# Spring框架的第四天 -- 拓薪教育

## 第一章：Spring框架的JDBC模板技术

### 1. JDBC模板技术概述

什么模板技术：Spring框架中提供了很多持久层的模板类来简化编程，使用模板类编写程序会变的简单

* template 模板
* 都是Spring框架提供XxxTemplate

提供了JDBC模板，Spring框架提供的

* JdbcTemplate类，Connection 表示连接，管理事务 Statement ResultSet

### 2. JDBC的模板类的使用

jdbc模板类的使用，创建maven工程，引入坐标依赖

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.0.2.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>commons-logging</groupId>  
 <artifactId>commons-logging</artifactId>  
 <version>1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.12</version>  
 </dependency>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 <version>5.0.2.RELEASE</version>  
 </dependency>  
   
 <dependency>  
 <groupId>aopalliance</groupId>  
 <artifactId>aopalliance</artifactId>  
 <version>1.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aspects</artifactId>  
 <version>5.0.2.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.8.13</version>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.6</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>5.0.2.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>5.0.2.RELEASE</version>  
 </dependency>  
  </dependencies>

编写测试代码（自己来new对象的方式）

package cn.tx.test;  
​  
import org.junit.Test;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
public class Demo1 {  
​  
    /\*\*  
     \* 使用new对象方式完成  
     \*/  
    @Test  
    public void run1(){  
        // 创建连接池对象，Spring框架内置了连接池对象  
        DriverManagerDataSource dataSource = new DriverManagerDataSource();  
        // 设置4个参数  
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");  
        dataSource.setUrl("jdbc:mysql:///spring\_db");  
        dataSource.setUsername("root");  
        dataSource.setPassword("root");  
          
        // 提供模板，创建对象  
        JdbcTemplate template = new JdbcTemplate(dataSource);  
        // 完成数据的增删改查  
        template.update("insert into account values (null,?,?)","熊大",1000);  
   }  
​  
}  
​

### 3. 使用Spring框架来管理模板类

刚才编写的代码使用的是new的方式，应该把这些类交给Spring框架来管理。

Spring管理内置的连接池

<?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:aop="http://www.springframework.org/schema/aop"  
       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  
                http://www.springframework.org/schema/aop  
                http://www.springframework.org/schema/aop/spring-aop.xsd">  
​  
    <!--配置连接池-->  
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />  
        <property name="url" value="jdbc:mysql:///spring\_db" />  
        <property name="username" value="root" />  
        <property name="password" value="root" />  
    </bean>  
​  
    <!--配置jdbc模板-->  
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
​  
</beans>

编写测试方法

package cn.tx.test;  
​  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.test.context.ContextConfiguration;  
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
@RunWith(SpringJUnit4ClassRunner.class)  
@ContextConfiguration(value = "classpath:applicationContext\_jdbc.xml")  
public class Demo1\_1 {  
​  
    @Autowired  
    private JdbcTemplate jdbcTemplate;  
​  
    /\*\*  
     \* 测试的方式  
     \*/  
    @Test  
    public void run1(){  
        jdbcTemplate.update("insert into account values (null,?,?)","熊二",500);  
   }  
​  
}  
​

### 4. Spring框架管理开源的连接池

配置开源的连接池，使用Druid开源的连接池，引入坐标如下

<dependency>  
    <groupId>com.alibaba</groupId>  
    <artifactId>druid</artifactId>  
    <version>1.1.10</version>  
</dependency>

将数据库连接的信息配置到属性文件中

   jdbc.driverClassName=com.mysql.jdbc.Driver  
    jdbc.url=jdbc:mysql:///spring\_db  
    jdbc.username=root  
    jdbc.password=root

完成核心配置

<?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:aop="http://www.springframework.org/schema/aop"  
       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  
                http://www.springframework.org/schema/aop  
                http://www.springframework.org/schema/aop/spring-aop.xsd">  
​  
    <!--配置连接池，使用的是Spring框架内置的连接池  
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />  
        <property name="url" value="jdbc:mysql:///spring\_db" />  
        <property name="username" value="root" />  
        <property name="password" value="root" />  
    </bean>  
    -->  
​  
    <!--使用开源连接池  
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />  
        <property name="url" value="jdbc:mysql:///spring\_db" />  
        <property name="username" value="root" />  
        <property name="password" value="root" />  
    </bean>  
    -->  
​  
    <!--加载属性文件  
    <bean id="placeholderConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
        <property name="location" value="classpath:jdbc.properties" />  
    </bean>  
    -->  
​  
    <!--第二种写法：使用提供标签的方式-->  
    <context:property-placeholder location="classpath:jdbc.properties" />  
​  
    <!--加载属性的文件-->  
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  
        <property name="driverClassName" value="${jdbc.driverClassName}" />  
        <property name="url" value="${jdbc.url}" />  
        <property name="username" value="${jdbc.username}" />  
        <property name="password" value="${jdbc.password}" />  
    </bean>  
​  
    <!--配置jdbc模板-->  
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
​  
</beans>

