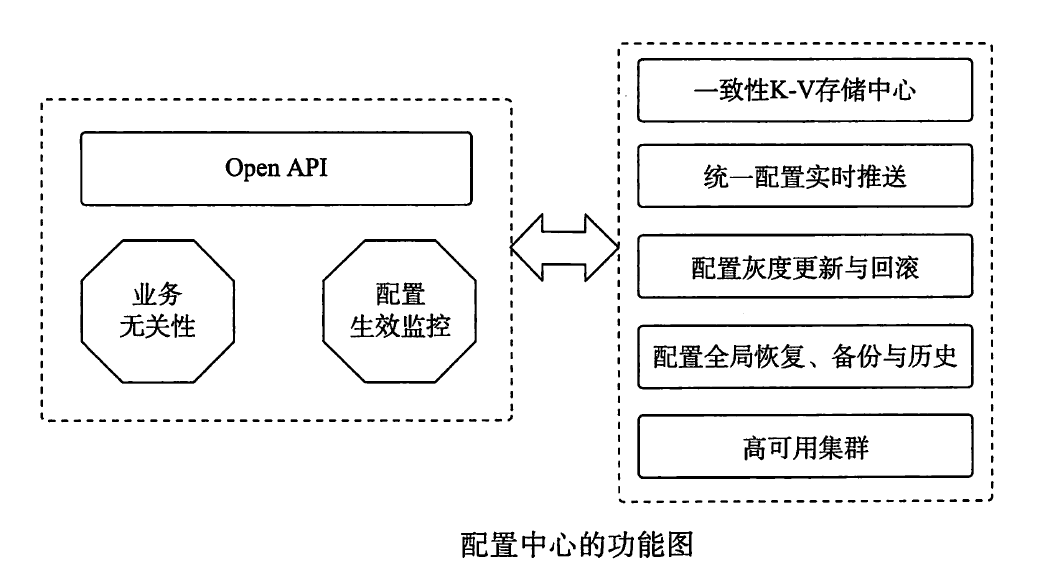
**Spring Cloud Config**

# 什么是配置中心

## 配置中心的由来

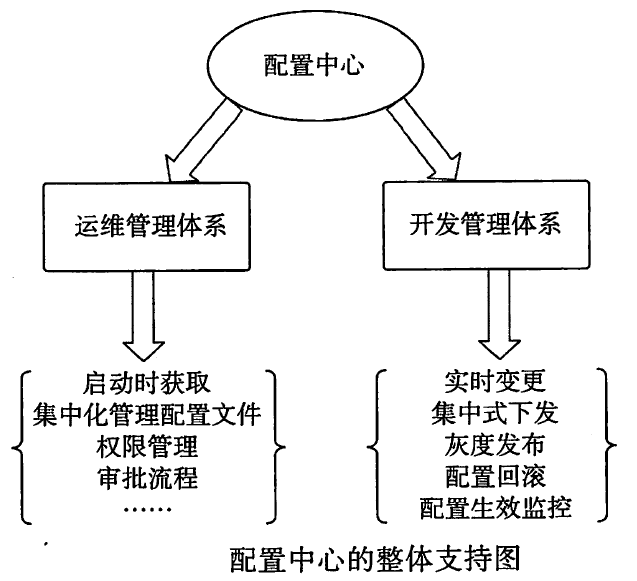
在集中式开发时代，配置文件已经基本足够了，因为那时配置的管理通常不会成为一个很大的问题。但是在互联网时代，应用都是分布式系统，部署在N台服务器上，想要去线上一台台地重启机器肯定不靠谱，并且维护成本也很高，所以配置中心应运而生。配置中心被用作集中管理不同环境(Dev, Test, Stage, Prod)和不同集群配置，以及在修改配置后将实时动态推送到应用上进行刷新。

