# 03-Nacos配置中心

## 一、配置中心介绍

**1、Spring Cloud Config**

Spring Cloud Config 为分布式系统的外部配置提供了服务端和客户端的支持方案。在配置的服务端您可以在所有环境中为应用程序管理外部属性的中心位置。客户端和服务端概念上的Spring Environment 和 PropertySource 抽象保持同步, 它们非常适合Spring应用程序，但是可以与任何语言中运行的应用程序一起使用。当应用程序在部署管道中从一个开发到测试直至进入生产时，您可以管理这些环境之间的配置，并确保应用程序在迁移时具有它们需要运行的所有内容。服务器存储后端的默认实现使用git，因此它很容易支持标记版本的配置环境，并且能够被管理内容的各种工具访问。很容易添加替代的实现，并用Spring配置将它们插入。

Spring Cloud Config 包含了Client和Server两个部分，server提供配置文件的存储、以接口的形式将配置文件的内容提供出去，client通过接口获取数据、并依据此数据初始化自己的应用。Spring cloud使用git或svn存放配置文件，默认情况下使用git。

**2、Nacos替换Config**

Nacos 可以与 Spring, Spring Boot, Spring Cloud 集成，并能代替 Spring Cloud Eureka, Spring Cloud Config。通过 Nacos Server 和 spring-cloud-starter-alibaba-nacos-config 实现配置的动态变更。

**（1）应用场景**

在系统开发过程中，开发者通常会将一些需要变更的参数、变量等从代码中分离出来独立管理，以独立的配置文件的形式存在。目的是让静态的系统工件或者交付物（如 WAR，JAR 包等）更好地和实际的物理运行环境进行适配。配置管理一般包含在系统部署的过程中，由系统管理员或者运维人员完成。配置变更是调整系统运行时的行为的有效手段。

如果微服务架构中没有使用统一配置中心时，所存在的问题：

- 配置文件分散在各个项目里，不方便维护

- 配置内容安全与权限

- 更新配置后，项目需要重启

nacos配置中心：系统配置的集中管理（编辑、存储、分发）、动态更新不重启、回滚配置（变更管理、历史版本管理、变更审计）等所有与配置相关的活动。

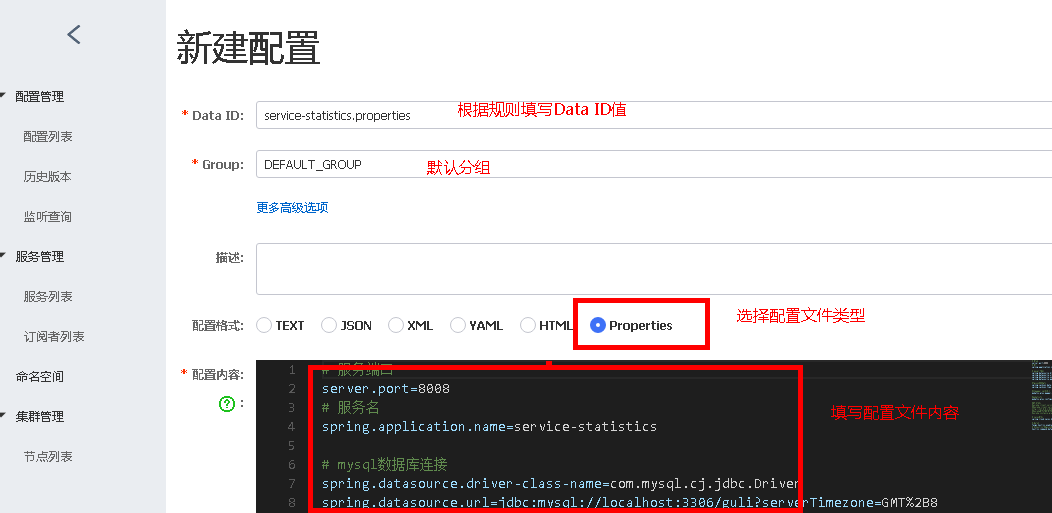
## 二、读取Nacos配置中心的配置文件

### 1、在Nacos创建统一配置文件

**（1）点击创建按钮**



**（2）输入配置信息**



**a）Data ID 的完整规则格式如下**

**${prefix}-${spring.profiles.active}.${file-extension}**

**- prefix** 默认为所属工程配置spring.application.name【服务名】 的值（即：nacos-provider），也可以通过配置项 spring.cloud.nacos.config.prefix来配置。

**- spring.profiles.active=dev**即为当前环境对应的 profile。 注意：当 spring.profiles.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【数据后缀】，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。

