

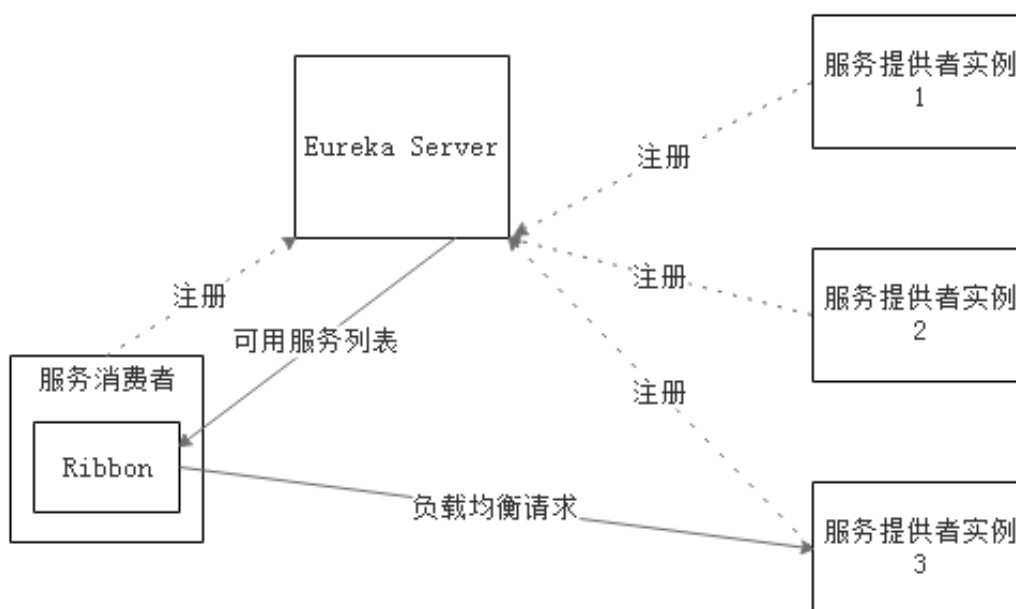
Ribbon

一 Ribbon介绍

使用 Eureka 后我们可以通过Eureka服务端找到我们注册的微服务客户端,然后通过我们的网络请求调用,但是在实际开发中,我们的微服务都是集群用于提高可用性,那么我们在调用这些服务的时候,如何实现负载均衡,我们就需要用到Ribbon 官方文档中说明 Feign 中自带 ribbon,所以使用Feign的时候不需要导入Ribbon的依赖包

Ribbon是Netflix发布的云中间层服务开源项目，其主要功能是提供客户端侧负载均衡算法。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，Ribbon是一个客户端负载均衡器，我们可以在配置文件中列出Load Balancer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。

下图展示了Eureka使用Ribbon时候的大致架构：



Ribbon工作时分为两步：第一步先选择 Eureka Server, 它优先选择在同Zone且负载较少的Server；第二步再根据用户指定的策略，在从Server取到的服务注册列表中选择一个地址。其中Ribbon提供了多种策略，例如轮询round robin、随机Random、根据响应时间加权等

二 Ribbon使用

2.1 基本使用

因为Ribbon是客户端侧负载均衡,所以我们只需要在消费者一方进行配置即可

参考06consumer-eureka-ribbon

04provider-eureka 作为提供者

05consumer-eureka 注册中心

2.1.1 POM 添加依赖

```
<!--添加Ribbon的starter-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-
ribbon</artifactId>
</dependency>
```