### 5. Spring框架的JDBC模板的简单操作

增删改查代码编写

package cn.tx.test;  
​  
import cn.tx.demo1.Account;  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.jdbc.core.RowMapper;  
import org.springframework.test.context.ContextConfiguration;  
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  
​  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.List;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
@RunWith(SpringJUnit4ClassRunner.class)  
@ContextConfiguration(value = "classpath:applicationContext\_jdbc.xml")  
public class Demo1\_1 {  
​  
    @Autowired  
    private JdbcTemplate jdbcTemplate;  
​  
    /\*\*  
     \* 测试的方式  
     \*/  
    @Test  
    public void run1(){  
        jdbcTemplate.update("insert into account values (null,?,?)","熊四",800);  
   }  
​  
    /\*\*  
     \* 修改  
     \*/  
    @Test  
    public void run2(){  
        jdbcTemplate.update("update account set name = ?,money = ? where id = ?","光头强",100,7);  
   }  
​  
    /\*\*  
     \* 删除  
     \*/  
    @Test  
    public void run3(){  
        jdbcTemplate.update("delete from account where id = ?",7);  
   }  
​  
    /\*\*  
     \* 通过id查询  
     \*/  
    @Test  
    public void run4(){  
        Account account = jdbcTemplate.queryForObject("select \* from account where id = ?", new BeanMapper(), 6);  
        System.out.println(account);  
   }  
​  
    /\*\*  
     \* 查询所有的数据  
     \*/  
    @Test  
    public void run5(){  
        List<Account> list = jdbcTemplate.query("select \* from account", new BeanMapper());  
        for (Account account : list) {  
            System.out.println(account);  
       }  
   }  
​  
}  
​  
/\*\*  
 \* 实现类，用来进行数据封装的  
 \*/  
class BeanMapper implements RowMapper<Account>{  
​  
    /\*\*  
     \* 是一行一行进行数据封装的  
     \* @param resultSet  
     \* @param i  
     \* @return  
     \* @throws SQLException  
     \*/  
    @Override  
    public Account mapRow(ResultSet resultSet, int i) throws SQLException {  
        Account account = new Account();  
        account.setId(resultSet.getInt("id"));  
        account.setName(resultSet.getString("name"));  
        account.setMoney(resultSet.getDouble("money"));  
        return account;  
   }  
​  
}

## 第二章：模拟转账开发

### 1. 完成转账代码的编写

搭建开发的环境

service代码的编写

package cn.tx.demo2;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
public interface AccountService {  
​  
    /\*\*  
     \* 转账的方法  
     \* @param out       付款人  
     \* @param in       收款人  
     \* @param money     金额  
     \*/  
    public void pay(String out,String in,double money);  
​  
}  
​  
​  
package cn.tx.demo2;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
public class AccountServiceImpl implements AccountService {  
​  
    private AccountDao accountDao;  
    public void setAccountDao(AccountDao accountDao) {  
        this.accountDao = accountDao;  
   }  
​  
    /\*\*  
     \* 转账方法  
     \* @param out       付款人  
     \* @param in       收款人  
     \* @param money     金额  
     \*/  
    @Override  
    public void pay(String out, String in, double money) {  
        // 调用dao方法  
        accountDao.outMoney(out,money);  
​  
        accountDao.inMoney(in,money);  
   }  
​  
}

