参考文档：<https://blog.csdn.net/liubenlong007/article/details/87098662>

<https://www.cnblogs.com/fengzheng/p/11242128.html>

config官网：

<https://cloud.spring.io/spring-cloud-static/spring-cloud-config/2.2.3.RELEASE/reference/html/#_quick_start>

1、最简单的config-server，client模式，当然git少不了。不能刷新，每次配置更新后，需要重启客户端，才能同步到新数据。

2、加入actuator后，更新数据后，不需要重启客户端，需要执行客户端的/actuator/refresh来更新数据。（这个只能刷新一个客户端实例的数据）

3、加入git的webhook功能，每次push数据了，通过webhook来调用/actuator/refresh，实现自动刷新（不需要手动）

4、加入spring-cloud-bus和mq来在客户端之间广播数据。通过/actuator/bus-refresh来更新数据。所有的客户端的数据都会更新。

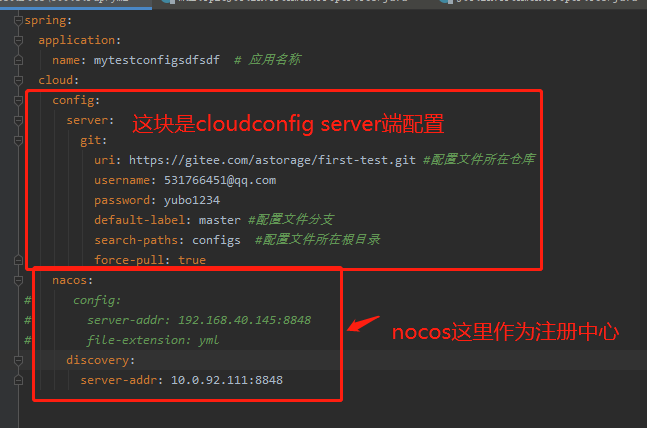
5、最后加入注册中心。服务端实现高可用。服务端可以启动多个实例来保证config-server服务的高可用。config-client不在指定server的uri，而是通过注册中心和server-id来联系config-server。

spring-cloud-config-server

引入依赖

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-config-server</artifactId>  
</dependency>

配置



启动了config-server程序

config-server拉了远程仓库的配置到，本地来：

C:/Users/10019/AppData/Local/Temp/config-repo-6442432611997010376

我在windows请求配置数据，报错了。

Failed to load property source from

'file:/C:/Users/10019/AppData/Local/Temp/config-repo-6442432611997010376/configs/mytestconfig-dev.yml'

这个原因是，因为配置文件不符合yml语法规范。所以报错了。

改了配置文件，提交到git仓库。

再请求config-server的api，获取的配置文件数据，已经好了。

我当时使用的api地址

http://localhost:3301/mytestconfig/test

配置config-client

spring:  
 application:  
 name: mytestconfig  
 cloud:  
 config:  
 uri: http://10.0.92.85:3301  
 *#profile: dev*

mytestconfig：这个应用名有讲究，必须和配置中心的配置文件名字对应起来。

/{application}/{profile}[/{label}]

/{application}-{profile}.yml

/{label}/{application}-{profile}.yml

/{application}-{profile}.properties

/{label}/{application}-{profile}.properties

1、label是git的分支。

2、application是应用名，也是配置文件的前缀

3、profile是程序环境（dev，test，pro等等），也是配置文件名的后半部分。

如果不配置profile，使用default，没拉下来配置文件。现在配置文件，只有dev和test两个版本。

2020-07-29 11:06:18.181 INFO 13808 --- [ main] c.c.c.ConfigServicePropertySourceLocator : Fetching config from server at : http://10.0.92.85:3301

2020-07-29 11:06:24.005 INFO 13808 --- [ main] c.c.c.ConfigServicePropertySourceLocator : Located environment: name=mytestconfig, profiles=[default], label=null, version=fa38f54fbc185b79bc16db3172ee23be19f680e2, state=null

2020-07-29 11:06:24.005 INFO 13808 --- [ main] b.c.PropertySourceBootstrapConfiguration : Located property source: [BootstrapPropertySource {name='bootstrapProperties-configClient'}]

2020-07-29 11:06:24.009 INFO 13808 --- [ main] c.j.c.ConfigclientApplication : No active profile set, falling back to default profiles: default

配置了profile为dev。拉下来了dev文件

2020-07-29 11:10:34.733 INFO 14872 --- [ main] c.c.c.ConfigServicePropertySourceLocator : Fetching config from server at : http://10.0.92.85:3301

