Spring Cloud Netflix Ribbon深度解析

Ribbon简介

```
@Bean
@LoadBalanced
public RestTemplate restTemplate() { return new RestTemplate(); }
```

Ribbon简介

Ribbon如何实现客户端负载

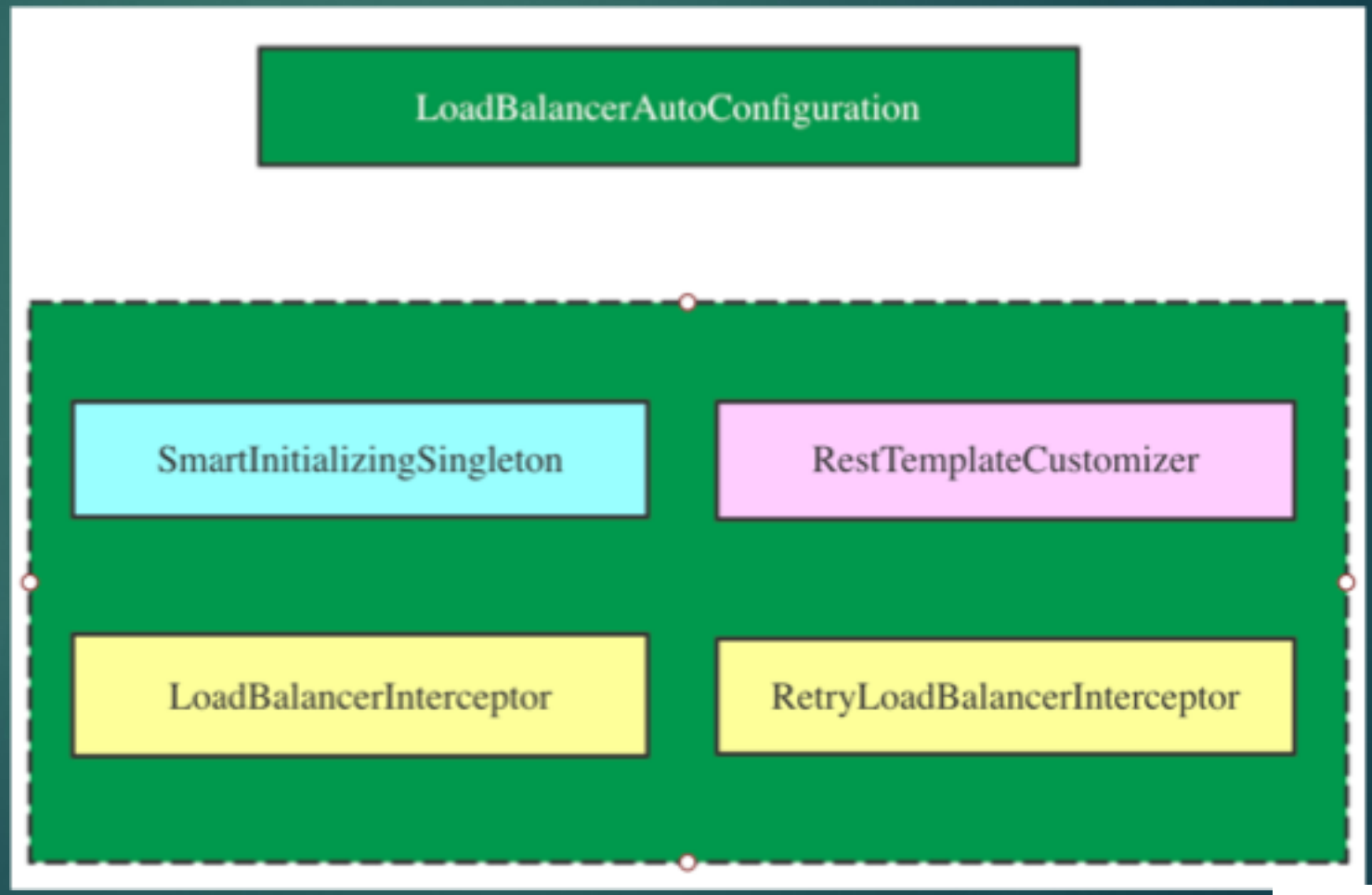
- Ribbon相关配置的初始化: META-INF/spring.factories

```
# AutoConfiguration
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
org.springframework.cloud.client.CommonsClientAutoConfiguration,\
org.springframework.cloud.client.discovery.composite.CompositeDiscoveryClientAutoConfiguration,\
org.springframework.cloud.client.discovery.noop.NoopDiscoveryClientAutoConfiguration,\
org.springframework.cloud.client.discovery.simple.SimpleDiscoveryClientAutoConfiguration,\
org.springframework.cloud.client.hypermedia.CloudHypermediaAutoConfiguration,\
org.springframework.cloud.client.loadbalancer.AsyncLoadBalancerAutoConfiguration,\
org.springframework.cloud.client.loadbalancer.LoadBalancerAutoConfiguration,\
org.springframework.cloud.client.loadbalancer.reactive.ReactiveLoadBalancerAutoConfiguration,\
org.springframework.cloud.client.serviceregistry.ServiceRegistryAutoConfiguration,\
org.springframework.cloud.commons.httpclient.HttpClientConfiguration,\
org.springframework.cloud.commons.util.UtilAutoConfiguration,\
org.springframework.cloud.client.serviceregistry.AutoServiceRegistrationAutoConfiguration


# Environment Post Processors
org.springframework.boot.env.EnvironmentPostProcessor=\
org.springframework.cloud.client.HostInfoEnvironmentPostProcessor
```