## 配置中心应具备的功能

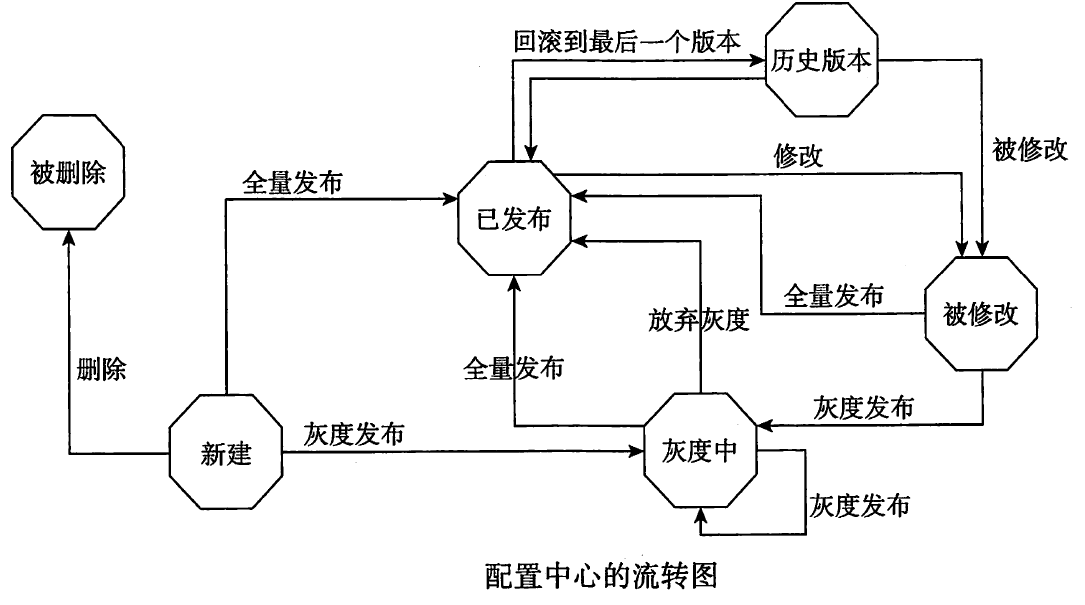


## 配置中心支撑体系

主要分为：运维管理体系、开发管理体系



## 配置中心流转

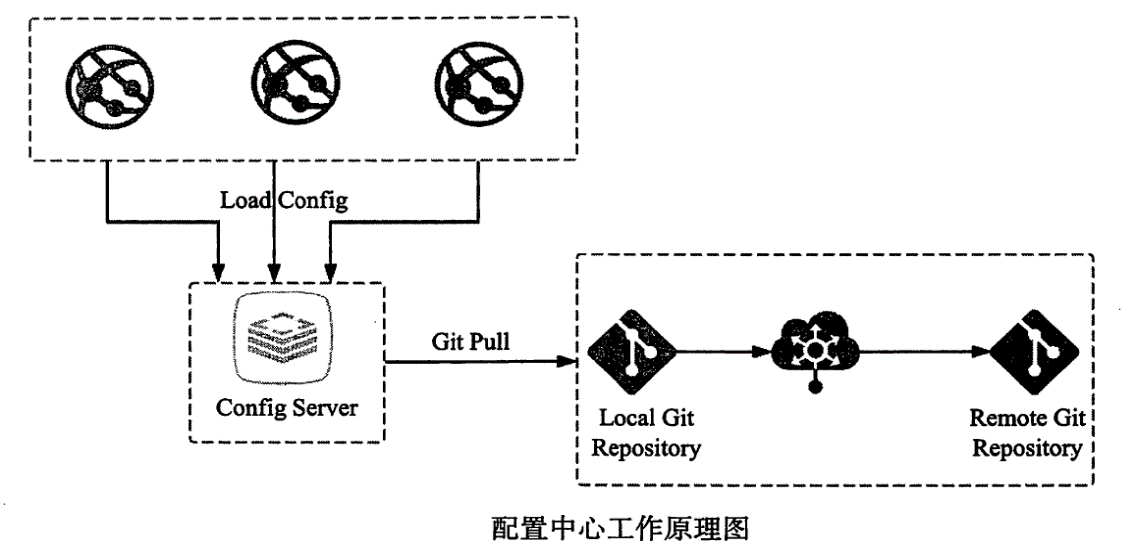


# Spring Cloud Config

## Spring Cloud Config概述

Spring Cloud Config是一个集中化外部配置的分布式系统，由服务端和客户端组成。它不依赖于注册中心，是一个独立的配置中心。