Nacos的使用

注册中心

- Nacos可以作为注册中心使用,主要作用是进行服务的注册与发现

Nacos作为注册中心的使用步骤

下载nacos

- 根据系统下载即可

```
https://github.com/alibaba/nacos/releases
```

在POM文件中添加nacous相关依赖

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

注意：版本 2.1.x.RELEASE 对应的是 Spring Boot 2.1.x 版本。版本 2.0.x.RELEASE 对应的是 Spring Boot 2.0.x 版本，版本 1.5.x.RELEASE 对应的是 Spring Boot 1.5.x 版本

编写配置文件

```
spring:
  application:
    name: 应用名    # 必须配置应用名 因为它是构成 Nacos 配置管理 dataId字段的一部分
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 # 配置Nacos所在服务器的ip以及端口

```

开启服务发现注册

```
@SpringBootApplication
@EnableDiscoveryClient  //开启发现注册功能 将该应用注册到Nacos
public class NacosApplication {
    public static void main(String[] args) {
        SpringApplication.run(NacosApplication.class,args);
    }
}
```

配置管理

- 将Nacos作为配置中心,其主要作用是可以方便服务配置文件的管理,使得服务可以动态加载配置文件信息

Nacos作为配置中心的使用步骤

下载nacos

- 根据系统下载即可

```
https://github.com/alibaba/nacos/releases
```

POM文件中添加相关依赖

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

编写配置文件

- **注意:** 使用的配置文件是优先级最高的 bootstrap.properties文件,否则无法创建configService

```
# 配置nacos所在服务器的IP地址和端口
spring.cloud.nacos.config.server-addr=loacalhost:8848
# 配置应用名
spring.application.name=应用名
```

配置自动更新

```
@RestController
@RequestMapping("/config")
@RefreshScope  //启动配置动态刷新
public class ConfigController {

    @Value("${config.username}")
    private String username;

    @Value("${config.password}")
    private String password;

    @GetMapping
    public getUserInfo get() {
        return "username: "+username+"\n"+"password: "+password;
    }
```

```
16 }
```

注意:

```
当Nacos作为配置中心时,springboot启动后 会优先加载Nacos中的配置文件,配置文件默认为: 应用名.properties, 当配置中心没有对应文件时,会加载本地配置,若本地也没有 则会发生异常.
```

小扩展:

- 在配置文件中没有对应属性时,@Value注解 可以通过 ":" 来为属性赋予默认值

```
    @Value("${config.username:张三}")
    private String username;

    @Value("${config.password:123}")
    private String password;
```