### 2、以service-statistics模块为例

**（1）在service中引入依赖**

*<*dependency*>*

*<*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-alibaba-nacos-config*</*artifactId*>  
</*dependency*>*

### （2）创建bootstrap.properties配置文件

#配置中心地址  
spring.cloud.nacos.config.server-addr=127.0.0.1:8848  
#spring.profiles.active=dev  
# 该配置影响统一配置中心中的dataId  
spring.application.name=service-statistics

### （3）把项目之前的application.properties内容注释，启动项目查看效果

### 3、补充：springboot配置文件加载顺序



其实yml和properties文件是一样的原理，且一个项目上要么yml或者properties，二选一的存在。推荐使用yml，更简洁。

bootstrap与application  
**（1）加载顺序**  
这里主要是说明application和bootstrap的加载顺序。

bootstrap.yml（bootstrap.properties）先加载  
application.yml（application.properties）后加载  
bootstrap.yml 用于应用程序上下文的引导阶段。

bootstrap.yml 由父Spring ApplicationContext加载。

父ApplicationContext 被加载到使用 application.yml 的之前。

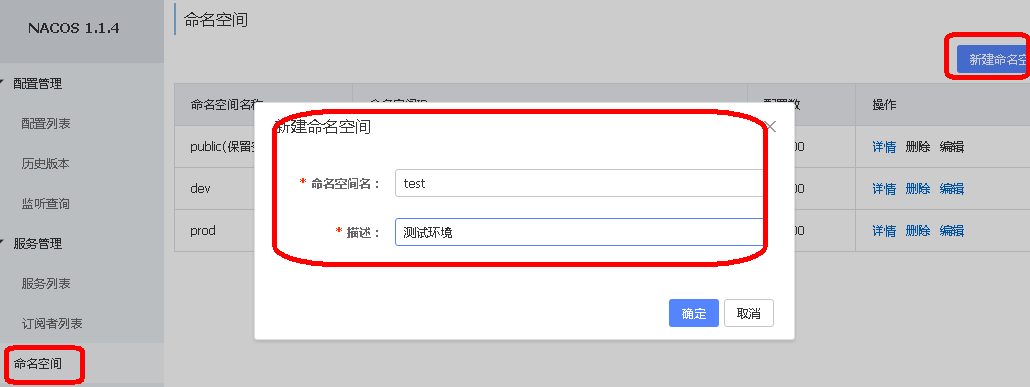
**（2）配置区别**  
bootstrap.yml 和application.yml 都可以用来配置参数。