Ribbon如何实现客户端负载

- Ribbon自动配置文件



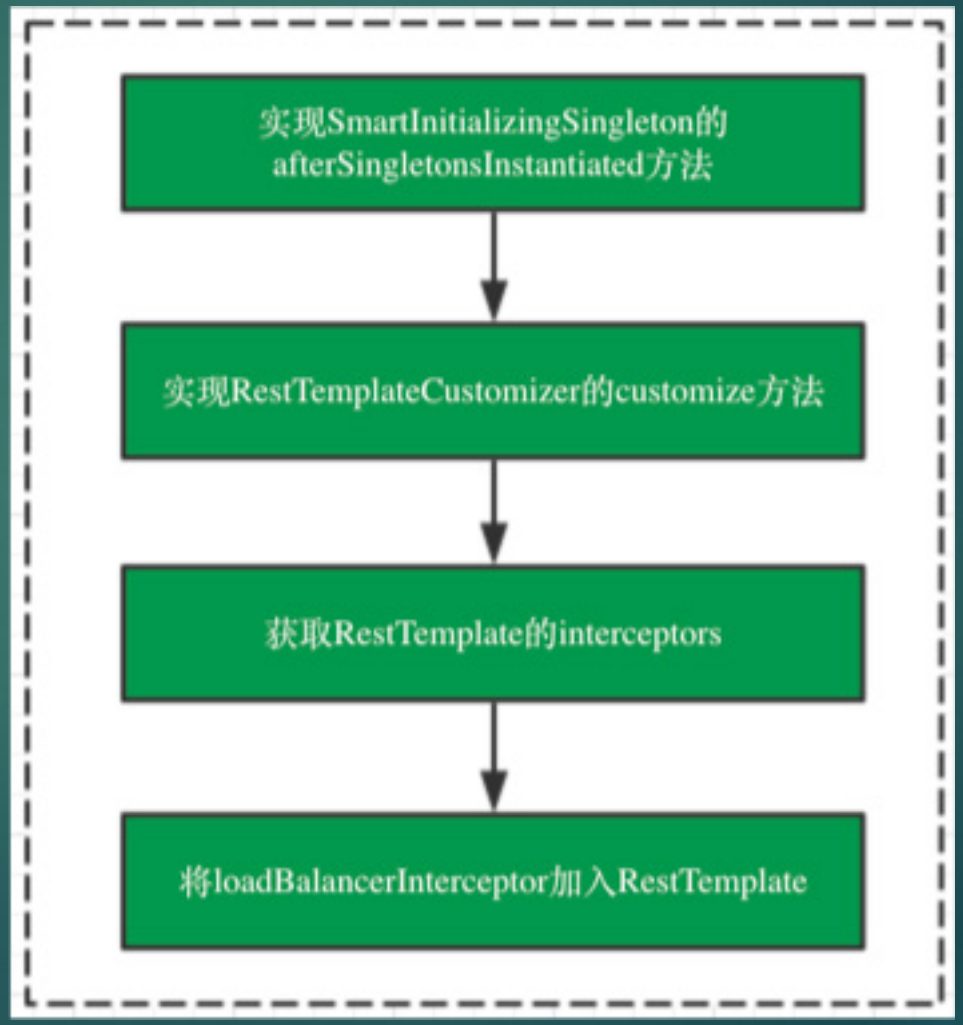
Ribbon如何实现客户端负载

- RestTemplate的收集

```
@LoadBalanced
@Autowired(required = false)
private List<RestTemplate> restTemplates = Collections.emptyList();
```

