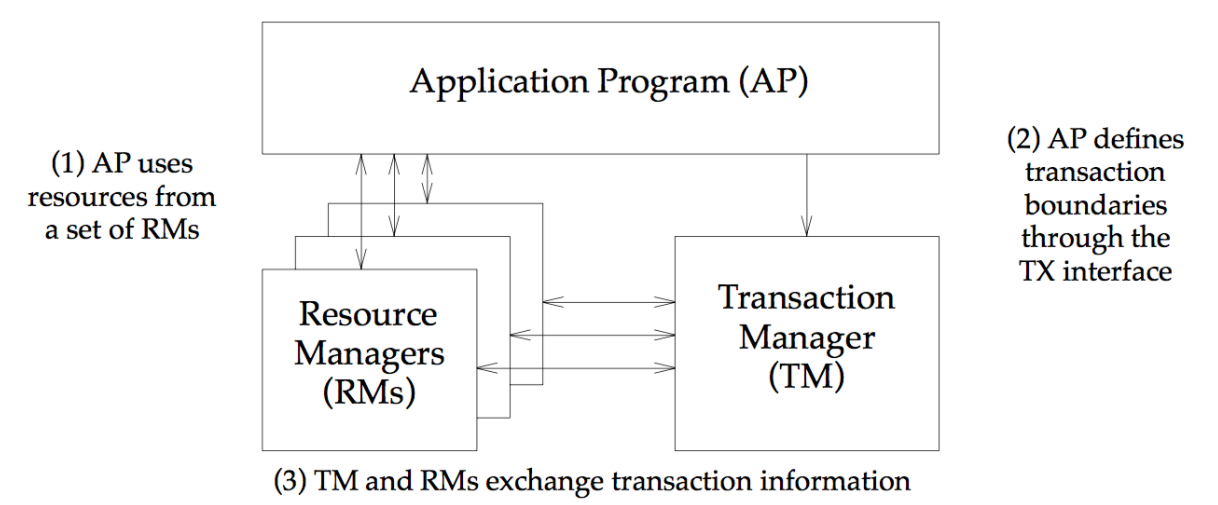
1. **X/Open DTP分布式事务模型**

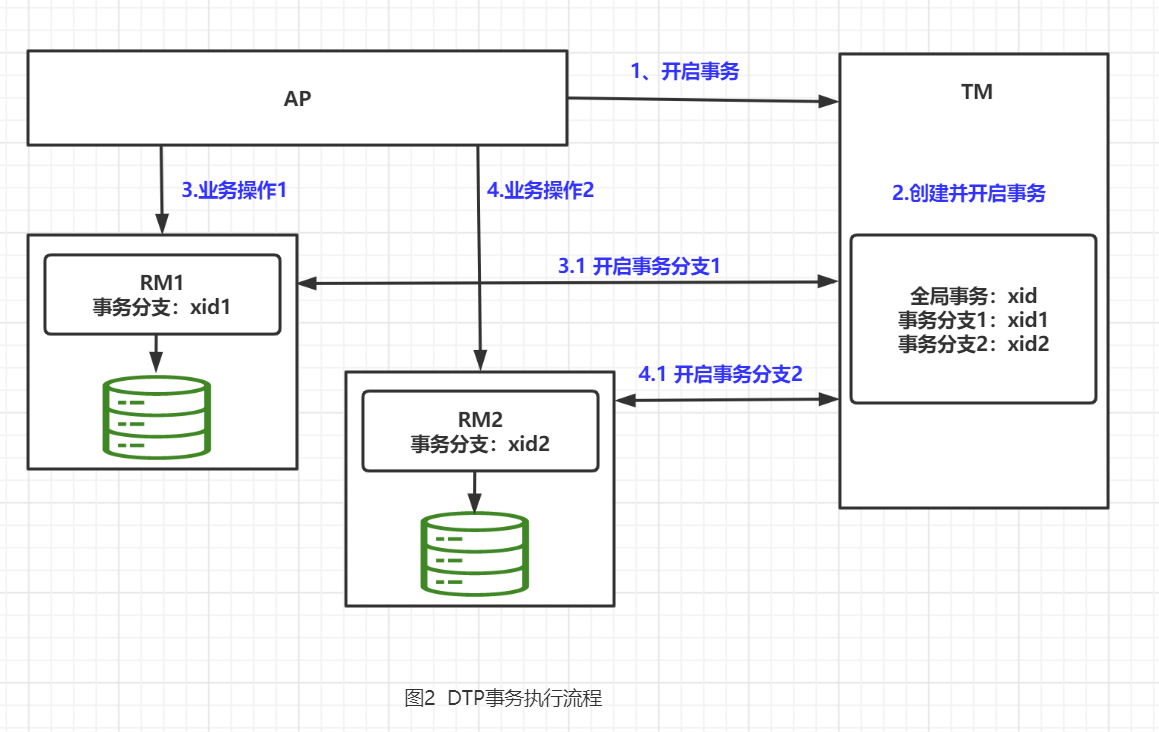
X/Open DTP（Distributed Transaction Processing）是X/Open组织定义的一套分布式事务的标准。该标准提出使用2PC来保证分布式事务的完整性。在DTP模型中，主要有如下三种角色：

