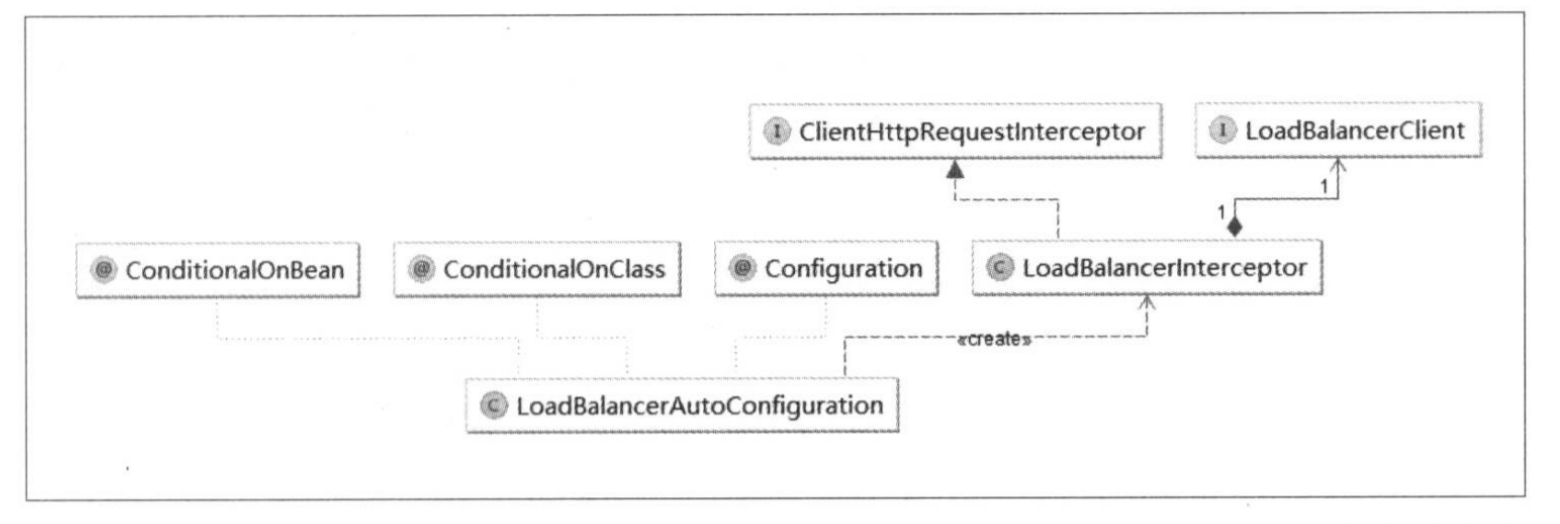
## 源码分析

启用负载均衡是通过@LoadBalanced配置的，然后就会通过LoadBalancerClient配置

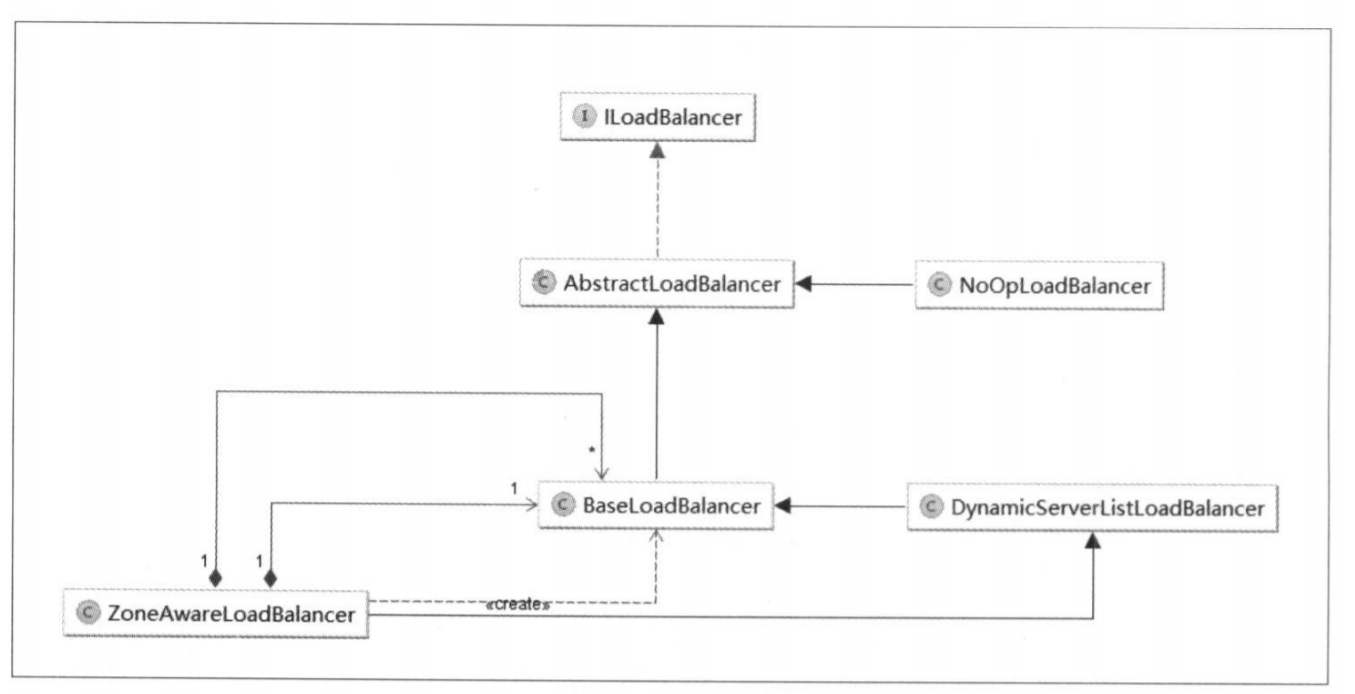
RestTemplate



LoadBalancerAutoConfiguration 里通过给RestTemplate列表的每一个RestTemplate都增加一个LoadBalancerInterceptor拦截器

LoadBalancerInterceptor的作用就是在发起请求时，拦截，然后去调用里面的LoadBalancerClient里的execute方法去进行负载均衡请求

在里面先通过里面的一个SpringClientFactory去获取一个ILoadBalancer，然后通过它根据服务去选择实例，它的实现其实是用IRule去选择的



Spring默认使用的是采用这个具体实现

ZoneAwareLoadBalancer

选择完实例后，将内容包装成RibbonServer

然后回调LoadBalancerRequest的apply函数传入一个服务实例信息

然后构造一个ServiceRequestWrapper通过重写里面的getURI方法，调用LoadBalancerClient

的reconstructURI来组织具体的请求服务地址

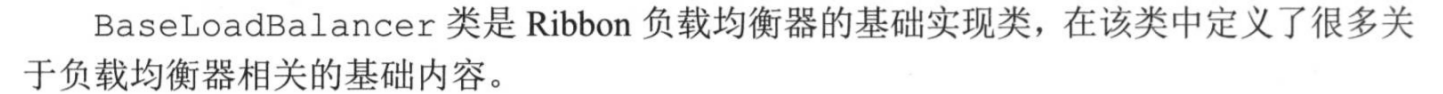
reconstructURI这个方法里，先用SpringClientFactory根据服务去获得RibbonLoadBalancerContext