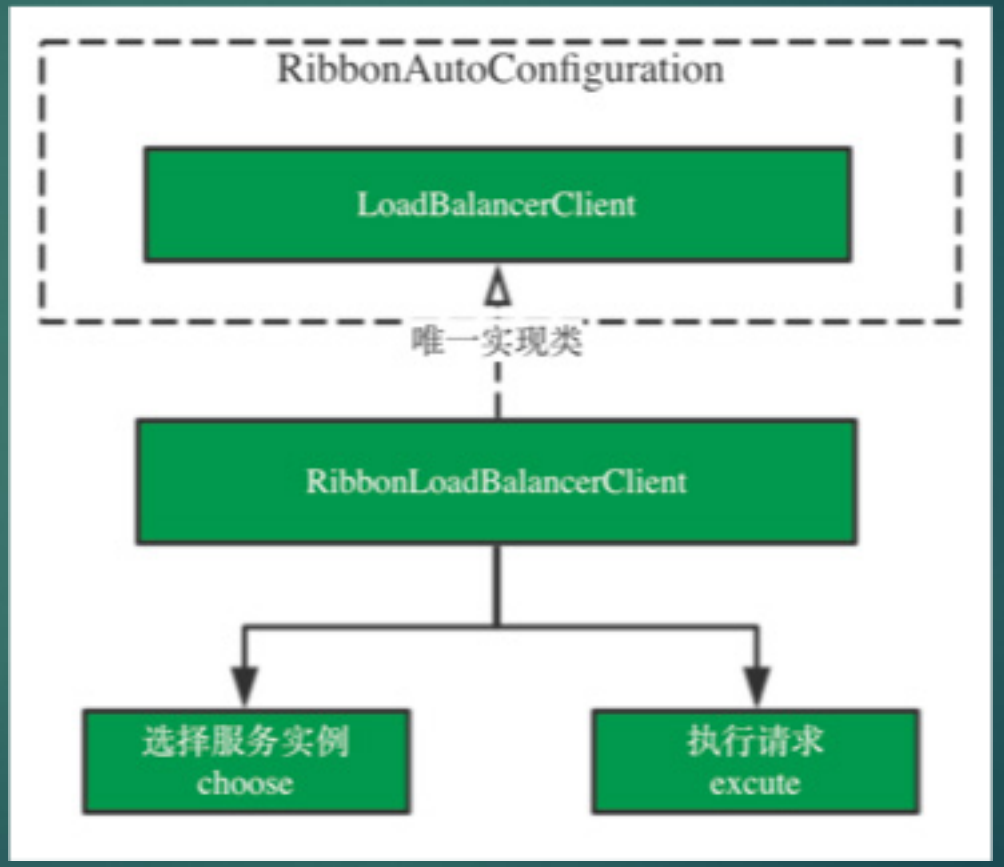
Ribbon如何实现客户端负载

- 向RestTemplate中加入Ribbon拦截器
- LoadBalancerInterceptor还是RetryLoadBalancerInterceptor



Ribbon如何实现客户端负载

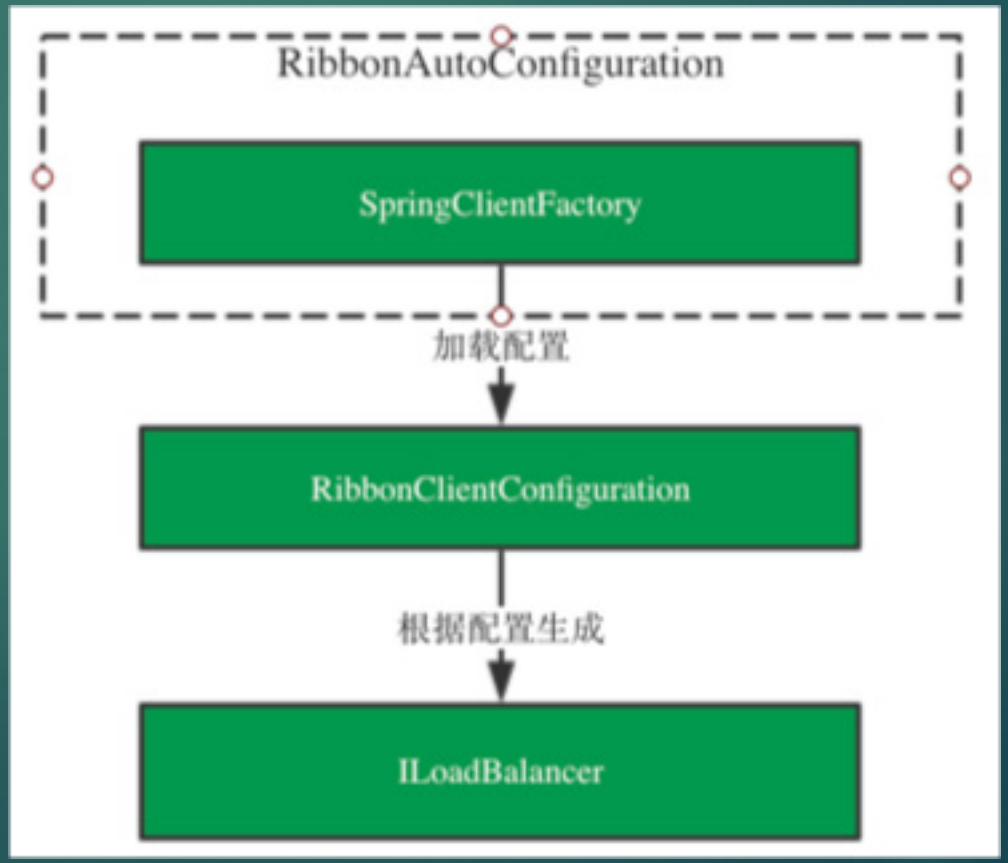
- 在构造LoadBalancerInterceptor拦截器时需要传入LoadBalancerClient。它是真正完成客户端负载均衡的接口。