2020-07-29 11:10:41.395 INFO 14872 --- [ main] c.c.c.ConfigServicePropertySourceLocator : Located environment: name=mytestconfig, profiles=[dev], label=null, version=fa38f54fbc185b79bc16db3172ee23be19f680e2, state=null

2020-07-29 11:10:41.396 INFO 14872 --- [ main] b.c.PropertySourceBootstrapConfiguration : Located property source: [BootstrapPropertySource {name='bootstrapProperties-configClient'}, BootstrapPropertySource {name='bootstrapProperties-https://gitee.com/astorage/first-test.git/configs/mytestconfig-dev.yml'}]

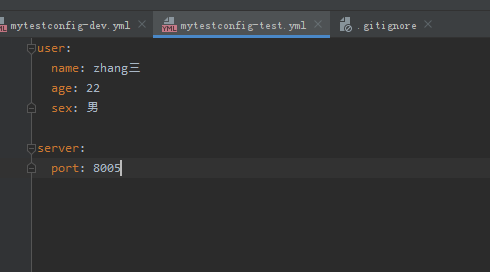
2020-07-29 11:10:41.429 INFO 14872 --- [ main] c.j.c.ConfigclientApplication : The following profiles are active: dev

问题：配置文件拉下来后，客户端程序启动成功后，却没有暴露端口。

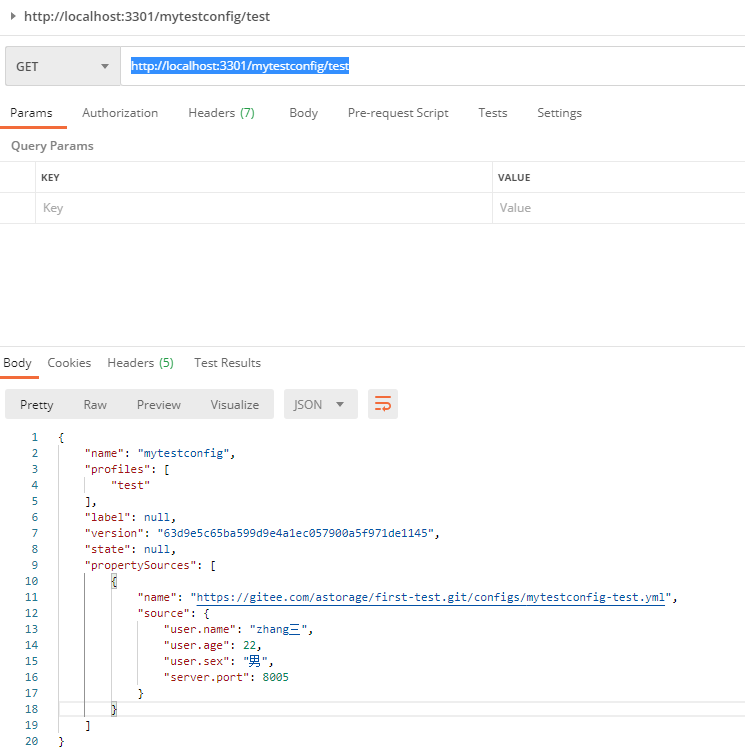
原因：拉下来的配置文件，影响了程序。配置中心的配置文件中配置了端口信息，它覆盖了本地配置文件的端口号。

server:  
 port: 8005

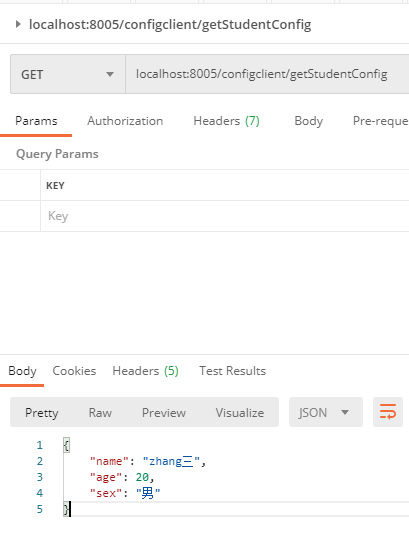
配置文件修改了，配置中心（config-server）可以实时感知到配置文件的变化。



