**【Spring Cloud Config】**

**【主要内容】**

1. Config简介
2. Config-Server搭建
3. Config-Client搭建
4. 配置文件动态刷新集群

**【学习目标】**

|  |  |
| --- | --- |
| 知识点 | 要求 |
| Config简介 | 了解 |
| Config-Server搭建 | 掌握 |
| Config-Client搭建 | 掌握 |
| 配置文件动态刷新集群 | 掌握 |

# 为什么需要配置文件中心

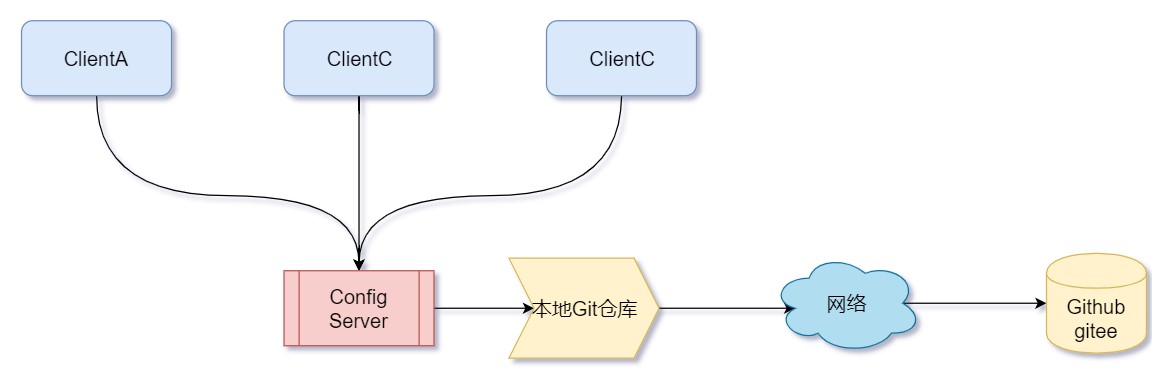
在分布式环境里面，很多服务都要做集群部署，那意味着这些服务都要提供一样的服务，所有它们的配置文件必须相同，我们提取一个配置文件中心，让它能够从一个中心处拉取配置文件，这样就能保证所有的配置都相同。我们在修改时，只需要修改一个地方就可以了。

## 回顾solr集群时的配置文件中心

# Spring Cloud Config简介

官网：<https://spring.io/projects/spring-cloud-config>

Spring Cloud Config 为微服务架构中的微服务提供集中化的外部配置支持，配置服务器为各个不同的微服务应用所有环境提供了一个中心化的外部配置，Config分为**服务端和客户端**两部分