2.1.2 修改主程序

添加@RibbonClient注解,并且给RestTemplate的创建方法添加 @LoadBalanced注解

这样当我们通过RestTemplate调用请求都时候会先判断当前请求的服务是不是被配置了RibbonClient注解,如果配置了,就通过EurekaClient去获取一个地址(EurekaClient带负载均衡功能),然后发起调用,实际上是对我们之前的代码的一个封装

```
import org.springframework.boot.SpringApplication;
```

```

import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.ribbon.RibbonClient;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

/**
 * Created by jackiechan on 19-6-8/下午12:04
 *
 * @Author jackiechan
 */
@SpringBootApplication
@EnableEurekaClient
//@RibbonClients({@RibbonClient("04provider-eureka"),@RibbonClient("05provider-eureka")})//给多个服务开启负载均衡
//@RibbonClient("04provider-eureka")//设置单个服务开启负载均衡,当我们使用这个服务
//的时候会自动开启负载均衡
public class ConsumerStarter {

    public static void main (String[] args){
        SpringApplication.run(ConsumerStarter.class,args);
    }

    @Bean
    @LoadBalanced// 给当前对象开启负载均衡,具体是在调用
    //@RibbonClient("04provider-eureka")这个配置设置的服务的时候开启
    public RestTemplate restTemplate() {
        RestTemplate restTemplate = new RestTemplate();
        return restTemplate;
    }
}

```

2.1.3 修改Controller

因为已经对RestTemplate开启了注解,所以之前我们通过Eurekaclient获取地址然后调用的方式就不需要了,直接恢复到一开始的请求方式即可,只是注意,中间的地址换成了我们的服务地址

如果没有给RestTemplate添加loadblanced注解的话就会提示找不到主机异常

```

import com.qianfeng.microservice.consumer.pojo.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

/**
 * Created by jackiechan on 19-6-8/下午12:00
 *
 * @Author jackiechan
 */
@RestController
@RequestMapping("/userconsumer")
public class UserController {
    @Autowired
    private RestTemplate template;

    @GetMapping("/info/{id}")
    public User getUserById(@PathVariable Long id) throws Exception {
        //注意 此处的请求地址变成了提供者在eureka上面的服务名
        User user = template.getForObject( "http://04provider-
eureka/user/info/" + id, User.class); //请求指定地址并将返回的json字符串转换为
User对象
        return user;
    }
    @PostMapping("/save")
    public String addUser(@RequestBody User user) {
        String result = template.postForObject("http://04provider-
eureka/user/save/", user, String.class); //post方式请求这个地址,并将user作为参
数传递过去(json格式),并将返回结果转换为String类型
        return result;
    }
}

```

2.1.4 启动测试

启动我们的Eureka,然后通过修改端口号的方式多次启动我们的提供者来模拟集群,然后再启动我们的消费者,进行调用,发现随便停止一个提供者,仍旧可以获取到数据

2.2 自定义Ribbon配置

在开发中我们可能会修改一些Ribbon的默认配置,比如负载均衡方式默认是轮询,可能修改为随机

https://cloud.spring.io/spring-cloud-static/Finchley.SR3/single/spring-cloud.html#_customizing_the_ribbon_client

参考 07consumer-eureka-ribbon-config

04provider-eureka 作为提供者

05consumer-eureka 注册中心

我们可以修改的主要是以下配置

The following table shows the beans that Spring Cloud Netflix provides by default for Ribbon:

Bean Type	Bean Name	Class Name
<code>IClientConfig</code>	<code>ribbonClientConfig</code>	<code>DefaultClientConfigImpl</code>
<code>IRule</code>	<code>ribbonRule</code>	<code>ZoneAvoidanceRule</code>
<code>IPing</code>	<code>ribbonPing</code>	<code>DummyPing</code>
<code>ServerList<Server></code>	<code>ribbonServerList</code>	<code>ConfigurationBasedServerList</code>
<code>ServerListFilter<Server></code>	<code>ribbonServerListFilter</code>	<code>ZonePreferenceServerListFilter</code>
<code>ILoadBalancer</code>	<code>ribbonLoadBalancer</code>	<code>ZoneAwareLoadBalancer</code>
<code>ServerListUpdater</code>	<code>ribbonServerListUpdater</code>	<code>PollingServerListUpdater</code>

2.2.1 创建自定义配置类

官方推荐我们使用配置类的方式来自定义配置,然后将自定义的配置类指定给对应的RibbonClient,通过不同的配置文件分配给不同的RibbonClient注解的方式来设置不同的规则



The `CustomConfiguration` class must be a `@Configuration` class, but take care that it is not in a `@ComponentScan` for the main application context. Otherwise, it is shared by all the `@RibbonClients`. If you use `@ComponentScan` (or `@SpringBootApplication`), you need to take steps to avoid it being included (for instance, you can put it in a separate, non-overlapping package or specify the packages to scan explicitly in the `@ComponentScan`).

