链滴

Spring-Cloud 系列第 7 篇：spring-cloud-zuul

作者：xjtushilei

原文链接：https://ld246.com/article/1497580348342

来源网站：链滴

自学spring-cloud系列，越来越感觉spring-cloud很强大！

主要分为以下几篇：

1. spring-cloud-config: 分布式配置管理
2. spring-cloud-eureka: 服务注册与发现
3. spring-cloud-eureka-consumer: 远程服务调用和及其负载均衡
4. spring-cloud-Hystrix: 熔断器保证服务高可用
5. spring-cloud-config-eureka-ribbon: 分布式配置管理的高可用
6. spring-cloud-bus: 配置信息的实时更新
7. spring-cloud-zuul: 网关基础服务

介绍

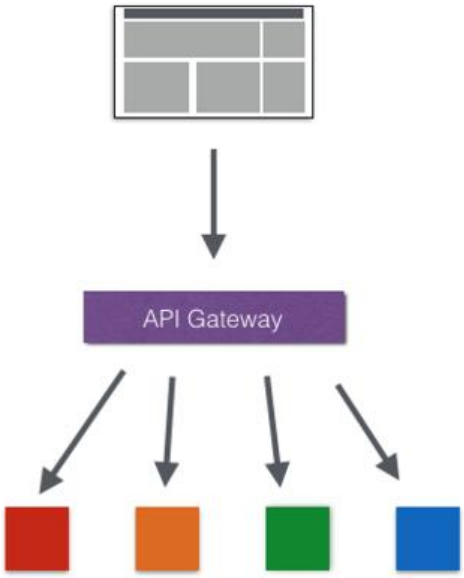
前面的文章我们介绍了，Eureka用于服务的注册于发现，Feign支持服务的调用以及均衡负载，Hystri处理服务的熔断防止故障扩散，Spring Cloud Config服务集群配置中心，似乎一个微服务框架已经成了。

我们还是少考虑了一个问题，外部的应用如何来访问内部各种各样的微服务呢？在微服务架构中，后服务往往不直接开放给调用端，而是通过一个API网关根据请求的url，路由到相应的服务。当添加AP网关后，在第三方调用端和服务提供方之间就创建了一面墙，这面墙直接与调用方通信进行权限控制后将请求均衡分发给后台服务端。

为什么需要API Gateway

1、简化客户端调用复杂度

在微服务架构模式下后端服务的实例数一般是动态的，对于客户端而言很难发现动态改变的服务实例访问地址信息。因此在基于微服务的项目中为了简化前端的调用逻辑，通常会引入API Gateway作为量级网关，同时API Gateway中也会实现相关的认证逻辑从而简化内部服务之间相互调用的复杂度。



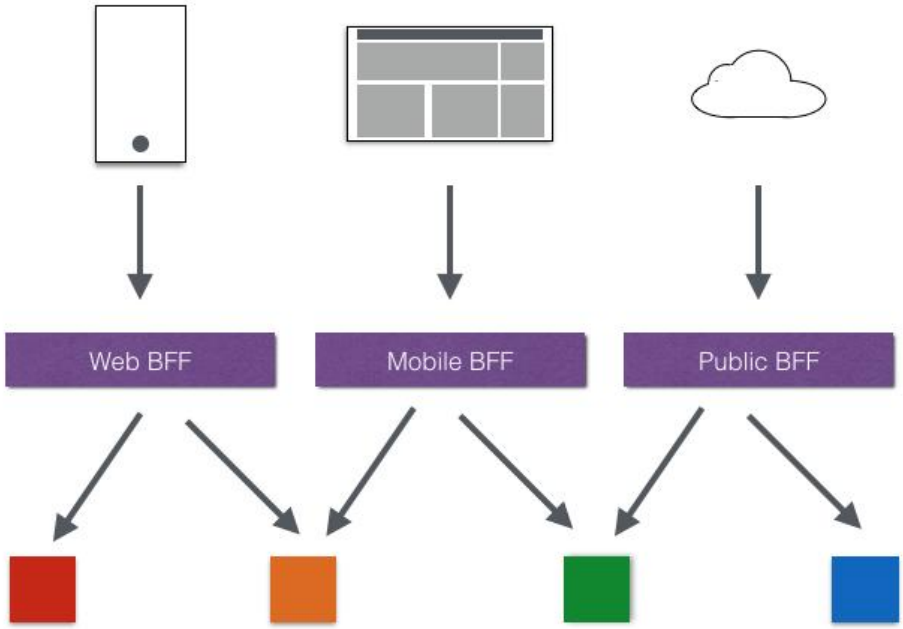
2、数据裁剪以及聚合

通常而言不同的客户端对于显示时对于数据的需求是不一致的，比如手机端或者Web端又或者在低延的网络环境或者高延迟的网络环境。

因此为了优化客户端的使用体验，API Gateway可以对通用性的响应数据进行裁剪以适应不同客户端使用需求。同时还可以将多个API调用逻辑进行聚合，从而减少客户端的请求数，优化客户端用户体验

