# 服务治理：Spring Cloud Eureka

### 服务治理

服务治理是微服务架构中最为核心和基础的模块，它主要用来实现各个微服务实例的自动化注册与发现。

**·**服务治理的优点：

服务实例自动化治理极大减少手工维护成本。

**·**静态配置的缺点：

如果没有服务治理，若想完成服务间的调用需要手工维护服务的实例清单。随着业务的发展，微服务应用不断增加，集群规模、服务位置、服务命名的变化都会导致大量的手工维护静态配置的成本以及容易发生的错误命名冲突等问题。

**·**服务注册：

服务治理框架会构建一个注册中心，每个服务向注册中心登记自己提供的服务（主机与端口号、版本号、通信协议等）。

心跳检测机制：

服务中心会以心跳检测机制来检查清单中的服务是否可用，若不可用则从服务清单中剔除，以此来排除故障服务。

**·**服务发现：

服务治理框架使得服务间可以通过服务名来相互调用，而非服务地址。

原理：

调用方调用某个服务会首先去注册中心咨询服务，注册中心会返回所有的实例清单，调用方收到清单根据服务名称获取服务地址等信息。

1. Eureka简介

Eureka既包含服务端，也包含了客户端，采用Java编写。Eureka服务端的服务治理机制提供了完备的Restful API。

**·**服务端：

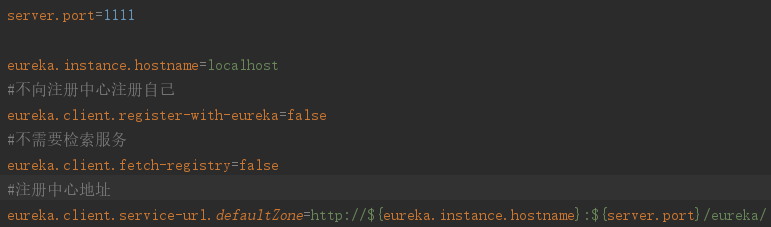
Eureka服务注册中心支持高可用，并且保证最终一致性。当集群中有分片故障，那么Eureka就进入保护模式，这期间对外继续提供服务，当故障恢复时，集群中的其他分片会把它们的状态再次同步回来。

**·**客户端：

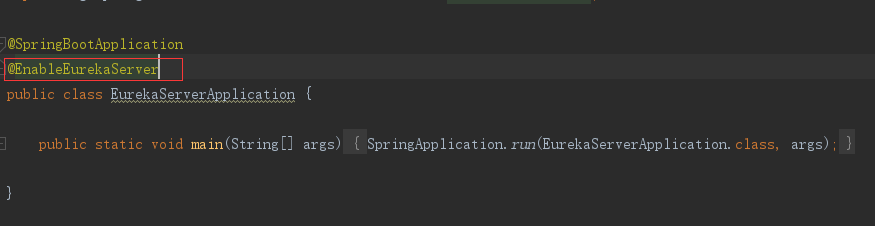
主要处理服务的注册与发现，客户端服务通过注解或者参数配置的方式嵌入在客户端程序的代码中，在应用运行时，Eureka客户端向注册中心注册自身提供的服务并周期性的发送心跳来更新它的服务租约。同时，它也能从服务端查询当前注册的服务信息并把它们缓存到本地并周期性的刷新服务状态。

