# Spring Cloud客户端负载均衡Ribbon笔记

## 负载均衡

**服务端负载均衡**：我在工作中常用就是Nginx+Tomcat集群。由Nginx负责接收请求，按照设置的算法（轮询，加权轮询，随机，加权随机，源地址哈希，最小连接数）将请求转发至后端。

也要有一个心跳监测模块，及时去掉不可用的服务节点，用来维护一个可用的服务节点清单。

大型公司一般都用F5硬件负载均衡。

**客户端负载均衡**：如果将可用服务节点清单挪到Client端，就是客户端负载均衡了。在Spring Cloud中，结合Eureka，客户端从中拿到服务节点，按照算法取出一个地址，进行调用。

## RestTemplate

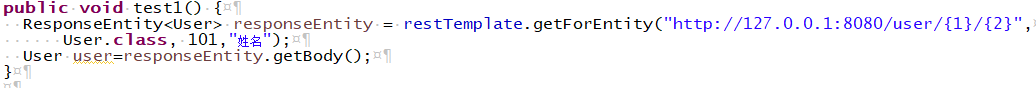
在Spring Cloud全家桶中，各个微服务的接口都是HTTP的形式，所以RestTemplate就是一个绕不过去的类，摘抄一些个人觉得的重点。

### GET

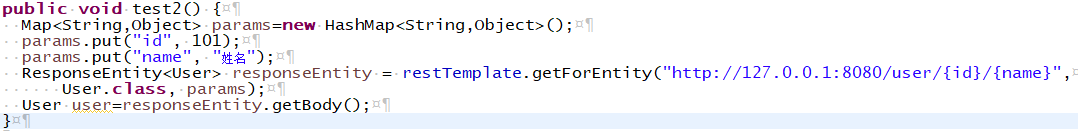
#### getForEntity

返回的是ResponseEntity，是Spring对Http响应的封装，包含了HTTP请求状态码的枚举对象（404,200此类），Http请求头对象，请求体对象。

重载实现有三种，这里只说两种。



参数（101、姓名）与前面的数字（1,2）顺序对应。



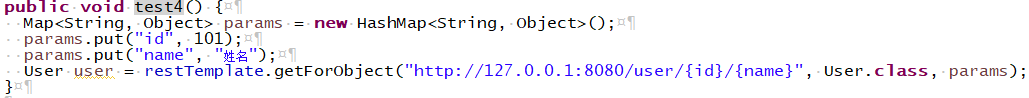
参数Map的Key，与前面的占位符（id、name）对应。

#### getForObject

与getForEntity相比，getForObject直接返回接收内容的对象，省略了responseEntity.getBody()这一步。

如果我们的不需要关注Http响应信息的时候，可以用此方法。

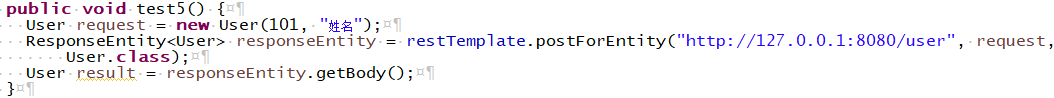


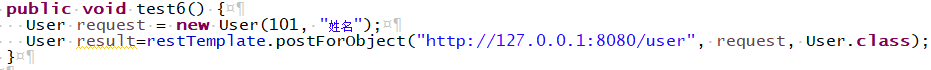


与getForEntity基本相同，不再赘述。

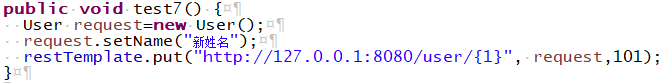
### POST

与GET基本类似。





### PUT



put是没有返回值的。

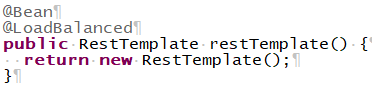
### DELETE



delete没有返回值。

## Ribbon负载均衡机制

Ribbon实现了客户端的负载均衡，调用服务使用的是用**@LoadBalanced**的RestTemplate：



通过对其跟踪，定位到**LoadBalancerClient**及其父类**ServiceInstanceChooser**，方法有如下：

reconstructURI：将<http://服务实例名/path>转成<http://HOST:PORT/path/>

choose：从负载均衡器中选一个服务实例。

execute：用挑出来的服务实例执行请求。

继续搜索，有**LoadBalancerAutoConfiguration**与**LoadBalancerInterceptor**两个类。主要功能：

1. 由**LoadBalancerAutoConfiguration**生成**LoadBalancerInterceptor**，用于发起请求时拦截。
2. 维护**RestTemplate**列表。
3. 生成**RestTemplateCustomizer**，给**RestTemplate**增加**LoadBalancerInterceptor**拦截器。

以上可得，我们在发起请求的时候，会被**LoadBalancerInterceptor**的intercept()方法拦截。

主要工作是通过服务实例名，交给**LoadBalancerClient**的execute()去执行。

具体实现类是**RibbonLoadBalancerClient**，其主要工作是：

1. 根据服务实例名获取负载均衡器ILoadBalancer。
2. 根据ILoadBalancer获取Server，生成具体的服务实例RibbonServer。
3. 将Request转到服务实例去执行。

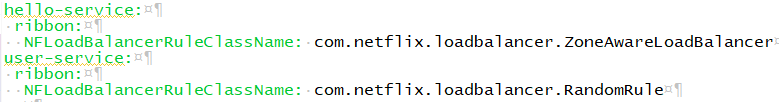
既然是负载均衡，那么最重要的就是根据负载策略去选择具体的服务了，也即ILoadBalancer的chooseServer()。其具体的实现类是**DynamicServerListLoadBalancer**，DSLB继承了**BaseLoadBalancer**，BLB继承了**AbstractLoadBalancer**，ALB实现了**ILoadBalancer**。

（就不画UML图了）。

而具体的chooseServer()的实现是在**BaseLoadBalancer**中，而它又是交给**IRule**的choose()去实现的。

默认的实现类是**RoundRobinRule**，轮询。

在实际应用中，针对不同的服务，我们可能需要执行不同的负载策略，Ribbon内置了7种（网上很多，本文不再列出），可在配置文件中进行配置，如图：



以上就是给hello-service这个服务配置了ZoneAwareLoadBalancer，给user-service配置了RandomRule。其他没有配的，使用默认的RoundRobinRule。

以上是Ribbon结合RestTemplate进行客户端负载均衡HTTP请求的流程主干，此外还有很多方法，比如结合Eureka从注册中心获取服务列表，对服务列表的维护。通过IPing的实现以10秒（默认）的频率去检查服务列表是否有更新等，不再赘述。