**Ribbon：Spring Cloud负载均衡与服务调用组件**

提供客户端的负载均衡算法和服务调用，是一个基于HTTP和TCP的客户端负载均衡器。当ribbon和eureka一起使用时，ribbon会从eureka server 中获取可用服务端列表，然后通过负载均衡策略将请求分摊给多个服务提供者，从而达到负载目的。

负载均衡策略：

Spring Cloud Ribbon提供了一个IRule接口定义负载均衡策略。默认有7种实现类。



制定自定义Ribbon负载均衡策略，创建XXXRule类继承AbstractLoadBalancerRule或实现IRule接口，重写choose方法。再创建配置类，将XXXRule实现类注册到容器中。最后再修改启动类让自定义负载生效。

官方文档给出警告：这个自定义的类不能放在@ComponentScan所扫描的当前包以及子包下，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，也就是我们达不到特殊化指定的目的了。





六大功能核心组件：

LoadBalancer（负载均衡器）：用于管理负载均衡，初始化的时候通过加载ymal文件创建的。

ServerList（服务列表）：获取所有服务地址，并保存到本地。

静态获取：从配置文件中获取服务节点列表并存储到本地

动态获取：从注册中心里获取服务节点列表并存储到本地

Rule（负载均衡策略）：七种策略。

Ping（心跳检测）：IPing接口类来检测哪些服务可用或不可用。默认轮询检测。

ServerListFilter（服务列表过滤）：获取到的服务列表按照规则进行过滤。通过Eureka的分区规则对服务实例进行过滤。比较服务实例的通信失败和并发连接数来剔除不够健康的实例。

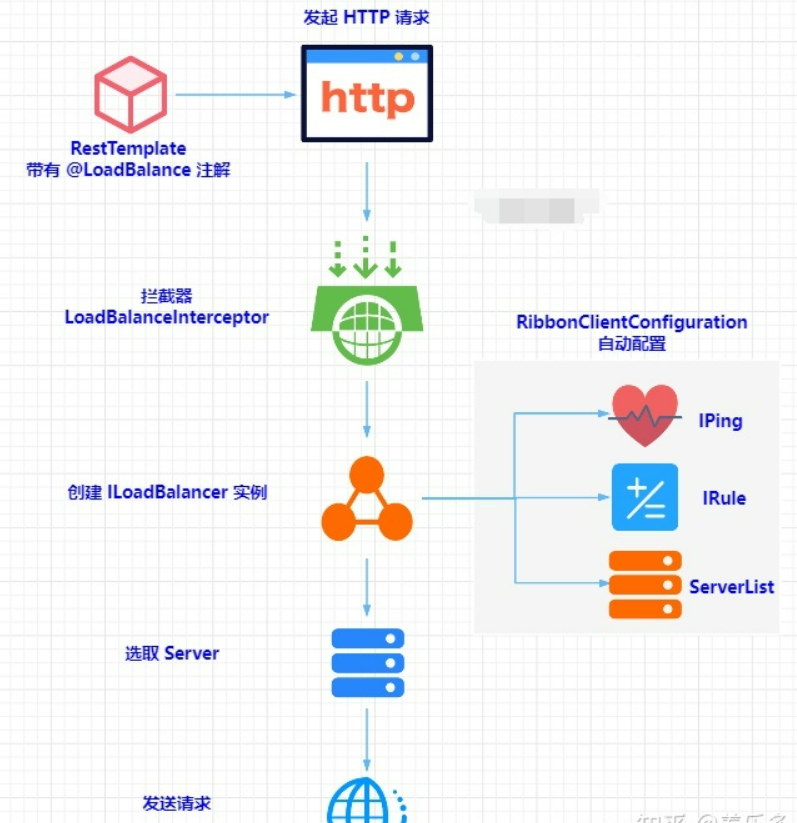
ServerListUpdater（服务列表更新）：Ribbon会从注册中心获取最新的注册信息，这是由ServerListUpdater定义的更新操作。

有两种更新方式：

·定时任务进行更新，通过实现类PollingServerListUpdater实现

·利用Eureka的事件监听器来更新，由实现类EurekaNotificationServerListUpdater 完成。

Ribbon的拦截处理：LoadBalancerInterceptor



第一步：Ribbon 拦截所有标注 @loadBalance 注解的 RestTemplate。RestTemplate 是用来发送 HTTP 请求的。第二步：将 Ribbon 默认的拦截器 LoadBalancerInterceptor 添加到 RestTemplate 的执行逻辑中，当 RestTemplate 每次发送 HTTP 请求时，都会被 Ribbon 拦截。第三步：拦截后，Ribbon 会创建一个 ILoadBalancer 实例。第四步：ILoadBalancer 实例会使用 RibbonClientConfiguration 完成自动配置。就会配置好 IRule，IPing，ServerList。第五步：Ribbon 会从服务列表中选择一个服务，将请求转发给这个服务。

**Ribbon的初始化原理**

第一步：Ribbon 有一个自动配置类 LoadBalancerAutoConfiguration，SpringBoot 加载自动配置类，就会去初始化 Ribbon。第二步：当我们给 RestTemplate 或者 AsyncRestTemplate 添加注解后，Ribbon 初始化时会收集加了 @LoadBalanced 注解的 RestTemplate 和 AsyncRestTemplate ，把它们放到一个 List 里面。第三步：然后 Ribbon 里面的 RestTemplateCustomizer 会给每个 RestTemplate 进行定制化，也就是加上了拦截器：LoadBalancerInterceptor。第四步：从 Eureka 注册中心获取服务列表，然后存到 Ribbon 中。第五步：加载 YMAL 配置文件，配置好负载均衡配置，创建一个 ILoadbalancer 实例

**Ribbon的心跳检测：**对自己本地的ServerList进行遍历，看每个服务的状态是不是Up。



默认间隔30s执行一下PingTask调度任务，对每个服务进行isAlive方法，判断状态。