## 一站式微服务解决方案Spring Cloud Alibaba

主讲：Cat 老师

北京动力节点教育科技有限公司  
2020 • 北京

动力节点•版权所有•禁止传播

## 快速回顾

1、Spring家族开源项目梳理

2、Spring Cloud下的开源项目梳理

3、Spring Cloud Alibaba下的开源及商业项目梳理

4、微服务的基础模型：服务消费者-注册中心-服务提供者

5、What is Nacos？

6、Nacos的运行环境部署（Java写的，springboot项目）

7、Nacos的后台web管控台

8、Nacos作为注册中心注册服务

9、Nacos作为注册中心发现/订阅服务

10、服务消费者负载均衡调用服务提供者（ribbon）restTemple、feign

11、Nacos宕机时服务消费者缓存注册中心信息

12、Nacos作为配置中心存储项目各种配置

13、Nacos作为配置中心支持自动配置刷新（不需要重启应用）

14、Nacos配置中心DataId+Group+Properties/yaml+配置内容（比较灵活）

15、Nacos配置中心多环境配置（profile）

即${spring.application.name}-${profile}.${file-extension:properties}

16、Nacos服务配置数据模型（命名空间、Group、Data Id）

17、Nacos数据持久化（mysql）

18、Ncaos集群部署（nginx）

19、主要调用方式：restTemplate、feign、ribbon（spring cloud）

20、流量控制Sentinel（流控、降级、热点、系统、授权 规则）

21、Sentinel Darshboard通信原理

22、Sentinel 对应用保护的三种方式

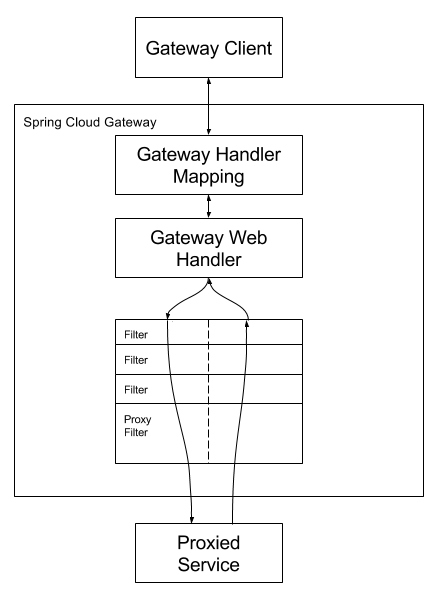
23、Sentinel整合RestTemplate流控熔断

24、Sentinel整合Feign流控熔断

25、Sentinel规则持久化（默认、pull模式、push模式）

26、Spring Cloud Gateway网关（核心概念、如何工作、路由、谓词11个、过滤器31个）

## Spring Cloud Gateway自定义谓词

****

Spring Cloud Gateway内置了一系列的路由谓词工厂，但是如果这些内置的路由谓词工厂不能满足业务需求的话，可以自定义路由谓词工厂来实现特定的需求；

下面列举两个例子：

1、要求请求必须携带一个token，并且token值等于指定的值，才能访问；  
2、要求某个服务的用户只允许在23:00 - 6:00这个时间段内才可以访问；

自定义谓词具体步骤：

（1）首先定义一个配置类，用于承载配置参数；

（2）定义一个路由谓词工厂；

注：TokenRoutePredicateFactory类，前面的Token与.yml配置文件里面配置的名字对应，后面的RoutePredicateFactory名字是固定的，不能随便写，这是Spring Cloud Gateway的约定，类名须为“谓词工厂名（比如：Token）” + RoutePredicateFactory

