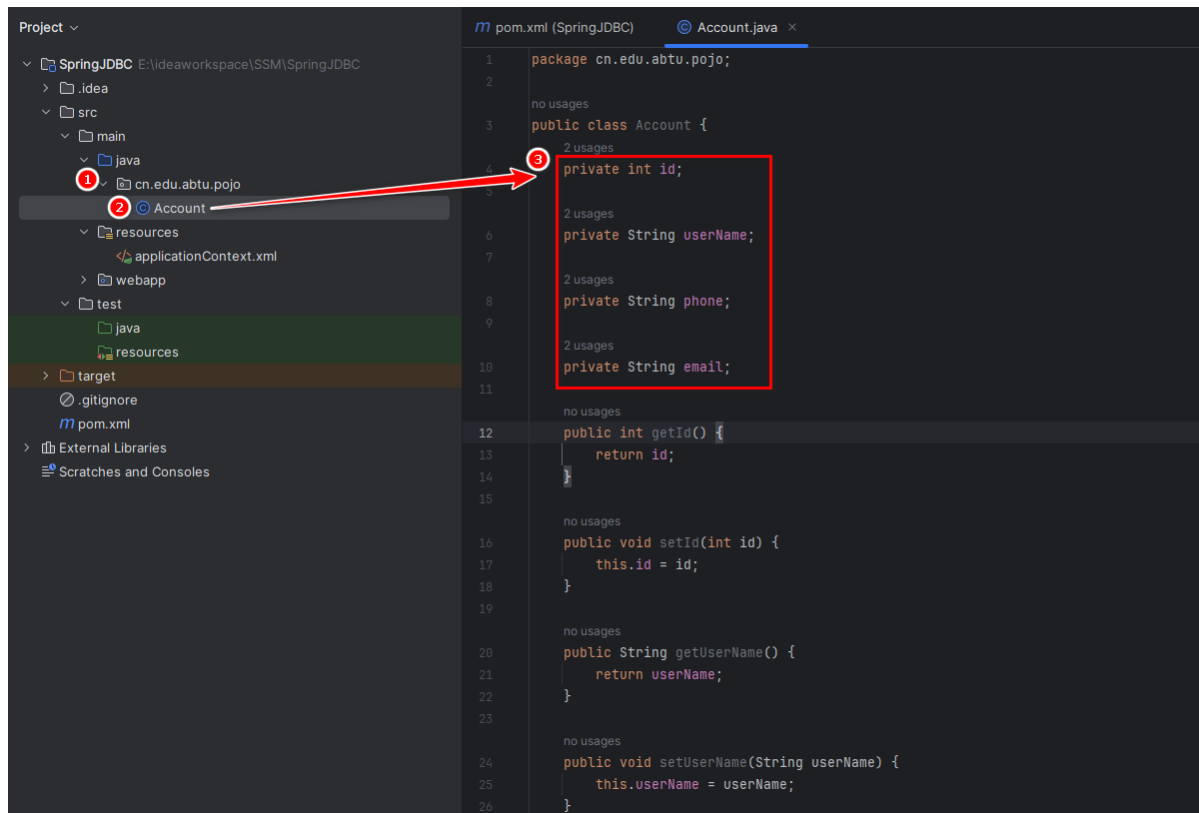
```
16 http://www.springframework.org/schema/tx
17 http://www.springframework.org/schema/tx/spring-tx.xsd
18 http://www.springframework.org/schema/task
19 http://www.springframework.org/schema/task/spring-task.xsd
20 http://www.springframework.org/schema/mvc
21 http://www.springframework.org/schema/mvc/spring-mvc.xsd
22 http://www.springframework.org/schema/util
23 http://www.springframework.org/schema/util/spring-util.xsd>
24
25 <!-- 配置数据源 dataSource -->
26 <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
27 <!-- 配置连接数据库所需要的四大属性：驱动、数据库连接地址、用户名和密码 -->
28 <!-- 配置数据库驱动程序 -->
29 <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
30 <!-- 配置数据库连接地址 -->
31 <property name="url" value="jdbc:mysql://localhost:3306/spring_jdbc"/>
32 <!-- 配置数据库用户名 -->
33 <property name="username" value="root"/>
34 <!-- 配置数据库密码 -->
35 <property name="password" value="123456"/>
36 </bean>
37
38 <!-- 配置数据库操作的模板类：JdbcTemplate -->
39 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
40 <property name="dataSource" ref="dataSource"/>
41 </bean>
42 </beans>
```

