API:

<https://spring-cloud-alibaba-group.github.io/github-pages/greenwich/spring-cloud-alibaba.html>

在common的module中的pom文件里定义出一个使用springcloud Alibaba的版本，然后这样我们所有引入springcloud的依赖都不需要version标签来定义版本

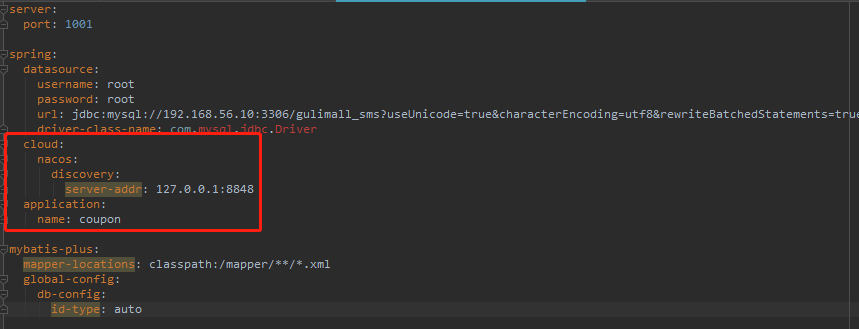
<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>com.alibaba.cloud</groupId>  
 <artifactId>spring-cloud-alibaba-dependencies</artifactId>  
 <version>2.1.0.RELEASE</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
</dependencyManagement>

**nacos**

1. 本地启动nacos
2. 依赖（放入common）：

<!--nacos-->  
<dependency>  
 <groupId>com.alibaba.cloud</groupId>  
 <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>  
</dependency>

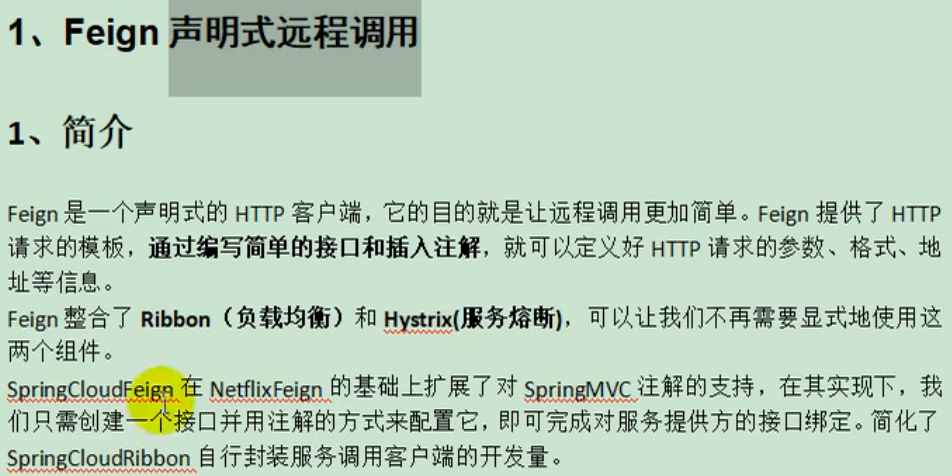
1. 将所有的子module的服务名称定义好，且配置所有子module指向nacos服务注册中心的ip地址和端口号：



4.在启动类上都加上@EnableDiscoveryClient（让当前微服务成为能向nacos注册的微服务）

**openFeign（大坑，如果有module有登录拦截,需要先去掉登录相关业务再进行调试，否则会报登录相关的  
错或者其他错）**

**1.基本配置：**



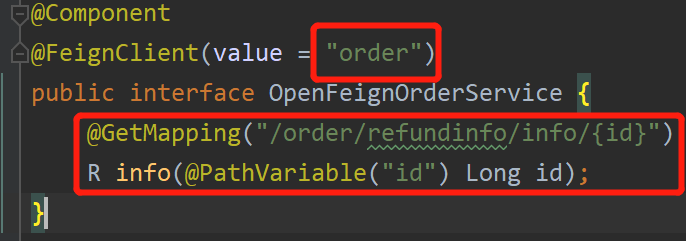
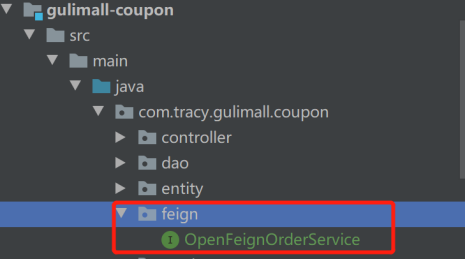