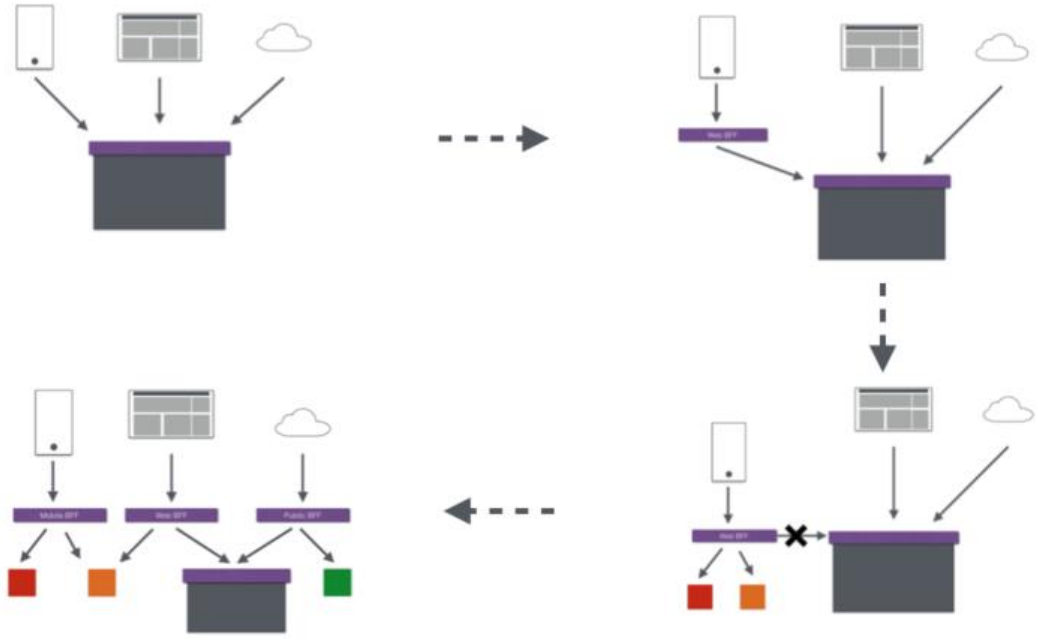
3、多渠道支持

当然我们还可以针对不同的渠道和客户端提供不同的API Gateway,对于该模式的使用由另外一个大家知的方式叫Backend for front-end, 在Backend for front-end模式当中，我们可以针对不同的客户分别创建其BFF，进一步了解BFF可以参考这篇文章：Pattern: Backends For Frontends



4、遗留系统的微服务化改造

对于系统系统而言进行微服务改造通常是由于原有的系统存在或多或少的问题，比如技术债务，代码量，可维护性，可扩展性等等。API Gateway的模式同样适用于这一类遗留系统的改造，通过微服务的改造逐步实现对原有系统中的问题的修复，从而提升对于原有业务响应力的提升。通过引入抽象层逐步使用新的实现替换旧的实现。



Spring Cloud Zuul

在Spring Cloud体系中， Spring Cloud Zuul就是提供负载均衡、反向代理、权限认证的一个API gat

原文链接：Spring-Cloud 系列第 7 篇：spring-cloud-zuul

way。

Spring Cloud Zuul路由是微服务架构的不可或缺的一部分，提供动态路由，监控，弹性，安全等的缘服务。Zuul是Netflix出品的一个基于JVM路由和服务端的负载均衡器。

简单使用

依赖：

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

配置文件:

```
server:
  port: 8088

spring:
  application:
    name: gateway-service-zuul
zuul:
  routes:
    name1:
      path: /test/*
      url: http://www.xjtushilei.com/
```

将所有 http://localhost:8088/test 开头的服务，重定向到我的博客。

运行：

```
@SpringBootApplication
@EnableEurekaClient //如果需要注册倒eureka服务中心的话，启用这个
@EnableZuulProxy
public class SpringCloudZuulApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringCloudZuulApplication.class, args);
    }
}
```

测试

访问 http://localhost:8088/test/312 得到如下图

可以看出重定向成功了，为什么css等不见了，因为没有给他们配置转发路由。

服务调用

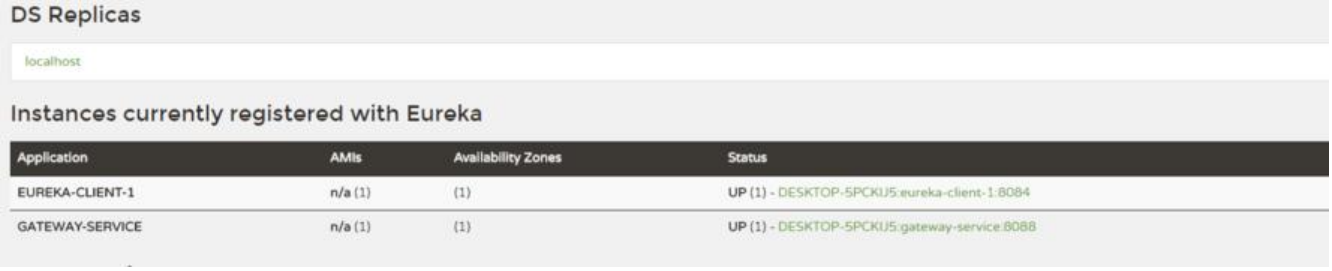
application.yml 配置文件：

```
server:
  port: 8088

spring:
  application:
    name: gateway-service

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

这里有一个bug，因为我喜欢采用激活profile的方式来进行开发，但是这次真的是遇到了好一个超级bg。

因为不能在 application-dev.yml 这样的文件里定义服务中心的Zone。有点不人性化。因为开发和终肯定不是在一个Zone里的，所以这里应该可以改进的。

随便启动一个之前的服务，有restapi即可



然后测试：

访问 http://localhost:8088/eureka-client-1/hi 和 http://localhost:8084/hi 返回的结果是一样的

网关的默认路由规则

spring cloud zuul已经帮我们做了默认网关配置。默认情况下，Zuul会代理所有注册到Eureka Serve的微服务，并且Zuul的路由规则如下：http://ZUUL_HOST:ZUUL_PORT/微服务在Eureka上的serviced/**会被转发到serviceId对应的微服务。

到此zuul的基本使用我们就介绍完了。关于zuul更高级使用，我们下篇再来介绍。

示例源码

所有源码在我的github仓库里，传送门：https://github.com/xjtushilei/spring-cloud-simples.git

支持

如果你喜欢~ 给个星