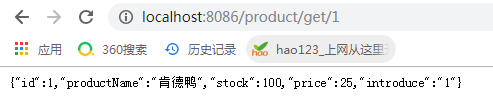
# Resttemplat

Order调用product



@GetMapping("/get/{id}")  
@ResponseBody  
public Product byId(@PathVariable(value = "id") Integer id) {  
 Product obj = productService.findAllById(id);  
 if (obj != null) {  
 return obj;  
 }  
 return null;  
}

上面是product提供的接口服务

下面为order调用这个接口

方式1 :

http://localhost:8080/order/create?productId=1&num=1

RestTemplate restTemplate = new RestTemplate();  
String productStr = restTemplate.getForObject("http://localhost:8086/product/get/"+productId,String.class);  
 *log*.info(productStr+"-----------------------");

{"id":1,"productName":"肯德鸭","stock":100,"price":25,"introduce":"1"}-----------------------

缺点 url写死了，服务器多台很麻烦，不知道ip

方式2:

@Autowired  
private LoadBalancerClient loadBalancerClient;

RestTemplate restTemplate = new RestTemplate();  
ServiceInstance serviceInstance = loadBalancerClient.choose("PRODUCT");  
String url = String.*format*("http://%s:%s",serviceInstance.getHost(),serviceInstance.getPort())+"/product/get/"+productId;  
String productStr = restTemplate.getForObject(url,String.class);  
//String productStr = restTemplate.getForObject("http://localhost:8086/product/get/"+productId,String.class);  
*log*.info(productStr+"-----------------------");

PRODUCT为eraeka服务名

方式2内部已经采取软负载均衡

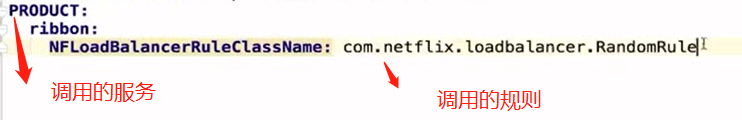
http://localhost:8080/order/create?productId=1&num=1



# Ribbon

服务发现 服务选择规则 服务监听

serviceList IRule serviceListFilter



### Feign

dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-feign</artifactId>  
 <version>1.4.4.RELEASE</version>  
</dependency>

启动类添加 @EnableFeignClients

@SpringBootApplication  
@EnableDiscoveryClient  
@EnableFeignClients  
public class OrderApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrderApplication.class, args);  
 }  
}

声明试rest客户端 为RPC 采用了基于接口的注解

Order调用product

1

@RequestMapping("/productId/{id}")  
@ResponseBody  
public String test(@PathVariable int id) {  
 *log*.info("-------byId-----");  
 return productClient.byId(id);  
}

2

@FeignClient(name = "product")  
public interface ProductClient {  
  
 @PostMapping("/getId")  
 String byId(@RequestParam("id") int id);  
}

3product

@PostMapping("/getId")  
@ResponseBody  
public Product byId(@RequestParam("id")int id) {  
 System.*out*.println("------------product----------");  
 Product obj = productService.findAllById(id);  
 if (obj != null) {  
 return obj;  
 }  
 return null;  
}

<https://www.jianshu.com/p/7ce46c0ebe9d>

<https://www.cnblogs.com/rulian/p/6434631.html>

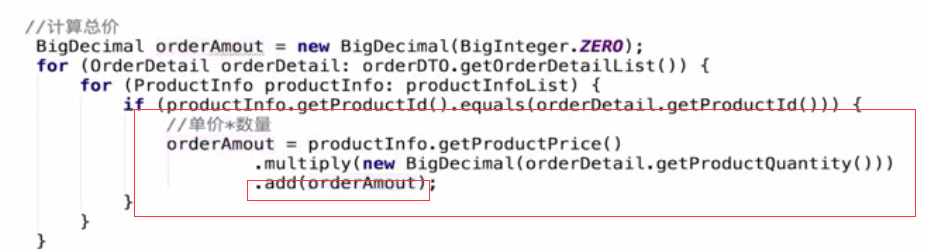
 1. 当参数比较复杂时，feign即使声明为get请求也会强行使用post请求

 2. 不支持@GetMapping类似注解声明请求，需使用@RequestMapping(value = "url",method = RequestMethod.GET)

 3. 使用@RequestParam注解时必须要在后面加上参数名

如果传参比较复杂如集合product

@PostMapping("/queryProducts")  
@ResponseBody  
public List<Product> findQueryByIds(@RequestBody List<Integer> productIds){  
 System.*out*.println("------------product----------");  
 return productService.findbyProductIdIn(productIds);  
}