* AP（Application）：表示应用程序
* RM（Resource Manager）：表示资源管理器，如数据库、文件系统等，并提供访问资源的方式
* TM（Transaction Manager）：表示事务管理器，一般指事务协调者，负责协调和管理事务



注：上图中的（1）表示AP-RM的交互接口，（2）表示AP-TM的交互接口，（3）表示RM-TM的交互接口（基于XA协议完成）

1. **DTP模型的标准事务执行流程**



具体执行流程说明如下

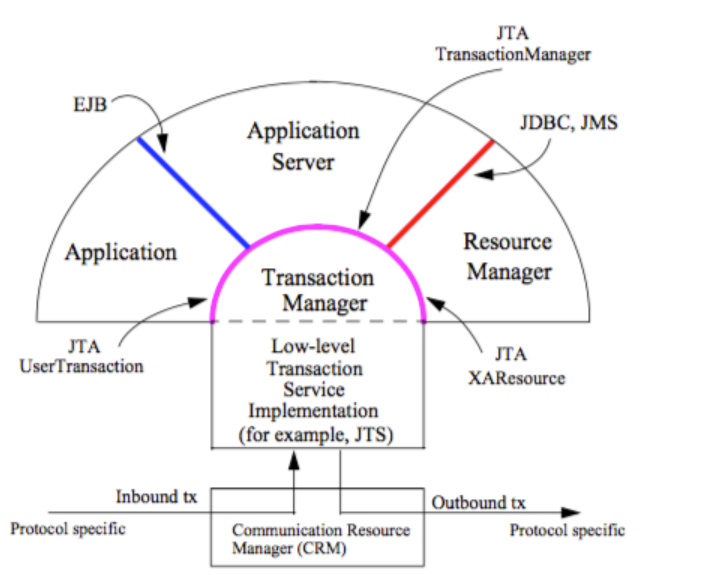
1. 配置TM，将多个RM注册到TM；
2. AP向TM发起一个全局事务，TM生成XID；
3. AP通过TM管理的RM1获取连接，进行相关的业务操作，AP每次操作时会把XID传递给RM；
4. AP结束全局事务，TM通知各个RM全局事务结束；
5. TM根据各个RM的事务执行结果，执行提交或者回滚操作；

简易实现Demo：[MysqlXATest.java](D:\\BaiduNetdiskDownload\\课件\\代码\\spring-annotation\\src\\main\\java\\com\\atguigu\\mysql\\MysqlXATest.java)

1. **JTA（Java Transaction API）模型**

JTA模型是DTP模型的java实现，将XA规范中定义的DTP模型交互接口抽象成java接口中的方法，并规定每个方法要实现的功能。在JTA模型中，主要包含如下角色：

* TM（Transaction Manager）：事务管理器，提供支持事务界定、事务资源管理、同步、事务上下文传播所需的服务和管理功能；
* AS（App Server）：应用服务器，提供事务状态管理；
* RM（Resource Manager）：资源管理器，为应用程序提供对资源的访问；
* AP（Application Program）：基于组件的事务型应用程序，通过声明型事务属性设置提供事务管理支持；
* CRM（Communication Resource Manager）：通信资源管理器；



1. **JTA接口**

**通过maven引入依赖库**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-jta-atomikos</artifactId>  
 <version>2.1.3.RELEASE</version>  
</dependency>

