讨论课7：Zuul网关

Zuul类图

@author: Ringoer

关于Filter生命周期

四种Filter生命周期：

> - pre：在 Zuul 按照规则路由到下级服务之前执行。如果需要对请求进行预处理，比如鉴权、限流等，都应考虑在此类 Filter 实现。
> - route：这类 Filter 是 Zuul 路由动作的执行者，是 Apache Http Client 或 Netflix Ribbon 构建和发送原始 HTTP 请求的地方，目前已支持 Okhttp。
> - post：这类 Filter 是在源服务返回结果或者异常信息发生后执行的，如果需要对返回信息做一些处理，则在此类 Filter 进行处理。
> - error：在整个生命周期内如果发生异常，则会进入 error Filter，可做全局异常处理。

实际上在本次课程设计中zuul主要用来处理token，并未用到`route`类型的Filter,因为路由的功能已经交付eureka实现了

自定义zuul

zuul的Filter们通常是需要根据项目需求自定义的，对filter的自定义主要工作为重载以下四个函数（继承自ZuulFilter）

> - filterType：过滤器的类型。它决定过滤器在请求的哪个生命周期中执行，定义为pre，代表会在请求被路由之前执行。
> - filterOrder：过滤器的执行顺序。当请求在一个阶段中存在多个过滤器时，需要根据该方法返回的值来依次执行。
> - shouldFilter：判断该过滤器是否需要被执行。
> - run：过滤器的具体执行逻辑。

关于token的实现

> JWT（JSON Web Token）是一种JSON格式来规约Token或者Session的协议，或者说JWT是一个加密的协议。用来给Token加密的。 包括三部分：
>
> Header头部：指定JWT使用的签名算法。
>
> Payload载荷：包含一些自定义与非自定义的认证信息。
>
> Signature签名：将头部与载荷使用“.”连接之后，使用头部的签名算法生成签名信息并拼接带尾部

几个实现Filter的重要的类

（代码向，复习可掠过）

RequestContext

> 用于filter之间共享状态，内部是用 ThreadLocal 保存每个请求的一些信息， 包括请求路由、错误信息、**HttpServletRequest**、**HttpServletResponse**，这使得一些操作是十分可靠的

HttpServletRequest

> HttpServletRequest对象代表客户端的请求，当客户端通过HTTP协议访问服务器时，HTTP请求头中的所有信息都封装在这个对象中，通过这个对象提供的方法，可以获得客户端请求的所有信息。

讨论课8：Eureka

三个核心角色

1. 服务注册中心

Eureka的服务端应用，提供服务注册和发现功能。对外暴露自己的地址

2.服务提供者

提供服务的应用，可以是SpringBoot应用，也可以是其它任意技术实现，只要对外提供的是Rest风格服务即可。提供者定期通过http方式向Eureka刷新自己的状态