然后根据RibbonUtils根据服务生成URI

然后调用RibbonLoadBalancerContext的reconstructURIWithServer

去构建服务实例的URI，并且返回，然后拦截器的作用就完事了，就拿着这个经过一定算法选择的实例去正常的调用请求

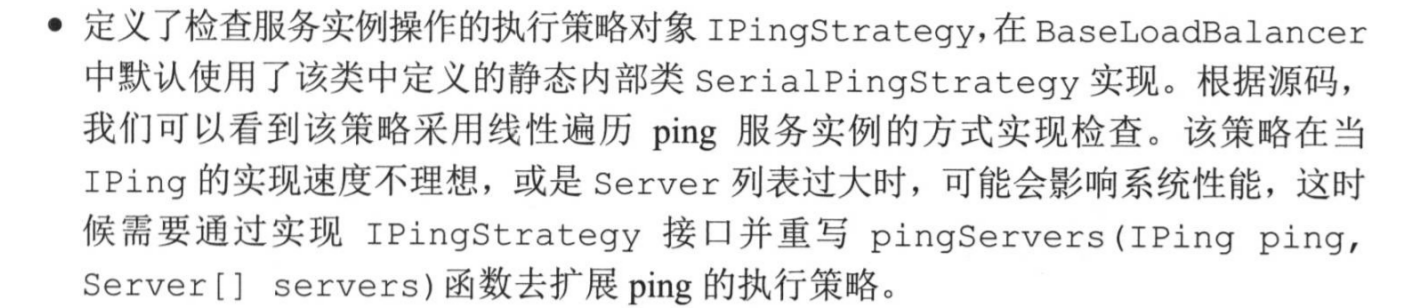
## 负载均衡器

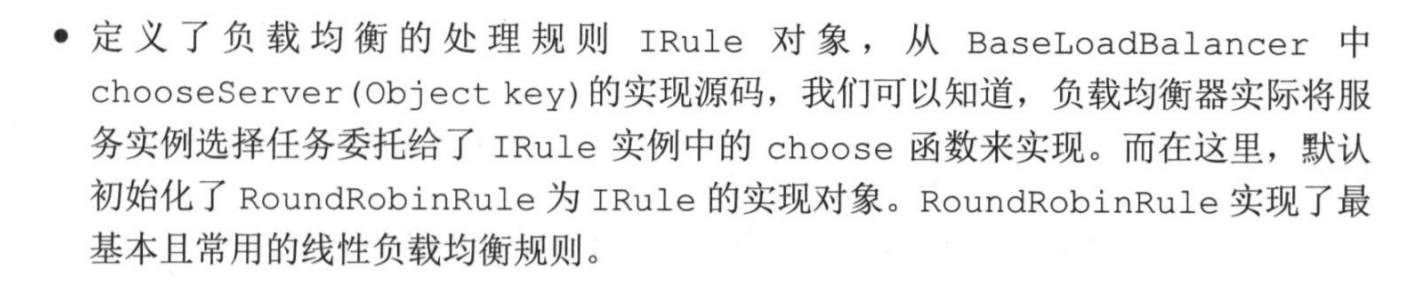


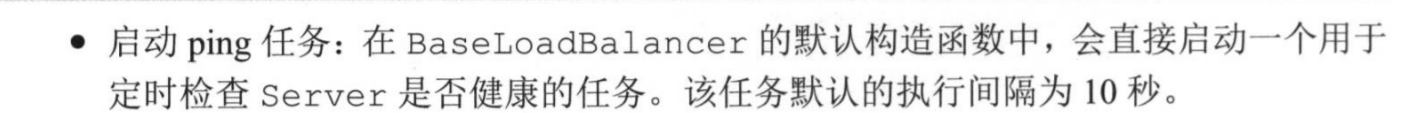
里面存储了两个Server对象的列表，一个是所有的，一个是正常的

存储了LoadBalancerStats对象，描述服务属性和统计信息







