# Ribbon是什么

Spring Cloud Ribbon是基于Netflix Ribbon实现的一套**客户端负载均衡**的工具。

简单的说，Ribbon是Netflix发布的开源项目，主要功能是提供客户端的软件负载均衡算法，将Netflix的中间层服务连接在一起。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。简单的说，就是在配置文件中列出Load Balancer（简称LB）后面所有的机器，Ribbon会自动的帮助你基于某种规则（如简单轮询，随机连接等）去连接这些机器。我们也很容易使用Ribbon实现自定义的负载均衡算法。

# 什么是负载均衡

LB，即负载均衡(Load Balance)，在微服务或分布式集群中经常用的一种应用。

负载均衡简单的说就是将用户的请求平摊的分配到多个服务上，从而达到系统的HA。

常见的负载均衡有软件Nginx，LVS，硬件 F5等。

相应的在中间件，例如：dubbo和SpringCloud中均给我们提供了负载均衡，**SpringCloud的负载均衡算法可以自定义**。

* **集中式LB**

即在服务的消费方和提供方之间使用独立的LB设施(可以是硬件，如F5, 也可以是软件，如nginx), 由该设施负责把访问请求通过某种策略转发至服务的提供方；

* **进程内LB**

将LB逻辑集成到消费方，消费方从服务注册中心获知有哪些地址可用，然后自己再从这些地址中选择出一个合适的服务器。

Ribbon就属于进程内LB，它只是一个类库，集成于消费方进程，消费方通过它来获取到服务提供方的地址。

# Ribbon初步配置

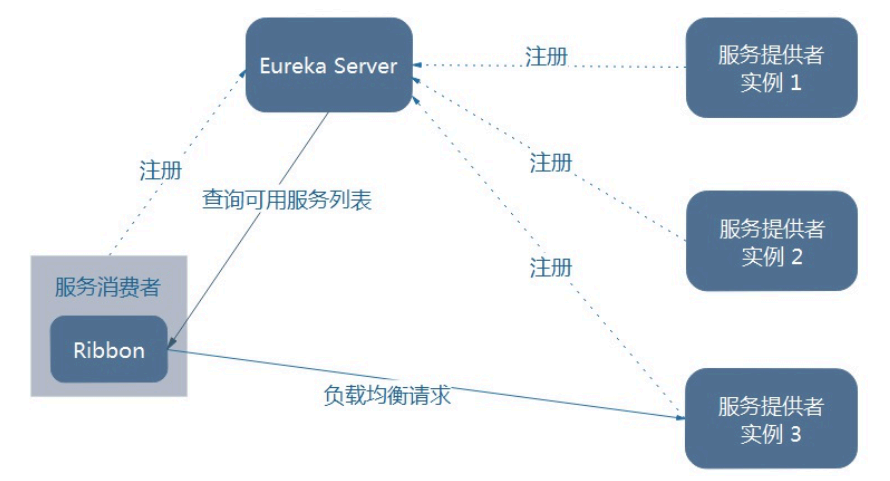
在RestTemplateConfig配置类配置RestTemplate加上Ribbon的注解**@LoadBalanced**

注意：Ribbon和Eureka整合后Consumer可以直接通过服务名调用服务提供者而不用再关心地址和端口号。

为了让大家更加方便的看到通过服务名调用服务提供者我会再建一个测试微服务专门用来调用服务提供者。

# Ribbon负载均衡

1. **Eureka与Ribbon配合使用架构图**



**Ribbon在工作时分成两步**

第一步先选择 EurekaServer ,它优先选择在同一个区域内负载较少的server.