1. 搭建服务注册中心

**·**配置文件：



**·**启动类：

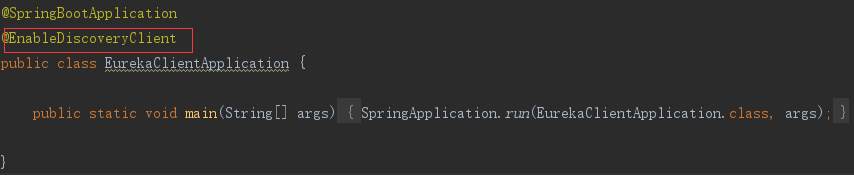


1. 注册服务提供者

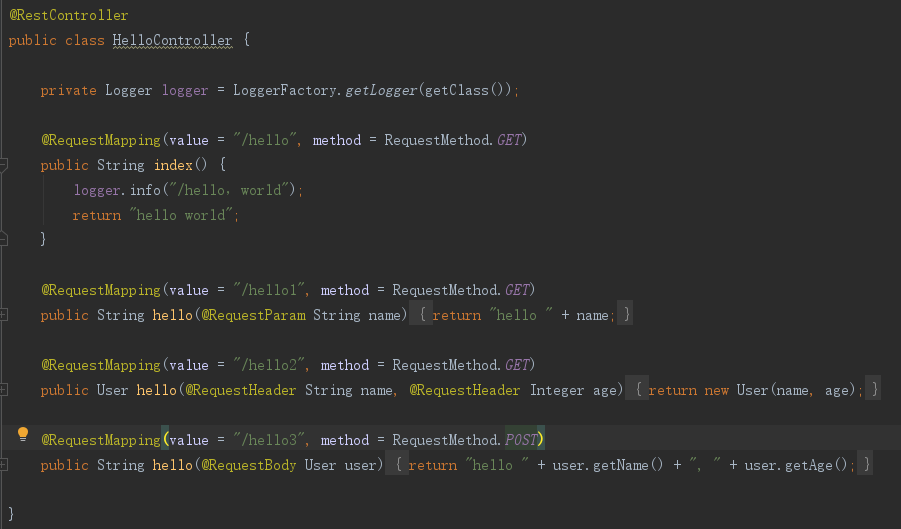
**·**配置文件：



**·**启动类：



**·**接口：

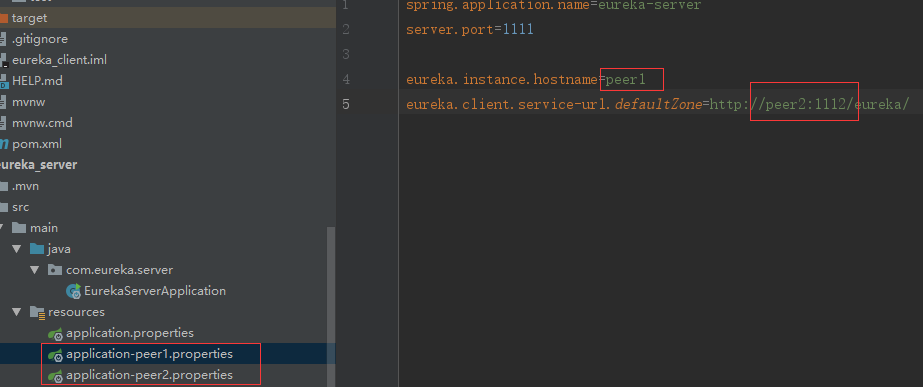


1. 高可用注册中心

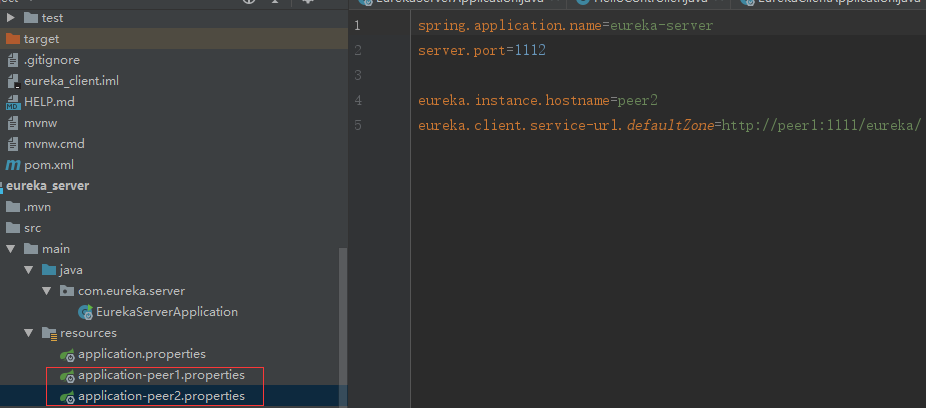
所有节点既是服务方也是消费方。

**·**配置文件：

服务1



服务2



**·**命令行启动

java -jar eureka-server-1.0.0.jar --spring.profiles.active=peer1

java -jar eureka-server-1.0.0.jar --spring.profiles.active=peer2

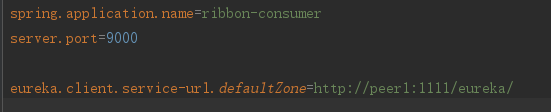
**·**服务提供方配置



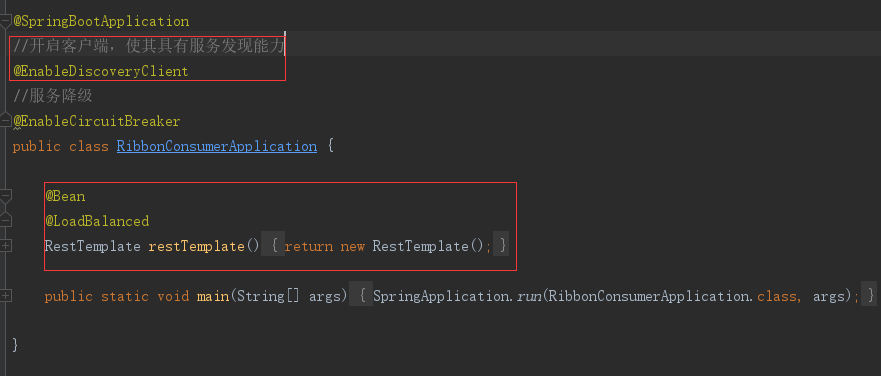
1. 服务发现与消费

服务消费的任务由Ribbon完成，Ribbon是一个基于HTTP和TCP的客户端负载均衡器。

**·**配置文件：



**·**启动类：



**·**接口：



### Eureka详解

1. 基础架构

**·**服务注册中心：

Eureka提供的服务端，提供服务注册与发现的功能。

**·**服务提供者：

提供服务的应用，可以是Spring Boot的应用，也可以是其他技术平台遵循Eureka通信机制的应用。它将自己提供的服务注册到Eureka，以供其他应用发现。

**·**服务消费者：

消费者应用从服务注册中心获取服务列表，从而使消费者可以知道去何处调用其所需要的服务。

很多时候，客户端既是服务提供者，也是服务消费者。

1. 服务治理机制

**·**服务提供者：

1. 服务注册

服务提供者在启动的时候会通过发送REST请求的方式将自己注册到Eureka Server，同时附带自身的元数据信息。Eureka Server收到这个REST请求后，将元数据存储到一个双层结构的Map中，其中第一层的key是服务名，第二层的key是具体服务的实例名称。

eureka.client.register-with-eureka=ture才会执行注册操作。

1. 服务同步

