## 服务发现简介

使用服务发现组件后的架构图：

|  |
| --- |
| IMG_256 |

* Eureka Server：服务的注册中心，负责维护注册的服务列表。
* Service Provider：服务提供方，作为一个Eureka Client，向Eureka Server做服务注册、续约和下线等操作，注册的主要数据包括服务名、机器ip、端口号、域名等等。
* Service Consumer：服务消费方，作为一个Eureka Client，向Eureka Server获取Service Provider的注册信息，并通过远程调用与Service Provider进行通信。

Service Provider和Service Consumer不是严格的概念，Service Consumer也可以随时向Eureka Server注册，来让自己变成一个Service Provider。

服务提供者、服务消费者、服务发现组件三者之间的关系：

1. 各个微服务在启动时，将自己的网络地址等信息注册到服务发现组件，服务发现组件会存储这些信息
2. 服务消费者可以从服务发现组件中查询这些服务提供者的网络地址，并使用该地址调用服务提供者的接口
3. 各个微服务与服务发现组件通过发送心跳，服务发现组件如长时间无法与某个微服务实例通信，就会注销该实例
4. 微服务网络地址发生变更，会重新注册到服务发现组件

服务发现组件：

Eureka server配置信息：

|  |
| --- |
| server.port=8080 spring.application.name= eureka-server eureka.instance.hostname=localhost  #避免eureka向自己进行注册(是否注册自己的信息到eureka server 默认是true) eureka.client.register-with-eureka=**false** #避免eureka查找服务列表(是否拉取其他服务的信息，默认是true) eureka.client.fetch-registry=**false** # 关闭自我保护机制 eureka.server.enable-self-preservation=**false** # 每隔10s扫描服务列表，移除失效服务 eureka.server.eviction-interval-timer-in-ms=10000 eureka.client.service-url.defaultZone=http://${eureka.instance.hostname}:${server.port}/eureka |

1.先要关闭自我保护 enable-self-preservation: false

2.eviction-interval-timer-in-ms 启用主动失效，并且每次主动失效检测间隔为10s

Eureka Server会定时（间隔值eureka.server.eviction-interval-timer-in-ms，默认值为0，默认情况不删除实例）进行检查，如果发现实例在在一定时间（此值由客户端设置的eureka.instance.lease-expiration-duration-in-seconds定义，默认值为90s）内没有收到心跳，则会注销此实例。

服务提供者：

配置信息

|  |
| --- |
| server.port=8081 spring.application.name= eureka-user eureka.instance.hostname=localhost #开启健康检查(将健康状态传播到eureka server) eureka.client.healthcheck.enabled=**true** # 每隔10s发送一次心跳（服务续约的间隔，默认为30） eureka.instance.lease-renewal-interval-in-seconds=10 # 告知服务端30秒还未收到心跳的话，就将该服务移除列表（服务失效时间，默认90） eureka.instance.lease-expiration-duration-in-seconds=30   #其他服务获取地址时，提供ip而不是主机名 eureka.instance.prefer-ip-address=**true** eureka.client.service-url.defaultZone=http://localhost:8080/eureka/  spring.datasource.url=jdbc:mysql://127.0.0.1:3306/springcloud?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC spring.datasource.driver-class-name=com.mysql.jdbc.Driver spring.datasource.password=asd3884 spring.datasource.username=root  #mybatis mybatis.mapper-locations=classpath:mapper/user.xml |

服务提供者要向eureka server注册服务，并且完成服务续约等工作

1. 服务注册：

服务提供者在启动时，会检测配置属性

#避免eureka向自己进行注册(是否注册自己的信息到eureka server 默认是true)  
eureka.client.register-with-eureka=**true**