Ribbon负载均衡行为定义

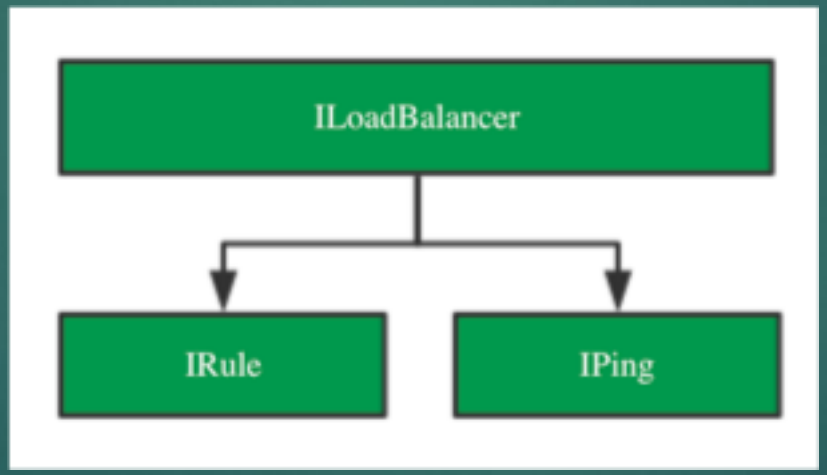
```
public RibbonLoadBalancerClient(SpringClientFactory clientFactory) {
    this.clientFactory = clientFactory;
}
```

- LoadBalancerClient的构造器以SpringClientFactory为入参
- 通过RibbonClientConfiguration获取我们在yml配置文件中的配置,来构造SpringClientFactory并生成ILoadBalancer



