seata 下载

http://seata.io/en-us/blog/download.html

选择自己的版本下载

seata和nacos整合开发分布式

第一步 配置seata的服务器端的数据库

0、全局事务会话信息由3块内容构成，全局事务-->分支事务-->全局锁，对应表global_table、branch_table、lock_table
1、建立一个数据库名字随意(我起的名字——seata)用来做seata服务端的库，存储全局事务的会话信息
2、拿到服务端数据库的脚本文件执行并且建立表第0步三张表
https://github.com/seata/seata/tree/1.2.0/script/server/db
上面地址可以拿到数据库脚本，选择mysql然后自己执行

第二步 修改服务端启动包的文件

1、启动包: seata-->conf-->file.conf，修改store.mode="db"
2、修改自己的db连接信息
这些信息是存在file.conf当中的也就是seata启动的时候会去读取这个配置文件；如果我们使用nacos可以把这些信息放到nacos的注册中心；从而实现动态更新；
如果你需要把这些信息放到nacos配置中心就需要修改seata-->conf-->registy.conf文件当中的注册中心和配置中心的信息，修改成为nacos；（为什么需要修改config和register呢？因为如果你的seata服务器想要去配置中心读取配置，那么一定到得把自己注册到nacos；所以registy.conf当中需要配置注册中心的地址也需要配置配置中心的地址）
这样我们如果后面启动seata就可以看到他是作为了一个nacos的客户端注册到了nacos的注册中心的；记住这点seata已经可以作为客户端注册到nacos了；

第三步 把配置信息上传到nacos配置中心

1、启动nacos
2、https://github.com/seata/seata/tree/1.2.0/script/config-center到这个地址上面获取config.txt，然后把config.txt当道idea当中去编辑；推荐使用idea编辑；因为可能有编码原因；保留自己想要的信息我这里给出的是精简后的，你们需要自己对应修改自己的信息；主要是数据库配置信息
注意这些信息是服务器端和客户端都要使用的；
由于上面我们已经把seata注册到了nacos；所以他的file.conf当中的信息可以直接从nacos读取；也就是下面我们配置的信息；换句话说如果你配置了seata作为nacos的一个客户端去读取配置那么file.conf可以不用配置了；这两步是重复的；这也是网上很多资料没有说明的；
换成大白话的意思就是你如果配置了registy.conf那么file.conf当中的信息基本无效——都是从配置中心读取；甚至可以删了file.conf；你们可以自己测试；如果你不配置registy.conf，那么seata就会从file.conf当中读取配置；所以file.conf和registy.conf其实只需要配置一个；

精简后的配置如下

```
#事务分组--my_test_tx_group 这值会在我们客户端对应，需要注意
service.vgroupMapping.my_test_tx_group=default
service.default.grouplist=127.0.0.1:8091
store.mode=db
store.db.datasource=druid
```

```
store.db.dbType=mysql
store.db.driverClassName=com.mysql.jdbc.Driver
store.db.url=jdbc:mysql://ip::3306/seata?useUnicode=true
store.db.user=username
store.db.password=password
store.db.minConn=5
store.db.maxConn=30
store.db.globalTable=global_table
store.db.branchTable=branch_table
store.db.queryLimit=100
store.db.lockTable=lock_table
store.db.maxWait=5000
```

那么这些精简后的配置如何传到nacos呢？
https://github.com/seata/seata/tree/1.2.0/script/config-center
从上面这个地址下载nacos文件下面的nacos-config.sh文件然后执行
sh 你下载后的路径/nacos-config.sh；当然如果你的nacos地址断后不上默认的，需要修改naocs-config.sh当中指定你的路径；也可以在sh命令后面指定；
https://github.com/seata/seata/tree/1.2.0/script/config-center这个地址里面有个readme文件有说明

执行完成之后，你可以看到nacos的配置中心上面多了很多配置；注意这个时候seata服务器用的就是这些配置了；你可以修改一个错误的试试是不能启动seata服务器的

第四步 启动seata服务器

讲道理可以启动成功——注意不要用jdk11；我课上测试过有问题

第五步 建立微服务项目----以spring cloud为例

maven引入seata-spring-boot-starter、spring-cloud-alibaba-seata这两个jar
其中seata-spring-boot-starter选择你对应的seata版本比如1.2；但是spring-cloud-alibaba-seata这个jar当中自动依赖了seata-spring-boot-starter但是版本对应不上；比如spring-cloud-alibaba-seata当中依赖的seata-spring-boot-starter可能是0.9；所以需要剔除他；什么意思呢？

```
        <dependency>
            <groupId>io.seata</groupId>
            <artifactId>seata-spring-boot-starter</artifactId>
            <version>你的版本比如1.2.0</version>
        </dependency>

        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-alibaba-seata</artifactId>
            <!-- 这里需要剔除，因为上面我们已经引入了自己对应的版本 -->
            <exclusions>
                <exclusion>
                    <artifactId>io.seata</artifactId>
                    <groupId>seata-spring-boot-starter</groupId>
                </exclusion>
            </exclusions>
        </dependency>
```

当然咯分布式事务肯定需要是至少两个项目；所以你在两个项目当中都加上这些依赖；如果你是nacos的自然还需要加上其他jar；这里不在说了；

建好项目之后；写好代码。自己模拟一个分布式事务的场景
比如A调用B项目，在B项目里面操作数据库
加上A项目当中的AController当中的a()；调用B项目当中的BController当中的b()；b方法操作数据库；
那么则在A项目当中的AController的a()上面加上@GlobalTransactional这个注解

```
@RestController
public class A{
    @GlobalTransactional
    @GetMapping("xxxxxx")
    public string a(){
        通过feign调用b
    }
}
```

第六步 配置客户端的seata

如果你的项目已经成为了nacos的客户端，那么直接可以从nacos读取第三步当中上传到nacos的各种配置；那么如何读取呢？
首先得在客户端进行配置告诉seata客户端需要去nacos注册中心去读取seata的配置；可能有同学会问我们的微服务项目不上已经指定了？为什么还需要配置seata去读取呢？这个我在补录的视频里面有解释

配置客户端的yml读取nacos上的seata的配置

打开这个地址https://github.com/seata/seata/tree/1.2.0/script/client
找到spring文件夹，找到application.yml；这个yml是通用配置；你需要精简；我给出精简后的吧

```
seata:
  enabled: true
  application-id: applicationName
  tx-service-group: my_test_tx_group
  enable-auto-data-source-proxy: true
  use-jdk-proxy: false
  config:
    type: nacos

    nacos:
      namespace:
      serverAddr: localhost:你的端口
      group: SEATA_GROUP
      userName: ""
      password: ""

  registry:
    type: nacos
    nacos:
      application: seata-server
      server-addr: localhost:你的端口
      namespace:
      userName: ""
      password: ""
```

这个配置需要在你的每一个参与分布式事务的项目当中加上——直接写到项目的yml当中就可以了。这个配置的意思就是让我们的seata客户端直接从配置中心拉取配置；

最后一步

需要在你的客户端操作的数据库当中建立undo_log表；这个表用来实现sql反向补偿也就是回滚的信息

这个表的见表语句——https://github.com/seata/seata/tree/1.2.0/script/client/at/db
注意是建立在你的微服务所对应的库中；比如你的B服务链接了X库；那么则在x库中建立这个表；如果你的A服务链接了Y库；则Y库也需要这个表

OK 开始测试吧