服务注册中心之间互相注册为服务，当服务提供者发送注册请求到一个服务中心时，它会将该请求转发给集群中相连的其他注册中心，从而实现注册中心之间的服务同步。

1. 服务续约

在注册完服务之后，服务提供者会维护一个心跳用来持续告诉Eureka Server：“我还活着”，以防止Eureka Server的“剔除任务”将该服务实例从服务列表中排除出去，该操作被称为服务续约。

服务续约任务的调用时间间隔

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

服务失效时间

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

1. 服务消费者

**·**获取服务：

启动服务消费者的时候，它会发送一个REST请求给服务注册中心，来获取上面注册的服务清单。为了性能考虑，Eureka Server会维护一份只读的服务清单来返回给客户端，同时该缓存清单会每隔固定时间（默认30s）更新一次。

**·**服务调用：

服务消费者在获取服务清单后，通过服务名可以获得具体提供服务的实例名和该实例的元数据信息。因为有这些服务实例的详细信息，所以客户端可以根据自己的需要决定具体调用哪个实例，在Ribbon中会默认采用轮询的方式进行调用，从而实现客户端的负载均衡。

Eureka中有Region和Zone的概念，一个Region中可以包含多个Zone，每个服务客户端需要被注册到一个Zone中，所以每个客户端对应一个Region和一个Zone。在进行服务调用的时候，优先访问同处一个Zone中的服务提供方，若访问不到，就访问其他的Zone。

**·**服务下线：

在服务关闭时，会发送一个REST请求给Eureka Server，告诉注册中心要下线了。服务端在收到请求后，就会将该服务状态置为下线状态，并把该事件传播出去。下线后的服务不会被调用。

1. 服务注册中心

**·**失效剔除：

当某些服务实例由于内存溢出、网络故障等原因无法正常工作，而注册中心不会收到“服务下线”请求。为了将这些无效实例从服务列表中剔除，Eureka Server会创建一个定时任务，默认每隔一段时间（默认60s）将当前清单中超时（默认90s）没有续约的服务剔除出去。

**·**自我保护：

Eureka Server在运行期间，会统计心跳失败的比例在15分钟之内是否低于85%，如果出现低于的情况，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，这段保护期间内实例若出现问题，那么客户端很容易难道实际已经不存在的服务实例，会出现调用失败的情况，所以客户端必须要有容错机制，比如可以使用请求重试，断路器等。