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

2、application.properties

a) server.port=

b) eureka.instance.hostname=

c) eureka.client.register-with-eureka=false（不向注册中心注册自己）

d) eureka.client.fetch-registry=false（不检索服务）

e) eureka.client.serviceUrl.defaultZone=http://${eureka.instance.hostname}:${server.port}/eureka/

3、@EnableDiscoveryClient注解，激活Eureka中的DiscoveryClient实现（自动化配置，创建DiscoveryClient接口针对Eureka客户端的EurekaDiscoveryClient实例），才能实现上述Controller中对服务信息的输出。

在客户端启动后，服务控制台中，Tomcat启动之后：

Com.netflix.discovery.DiscoveryClient : DiscoveryClient\_HELLO-SERVICE/PC-

201602152056:hello-service-registration status:204

代表服务注册成功。

c.n.e.registry.AbstractInstanceRegistry : Registered instance

HELLO-SERVICE/PC-201602152056:hello-service with status UP (replication=true)

代表名为hello-service的服务被注册成功了。

com.didispace.web.HelloController : /hello, host:PC-201602152056

service\_id:hello-service

代表之前在HelloController中注入的DiscoveryClient接口对象，从服务注册中心获取的服务相关信息。

4、高可用的注册中心

a) application.properties：[+]

（1）eureka.client.register-with-eureka=true

（2）eureka.client.fetch-registry=false

b) application-peer1.properties:

（1）spring.application.name=eureka-server

（2）server.port=1111

（3）eureka.instance.hostname=peer1

（4）eureka.client.serviceUrl.defaultZone=http://peer2:1112/eureka/

c) application-peer2.properties

（1）spring.application.name=eureka-server

（2）server.port=1112

（3）eureka.instance.hostname=peer2

（4）eureka.client.serviceUrl.defaultZone=http://peer1:1111/eureka/

d) 在/etc/hosts 文件中添加

127.0.0.1 peer1

127.0.0.1 peer2

[windows系统路径为C:\Windows\System32\drivers\etc\hosts]

e) 通过spring.profiles.active属性来分别启动peer1和peer2

（1）java –jar eureka-server-1.0.0.jar - -spring.profiles.active=peer1

（2）java –jar eureka-server-1.0.0.jar - -spring.profiles.active=peer1

f) Eureka界面General Info下available-replicas显示地址

关闭其中一个Eureka后，被关闭的地址显示在unavailable-replicas一栏

g) 在服务的配置文件application.properties中：

spring.application.name=hello-service

eureka.client.serviceUrl.defaultZone=http://peer1:1111/eureka/,<http://peer2:1112/eureka/>

【不想使用主机名可以使用IP地址定义，eureka.instance.prefer-ip-address=true】

5、服务发现与消费

Ribbon结合Eureka，Ribbon的服务实例清单扩展成从Eureka注册中心获取服务端列表。

1. 创建应用，@EnableDiscoveryClient注解让该应用注册为Eureka客户端应用，以获得服务发现的能力。

**6、基础架构**

a) 服务注册中心：eureka-server

b) 服务提供者：hello-service等

（1）发送REST请求的方式将自己注册到Eureka Server上

（2）确认eureka.client.register-with-eureka=true

c) 服务消费者：Ribbon【在高可用中，指向一个eureka。两个Ribbon分别指向两个eureka注册中心】、Feign

**服务同步：**

**{**

1. **hello-service：5800注册到peer1**
2. **hello-service：5801注册到peer2**
3. **因peer1，peer2之间相互注册，因此服务同步后，每个注册中心都能监测到两个服务。**

**}**

**服务续约**

**{**

**服务提供者持续告诉Eureka Server一个心跳以防止被“剔除任务”，这一操作叫做服务续约（Renew）**

**eureka.instance.lease-renewal-interval-in-seconds=30**

**[服务续约任务的调用间隔时间，默认为30秒]**

**eureka.instance.lease-expiration-duration-in-seconds=90**

**[服务失效的时间，默认为90秒]**

**}**

**服务下线**

**{**

**服务正常关闭后，告诉服务注册中心：“我要下线了”。服务端接收到消息后，设置服务状态为下线（DOWN），并把下线事件传播出去**

**}**

**服务注册中心**

**失效剔除**

**{**

**从服务列表中将无法提供服务的实例剔除，Eureka Server在启动的时候创建一个定时任务，默认每隔一段时间（默认为60秒）将当前清单中超时（默认为90秒）没有续约的服务剔除出去。**

**}**

**自我保护【经常出现在信息面板中的红色警告就是自我保护引起的】**

**{**

**Eureka Server会将当前的实例注册信息保护起来，让这些实例不会过期，尽可能保护这些注册信息。但是，在这段保护期间内实例若出现问题，那么客户端很容易拿到实际已经不存在的服务实例，故，客户端需有容错机制（请求重试、断路器等）**

**Eureka.server.enable-self-preservation=false**

**关闭保护机制，以确保注册中心可以将不可用的实例正确剔除**

**}**