**服务端：**

也称为分布式配置中心，它是一个**独立的微服务应用**，用来连接配置服务器并为客户端提供获取配置住处、加密、解密住处等访问接口

**客户端：**

则是通过指定的配置中心来管理应用资源，以及与业务相关的配置内容，并在启动的时候从配置中心获取加载配置信息

**配置服务器：**

默认使用**git**来存储配置住处，这样有助于对环境配置进行版本管理，并可以通过git客户端工具来方便的管理和访问配置内容

# Spring Cloud Config-server快速入门

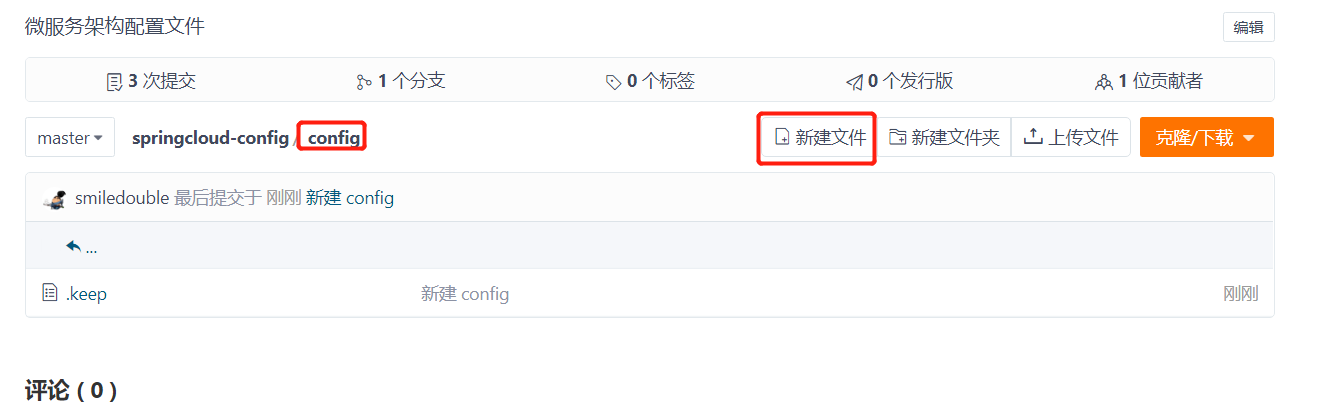
## 创建远端配置文件仓库（gitee上）



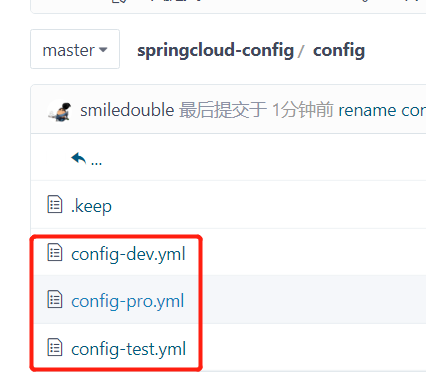
## 在远端仓库创建三个配置文件



**-------------------------------------------------**



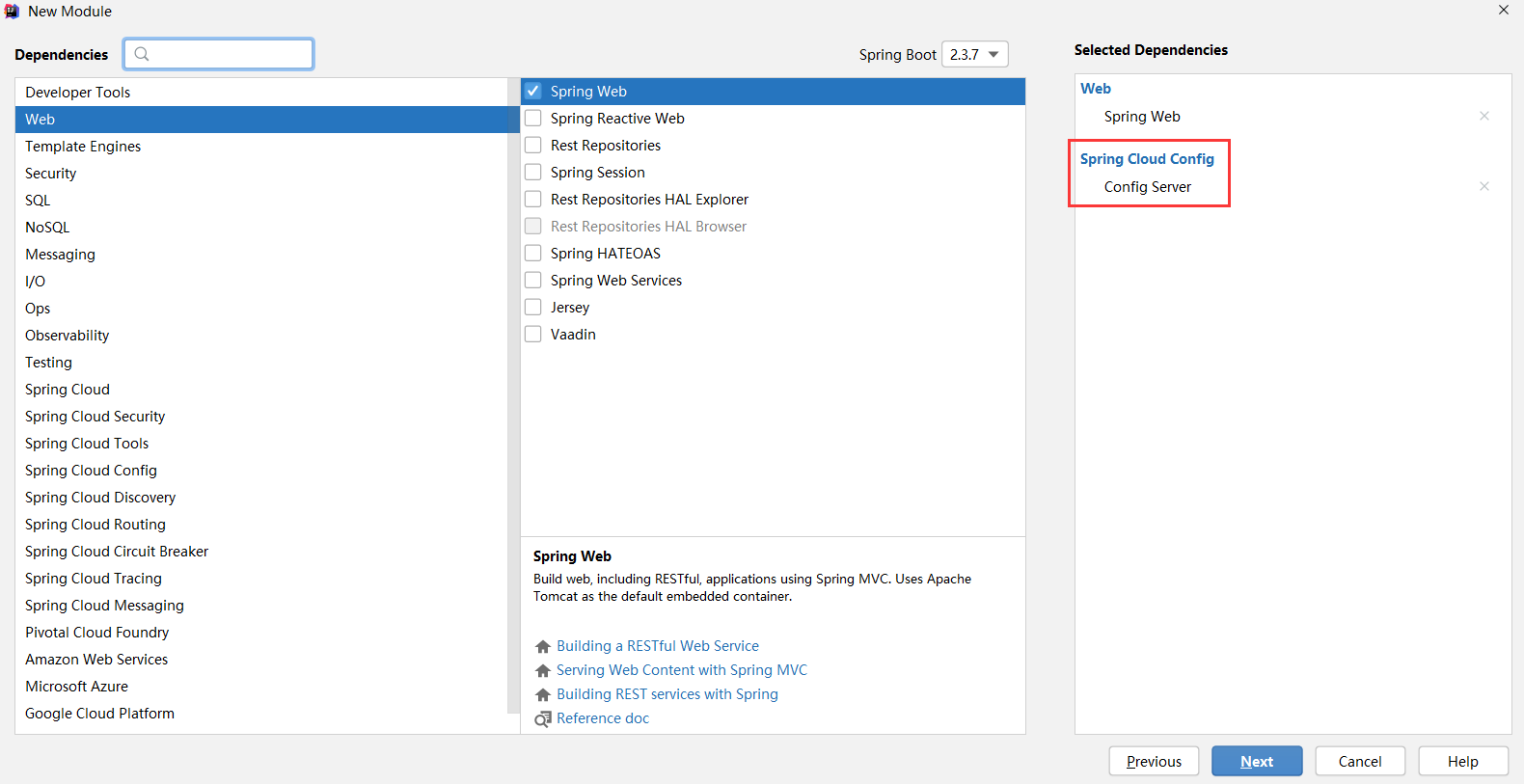
**-------------------------------------------------**