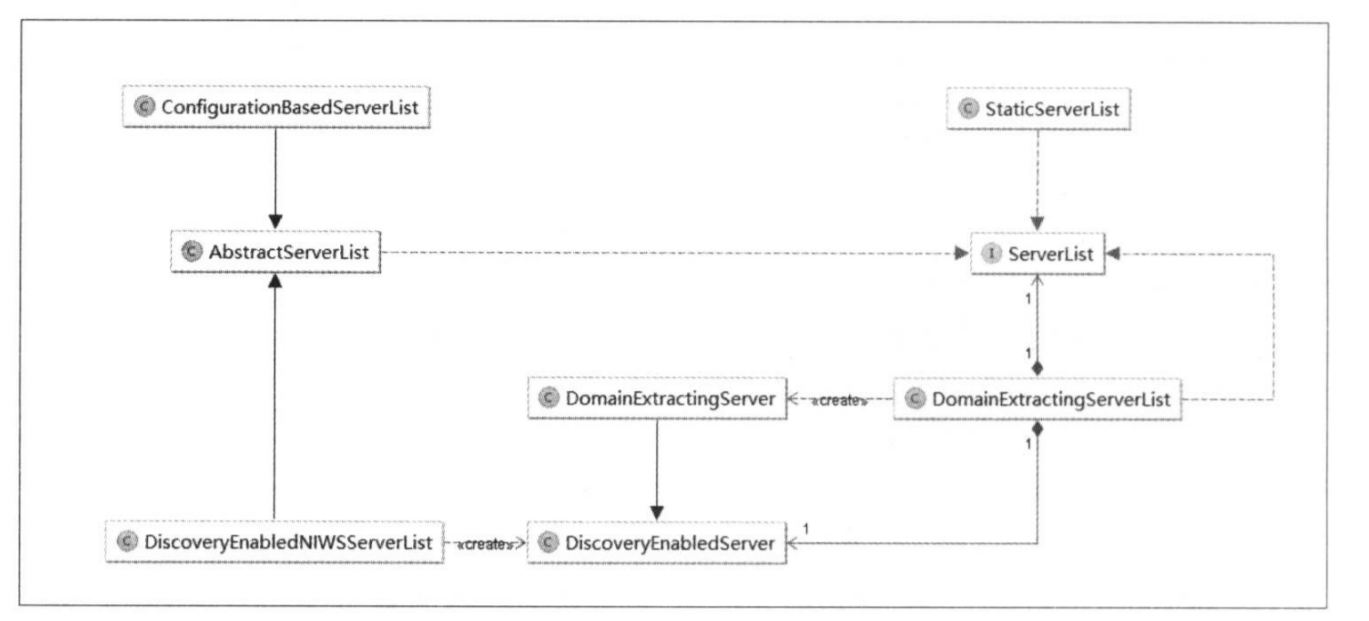


DynamicServerListLoadBalancer是BaseLoadBalancer类的扩展

可以在运行期动态更新服务清单，同时还可以通过过滤条件去获得服务清单

里面有一个ServerList接口

有两个方法，一个获取初始化的实例列表，一个获取更新的列表



默认是DomainExtractingServerList，但是它实现了这个接口，里面还有一个这个接口

类似于装饰模式

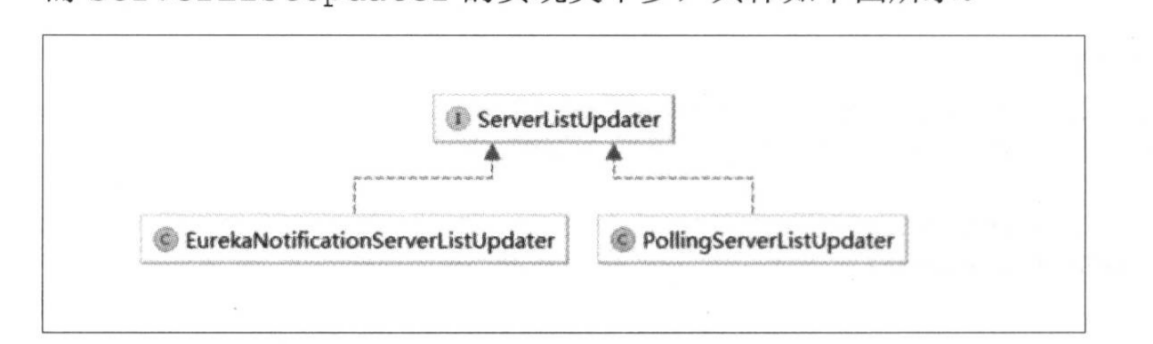
构造方法里传入的是DiscoveryEnabledNiWSServerList