dao代码的编写

package cn.tx.demo2;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
public interface AccountDao {  
​  
    /\*\*  
     \* 付款  
     \* @param out  
     \* @param money  
     \*/  
    public void outMoney(String out,double money);  
​  
    /\*\*  
     \* 收款  
     \* @param in  
     \* @param money  
     \*/  
    public void inMoney(String in,double money);  
​  
}  
​  
​  
package cn.tx.demo2;  
​  
import org.springframework.jdbc.core.JdbcTemplate;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
public class AccountDaoImpl implements AccountDao {  
​  
    private JdbcTemplate jdbcTemplate;  
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {  
        this.jdbcTemplate = jdbcTemplate;  
   }  
​  
    /\*\*  
     \* 付款  
     \* @param out  
     \* @param money  
     \*/  
    @Override  
    public void outMoney(String out, double money) {  
        jdbcTemplate.update("update account set money = money - ? where name = ?",money,out);  
   }  
​  
    /\*\*  
     \* 收款  
     \* @param in  
     \* @param money  
     \*/  
    @Override  
    public void inMoney(String in, double money) {  
        jdbcTemplate.update("update account set money = money + ? where name = ?",money,in);  
   }  
​  
}  
​

配置文件代码编写

<?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:aop="http://www.springframework.org/schema/aop"  
       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  
                http://www.springframework.org/schema/aop  
                http://www.springframework.org/schema/aop/spring-aop.xsd">  
​  
    <!--第二种写法：使用提供标签的方式-->  
    <context:property-placeholder location="classpath:jdbc.properties" />  
​  
    <!--加载属性的文件-->  
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  
        <property name="driverClassName" value="${jdbc.driverClassName}" />  
        <property name="url" value="${jdbc.url}" />  
        <property name="username" value="${jdbc.username}" />  
        <property name="password" value="${jdbc.password}" />  
    </bean>  
​  
    <!--配置Jdbc模板类-->  
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
​  
    <!--配置service-->  
    <bean id="accountService" class="cn.tx.demo2.AccountServiceImpl">  
        <property name="accountDao" ref="accountDao"/>  
     </bean>  
​  
    <!--配置service-->  
    <bean id="accountDao" class="cn.tx.demo2.AccountDaoImpl">  
        <property name="jdbcTemplate" ref="jdbcTemplate" />  
    </bean>  
​  
</beans>

测试代码编写

package cn.tx.test;  
​  
import cn.tx.demo2.AccountService;  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.test.context.ContextConfiguration;  
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
@RunWith(SpringJUnit4ClassRunner.class)  
@ContextConfiguration(value = "classpath:applicationContext\_dao1.xml")  
public class Demo2 {  
​  
    @Autowired  
    private AccountService accountService;  
​  
    /\*\*  
     \* 测试转账的方法  
     \*/  
    @Test  
    public void testPay(){  
        accountService.pay("熊大","熊二",100);  
   }  
​  
}  
​

### 2. Dao编写的方式（第二种方式）

service编写

package cn.tx.demo3;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
public class AccountServiceImpl implements AccountService {  
​  
    private AccountDao accountDao;  
    public void setAccountDao(AccountDao accountDao) {  
        this.accountDao = accountDao;  
   }  
​  
    /\*\*  
     \* 转账方法  
     \* @param out       付款人  
     \* @param in       收款人  
     \* @param money     金额  
     \*/  
    @Override  
    public void pay(String out, String in, double money) {  
        // 调用dao方法  
        accountDao.outMoney(out,money);  
​  
        accountDao.inMoney(in,money);  
   }  
​  
}  
​

dao编写

package cn.tx.demo3;  
​  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.jdbc.core.support.JdbcDaoSupport;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
public class AccountDaoImpl extends JdbcDaoSupport implements AccountDao {  
​  
    /\*\*  
     \* 付款  
     \* @param out  
     \* @param money  
     \*/  
    @Override  
    public void outMoney(String out, double money) {  
        this.getJdbcTemplate().update("update account set money = money - ? where name = ?",money,out);  
   }  
​  
    /\*\*  
     \* 收款  
     \* @param in  
     \* @param money  
     \*/  
    @Override  
    public void inMoney(String in, double money) {  
        this.getJdbcTemplate().update("update account set money = money + ? where name = ?",money,in);  
   }  
​  
}  
​

