官网：

<https://github.com/alibaba/spring-cloud-alibaba/blob/master/README-zh.md>

Spring Cloud Alibaba 致力于提供微服务开发的一站式解决方案。此项目包含开发分布式应用微服务的必需组件，方便开发者通过 Spring Cloud 编程模型轻松使用这些组件来开发分布式应用服务。

依托 Spring Cloud Alibaba，您只需要添加一些注解和少量配置，就可以将 Spring Cloud 应用接入阿里微服务解决方案，通过阿里中间件来迅速搭建分布式应用系统。

## 主要功能

* **服务限流降级**：默认支持 WebServlet、WebFlux, OpenFeign、RestTemplate、Spring Cloud Gateway, Zuul, Dubbo 和 RocketMQ 限流降级功能的接入，可以在运行时通过控制台实时修改限流降级规则，还支持查看限流降级 Metrics 监控。
* **服务注册与发现**：适配 Spring Cloud 服务注册与发现标准，默认集成了 Ribbon 的支持。
* **分布式配置管理**：支持分布式系统中的外部化配置，配置更改时自动刷新。
* **消息驱动能力**：基于 Spring Cloud Stream 为微服务应用构建消息驱动能力。
* **分布式事务**：使用 @GlobalTransactional 注解， 高效并且对业务零侵入地解决分布式事务问题。。
* **阿里云对象存储**：阿里云提供的海量、安全、低成本、高可靠的云存储服务。支持在任何应用、任何时间、任何地点存储和访问任意类型的数据。
* **分布式任务调度**：提供秒级、精准、高可靠、高可用的定时（基于 Cron 表达式）任务调度服务。同时提供分布式的任务执行模型，如网格任务。网格任务支持海量子任务均匀分配到所有 Worker（schedulerx-client）上执行。
* **阿里云短信服务**：覆盖全球的短信服务，友好、高效、智能的互联化通讯能力，帮助企业迅速搭建客户触达通道。

[**Sentinel**](https://github.com/alibaba/Sentinel)：把流量作为切入点，从流量控制、熔断降级、系统负载保护等多个维度保护服务的稳定性。

[**Nacos**](https://github.com/alibaba/Nacos)：一个更易于构建云原生应用的动态服务发现、配置管理和服务管理平台。

[**Seata**](https://github.com/seata/seata)：阿里巴巴开源产品，一个易于使用的高性能微服务分布式事务解决方案。

## Nacos

一个更易于构建云原生应用的动态服务发现、配置管理和服务管理平台。  
Nacos：Dynamic Naming and Configuration Service

Nacos就是 注册中心 + 配置中心的组合

Nacos = Eureka + Config + Bus

官网：<https://nacos.io/zh-cn/docs/what-is-nacos.html>

GitHub：<https://github.com/alibaba/nacos>



### 安装并运行Nacos

1. 本地Java8 + Maven环境已经OK
2. 先从官网下载Nacos
3. 解压安装包，直接运行bin目录下的startup.cmd
4. 运行成功后直接访问http://localhost:8848/nacos默认账号密码都是nacos

### Nacos作为服务注册中心

#### 服务提供者：

新建工程：cloudalibaba-provider-payment9001

父工程pom.xml

|  |
| --- |
| *<!--spring cloud alibaba 2.1.0.RELEASE-->* <**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**> |

