一.准备工作:

连接数据库:所用的版本-mysql5.0.8

依赖管理pom.xml:spring-context,spring-jdbc,mysql-connector-java(驱动),druid(数据库连接池),junit-jupiter-api(测试类),spring-test

1.在resource中新建Spring-context,来配置jdbc和配置文件

```
<context:component-scan base-package="com.lanou"/>
<!--导入properties文件-->
<context:property-placeholder location="jdbc.properties"/>

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">
  <property name="driverClassName" value="${jdbc.driver}"/>
  <property name="url" value="${jdbc.url}"/>
  <property name="username" value="${jdbc.username}"/>
  <property name="password" value="${jdbc.password}"/>
</bean>
```

2.新建jdbc.properties文件,日志管理

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/db1227
jdbc.username=root
jdbc.password=root
```

3.写数据库操作

```
<!--jdbcTemplate:封装了数据库的操作-->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

二.工程写入:

1.新建POJO类user,写dao层的接口并实现接口.

```

public interface UserDao {
  //数据库查询
```

```
4   //添加用户
5   int insert(String username,String password);
6
7   //通过id,删除用户
8   int deleted(int id);
9
10   //通过id,修改用户密码
11   int mode(String password,int id);
12
13   //通过id,查询某个用户
14   User selectOne(int id);
15   //查询所有用户
16   List<User> selectALl(int id);
17
18   //查询用户个数
19   int selectcount(int id);
20  }
```

2.实现这个接口为:

```
1  @Repository
2  public class UserDaoImpl implements UserDao {
3   //容器会自动创建
4   @Autowired
5   private JdbcTemplate jdbcTemplate;
6
7   public int insert(String username, String password) {
8   return jdbcTemplate.update("insert into user (username, password) values
(?,?);", username, password);
9   }
10
11   public int deleted(int id) {
12   return jdbcTemplate.update("delete from user where id=?", id);
13   }
14
15   public int mode(String password, int id) {
16   return jdbcTemplate.update("update user set password=? where id=?", pas
sword, id);
17   }
18
```

```
19   public User selectOne(int id) {
20   return jdbcTemplate.queryForObject("select * from user where id=?",new BeanPropertyRowMapper<User>(User.class),id);
21   }
22
23   public List<User> selectALl(int id) {
24   return jdbcTemplate.query("select * from user",new BeanPropertyRowMapper<User>(User.class));
25   }
26
27   public int selectcount(int id) {
28   return Optional.ofNullable(jdbcTemplate.queryForObject("select count(id) from user",Integer.class)).orElse(0);
29   }
30 }
```

3.可以通过idea在实现类中直接测试mysql是否写对,直接在实现类中快捷方式,创建测试类,

```
1  //支持Junit5测试框架
2  @ExtendWith(SpringExtension.class)
3  //测试时会自动生成Spring容器
4  @ContextConfiguration("classpath:Spring-context.xml")
5  class UserDaoImplTest {
6   @Autowired
7   private UserDao userDao;
8
9   @org.junit.jupiter.api.Test
10   void seletAll() {
11   //ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("Spring-context.xml");
12   //UserDao userDao = context.getBean(UserDao.class);
13   int row = userDao.insert("薛之谦", "12234");
14   System.out.println(row);
15   }
16
17   @org.junit.jupiter.api.Test
18   void deleted() {
19   int row = userDao.deleted(2);
20   System.out.println(row);
```

```
21   }
22
```

4.新建server层,业务处理层,需要开启事务,配置xml

```
1  <!--配置事务管理-->
2  <bean id="transactionManager" class="org.springframework.jdbc.datasource.
DataSourceTransactionManager">
3   <property name="dataSource" ref="dataSource"/>
4  </bean>
5
6  <!--开启事务注解-->
7  <tx:annotation-driven transaction-manager="transactionManager"/>
```

事务的特征:(原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)、持久性(Durability)):

1.原子性:事务是一个原子操作,由一系列动作组成.事务原子性确保动作要么全不起作用,要么全部不起作用

2.一致性:一旦事务完成,会确保业务处于一致状态,不会出现部分失败的情况

3.隔离性;如果多个事务处理相同数据,事务是隔离的,避免事务损坏

4.持久性:一旦事务完成,无论发生什么系统错误,事务的结果会被写入持久化存储器中

5.写service层的接口,实现这个接口

```
1  //写在容器中
2  //出现异常时,回滚
3  @Transactional(rollbackFor = Exception.class)
4  @Service
5  public class UserseriveImpl implements Userserive {
6   @Override
```

```
7  public User Login(String username, String password) {
8  return null;
9  }
10 }
```