# ­­实验七 AOP的实现

## 实验目的

1、了解AOP的概念和作用；

2、理解AOP中的相关术语；

3、了解Spring中两种动态代理方式的区别；

4、掌握基于XML和注解的AspectJ开发。

## 实验内容

创建Java项目，定义Bank类，类中封装账户ID、账户姓名、账户余额三个属性。用三层架构实现银行业务处理：存钱、取钱、转账。以两个账户的存钱、取钱和转账为例，当账户发生银行业务时，系统日志打印交易类型，业务结束后，系统日志打印账户姓名和余额消息，日志打印用切面类的增强处理。

## 实验代码：

*<?*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/aop  
 http://www.springframework.org/schema/aop/spring-aop.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd"*>  
  
 <*context:component-scan base-package="com.shf.sy7"*/>  
  
 <*aop:aspectj-autoproxy*/>  
</*beans*>*

package com.shf.sy7.pojo;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import java.math.BigDecimal;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class Bank *{* private Integer Id;  
 private String name;  
 private Double balance;  
*}*

package com.shf.sy7.dao;  
  
import com.shf.sy7.pojo.Bank;  
  
import java.util.List;  
  
  
public interface BankDao *{* Bank Deposit*(*Bank bankFrom, Bank bankTo, Double money*)*;  
 Bank Withdraw*(*Bank bankFrom, Bank bankTo, Double money*)*;  
 List*<*Bank*>* Transfer*(*Bank bankFrom, Bank bankTo, Double money*)*;  
*}*

package com.shf.sy7.dao;  
  
import com.shf.sy7.pojo.Bank;  
import org.springframework.stereotype.Repository;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
@Repository*(*"BankDao"*)*public class BankDaoImpl implements BankDao *{* @Override  
 public Bank Deposit*(*Bank bankFrom, Bank bankTo, Double money*) {* bankFrom.setBalance*(*bankFrom.getBalance*()*+money*)*;  
 return bankFrom;  
 *}* @Override  
 public Bank Withdraw*(*Bank bankFrom, Bank bankTo, Double money*) {* bankFrom.setBalance*(*bankFrom.getBalance*()*-money*)*;  
 return bankFrom;  
 *}* @Override  
 public List*<*Bank*>* Transfer*(*Bank bankFrom, Bank bankTo, Double money*) {* bankFrom.setBalance*(*bankFrom.getBalance*()*-money*)*;  
 bankTo.setBalance*(*bankTo.getBalance*()*+money*)*;  
 ArrayList*<*Bank*>* banks = new ArrayList*<>()*;  
 banks.add*(*bankFrom*)*;  
 banks.add*(*bankTo*)*;  
 return banks;  
 *}  
}*

package com.shf.sy7.service;  
  
import com.shf.sy7.pojo.Bank;  
  
import java.util.List;  
  
public interface BankService {  
 Bank Deposit(Bank bankFrom, Bank bankTo, Double money);  
 Bank Withdraw(Bank bankFrom, Bank bankTo, Double money);  
 List<Bank> Transfer(Bank bankFrom, Bank bankTo, Double money);  
}

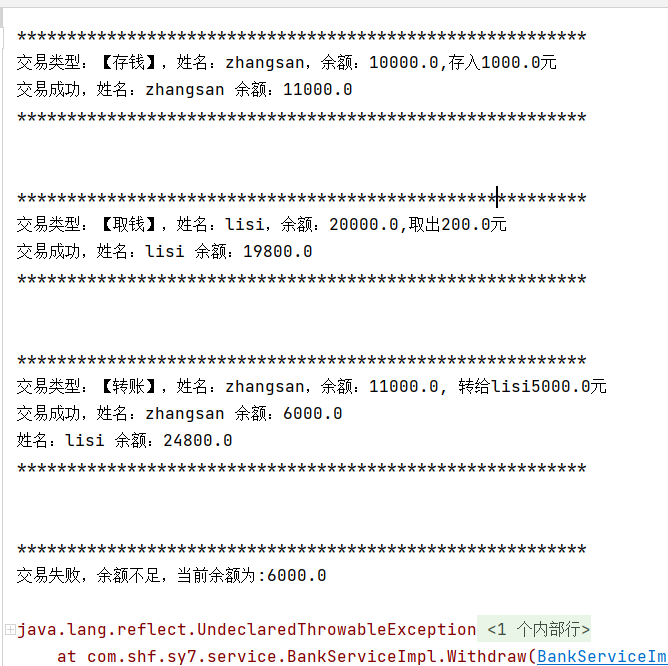
package com.shf.sy7.service;  
  
import com.shf.sy7.dao.BankDao;  
import com.shf.sy7.pojo.Bank;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
*/\*\*  
 \* @author shuho  
 \*/*@Service*(*"BankService"*)*public class BankServiceImpl implements BankService *{* @Autowired  
 private BankDao bankDao;  
  
