启动事务管理中心：TX-Manager

使用教程

1. 启动redis服务
2. 启动eureka服务
3. 配置bootstrap.yml文件下的eureka服务地址

```
eureka:
  instance:
    hostname: ${hostname:localhost}
    preferIpAddress: true
  server:
    peerEurekaNodesUpdateIntervalMs: 60000
    enableSelfPreservation: false
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
    healthcheck:
      enabled: true
    eurekaServiceUrlPollIntervalSeconds: 60

endpoints:
  health:
    sensitive: false
```

4. 配置application.properties文件

```
##########################################txmanager-
start##################################################
#服务端口
server.port=8899

#tx-manager不得修改
spring.application.name=tx-manager
```

```
spring.mvc.static-path-pattern=/**
spring.resources.static-locations=classpath:/static/
########################################txmanager-
end##################################################



########################################redis-
start#################################################
#redis 配置文件，根据情况选择集群或者单机模式

##redis 集群环境配置
##redis cluster
#spring.redis.cluster.nodes=127.0.0.1:7001,127.0.0.1:7002,127.0.0.1
:7003
#spring.redis.cluster.commandTimeout=5000

##redis 单点环境配置
#redis
#redis主机地址
spring.redis.host=127.0.0.1
#redis主机端口
spring.redis.port=6379
#redis链接密码
spring.redis.password=
spring.redis.pool.maxActive=10
spring.redis.pool.maxWait=-1
spring.redis.pool.maxIdle=5
spring.redis.pool.minIdle=0
spring.redis.timeout=0
########################################redis-
end##################################################



########################################LCN-
start#################################################
#业务模块与TxManager之间通讯的最大等待时间（单位：秒）
#通讯时间是指：发起方与响应方之间完成一次的通讯时间。
#该字段代表的是Tx-Client模块与TxManager模块之间的最大通讯时间，超过该时间未响
应本次请求失败。
tm.transaction.netty.delaytime = 5

#业务模块与TxManager之间通讯的心跳时间（单位：秒）
tm.transaction.netty.hearttime = 15
```

```
#存储到redis下的数据最大保存时间（单位：秒）
#该字段仅代表的事务模块数据的最大保存时间，补偿数据会永久保存。
tm.redis.savemaxtime=30

#socket server Socket对外服务端口
#TxManager的LCN协议的端口
tm.socket.port=9999

#最大socket连接数
#TxManager最大允许的建立连接数量
tm.socket.maxconnection=100

#事务自动补偿（true:开启，false:关闭）
# 说明：
# 开启自动补偿以后，必须要配置 tm.compensate.notifyUrl 地址，仅当tm.compensate.notifyUrl 在请求补偿确认时返回success或者SUCCESS时，才会执行自动补偿，否则不会自动补偿。
# 关闭自动补偿，当出现数据时也会 tm.compensate.notifyUrl 地址。
# 当tm.compensate.notifyUrl 无效时，不影响TxManager运行，仅会影响自动补偿。
tm.compensate.auto=false

#事务补偿记录回调地址(rest api 地址，post json格式)
#请求补偿是在开启自动补偿时才会请求的地址。请求分为两种：1.补偿决策，2.补偿结果通知，可通过通过action参数区分compensate为补偿请求、notify为补偿通知。
#*注意当请求补偿决策时，需要补偿服务返回"SUCCESS"字符串以后才可以执行自动补偿。
#请求补偿结果通知则只需要接受通知即可。
#请求补偿的样例数据格式:
#{"groupId":"TtQxTwJP","action":"compensate","json":"
{\"address\":\"133.133.5.100:8081\",\"className\":\"com.example.demo.service.impl.DemoServiceImpl\",\"currentTime\":1511356150413,\"data\":\"C5IBLWNvbS5leGFtcGxlLmRlbW8uc2VydmljZS5pbXBsLkRlbW9TZXJ2aWNlSW1wbAwSBHNhdmUbehBqYXZhLmxhbmcuT2JqZWN0GAAQARwjeg9qYXZhLmxhbmcuQ2xhc3MYABABJCo/cHVibGljIGludCBjb20uZXhhbXBsZS5kZW1vLnNlcnZpY2UuaW1wbC5EZW1vU2VydmljZUltcGwuc2F2ZSgp\",\"groupId\":\"TtQxTwJP\",\"methodStr\":\"public int
com.example.demo.service.impl.DemoServiceImpl.save()\",\"model\":\"demo1\",\"state\":0,\"time\":36,\"txGroup\":
{\"groupId\":\"TtQxTwJP\",\"hasOver\":1,\"isCommit\":0,\"list\":
[{\"address\":\"133.133.5.100:8899\",\"isCommit\":0,\"isGroup\":0,\"kid\":\"wnlEJoSl\",\"methodStr\":\"public int
com.example.demo.service.impl.DemoServiceImpl.save()\",\"model\":\"demo2\",\"modelIpAddress\":\"133.133.5.100:8082\",\"modelName\":\"/133.133.5.100:64153\",\"notify\":1,\"uniqueKey\":\"bc13881a5d2ab2ace89ae5d34d608447\"}],\"nowTime\":0,\"startTime\":1511356150379,\"state\":1},\"uniqueKey\":\"be6eea31e382f1f0878d07cef319e4d7\"}"}
```