1.引入依赖  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
</dependency>

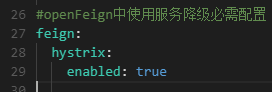
2.启动类加注解并指定扫描路径（注意，openFeign是需要配合注册中心，如nacos来一起搭配使用的，因为openFeign不需要配置远程调用的微服务端口号和ip，是直接从nacos中注册的微服务applicationname来发现服务的，此处埋下伏笔，openFeign猜测肯定内置有负载均衡功能）

@EnableFeignClients("com.tracy.gulimall.coupon.feign")

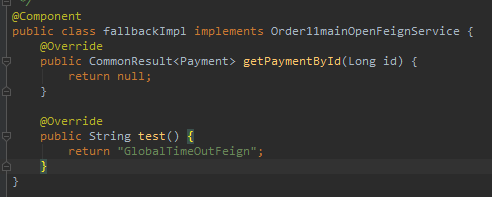
3.在相应的包下面创建一个远程调用的接口（注解component（让该接口被扫描到）和FeignClient（指定被调用的服务名称）一个不能少）



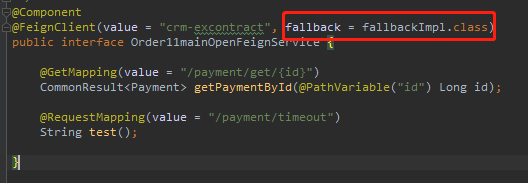
1. **服务降级：**
2. 配置使用feign时hystrix必须是enable



2.创建服务降级时需要执行的类，和方法，必须是openFeign远程调用接口的方法，如上，component必须要有



1. 引用fallback实现类

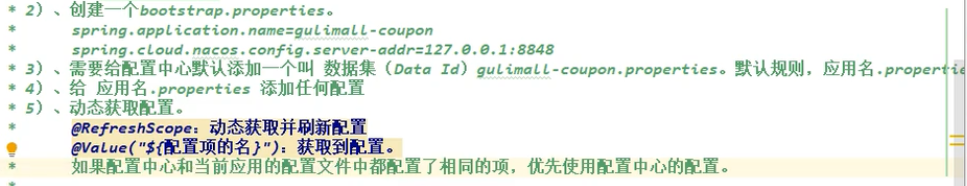


1. **open Feign中的自动Ribbon（默认轮询）：**

创建openFeign某一个消费者的集群即可完成自动轮询，注意spring.application.name必须一致

**Nacos-config功能**



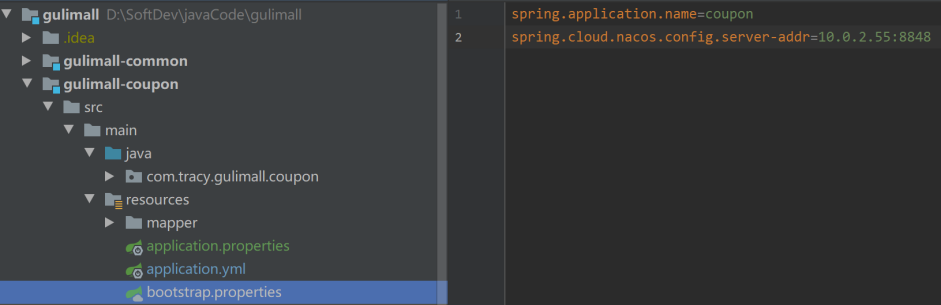


一（配置入门）：

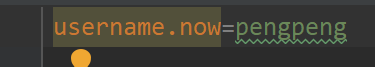
1.common引入依赖

<!--nacos-config-->  
<dependency>  
 <groupId>com.alibaba.cloud</groupId>  
 <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>  
</dependency>

