Spring事务

1. 什么是事务

事务由一条或多条操作数据库的SQL语句组成的一个不可分割的工作单元，这些操作要么全部执行成功，要么全部失败；

1.1. 事务的特性

1. **原子性（Atomicity）**

   事务中不可分割的一个单元，事务中的全部操作要么全部成功，要么全部失败；

2. **一致性（Consistency）**

   表示事务完成时，所有的数据必须保持一致；

3. **隔离性（Isolation）**

   一个事务的执行不受其他事务的影响，也就是一个事务的内部操作和使用的数据对并发的其它事务是隔离的，并发执行的各个事务之间不能相互影响；

4. **持久性（Durability）**

   也称为永久性，指一个事务一旦提交，对数据库中数据的修改应该是永久性的；

1.2. 事务的隔离级别

1. **为什么要有事务的隔离?**

   数据库中的数据有可能同时被多个用户访问，当多个用户操作相同数据的时候，可能会出现并发问题，下面的几个并发问题是事务的隔离出现的原因：

   1. **脏读**
   2. **幻读**
   3. **不可重复读**

2. **隔离级别有哪些**

   1. **未提交读（Read Uncommitted）**

      > 该隔离级别表示一个事务可以读取另一个事务修改但还没有提交的数据。
      > 该级别不能防止脏读、不可重复读和幻读，因此很少使用该隔离级别。

   2. **已提交读（Read Commited）**

该隔离级别表示一个事务只能读取另一个事务已经提交的数据。
该级别可以防止脏读，这也是大多数情况下的推荐值。

3. **可重复读**

该隔离级别表示一个事务在整个过程中可以多次重复执行某个查询，并且每次返回的记录都相同。
该级别可以防止不可重复读。

4. **串行化（Serializable ）**

所有的事务依次逐个执行，这样事务之间就完全不可能产生干扰；
也就是说，该级别可以防止脏读、不可重复读以及幻读。
但是这将严重影响程序的性能。通常情况下也不会用到该级别。

总结

隔离级别	解决脏读	解决不可重复读	解决幻读
Read Uncommitted	不会	不会	不会
Read Commited	会	不会	不会
Repeatable read	会	会	不会
Serializable	会	会	会

