**Spring Cloud**

**组件学习**

# Eureka（服务发现，注册）

英 [juəˈri:kə] 美 [jʊˈrikə]

Eureka是Spring Cloud Netflix的一个子模块，也是核心模块之一。用于云端服务发现，一个基于REST的服务，用于定位服务，以实现云端中间层服务发现和故障转移。

服务注册与发现对于微服务系统来说非常重要。有了服务发现与注册，你就不需要整天改服务调用的配置文件了，你只需要使用服务的标识符，就可以访问到服务。他的功能类似于dubbo的注册中心（register）。

服务发现：服务发现是微服务基础架构的关键原则之一。试图着手配置每个客户端或某种格式的约定可以说是非常困难的和非常脆弱的。Eureka是Netflix服务发现的一种服务和客户端。这种服务是可以被高可用性配置的和部署,并且在注册的服务当中，每个服务的状态可以互相复制给彼此。

 通过@EnableEurekaServer注解启动一个服务注册中心提供给其他应用进行对话。

服务注册:当一个客户端注册到Eureka,它提供关于自己的元数据（诸如主机和端口，健康指标URL，首页等）Eureka通过一个服务从各个实例接收心跳信息。如果心跳接收失败超过配置的时间，实例将会正常从注册里面移除。

 通过@EnableEurekaClient注解启动一个服务注册中心提供给其他应用进行对话。

## 架包依赖

1. <dependency>
2. <groupId>org.springframework.cloud</groupId>
3. <artifactId>spring-cloud-starter-eureka-server</artifactId>
4. </dependency>

说明：SpringCloud提供了多种服务发现的实现方式，例如：Eureka、Consul、Zookeeper。Spring Cloud支持得最好的是Eureka，其次是Consul，最次是Zookeeper。

## Service

#### 第一步：配置服务中心

*server.port*=8761  
*#Eureka Server configuration  
eureka.instance.hostname*=localhost  
*# 自我保护机制  
# 当Eureka Server节点在短时间内丢失过多的客户端时（可能发送了网络故障），  
# 那么这个节点将进入自我保护模式，不再注销任何微服务，当网络故障回复后，该节点会自动退出自我保护模式。  
#自我保护模式的架构哲学是宁可放过一个，决不可错杀一千  
eureka.server.enable-self-preservation*=true  
  
*eureka.server.eviction-interval-timer-in-ms*=60000  
*eureka.client.register-with-eureka*=false  
*eureka.client.fetch-registry*=false  
*#心跳间隔  
eureka.instance.lease-renewal-interval-in-seconds*=30  
*#可以用来修改自我保护系数（默认0.85）。  
eureka.server.renewal-percent-threshold*=0.85  
*#配置集群节点 如果是多个用逗号隔开如： http://peer1:8761/eureka/,http://peer2:8762/eureka/,http://peer3:8763/eureka/  
eureka.client.serviceUrl.defaultZone*=http://${*eureka.instance.hostname*}:${*server.port*}/eureka/

### 第二步：构建服务中心@EnableEurekaServer

@SpringBootApplication  
@EnableEurekaServer  
*public class* EurekaServerApplication {  
  
 *public static void* main(String[] args) {  
 SpringApplication.*run*(EurekaServerApplication.*class*, args);  
 }  
}

## Client

#### 第一步：配置提供服务的相关信息：如服务项目名称

*server.port*=8762  
*spring.application.name*=service-hi  
*#Eureka Client  
# spring-boot-starter-actuator  
eureka.client.healthcheck.enabled*=true  
*eureka.instance.lease-renewal-interval-in-seconds*=30  
*eureka.instance.lease-expiration-duration-in-seconds*=30  
*eureka.client.serviceUrl.defaultZone*=http://localhost:8761/eureka/

#### 第二步：构建服务信息@EnableEurekaClient

@SpringBootApplication  
@EnableEurekaClient  
@RestController   
*public class* EurekaClientApplication {  
  
 *public static void* main(String[] args) {  
  
 SpringApplication.*run*(EurekaClientApplication.*class*, args);  
 }  
 @Value("${server.port}")  
 String port;  
 @RequestMapping("/hi")  
 *public* String home(@RequestParam String name) {  
 *return* "hi "+name+",i am from port:" +port;  
 }  
}

