## LCN分布式事务控制原理

## 概述

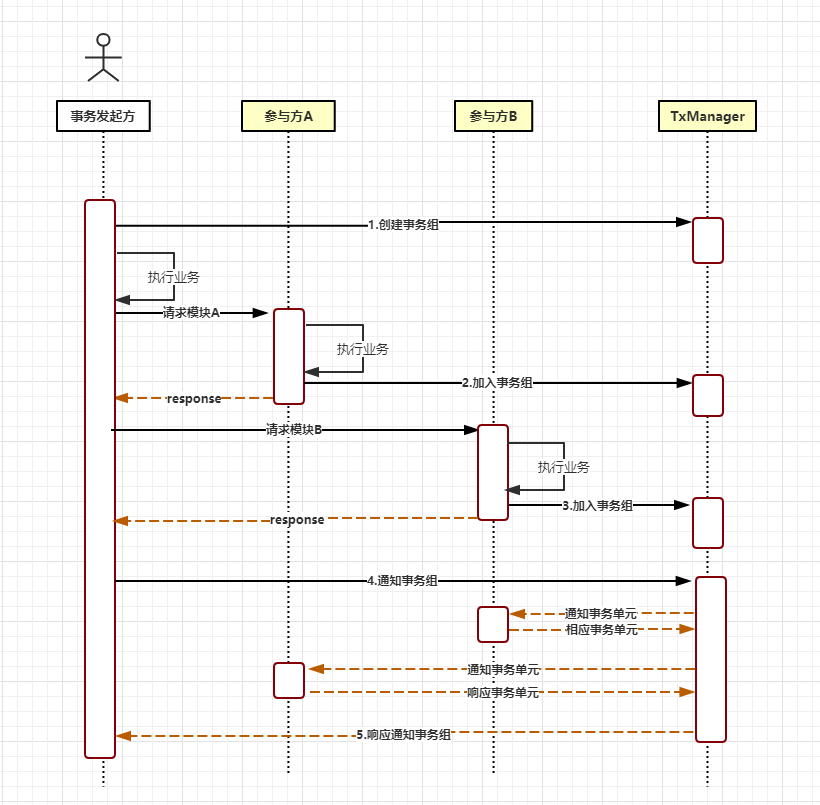
TX-LCN是一款高性能、易于使用的分布式事务框架。

## 功能特点

* 无侵入式支持分布式事务。
* 支持多种模式混合使用。
* 支持关系型数据、NoSQL数据库。
* 支持SpringCloud Dubbo等RPC框架。
* 支持负载均衡与集群化部署。
* 高性能与高稳定性。

TX-LCN官网：<http://www.txlcn.org/zh-cn/>

TX-LCN由两大模块组成：TM(TxManager)、TC(TxClient)。TM是全局事务协调者,TC为事务参与者（也包含事务发起者）。其事务的控制原理如下：



**核心步骤：**

1.事务发起者创建一个全局事物组group。

2.全局事务组通过Tracing（调用链子模块）进行服务间的全局数据传参。

3.其它参与者在调用链过程中加入到事务组

4.事务通知（各TC将事务状态上报统一交由TM进行管理）

5.TC响应TM的事务通知(**底层采用netty进行通讯**)

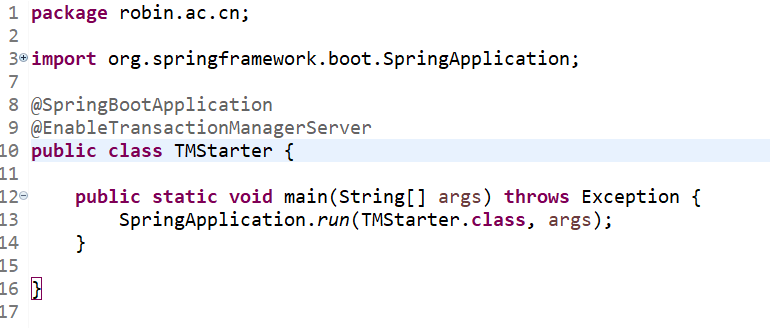
## 二、TM部署启动

### 1.maven引入tm，pom.xml配置

<dependency> <groupId>com.codingapi.txlcn</groupId> <artifactId>txlcn-tm</artifactId> <version>5.0.2.RELEASE</version> </dependency> <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-data-redis</artifactId> </dependency>

### 2.开启事务管理

在springboot启动类上加入注解@EnableTransactionManagerServer:



### 3.TM环境

**创建数据表（t\_tx\_exception）**

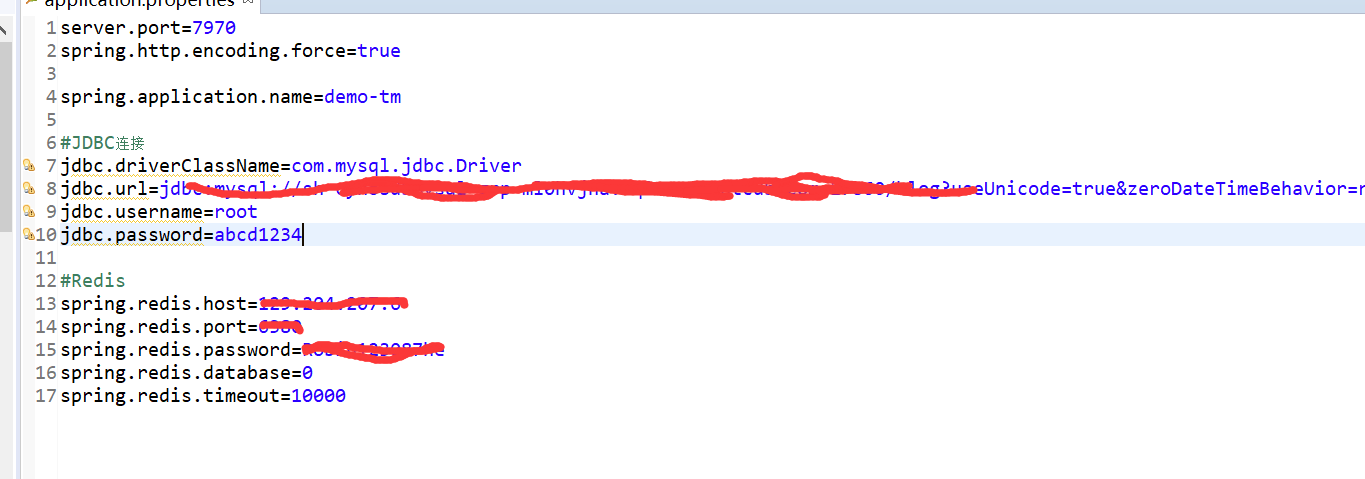
CREATE TABLE `t\_tx\_exception` ( `id` bigint(20) NOT NULL AUTO\_INCREMENT, `group\_id` varchar(64) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL, `unit\_id` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL, `mod\_id` varchar(128) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL DEFAULT NULL, `transaction\_state` tinyint(4) NULL DEFAULT NULL, `registrar` tinyint(4) NULL DEFAULT NULL, `remark` varchar(4096) NULL DEFAULT NULL, `ex\_state` tinyint(4) NULL DEFAULT NULL COMMENT '0 未解决 1已解决', `create\_time` datetime(0) NULL DEFAULT NULL, PRIMARY KEY (`id`) USING BTREE ) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