**-------------------------------------------------**

|  |
| --- |
| -------config-dev.yml--------  hero:  name: 亚索  age: 18  desc: 哈萨克  -------config-pro.yml--------  hero:  name: 超人  age: 18  desc: 不会飞  -------config-test.yml--------  hero:  name: 德玛  age: 18  desc: 大保健 |

## 创建项目config-server，选择依赖



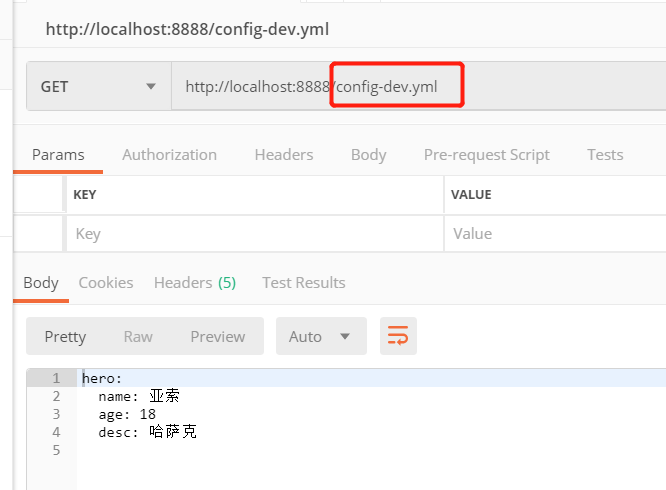
## 修改启动类

|  |
| --- |
| @SpringBootApplication @EnableConfigServer //开启配置文件中心 public class ConfigServerApplication {   public static void main(String[] args) {  SpringApplication.*run*(ConfigServerApplication.class, args);  } |

## 修改配置文件

|  |
| --- |
| server:  port: 8888 spring:  cloud:  config:  server:  git:  uri: https://gitee.com/smiledouble/springcloud-config  username: smiledouble  password: \*\*\*\*\*\*  search-paths: /config |

