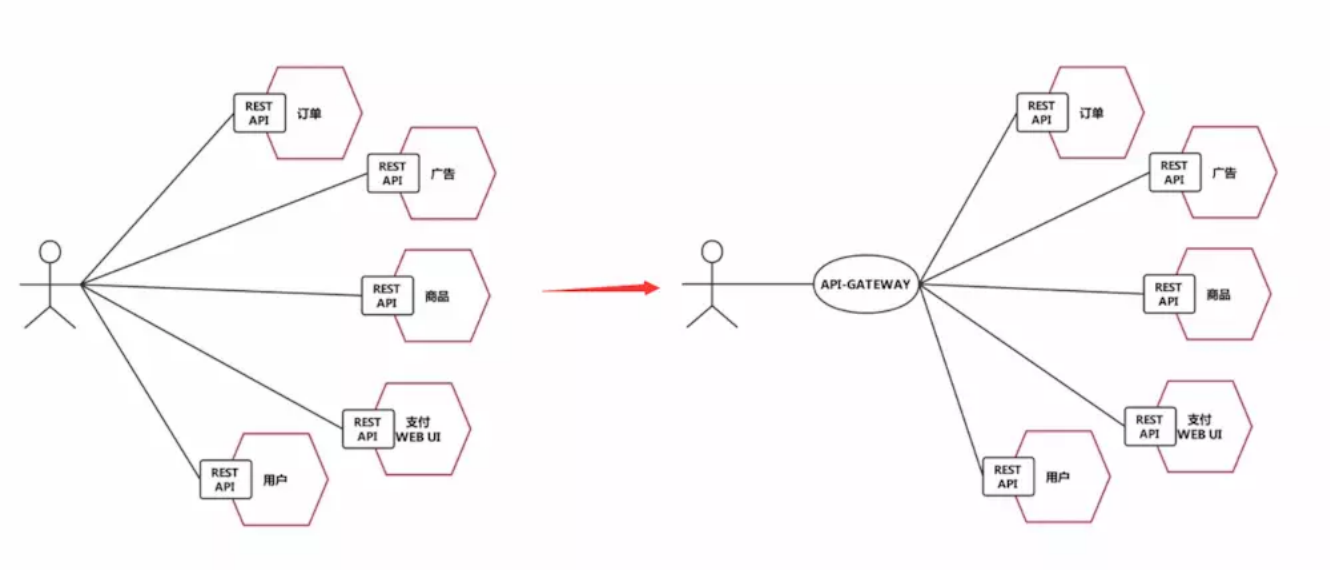
路由网关（zuul）

> 更多学习资料，一点课堂：www.yidiankt.com
>
> 一点课堂QQ群：984370849 ， QQ：2868289889
>
> 微信号：chengweixin9
>
> 免费公开课：https://ke.qq.com/course/394307

为什么需要服务网关



在分布式系统系统中，有商品、订单、用户、广告、支付等等一大批的服务，前端怎么调用呢？和每个服务一个个打交道？这显然是不可能的，这就需要有一个角色充当所有请求的入口，这个角色就是服务网关（API gateway）。

客户端直接与微服务通讯的问题

1. 客户端会多次请求不同的微服务，增加了客户端的复杂性。
2. 存在跨域请求，在一定场景下处理相对复杂。
3. 认证复杂，每个服务都需要独立认证。
4. 难以重构，随着项目的迭代，可能需要重新划分微服务。例如，可能将多个服务合并成一个或者将一个服务拆分成多个。如果客户端直接与微服务通讯，那么重构将会很难实施。

网关的优点

1. 易于监控。可在微服务网关收集监控数据并将其推送到外部系统进行分析。
2. 易于认证。可在微服务网关上进行认证。然后再将请求转发到后端的微服务，而无须在每个微服务中进行认证。
3. 减少了客户端与各个微服务之间的交互次数。

> 为了解决上面这些问题，我们需要将权限控制这样的东西从我们的服务单元中抽离出去，而最适合这些逻辑的地方就是处于对外访问最前端的地方，我们需要一个更强大一些的均衡负载器，它就是本文将来介绍的：服务网关。

什么是网关?

服务网关是微服务架构中一个不可或缺的部分。通过服务网关统一向外系统提供REST API的过程中，除了具备服务路由、均衡负载功能之外，它还具备了权限控制等功能。Spring Cloud Netflix 中的 Zuul 就担任了这样的一个角色，为微服务架构提供了前门保护的作用，同时将权限控制这些较重的非业务逻辑内容迁移到服务路由层面，使得服务集群主体能够具备更高的可复用性和可测试性。

使用zuul

1. 新建zuul的module工程
2. pom文件

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
    </dependency>
</dependencies>
```

3. application.yml配置文件

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8888/eureka/

server:
  port: 9000

spring:
  application:
    name: yidiankt-zuul

zuul:
  routes:
    api-order:
      path: /api-order/**
      serviceId: yidiankt-order
    api-user:
      path: /api-user/**
      serviceId: yidiankt-user
```

4. 启动类

```
@SpringBootApplication
// 开启zuul功能
@EnableZuulProxy
@EnableEurekaClient
public class ZuulApp {
    public static void main(String[] args) {
        SpringApplication.run(ZuulApp.class, args);
    }
}
```

5. 测试路由访问

6. 配置统一前缀访问

```
zuul:
  routes:
    api-order:
      path: /api-order/**
      serviceId: yidiankt-order
    api-user:
      path: /api-user/**
      serviceId: yidiankt-user
  #前缀访问
  prefix: /yidiankt
```

7. 忽略服务名serviceId访问

```
zuul:
  routes:
    api-order:
      path: /api-order/**
      serviceId: yidiankt-order
    api-user:
      path: /api-user/**
      serviceId: yidiankt-user
  prefix: /yidiankt
  ignored-services: "*"
```

8. 配url绑定映射

```
zuul:
  routes:
    testurl:
      url: http://www.iduoan.com
      path: /testurl/**
```

9. 配置URL映射负载

```
ribbon:
  eureka:
    enabled: false
```

```
#Ribbon请求的微服务serviceId
yidiankt-user:
  ribbon:
    listOfServers: http://www.huya.com,http://www.douyu.com

zuul:
  routes:
    testurl:
      serviceId: yidiankt-user
      path: /testurl/**
```

zuul过滤器

Zuul本身是一系列过滤器的集成，那么他当然也就提供了自定义过滤器的功能，zuul提供了四种过滤器：前置过滤器，路由过滤器，错误过滤器，简单过滤器，实现起来也十分简单，只需要编写一个类去实现zuul提供的接口。

使用zuul过滤器

1. 添加过滤器类

```
@Component
public class MyFilter2 extends ZuulFilter {

    /**
     * 类型包含 pre post route error
     * pre 代表在路由代理之前执行
     * route 代表代理的时候执行
     * error 代表出现错的时候执行
     * post 代表在route 或者是 error 执行完成后执行
     */
    @Override
    public String filterType() {
        // 路由之前(前置过滤器)
        return FilterConstants.PRE_TYPE;
    }

    @Override
    public int filterOrder() {
        // 优先级，数字越大，优先级越低
        return 2;
    }

    @Override
    public boolean shouldFilter() {
        // 是否执行该过滤器，true代表需要过滤
        return true;
    }

    @Override
    public Object run() {
```

```
        System.out.println("22222222222222222222");
        return null;
    }
}
```