8. 在applicationContext.xml中配置事务，并开启事务注解

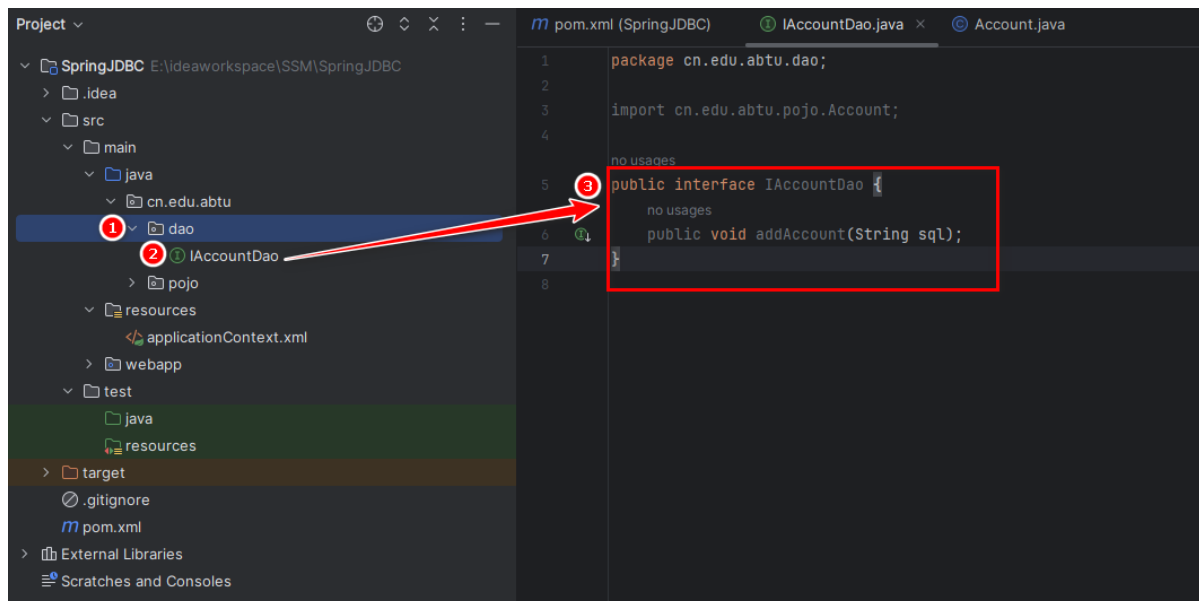


```
21 http://www.springframework.org/schema/mvc/spring-mvc.xsd
22 http://www.springframework.org/schema/util
23 http://www.springframework.org/schema/util/spring-util.xsd>
24
25 <!-- 配置数据源 dataSource -->
26 <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
27 <!-- 配置连接数据库所需要的四大属性：驱动、数据库连接地址、用户名和密码 -->
28 <!-- 配置数据库驱动程序 -->
29 <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
30 <!-- 配置数据库连接地址 -->
31 <property name="url" value="jdbc:mysql://localhost:3306/spring_jdbc"/>
32 <!-- 配置数据库用户名 -->
33 <property name="username" value="root"/>
34 <!-- 配置数据库密码 -->
35 <property name="password" value="123456"/>
36 </bean>
37
38 <!-- 配置数据库操作的模板类：JdbcTemplate -->
39 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
40 <property name="dataSource" ref="dataSource"/>
41 </bean>
42
43 <!-- 配置Spring事务管理器，并向管理器中注入所依赖的数据源 -->
44 <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
45 <property name="dataSource" ref="dataSource"/>
46 </bean>
47 <!-- 开启事务注解（有了此注解驱动，在代码中使用 @Transactional 即可绑定事务） -->
48 <tx:annotation-driven transaction-manager="transactionManager"/>
49
50 <!-- 配置dao层实例 -->
51 <bean id="accountDao" class="cn.edu.abtu.dao.impl.AccountDaoImpl">
52 <property name="jdbcTemplate" ref="jdbcTemplate"/>
53 </bean>
54
55 <!-- 配置service层实例 -->
56 <bean id="accountService" class="cn.edu.abtu.service.impl.AccountServiceImpl">
57 <property name="accountDao" ref="accountDao"/>
58 </bean>
59 </beans>
```

