**史上最简单的SpringCloud教程 | 第五篇: 服务网关(zuul)**

[方志朋](https://www.jianshu.com/u/f2550db5eca3)

0.1062017.04.09 21:19:18字数 743阅读 5,957

在微服务架构中，需要几个关键的组件，服务注册与发现、服务消费、负载均衡、断路器、智能路由、配置管理等，由这几个组件可以组建一个简单的微服务架构，如下图：

Azure (1).png

注意：A服务和B服务是可以相互调用的，作图的时候忘记了。并且配置服务也是注册到服务注册中心的。

客户端的请求首先经过负载均衡（zuul、Ngnix），再到达服务网关（zuul集群），然后再到具体的服务，服务统一注册到高可用的服务注册中心集群，服务的所有的配置文件由配置服务管理（下一篇文章讲述），配置服务的配置文件放在git仓库，方便开发人员随时改配置。

一、Zuul简介

Zuul的主要功能是路由和过滤器。路由功能是微服务的一部分，比如／api/user映射到user服务，/api/shop映射到shop服务。zuul实现了负载均衡。

zuul有以下功能：

Authentication

Insights

Stress Testing

Canary Testing

Dynamic Routing

Service Migration

Load Shedding

Security

Static Response handling

Active/Active traffic management

二、准备工作

继续使用上一节的工程。在原有的工程上，创建一个新的工程。

三、创建service-zuul工程

其pom.xml文件如下：

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.forezp</groupId>
<artifactId>service-zuul</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>service-zuul</name>
<description>Demo project for Spring Boot</description>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>1.5.2.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
<java.version>1.8</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-eureka</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-zuul</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>Dalston.RC1</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
<repositories>
<repository>
<id>spring-milestones</id>
<name>Spring Milestones</name>
<url>https://repo.spring.io/milestone</url>
<snapshots>
<enabled>false</enabled>
</snapshots>
</repository>
</repositories>
</project>

在其入口applicaton类加上注解@EnableZuulProxy，开启zuul：

@EnableZuulProxy
@EnableEurekaClient
@SpringBootApplication
public class ServiceZuulApplication {
public static void main(String[] args) {
SpringApplication.run(ServiceZuulApplication.class, args);
}
}

加上配置文件：

eureka:
client:
serviceUrl:
defaultZone: http://localhost:8761/eureka/
server:
port: 8769
spring:
application:
name: service-zuul
zuul:
routes:
api-a:
path: /api-a/\*\*
serviceId: service-ribbon
api-b:
path: /api-b/\*\*
serviceId: service-feign

首先向eureka注册自己，端口为8769，服务名为service-zuul；以/api-a/ 开头的请求都指向service-ribbon；以/api-b/开头的请求都指向service-feign；

依次运行这五个工程;打开浏览器访问：[http://localhost:8769/api-a/hi?name=forezp](https://link.jianshu.com/?t=http://localhost:8769/api-a/hi?name=forezp) ;浏览器显示：

hi forezp,i am from port:8762

打开浏览器访问：[http://localhost:8769/api-b/hi?name=forezp](https://link.jianshu.com/?t=http://localhost:8769/api-b/hi?name=forezp) ;浏览器显示：

hi forezp,i am from port:8762

这说明zuul起到了路由的作用；

四、服务过滤

zuul不仅只是路由，并且还能过滤，做一些安全验证。继续改造工程；

@Component
public class MyFilter extends ZuulFilter{
private static Logger log = LoggerFactory.getLogger(MyFilter.class);
@Override
public String filterType() {
return "pre";
}
@Override
public int filterOrder() {
return 0;
}
@Override
public boolean shouldFilter() {
return true;
}
@Override
public Object run() {
RequestContext ctx = RequestContext.getCurrentContext();
HttpServletRequest request = ctx.getRequest();
log.info(String.format("%s >>> %s", request.getMethod(), request.getRequestURL().toString()));
Object accessToken = request.getParameter("token");
if(accessToken == null) {
log.warn("token is empty");
ctx.setSendZuulResponse(false);
ctx.setResponseStatusCode(401);
try {
ctx.getResponse().getWriter().write("token is empty");
}catch (Exception e){}
return null;
}
log.info("ok");
return null;
}
}

filterType：返回一个字符串代表过滤器的类型，在zuul中定义了四种不同生命周期的过滤器类型，具体如下：

pre：路由之前

routing：路由之时

post： 路由之后

error：发送错误调用

filterOrder：过滤的顺序

shouldFilter：这里可以写逻辑判断，是否要过滤，本文true,永远过滤。

run：过滤器的具体逻辑。可用很复杂，包括查sql，nosql去判断该请求到底有没有权限访问。