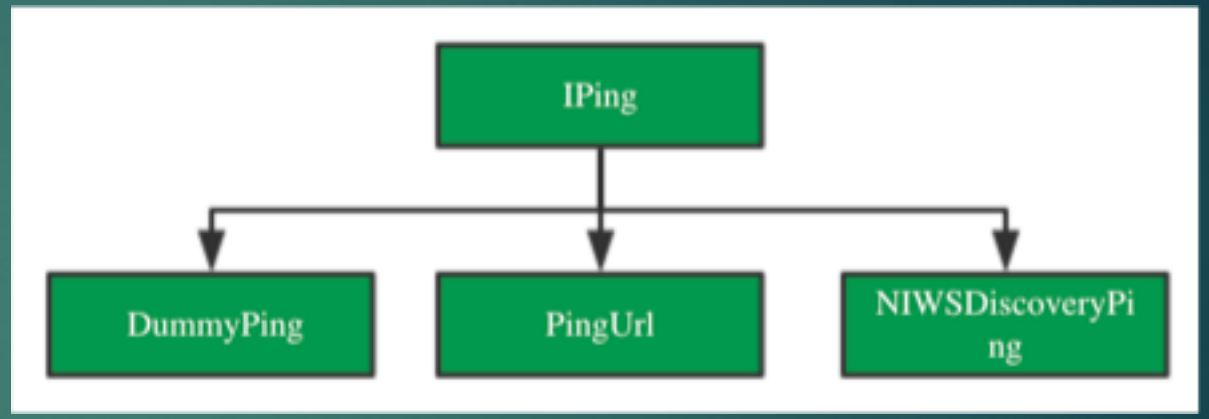
Ribbon负载均衡行为定义

- ILoadBalancer——定义客户端负载行为。
- LoadBalancerClient通过ILoadBalancer来选择服务实例

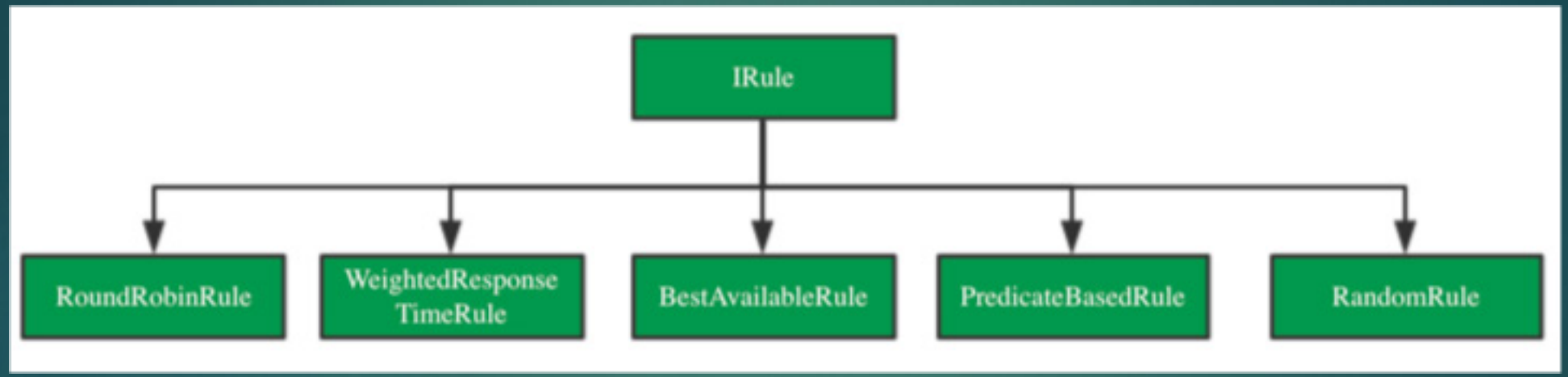


Ribbon负载均衡行为定义-IPing

- DummyPing:直接返回true
- PingUrl：realPing
- NIWSDiscoveryPing：从discovery比如eureka里面获取实例的状态，不是真的ping



Ribbon负载均衡行为定义-IRule

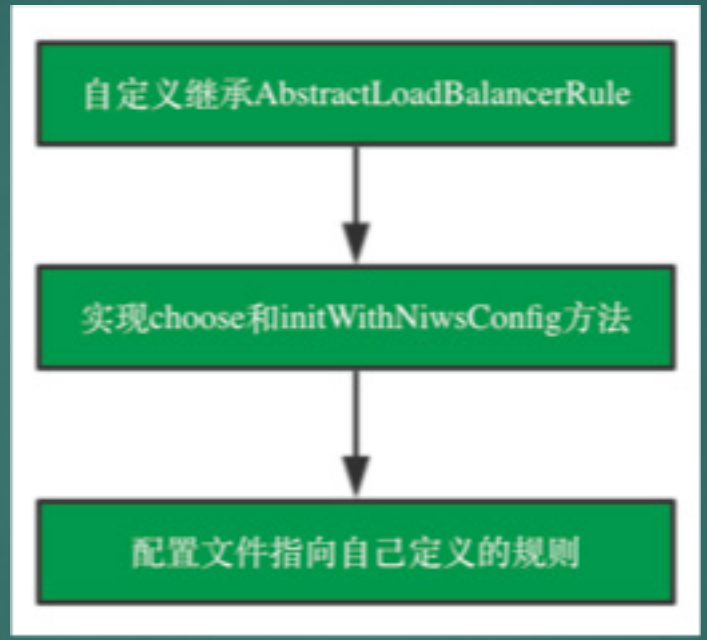


Ribbon的负载均衡策略-加权轮训

服务实例	A	B	C	D
响应时间(Rt)	1s	2s	2s	5s
权重Wt(totle-Rt)	9	8	8	5
权重累计Wtsofar	9	17	25	30
区间	0~9	9~17	17~25	25~30