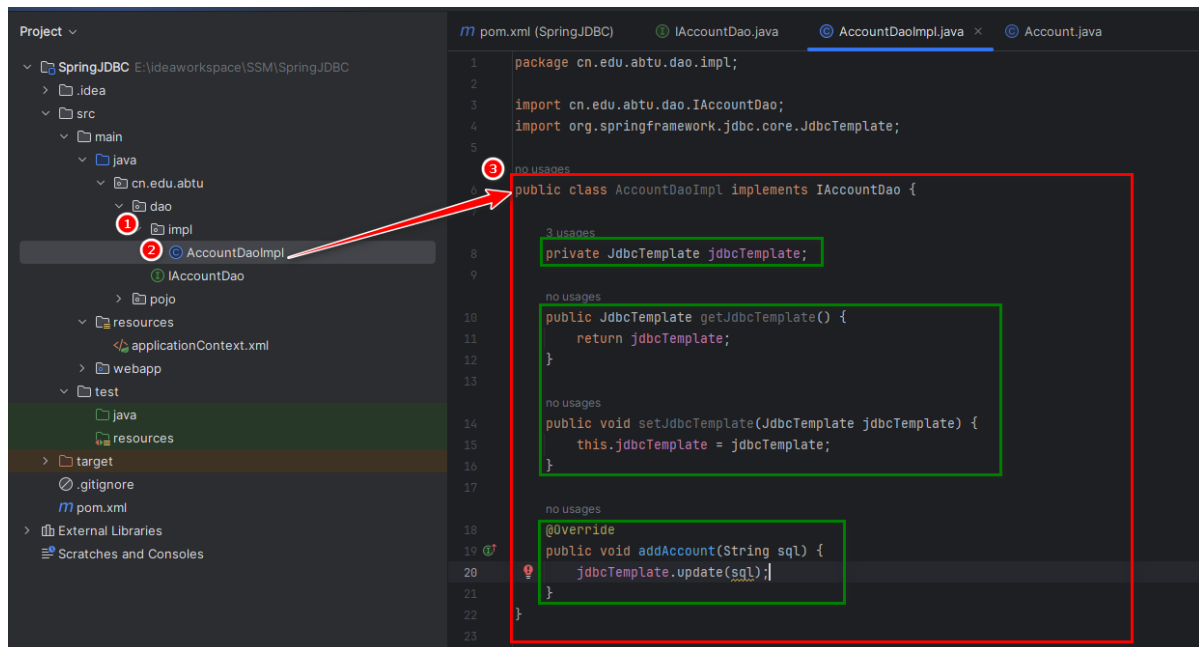
9. 在src->main->java下新建cn.edu.abtu.pojo包，并在包下新建Account类，在类中声明id、userName、phone、email变量，并生成setter/getter方法



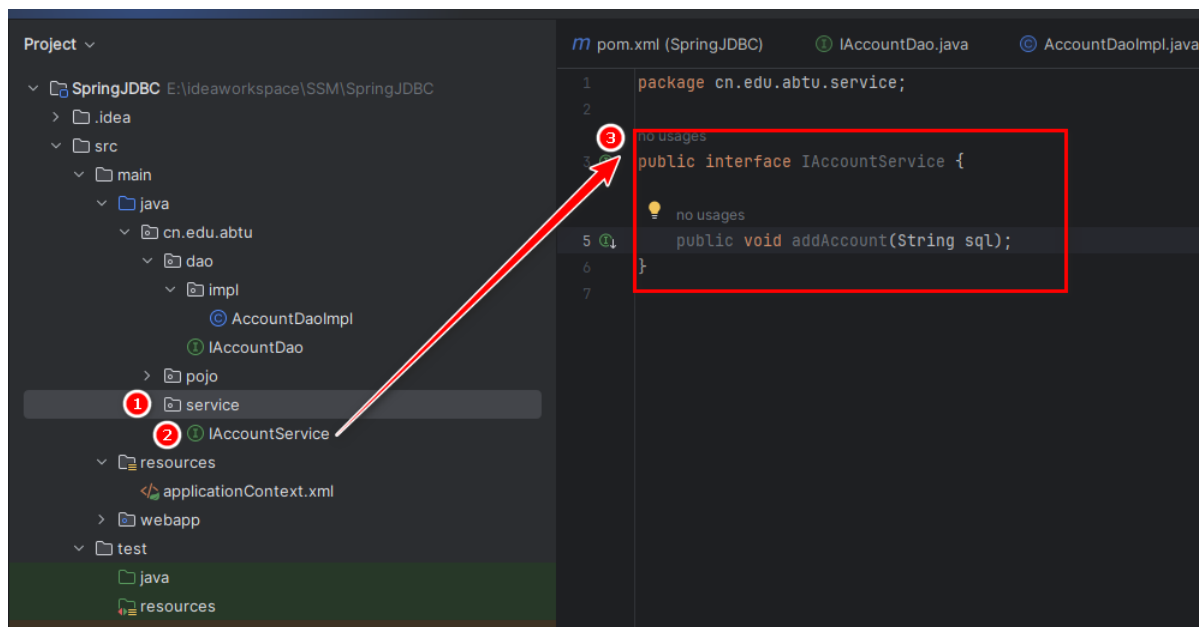
10. 在src->main->java下新建cn.edu.abtu.dao包，并在该包下新建IAccountDao接口，并在接口中声明addAccount(String sql)方法