3.服务消费者

消费应用从注册中心获取服务列表，从而得知每个服务方的信息，知道去哪里调用服务方

心跳💗：如何实现服务的自动注册、发现、状态监控。

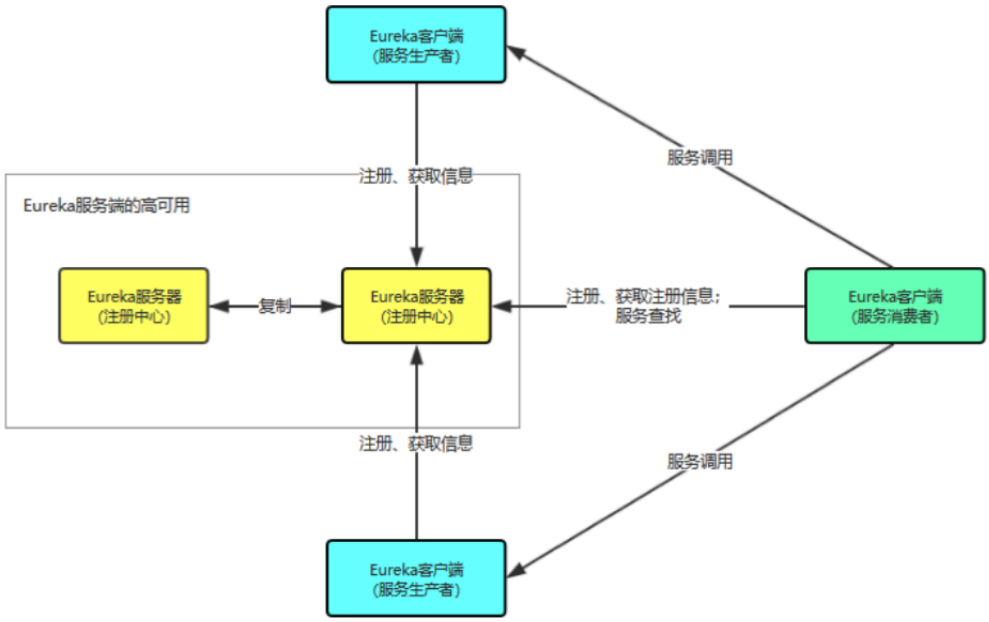
服务提供方与Eureka之间通过“心跳”机制进行监控，当某个服务提供方出现问题，Eureka自然会把它从服务列表中剔除。

> 在应用启动后，节点们将会向Eureka Server发送心跳,默认周期为30秒，如果Eureka Server在多个心跳周期内没有接收到某个节点的心跳，Eureka Server将会从服务注册表中把这个服务节点移除

高可用的Eureka Server

所谓的高可用注册中心，其实就是把EurekaServer自己也作为一个服务进行注册，这样多个EurekaServer之间就能互相发现对方，从而形成集群。

主要流程



关于微服务的负载均衡

每个微服务都是一个Eureka-Client，我们把每个SpringBootApplication都向注册中心注册一个服务。

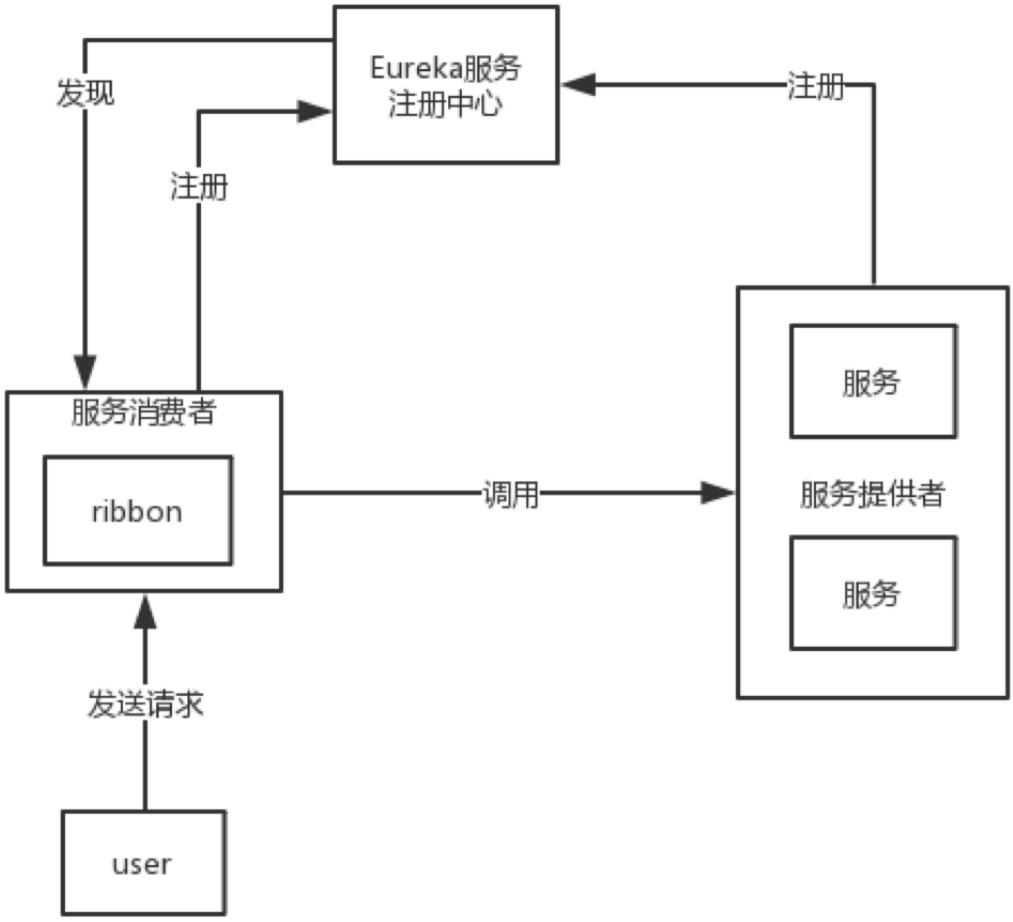
有时候，某个服务的工作量比较大的时候，我们可以多注册几个同名称的微服务，从而让他们交替工作，减轻单个服务的压力。多台提供者在交替工作，从而达到了一个负载均衡的作用。