**application.properties配置信息：**

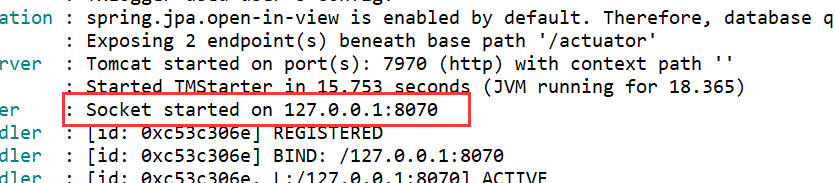
以下是推荐配置：

spring.application.name=tx-manager server.port=7970 spring.datasource.driver-class-name=com.mysql.jdbc.Driver spring.datasource.url=jdbc:mysql://127.0.0.1:3306/tx-manager?characterEncoding=UTF-8 spring.datasource.username=root spring.datasource.password=root mybatis.configuration.map-underscore-to-camel-case=true mybatis.configuration.use-generated-keys=true #tx-lcn.logger.enabled=true # TxManager Host Ip #tx-lcn.manager.host=127.0.0.1 # TxClient连接请求端口 #tx-lcn.manager.port=8070 # 心跳检测时间(ms) #tx-lcn.manager.heart-time=15000 # 分布式事务执行总时间 #tx-lcn.manager.dtx-time=30000 #参数延迟删除时间单位ms #tx-lcn.message.netty.attr-delay-time=10000 #tx-lcn.manager.concurrent-level=128 # 开启日志 #tx-lcn.logger.enabled=true #logging.level.com.codingapi=debug #redis 主机 #spring.redis.host=127.0.0.1 #redis 端口 #spring.redis.port=6379 #redis 密码 #spring.redis.password=

配置样例：

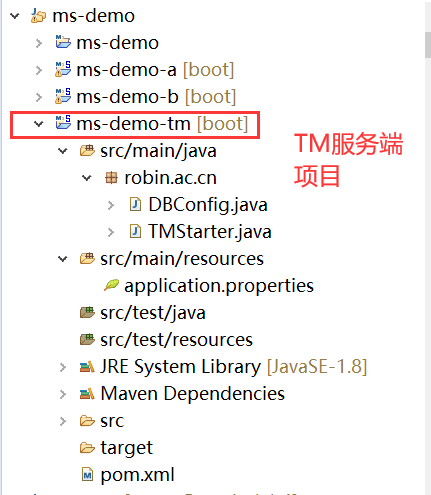


TM启动：

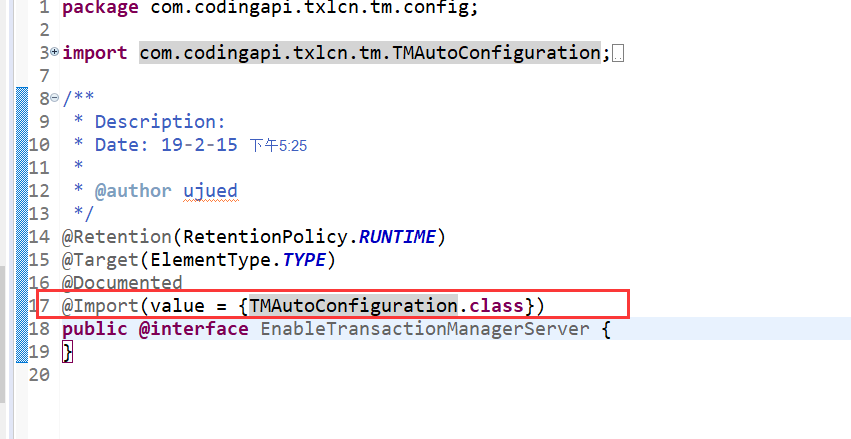


## 三、TM启动过程跟进

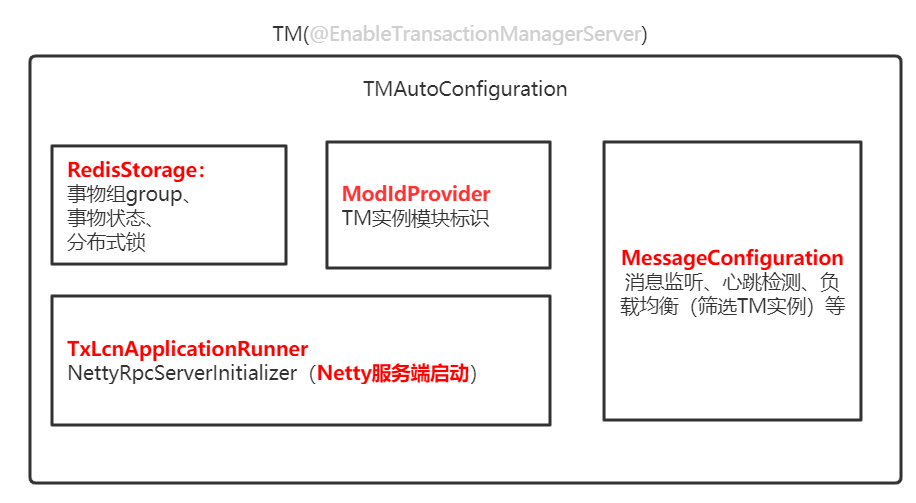
**tm可以独立部署应用为lcn服务端项目**



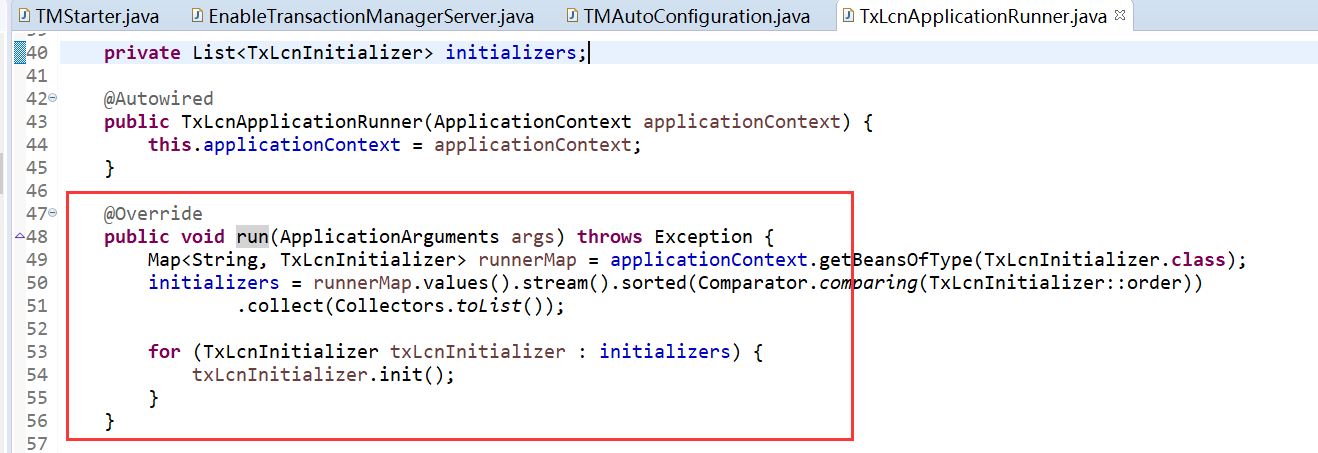
1.@EnableTransactionManagerServer注解导入了TMAutoConfiguration自动配置：



