[**spring注解式事务管理实例**](http://www.cnblogs.com/zqrferrari/archive/2010/08/06/1794400.html)

一.spring配置文件

这里使用spring命名空间，如下：

<?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:tx="<http://www.springframework.org/schema/tx>"  
 xsi:schemaLocation="<http://www.springframework.org/schema/beans>  
    <http://www.springframework.org/schema/beans/spring-beans-2.5.xsd>  
    <http://www.springframework.org/schema/context>  
    <http://www.springframework.org/schema/context/spring-context-2.5.xsd>  
    <http://www.springframework.org/schema/tx>  
    <http://www.springframework.org/schema/tx/spring-tx-2.5.xsd>"  
    default-autowire="byName"  default-lazy-init="true">

 <context:property-placeholder location="classpath:jdbc.properties" />

 <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"  
  destroy-method="close">  
  <property name="driverClassName" value="${jdbc.driverClassName}" />  
  <property name="url" value="${jdbc.url}" />  
  <property name="username" value="${jdbc.username}" />  
  <property name="password" value="${jdbc.password}" />  
  <property name="initialSize" value="${jdbc.initialSize}" />  
  <property name="maxActive" value="${jdbc.maxActive}" />  
  <property name="maxIdle" value="${jdbc.maxIdle}" />  
  <property name="minIdle" value="${jdbc.minIdle}" />  
  <property name="maxWait" value="${jdbc.maxWait}" />  
 </bean>

 <!-- 设定transactionManager -->  
 <bean id="transactionManager"  
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
  <property name="dataSource" ref="dataSource" />  
 </bean>

<!--启动spring注解功能-->

 <tx:annotation-driven transaction-manager="transactionManager" />

</beans>

说明：

1.如果事务管理器的id是transactionManager，这里可以不对transaction-manager进行配置，即<tx:annotation-driven />就可以。

2.这个配置是告诉spring在类（接口）层面或者方法层面上检查应用程序上下文中的所有标准了@Transactional的bean，spring将自动把事务通知的内容通知给它。

3.这个通知的事务参数将由@Transactional注释的参数来定义。

4.如果注释的是接口，则该接口的所有实现类都将被事务化。

二.使用@Transactional标注bean

package com.netqin.bbs.initUserData;

import org.springframework.transaction.annotation.Propagation;  
import org.springframework.transaction.annotation.Transactional;

import com.netqin.bbs.initUserData.service.InitUserDataService;  
import com.netqin.bbs.utils.Constant;  
import com.netqin.bbs.utils.ReadFileUtil;

@Transactional(propagation=Propagation.SUPPORTS,readOnly=true)//设置默认的事务管理策略，即没有标注@Transactional的方法的事务处理方式，意思为不要求方法必须在一个事务中运行  
public class InitUserData {  
 /\*\*                                                            
 \* 描述 :使用方法上声明的事务管理策略，这里的意思为需要在一个事务中运行. <br>  
 \*<p>                                                   
                                                                                                                                                                                                                                                                                                  
 \*/  
 @Transactional(propagation=Propagation.REQUIRED,readOnly=false)  
 public void method1(){

  ……………………

  ……………………      
    
 }

 /\*\*                                                            
 \* 描述 : 使用默认策略. <br>  
 \*<p>                                                   
                                                                                                                                                                                                                                                                                                  
 \*/

 public void method2(){

  ……………………

  ……………………      
    
 }

}

记得要将这个bean加入到spring上下文中。

一般来说，上述两种事务策略就可以满足要求了，不过需要注意，注解功能的使用需要在项目中加入cglib-nodep-x.x\_x.jar