注意在上图中,官方说明,我们的自定义的配置文件应该添加Configuration注解,但是不能把他放在我们的CompentScan或者SpringBootApplication(里面内置了CompentScan)注解能扫描到的包内,也就是默认情况下,我们的自定义的配置文件的目录层级要比SpringBootApplication注解修饰的主类层级要高,如果被扫描到了,那么当前配置文件将会成

功所有RibbinClient的默认配置

```
import com.netflix.loadbalancer.IRule;
import com.netflix.loadbalancer.RandomRule;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * Created by jackiechan on 19-6-9/上午9:45
 *
 * @Author jackiechan
 */
@Configuration
public class RibbonConfig {
    @Bean
    public IRule iRule() { //自定义负载均衡规则为随机
        return new RandomRule();
    }
}
```

2.2.2 修改主程序

```
import com.qianfeng.microservice.RibbonConfig;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.ribbon.RibbonClient;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

/**
 * Created by jackiechan on 19-6-8/下午12:04
 *
 * @Author jackiechan
 */
@SpringBootApplication
@EnableEurekaClient
```

```
//@RibbonClients({@RibbonClient("04provider-eureka"),@RibbonClient("04provider-eureka")})//给多个服务开启负载均衡,通过此方式只能给所有的开启统一的自定义配置
@Bean
@RibbonClient(value = "04provider-eureka",configuration =
RibbonConfig.class)//给04provider-eureka开启负载均衡,并指定了配置文件为
RibbonConfig.class,注意 RibbonConfig这个文件层级比当前类高,如果想给多个服务开启不同配置,那么通过多个RibbonClient来分开指定不同配置文件
public class ConsumerStarter {

    public static void main (String[] args){
        SpringApplication.run(ConsumerStarter.class,args);
    }

    @Bean
    @LoadBalanced// 给当前对象开启负载均衡,具体是在调用
    @RibbonClient("04provider-eureka")这个配置设置的的服务的时候开启
    public RestTemplate restTemplate() {
        RestTemplate restTemplate = new RestTemplate();
        return restTemplate;
    }
}
```

