5、声明式事务

笔记本: spring

创建时间: 2022/4/3 17:03

作者: 雷丰阳

环境搭建:

1、导入sql文件、导入jar

2、写几个类和方法模拟结账操作;

```
package com.atguigu.dao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
@Repository
public class BookDao {
   @Autowired
   JdbcTemplate jdbcTemplate;
    * 1、减余额
    * 减去某个用户的余额
   public void updateBalance(String userName,int price){
       String sql = "UPDATE## account SET balance=balance-? WHERE
username=?";
       jdbcTemplate.update(sql, price,userName);
    * 2、按照图书的ISBN获取某本图书的价格
    * @return
   public int getPrice(String isbn){
       String sql = "SELECT price FROM book WHERE isbn=?";
       return jdbcTemplate.queryForObject(sql, Integer.class, isbn);
   }
    * 3、减库存;减去某本书的库存;为了简单期间每次减一
   public void updateStock(String isbn){
       String sql = "UPDATE book stock SET stock=stock-1 WHERE isbn=?";
       jdbcTemplate.update(sql, isbn);
   }
}
```

```
package com.atguigu.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.atguigu.dao.BookDao;
@Service
public class BookService {
```

```
@Autowired
    BookDao bookDao;
     * 结账; 传入哪个用户买了哪本书
     * @param username
     * @param isbn
    public void checkout(String username, String isbn){
        bookDao.updateStock(isbn);
        int price = bookDao.getPrice(isbn);
        //2、减余额
        bookDao.updateBalance(username, price);
    }
 }
声明式事务:
  以前通过复杂的编程来编写一个事务,替换为只需要告诉Spring哪个方法是事务方法即
  Spring自动进行事务控制;
编程式事务:
  TransactionFilter{
       try{
          //获取连接
          //设置非自动 提交
         chain.doFilter();
         //提交
       }catch(Exception e){
         //回滚
       }finllay{
         //关闭连接释放资源
  }
AOP: 环绕通知可以去做;
```

```
AOP:环绕通知可以去做;
//获取连接
//设置非自动 提交
目标代码执行
//正常提交
//异常回滚
//最终关闭
```

最终效果:

BookService{

事务管理代码的<mark>固定模式</mark>作为一种<mark>横切关注点</mark>,可以通过AOP方法模块化,进而借助 Spring AOP框架实现声明式事务管理。

自己要写这个切面还是很麻烦; 这个切面已经有了; (事务切面===事务管理器);

```
PlatformTransactionManager org.springframework.transaction

    CciLocalTransactionManager - org.springframework.jca.cci.connection

        DataSourceTransactionManager org.springframework.jdbc.datasource
        G HibernateTransactionManager - org.springframework.orm hibernate4
        G HibernateTransactionManager org.springframework.orm hibernate3
        JdoTransactionManager - org.springframework.orm.jdo

    JpaTransactionManager org.springframework.orm.jpa

    • JtaTransactionManager - org.springframework.transaction.jta

    WebLogicJtaTransactionManager - org.springframework.transaction.jta

    WebSphereUowTransactionManager - org.springframework.transaction.jta

▼ 10 CallbackPreferringPlatformTransactionManager - org.springframework.transaction.support

    WebSphereUowTransactionManager - org.springframework.transaction.jta

    ResourceTransactionManager - org.springframework.transaction.support

    CciLocalTransactionManager - org.springframework.jca.cci.connection

    DataSourceTransactionManager - org.springframework.jdbc.datasource

    HibernateTransactionManager - org.springframework.orm.hibernate4

        HibernateTransactionManager - org.springframework.orm.hibernate3

    JdoTransactionManager - org.springframework.orm.jdo

    JpaTransactionManager - org.springframework.orm.jpa
```

这个事务管理器就可以在目标方法运行前后进行事务控制(事务切面); 我们目前都使用DataSourceTransactionManager;即可;

快速的为某个方法添加事务:

- 1) 、配置出这个事务管理器让他工作;
- 2) 、开启基于注解的事务
- 3) 、给事务方法加@Transactional注解

```
<!-- 配置数据源-->
   <bean id="pooledDataSource"</pre>
class="com.mchange.v2.c3p0.ComboPooledDataSource">
       cproperty name="user" value="${jdbc.user}"></property>
       cproperty name="driverClass" value="${jdbc.driverClass}">
   </bean>
   <!-- 配置JdbcTemplate -->
   <bean class="org.springframework.jdbc.core.JdbcTemplate">
       cproperty name="dataSource" ref="pooledDataSource"></property>
   </bean>
   <!-- 事务控制 -->
   <!--1:配置事务管理器(切面)让其进行事务控制;一定导入面向切面编程的几个包
          spring-aspects-4.0.0.RELEASE.jar
          com.springsource.net.sf.cglib-2.2.0.jar
          com.springsource.org.aopalliance-1.0.0.jar
          com.springsource.org.aspectj.weaver-1.6.8.RELEASE.jar
   -->
   <bean id="tm"</pre>
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
       <!-- 控制住数据源 -->
       cproperty name="dataSource" ref="pooledDataSource"></property>
   </bean>
   <!--2:开启基于注解的事务控制模式; 依赖tx名称空间 -->
   <tx:annotation-driven transaction-manager="tm"/>
   <!--3:给事务方法加注解@Transactional -->
```