分布式事务

1. 事务

Java通过JDBC与数据库进行交互，这是一个如今多数程序员都不会直接使用的技术，我们更倾向于使用Hibernate和Mybatis，但是，他们在底层都需要JDBC与数据库通信，事务处理亦是如此。

JDBC提供的事务处理API非常少，主要通过以Connection类的方法完成：

Connection.setAutoCommit(boolean);

Connection.commit();

Connection.rollback();

Java事务是作用于Connection之上的。

jdbc事务中Connection负担了两方面的功能，

可以通过操作Connection连接来执行事务，这并不代表Connection具有事务功能，而是使用了数据库自身的事务功能，Connection仅仅是把一些命令如commit、rollback传递给数据库。

由于上述各家实现事务功能的方式各不相同，Spring进行了统一的抽象，形成了PlatformTransactionManager事务管理器接口，事务的提交、回滚等操作全部交给它来实现。Spring的事务体系也是在PlatformTransactionManager事务管理器接口上开展来的，所以先来了解下PlatformTransactionManager事务管理器。

事务功能的总体接口设计

先来看下三大接口

PlatformTransactionManager ： 事务管理器

TransactionDefinition ： 事务的一些基础信息，如超时时间、隔离级别、传播属性等

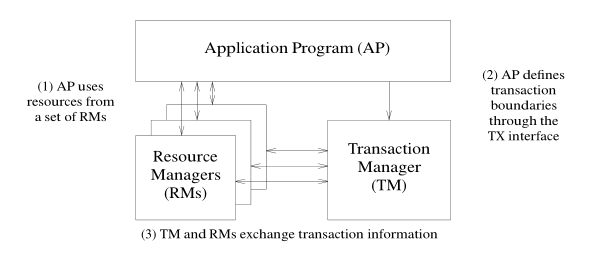
TransactionStatus ： 事务的一些状态信息，如是否是一个新的事务、是否已被标记为回滚

1. 本地事务和分布式事务

本地（Local Transaction）事务：只有一个数据源参与的事务（紧密依赖于底层资源管理器（例如数据库连接 )，事务处理局限在当前事务资源内）

分布式事务(Distributed Transaction)：有多个数据源同时参与的事务。对于分布式事务，Java提供了JTA规范，它的原理与本地事务存在不同。

1. 分布式的几种解决方式
2. 二阶段提交
3. 本地事务+消息
4. TCC
5. DTP是一种分布式事务处理模型



上面主要涉及了三个对象：

AP(Application Program)：应用程序

TM(Transaction Manager)：事务管理器

负责协调和管理事务，它和之前说的Spring的事务管理器PlatformTransactionManager不一样。

RM(Resource Manager)：资源管理器

可以理解为数据库或者JMS。

他们三者的关系：

AP通过TM来操作多个RM，AP也可以通过RM的本地事务接口来操作单个RM

TM和RM可以互相通信，他们之前的通信协议就是XA协议

1. 两阶段提交

全称The two-phase commit protocol，简称2PC

当需要协调多个资源管理器进行分布式事务的时候，就需要使用到两阶段提交协议

1. JTA:

一种规范，不同的应用服务器都包含有自己的实现（比如JbossJTA），同时还存在独立于应用服务器的单独JTA实现，比如Atomikos。

分布式事务（Distributed Transaction）包括事务管理器（Transaction Manager）和一个或多个支持 XA 协议的资源管理器 ( Resource Manager )。我们可以将资源管理器看做任意类型的持久化数据存储；事务管理器承担着所有事务参与单元的协调与控制。

将 JTA 的事务管理器和资源管理器理解为两个方面：面向开发人员的使用接口（事务管理器）和面向服务提供商的实现接口（资源管理器）。

在 JTA 事务 代码中获得的数据库源 ( DataSource ) 必须是支持分布式事务的。支持事务的数据源与普通的数据源是不同的，它实现了额外的 XADataSource 接口。我们可以简单的将 XADataSource 理解为普通的数据源（继承了 java.sql.PooledConnection），只是它为支持分布式事务而增加了 getXAResource 方法。另外，由 XADataSource 返回的数据库连接与普通连接也是不同的，此连接除了实现 java.sql.Connection 定义的所有功能之外还实现了 XAConnection 接口。我们可以把 XAConnection 理解为普通的数据库连接，它支持所有 JDBC 规范的数据库操作，不同之处在于 XAConnection 增加了对分布式事务的支持。