**重要接口如下：**

**XAResource：**XA资源管理器接口，RM提供给TM调用。XA规范中XA接口的java映射；

**Xid：**XA规范中事务标识Xid的java映射。分别获取全局事务的格式化ID、全局事务ID、分支限定符；

**Status：**定义事务状态。一共10种状态；

**Synchronization：**同步接口，事务同步允许AS在事务完成之前和之后从TM获得通知；

**Transaction：**事务定义接口，每个全局事务都与一个事务对象关联；

**TransactionManager：**事务管理器，管理事务的全生命周期；

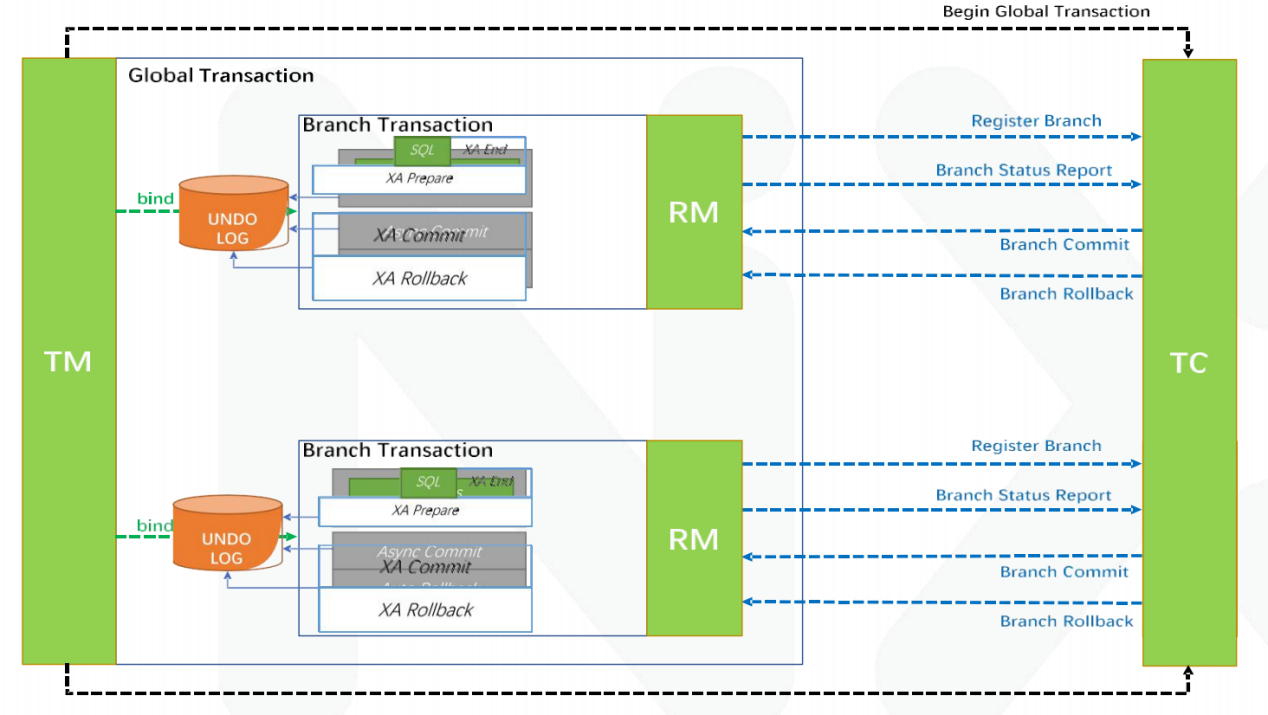
**TransactionSynchronizationRegistry：**事务同步注册器，

**UserTransaction：**事务客户端，封装了用户可直接操作事务的接口。

**XAResource、Xid由数据库产商实现，如MysqlXAConnection、MysqlXid**

**TransactionManager、UserTransaction等事务操作相关的接口由AS产商实现，如开源的第三方库jotm、Atomikos等**

1. **Seata框架的标准事务执行流程**



Seata事务框架支持AT、TCC、Saga、XA四种事务模式，其中AT模式是Seata主推的分布式事务解决方案。它是基于XA演进而来，包含TM（事务管理器，负责向TC注册一个全局事务，并生成一个全局唯一的XID）、RM（资源管理器，用于访问数据库资源）、TC（事务协调器，维护全局事务的运行状态，负责协调并驱动全局事务的提交和回滚，作为单独的服务独立部署）三大模块。具体执行流程如下：

1. TM向TC注册全局事务，TC创建全局事务并返回全局唯一的XID，XID会在全局事务的上下文中传递；
2. RM向TC注册分支事务，并将其纳入该XID对应的全局事务范围；
3. RM向TC汇报资源的准备状态；
4. TC汇总所有事务参与者的执行状态，决定分布式事务是全部提交还是全部回滚；
5. TC通知所有RM提交或者回滚事务。
6. **复赛亮点提炼**
7. TM高可用设计
8. 丰富的监控界面
9. 配置简单
10. **模块划分**
11. 事务详细流程设计【尹冲负责】

建议RM和TM两端都增加超时设计，避免阻塞太久

1. TM高可用设计【蒋再武负责】
2. TM元数据存储设计【蒋再武负责】
3. 监控模块设计【单建负责】
4. 监控面板炫酷设计【李洁颖负责】
5. 总体汇报PPT和冒烟测试用例设计【邵亚雄负责】