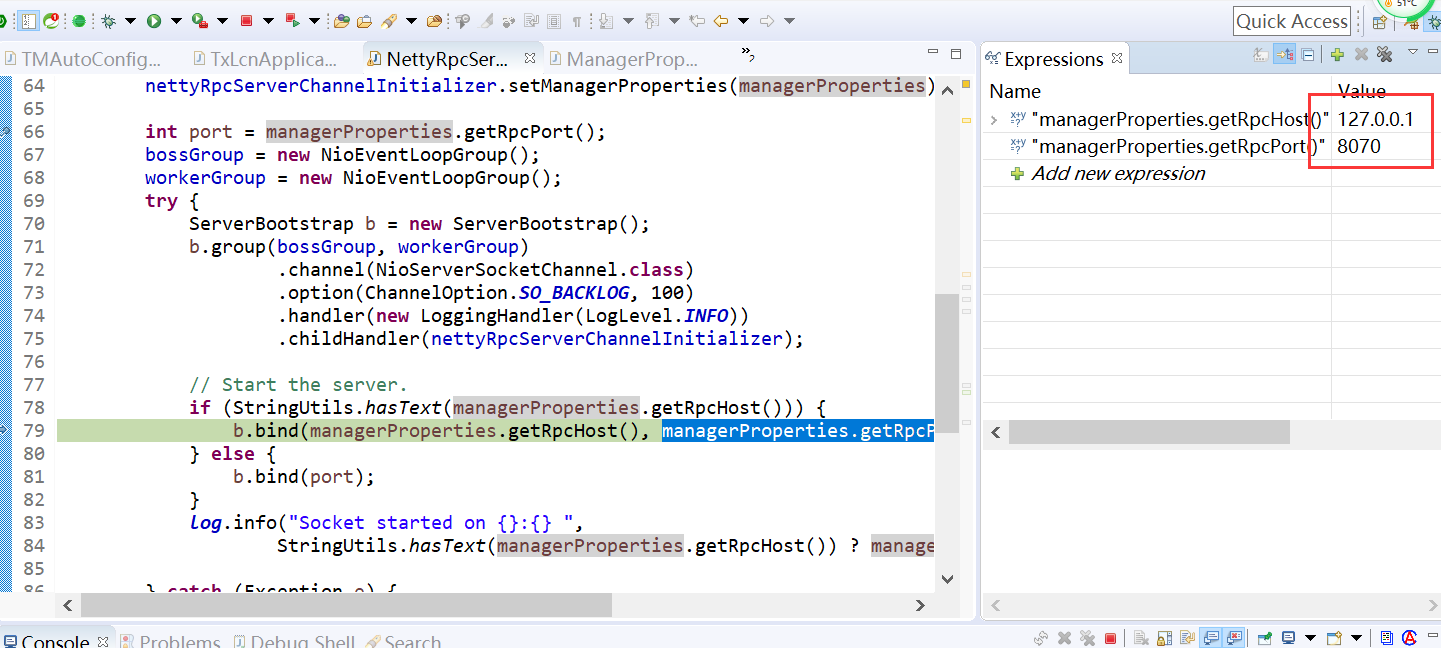
2.TMAutoConfiguration核心配置类：



3.TxLcnApplicationRunner实现了ApplicationRunner接口，springboot在IOC容器初始化过程中会自动调用run方法：



4.initializers的初始化器的其中NettyRpcServerInitialize对TM启动开启了netty服务（客户端TC连接的服务器）



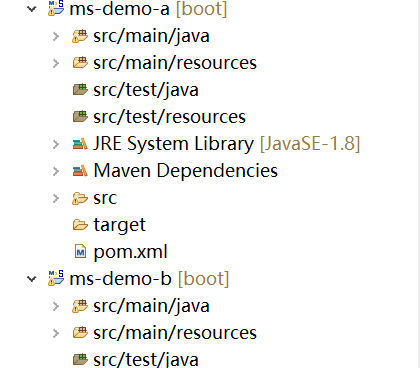
启动netty服务默认的ip/port是：127.0.0.1/8070

5.TMAutoCluster将TM实例信息存储到Redis



## 四、TC部署启动

**tc可以在你的任何一个独立微服务项目中引入作为客户端应用。**



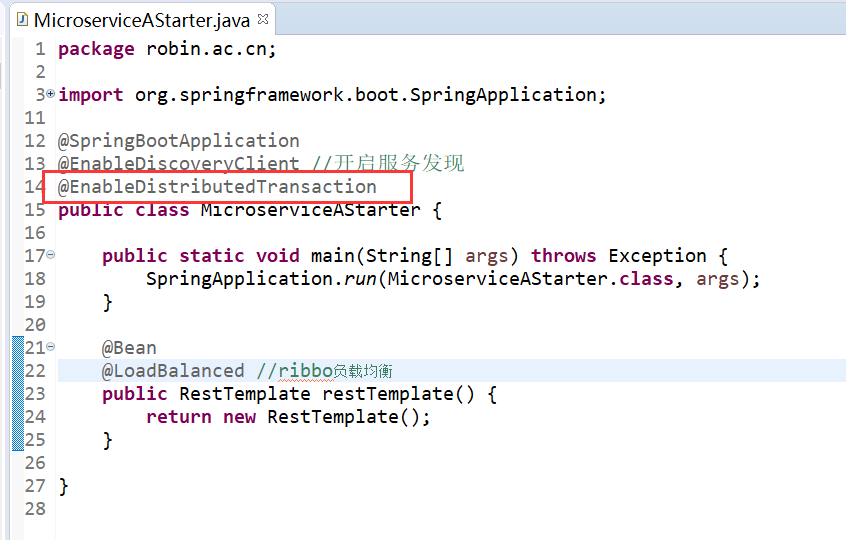
1.pom.xml配置：

<dependency> <groupId>com.codingapi.txlcn</groupId> <artifactId>txlcn-tc</artifactId> <version>5.0.2.RELEASE</version> </dependency> <dependency> <groupId>com.codingapi.txlcn</groupId> <artifactId>txlcn-txmsg-netty</artifactId> <version>5.0.2.RELEASE</version> </dependency>