本工程pom.xml

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

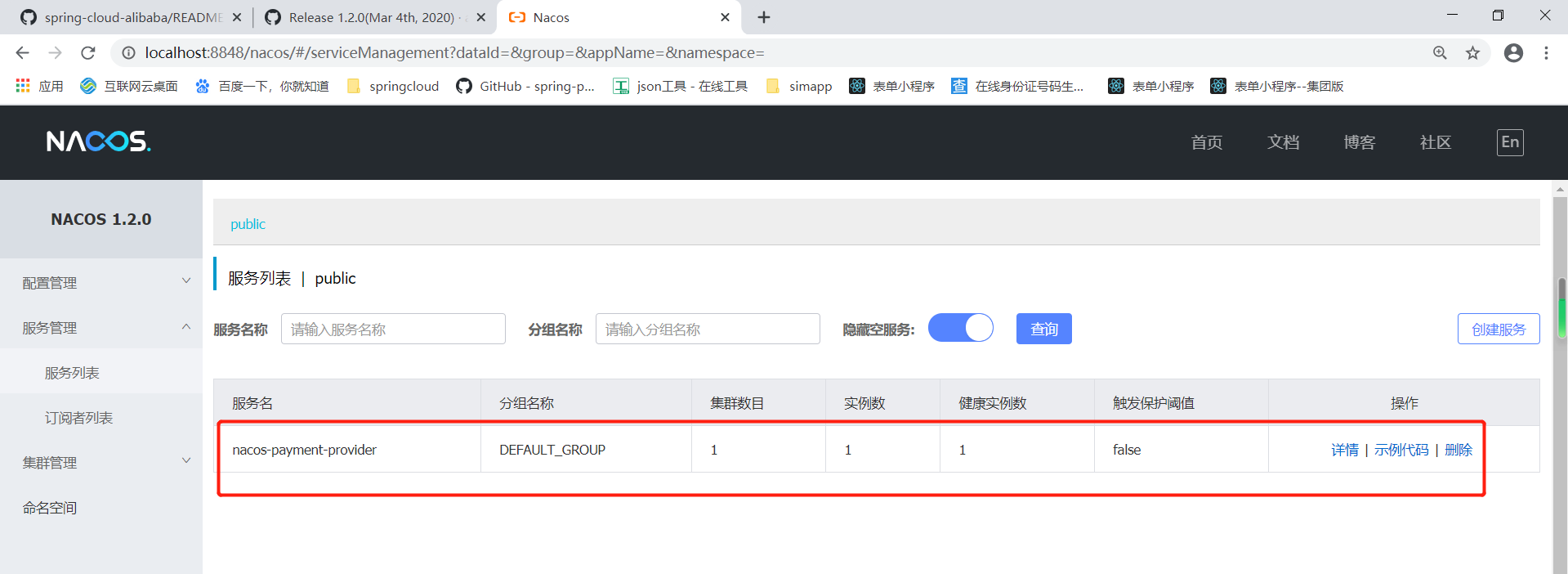
配置文件：application.yml

|  |
| --- |
| **server:  port:** 9001  **spring:  application:  name:** nacos-payment-provider  **cloud:  nacos:  discovery:  server-addr:** localhost:8848 *#配置Nacos地址* **management:  endpoints:  web:  exposure:  include: '\*'** *#监控* |

主配置类添加注解：

|  |
| --- |
| @SpringBootApplication @EnableDiscoveryClient |

启动成功，服务注册进nacos



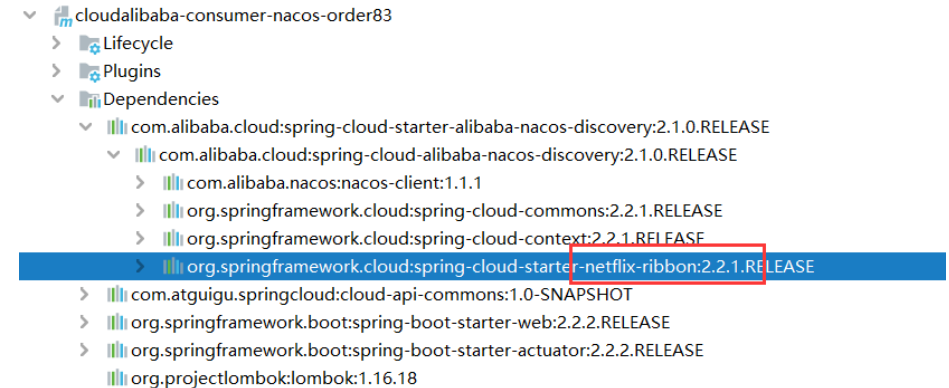
#### 服务消费者

新建工程：cloudalibaba-consumer-nacos-order83

其余依赖跟生产者一致

|  |
| --- |
| **server:  port:** 83  **spring:  application:  name:** nacos-order-consumer  **cloud:  nacos:  discovery:  server-addr:** localhost:8848  *#消费者将要去访问的微服务名称（成功注册进nacos的微服务提供者），在这配置了访问的服务，业务类就不用在定义常量了* **service-url:  nacos-user-service:** http://nacos-payment-provider |