Ribbon的负载均衡策略-自定义策略

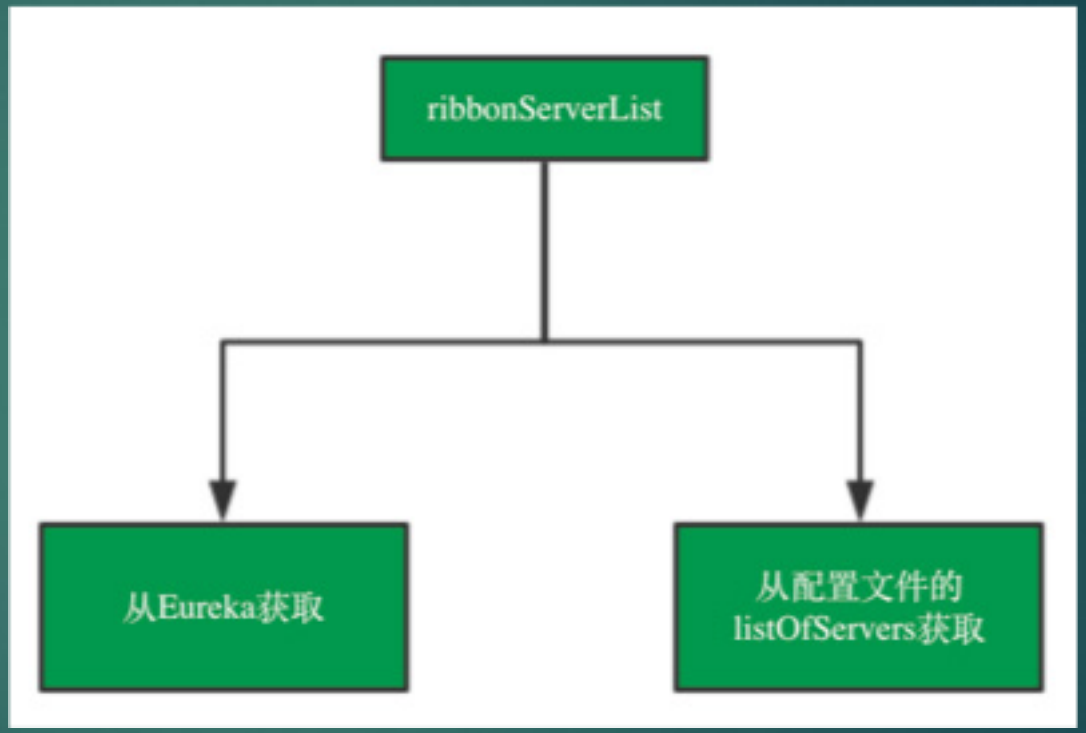
- 我们自定义的类是可以通过父类拿到服务列表的
- 然后可以根据自己的需求实现负载均衡算法




```
provider-user:
  ribbon:
    NFLoadBalancerRuleClassName: com.**.CustomRule
```