2.application.properties配置信息：

# 默认之配置为TM的本机默认端口 tx-lcn.client.manager-address=127.0.0.1:8070

3.启动类开启注解@EnableDistributedTransaction

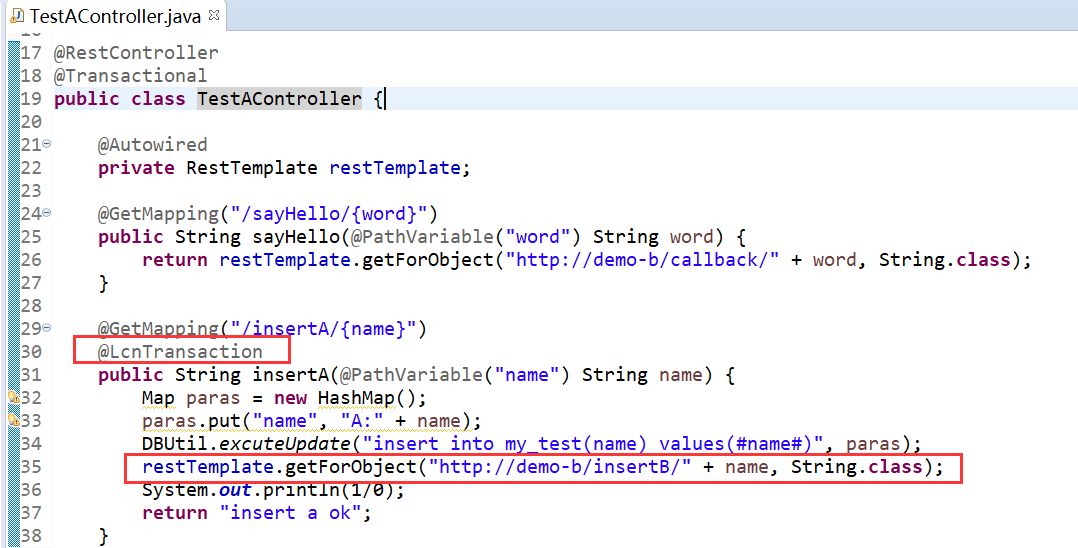


## 五、开始使用lcn分布式事务

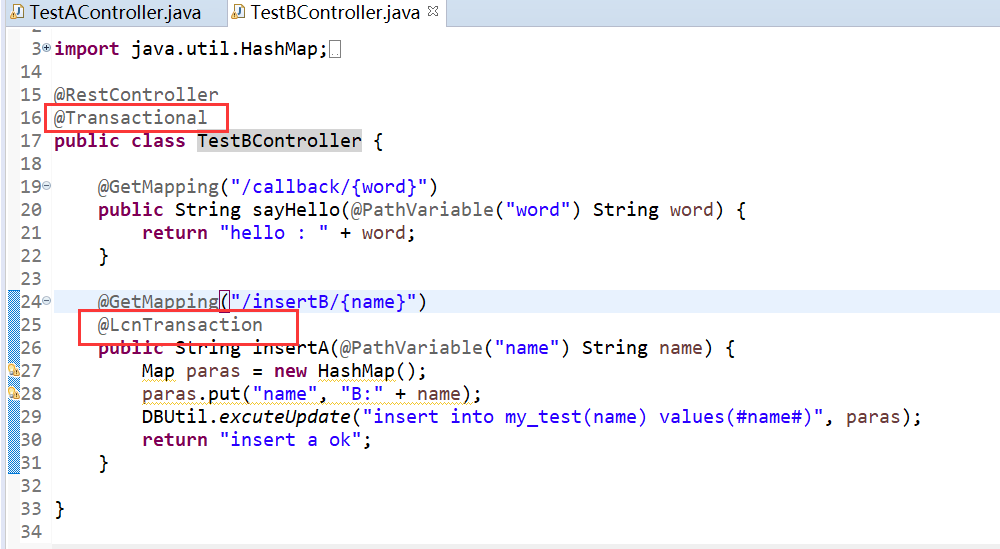
**使用事务的前提是接口本身开启了本地@Transactional事务（lcn不产生新事务）**

**1.项目环境介绍：**

微服务**demo-a**--->TestAController开启lcn事务，往表my\_test插入一条数据并调用**demo-b**微服务接口，调用demo-b以后打印1/0模拟错误：

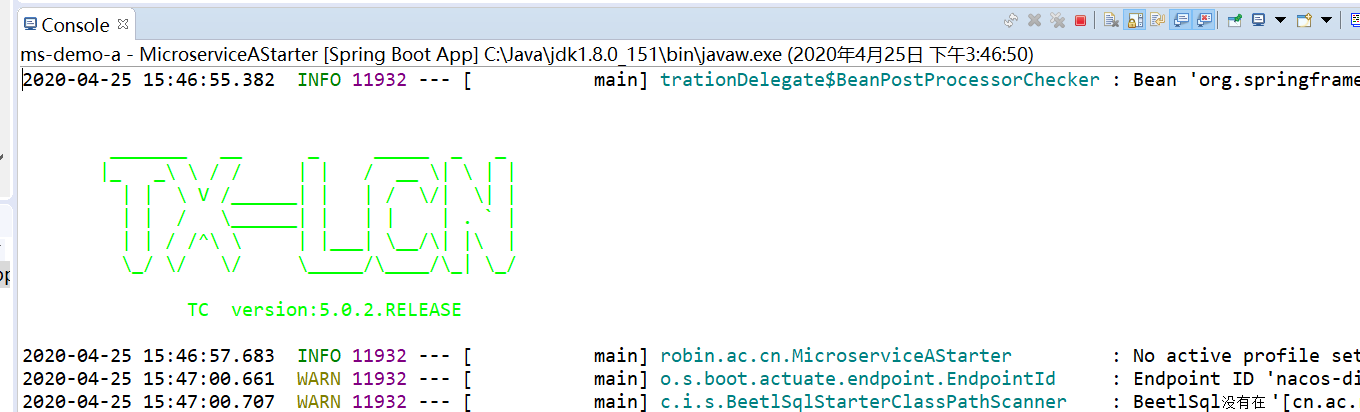


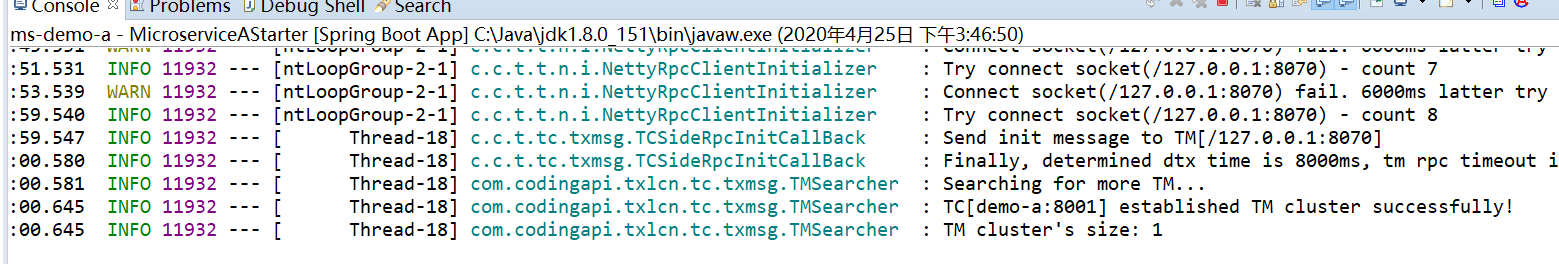
微服务**demo-b**--->TestBController开启lcn事物，往my\_test出入一条数据：