（3）在配置文件中启用该路由谓词工厂，即配置一个Token=123456；

时间格式不是随便配置，而是Spring Cloud Gateway的默认时间格式，采用JDK8里面的格式化：

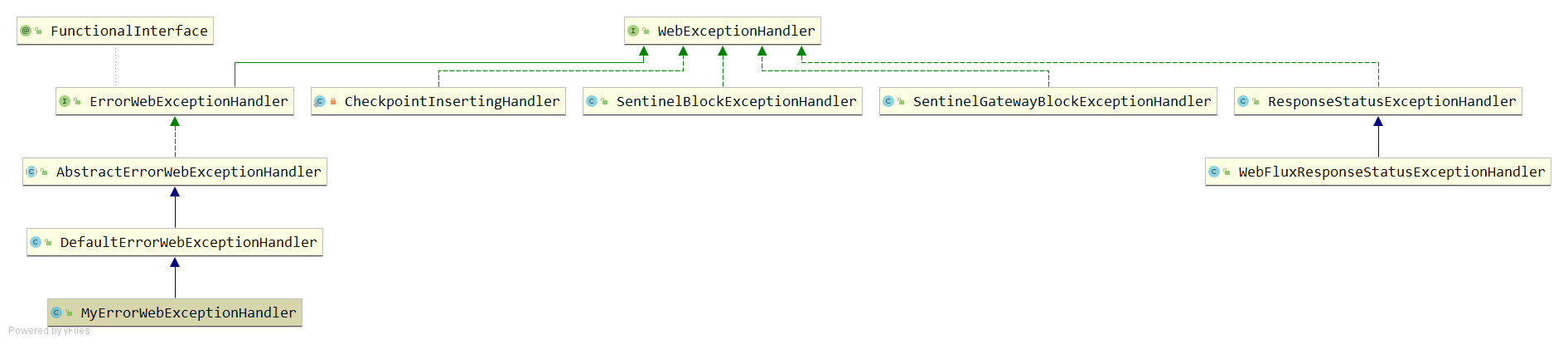
DateTimeFormatter dateTimeFormatter = DateTimeFormatter.*ofLocalizedTime*(FormatStyle.*SHORT*);  
String nowTime = dateTimeFormatter.format(ZonedDateTime.*now*());  
System.*out*.println(nowTime);

到此为止就实现了一个自定义路由谓词工厂，若此时token值不相等，不在允许的访问时间段内，访问就会报404；

## Spring Cloud Gateway谓词不匹配404处理

处理的顶层接口是WebExceptionHandler

默认实现是DefaultErrorWebExceptionHandler



我们需要覆盖它的默认实现DefaultErrorWebExceptionHandler，覆盖里面的方法，在方法里面编写我们想要返回的结果，参看我们git代码；

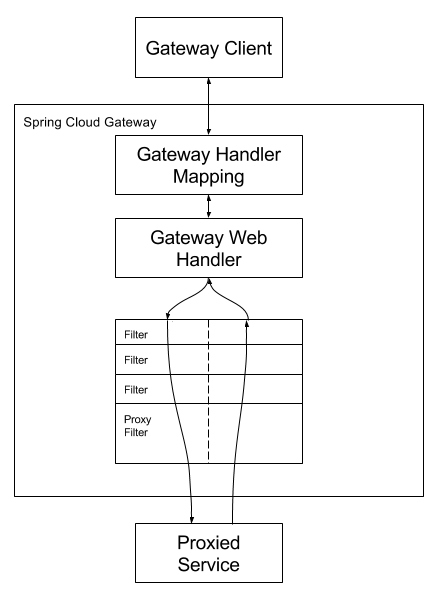
## Spring Cloud Gateway自定义路由过滤器

网关过滤器的顶层接口是GatewayFilterFactory

通常情况下可以继承AbstractGatewayFilterFactory实现自定义网关过滤器；

或者继承AbstractNameValueGatewayFilterFactory，该方式配置方式更简单，然后覆盖里面的一个方法，具体参考一下我们的样例代码，实际上理解流程思路即可，真正需要做的时候，查一下就可以；