Nacos自带负载均衡：



业务类引入RestTemplate

|  |
| --- |
| **package** com.demo.springcloud.config;  **import** org.springframework.cloud.client.loadbalancer.LoadBalanced; **import** org.springframework.context.annotation.Bean; **import** org.springframework.context.annotation.Configuration; **import** org.springframework.web.client.RestTemplate;  *//Nacos自带Ribbon，引入RestTemplate* @Configuration **public class** ApplicationContextConfig {   @Bean  @LoadBalanced *//RestTemplate结合Ribbon做负载均衡一定要加@LoadBalanced* **public** RestTemplate getRestTemplate() {  **return new** RestTemplate();  }  } |

controller层

|  |
| --- |
| @RestController **public class** OrderNacosController {   */\*  因为在yml中配置了service-url.nacos-user-service，  这里不需要再定义要访问微服务名常量，而是通过boot直接读出来  \*/* @Value(**"${service-url.nacos-user-service}"**)  **private** String **serverURL**;   @Resource  **private** RestTemplate **restTemplate**;   @GetMapping(**"/consumer/payment/nacos/{id}"**)  **public** String paymentInfo(@PathVariable(**"id"**) Long id){  **return restTemplate**.getForObject(**serverURL**+**"/payment/nacos/"**+id,String.**class**);  }  } |

#### 各个注册中心对比



Nacos 支持AP 和 CP模式的转换

C是所有节点在同一时间看到的数据是一致的；而A的定义是所有的请求都会受到响应。

何时选择使用何种模式？

一般来说，如果不需要存储服务级别的信息且服务实例是通过nacos-client注册，并能够保持心跳上报，那么就可以选择AP模式。当前主流的服务如SpringCloud和Dubbo服务，都适用与AP模式，AP模式为了服务的可能性而减弱了一致性，因此AP模式下只支持注册临时实例。

如果需要在服务级别编辑或存储配置信息，那么CP是必须，K8S服务和DNS服务则适用于CP模式。CP模式下则支持注册持久化实例，此时则是以Raft协议为集群运行模式，该模式下注册实例之前必须先注册服务，如果服务不出存在，则会返回错误。

#### CP和AP模式切换

用下面命令切换

curl -X PUT '$NACOS\_SERVER:8848/nacos/v1/ns/operator/switches?entry=serverMode&value=CP'

### Nacos作为服务配置中心

**官网：Nacos Spring Cloud 快速开始**

<https://nacos.io/zh-cn/docs/quick-start-spring-cloud.html>

1. 新建module cloudalibaba-config-nacos-client3377

2、pom.xml

|  |
| --- |
| <**dependency**>  <**groupId**>com.alibaba.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-alibaba-nacos-config</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>com.alibaba.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-alibaba-nacos-discovery</**artifactId**> </**dependency**> |

3、bootstrap.yml 和 application.yml

Nacos同springcloud-config一样，在项目初始化时，要保证先从配置中心进行配置拉取，拉取配置后，才能保证项目的正常启动。

SpringCloud中配置文件的加载是存在优先级顺序的，bootstrap优先级高于application

bootstrap.yml

|  |
| --- |
| *#bootstrap.xml* **server:  port:** 3377  **spring:  application:  name:** nacos-config-client  **cloud:  nacos:  discovery:  server-addr:** localhost:8848 *#Nacos服务注册中心地址* **config:  server-addr:** localhost:8848 *#Nacos作为配置中心地址* **file-extension:** yml *#指定yaml格式的配置 # group: TEST\_GROUP* **group:** DEFAULT\_GROUP  **namespace:** 58a0f6a1-dba6-403c-a953-0ce0a6d6200d  *# ${spring.application.name}-${spring.profile.active}.${spring.cloud.nacos.config.file.extension} # nacos-config-client-dev.yml* |

application.yml

|  |
| --- |
| **spring:  profiles:  active:** info *# active: dev #开发环境 # active: test #测试环境* |