Spring Cloud config支持多种存储配置信息的形式，目前主要有jdbc, Vault, Native, svn, git,其中默认为git。接下来开始学习Spring Cloud Config配置中心的构建和客户端的使用。

## Git版工作原理

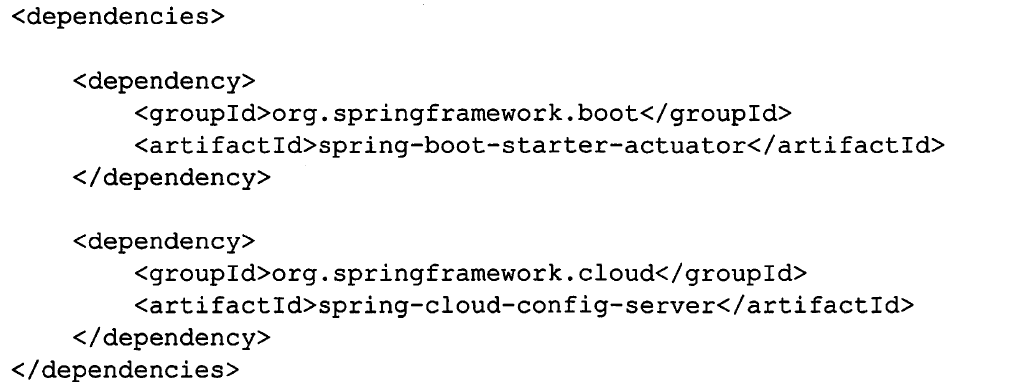


配置客户端启动时会向服务端发起请求，服务端接收到客户端的请求后，根据配置的仓库地址，将git上的文件克隆到本地的一个临时目录中，这个目录是一个git的本地仓库目录，然后服务端再读取本地文件返回给客户端。这样做的好处是，当git服务器故障或者网络请求异常时，保证服务端仍然可以正常工作。

