# SpringCloud

* **eureka**  
  微服务治理，服务注册和发现
* **ribbon**  
  负载均衡、请求重试
* **hystrix**  
  断路器，服务降级、熔断
* **feign**  
  ribbon + hystrix 集成，并提供生命式客户端
* **hystrix dashboard 和 turbine**  
  hystrix 微服务监控
* **zuul**  
  API 网关，提供微服务的统一入口，并提供统一的权限验证
* **config**  
  配置中心
* **bus**  
  配置更新消息总线
* **sleuth**  
  链路追踪

## Eureka 注册与发现

eureka:

server:

enable-self-preservation: false

instance:

hostname: eureka1

client:

register-with-eureka: false

fetch-registry: false

* eureka 集群服务器之间，通过 hostname 来区分
* eureka.server.enable-self-preservation  
  eureka 的自我保护状态：心跳失败的比例，在15分钟内是否低于85%,如果出现了低于的情况，Eureka Server会将当前的实例注册信息保护起来，同时提示一个警告，一旦进入保护模式，Eureka Server将会尝试保护其服务注册表中的信息，不再删除服务注册表中的数据。也就是不会注销任何微服务
* eureka.client.register-with-eureka=false  
  不向自身注册
* eureka.client.fetch-registry=false  
  不从自身拉去注册信息
* eureka.instance.lease-expiration-duration-in-seconds  
  最后一次心跳后，间隔多久认定微服务不可用，默认90

### Eureka 高可用配置

**---**

spring:

profiles: eureka1

server:

port: 2001

*# eureka1 向 eureka2 注册*

eureka:

instance:

hostname: eureka1

client:

service-url:

defaultZone: http://eureka2:2002/eureka

**---**

spring:

profiles: eureka2

server:

port: 2002

*# eureka2 向 eureka1 注册*

eureka:

instance:

hostname: eureka2

client:

service-url:

defaultZone: http://eureka1:2001/eureka

只需要在启动项目时，选择Spring.profiles.active参数即可

### Eureka 注册配置

eureka:

client:

service-url:

defaultZone: http://eureka1:2001/eureka

* eureka.instance.lease-renewal-interval-in-seconds  
  心跳间隔时间，默认 30 秒
* defaultZone，默认位置，可以修改为具体地理位置，比如：beiJing, shangHai, shenZhen 等，表示 eureka 服务器的部署位置
* eureka.client.registry-fetch-interval-seconds  
  拉取注册信息间隔时间，默认 30 秒
* 主程序添加 @EnableDiscoveryClient注解

## Ribbon 负载均衡与重试（5-10），重试在（11-14）

* 创建 RestTemplate 实例  
  RestTemplate 是用来调用其他微服务的工具类，封装了远程调用代码，提供了一组用于远程调用的模板方法，例如：getForObject()、postForObject() 等
* 如果要实现负载均衡，则需要在主程序中加@LoadBalanced, @LoadBalanced 负载均衡注解，会对 RestTemplate 实例进行封装，创建动态代理对象，并切入（AOP）负载均衡代码，把请求分散分发到集群中的服务器，配置如下

ribbon:

MaxAutoRetriesNextServer: 2 #更换实例的次数

MaxAutoRetries: 1 #当前实例重试次数，尝试失败会更换下一个实例

OkToRetryOnAllOperations: true #对连接超时、读取超时都进行重试

## Hystrix 断路器

* 重试机制需要添加@HystrixCommand(fallbackMethod=”降级的方法名”)

### 暴露监控端点

management:

endpoints:

web:

exposure:

include: hystrix.stream

如果需要全部，则填”\*”,记得带双引号

## Feign整合Ribbon 与 Hystrix

### Feign负载均衡与重试

* 无需额外配置，feign 默认已启用了 ribbon 负载均衡和重试机制。可以通过配置对参数进行调整
* Feign 只需要定义接口，在对应的接口中定义接口方法，就可以实现调用，例子如下：

注意下面的url不能有斜杠/

**@FeignClient**("order-service")

**public** **interface** **OrderFeignService** {

**@GetMapping**("/{orderId}")

JsonResult<Order> **getOrder**(@PathVariable String orderId);

**@GetMapping**("/")

JsonResult **addOrder**();

}

* Controller层中直接调用上面定义的接口方法即可，例子如下：

**@RestController**

**public** **class** **FeignController** {

**@Autowired**

**private** ItemFeignService itemServcie;

**@Autowired**

**private** UserFeignService userServcie;

**@Autowired**

**private** OrderFeignService orderServcie;

**@GetMapping**("/item-service/{orderId}")

**public** JsonResult<List<Item>> getItems(**@PathVariable** String orderId) {

**return** itemServcie.getItems(orderId);

}

**@PostMapping**("/item-service/decreaseNumber")

**public** JsonResult **decreaseNumber**(@RequestBody List<Item> items) {

**return** itemServcie.decreaseNumber(items);

}

### Feign降级处理

* feign 默认没有启用 hystrix，添加配置，启用 hystrix

feign:

hystrix:

enabled: true

默认1秒会快速失败，没有降级方法时，会显示白板页，要设置超时失败，可以如下配置

hystrix:

command:

default:

execution:

isolation:

thread:

timeoutInMilliseconds: 500

* 如果失败降级，就会执行其接口的实现类,但是要在其接口的注解添加FallBack属性的值,不要忘记了实现类中加@Service注解

**@Component**

**public** **class** **ItemFeignServiceFB** **implements** **ItemFeignService** {

**@Override**

**public** JsonResult<List<Item>> getItems(String orderId) {

**return** JsonResult.err("无法获取订单商品列表");

}

**@Override**

**public** JsonResult **decreaseNumber**(List<Item> items) {

**return** JsonResult.err("无法修改商品库存");

}

}

### Feign监控和熔断

添加actuator（监控系统）依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

暴露端点

management:

endpoints:

web:

exposure:

include: hystrix.stream

* 关于熔断的一些配置如下：

<https://github.com/Netflix/Hystrix/wiki/Configuration>

* hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds   
  请求超时时间，超时后触发失败降级
* hystrix.command.default.circuitBreaker.requestVolumeThreshold   
  10秒内请求数量，默认20，如果没有达到该数量，即使请求全部失败，也不会触发断路器打开
* hystrix.command.default.circuitBreaker.errorThresholdPercentage   
  失败请求百分比，达到该比例则触发断路器打开
* hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds   
  断路器打开多长时间后，再次允许尝试访问（半开），仍失败则继续保持打开状态，如成功访问则关闭断路器，默认 5000