中的参数是否正确（默认为true）

如果确实为true，则会向eureka server 发起一个Rest请求，并携带自己的元数据信息，eureka server会把这些信息储存

1. 服务续约：

注册服务完成后，服务提供者会维持一个心跳（定时向eureka server发起Rest请求）

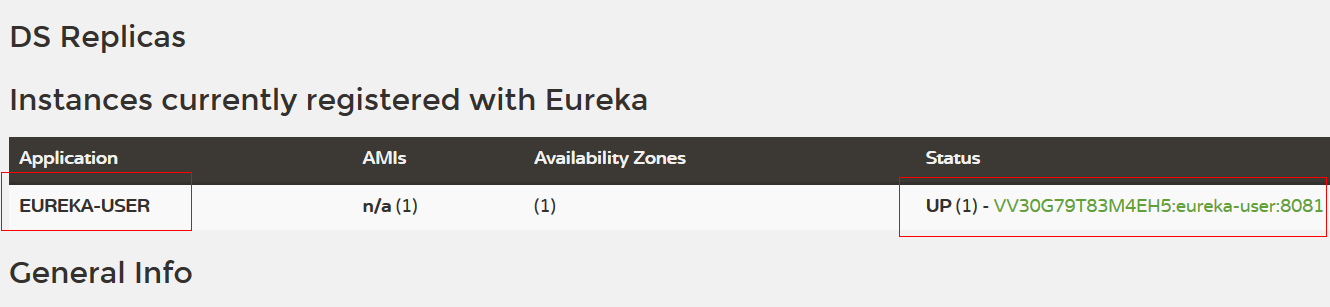
两个参数可以修改服务续约的行为：

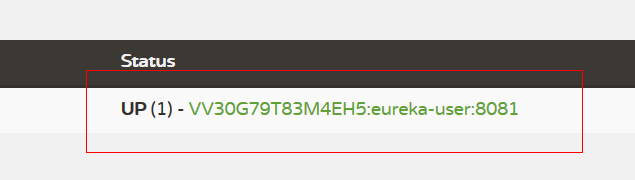
# 每隔10s发送一次心跳(服务续约的间隔，默认30秒)  
eureka.instance.lease-renewal-interval-in-seconds=10  
# 告知服务端30秒还未收到心跳的话，就将该服务移除列表(服务失效时间，默认90秒)  
eureka.instance.lease-expiration-duration-in-seconds=30

默认情况下：每隔30秒服务会向注册中心发送一次心跳，超过90秒没有发送心跳，eureka server会认为服务宕机，会从服务列表中移除

分别启动eureka、user

访问http://localhost:8080





如上：eureka监控页面 status：up(1) 启动了一个实例，没有集群

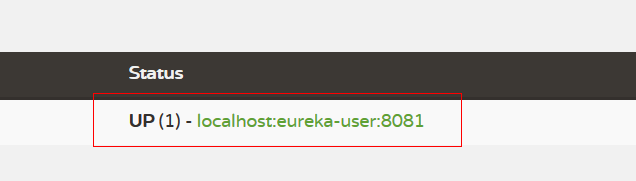
默认格式：${hostname}:${spring.application.name}:${server.port}

可以通过属性修改格式：

#eureka 监控页面status修改格式  
eureka.instance.instance-id=

${eureka.instance.hostname}:${spring.application.name}:${server.port}

再次访问http://localhost:8080如下：



服务消费者：

配置信息：

|  |
| --- |
| server.port=8082  spring.application.name= eureka-movie eureka.instance.hostname=localhost  #开启健康检查(将健康状态传播到eureka server) eureka.client.healthcheck.enabled=**true** #获取并更新数据的间隔 eureka.client.registry-fetch-interval-seconds=10   #其他服务获取地址时，提供ip而不是主机名 eureka.instance.prefer-ip-address=**true** eureka.client.service-url.defaultZone=http://localhost:8080/eureka/  spring.datasource.url=jdbc:mysql://127.0.0.1:3306/springcloud?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC spring.datasource.driver-class-name=com.mysql.jdbc.Driver spring.datasource.password=asd3884 spring.datasource.username=root |

获取服务列表

当服务消费者启动时，会检测

#避免eureka向自己进行注册(是否注册自己的信息到eureka server 默认是true)  
eureka.client.register-with-eureka=**true**

中的参数是否正确（默认为true）

如果确实为true，则会从eureka server服务的列表中只读备份，然后缓存到本地，并且每隔30秒会重新获取并更新数据

#获取并更新数据的间隔  
eureka.client.registry-fetch-interval-seconds=10

Ribbon实现负载均衡

Spring Cloud Ribbon是一个基于HTTP和TCP的客户端负载均衡工具，它基于Netflix Ribbon实现。通过Spring Cloud的封装，可以让我们轻松地将面向服务的REST模板请求自动转换成客户端负载均衡的服务调用。

微服务架构中使用客户端负载均衡调用非常简单，只需要如下两步：

​ 1.服务提供者只需要启动多个服务实例并注册到一个注册中心或是多个相关联的服务注册中心。

​ 2.服务消费者直接通过调用被@LoadBalanced注解修饰过的RestTemplate来实现面向服务的接口调用。