请求服务端：数据已经变化了



请求客户端：数据仍未变化

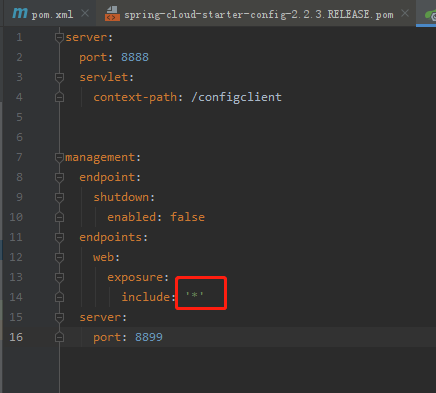


引入actuator

引入依赖：

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>

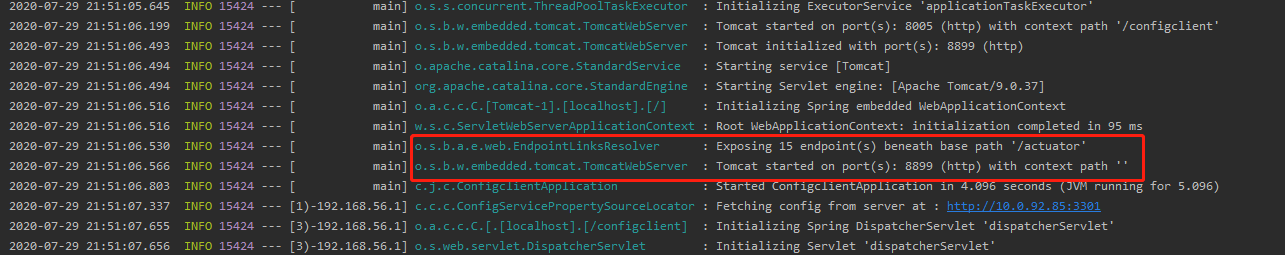
配置如下：

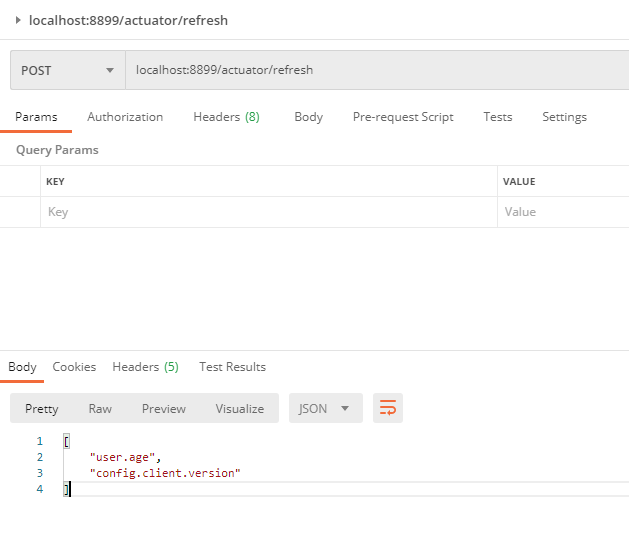


\*必须有引号（单引号双引号都可以），不然会报错

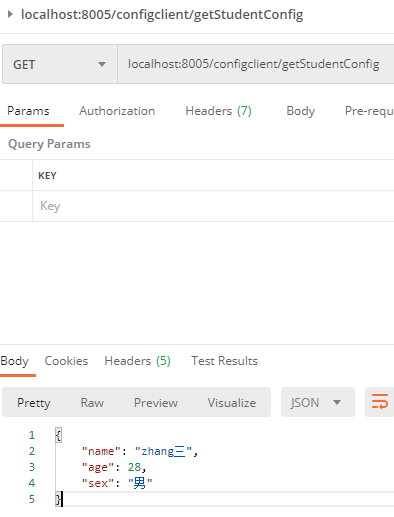
配置了actuator后，

启动程序，有如下日志，说明actuator配置好了

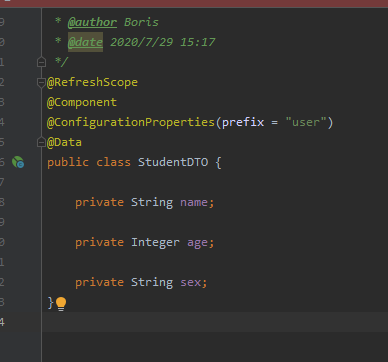
请求localhost:8899/actuator/refresh 会返回如下信息，说明配置文件中user.age配置变化了