bootstrap.yml 可以理解成系统级别的一些参数配置，这些参数一般是不会变动的。  
application.yml 可以用来定义应用级别的。

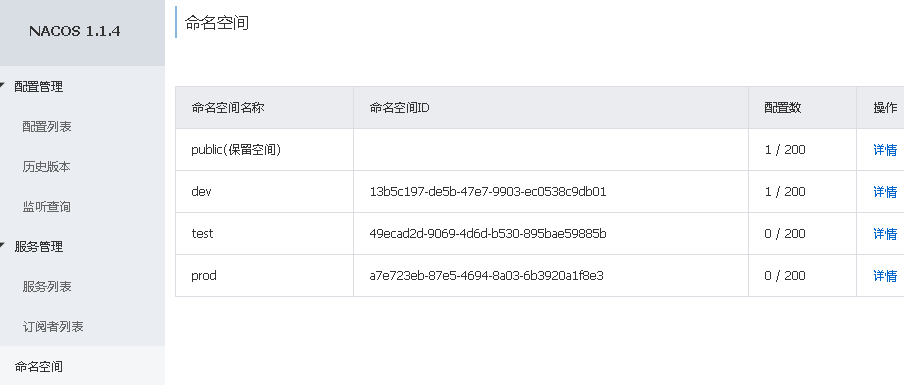
## 三、名称空间切换环境

在实际开发中，通常有多套不同的环境（默认只有public），那么这个时候可以根据指定的环境来创建不同的 namespce，例如，开发、测试和生产三个不同的环境，那么使用一套 nacos 集群可以分别建以下三个不同的 namespace。以此来实现多环境的隔离。

### 1、创建命名空间

****

**默认只有public，新建了dev、test和prod命名空间**



### 2、克隆配置

**（1）切换到配置列表：**

****

**可以发现有四个名称空间：public（默认）以及我们自己添加的3个名称空间（prod、dev、test），可以点击查看每个名称空间下的配置文件，当然现在只有public下有一个配置。**

**默认情况下，项目会到public下找 服务名.properties文件。**

**接下来，在dev名称空间中也添加一个nacos-provider.properties配置。这时有两种方式：**

**第一，切换到dev名称空间，添加一个新的配置文件。缺点：每个环境都要重复配置类似的项目**

**第二，直接通过clone方式添加配置，并修改即可。推荐**

****

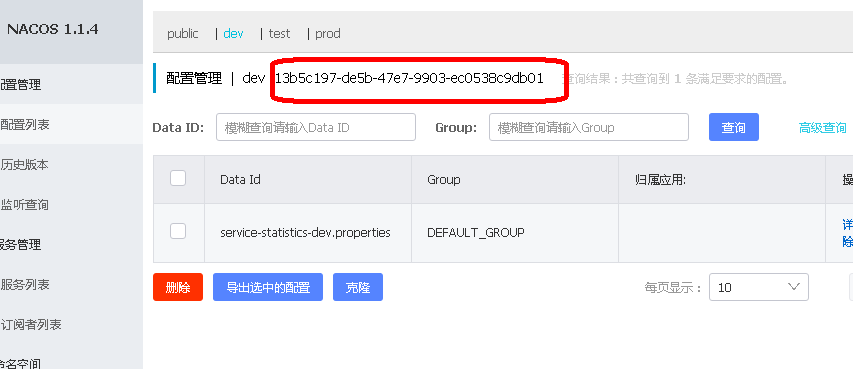
**点击编辑：修改配置内容，端口号改为8013以作区分**

****

**在项目模块中，修改bootstrap.properties添加如下配置**

spring.cloud.nacos.config.server-addr=127.0.0.1:8848  
spring.profiles.active=dev  
# 该配置影响统一配置中心中的dataId，之前已经配置过  
spring.application.name=service-statistic  
spring.cloud.nacos.config.namespace=1b731927-2025-43e7-b264-8e975b25796f

namespace的值为：

****

**重启服务提供方服务，测试修改之后是否生效**

## 四、多配置文件加载

在一些情况下需要加载多个配置文件。假如现在dev名称空间下有三个配置文件：service-statistics.properties、redis.properties、jdbc.properties



**添加配置，加载多个配置文件**

spring.cloud.nacos.config.server-addr=127.0.0.1:8848

spring.profiles.active=dev

# 该配置影响统一配置中心中的dataId，之前已经配置过

spring.application.name=service-statistics

spring.cloud.nacos.config.namespace=13b5c197-de5b-47e7-9903-ec0538c9db01

spring.cloud.nacos.config.ext-config[0].data-id=redis.properties

# 开启动态刷新配置，否则配置文件修改，工程无法感知

spring.cloud.nacos.config.ext-config[0].refresh=true

spring.cloud.nacos.config.ext-config[1].data-id=jdbc.properties

spring.cloud.nacos.config.ext-config[1].refresh=true