讨论课9：Ribbon

是什么

是Netflix发布的开源项目，主要功能是提供用于Client的软件负载均衡算法。Ribbon客户端组件提供一系列完善的配置项如连接超时，重试等

一张加了ribbon的eureka流程图



RestTemplate是什么

RestTemplate是Spring提供的用于访问Rest服务的客户端

RestTemplate提供了多种便捷访问远程Http服务的方法,能够大大提高客户端的编写效率。

RestTemplate自动封装request，无需手动封装

RestTemplate工作流程

1. 当服务消费者调用服务提供者时， 一个被@LoadBalanced注解修饰的RestTemplate对象向外发起HTTP请求（以服务名为host的URI请求）
2. LoadBalancerInterceptor类的intercept函数：拦截请求
3. LoadBalancerClient类的execute函数：根据服务名来选择实例并发起实际的请求（实际的host:port形式的具体地址）

Feign

Feign是声明式的web service客户端，它让微服务之间的调用变得更简单了，类似controller调用service。

Spring Cloud集成了Ribbon和Eureka，可在使用Feign时提供负载均衡的http客户端。

Feign工作流程

1. Feign 请求URL (http://<服务名称>/)

2. LoadBalancerFeignClient， 提取服务标识<服务名称>。根据服务名称在上下文中查找对应服务的负载均衡器FeignLoadBalancer
3. 负载均衡器：挑选一个合适的Server实例，并记录对应Server实例的调用统计信息。

负载均衡器核心LoadBalancer的核心

三个职责：

1. 维护Sever列表的数量(新增、更新、删除等)
2. 维护Server列表的状态(状态更新)
3. 当请求Server实例时，返回最合适的Server实例

1.如何维护Server表(新增、更新、删除)

1. 基于定时任务拉取服务列表方式

> 其内部维护了一个周期性的定时任务，拉取最新的服务列表，然后将最新的服务列表更新到ServerList之中
>
> 默认频率为30s

2. 基于Eureka服务事件通知的方式更新

> 当Eureka注册中心发生了Server服务注册信息变更时，会将消息通知发送到EurekaNotificationServerListUpdater上，然后此Updater触发刷新ServerList：

2. 负载均衡器如何维护服务实例的状态?

NIWDiscoveryPing类：使用Eureka作为服务注册和发现，校验服务是否可用，通过监听Eureka 服务更新来更新Ribbon的Server状态，即取 Eureka Server 的实例状态作为Ribbon服务的状态

3.如何从服务列表中挑选一个合适的服务实例?

步骤1：通过ServerListFilter将服务列表进行一次过滤，返回满足过滤器条件的服务实例列表。 步骤2：应用Rule规则，结合服务实例的统计信息,返回满足规则的某一个服务实例。

熔断的工作原理

当链接失败超过阈值时，将进行熔断

当有某个服务存在多个实例时，在请求的过程中，负载均衡器会统计每次请求的情况（请求相应时间，是否发生网络异常等），当出现了请求出现累计重试时，负载均衡器会标识当前服务实例，设置当前服务实例的断路的时间区间，在此区间内，当请求过来时，负载均衡器会将此服务实例从可用服务实例列表中暂时剔除，优先选择其他服务实例