# 分布式事务组件使用说明

## Maven引用

|  |
| --- |
| <dependency>  <groupId>com.yqkj</groupId>  <artifactId>yqkj-dtransaction</artifactId> </dependency> |

## application.yml配置文件，设置datasrouce和JPA

第一次运行，设置spring.jpa.properties.hibernate.hbm2ddl.auto为create，创建表结构(dtx\_main、dtx\_step、hibernate\_sequence)；

之后运行，将该设置修改为none，启动时将不创建和修改表结构；

## 编写代码

* 创建类A
* 类须增加@DTransaction注解
* 类须实现IDTransaction接口
* execute、confirm和rollback的入参和出参必须要实现IDTransactionHeader和IDTransactionResponse两个接口

备注：系统已经创建了AbstractDTransactionRequest抽象类，它实现了IDTransactionHeader接口，自定义的入参可以统一继承该抽象类。

|  |
| --- |
| **@Service**  **@DTransaction**  **Public class A implement IDTransaction {**  @Override  **public** String getDTServiceName();  @Override  **public** IDTransactionResponse execute(IDTransactionHeader request);  @Override  **public** IDTransactionResponse confirm(IDTransactionHeader request);  @Override  **public** IDTransactionResponse rollback(IDTransactionHeader request);  **}** |

* 业务与分布式事务组件交互

交互是通过入参实现的，入参实现了IDTransactionHeader接口，分布式事务组件通过前置拦截器设置其中的transactionHeader属性，该属性包含sessionId和stepId。

**sessionId**：会话ID，每次事务都会产生一个唯一的sessionId，格式为时间戳+4位随机数的长整形。

**stepId**：子事务ID，对应数据库中dtx\_step表的主键ID。

提交（Confirm）和回滚（Rollback）方法可以通过sessionId和stepId来进行业务关联，以实现对应的提交和回滚操作。

* 异常和回滚，在编写业务代码时，不满足业务需求的场景应该抛出运行时异常（RuntimeException），分布式事务组件会识别异常，触发各个子事务的回滚操作；

### 待优化、实现功能

1. 提交、回滚过程异常处理；
2. 幂等；
3. 响应消息头处理；