所以实际上是DiscoveryEnabledNiWSServerList

主要就是通过EurekaClient从服务注册中心获取到的具体的服务实例

然后更新列表是通过ServerListUpdater和UpdateAction实现的

UpdateAction接口在ServerListUpdater接口里



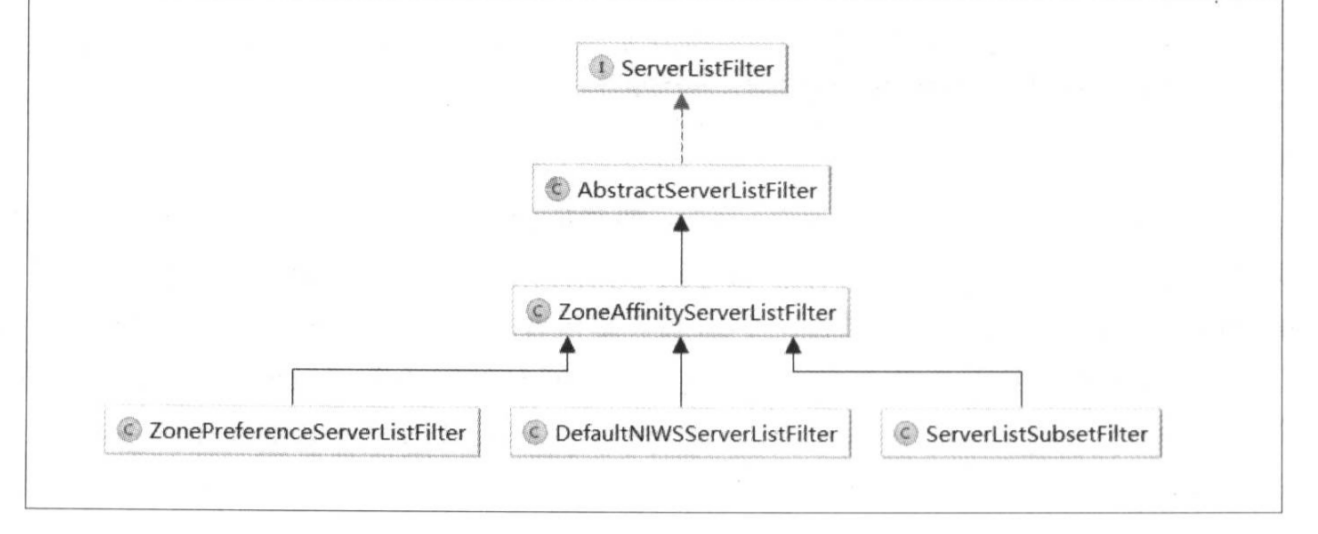
PollingServerListUpdater默认策略，通过定时任务的方式进行服务列表的更新

另一个需要事件监听器来更新

而执行的任务就是UpdateAction接口的更新方法，而里面的具体操作则委托给了DynamicServerListLoadBalancer的updateListOfServers方法

