参考文档： <https://www.jianshu.com/p/1bd66db5dc46>

Ribbon是基于http和tcp的客户端负载均衡工具

Ribbon实现客户端负载均衡

微服务的负载均衡都是在客户端做的。比如dubbo，有注册中心的负载均衡都是在客户端

服务端负载均衡，分为软件负载均衡（nginx）、硬件负载均衡（f5、netscaler）

Web负载均衡方案 keepalive+nginx/haproxy， mysql负载均衡用haproxy/Lvs+mycat+keepalive

通过Spring Cloud Ribbon的封装，我们在微服务架构中使用客户端负载均衡调用非常简单，只需要如下两步：

        ▪️服务提供者只需要启动多个服务实例并注册到一个注册中心或是多个相关联的服务注册中心。

        ▪️服务消费者直接通过调用被@LoadBalanced注解修饰过的RestTemplate来实现面向服务的接口调用。

        这样，我们就可以将服务提供者的高可用以及服务消费者的负载均衡调用一起实现了。

@LoadBalanced开始客户单负载均衡。

针对各个服务治理框架的Ribbon自动化整合配置，

比如Eureka中的

org.springframework.cloud.netflix.ribbon.eureka.RibbonEurekaAutoConfiguration，

Consul中的

org.springframework.cloud.consul.discovery.RibbonConsulAutoConfiguration

RestTemplate是spring中调用http接口的工具，比httpclient简单方便很多。

RoadbalancerClient中的URI reconstructURI（ServiceInstance instance， URI original）来讲服务名换成真实的ip和port

ServiceInstanceChooser的ServiceInstance choose（String serviceId）根据负载均衡选出一个实例

T execute（String serviceId， ServiceInstance instance， LoadBalancerRequest request）

ribbon负载均衡自动配置类LoadBalancerAutoConfiguration

当一个被@LoadBalanced注解修饰的RestTemplate对象向外发起HTTP请求时，会被LoadBalancerInterceptor类的intercept函数所拦截。由于我们在使用RestTemplate时候采用了服务名作为host，所以直接从HttpRequest的URI对象中通过getHost()就可以拿到服务名，然后调用execute函数去根据服务名来选择实例并发起实际的请求

获取服务实例的时候，没有使用LoadBalancerClient中的choose函数，而是用了ILoadBalancer中的chooseServer函数。

ILoadBalancer定义了客户端负载均衡需要的方法

**addServers**:向负载均衡器中维护的实例列表增加服务实例。

**chooseServer**:通过某种策略，从负载均衡器中挑选出一个具体实例已经停止服务，不然负载均衡器在下一次获取服务实例清单前都会认为服务实例均是正常服务。

**markServerDwon**:用来通知和标识负载均衡器中某个实例已经停止服务，不然负载均衡器在下一次获取服务实例清单前都会认为服务实例均是正常服务的。

**getReachableServers**:获取当前正常服务的实例列表。

**getAllServers**:获取所有已知的服务实例列表，包括正常服务和停止服务的实例。

BaseloadBalancer实现了负载均衡最基本的功能。

DynamicServerListLoadBalancer和ZoneAwareLoadBalancer做了一些扩展

我们再回到RibbonLoadBalancer的execute函数逻辑，在通过ZoneAwareLoaderBalancer的chooseServer函数获取了负载均衡策略分配到的服务实例对象server之后，将其内容包装成RibbonServer对象（该对象除了存储了服务实例的信息之外，还增加了服务名serviceId、是否需要使用HTTPS等其他信息），然后使用该对象再回调LoadBalancerInterceptor请求拦截器中LoadBalancerRequest的apply(final ServiceInstance instance)函数，向一个实际的具体服务实例发起请求，从而实现一开始以服务名为host的URI请求，到实际的host:post形式的具体地址的转换。

RibbonServer是ServiceInstance的实现

那么apply(final ServiceInstance instance)函数，在接收到了具体ServiceInstance实例后，是如何通过LoadBalancerClient接口中的reconstructURI操作来组织具体请求地址的呢？

Ribbon是一个远程调用库，客户端负载均衡

参考<https://blog.csdn.net/hry2015/article/details/82693456>

ServerList:定义获取服务器列表

ServerListFilter:对ServerList服务器列表进行二次过滤

ServerListUpdater: 定义服务更新策略

IPing: 检查服务列表是否存活

IRule :根据算法中从服务列表中选取一个要访问的服务

ILoadBalancer:软件负载平衡器入口，整合以上所有的组件实现负载功能

ServerList和ServerListFilter生成客户端可以访问的服务列表

ServerListUpdater和IPing：根据服务的状态更新服务列表

IRule：服务的选择策略

ILoadBalancer：将以上组件组合到这个类中一起工作

不错的文章：

<https://www.jianshu.com/p/795a54ace95b>

ribbon实现的关键点是为ribbon定制的RestTemplate，ribbon利用了RestTemplate的拦截器机制，在拦截器中实现ribbon的负载均衡。负载均衡的基本实现就是利用applicationName从服务注册中心获取可用的服务地址列表，然后通过一定算法负载，决定使用哪一个服务地址来进行http调用。