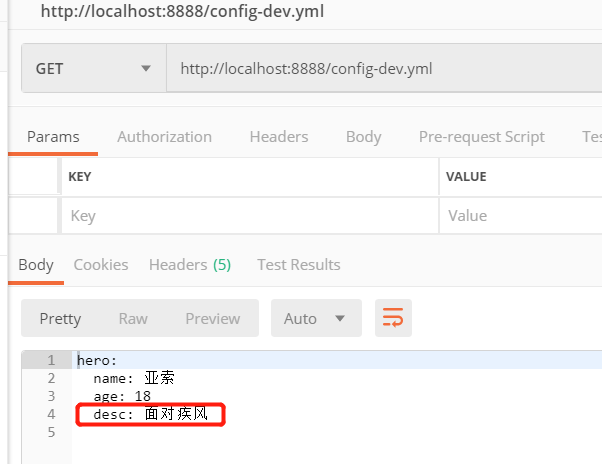
## 启动访问



## 修改gitee上的配置文件，继续访问测试



访问查看，也更新了



## 配置文件读取规则

官网说明

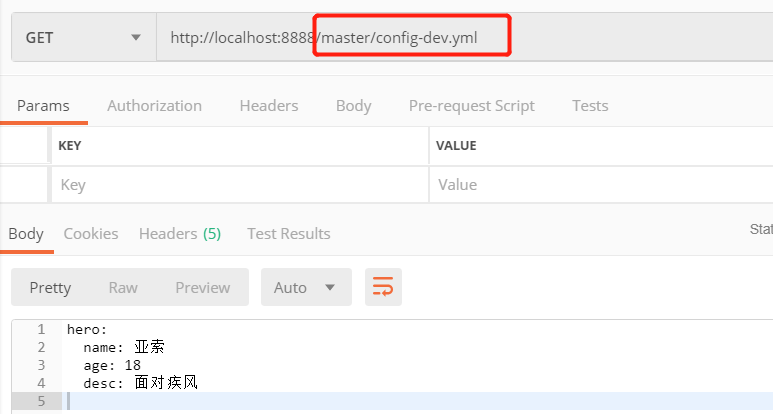
<https://docs.spring.io/spring-cloud-config/docs/2.2.5.RELEASE/reference/html/#_quick_start>

|  |
| --- |
| /{application}/{profile}[/{label}]  /{application}-{profile}.yml  **/{label}/{application}-{profile}.yml 推荐使用这一种**  /{application}-{profile}.properties  /{label}/{application}-{profile}.properties |