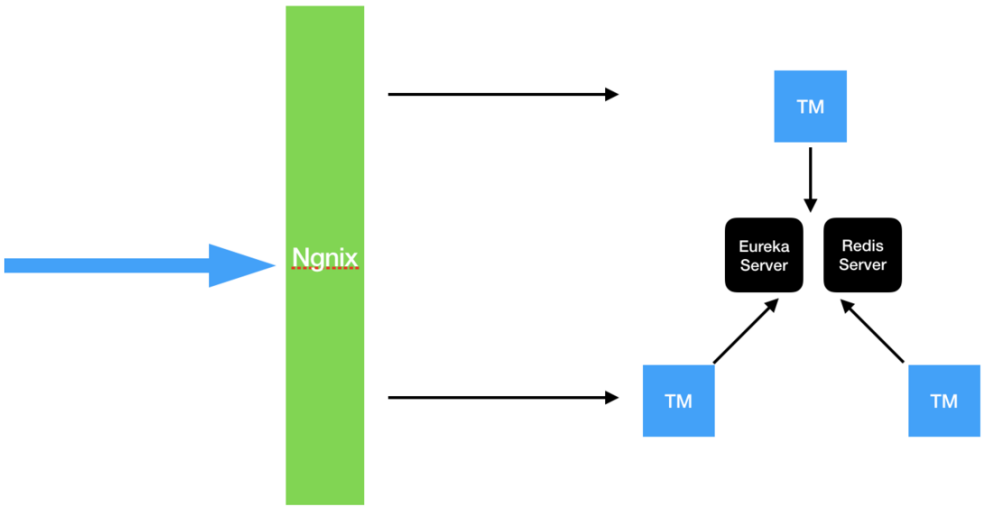
```
#请求补偿的返回数据样例数据格式:
#SUCCESS
#请求补偿结果通知的样例数据格式:
#{"resState":true,"groupId":"TtQxTwJP","action":"notify"}
tm.compensate.notifyUrl=http://ip:port/path

#补偿失败，再次尝试间隔（秒），最大尝试次数3次，当超过3次即为补偿失败,失败的数据依旧还会存在TxManager下。
tm.compensate.tryTime=30

#各事务模块自动补偿的时间上限(秒)
#指的是模块执行自动超时的最大时间，该最大时间若过段会导致事务机制异常，该时间必须要模块之间通讯的最大超过时间。
#例如，若模块A与模块B，请求超时的最大时间是5秒，则建议改时间至少大于5秒。
tm.auto.compensate.limit=20
########################################LCN-end##################################################
```

5. 启动ace-transaction-manager.jar。若从源码下则需要启动运行TxManagerApplication

然后访问 `http://127.0.0.1:8899/index.html`

TxManager集群说明

原理图:



1. 配置Redis集群
2. 配置TM服务 部署多分tm，然后修改各个配置文件的 eureka.client.serviceUrl.defaultZone，指向各服务的tm地址中间用"，"分割。

例如：
http://192.168.1.101:8761/eureka/,http://192.168.1.102:8761/eureka/,http://192.168.1.103:8761/eureka/

3. 配置nginx负载均衡

负载均衡TM服务。

4. 修改各事务模块的tx.properties配置文件url地址参数为nginx负载均衡的地址。

分布式事务开发

- 利用服务Cli，生成工程
- 配置（`bootstrap.yml`)指向事务中心

```
# txmanager地址
tm:
  manager:
    url: http://127.0.0.1:8899/tx/manager/
```

- 事务管控 所有的跟事务相关的代码，写在`biz`的路劲下，并且类名以`Biz`结尾，如果有个性化，可以去interceptor包下修改拦截器的扫描配置
- 在事务管控代码源头加入如下注解

```
@TxTransaction //在整个事务的开头加
@Transactional // 必须强制加
public void test(){
    prodFeign.test();
    Account account = mapper.selectByPrimaryKey(1);
    Integer balance = account.getBalance();
    account.setBalance(balance-100);
    mapper.updateByPrimaryKey(account);
    int i =  1/0;
}
```

- 更多例子可以参考：`ace-transaction-demo`工程