值为null也会拷贝

# 统一配置中心异步机制

新建项目

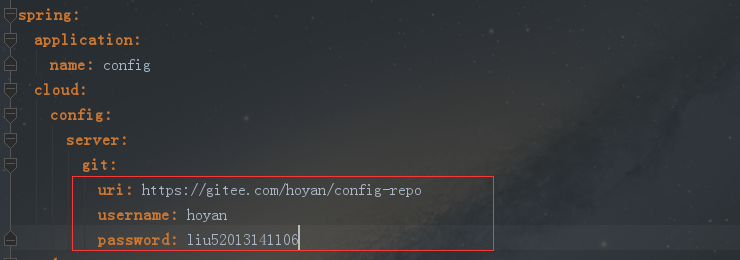
添加eureka和config依赖

启动类添加注解

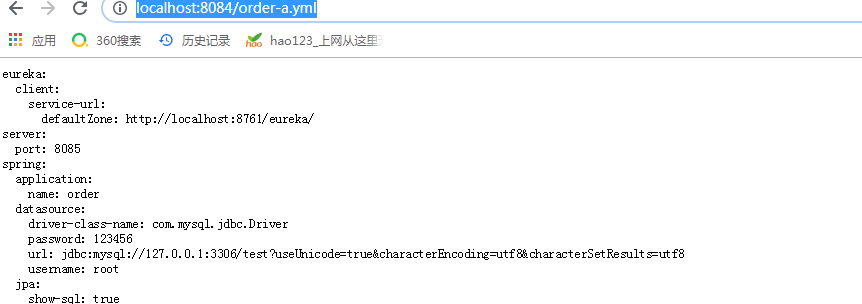
@EnableConfigServer

错误Git仓库 ： You need to configure a uri for the git repository

<http://localhost:8084/order-a.yml>



<http://localhost:8084/order-a.yml> 或者 <http://localhost:8084/order-a.properties> 或者http://localhost:8084/order-a.json（会自动转换格式）如果格式错误将不会展示



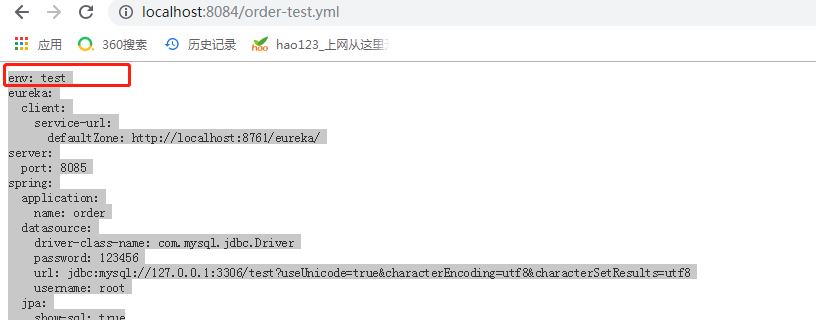
为什么要order-a.yml呢？

解释==== /{name}-{profiles}.yml 还有就是 /{label}/{name}/{profiles}.yml

Label分支 name 服务名 profiles 环境



<http://localhost:8084/order-test.yml>

这个演示没有分支，如果分支演示 添加分支即可

Order项目引入删除多余配置 启动报错加载数据库失败

原因 是加载顺序的问题

bootstrap.yml 这个文件可以使文件加载最前面

<http://localhost:8085/env/dev>获取git配置上的信息

配置中心也需要高可用

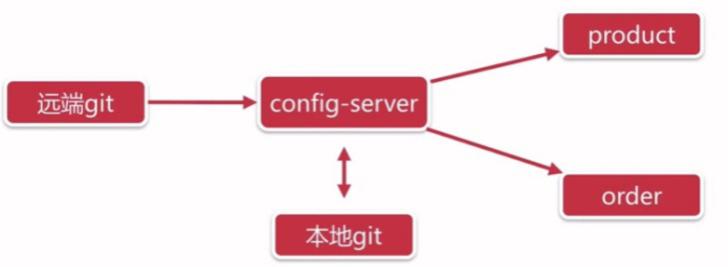
如果修改码云上的配置，访问并没有修改的时候

问题:

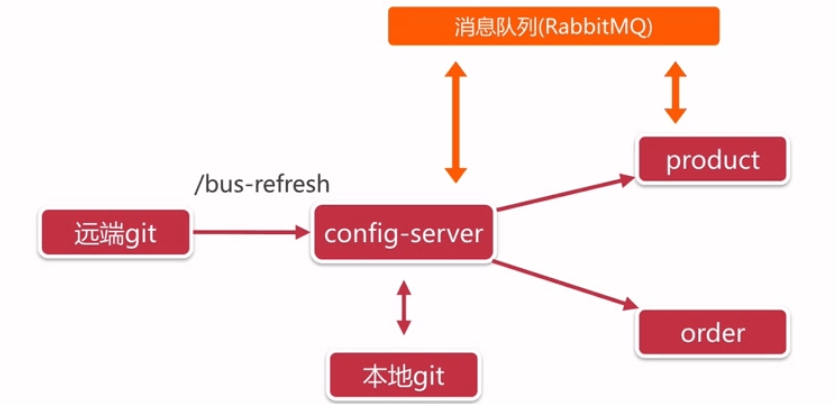
查看日志问题是访问这两个文件进行合并返回的 所以两个要改成一致

更改git配置 服务order无法制动刷新

解决 springCloud Bus 总线



需要使用rabbitmq通知更新



ORDER项目和config项目都要修改和添加amqp

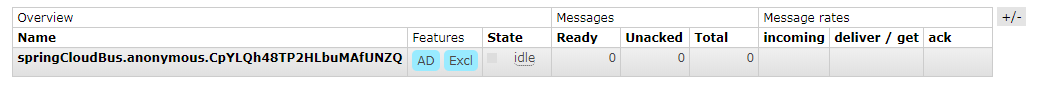
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-bus-amqp</artifactId>  
</dependency>