application 配置文件名称前缀 **config**-dev.yml

profile 配置文件名称的简介 config-**dev**.yml

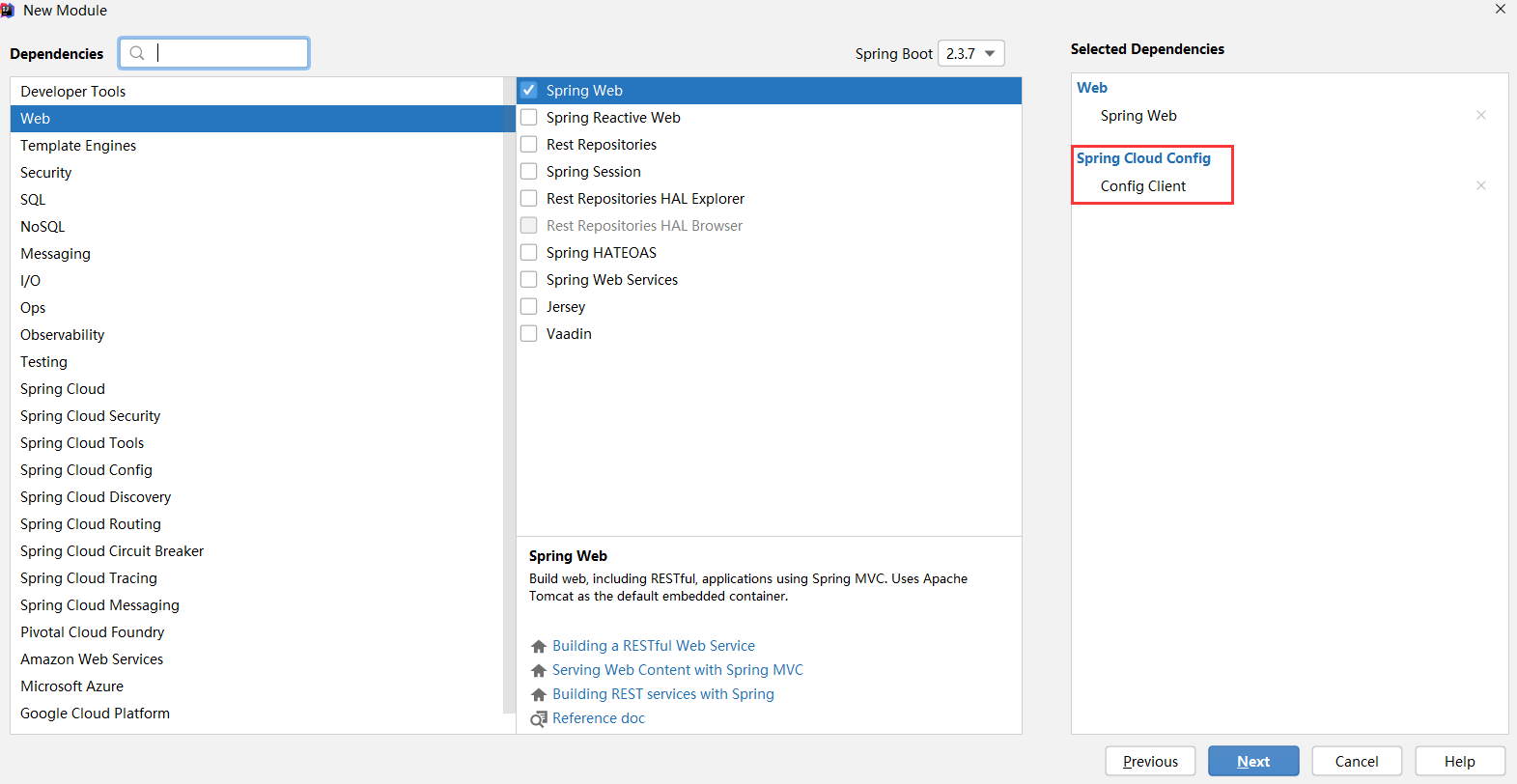
label 配置文件位于分支名称 例如master，dev 可以自己创建分支



# Spring Cloud Config-client快速入门

Config-client，配置文件客户端，和具体的微服务放在一起

## 创建项目，选择依赖



## 修改配置文件

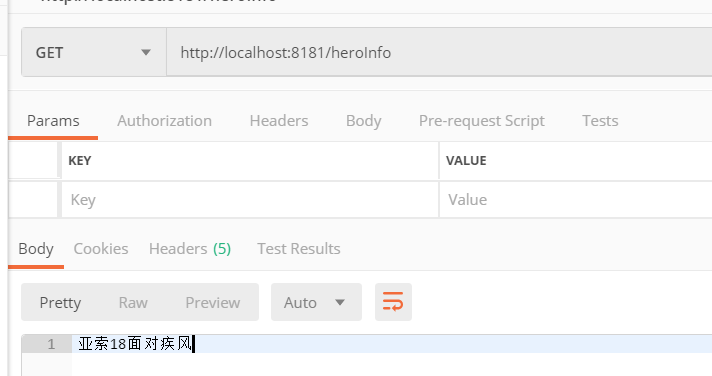
**这里将application.yml改成bootstrap.yml**

|  |
| --- |
| server:  port: 8181 spring:  application:  name: config-client  cloud:  config:  uri: http://localhost:8888/ #找到配置文件中心的地址，  profile: dev #选择哪一个配置文  name: config #对应application  label: master #对应分支 |

## 编写测试

|  |
| --- |
| @RestController public class TestController {    @Value("${hero.name}")  private String name;   @Value("${hero.age}")  private Integer age;   @Value("${hero.hobby}")  private String hobby;    /\*\*  \* 查看英雄信息  \*  \* @return  \*/  @GetMapping("heroInfo")  public String heroInfo() {  return name + ":" + age + ":" + hobby;  } } |

