链滴

seata+nacos 实现 TCC 模式分布式事务

作者：wenyl

原文链接：https://ld246.com/article/1635916035380

来源网站：链滴

1、简介

AT 模式（参考链接 TBD）基于 **支持本地 ACID 事务** 的 **关系型数据库**：

TCC 模式，不依赖于底层数据资源的事务支持。

2、建立项目

TCC模式的maven依赖引入，项目配置和AT模式相同，可以参考seata+nacos实现AT模式分布式事务第2节，完成代码到git上下载，TCC模式在tcc分支下，脚本在script目录下（和AT模式的数据库脚本一致，选择一个执行即可）

https://gitee.com/WylLoveX/seata.git

3、使用

TM端依然使用@GlobalTransactional注解标识

```
@Override
@GlobalTransactional
public int insert(OrderForGoods orderForGoods){
    int ret = orderForGoodsMapper.insert(orderForGoods);
    productServiceApi.reduceStock( id: "1", num: 1);
    bankServiceApi.reduceAccount( id: "1", num: 10);
    return ret;
}
```

RM端需要单独标识出来

@LocalTCC将资源注册到TC

@TwoPhaseBusinessAction 将资源标时为二阶段提交的try阶段，这里需要在指定二阶段对应的Confirm和Cancel

@BusinessActionContextParameter标时这个一个二阶段参数，后续可以在BusinessActionContex中获取

```
/**
 * @author Mr.Wen
 * @version 1.0
 * @date 2021-10-25 14:03
 */
@LocalTCC
public interface BankUserService {
    @TwoPhaseBusinessAction(name = "reduceAccount",commitMethod = "commitReduceAccount",rollbackMethod = "rollbackReduceAccount")
    int reduceAccount(
            @BusinessActionContextParameter(paramName ="id") String id,
            @BusinessActionContextParameter(paramName ="num") Integer num);
    boolean commitReduceAccount(BusinessActionContext context);
    boolean rollbackReduceAccount(BusinessActionContext context);
}
```

提交的时候，我们减少了账户的金额，此时发生故障，在二阶段的cancel中自定义回滚（这里将金额为原来的值），若正常提交，就执行二阶段的confirm

```
/**
 * @author Mr.Wen
 * @version 1.0
 * @date 2021-10-25 14:04
 */
@Service
public class BankUserServiceImpl implements BankUserService {
    @Resource
    private BankUserMapper bankUserMapper;

    @Override
    public int reduceAccount(String id,Integer num) {
        int effectNum = bankUserMapper.reduceAccount(id,num);
        if(effectNum == 0){
            throw new RuntimeException("减库存失败");
        }
        int i= 1/0;
        return effectNum;
    }

    @Override
    public boolean commitReduceAccount(BusinessActionContext context) {
        System.out.println("xid="+context.getXid()+"提交成功");
        // todo 如果有预留资源提交预留资源
        return true;
    }

    @Override
    public boolean rollbackReduceAccount(BusinessActionContext context) {
```

```
        System.out.println("xid="+context.getXid()+"提交失败");
        // todo 这里需要执行补偿逻辑,
        String id = (String) context.getActionContext("id");
        Integer num = (Integer) context.getActionContext("num");

        bankUserMapper.decreaseAccount(id,num);
        return true;
    }
}
=
```