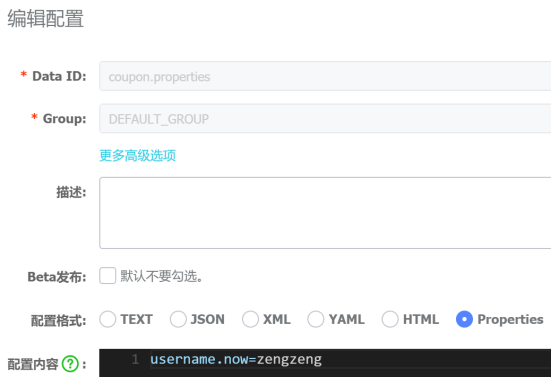
2.用于连接nacos且让nacos知道我的服务名是什么



3/4.给项目的resource路径下新增一个配置文件application.properties，内容随意，可以如下：



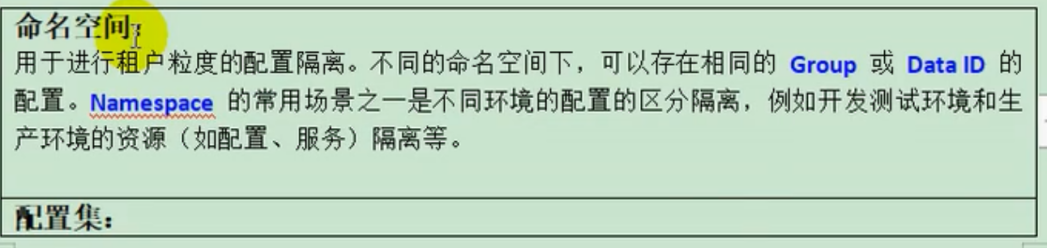
在nacos可视化界面的配置列表上添加一个名字为“服务名.properties”且content的内容为properties的文件，用于覆盖application.properties需要被修改的配置（在不重启项目的情况下），内容中的key和application.properties中的key保持一致，value不一样，用于测试nacos的配置文件在项目中是否生效，内容如下：

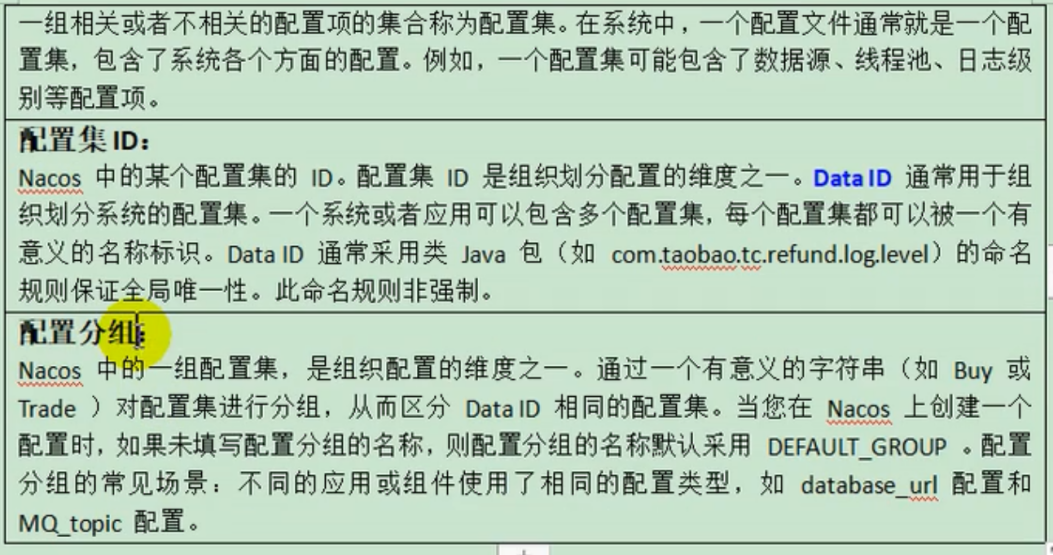


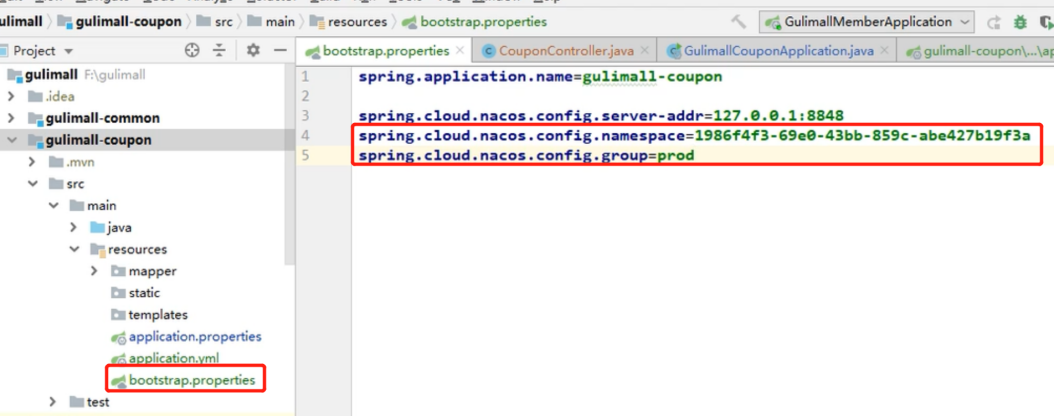
5.如下配置一个能获取配置文件的value值的接口，并配置好当前controller能实时的刷新到nacos-config更新的application.properties的值