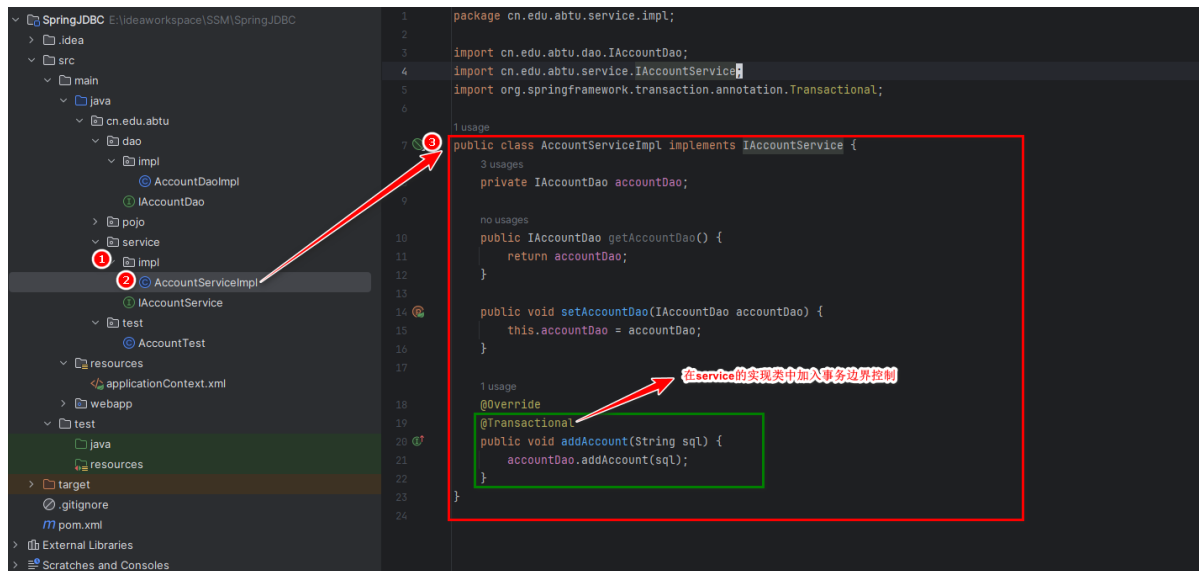
11. 在src->main->java下新建cn.edu.abtu.dao.impl包，新建AccountDaoImpl类，并实现IAccountDao接口。并在类中声明操作数据库的模板类JdbcTemplate的变量，并生成getter和setter方法。在addAccount方法中通过JdbcTemplate向数据库执行sql语句对象。