里面先DomainExtractingServerList去从服务注册中心获取到更新之后的列表，然后

通过过滤器的规则过滤掉一些服务，然后返回更新后的列表

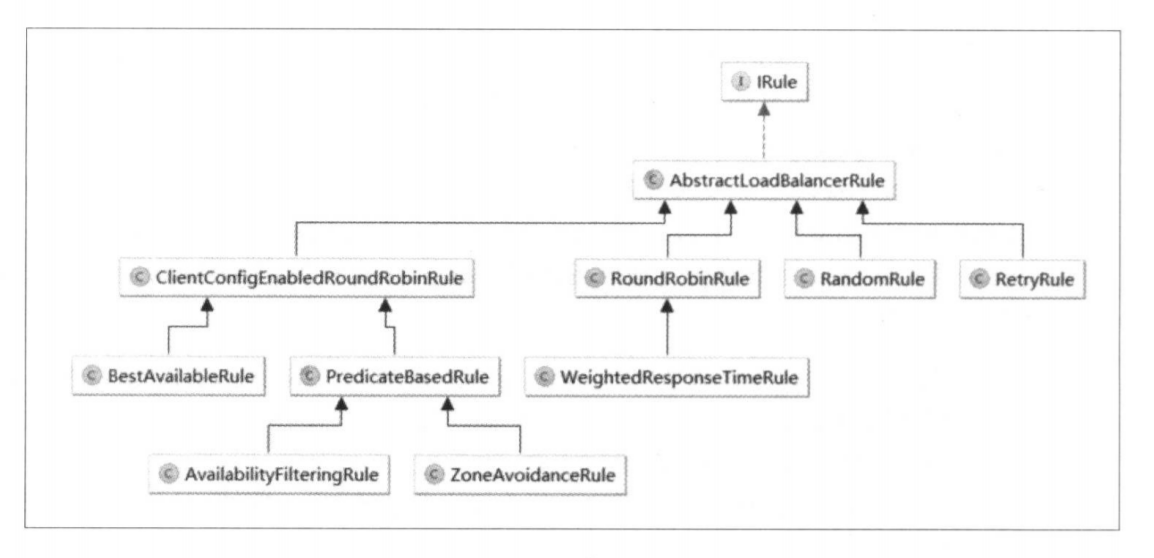


过滤器有很多种类型，不明白的可以去百度查，来选择使用哪个过滤器

ZoneAeareLoadBalancer是DynamicServerListLoadBalancer的扩展

它是有区域（Zone）概念的

## 负载均衡策略



RandomRule：

随机选择一个

RoundRobinRule:

线性轮询选择一个

RetryRule:

根据一定时间去重试，默认线性轮询，如果超过了重试时间返回null

WeightedResponseTimeRule:

根据权重去选择实例

权重如何计算自己查百度吧

ClientConfigEnabledRoundRobinRule这个默认就是RoundRobinRule

就是为了继承它，然后实现一些更高级的策略，如果策略无法实施，则会调用父类的RoundRobinRule

BestAvailableRule继承了ClientConfigEnabledRoundRobinRule

作用是选出最空闲的实例

依赖的是loadBalancerStats,当它为空的时候，这个策略是无法实施的，会采用父类的线性轮询

PredicateBasedRule继承了ClientConfigEnabledRoundRobinRule

先过滤清单，再轮询选择，对于怎样过滤，需要我们在AbstractServerPredicate子类中实现apply方法确定具体的过滤策略

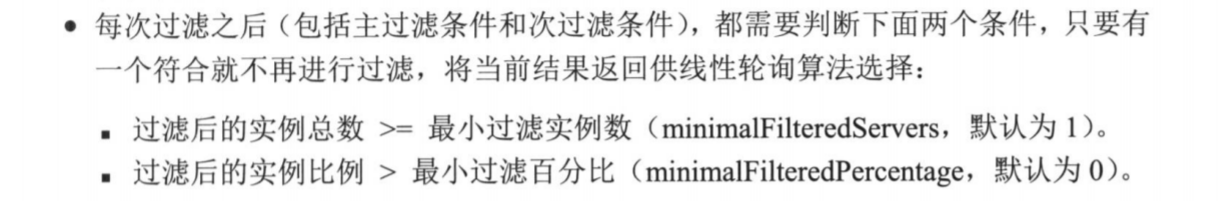
AvailabilityFilteringRule继承PredicateBasedRule