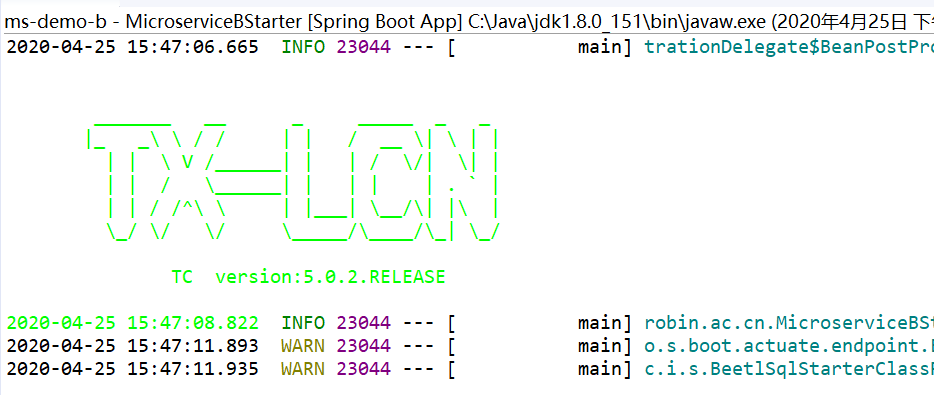


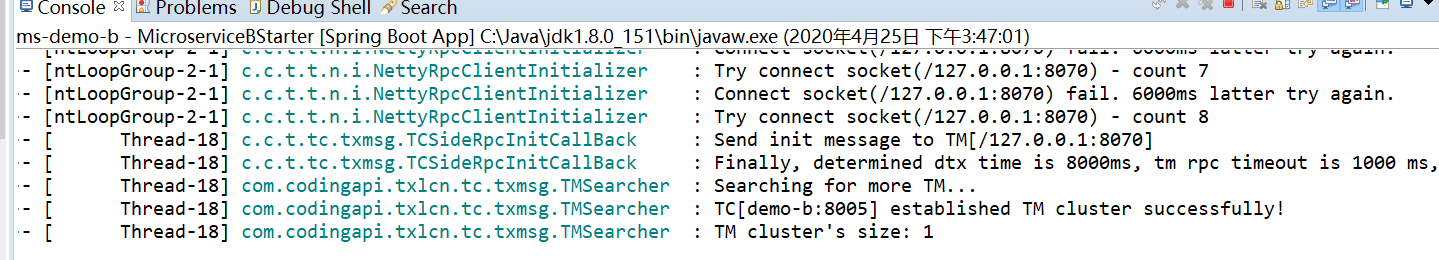
**2.测试：**

分别启动微服务demo-a、demo-b:

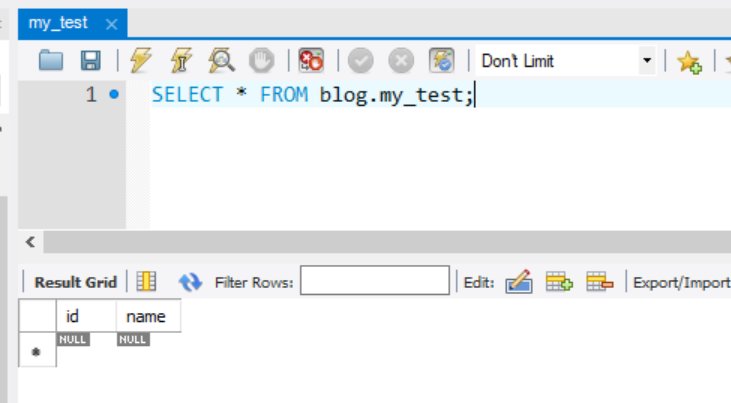








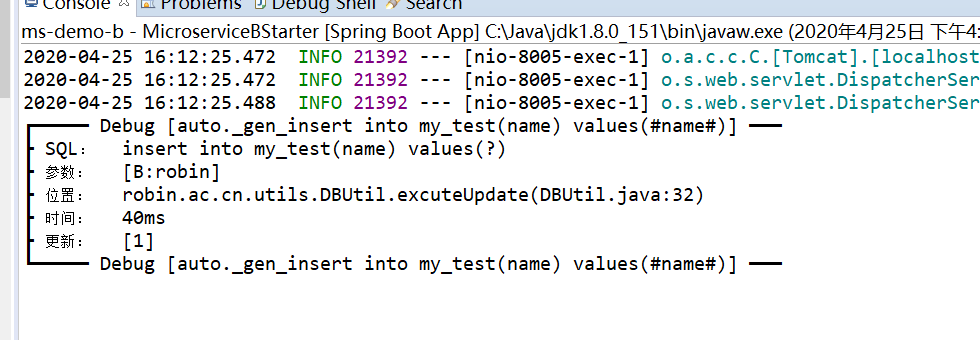
接口调用前数据库表数据：



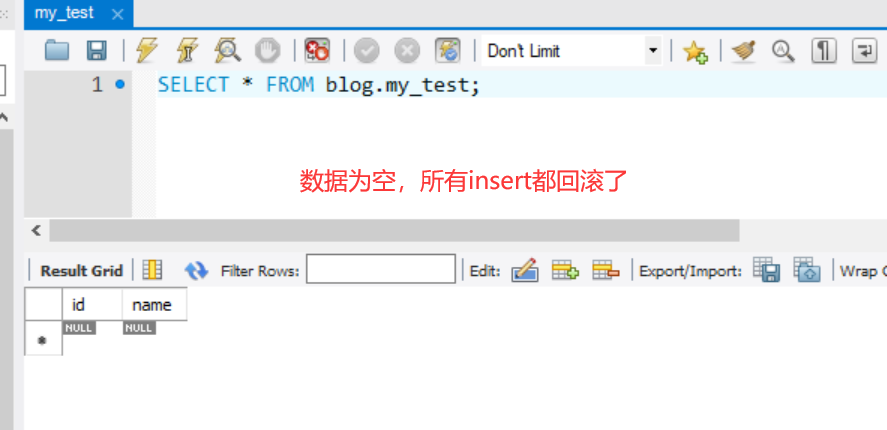
接口调用：



微服务demo-b执行了insert语句：



再查看数据库：

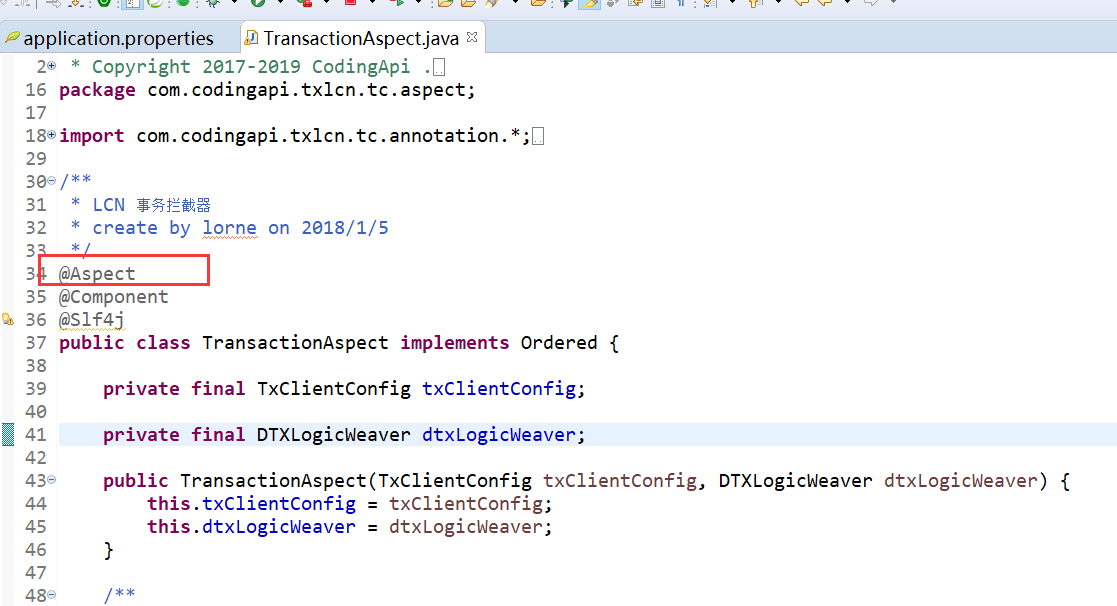


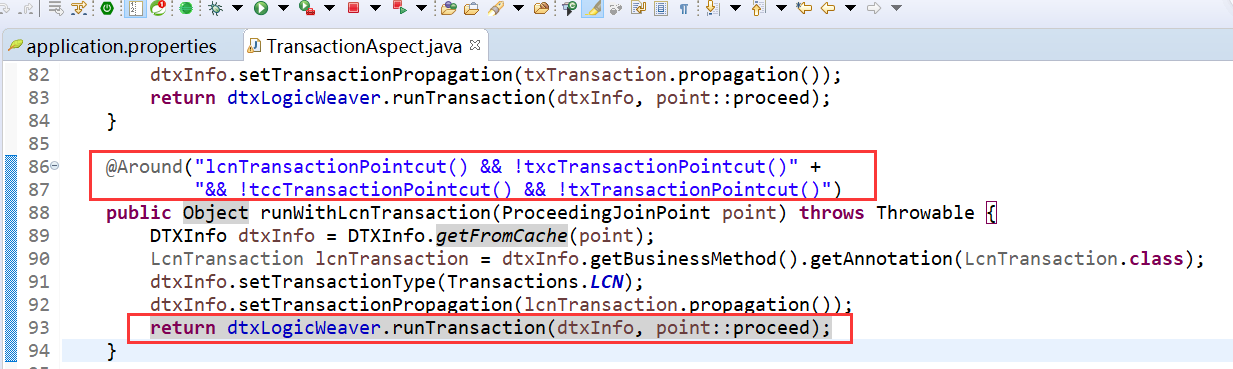
结果说明了，lcn的事物产生了作用，对demo-b微服务进行了回滚。

## 六、lcn调用过程跟进

**核心：事物拦截器、tracing调用链**

lcn事物代理主要是通过TransactionAspect切面拦截实现的：