Spring boot版本修改 <version>1.5.3.RELEASE</version>

SpringCloud版本修改<spring-cloud.version>Dalston.SR1</spring-cloud.version>

还有注册中心

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

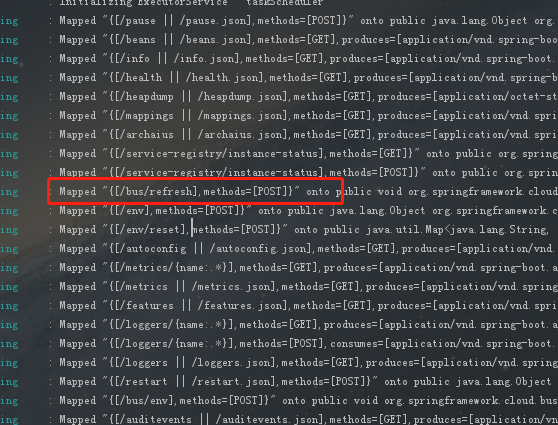


Rabbitmq会自动添加一个队列

修改版本之后feign组件有些内容也已经发生改变需要修改

Config项目需要添加一下的配置文件暴露一下接口

**management:  
 security:  
 enabled:** false  
 **endpoints:  
 web:  
 expose:** "\*"





@RefreshScope()

ORDER的一些配置文件已经放在码云，

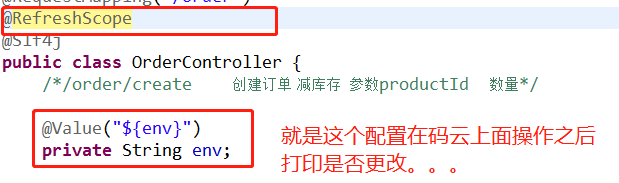
然后通过config项目和码云建立关系，项目中的演示为

更改码云的配置文件，不需要重启项目就可以自动加载

步骤1：修改码云的配置，

步骤2：发送<http://localhost:8084/bus/refresh>请求post方式（Spring cloud版本如果不一致，请求方式和地址可能不一样）具体看config日志打印

步骤3：稍待片刻 请求接口查看是否更改内容



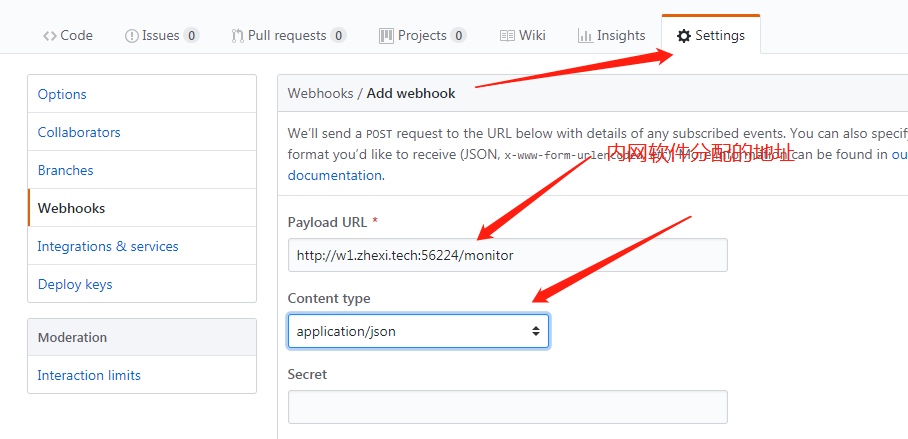
问题:总不能每次手动请求接口来请求更新文件吧

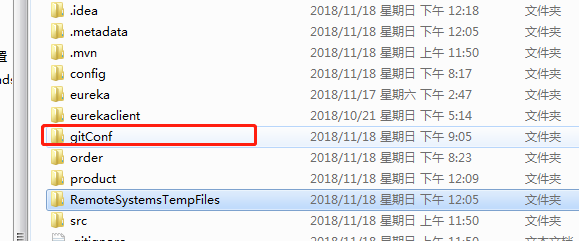
码云的配置现在改用github

**uri:** https://gitee.com/hoyan/config-repo  
**username:** hoyan  
**password:** liu52013141106  
**basedir:** F:/springcloud/gitConf

//此过程没有实现

<https://xd.zhexi.tech/remote/mapping> 来个内网ip





修改git order-test.yml文件config项目将会有新日志打出

config完结