第二步再根据用户指定的策略，在从server取到的服务注册列表中选择一个地址。

其中Ribbon提供了多种策略：比如轮询、随机和根据响应时间加权。

1. **使用不同的配置启动三个品牌微服务实例，并各自完成测试**

为了更加方便大家看到负载均衡的效果，我们三个品牌微服务实例将连接不同的数据库

1. **启动EurekaServer集群**
2. **在用于测试的微服务中调用品牌微服务**

注意观察看到返回的数据库名字，各不相同，负载均衡实现

# Ribbon核心组件IRule

**根据特定算法中从服务器列表中选取一个要访问的服务，Ribbon默认的算法为轮询算法。**

**Ribbon中的7种负载均衡算法：**

* RoundRobinRule：轮询；
* RandomRule：随机；
* AvailabilityFilteringRule：会先过滤掉由于多次访问故障而处于断路器状态的服务，还有并发的连接数量超过阈值的服务，然后对剩余的服务列表按照轮询策略进行访问；
* WeightedResponseTimeRule：根据平均响应时间计算所有服务的权重，响应时间越快的服务权重越大被选中的概率越大。刚启动时如果统计信息不足，则使用RoundRobinRule（轮询）策略，等统计信息足够，会切换到WeightedResponseTimeRule；
* RetryRule：先按照RoundRobinRule（轮询）策略获取服务，如果获取服务失败则在指定时间内进行重试，获取可用的服务；
* BestAvailableRule：会先过滤掉由于多次访问故障而处于断路器跳闸状态的服务，然后选择一个并发量最小的服务；
* ZoneAvoidanceRule：复合判断Server所在区域的性能和Server的可用性选择服务器；

# 配置指定的负载均衡算法

# Ribbon自定义负载均衡算法

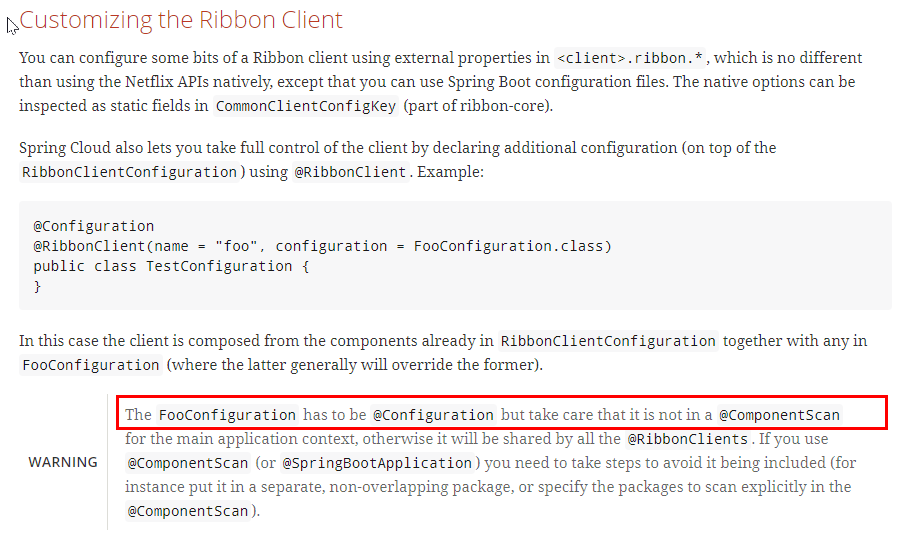
1. **Ribbon配置类需要注意的细节**

官方文档明确给出了警告：

这个自定义配置类不能放在@ComponentScan所扫描的当前包下以及子包下，

否则我们自定义的这个配置类就会被所有的Ribbon客户端所共享，也就是说

我们达不到特殊化定制的目的了。



1. **在源码包下新建一个目录，然后在该目录下新建一个Robbin配置类**
2. **自定义规则深度解析**

* 问题：

依旧轮询策略，但是加上新需求，每个服务器要求被调用5次。也即以前是每台机器一次，现在是每台机器5次

* 解析源码：

<https://github.com/Netflix/ribbon/blob/master/ribbon-loadbalancer/src/main/java/com/netflix/loadbalancer/RandomRule.java>

* 参考源码新建一个符合我们需求的规则类

1. **在Robbin配置类中定义咱们新建的规则类**
2. **在启动该微服务的时候就能去加载我们的自定义Ribbon配置类，从而使配置生效,形如：**

@RibbonClient(name="服务名",configuration=RibbonConfig.class)

1. **进行测试**