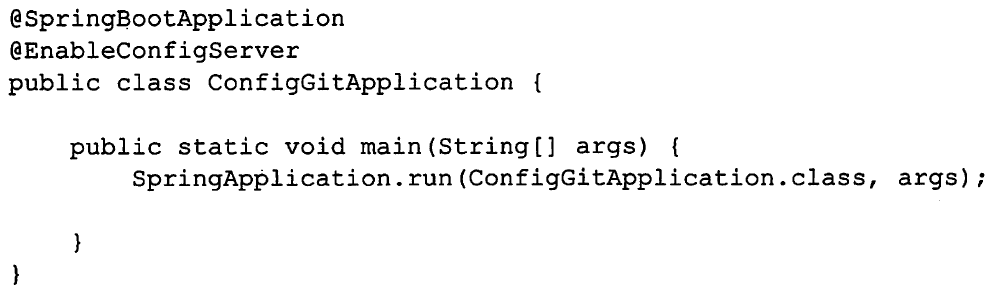
# 工程搭建

## Config Server配置

### Pom中添加包依赖



### 入口服务



在程序人口代码中的@EnableConfigServer注解是开启Spring Cloud Config的服务功能。

### 配置信息

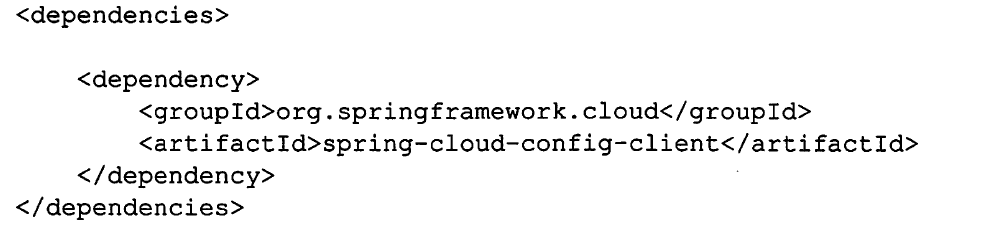


这里解释一下相关配置，配置中的uri指的是git服务器的地址;username和password指的是git访问的用户名和密码，如果设置访问需要用户名和密码，添加上对应的值即可;search-paths这个属性值就比较厉害了，可以搜索SC-BOOK-CONFIG目录下所有满足条件的配置文件，用户可以根据需求添加多个目录，目录之间用逗号隔开;config Server就创建完成了。

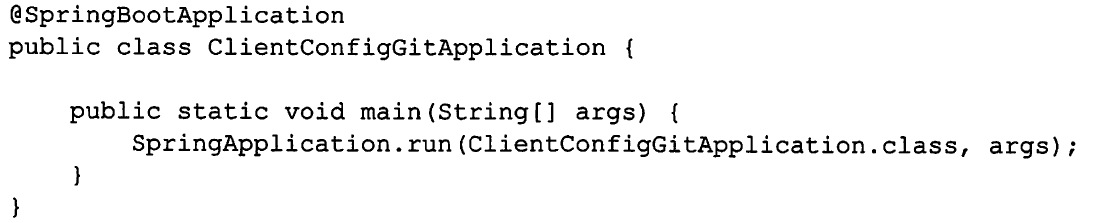
访问：http://localhost:9090/config-info/test/master

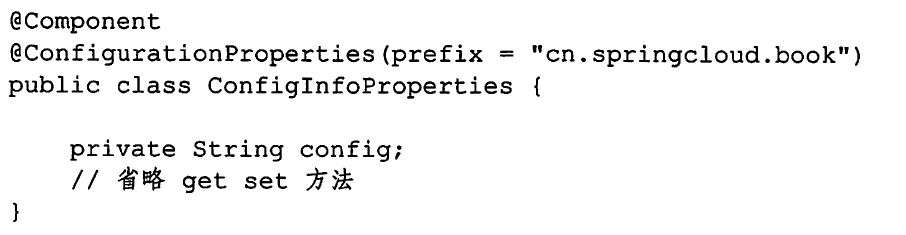
## Config Client配置

### Pom中添加包依赖



### 入口服务和加载配置类





创建实体类ConfigInfoProperties，并且添加@Component作为一个组件，同时使用@ConfigurationProperties加载配置属性和指定配置前缀。

### 配置信息



下面解释一下上述bootstrap.yml中的配置项。

label代表要请求哪个git分支，本案例中访问的是主分支。

uri代表请求的config server的地址。

name代表要请求哪个名称的远程文件，可以写多个，通过逗号隔开。

Profile 代表哪个分支的文件，例如dev, test, prod等。

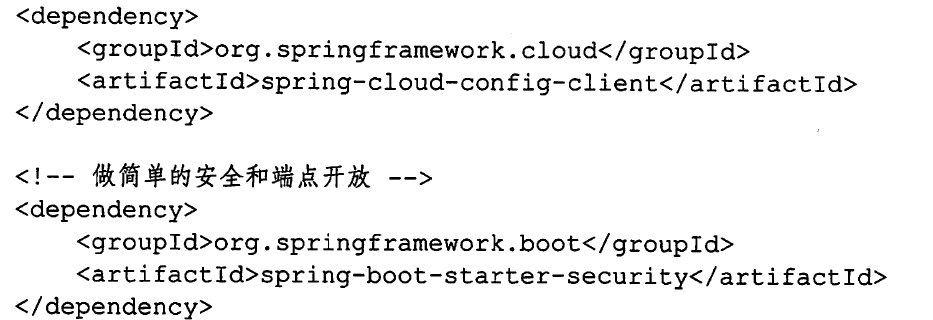
疑问：这些配置为什么要放在bootrap.yml里，而不放在application.yml中呢?