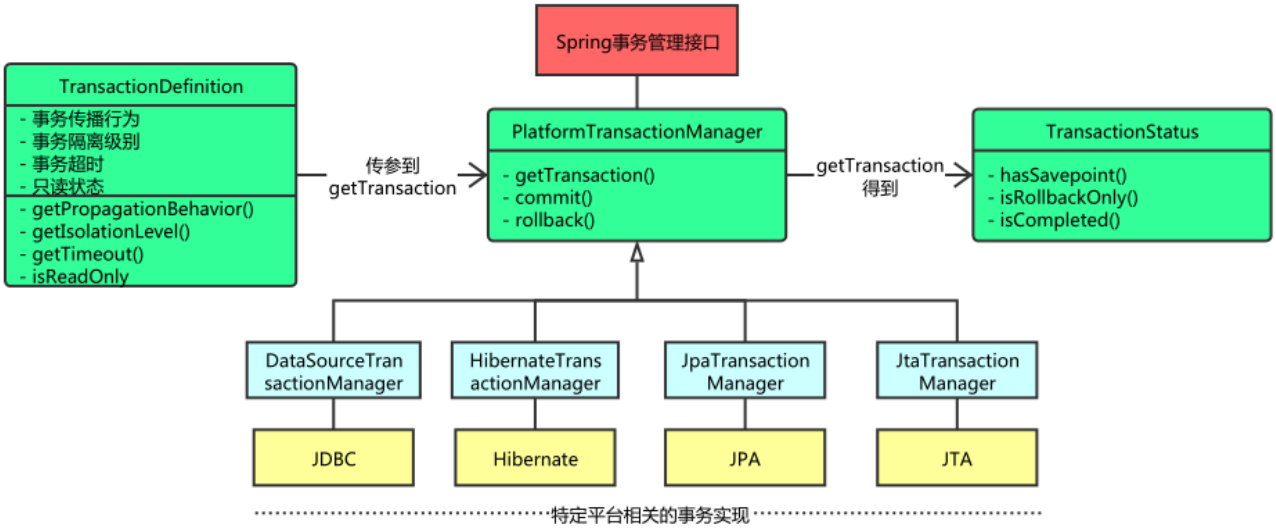
2. Spring事务

Spring提供了编程式事务和AOP声明式事务；

Spring的编程式事务在实际开发中很少使用；

这里主要讲声明式事务的两种使用方式；

2.1 Spring事务的核心API



2.1.1 PlatformTransactionManager - 事务管理器

Spring将事务管理交给了事务管理器 `PlatformTransactionManager` ，Spring为各个平台提供了相应的事务管理器，比如为JDBC提供的 `DataSourceTransactionManager` 。

```
public interface PlatformTransactionManager extends TransactionManager {

  /** 由TransactionDefinition得到事务状态对象(TransactionStatus) */
  TransactionStatus getTransaction(TransactionDefinition definition) throws
TransactionException;

  /** 提交事务 */
  void commit(TransactionStatus status) throws TransactionException;

  /** 回滚事务 */
  void rollback(TransactionStatus status) throws TransactionException;
}
```

2.1.2 TransactionDefinition - 事务的定义

TransactionDefinition描述事务以下几个方面内容

1. 事务的隔离级别
2. 事务的传播行为、传播机制
3. 事务的超时时间
4. 事务是否是只读状态

```
public interface TransactionDefinition {
    /** 传播行为 */
    default int getPropagationBehavior() {
        return PROPAGATION_REQUIRED;
    }
```

```
    /** 隔离级别 */
    default int getIsolationLevel() {
        return ISOLATION_DEFAULT; // 根据数据自动选择隔离级别
    }

    /** 超时时间 */
    default int getTimeout() {
        return TIMEOUT_DEFAULT;
    }

    /** 事务的只读状态 */
    default boolean isReadOnly() {
        return false;
    }

    @Nullable
    default String getName() {
        return null;
    }

    static TransactionDefinition withDefaults() {
        return StaticTransactionDefinition.INSTANCE;
    }
}
```

2.2 Spring的声明式事务配置 - xml配置

1. Maven依赖：

   Spring事务需要两个spring-jdbc、spring-tx两个模块，另外还有spring-aspects

   在Spring MVC项目中增加如下依赖

```
<!-- spring-jdbc传递依赖了spring-tx -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>5.3.19</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aspects</artifactId>
  <version>5.3.19</version>
</dependency>
```

2. Spring事务配置

```
<!-- 配置数据源 -->
```

```xml
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
  <property name="driverClassName" value="${jdbc.driver}" />
  <property name="url" value="${jdbc.url}" />
  <property name="username" value="${jdbc.username}" />
  <property name="password" value="${jdbc.password}" />
</bean>

<!-- 配置事务管理器 -->
<bean id="txManager"
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>

<!-- 事务通知（增强） -->
<tx:advice id="txAdvice" transaction-manager="txManager">
  <tx:attributes>
   <!-- 对哪些方法执行事务，配置事务定义信息（隔离级别、传播行为、是否只读、超时时间、回滚机制）-->
  <!-- 1. name属性：为哪些方法添加事务 -->
  <!-- 2. isolation属性：使用什么隔离级别，这里有5个隔离级别 -->
  <!-- 3. timeout属性：-1：表示永不超时；3：3秒钟超时 -->
  <!-- 4. read-only属性 -->
  <!-- 5. rollback-for属性：指定对哪些异常起作用 -->
  <!-- 6. no-rollback-for属性：指定对哪些异常不起作用 -->
  <!-- 7. propagation属性:配置事务的传播行为 -->
    <tx:method name="add*"
               isolation="DEFAULT"
               propagation="REQUIRED" />
    <tx:method name="insert*"
               isolation="DEFAULT"
               propagation="REQUIRED"
               rollback-for="Throwable"
               timeout="5"/>
    <tx:method name="delete*" isolation="DEFAULT"
               propagation="REQUIRED" />
    <tx:method name="query*" isolation="READ_UNCOMMITTED"
               read-only="true" />
  </tx:attributes>
</tx:advice>

<!-- aop配置 -->
<aop:config>
  <!-- 第一个*：任意方法返回类型 -->
  <!-- 第二个*：任意类名 -->
  <!-- 第三个*：任意方法名称 -->
  <!-- 第一个..：表示com.etoak.service包及其任意层级子包 -->
  <!-- 第二个..：任意方法参数 -->
  <aop:pointcut id="txPoint"
          expression="execution(* com.etoak.service..*.*(..))" />
  <aop:advisor advice-ref="txAdvice" pointcut-ref="txPoint" />
</aop:config>
```