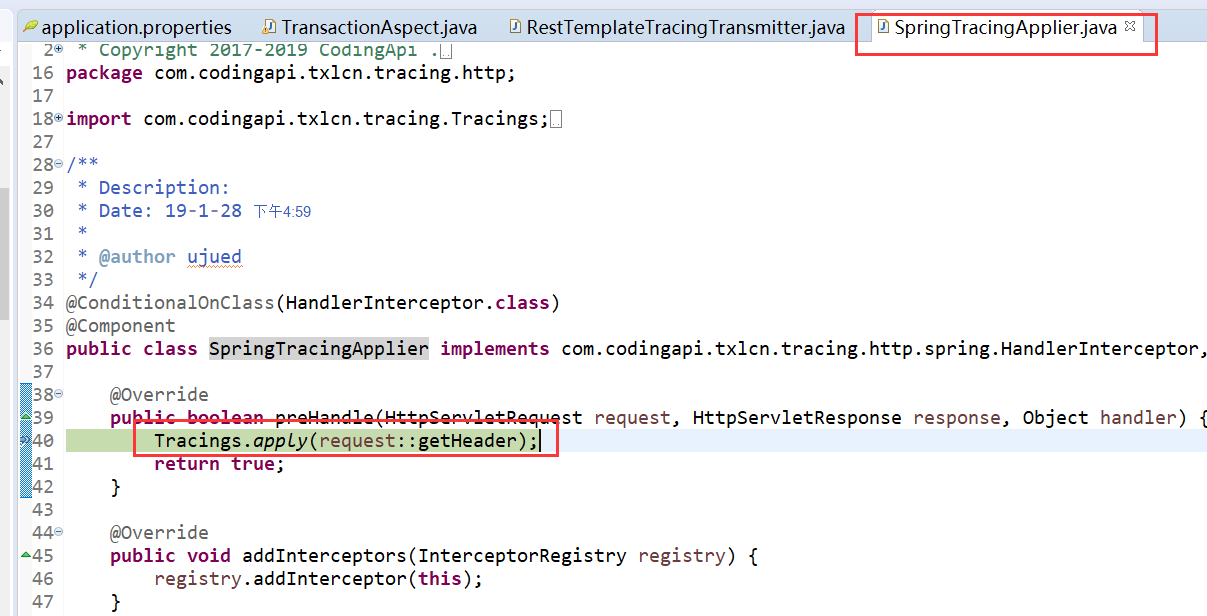




下面开始跟进：

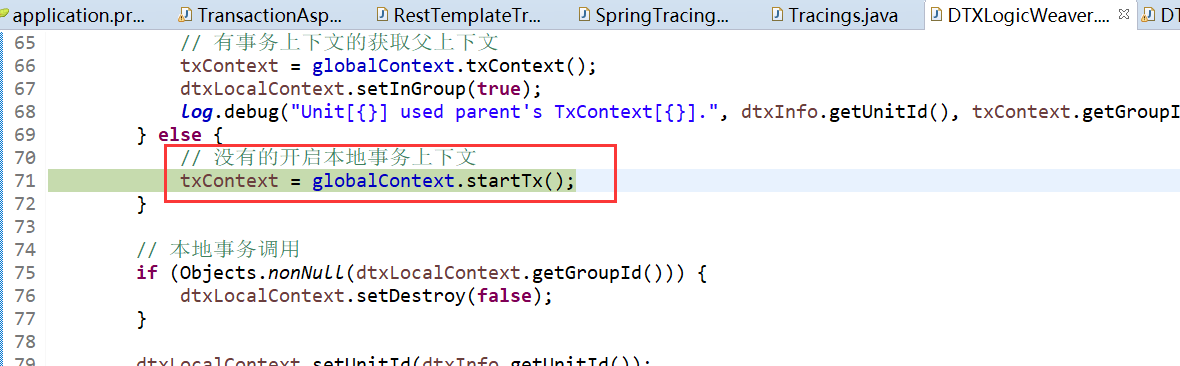
demo-a:事物发起方

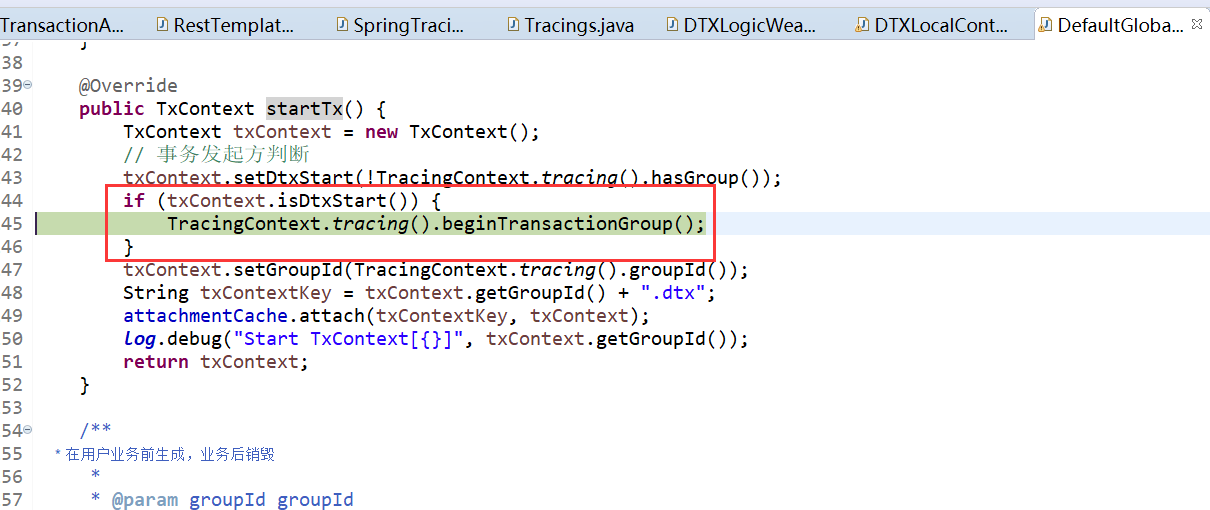
先进入SpringTracingApplier，获取groupId



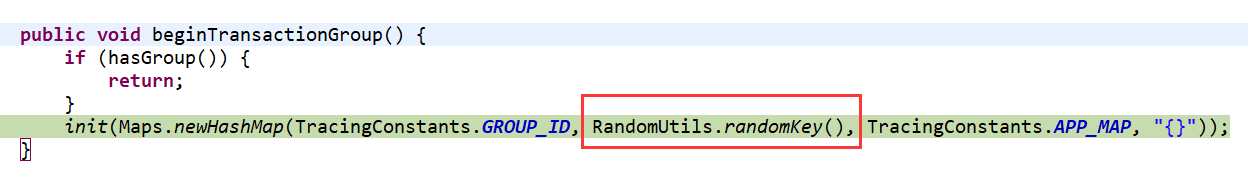


首次调用groupId为空,接下来进入TransactionAspect拦截器开始方法拦截，开启事物，并创建全局事务组：





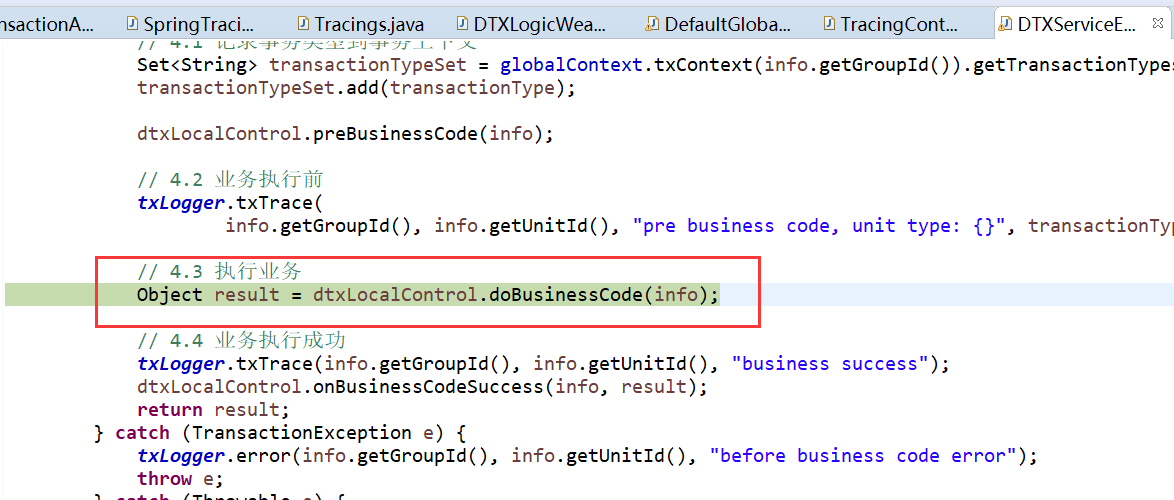
随机生成事物组ID（**这块可以优化，建议采用雪花算法生成，确保全球唯一ID**）