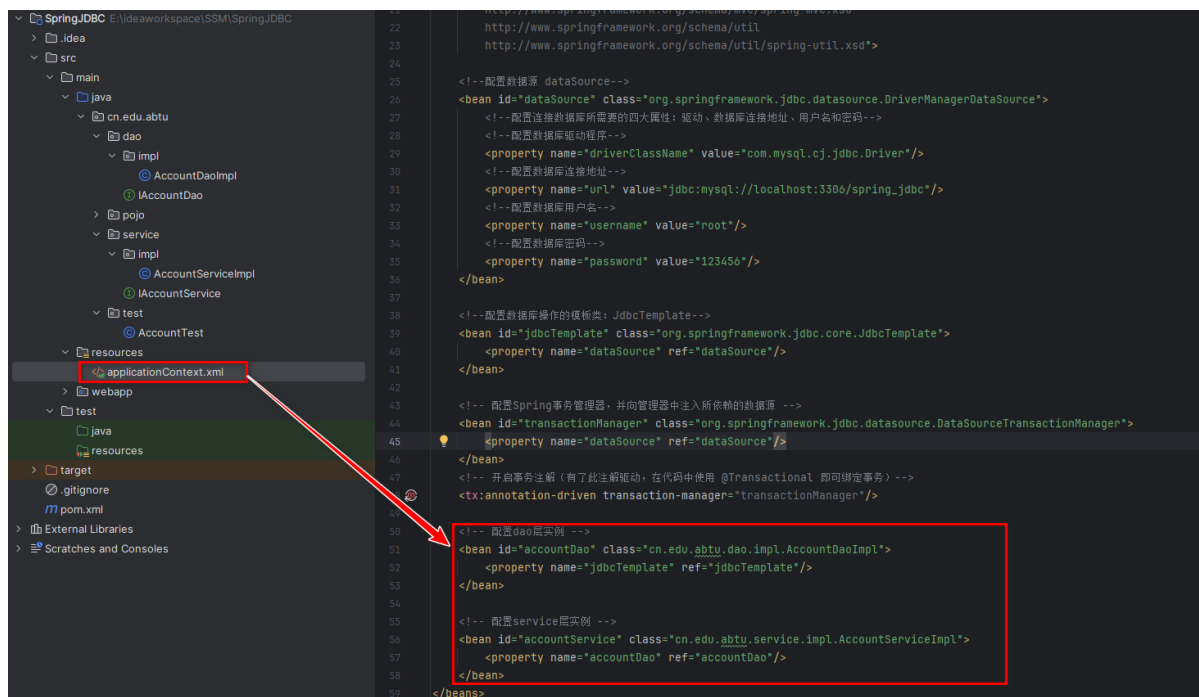
12. 在src->main->java下新建cn.edu.abtu.service包，并新建IAccountService接口，在接口中声明addAccount(String sql)方法。



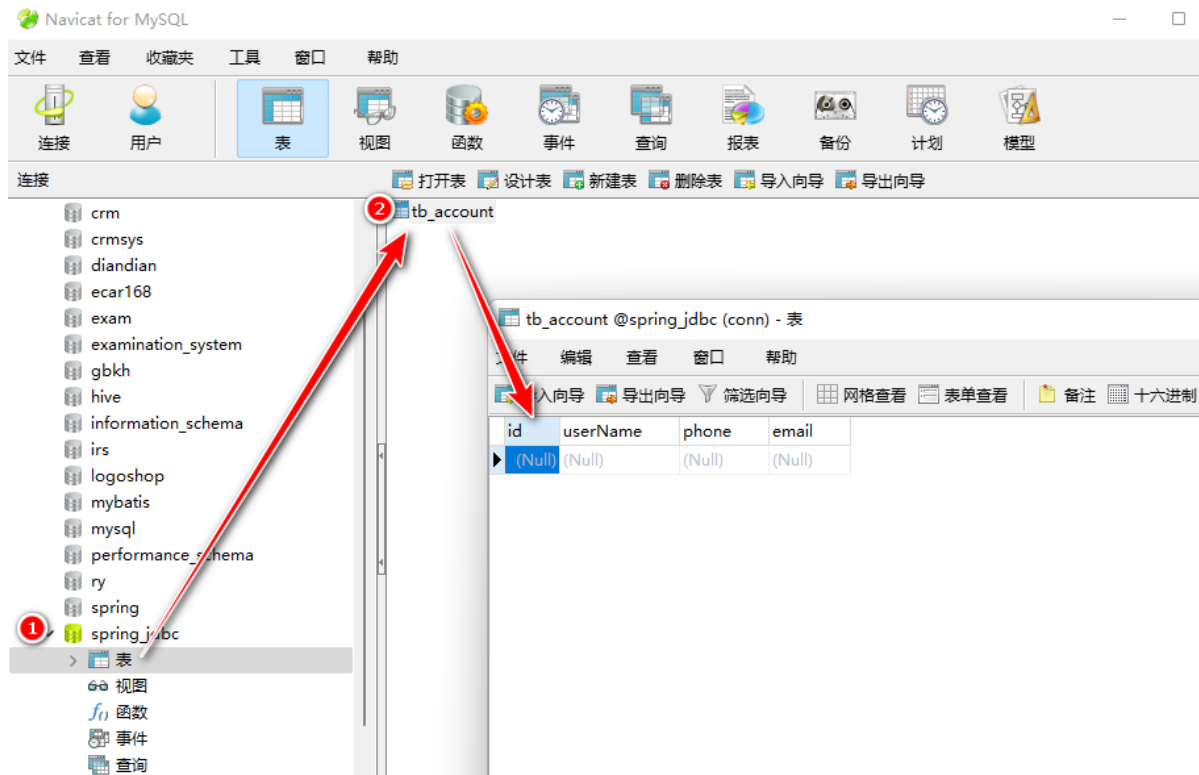
13. 在src->main->java下新建cn.edu.abtu.service.impl包，新建AccountServiceImpl类，并实现IAccountService接口。在该类中声明IAccountDao变量，并生成setter/getter方法。在addAccount方法中调用Dao层的方法，并在addAccount方法上加上@Transactional注解，实现事务控制。



