微服务之Spring Cloud Ribbon

一、简介

1.1 Ribbon是什么

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。

官方给 Ribbon 的定义是一个用于远程调用的库（实现服务消费），而它更为人熟知的作用是可以进行客户端的 Load Balance（负载均衡）。

1.2 Ribbon核心概念

Rule — 这个组件决定了从一堆候选服务器中返回那个服务器地址进行远程调用的操作。 Ping — 在后台运行的组件，确认哪些服务器是存活可用的。 ServerList — 当前可以用作 LB 的服务器列表，这个列表可以是静态的，也可以是动态的。如果是动态列表(例如从 Eurka 服务器获取)，则会有一个后台线程按照时间间隔刷新列表。

1.3 Ribbon的实现原理

ribbon实现的关键点是为ribbon定制的RestTemplate，ribbon利用了RestTemplate的拦截器机制，在拦截器中实现ribbon的负载均衡。负载均衡的基本实现就是利用applicationName从服务注册中心获取可用的服务地址列表，然后通过一定算法负载，决定使用哪一个服务地址来进行http调用。

RestTemplate中有一个属性是List interceptors,如果interceptors里面的拦截器数据不为空，在RestTemplate进行http请求时，这个请求就会被拦截器拦截进行，拦截器实现接口ClientHttpRequestInterceptor

1.4 @LoadBalanced

严格上来说，这个注解是spring cloud实现的，不是ribbon中的，它的作用是在依赖注入时，只注入实例化时被@LoadBalanced修饰的实例。为我们定义的RestTemplate注入拦截器.

目前主流的LB方案可分成两类：一种是集中式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；另一种是进程内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就属于后者，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。

1.5 负载均衡算法

1:简单轮询负载均衡（RoundRobinRule):以轮询的方式依次将请求调度不同的服务器，即每次调度执行i = (i + 1)，并选出第i台服务器。

2:随机负载均衡 （RandomRule): 随机选择状态为UP的Server

3:加权响应时间负载均衡 （WeightedResponseTimeRule):根据相应时间分配一个weight，相应时间越长，weight越小，被选中的可能性越低。

4:区域感知轮询负载均衡（ZoneAvoidanceRule):复合判断server所在区域的性能和server的可用性选择server

5:并发量小的（BestAvailableRule）：根据判断当前所有的服务，找到并发量最小的服务

SpringCloud实现服务的消费，有两种方式：

1、基于Feign实现服务消费（接口-映射路径）

2、基于Ribbon+RestTemplate 实现服务消费（工具类 类 手动请求）

二、初体验

1、依赖

``

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
</dependency>
```

2、编写配置类

实现RestTemplate对象的创建，并设置负载均衡算法

``

```
@Configuration
public class RibbonConfig {
    @Bean
    @LoadBalanced  //标记 启用负载均衡
    public RestTemplate createRT(){
        return new RestTemplate();
    }
    @Bean
    public IRule create(){
        //
//        new RoundRobinRule();//轮询
//        new WeightedResponseTimeRule();//权重
//        new ZoneAvoidanceRule();//区域感知
//        new BestAvailableRule();//分配当前并发量小的服务
        return new RandomRule();//随机
    }
```

3、编写代码实现实现服务消费

创建类，注入RestTemplate

根据服务提供者的映射路径，创建对应的方法实现接口的请求

RestTemplate的常用方法说明：

getForObject：进行get请求

参数说明：

1、请求的服务提供者的路径 http://服务名称/映射路径

2、数据接口的返回值的Class对象

3、需要传递的参数

postForObject：进行post请求

参数说明：

1、请求的服务提供者的路径 格式：http://服务名称/映射路径

2、数据接口的返回值的Class对象

3、需要传递的参数

put：进行put请求

参数说明：

1、请求路径名称 格式：http://服务名称/映射路径

2、传递的参数

delete：进行delete请求

参数说明：

1、请求路径名称 格式：http://服务名称/映射路径

2、传递的参数

``

```
@Service
public class HelloService {
    @Autowired
    private RestTemplate restTemplate;
    public String num1(){
//        restTemplate.postForObject() //post请求
//        restTemplate.put(); //put请求
//        restTemplate.delete(); //delete请求
        return
restTemplate.getForObject("http://HelloProvider/hello/first.do",String.class);
    }
}
```

编写代码实现控制器

``

```
@RestController
public class HelloController {
    @Autowired
    private HelloService helloService;
    @GetMapping("/api/hello/num.do")
    public R hello(){
        return helloService.num();
    }
}
```

4、配置全局配置文件

设置注册中心

``

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
server:
    port: 9811
spring:
    application:
        name: HelloRibbon
```

5、设置开关类

使用注解:

@EnableDiscoveryClient 发现服务

``

```
@SpringBootApplication
@EnableDiscoveryClient
@RibbonClients
public class CloudRibbonconsumerApplication {
    public static void main(String[] args) {
        SpringApplication.run(CloudRibbonconsumerApplication.class, args);
    }
}
```

6、运行测试