# Ribbon（服务消费者）：restTemplate进行消费

英 [ˈrɪbən] 美 [ˈrɪbən]

ribbon是一个负载均衡客户端，可以很好的控制htt和tcp的一些行为。Feign默认集成了ribbon。

ribbon 已经默认实现了这些配置bean：

* IClientConfig ribbonClientConfig: DefaultClientConfigImpl
* IRule ribbonRule: ZoneAvoidanceRule
* IPing ribbonPing: NoOpPing
* ServerList ribbonServerList: ConfigurationBasedServerList
* ServerListFilter ribbonServerListFilter: ZonePreferenceServerListFilter
* ILoadBalancer ribbonLoadBalancer: ZoneAwareLoadBalancer

### 架包依赖

1. <dependency>
2. <groupId>org.springframework.cloud</groupId>
3. <artifactId>spring-cloud-starter-eureka-ribbon </artifactId>
4. </dependency>

### 案例

#### 第一步：配置消费中心

*server.port*=8764  
*spring.application.name*=service-ribbon  
*#Eureka Client  
eureka.client.healthcheck.enabled*=true  
*eureka.instance.lease-renewal-interval-in-seconds*=30  
*eureka.instance.lease-expiration-duration-in-seconds*=30  
*eureka.client.serviceUrl.defaultZone*= http://localhost:8761/eureka/

#### 第二步：初始化服务消费端@EnableDiscoveryClient 及RestTemplate

@SpringBootApplication  
@EnableDiscoveryClient   
*public class* ServiceRibbonApplication {  
  
 *public static void* main(String[] args) {  
 SpringApplication.*run*(ServiceRibbonApplication.*class*, args);  
 }  
  
 @Bean  
 @LoadBalanced  
 RestTemplate restTemplate() {  
 *return new* RestTemplate();  
 }  
}

#### 第三步：指定需要消费的服务RestTemplate进行消费，并制定失败回掉fallbackMethod的方法

@Service  
*public class* HelloService {  
  
 @Autowired  
 RestTemplate restTemplate;  
  
 @HystrixCommand(fallbackMethod = "hiError")  
 *public* String hiService(String name) {  
 *return* restTemplate.getForObject("http://SERVICE-HI/hi?name="+name,String.*class*);  
 }  
 *public* String hiError(String name){  
 *return* "sorrry "+name+" server is Error";  
 }  
}

说明：当我们通过调用restTemplate.getForObject(“[http://SERVICE-HI/hi?name=](http://service-hi/hi?name=)“+name,String.class)方法时，已经做了负载均衡，访问了不同的端口的服务实例。

# Feign（服务消费者）

英 [feɪn] 美 [fen]

Feign是一个声明式的伪Http客户端，它使得写Http客户端变得更简单。使用Feign，只需要创建一个接口并注解。它具有可插拔的注解特性，可使用Feign 注解和JAX-RS注解。Feign支持可插拔的编码器和解码器。Feign默认集成了Ribbon，并和Eureka结合，默认实现了负载均衡的效果。

简而言之：

* Feign 采用的是基于接口的注解
* Feign 整合了ribbon

## 架包依赖

1. <dependency>
2. <groupId>org.springframework.cloud</groupId>
3. <artifactId>spring-cloud-starter-eureka-feign</artifactId>
4. </dependency>

## 案例

### 第一步：配置消费中心

*server.port*=8765  
*spring.application.name*=service-feign  
*#Eureka Client  
eureka.client.healthcheck.enabled*=true  
*eureka.instance.lease-renewal-interval-in-seconds*=30  
*eureka.instance.lease-expiration-duration-in-seconds*=30  
*feign.hystrix.enabled*=true  
*eureka.client.serviceUrl.defaultZone*= http://localhost:8761/eureka/

### 第二步：初始化服务消费端@EnableFeignClients 和@EnableDiscoveryClient

@SpringBootApplication  
@EnableDiscoveryClient  
@EnableFeignClients  
*public class* ServiceFeignApplication {  
  
 *public static void* main(String[] args) {  
 SpringApplication.*run*(ServiceFeignApplication.*class*, args);  
 }  
}

说明：如果选用的注册中心是eureka，那么就推@EnableEurekaClient，如果是其他的注册中心，那么推荐使用@EnableDiscoveryClient

### 第三步：指定消费的服务

@FeignClient(value = "service-hi",fallback = HiServiceHystric.*class*)  
*public interface HiService* {  
  
 @RequestMapping(value = "/hi",method = RequestMethod.GET)  
 String sayHiFromClientOne(@RequestParam(name = "name")String name);  
}