Spring 事务详解

# Spring事务管理

## Spring事务管理机制 三个核心组件

PlatformTransactionManager 平台事务管理器 ：commit 提交事务,rollback   
回滚事务 , TransactionStatus getTransaction(TransactionDefinition definition) 获取事务（只提供接口，具体有实现类来完成。）

TransactionDefinition 事务定义信息（和事务管理相关一些参数）：隔离级别、传播级别、超时 、事务是否只读 （静态信息）

TransactionStatus 事务状态 ：（事务运行过程中 一些动态信息 ）

关系： 事务管理参数信息由 TransactionDefinition 加载， Spring事务管理由PlatformTransactionManager 完成， PlatformTransactionManager 管理事务过程中根据 TransactionDefinition 定义事务管理信息来管理事务 ，在事务运行过程中不同时刻 事务状态信息不同， TransactionStatus 就表示不同时刻事务状态信息

## 事务管理器 PlatformTransactionManager 接口

org.springframework.jdbc.datasource.DataSourceTransactionManager ------- 针对jdbc开发或者 iBatis开发 进行事务管理

org.springframework.orm.hibernate3.HibernateTransactionManager ------- 针对Hibernate3开发 进行事务管理

## 事务定义信息 TransactionDefinition

### 隔离级别

|  |
| --- |
| 1) **read\_uncommitted** 读未提交 ：引发所有事务隔离问题（脏读、不可重复读、虚读）  2) **read\_committed** 读已提交 不会发生脏读，会导致不可重复读和虚读  3) **repeatable\_read** 重复读 不会发生 脏读和不可重复读，会导致虚读  4) **serializable**  序列化 采用串行方式管理事务，同一时间，只能有一个事务在操作 ，阻止所有隔离问题发生  DEFAULT 采取数据库默认隔离级别： Oracle read\_committed、 Mysql repeatable\_read |

### 事务传播行为

|  |
| --- |
| 1) PROPAGATION\_REQUIRED : 支持当前事务，如果不存在 就新建一个 , 将 deleteUser 和 removeAllOrders 放入同一个事务 (Spring默认提供事务传播级别)  2) PROPAGATION\_SUPPORTS : 支持当前事务，如果不存在，就不使用事务  3) PROPAGATION\_MANDATORY : 支持当前事务，如果不存在，抛出异常  4) PROPAGATION\_REQUIRES\_NEW : 如果有事务存在，挂起当前事务，创建一个新的事务 , deleteUser和removeAllOrder 两个事务执行，无任何关系  5) PROPAGATION\_NOT\_SUPPORTED : 以非事务方式运行，如果有事务存在，挂起当前事务  6) PROPAGATION\_NEVER : 以非事务方式运行，如果有事务存在，抛出异常  7) PROPAGATION\_NESTED : 如果当前事务存在，则嵌套事务执行 |

# 声明式事务管理 （使用注解配置 ）

## spring配置

|  |
| --- |
| <context:component-scan base-package="com.imooc" />  <aop:aspectj-autoproxy />  <bean id="propertyConfigurer"  class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  <property name="locations">  <list>  <value>classpath:env/env.properties</value>  <value>classpath:app-config.properties</value>  </list>  </property>  </bean>  <!-- 1.配置资源datesource -->  <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource">  <property name="driverClassName" value="${jdbc.driverClass}" />  <property name="url" value="${jdbc.url}" />  <property name="username" value="${jdbc.user}" />  <property name="password" value="${jdbc.password}" />  <property name="initialSize" value="${connection\_pools.initial\_pool\_size}" />  <property name="maxTotal" value="${connection\_pools.max\_pool\_size}" />  <property name="maxIdle" value="${connection\_pools.max\_idle\_size}" />  <property name="validationQuery" value="select 1" />  </bean>  <!-- bean后处理器，告诉spring处理DAO组件中的@PersistenceContext注解 -->  <bean  class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor" />  <!-- 2.entityManagerFactory配置 -->  <!-- 定义实体管理器工厂 Jpa配置 LocalContainerEntityManagerFactoryBean这个选项Spring扮演了容器的角色。完全掌管JPA -->  <bean id="entityManagerFactory"  class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">  <!-- 指定数据源 -->  <property name="jtaDataSource" ref="dataSource" />  <!-- 指定Jpa持久化实现厂商类,这里以Hibernate为例 -->  <property name="jpaVendorAdapter">  <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">  <property name="database" value="MYSQL" />  </bean>  </property>  <!-- 指定Entity实体类包路径 -->  <property name="packagesToScan" value="com.imooc.entity" />  <!-- 指定JPA属性；如Hibernate中指定是否显示SQL的是否显示、方言等 -->  <property name="jpaPropertyMap">  <map>  <entry key="hibernate.show\_sql" value="${hibernate.show\_sql}" />  <entry key="hibernate.format\_sql" value="false" />  <entry key="hibernate.use\_sql\_comments" value="false" />  <entry key="javax.persistence.transactionType" value="RESOURCE\_LOCAL" />  <entry key="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />  <!-- <entry key="javax.persistence.transactionType" value="JTA" /> -->  <!-- <entry key="hibernate.current\_session\_context\_class" value="jta"  /> -->  <!-- <entry key="hibernate.transaction.factory\_class" -->  <!-- value="org.hibernate.engine.transaction.internal.jta.CMTTransactionFactory"  /> -->  <!-- <entry key="hibernate.transaction.jta.platform" -->  <!-- value="org.hibernate.engine.transaction.jta.platform.internal.JBossAppServerJtaPlatform"  /> -->  </map>  </property>  </bean>  <!-- Jpa 事务管理器 -->  <bean id="jpaTransactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">  <property name="entityManagerFactory" ref="entityManagerFactory" />  </bean>  <!-- 重要配置：启用扫描并自动创建代理的功能 -->  <jpa:repositories base-package="com.imooc"  transaction-manager-ref="jpaTransactionManager"  entity-manager-factory-ref="entityManagerFactory">  <!-- <repository:include-filter type="regex" -->  <!-- expression=".\*\.[^.]\*Repository" /> -->  <repository:include-filter type="assignable"  expression="org.springframework.data.repository.Repository" />  </jpa:repositories>  <!--3 采用声明式管理事务 -->  <tx:annotation-driven transaction-manager="jpaTransactionManager" /> |

## spring事务使用方法

|  |
| --- |
| 到对需要管理事务的方法，添加注解@Transactionnal  @Transactional(propagation=Propagation.REQUIRED,isolation=Isolation.DEFAULT,readOnly=false,rollbackFor=ArithmeticException.class)  说明：  propagation=Propagation.REQUIRED **传播行为**  isolation=Isolation.DEFAULT **隔离级别**  readOnly=false **是否只读**  rollbackFor=ArithmeticException.class **发生异常回滚**  noRollbackFor=xxx.class **发生异常 仍然提交事务** |
| **@Transactional**  @Override  public void delDepartment(Department department) {  departmentDao.delete(department);    } |