Nacos配置中心的常见配置

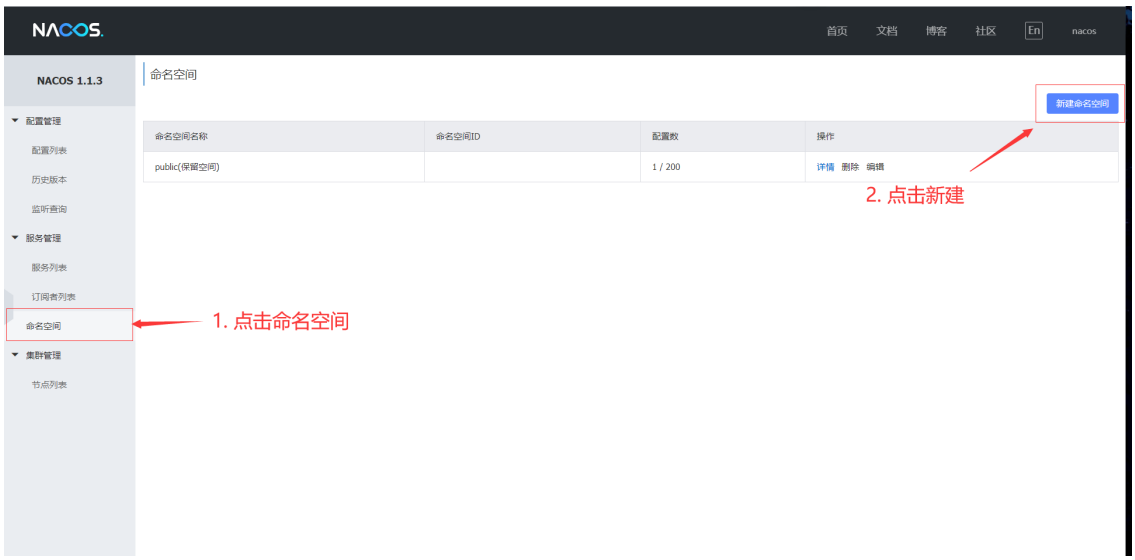
命名空间

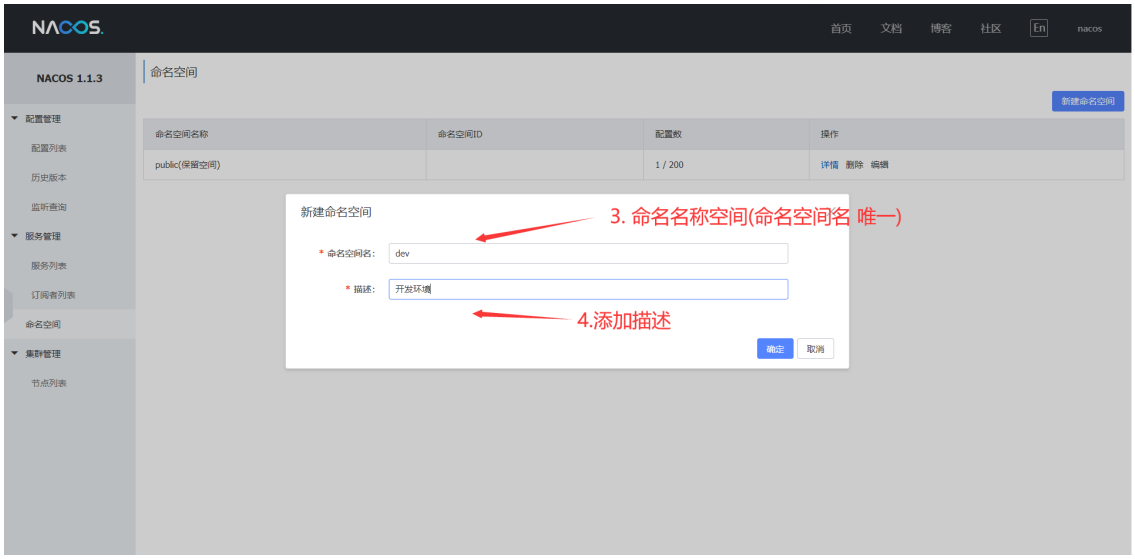
命名空间简介

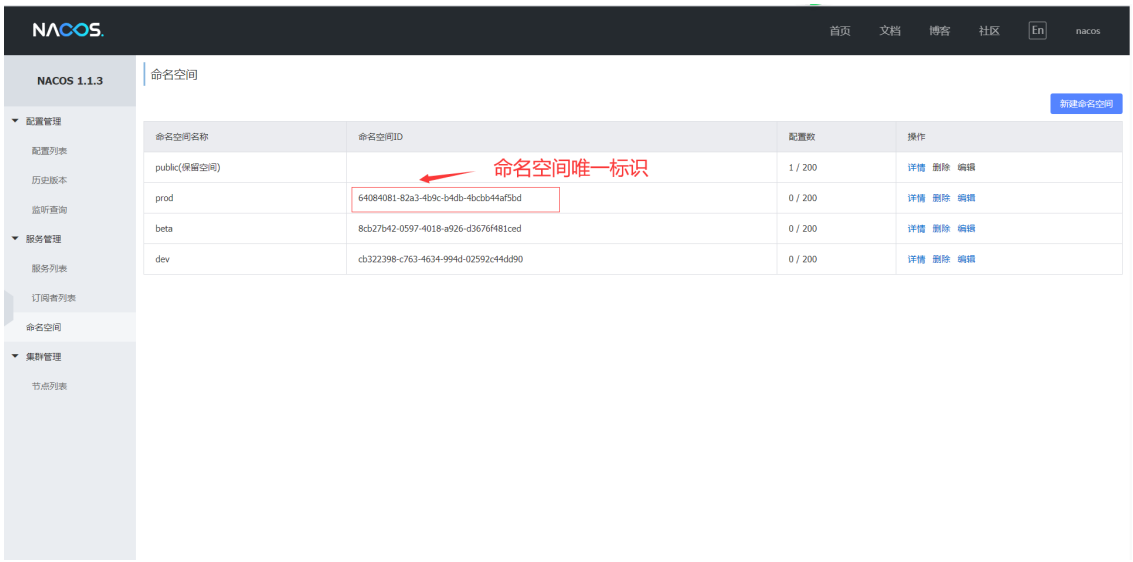
```
* 命名空间的主要作用是用于配置隔离. 不同的命名空间下的Group和Data ID互相隔离,也就是说某一命名空间下的Group或Data ID 只作用于其所属的命名空间; 命名空间默认为public(保留空间), 新增的配置信息默认在public空间内
```

