

# SpringCloud 微服务架构系列 -- 服务的注册、发现与调用

作者: [Qiyue0726](#)

原文链接: <https://ld246.com/article/1566920510750>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

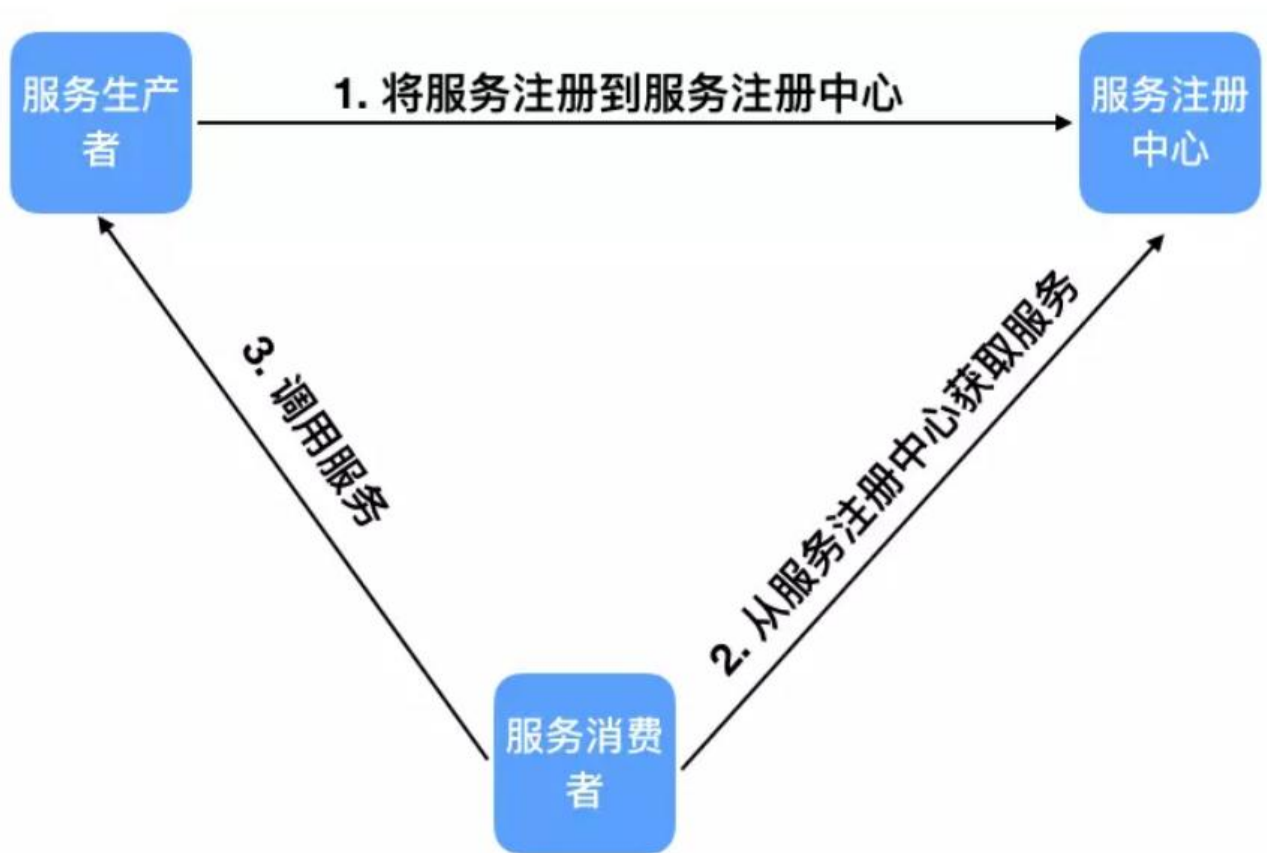
# 简介

众所周知，SpringCloud是一个微服务框架，本质上是基于SpringBoot的一整套实现微服务的框架。含了服务发现、负载均衡、断路器、服务网关、分布式配置等组件。其中服务注册、发现又有Netflix Eureka，阿里的Dubbo，Apache的Consul等，本篇文章将以Eureka进行讲解。