它里面就是实现了AbstractServerPredicate

并且也重写了choose，已改进优化

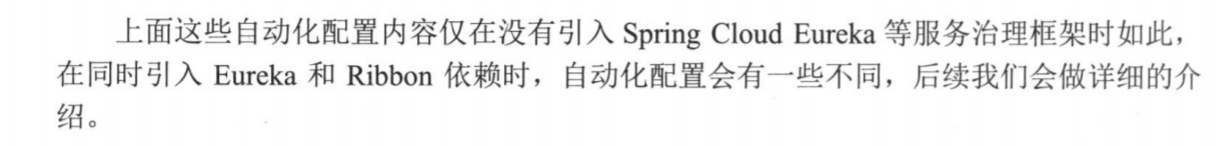
ZoneAvoidanceRule继承PredicateBasedRule

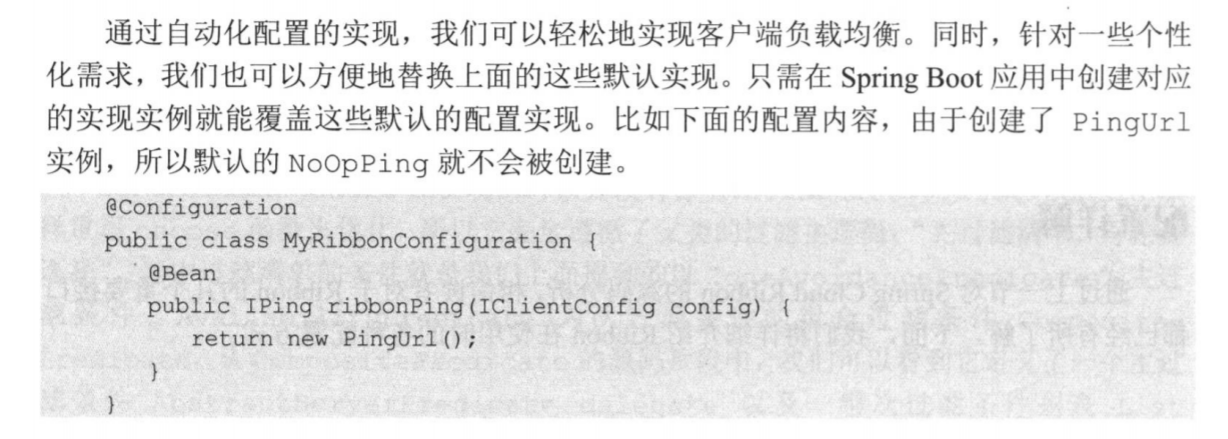
它是分主过滤条件和次过滤条件列表