命名空间的使用

- 不同环境下(dev,beta,prod)的配置文件隔离
- 使用步骤
  - 1.在Nacos控制台新建命名空间







- 2.在配置文件中指定当前使用的名称空间

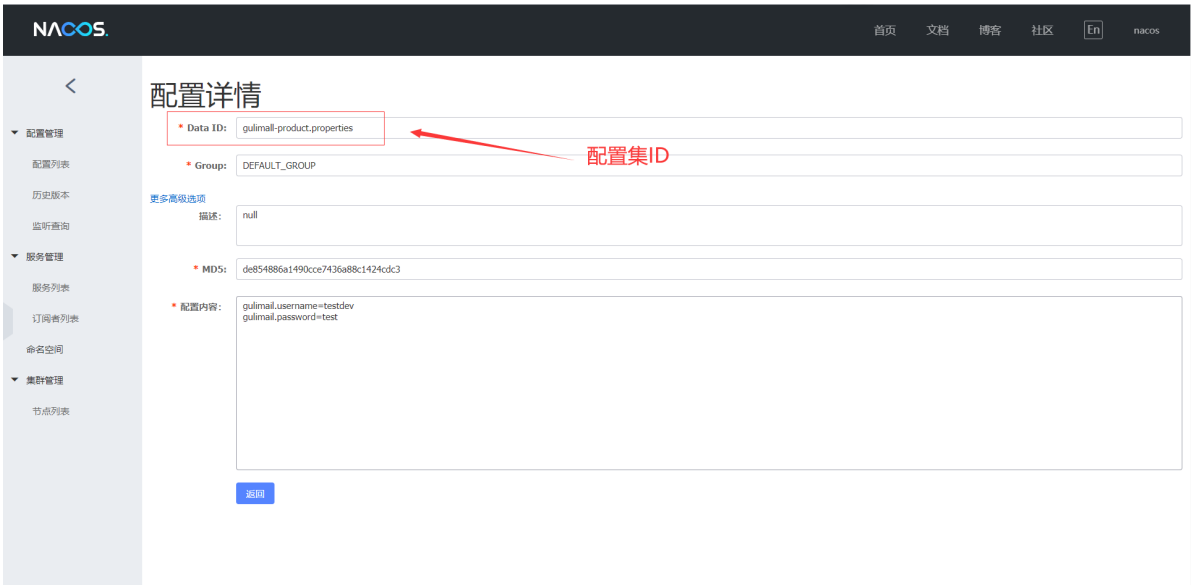
```
spring.cloud.nacos.config.server-addr=nacos所在服务器ip:8848
spring.application.name=服务名
spring.cloud.nacos.config.namespace=名称空间id  # 不配置默认使用public(保留空间)
```