在请求数据，在config-client获取到的数据跟配置变化一致

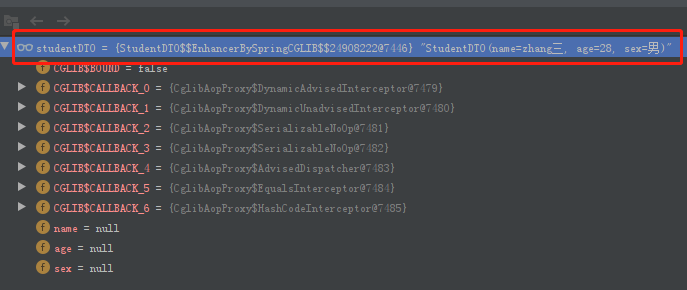


如果需要自动刷新，可以用@RefreshScope注解



这样修改了配置中心的配置，可以直接获取到修改后的数据，不用再调用/actuator/refresh

不过用这个注解有点问题：就是student实参springcglib代理对象。注入的对象会多出很多信息。

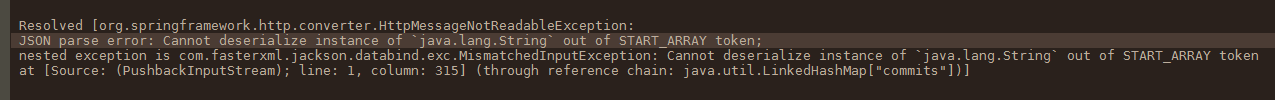


如果将@RefreshScope放在Controller上，却不能实现自动更新。需要调用/actuator/refresh接口。

奇怪的地方：我后来加了mapstruct，就不能自动更新了。必须要调用/actuator/refresh刷新接口。代码恢复到以前还是不行。怎么都不能自动更新了，我好气！这么诡异！咋回事？吐血。。

@ConfigurationProperties搭配使用能能手动刷新更新。@Value不行。

问题：在gitee，github, gitlab中配置webhook请求/actuator/refresh会报错。可能原因是因为请求的参数不一致



自己写一个接口来转接刷新接口。解决这个问题



/actuator/refresh只能刷新一个实例，如果客户端有多个的情况（少量还无所谓，多个的话就有点尴尬）

Spring cloud bus就发挥作用了。使用bus的前提条件是要有mq。我们这里用rabbitmq

bus在客户端之间传递消息

在spring-cloud-config-client端变化

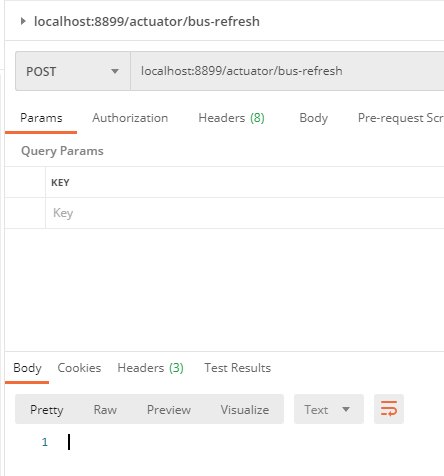
引入依赖

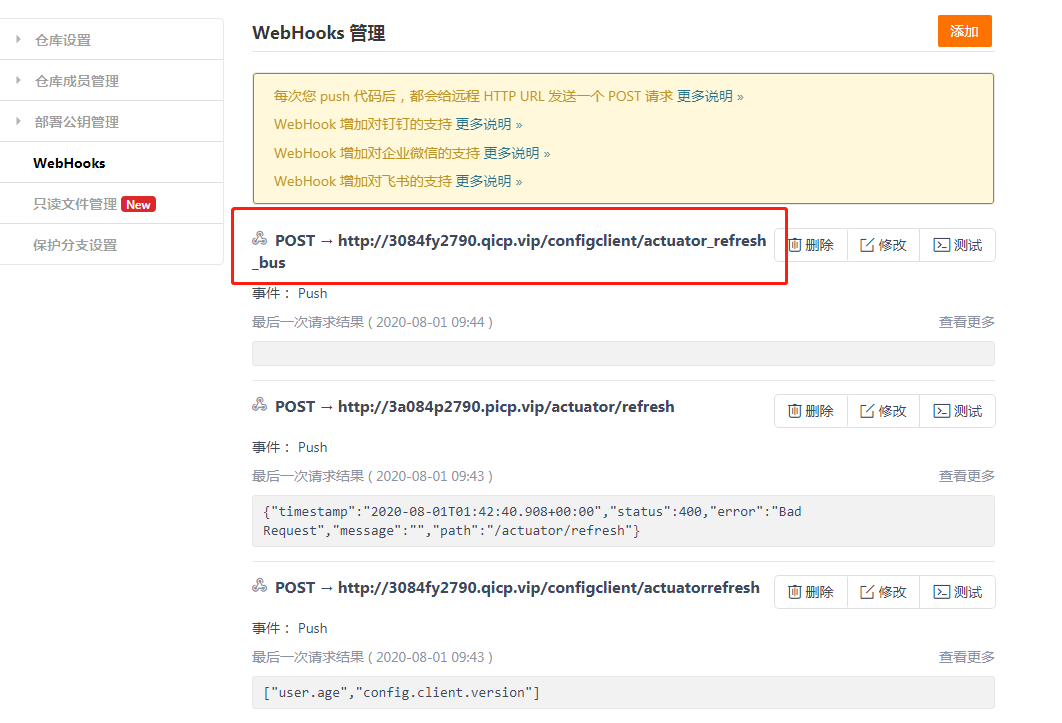
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-bus-amqp</artifactId>  
</dependency>