## 配置详解











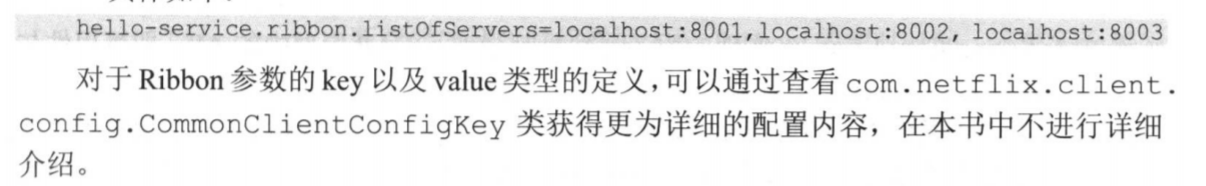


## 参数配置

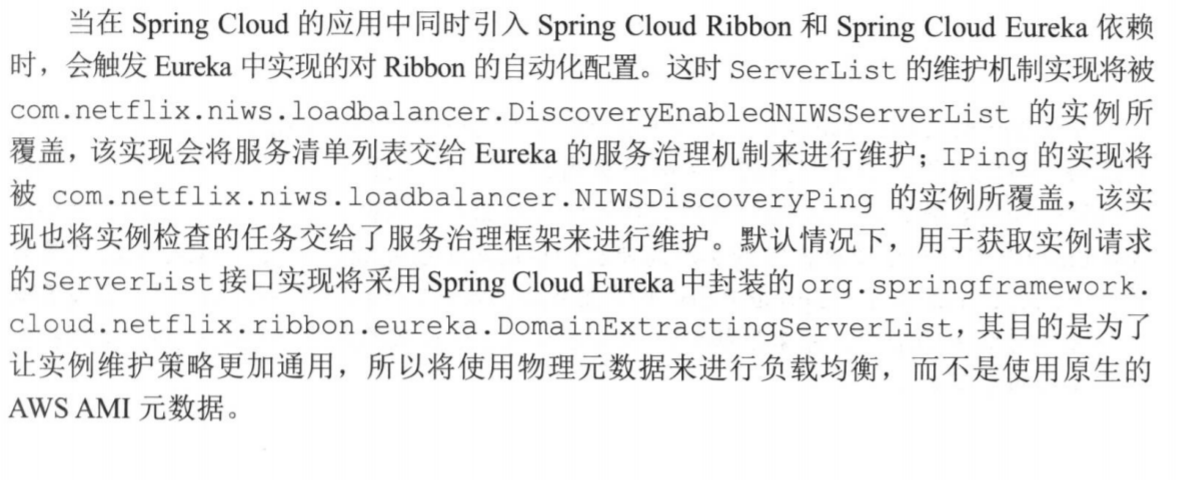
全局配置：

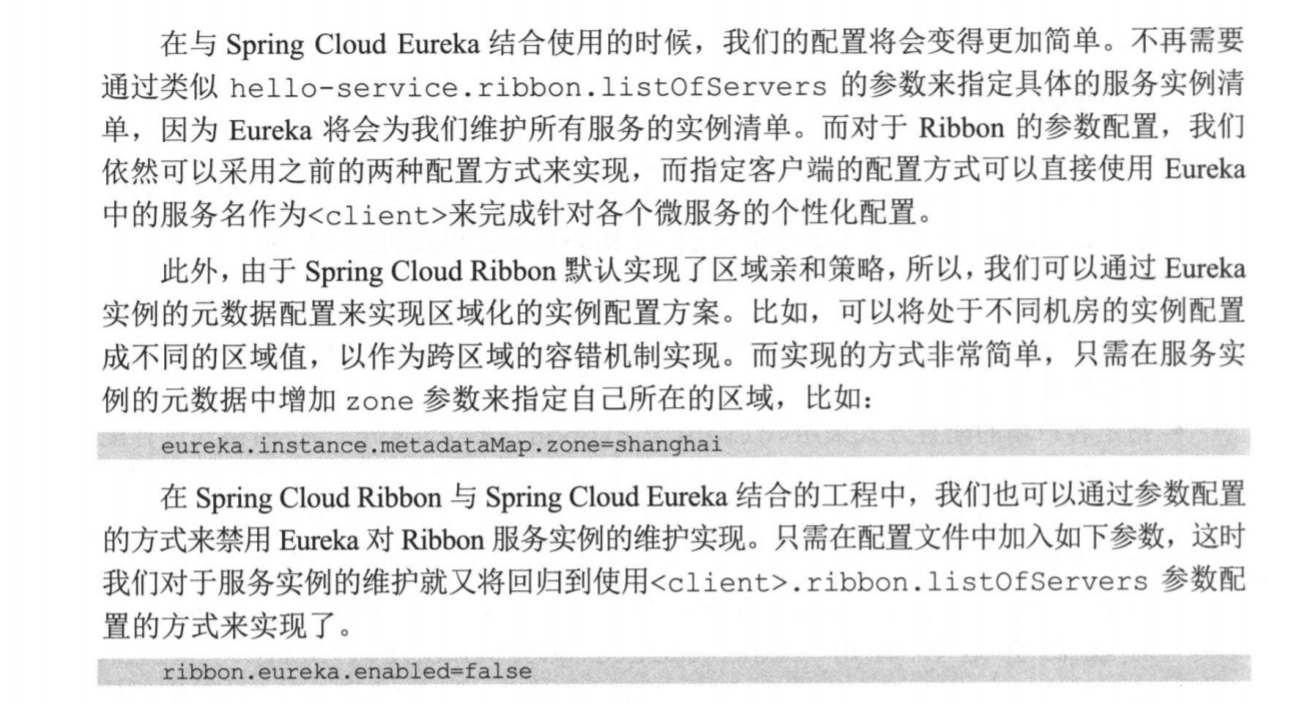


在没有引入Eureka的帮助下



## 与Eureka结合





## 重试机制：