## Spring Cloud Gateway全局过滤器

****

上面的过滤器工厂是执行在指定路由之上，可以称为路由过滤器（或者局部过滤器），而全局过滤器是作用于所有的路由上，对所有的路由进行过滤；

全局过滤器的顶层接口是GlobalFilter ，和GatewayFilter 有一样的接口定义，只不过GlobalFilter 会作用于所有路由；

全局过滤器有执行顺序问题，通过getOrder()方法的返回值决定执行顺序，数值越小越靠前执行；

Spring cloud gateway默认内置了很多全局过滤器，比如：

1. Combined Global Filter and GatewayFilter Ordering

2. Forward Routing Filter

3. The LoadBalancerClient Filter

4. The ReactiveLoadBalancerClientFilter

5. The Netty Routing Filter

6. The Netty Write Response Filter

7. The RouteToRequestUrl Filter

8. The Websocket Routing Filter

9. The Gateway Metrics Filter

10. Marking An Exchange As Routed

当然我们也可以自定义全局过滤器，具体在我们的代码中有样例实现，可以参考；

## Spring Cloud Gateway集成ribbon负载均衡

实现原理是在全局LoadBalancerClientFilter中进行拦截，然后再该过滤器中依赖LoadBalancerClient loadBalancer，而此负载均衡接口的具体实现是RibbonLoadBalancerClient，即spring cloud gateway已经整合好了ribbon，已经可以实现负载均衡，我们不需要做任何工作，网关对后端微服务的转发就已经具有负载均衡功能；

## Spring Cloud Gateway集成Sentinel

网关集成Sentinel是为了流控熔断降级，具体集成整合步骤如下：

1. 添加依赖；

*<!-- sentinel-spring-cloud-gateway-adapter -->*<dependency>  
 <groupId>com.alibaba.csp</groupId>  
 <artifactId>sentinel-spring-cloud-gateway-adapter</artifactId>  
 <version>1.7.2</version>  
</dependency>

1. 添加sentinel控制台配置；

*#sentinel dashboard管理后台*

spring：

cloud： sentinel:  
 eager: true  
 transport:  
 dashboard: 192.168.172.128:8080

1. 写代码，在spring容器中配置一个Sentinel的全局过滤器；

@Bean  
@Order(-1)  
public GlobalFilter sentinelGatewayFilter() {  
 return new SentinelGatewayFilter();  
}

4、可以进行测试；

## Spring Cloud Gateway集成Sentinel规则持久化

规则持久化到文件的步骤：

1. 、配置依赖；

*<!--sentinel-datasource-extension数据源扩展-->*<dependency>  
 <groupId>com.alibaba.csp</groupId>  
 <artifactId>sentinel-datasource-extension</artifactId>  
</dependency>

（2）配置文件；

暂时不需要配置；

（3）写代码；

见项目中的 FileDataSourceInit 类；

（4）配置SPI；

resources\META-INF\services\com.alibaba.csp.sentinel.init.InitFunc

规则持久化到Nacos的步骤；

1、添加sentinel-datasource-nacos依赖；

*<!--sentinel数据持久化-->*<dependency>  
 <groupId>com.alibaba.csp</groupId>  
 <artifactId>sentinel-datasource-nacos</artifactId>  
</dependency>

2、application.properties配置持久化数据源；

*#配置sentinel规则持久化到nacos*datasource:  
 ncs1:  
 nacos:  
 server-addr: 192.168.172.128:80  
 data-id: ${spring.application.name}.json  
 group-id: DEFAULT\_GROUP  
 rule-type: flow  
 data-type: json

degrade:

nacos:

server-addr: 192.168.172.128:80

data-id: ${spring.application.name}-degrade.json

group-id: DEFAULT\_GROUP

rule-type: degrade

data-type: json

3、在nacos配置中心配置流控规则；

[

{

"resource": "abc",

"controlBehavior": 0,

"count": 1.0,

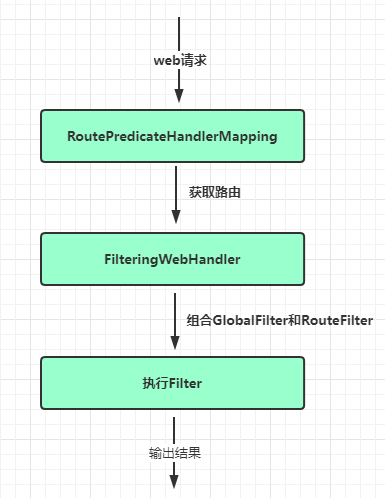
"grade": 1,

"limitApp": "default",

"strategy": 0

}  
]

## Spring Cloud Gateway内部流程源码分析



（1）根据自动装配spring-cloud-gateway-core.jar的spring.factories；

（2）GatewayClassPathWarningAutoConfiguration检查前端控制器；

（3）网关自动配置GatewayAutoConfiguration；

（4）RoutePredicateHandlerMapping.getHandlerInternal(...)获取Route；

（5）执行FilteringWebHandler