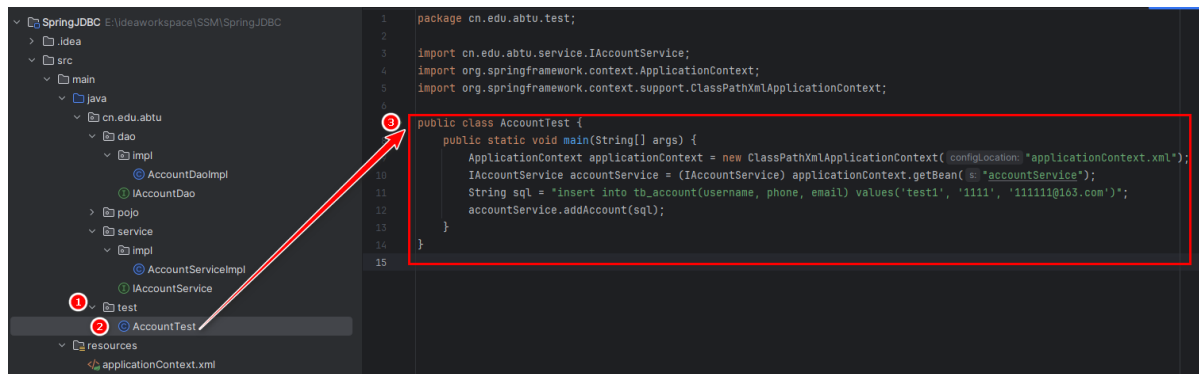
14. 在applicationContext.xml中配置service层和dao层的实例



15. 打开mysql数据库新建数据库spring_jdbc，并在数据库中新建表tb_account，该表的字段有：id (int) ， userName (varchar) 、 phone (varchar) 、 email (varchar)



16. 在src->main->java下新建cn.edu.abtu.test包，并在该包下新建AccountTest测试类，生成main方法，在方法中实现以下代码



17. 运行测试类，并查看数据表，有记录证明插入成功



18. 修改AccountServiceImpl中addAccount方法中的代码，在插入语句后添加异常代码

```
public class AccountServiceImpl implements IAccountService {  
    3 usages  
    private IAccountDao accountDao;  
  
    no usages  
    public IAccountDao getAccountDao() {  
        return accountDao;  
    }  
  
    public void setAccountDao(IAccountDao accountDao) {  
        this.accountDao = accountDao;  
    }  
  
    1 usage  
    @Override  
    @Transactional  
    public void addAccount(String sql) {  
        accountDao.addAccount(sql);  
        int i = 1/0;  
    }  
}
```

19. 执行程序，然后查看数据是否添加成功

此时查看数据库是没有变化的

20. 去掉AccountServiceImpl中addAccount方法中int i=1/0。然后再运行程序。然后查看数据库的变化。

tb_account @spring_jdbc (conn) - 表

文件	编辑	查看	窗口	帮助
导入向导	导出向导	筛选向导	网格查看	表单查看
id	userName	phone	email	
1	test1	1111	111111@163.com	
3	test1	1111	111111@163.com	

观察数据库id值的变化，缺少了一个，证明在上次发生异常时，数据被插入到数据库，id被占用，发生了异常，事务回滚，被占用的id不再显示，从下一个值开始。

(五) 实验总结

通过观察输出结果和自己实验整个过程总结一下自己的学习所得