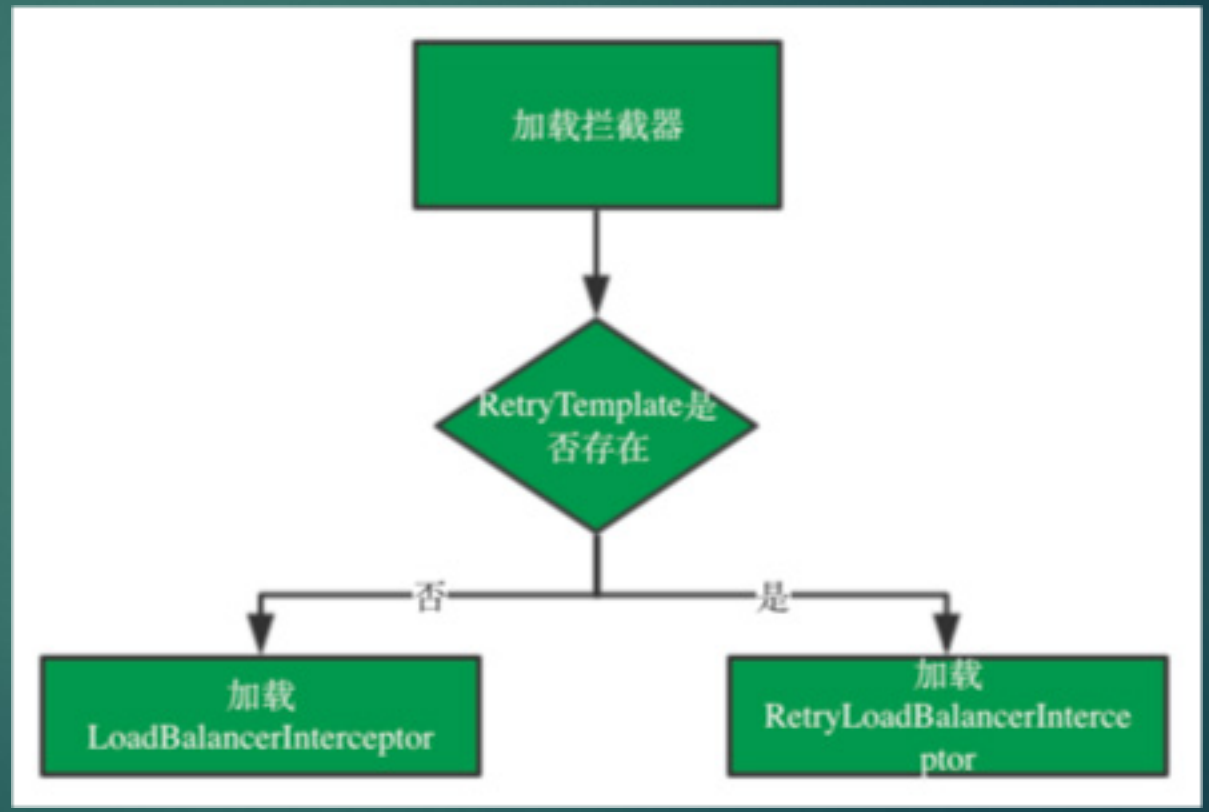
Ribbon服务列表的获取

- 服务列表的获取
- @ConditionalOnRibbonAndEurekaEnabled 如果Eureka和Ribbon都是enabled则通过Eureka获取服务列表
- 否则会直接从我们yml配置文件的listOfServers中获取服务列表



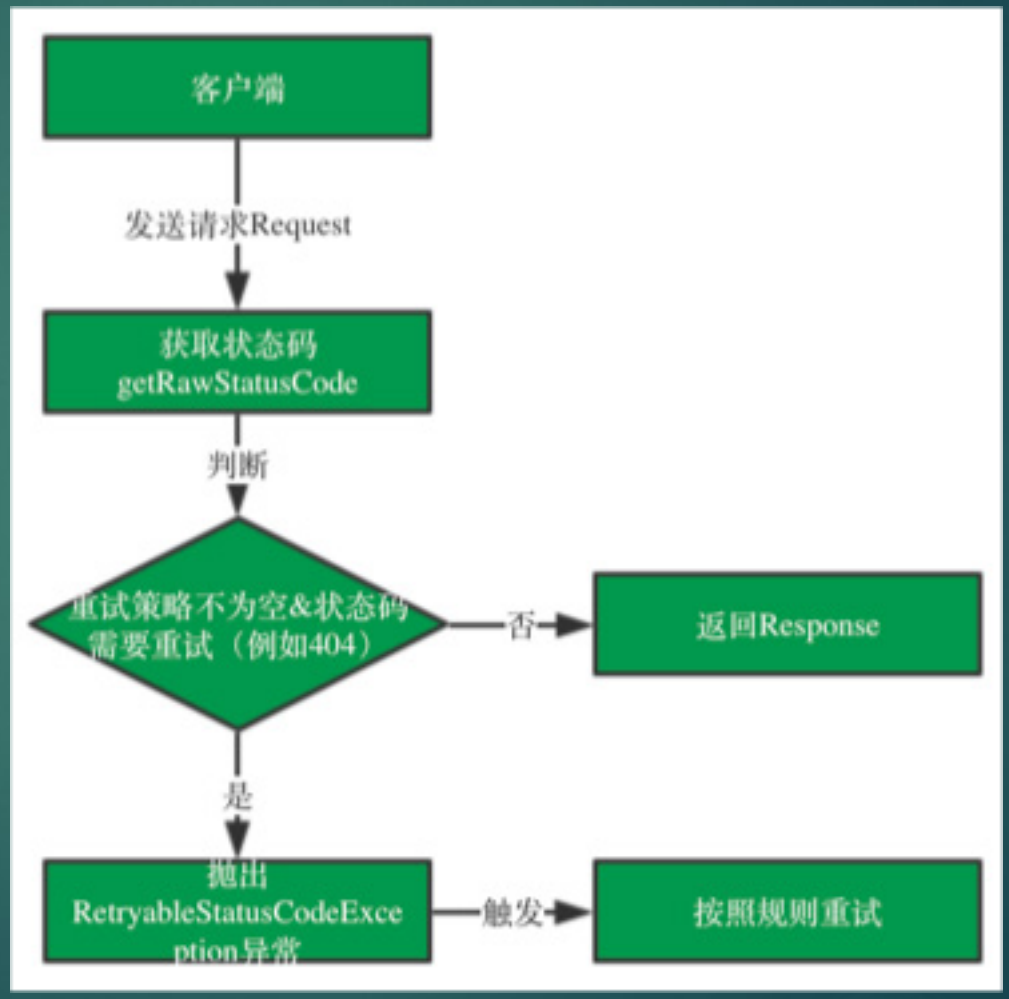
Ribbon的重试- Spring-Retry

- 在加载拦截器的时候通过@ConditionalOnClass和@ConditionalOnMissingClass来控制是加入LoadBalancer拦截器还是加入RetryLoadBalancer拦截器。



