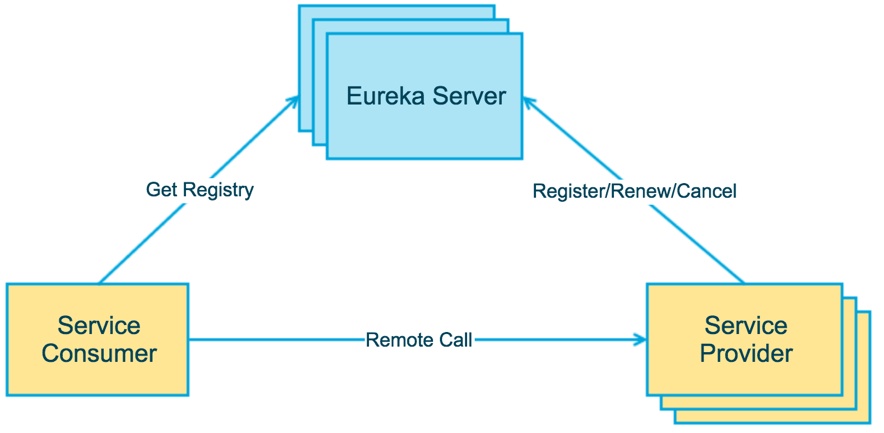
## 3001注册中心

Eureka注册中心是springcloud的核心，管理各种服务功能包括服务的注册、发现、熔断、负载、降级等。

Eureka由两个组件组成：Eureka服务器和Eureka客户端。Eureka服务器用作服务注册服务器。Eureka客户端是一个java客户端（我们使用springcloud框架的微服务都是一个eureka客户端），用来简化与服务器的交互、作为轮询负载均衡器，并提供服务的故障切换支持。、

用一张图来认识一下：



上图简要描述了Eureka的基本架构，由3个角色组成：

### 1、Eureka Server

提供服务注册和发现

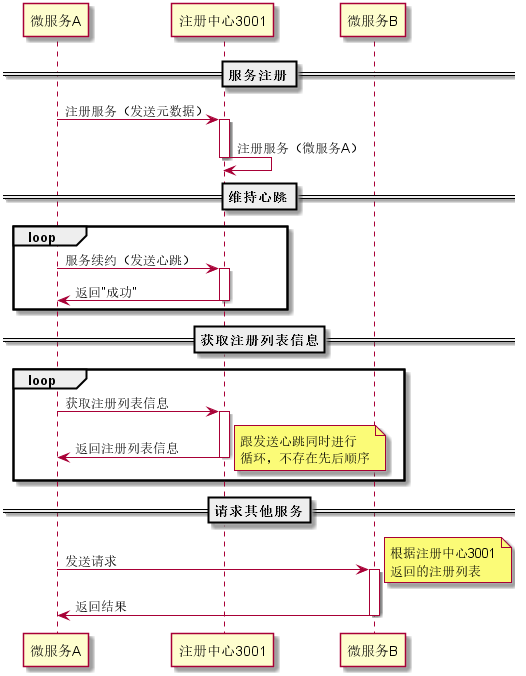
### 2、Service Provider

服务提供方，将自身服务注册到Eureka，从而使服务消费方能够找到

### 3、Service Consumer

服务消费方，从Eureka获取注册服务列表，从而能够消费服务

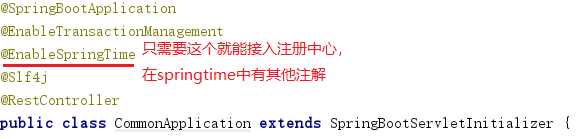
### 微服务与注册中心互动图



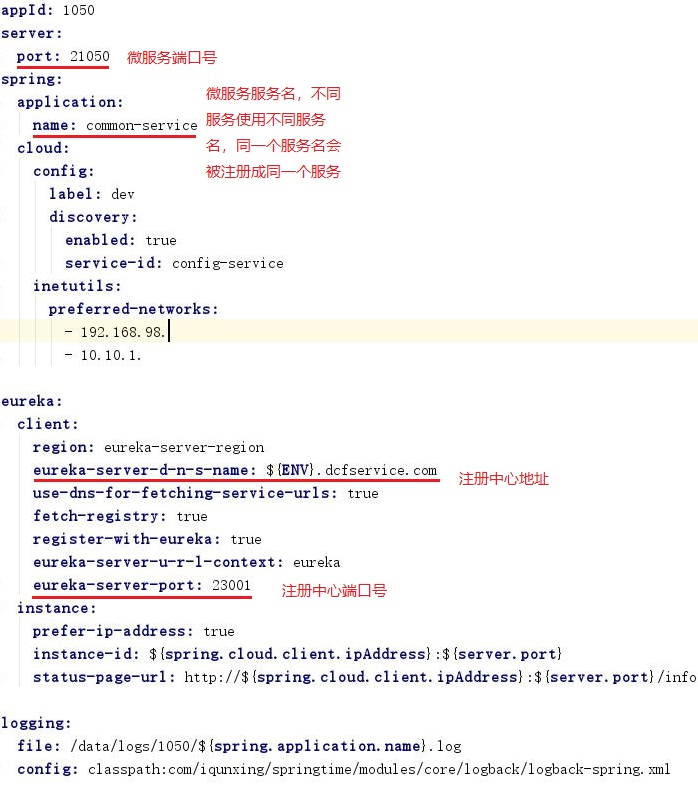
中文文档：<https://springcloud.cc/spring-cloud-dalston.html>