2.2.3 修改controller

在controller中添加方法来查看新的负载均衡是否生效,通过LoadBalancerClient来判断

```
package com.qianfeng.microservice.consumer.controller;

//
//
//          _oo0oo_
//          o8888888o
//          88" . "88
//          (| -_- |)
//          O\ = /O
//          ____/'---'\____
//          .   '  \ \ | /  '   .
//          / \ \ | | : | | \ \
//          / _||| -:- |||| - \
//          | | \ \ - /// | |
//          | \ | ' \ ---/ ' | |
//          \ .- \_ _ ' - \_ _/ - . /
```

```
//      _ _ _ \ . . ' / -- . -- \ \ . . _ _
//      . " " ' < \ . _ _ _ \ _ < | > _ / _ _ _ . ' > ' " " .
//      | | : \ - \ \ . ; \ \ _ / ; . \ / - \ ` : | |
//      \ \ \ \ - . \ _ _ _ \ / _ _ _ / . - \ / /
//      ===== \ - . _ _ _ _ \ - . _ _ _ _ \ / _ _ _ _ . - \ _ _ _ _ _ - ' =====
//      \ = - - - = '
//
//
//      .....
//
//      佛祖镇楼                      BUG辟易
//
//      佛曰：
//
//      写字楼里写字间，写字间里程序员；
//      程序人员写程序，又拿程序换酒钱。
//      酒醒只在网上坐，酒醉还来网下眠；
//      酒醉酒醒日复日，网上网下年复年。
//      但愿老死电脑间，不愿鞠躬老板前；
//      奔驰宝马贵者趣，公交自行程序员。
//      别人笑我忒疯癫，我笑自己命太贱；
//
//
import com.qianfeng.microservice.consumer.pojo.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.loadbalancer.LoadBalancerClient;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

/**
 * Created by jackiechan on 19-6-8/下午12:00
 *
 * @Author jackiechan
 */
@RestController
@RequestMapping("/userconsumer")
public class UserController {
    @Autowired
    private RestTemplate template;

    @Autowired
    private LoadBalancerClient loadBalancerClient; // Ribbon内部使用的负载均衡的客户端，spring会自动创建

    @GetMapping("/info/{id}")
    public User getUserById(@PathVariable Long id) throws Exception {
        //注意 此处的请求地址变成了提供者在eureka上面的服务名
    }
}
```



```

        User user = template.getForObject( "http://04provider-
eureka/user/info/" + id, User.class); //请求指定地址并将返回的json字符串转换为
User对象
        return user;
    }
    @PostMapping("/save")
    public String addUser(@RequestBody User user) {
        String result = template.postForObject("http://04provider-
eureka/user/save/", user, String.class); //post方式请求这个地址, 并将user作为参
数传递过去(json格式), 并将返回结果转换为String类型
        return result;
    }

    /**
     * 通过访问此方法查看返回的端口来判断负载均衡的规则
     * @return
     */
    @GetMapping("/test")
    public String test() {
        int port = loadBalancerClient.choose("04provider-
eureka").getPort(); //获取当前提供者的端口
        System.out.println("当前请求的是:" + port);
        return port + "";
    }
}

```

2.2.4 启动测试

同上, 启动eureka和多个服务提供者, 然后启动当前这个消费者, 通过访问test接口, 可以看到获取到的提供者的端口是随机变化的

2.3 注解配置方式(了解)

在上面的配置方式中, 我们的配置文件必须将包层级提高, 以防出现意外情况, 这样会导致我们项目的包层级看起来很乱, 所以我们期望配置文件还是放在专门的目录下, 然后可以单独使用, 其实我们上面的主要问题是, 我们的配置文件被spring的包扫描机制扫描到了, 所以我们只要想办法让它不要被扫描到就行了, 通过配置包扫描到属性排除掉我们的配置文件类就

行了

2.3.1 移动配置类

移动配置类到能被扫描到底包中

参考 08consumer-eureka-ribbon-config-annotation

04provider-eureka 作为提供者

05consumer-eureka 注册中心

2.3.2 编写一个注解

这个注解就是一个标记注解,所以可以随便写,没有任何含义

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

/**
 * Created by jackiechan on 19-1-9/上午10:05
 * 这个注解没有任何含义,我们拿来只是最终作为一个标记,所以此处名字是ABC,就是为了说明可以随便写
 * @Author jackiechan
 */
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ABC {
}
```

2.3.3 给配置类添加注解

```
import com.netflix.loadbalancer.IRule;
import com.netflix.loadbalancer.RandomRule;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * Created by jackiechan on 19-1-9/上午9:45
```

```

*
* @Author jackiechan
*/
@Configuration//按照官方提示,建议我们把这个类放到不被compentscan或者
springbootapplicaiton扫描到的位置,原因是一旦被扫描到,会变成全局规则
@ABC// 此注解没有任何意义,只是一个标记,给compentscan看的
public class RibbonConfig {
    @Bean
    public IRule iRule() {
        return new RandomRule();
    }
}

```

2.3.4 修改启动类

```

package com.qianfeng.microservice.consumer;