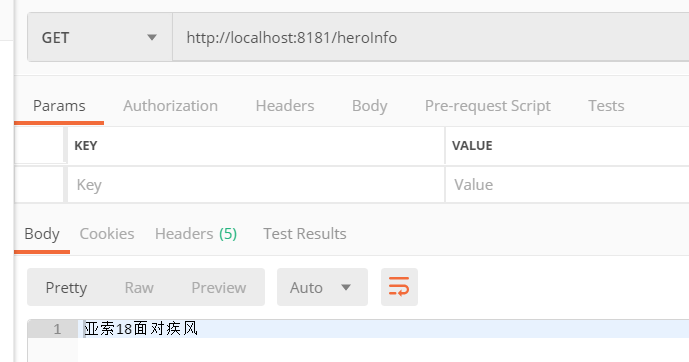
## 测试访问



## 修改gitee上配置文件内容

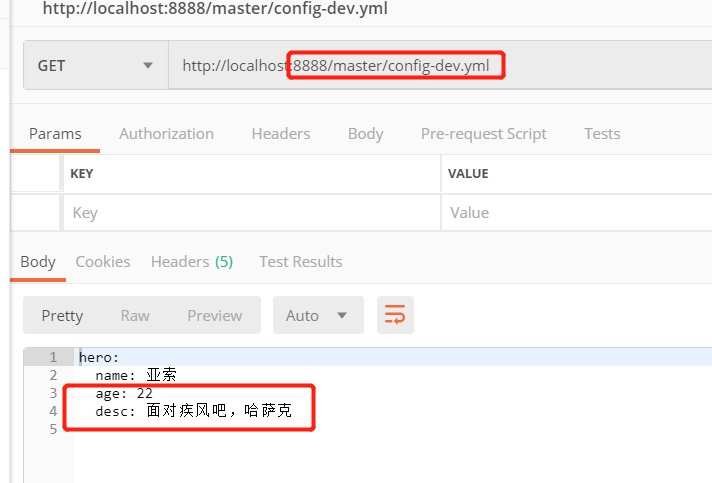


## 测试从config-client访问



**我们发现并没有同步更新**

## 测试从config-server访问



发现已经同步访问了，得出结论

**Git--->config-server --x-->config-client**

**那么问题来了，如何实现配置文件的动态刷新？？**

# Config-client配置文件动态刷新

## 重启Config-client（可以实现但没必要）

重启可以实现配置文件刷新，相当于重新拉取了一次，但是我们不想让微服务重启

## 改造Config-client

### 添加一个依赖

|  |
| --- |
| <!-- actuator 健康检查，暴露端点的一个包，配置文件的监控和刷新--> <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-actuator</artifactId> </dependency> |

### 修改config-client的配置文件bootstrap.yml

|  |
| --- |
| server:  port: 8181 spring:  application:  name: config-client  cloud:  config:  uri: http://localhost:8888/ #找到配置文件中心的地址，  profile: dev #选择哪一个配置文  name: config #对应application  label: master #对应分支 management:  endpoints:  web:  exposure:  include: '\*' #暴露监听的端口 |

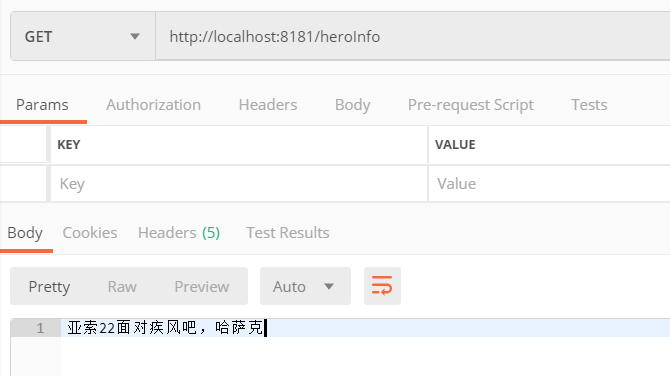
### 创建HeroProperties类

|  |
| --- |
| @Component @RefreshScope //刷新域 @Data public class HeroProperties {   @Value("${hero.name}")  private String name;   @Value("${hero.age}")  private Integer age;   @Value("${hero.desc}")  private String desc;  } |

### 修改controller

|  |
| --- |
| @Autowired private Hero hero;   /\*\*  \* 查看英雄信息  \*  \* @return  \*/ @GetMapping("heroInfo") public String heroInfo() {  return hero.getName() + ":" + hero.getAge() + ":" + hero.getHobby(); } |

### 测试访问



### 修改git上的配置内容

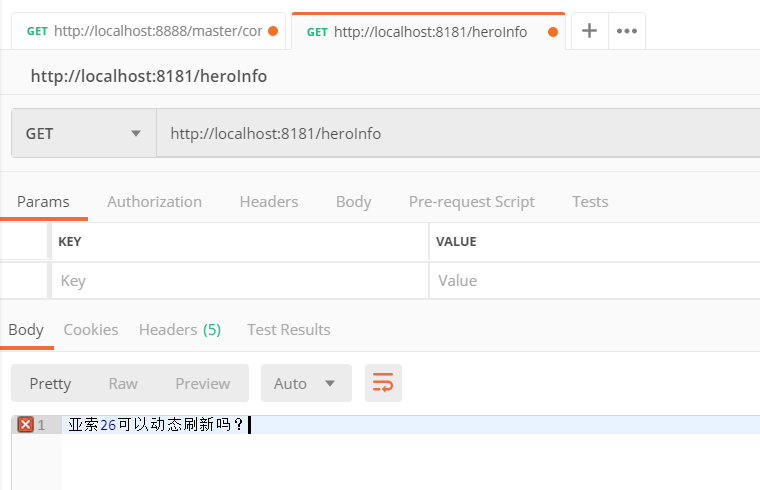


### 再次从config-client访问并没有同步生效

### 需要手动刷新

**<http://localhost:8181/actuator/refresh> 发一个POST请求**

再访问查看已经刷新了



# 配置**文件**总结

每次修改远端仓库git上的配置文件后，config-server可以立刻感知到，但是config-client感知不到，需要手动发送一个post请求刷新同步，问题来了，如果有很多微服务，每一个都需要手动刷新，那么有没有什么办法，让全部的client都刷新呢？

下一节我们引入消息总线Bus来解决这个问题