4、主启动类添加注解

|  |
| --- |
| @SpringBootApplication @EnableDiscoveryClient |

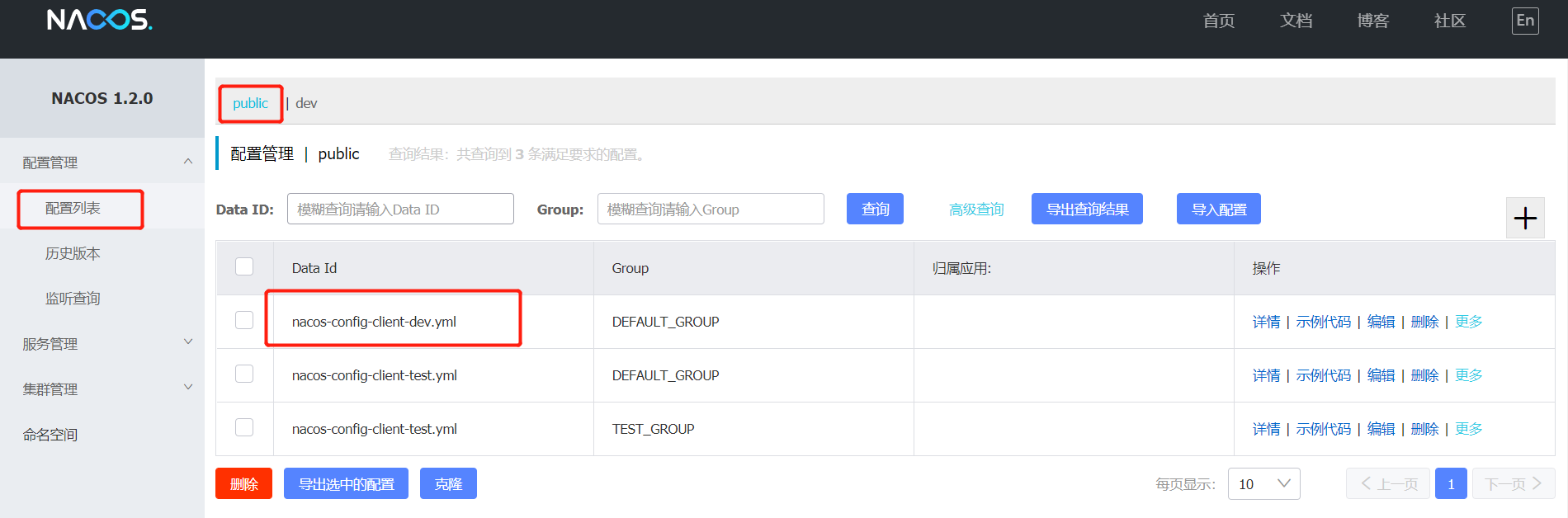
5、业务类

|  |
| --- |
| @RestController @RefreshScope *//SpringCloud原生注解 支持Nacos的动态刷新功能* **public class** ConfigClientController {   @Value(**"${config.info}"**) *//从配置中心读取*  **private** String **configInfo**;   @GetMapping(**"/config/info"**)  **public** String getConfigInfo() {  **return configInfo**;  }  } |

6、在Nacos中添加配置信息

nacos-config-client-dev.yml

|  |
| --- |
| config:  info: nacos-config center,version=2,test |



#### dataid组成规则：

在 Nacos Spring Cloud 中，dataId 的完整格式如下：

|  |
| --- |
| ${prefix}-${spring.profile.active}.${file-extension}  eg:  nacos-config-client-dev.yml  nacos-config-client-test.yml  nacos-config-client-info.yml |

* prefix 默认为 spring.application.name 的值，也可以通过配置项 spring.cloud.nacos.config.prefix来配置。
* spring.profile.active 即为当前环境对应的 profile，详情可以参考 [Spring Boot文档](https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-profiles.html#boot-features-profiles)。 **注意：当 spring.profile.active 为空时，对应的连接符 - 也将不存在，dataId 的拼接格式变成 ${prefix}.${file-extension}**
* file-exetension 为配置内容的数据格式，可以通过配置项 spring.cloud.nacos.config.file-extension 来配置。目前只支持 properties 和 yaml 类型。