//
//
//          _oo0oo_
//          o8888888o
//          88" . "88
//          (| -_- |)
//          O\ = /O
//          ____/'\---'\____
//          .   '    \ \ | /   '   .
//          / \ \ | | : | | \ \
//          / _ \| | | -:- | | | \
//          | | \ \ - / / | |
//          | \ | ' ' \---/ ' ' | |
//          \ .-\_ _ ' ' \---/ _.-/
//          ___`'. .' /--.--\ `'. .' ___
//          ."" ' < ` .___ \_<|>/_ ___.' >'"" .
//          | | : ' \ \ .; \ _ /'; \ / - ` : | |
//          \ \ '-. \ _ __ \ /__ _/ .-' / /
//          =====`-.____'-.____\_____/_____.-`____.-'=====
//          \=====
//
//
//          .....
//          佛祖镇楼                      BUG辟易
//          佛曰：

```

```

//          写字楼里写字间，写字间里程序员；
//          程序人员写程序，又拿程序换酒钱。
//          酒醒只在网上坐，酒醉还来网下眠；
//          酒醉酒醒日复日，网上网下年复年。
//          但愿老死电脑间，不愿鞠躬老板前；
//          奔驰宝马贵者趣，公交自行程序员。
//          别人笑我忒疯癫，我笑自己命太贱；
//

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.ribbon.RibbonClient;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.FilterType;
import org.springframework.web.client.RestTemplate;

/**
 * Created by jackiechan on 19-1-8/下午12:04
 *
 * @Author jackiechan
 */
@SpringBootApplication
@EnableEurekaClient
//@RibbonClients({@RibbonClient(value = "04provider-eureka"),@RibbonClient(value = "04provider-eureka")})//给多个服务开启负载均衡
@RibbonClient(value = "04provider-eureka",configuration =
RibbonConfig.class)//设置单个服务开启负载均衡
@ComponentScan(excludeFilters = {@ComponentScan.Filter(type =
FilterType.ANNOTATION,value = ABC.class)} )//给包扫描添加不扫描的例外情况,判断
的标准就是目标类上面有ABC注解的都不会被扫描,ABC本身来说没有任何意义,只是一个标记,用来
排除掉标记,所以这个地方可以使用任何注解,只要配置文件上面的注解和排除的注解是一个类型就
行,当然我们项目中本身有意义的注解不能拿来当标记,以免导致失效
public class ConsumerStarter {

    public static void main (String[] args){
        SpringApplication.run(ConsumerStarter.class,args);
    }

    @Bean

```

```

@LoadBalanced// 给当前对象开启负载均衡, 具体是在调用
@RibbonClient("04provider-eureka")这个配置设置的服务的时候开启
public RestTemplate restTemplate() {
    RestTemplate restTemplate = new RestTemplate();
    return restTemplate;
}
}

```

2.4 配置文件方式

我们可以通过配置文件给我们的目标服务指定负载均衡方式,相比较前面来说,别的地方没有改变,只是配置文件有变化

参考 09consumer-eureka-ribbon-config-properties

04provider-eureka 作为提供者

05consumer-eureka 注册中心

支持以下属性

The following list shows the supported properties>:

```

<clientName>.ribbon.NFLoadBalancerClassName: Should implement
ILoadBalancer
<clientName>.ribbon.NFLoadBalancerRuleClassName: Should implement IRule
<clientName>.ribbon.NFLoadBalancerPingClassName: Should implement IPing
<clientName>.ribbon.NIWSServerListClassName: Should implement ServerList
<clientName>.ribbon.NIWSServerListFilterClassName: Should implement
ServerListFilter

```

2.4.1 application.yml

```

server:
  port: 9000
eureka: #注册中心的地址
  client:
    service-url:
      defaultZone: http://zhangsan:abc@localhost:10000/eureka #curl风格
spring:
  application:
    name: 09consumer-eureka-ribbon-config-properties
04provider-eureka: #给这个服务配置负载均衡规则,这里写服务的名字,多个就配置多次
  ribbon:
    NFLoadBalancerRuleClassName: com.netflix.loadbalancer.RandomRule #指
    定为随机

```

2.5 禁用ribbon

我们可以在eureka'中禁用ribbon,只需要在对应的项目的application.yml中添加以下配置即可

```

ribbon:
  eureka:
    enabled: false    #禁用 ribbon

```