Ribbon的重试—重试的触发

- 判断返回状态码
- 需要重试时抛出特定异常
- 按照规则重试



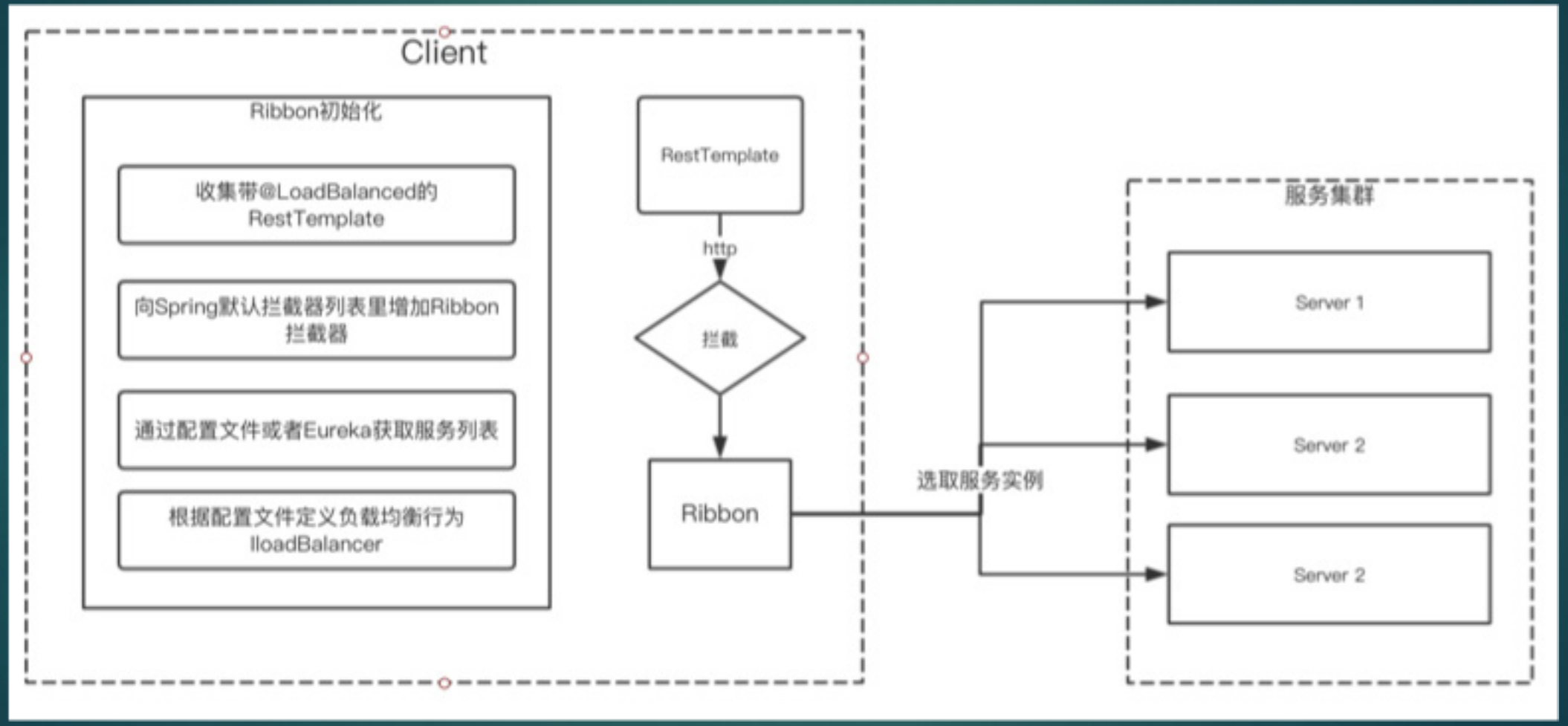
Ribbon的重试-默认规则

- 默认重试规则

```
# Max number of retries on the same server (excluding the first try)
sample-client.ribbon.MaxAutoRetries=1

# Max number of next servers to retry (excluding the first server)
sample-client.ribbon.MaxAutoRetriesNextServer=1
```

总结



感谢