配置文件编写

<?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:aop="http://www.springframework.org/schema/aop"  
       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  
                http://www.springframework.org/schema/aop  
                http://www.springframework.org/schema/aop/spring-aop.xsd">  
​  
    <!--第二种写法：使用提供标签的方式-->  
    <context:property-placeholder location="classpath:jdbc.properties" />  
​  
    <!--加载属性的文件-->  
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  
        <property name="driverClassName" value="${jdbc.driverClassName}" />  
        <property name="url" value="${jdbc.url}" />  
        <property name="username" value="${jdbc.username}" />  
        <property name="password" value="${jdbc.password}" />  
    </bean>  
​  
    <!--配置Jdbc模板类  
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
    -->  
​  
    <!--配置service-->  
    <bean id="accountService" class="cn.tx.demo3.AccountServiceImpl">  
        <property name="accountDao" ref="accountDao"/>  
     </bean>  
​  
    <!--配置dao  
    <bean id="accountDao" class="cn.tx.demo3.AccountDaoImpl">  
        <property name="jdbcTemplate" ref="jdbcTemplate" />  
    </bean>  
    -->  
​  
    <bean id="accountDao" class="cn.tx.demo3.AccountDaoImpl">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
​  
</beans>

测试方法

package cn.tx.test;  
​  
import cn.tx.demo3.AccountService;  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.test.context.ContextConfiguration;  
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
@RunWith(SpringJUnit4ClassRunner.class)  
@ContextConfiguration(value = "classpath:applicationContext\_dao2.xml")  
public class Demo3 {  
​  
    @Autowired  
    private AccountService accountService;  
​  
    /\*\*  
     \* 测试转账的方法  
     \*/  
    @Test  
    public void testPay(){  
        accountService.pay("熊大","熊二",100);  
   }  
​  
}  
​

## 第三章：Spring框架的事务管理

### 1. Spring框架的事务管理相关的类和API

PlatformTransactionManager接口

​ 平台事务管理器。该接口有具体的实现类，根据不同的持久层框架，需要选择不同的实现类！

​ 接口方法如下：

​ void commit(TransactionStatus status)

​ void rollback(TransactionStatus status)

​ 如果使用的Spring的JDBC模板或者MyBatis框架，需要选择DataSourceTransactionManager实现类

​ 如果使用的是Hibernate的框架，需要选择HibernateTransactionManager实现类

TransactionDefinition接口，事务定义信息接口

​ 定义了事务隔离级别

​ 定义了事务传播行为

### 2. Spring框架声明式事务管理

配置文件的方式

<?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:aop="http://www.springframework.org/schema/aop"  
       xmlns:tx="http://www.springframework.org/schema/tx"  
       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  
                http://www.springframework.org/schema/aop  
                http://www.springframework.org/schema/aop/spring-aop.xsd  
                http://www.springframework.org/schema/tx  
            http://www.springframework.org/schema/tx/spring-tx.xsd">  
​  
    <!--第二种写法：使用提供标签的方式-->  
    <context:property-placeholder location="classpath:jdbc.properties" />  
​  
    <!--加载属性的文件-->  
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  
        <property name="driverClassName" value="${jdbc.driverClassName}" />  
        <property name="url" value="${jdbc.url}" />  
        <property name="username" value="${jdbc.username}" />  
        <property name="password" value="${jdbc.password}" />  
    </bean>  
​  
    <!--配置平台事务管理器-->  
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
​  
    <!--配置事务的通知（没有自己编写切面类，通知方法也不是自己编写，Spring框架提供的）-->  
    <tx:advice id="txAdvice" transaction-manager="transactionManager">  
        <tx:attributes>  
            <!--对pay进行增强，设置隔离级别，传播行为，超时的时间-->  
            <tx:method name="pay" isolation="DEFAULT" propagation="REQUIRED" />  
            <tx:method name="find\*" read-only="true" />  
        </tx:attributes>  
    </tx:advice>  
​  
    <!--配置AOP的增强-->  
    <aop:config>  
        <!--Spring框架提供系统通知，使用advisor标签-->  
        <aop:advisor advice-ref="txAdvice" pointcut="execution(public \* cn.tx.demo4.AccountServiceImpl.pay(..))" />  
    </aop:config>  
​  
    <!--配置Jdbc模板类  
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
    -->  
​  
    <!--配置service-->  
    <bean id="accountService" class="cn.tx.demo4.AccountServiceImpl">  
        <property name="accountDao" ref="accountDao"/>  
     </bean>  
​  
    <!--配置dao  
    <bean id="accountDao" class="cn.tx.demo3.AccountDaoImpl">  
        <property name="jdbcTemplate" ref="jdbcTemplate" />  
    </bean>  
    -->  
   
    <bean id="accountDao" class="cn.tx.demo4.AccountDaoImpl">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
   