2.3 Spring的声明式事务配置 - 注解配置

2.3.1 Spring注解事务配置

```
<!-- 配置数据源 -->
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
  <property name="driverClassName" value="${jdbc.driver}" />
  <property name="url" value="${jdbc.url}" />
  <property name="username" value="${jdbc.username}" />
  <property name="password" value="${jdbc.password}" />
</bean>

<!-- 配置事务管理器 -->
<bean id="txManager"
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>

<!-- 开启注解事务 -->
<tx:annotation-driven transaction-manager="txManager" />
```

2.3.2 @Transactional注解属性说明

1. isolation => Isolation(5种) - 隔离级别
2. timeout => int: 秒为单位，-1表示不超时；5:表示5秒超时
3. readOnly => boolean 用于加速查询效率
4. rollbackFor => Class[] 表示哪些异常需要回滚
5. rollbackForClassName => String[] 填写要回滚的异常的全限定类名称
6. noRollbackFor => Class[] 表示哪些异常不需要回滚
7. noRollbackForClassName => String[] 填写不需要回滚的异常的全限定类名称
8. transactionManager => String 指定事务管理器bean的id
9. propagation => Propagation 事务传播行为

2.4 Spring事务的隔离级别

TransactionDefinition接口中定义了五个表示隔离级别的常量：

1. `DEFAULT` ：默认使用数据库的隔离级别
2. **READ_UNCOMMITTED**
3. **READ_COMMITTED**
4. **REPEATABLE_READ**
5. **SERIALIZABLE**

注意：这里主要说一下DEFAULT配置，这是默认值，表示使用底层数据库的默认隔离级别。对大部分数据库而言，通常这值就是ISOLATION_READ_COMMITTED。

2.5 Spring事务的传播行为、传播特性、传播机制

Spring事务的传播行为是指 `多个事务` 同时存在时，Spring如何处理事务，它们是否需要在同一个事务中运行；

一个有事务的方法被另外有事务的方法调用时，这个事务应该如何运行；

```
methodA() { // 有事务
  update;
  ServiceB.methodB(); // 又调用了另一个有事务的方法
  异常;
}

methodB() { // 有事务
  update;
}
```

2.5.1 Spring的支持七种传播特性

1. REQUIRED：表示如果当前存在一个事务，则加入该事务，否则将新建一个事务；**这个是Spring事务默认传播行为**
2. REQUIRES_NEW：表示不管是否存在事务，都创建一个新的事务，把原来的事务挂起，新的事务执行完毕，继续执行老的事务；
3. SUPPORTS：表示如果当前存在事务，就加入该事务；如果当前没有事务，那就不使用事务；
4. NOT_SUPPORTED：表示不使用事务；如果当前存在事务，就把当前事务暂停，以非事务方式执行；
5. MANDATORY：表示必须在一个已有的事务中执行，如果当前没有事务，则抛出异常；
6. NEVER：表示以非事务方式执行，如果当前存在事务，则抛出异常；
7. NESTED：这个是嵌套事务；
   如果当前存在事务，则在嵌套事务内执行；
   如果当前不存在事务，则创建一个新的事务；
   嵌套事务使用数据库中的保存点来实现，**即嵌套事务回滚不影响外部事务**，但外部事务回滚将导致嵌套事务回滚；

2.6 Spring事务的回滚规则

1. Spring事务默认只对 `运行时异常[非检查型]` 起作用（也就是出现运行时异常时，回滚数据）；
2. Spring事务默认对 `非运行时异常[检查型异常、编译期异常]` 不起作用；
3. 可以使用 `rollbackFor` 指定Spring事务对哪些异常起作用；
4. 可以使用 `noRollbackFor` 指定Spring事务对哪些异常不起作用；