英文文档：<http://cloud.spring.io/spring-cloud-static/Camden.SR7/>

### 微服务配置



其他微服务注册到注册中心，改服务需要加入@EnableSpringTime的注解

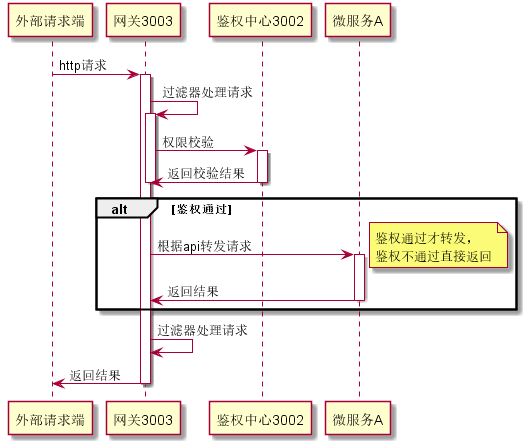


另外，需要在resources文件夹下创建bootstrap.yml文件，添加如下配置

## 3003网关

群星3003网关的主要功能是路由和过滤器。从网关工作的流程图可以看出网关会把外部的请求转发给目标微服务。在转发的过程中，网关的过滤器会对请求进行处理。过滤器主要负责把外部请求转发给鉴权中心，校验该请求是否有访问的权限。除此之外，过滤器还会对微服务的Response和整个流程中的错误进行处理。这里大致介绍网关的流程，详细情况可以看Spring Cloud 网关的文档。

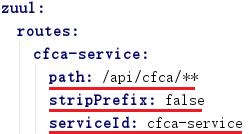
### 网关工作的流程图



## 网关配置

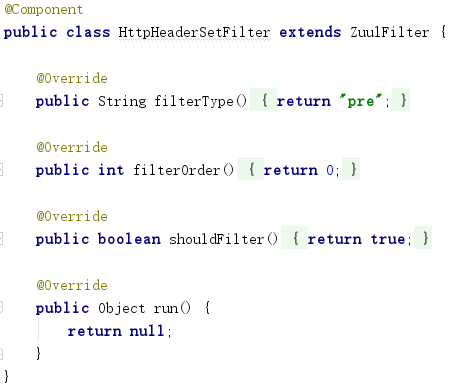
在pom文件中添加spring-cloud-starter-netflix-zuul模块，然后启动类中使用@EnableZuulProxy注解，就可以使这个项目具有网关的功能。

### 网关路由配置



如果想要在群星的3003网关中添加对新的微服务的路由，可以参照下图所示范例配置。途中path是外部请求访问网关的api，stripPrefix表示是否要省略前缀，当stripPrefix=true时网关会根据/\*\*之后的地址亲求指定微服务对应的接口，当stripPrefix=false时，网关会请求指定微服务的/api/cfca/\*\*接口。serviceId就是目标微服务注册到注册中心的服务名。

### 网关过滤器配置

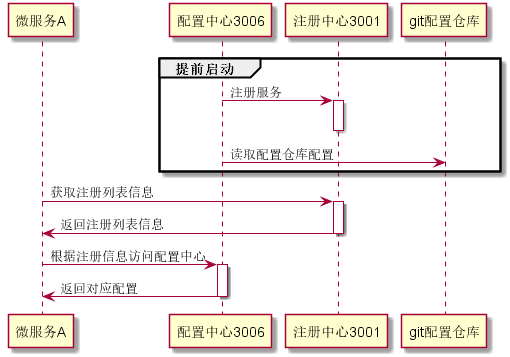


网关的过滤器继承了ZuulFilter。filterType是指过滤的类型，过滤会在哪个步骤进行，pre，routing，post，error分别表示路由之前，路由的时候，路由之后，发生错误的时候。filterOrder是过滤的顺序，shouldFilter表示是否过滤，可以写逻辑判断。run之中是过滤的具体逻辑。

## 3006配置中心

在Spring Cloud框架中，spring cloud config提供一个服务端和客户端去提供可扩展的配置服务。群星几乎所有接入Spring Cloud的微服务，都从配置中心获取配置。