  
 @Override  
 public Bank Deposit*(*Bank bankFrom, Bank bankTo, Double money*) {* return bankDao.Deposit*(*bankFrom,null,money*)*;  
 *}* @Override  
 public Bank Withdraw*(*Bank bankFrom, Bank bankTo, Double money*) {* return bankDao.Withdraw*(*bankFrom,null,money*)*;  
 *}* @Override  
 public List*<*Bank*>* Transfer*(*Bank bankFrom, Bank bankTo, Double money*) {* return bankDao.Transfer*(*bankFrom, bankTo, money*)*;  
 *}  
}*

package com.shf.sy7.exception;  
  
public class MyException extends Exception*{* public MyException*(){  
  
 }* public MyException*(*String msg*){* super*(*msg*)*;  
 System.*out*.println*(*msg*)*;  
 *}  
}*

package com.shf.sy7.aop;  
  
import com.shf.sy7.exception.MyException;  
import com.shf.sy7.pojo.Bank;  
import org.aspectj.lang.JoinPoint;  
import org.aspectj.lang.annotation.\*;  
import org.springframework.stereotype.Component;  
  
import java.util.List;  
  
@Component  
@Aspect  
public class BankAdvice *{  
 //切点* @Pointcut*(*value = "execution( \* com.shf.sy7.dao.BankDaoImpl.\*(..))"*)* public void poincut*(){  
 }  
  
 //前置通知 系统日志打印交易类型* @Before*(*value = "poincut()&&args(from,to,mon)"*)* public void before*(*JoinPoint joinPoint,Object from,Object to,Object mon*)* throws MyException *{* String name = joinPoint.getSignature*()*.getName*()*;  
 Bank bankFrom= *(*Bank*)* from;  
 Bank bankTo= *(*Bank*)* to;  
 Double money = *(*Double*)* mon;  
 System.*out*.println*(*"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"*)*;  
  
*// 判断余额是否充足* if *((*name=="Withdraw" || name=="Transfer"*)* && *(*bankFrom.getBalance*()*-money<0*)){* throw new MyException*(*"交易失败，余额不足，当前余额为:"+bankFrom.getBalance*())*;  
 *}* if *(*name=="Deposit"*){* System.*out*.println*(*"交易类型：【存钱】，姓名："+bankFrom.getName*()*+"，余额："+bankFrom.getBalance*()*+",存入"+money+"元"*)*;  
 *}* else if *(*name=="Withdraw"*){* System.*out*.println*(*"交易类型：【取钱】，姓名："+bankFrom.getName*()*+"，余额："+bankFrom.getBalance*()*+",取出"+money+"元"*)*;  
 *}* else if*(*name=="Transfer"*){* System.*out*.println*(*"交易类型：【转账】，姓名："+bankFrom.getName*()*+"，余额："+bankFrom.getBalance*()*+", 转给"+bankTo.getName*()*+ money +"元"*)*;  
 *}  
  
 }  
  
 //返回通知 系统日志打印账户姓名和余额消息* @AfterReturning*(*value = "poincut()",returning = "result"*)* public void afterReturning*(*JoinPoint joinPoint,Object result*){* String name = joinPoint.getSignature*()*.getName*()*;  
 if *(*name=="Transfer"*){* List*<*Bank*>* bankList = *(*List*<*Bank*>)* result;  
 System.*out*.println*(*"交易成功，姓名："+bankList.get*(*0*)*.getName*()*+" 余额："+bankList.get*(*0*)*.getBalance*())*;  
 System.*out*.println*(*"姓名："+bankList.get*(*1*)*.getName*()*+" 余额："+bankList.get*(*1*)*.getBalance*())*;  
 *}* else *{* Bank bank = *(*Bank*)* result;  
 System.*out*.println*(*"交易成功，姓名："+bank.getName*()*+" 余额："+bank.getBalance*())*;  
 *}* System.*out*.println*(*"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"*)*;  
 System.*out*.println*()*;  
 System.*out*.println*()*;  
 *}  
  
}*

package com.shf.sy7.test;  
  
import com.shf.sy7.dao.BankDao;  
import com.shf.sy7.pojo.Bank;  
import com.shf.sy7.service.BankService;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.testng.annotations.Test;  
  
import java.math.BigDecimal;  
  
public class BankTest *{* @Test  
 public void bankTest1*(){* Bank zhangsanBank = new Bank*(*1, "zhangsan", 10000.0*)*;  
 Bank lisiBank = new Bank*(*2, "lisi", 20000.0*)*;  
  
 ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext*(*"applicationContext.xml"*)*;  
 BankService bankService = *(*BankService*)* context.getBean*(*"BankService"*)*;  
  
 bankService.Deposit*(*zhangsanBank, null,1000.0*)*;  
  
 bankService.Withdraw*(*lisiBank,null,200.0*)*;  
  
 bankService.Transfer*(*zhangsanBank,lisiBank,5000.0*)*;  
  
 bankService.Withdraw*(*zhangsanBank, null,10000.0*)*;  
 *}  
}*

## 运行截图



## 实验小结

通过本次实验我学会使用spring中AOP面向切面编程的动态代理，动态代理的作用是在不改变源代码的情况下，增加功能。掌握了前置通知 @Before，后置通知 @AfterReturning，环绕通知 @Around，异常通知 @AfterThrowing，最终通知 @After。