配置集

- 所有配置(数据源, 端口等)的集合 就称之为配置集

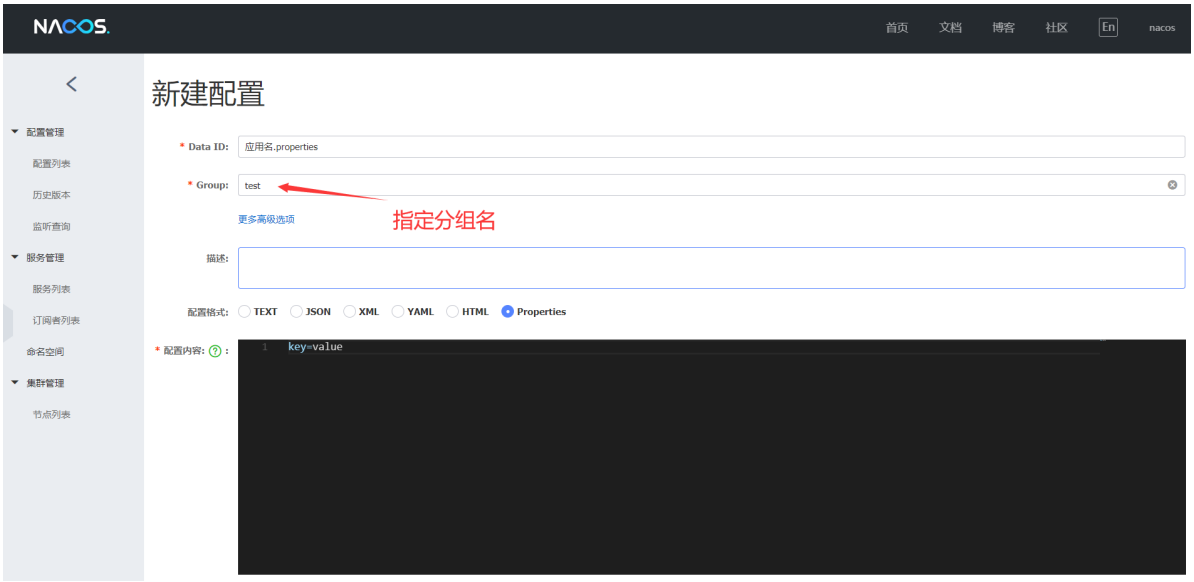
配置集ID

- 配置集id 相当于微服务中的配置文件名, 默认为: 服务名.properties ,在Nacos中对应属性为: Data Id



配置分组

- 配置分组指的是配置集 所属的分组,默认分组为DEFAULT_GROUP, 可以在创建配置时进行定制



- 在微服务的配置文件中指定分组

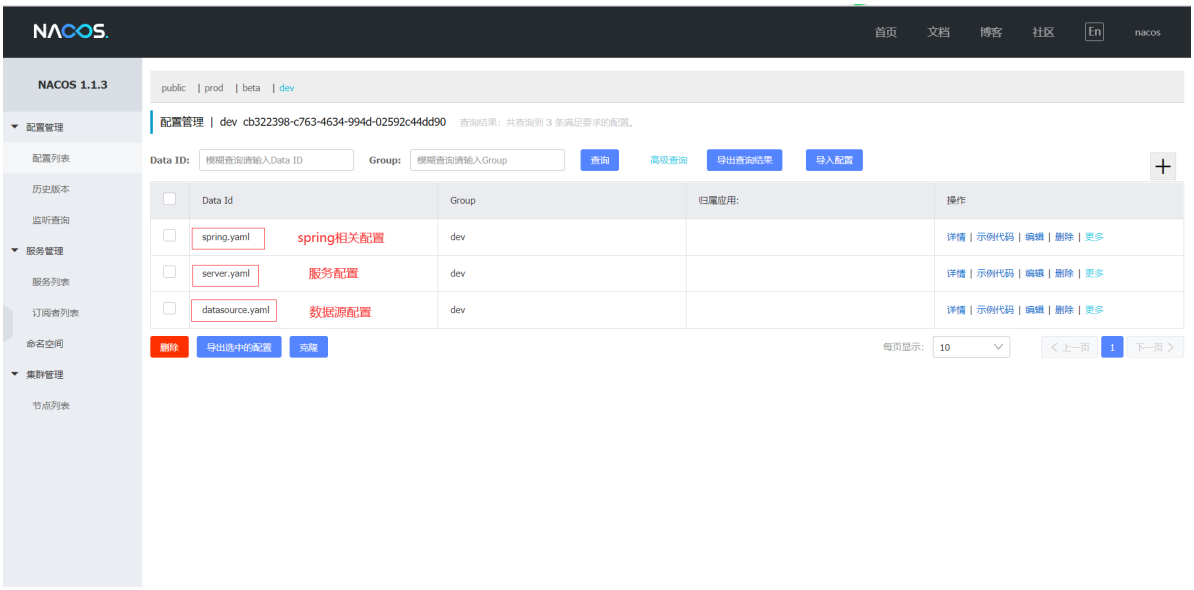
```
spring.cloud.nacos.config.server-addr=nacos所在服务器ip:8848
spring.application.name=服务名
spring.cloud.nacos.config.namespace=名称空间id  # 不配置默认使用public(保留空间)
spring.cloud.nacos.config.group=分组名 # 不配置默认使用 DEFAULT_GROUP
```

多配置集加载

- 多配置集的目的是为了加载多个配置文件,将配置文件进行抽取,方便管理

配置步骤:(以数据源,spring,server为例)

- 在Nacos中创建对应配置 (与配置文件一致)



- 在微服务的配置文件中添加配置

```
spring.cloud.nacos.config.server-addr=nacos所在服务器IP:8848
spring.application.name=服务名
spring.cloud.nacos.config.namespace=命名空间id
# spring.cloud.nacos.config.group=分组名  当分组中的配置与下列配置中有重叠部分时 分组配置优先级更高

# 服务配置
spring.cloud.nacos.config.ext-config[0].data-id=server.yaml
spring.cloud.nacos.config.ext-config[0].group=dev
spring.cloud.nacos.config.ext-config[0].refresh=true
# spring配置
spring.cloud.nacos.config.ext-config[1].data-id=spring.yaml
spring.cloud.nacos.config.ext-config[1].group=dev
spring.cloud.nacos.config.ext-config[1].refresh=true
# 数据源配置
spring.cloud.nacos.config.ext-config[2].data-id=datasource.yaml
spring.cloud.nacos.config.ext-config[2].group=dev
spring.cloud.nacos.config.ext-config[2].refresh=true
# 其他配置
spring.cloud.nacos.config.ext-config[3].data-id=other.yaml
spring.cloud.nacos.config.ext-config[3].group=dev
spring.cloud.nacos.config.ext-config[3].refresh=true

# spring.cloud.nacos.config.ext-config是一个集合 [i] 代表第几个配置 必须从0取值且连续
```