这与spring boot的加载顺序有关，bootstrap.yml文件会优先于application.yml加载，因此会去加载远程的配置文件信息，到此客户端中配置完成。

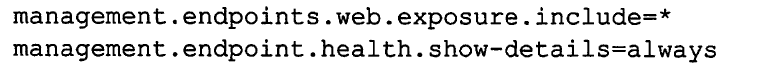
# 刷新配置中心信息

## Config 手动刷新配置

### 配置依赖

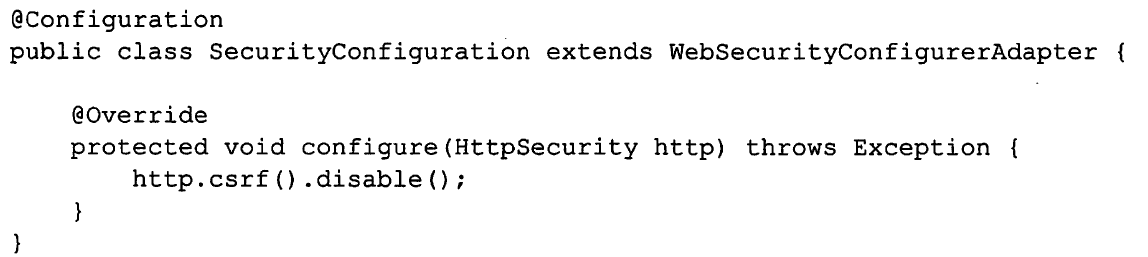


### 配置文件配置



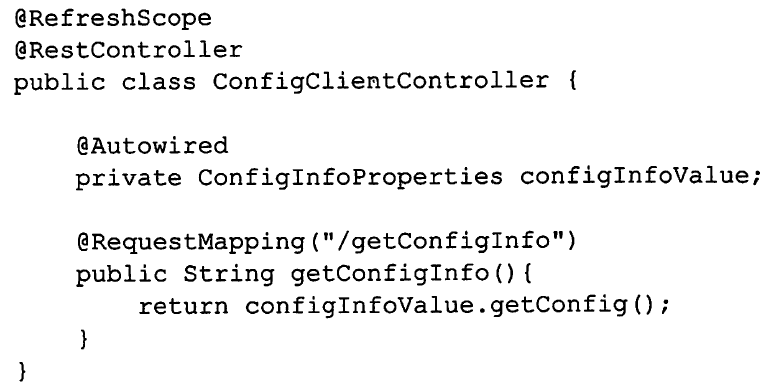
management.endpoints.web.exposure.include=\*来表示包含所有端点的信息，默认情况下，只是打开了info, health的端点；另外的配置management.endpoint.health.show-details=always总是表示详细信息的显示。