</beans>

配置文件+注解的方式

<?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:aop="http://www.springframework.org/schema/aop"  
       xmlns:tx="http://www.springframework.org/schema/tx"  
       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  
                http://www.springframework.org/schema/aop  
                http://www.springframework.org/schema/aop/spring-aop.xsd  
                http://www.springframework.org/schema/tx  
            http://www.springframework.org/schema/tx/spring-tx.xsd">  
​  
    <!--开启注解的扫描-->  
    <context:component-scan base-package="cn.tx.demo5" />  
​  
    <!--第二种写法：使用提供标签的方式-->  
    <context:property-placeholder location="classpath:jdbc.properties" />  
​  
    <!--加载属性的文件-->  
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">  
        <property name="driverClassName" value="${jdbc.driverClassName}" />  
        <property name="url" value="${jdbc.url}" />  
        <property name="username" value="${jdbc.username}" />  
        <property name="password" value="${jdbc.password}" />  
    </bean>  
​  
    <!--配置平台事务管理器-->  
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
​  
    <!--配置Jdbc模板类-->  
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
​  
    <!--开启事务注解的支持-->  
    <tx:annotation-driven transaction-manager="transactionManager" />  
​  
</beans>

service代码

package cn.tx.demo5;  
​  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Isolation;  
import org.springframework.transaction.annotation.Transactional;  
​  
/\*\*  
 \* 拓薪教育 -- 腾讯课程认证机构  
 \* 樱木老师  
 \*/  
@Service  
@Transactional(isolation = Isolation.DEFAULT)  
public class AccountServiceImpl implements AccountService {  
​  
    @Autowired  
    private AccountDao accountDao;  
​  
    /\*\*  
     \* 转账方法  
     \* @param out       付款人  
     \* @param in       收款人  
     \* @param money     金额  
     \*/  
    @Override  
    public void pay(String out, String in, double money) {  
        // 调用dao方法  
        accountDao.outMoney(out,money);  
​  
        // 模拟异常  
        int a = 1/0;  
​  
        accountDao.inMoney(in,money);  
   }  
​  
}  
​

纯注解的方式

package cn.tx.demo6;  
   
 import javax.annotation.Resource;  
 import javax.sql.DataSource;  
   
 import org.springframework.context.annotation.Bean;  
 import org.springframework.context.annotation.ComponentScan;  
 import org.springframework.context.annotation.Configuration;  
 import org.springframework.jdbc.core.JdbcTemplate;  
 import org.springframework.jdbc.datasource.DataSourceTransactionManager;  
 import org.springframework.transaction.PlatformTransactionManager;  
 import org.springframework.transaction.annotation.EnableTransactionManagement;  
   
 import org.springframework.jdbc.datasource.DriverManagerDataSource;  
   
 /\*\*  
 \* 配置类  
 \* @author Administrator  
 \*/  
 @Configuration  
 @ComponentScan(basePackages="cn.tx.demo6")  
 @EnableTransactionManagement // 开启事务注解  
 public class SpringConfig {  
   
 /\*\*  
 \* @return  
 \* @throws Exception   
 \*/  
 @Bean(name="dataSource")  
 public DataSource createDataSource() throws Exception{  
 // 创建连接池对象，Spring框架内置了连接池对象  
       DriverManagerDataSource dataSource = new DriverManagerDataSource();  
       // 设置4个参数  
       dataSource.setDriverClassName("com.mysql.jdbc.Driver");  
       dataSource.setUrl("jdbc:mysql:///spring\_db");  
       dataSource.setUsername("root");  
       dataSource.setPassword("root");  
 return dataSource;  
 }  
   
 /\*\*  
 \* 创建模板对象  
 \* @return  
 \*/  
 @Resource(name="dataSource") // 不仅可以作用在属性上，也可以作用方法上。  
 @Bean(name="jdbcTemplate") // 把JdbcTemplate保存到IOC容器中  
 public JdbcTemplate createJdbcTemplate(DataSource dataSource){  
 JdbcTemplate template = new JdbcTemplate(dataSource);  
 return template;  
 }  
   
 /\*\*  
 \* 创建平台事务管理器对象  
 \* @param dataSource  
 \* @return  
 \*/  
 @Resource(name="dataSource")  
 @Bean(name="transactionManager")  
 public PlatformTransactionManager createTransactionManager(DataSource dataSource){  
 DataSourceTransactionManager manager = new DataSourceTransactionManager(dataSource);  
 return manager;  
 }  
   
 }