添加链接rabbitmq的配置

spring:  
 rabbitmq:  
 host: 10.0.92.102  
 port: 5672  
 username: guest  
 password: guest

这里刷新的接口也不再是/actuator/refresh。bus刷新用/actuator/bus-refresh





分布式程序都是用注册中心的。

nacos

nacos安装配置

安装java， 配置环境变量

nacos控制台地址<http://10.0.92.111:8848/nacos/index.html>

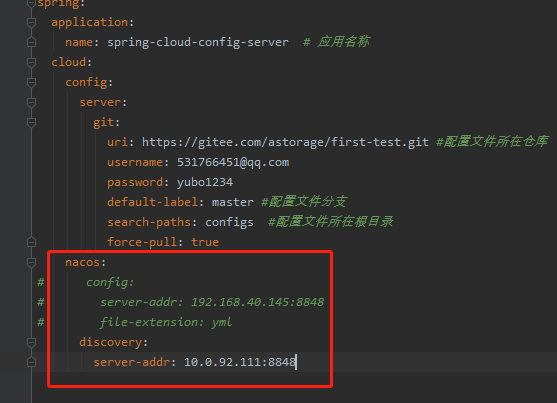
默认用户名密码 nacos/nacos

config-server使用注册中心

添加依赖

<dependency>  
 <groupId>com.alibaba.cloud</groupId>  
 <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>  
 <version>2.2.1.RELEASE</version>  
</dependency>

配置注册中心

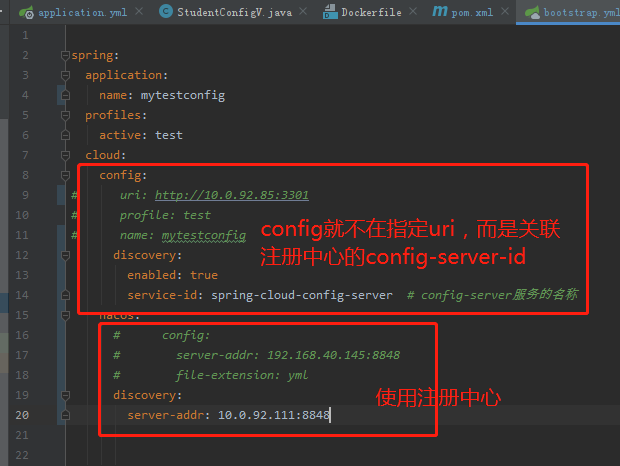


config-client使用注册中心（在前面的基础上增加配置中心）

添加依赖：

<dependency>  
 <groupId>com.alibaba.cloud</groupId>  
 <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>  
 <version>2.2.1.RELEASE</version>  
</dependency>

添加配置



到这里。spring-cloud-config使用就结束了。

总结一下：

用到的组件：

1、spring-cloud-config 主角，config-server将git仓库中的数据实时拉去下来到本地。config-client从config-server拉去数据

2、spring-boot-starter-actuator 用于不停机刷新数据

3、spring-cloud-bus 将消息在多个客户端之间传递

4、rabbitMQ cloud-bus需要借助mq的发布订阅模式来广播消息

5、注册中心（nacos）

6、git 用来管理配置文件，如果配置文件又修改，通过webhook功能实时告知客户端配置数据有变化

不好的地方：

因为要考git的webhook来刷新某一个客户端的数据。问题就来了

1、这个客户端必须保证可用，不然就不能刷新了

2、如果实例实例换掉了，必须要提供相同刷新数据的url来保证客户端能接收刷新信息。也可以在git上多配置几个webhook地址。