业务层 service

事务 transaction

Commit rollback

需要用面向切面

第一步：导入事务的命名空间

xmlns:tx=[*http://www.springframework.org/schema/tx*](http://www.springframework.org/schema/tx)

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-3.1.xsd*

第二部：定义事务管理器

Spring为hibernate提供了事务管理类*HibernateTransactionManager*

<!-- 定义事务管理器 -->

<bean id=*"txManager"* class=*"org.springframework.orm.hibernate3.HibernateTransactionManager"*>

<property name=*"sessionFactory"* ref=*"sessionFactory"* />

</bean>

第三部：定义事务增强并指定事务管理器

<!-- 定义事务增强并指定事务管理器 -->

<tx:advice id=*"txAdvice"* transaction-manager=*"txManager"*>

<!-- 定义属性，声明事务规则 -->

<tx:attributes>

<tx:method name=*"get\*"* read-only=*"true"*/>

<tx:method name=*"update\*"* propagation=*"REQUIRED"*/>

<tx:method name=*"delete\*"* propagation=*"REQUIRED"*/>

<tx:method name=*"add\*"* propagation=*"REQUIRED"*/>

<tx:method name=*"\*"* propagation=*"REQUIRED"* read-only=*"true"*/>

</tx:attributes>

</tx:advice>

Read-only:true 当为查询时，可以提高查询效率

Propagation：事务的传播机制

propagation=*"REQUIRED" 如果存在一个事务则支持当前事务，如果不存在则开启一个新的事务*

*第四部：定义切面*

<!-- 定义切面 -->

<aop:config>

<aop:pointcut id=*"serviceMethod"* expression=*"execution(\* dao..\*.\*(..))"*/>

<aop:advisor advice-ref=*"txAdvice"* pointcut-ref=*"serviceMethod"*/>

</aop:config>

Propagation：事务的传播机制

REQUIRED: *如果存在一个事务则支持当前事务，如果不存在则开启一个新的事务*

Dao1

Dao2

用同一个事务

后面几乎用不到

SUPPORTS:如果存在一个事务，支持当前事务，没有食物，按照非事务方式执行

MANDATORY:如果存在一个事务支持当前事务，如果当前没有一个活动的事务，则抛出异常

REQUIRES\_NEW:总是开启一个新事务

会有2个事务的

NOT\_SUPPORTED:总是以非事务方式执行

NEVER:总是以非事务方式执行，如果当前存在一个活动事务则抛出异常

NESTED:如果存在一个活动事务，则创建一个事务作为当前事务的嵌套事务运行，如果没有当前事务，与REQUIRED相同

事务隔离等级:isolation

DEFAULT:默认值，表示使用数据库默认的事务隔离级别

READ\_UNCOMMITTED:未提交读

READ\_COMMITTED:提交读

REPEATABLE\_READ:可重复读

SERIALIZABLE:串行读

自己建立2个test 表,然后添加,当有一个不成功, 就事务回滚

Aop 切面就事务切面就切入 了

Read\_only 提高了查询效率