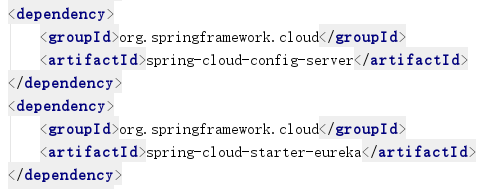
### 从配置中心获取配置流程图



配置中心的运作需要一个服务端和一个客户端。群星3006微服务就是哪个服务端。而所有其他从3006获取配置的微服务都是客户端。

如上面的流程图所示，3006在启动的时候，会先将自己注册到注册中心上，同时，3006会从群星的git配置仓库获取所有的配置。之后，在微服务A启动的时候，微服务A会把自己注册到注册中心，并从注册中心获取注册列表信息。然后，微服务A会根据获取到的注册列表信息（3006的信息），从配置中心获取自身的配置。

### 配置中心配置服务端



要配置一个配置中心服务端，首先需要在pom文件中增加相应的配置。之后，在微服务的启动类上加上@EnableConfigServer配置开启配置服务器。



在配置文件中加上如下配置：

url表示配置仓库地址

searchPaths表示查询的目录（文件夹），上图表示文件夹是各自的微服务名

username表示访问配置仓库的用户名

password表示访问配置仓库的密码

default-label表示配置中心获取配置的默认分支（branch）

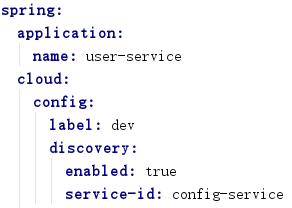
完成上述配置之后，配置中心基本可以正常工作了。如果一切顺利，访问http://localhost:8888/applicationn/dev应该可以通过配置中心获取到配置仓库中的application-dev.yml文件的配置。

群星的3006配置中心，因为要服务各个环境，所以会有略微复杂的配置文件，同时，启动类中也会有针对各个环境的代码。

### 配置中心客户端



配置中心服务端只需要上述配置（在SpringTime模块中已经有了，如果引用了SpringTime模块，则不再需要单独引用这个包），启动类上不需要其他特殊的注解。



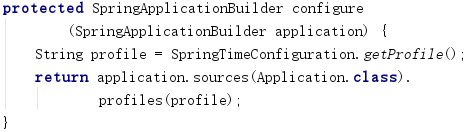
只需要在bootstrap.yml文件中加入连接配置中心服务端的配置。注意，如果微服务将要从配置中心获取的配置，包含微服务启动阶段要使用的数据，那么配置文件就不能用application.yml，而要用bootstrap.yml。因为bootstrap.yml会优先于其他东西加载。如果用application.yml，可能会造成“加载的时候还没获取到配置，然后报错”的情况。

label表示分支（branch）

discovery.enabled表示是否通过服务名发现配置中心

service-id表示配置中心的服务名

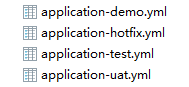
这边省略了name这个配置，这个配置代表了将要被客户端读取的配置文件的文件名，默认值为客户端的服务名。此处为user-service。

另外，配置中心客户端还有一个profile的配置项，声明了客户端读取的配置文件的后缀。假设我的profile=dev，那么再这个例子中，文件名为user-service-dev的文件会被客户端会读取。但是群星的微服务中，profile没有在配置中声明，而是在程序的启动类中声明的。因为群星有好多环境，我们需要把不同环境的微服务注册到不同环境的注册中心，并读取不同环境的配置。

如上面代码所示，微服务启动的时候，会先从SpringTimeConfiguration中获取profile（SpringTime后台会从每个环境的环境变量中获取ENV这个key对应的变量，根据这个变量获取profile），然后再把它加载到微服务中。

### 配置仓库

群星的配置中心访问的配置仓库，是git上的一个项目，叫config-repo。在这个项目中，每个微服务都有和它服务名相对应的文件夹（对应配置中心服务端配置中的searchPaths）。每个文件夹中有对应的文件，这些文件的后缀都是-dev，因为每个项目启动的时候，都会通过SpringTime加载profile，其中就有dev。



上图中的几个配置文件，则是公共配置（application这个文件名是默认公共配置），部署在对应环境的微服务，就会读取对应后缀的配置。举个例子，部署在uat环境的微服务，会从uat环境的环境变量中获取到ENV=uat的数据，然后把profile设置成dev，uat。然后微服务就会去读取serviceName-dev.yml，serviceName-uat.yml，application-dev.yml，application-uat.yml这四个文件。因为我们的配置仓库中只有serviceName-dev.yml和application-uat.yml这两个文件，所以只有这两个文件会被读取到。

中文文档：<https://springcloud.cc/spring-cloud-dalston.html>

英文文档：<http://cloud.spring.io/spring-cloud-static/Camden.SR7/>