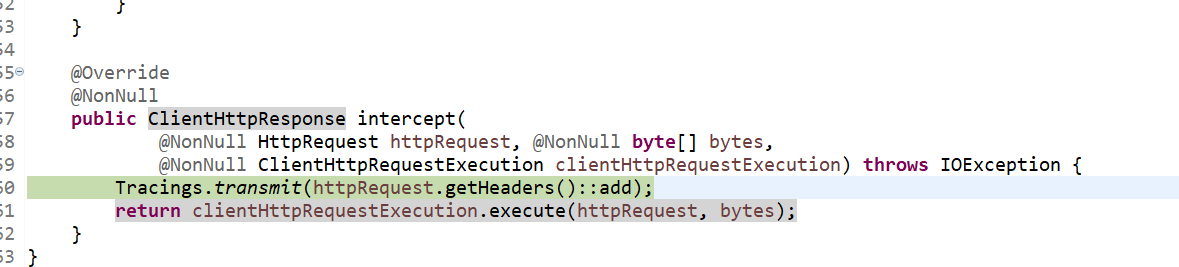
事物参数进行封装：

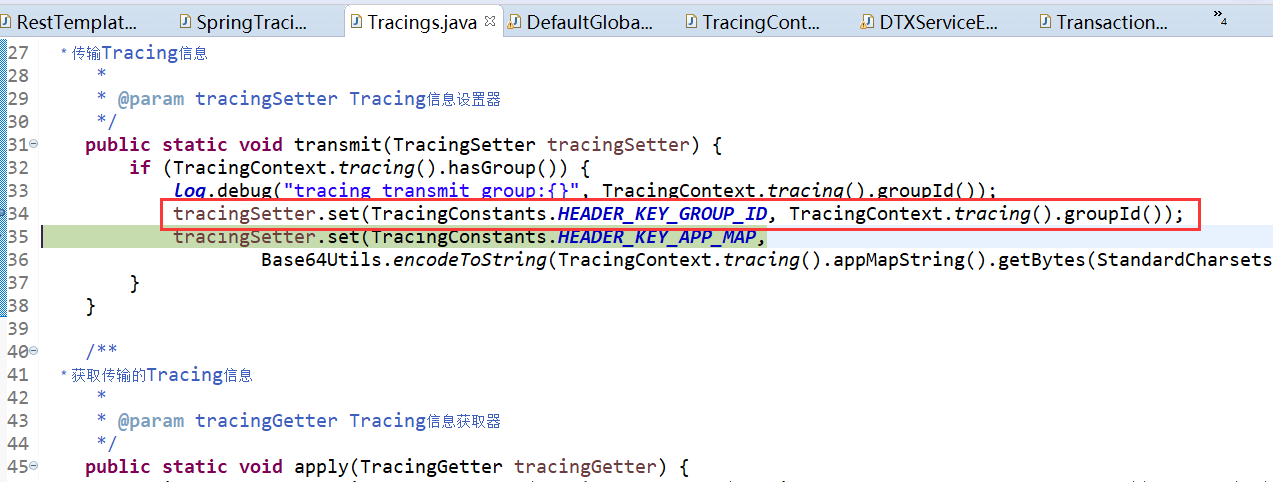


执行本地业务逻辑：

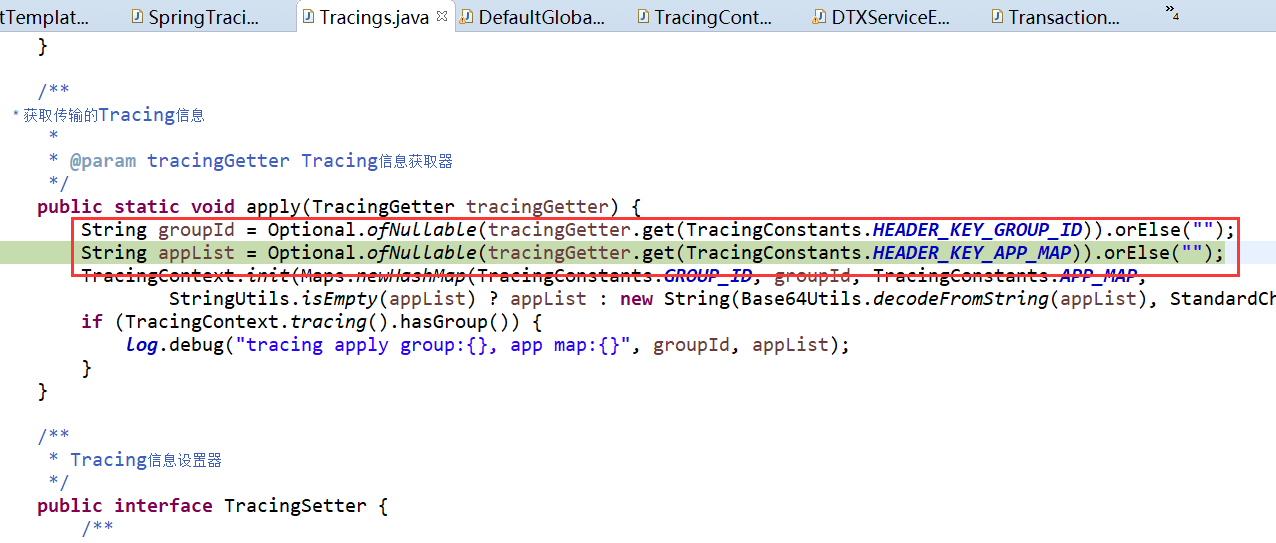


resttemplate监听，将全局事物组加入http header(X-Group-ID,X-App-Map)中进行下游微服务的传递

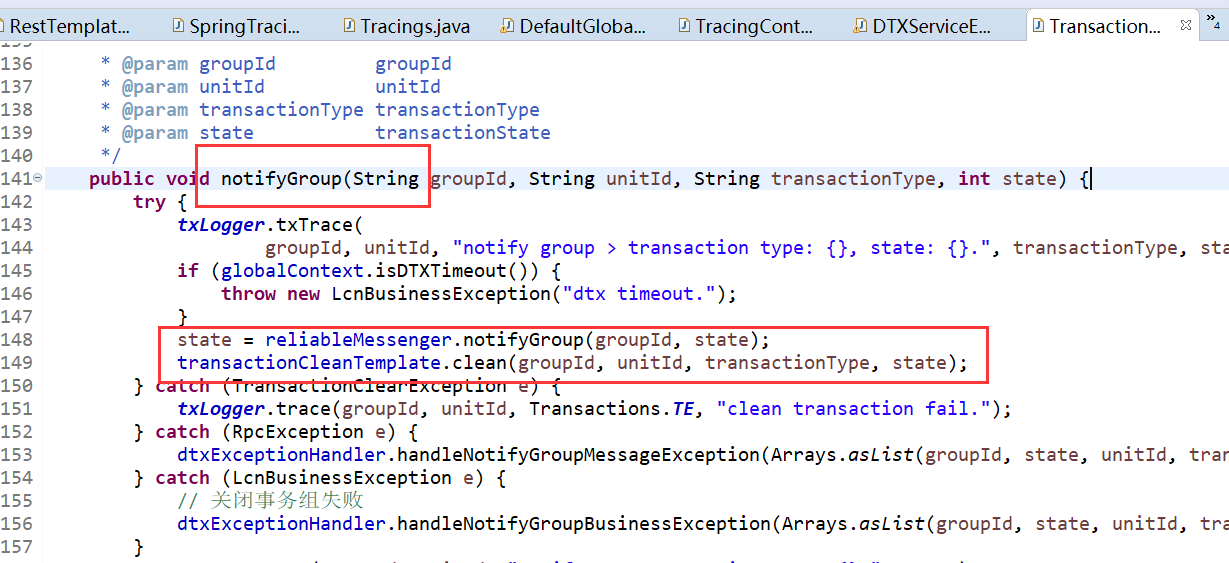




开始进入微服务demo-b(获取demo-a传递的header参数),加入事务组:



本地事物状态上传TM：



本文介绍到这里就把lcn分布式事物的核心原理，并对核心源码进行了跟踪，后续事物通知采用的是netty进行通讯。