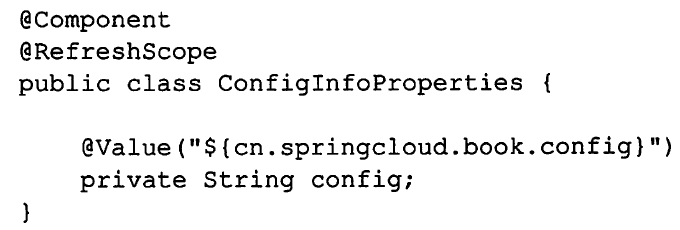
### 添加安全配置



configure方法中的代码主要是用来关闭端点的安全校验。

### Controller类的变更





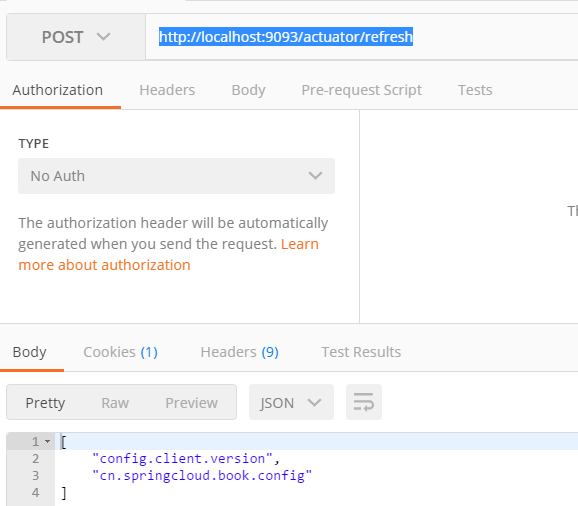
在类的头部添加了@RefreshScope注解，被@RefreshScope修饰的Bean都是延迟加载的，只有在第一次访问时才会被初始化，刷新Bean也是同理，下次访问时会创建一个新的对象。

### 手动刷新

修改一下远程仓库中的config-info-dev.yml配置信息

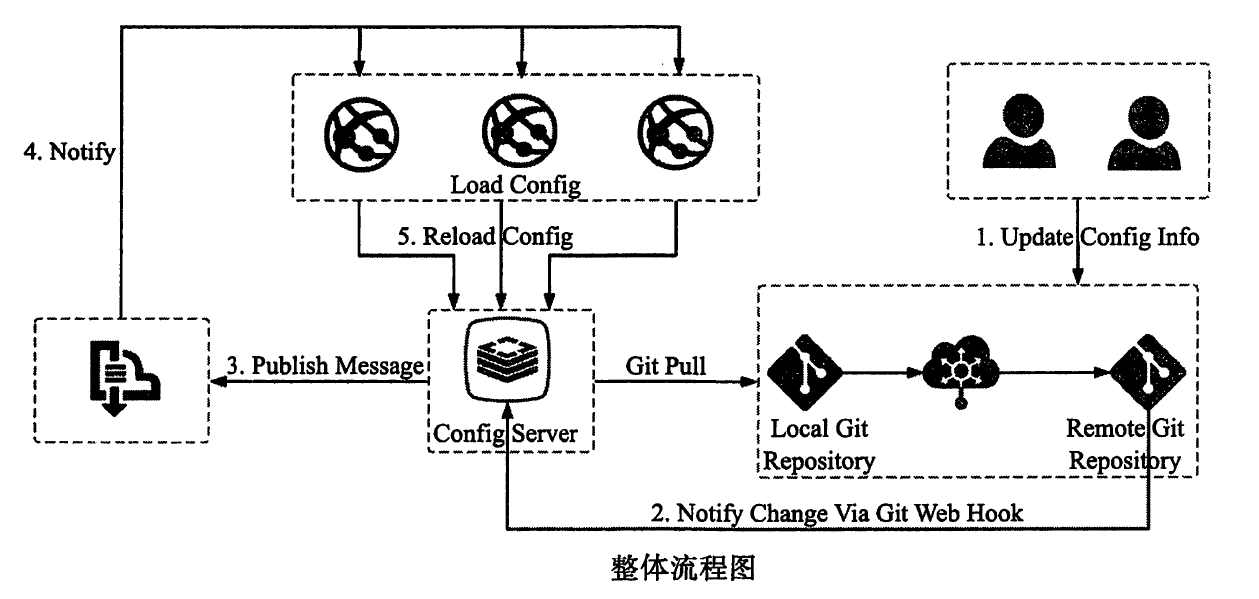
测试：<http://localhost:9093/configConsumer/getConfigInfo>

刷新：<http://localhost:9093/actuator/refresh>（一定是post请求）



## 结合Spring Cloud Bus热刷新

参考网址：https://www.jianshu.com/p/d6c56beaa743



通过客户端访问配置<http://localhost:9095/configConsumer/getConfigInfo>

方案一：

文件路径：C:\Users\Administrator\Downloads\ngrok-stable-windows-amd64

使用ngrok.exe 命令



网址：http://208a9381.ngrok.io/refresh

内网映射——本地服务127.0.0.1:9090

方案二：

使用花生壳软件，映射。



网址：<http://250l87291q.qicp.vip:22313/refresh>