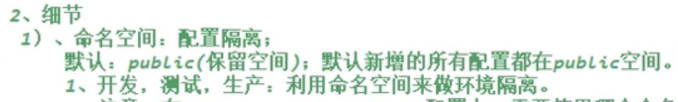


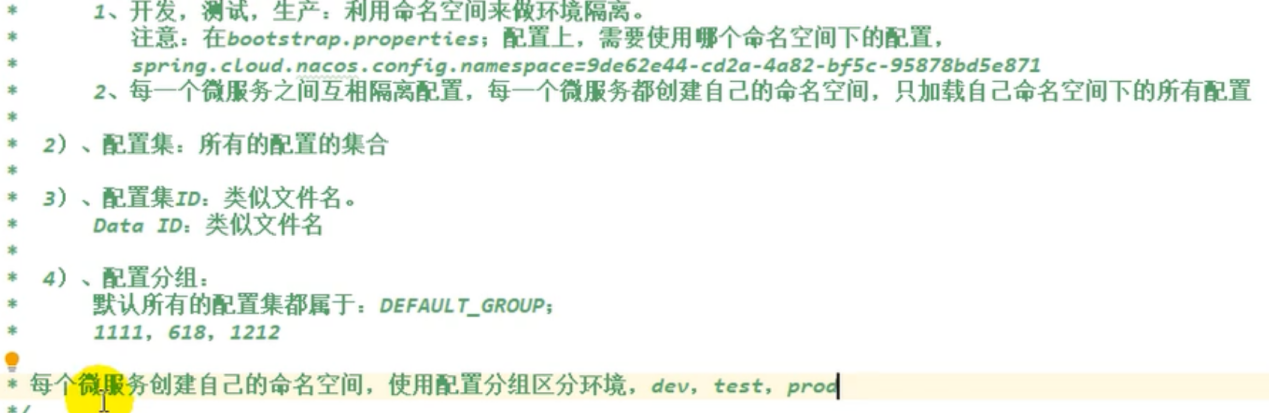
二（命名空间和分组的使用）：







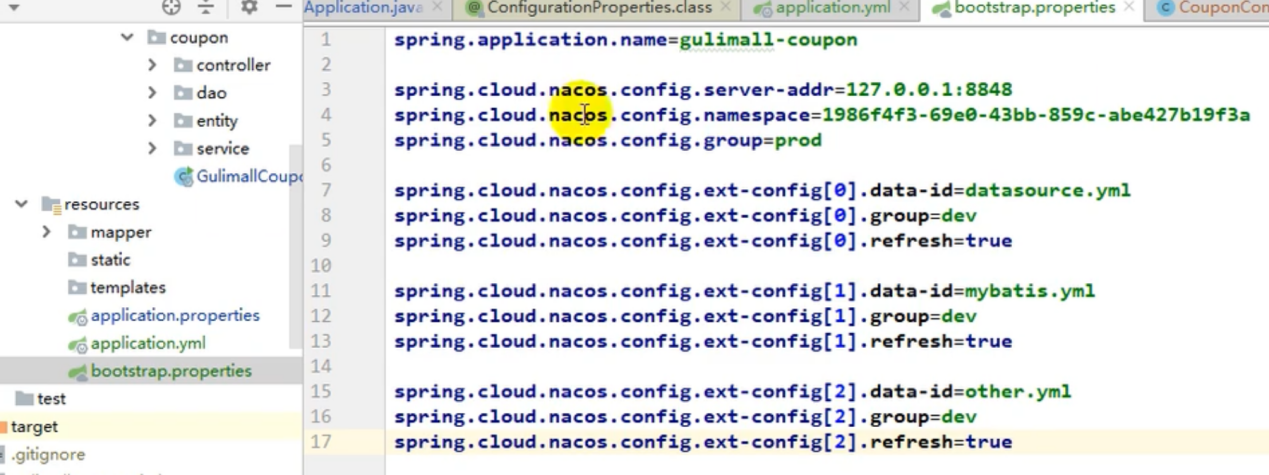


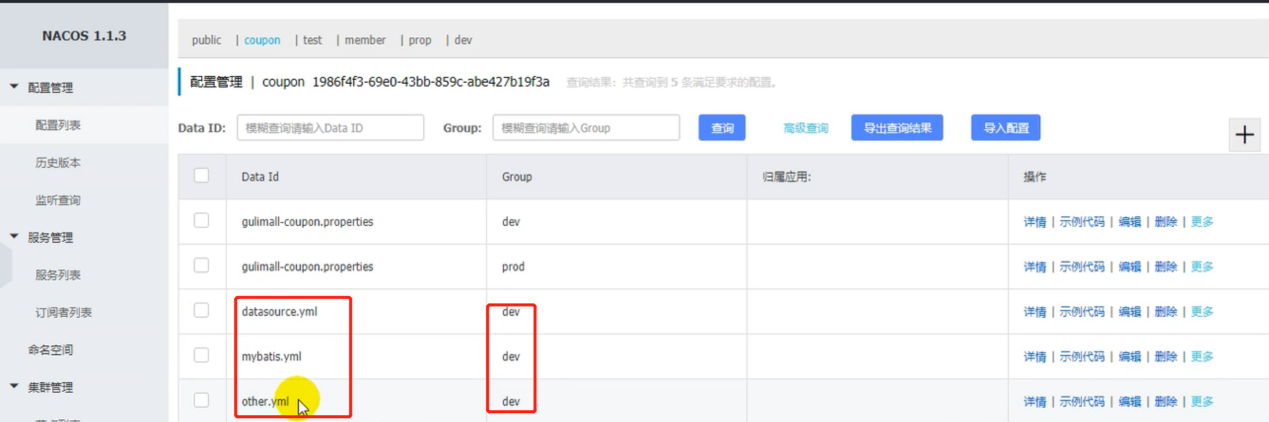


三（配置多个nacos配置文件（mybatis.yml+datasourceyml…））：

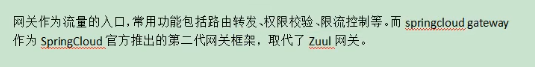
注意：spring.cloud.nacos.config.group配置的是application.properties在nacos中的配置文件的group；一般感觉都不需要配置这个，

spring.cloud.nacos.config.ext-config[0].group是application.yml在nacos中的配置的group





SpringCloudGetWay



雷总讲得比较混乱，这里直接看阳哥的

跨域解决方式：

1. 部署到同一nginx中；
2. 在网关中设置跨域的拦截器、

网关处如此设置即可（第二种方法）：



配置方式：

1.pom:

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <parent>  
 <artifactId>sefonsoft-crm</artifactId>  
 <groupId>com.sefonsoft.oa</groupId>  
 <version>1.0-SNAPSHOT</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>crm-getway9527</artifactId>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-gateway</artifactId>  
 <version>2.1.0.RELEASE</version>  
 <exclusions>  
 <exclusion>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-tomcat</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>3.1.0</version>  
 </dependency>  
 <!--一般基础配置类-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
</project>

2.yml(过滤配置仍有问题，需要纠正):

server:  
 port: 9527  
  
spring:  
 application:  
 name: cloud-gateway  
 cloud:  
 nacos:  
 discovery:  
 server-addr: 10.0.2.55:8848  
 gateway:  
 discovery:  
 locator:  
 enabled: true  
 routes:  
 - id: crm-excontract\_route  
 uri: http://10.0.2.55:7001/openFeign/test  
 *# - id: crm-excontract\_route  
 # uri: lb://crm-main* predicates:  
 - Path=/gateway  
 datasource:  
 url: jdbc:mysql://10.0.8.28:3306/sefonoa?useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai&allowMultiQueries=true  
 username: root  
 password: rootroot  
  
  
  
mybatis:  
 mapper-locations: classpath:mapper/\*.xml  
 configuration:  
 map-underscore-to-camel-case: true  
  
logging:  
 level:  
 com.sefonsoft.oa.dao: debug

3.过滤规则：

若有需要则自行添加