## Eureka

Eureka是Netflix开发的服务发现框架，本身是一个基于REST的服务。由两个组件组成：

- Eureka Server：也被称作是服务注册中心，用于提供服务的注册与发现。
- Eureka Client：包含服务消费者与服务生产者。



### Eureka 服务与发现调用关系

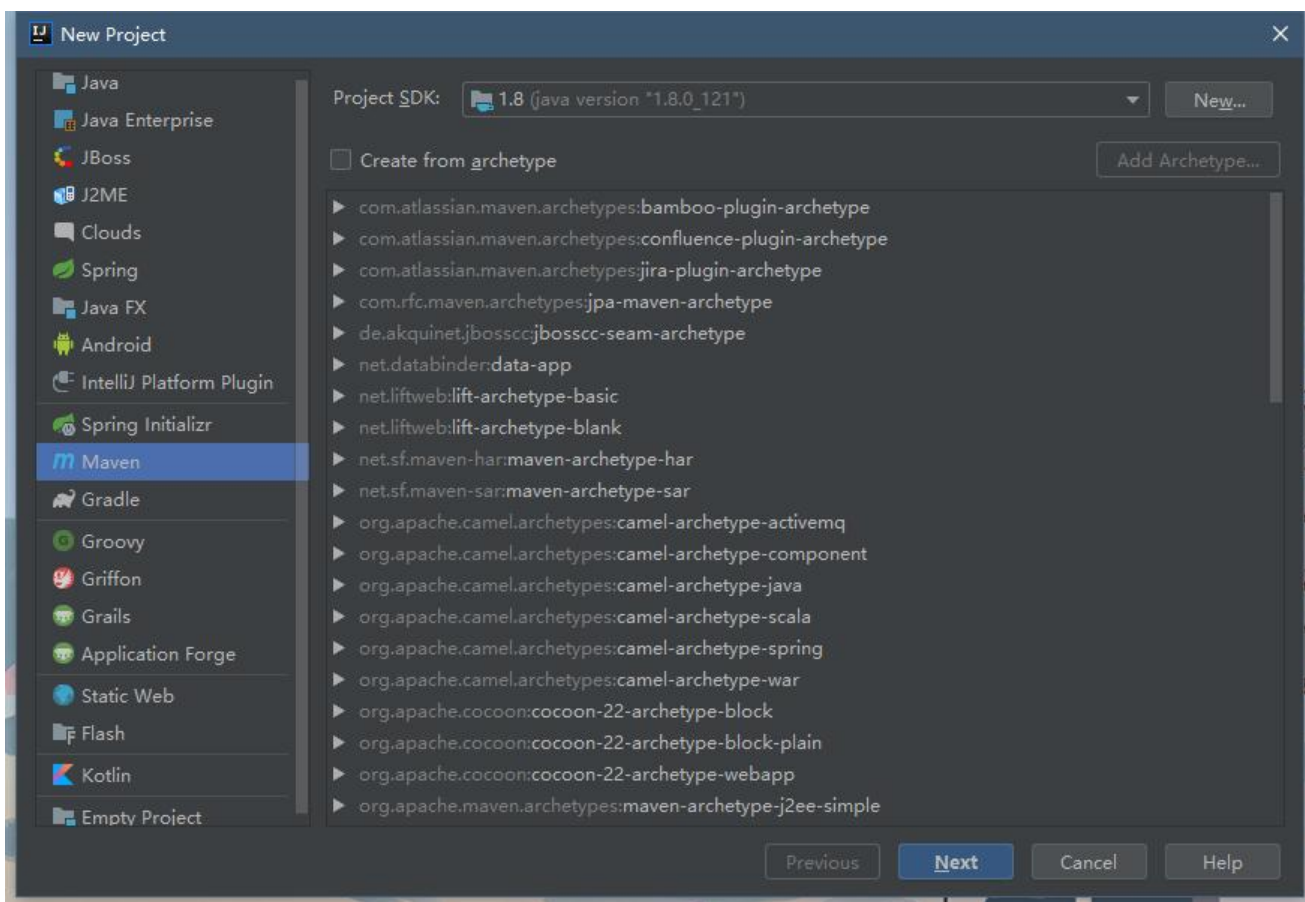
图片来源于互联网，如侵权可联系博主

Eureka的作用就是我们将定义的API接口注册到Eureka服务器上，方便管理，调用的时候只需要知道务名就可以，不再通过IP加端口号的方式调用，利于解耦。

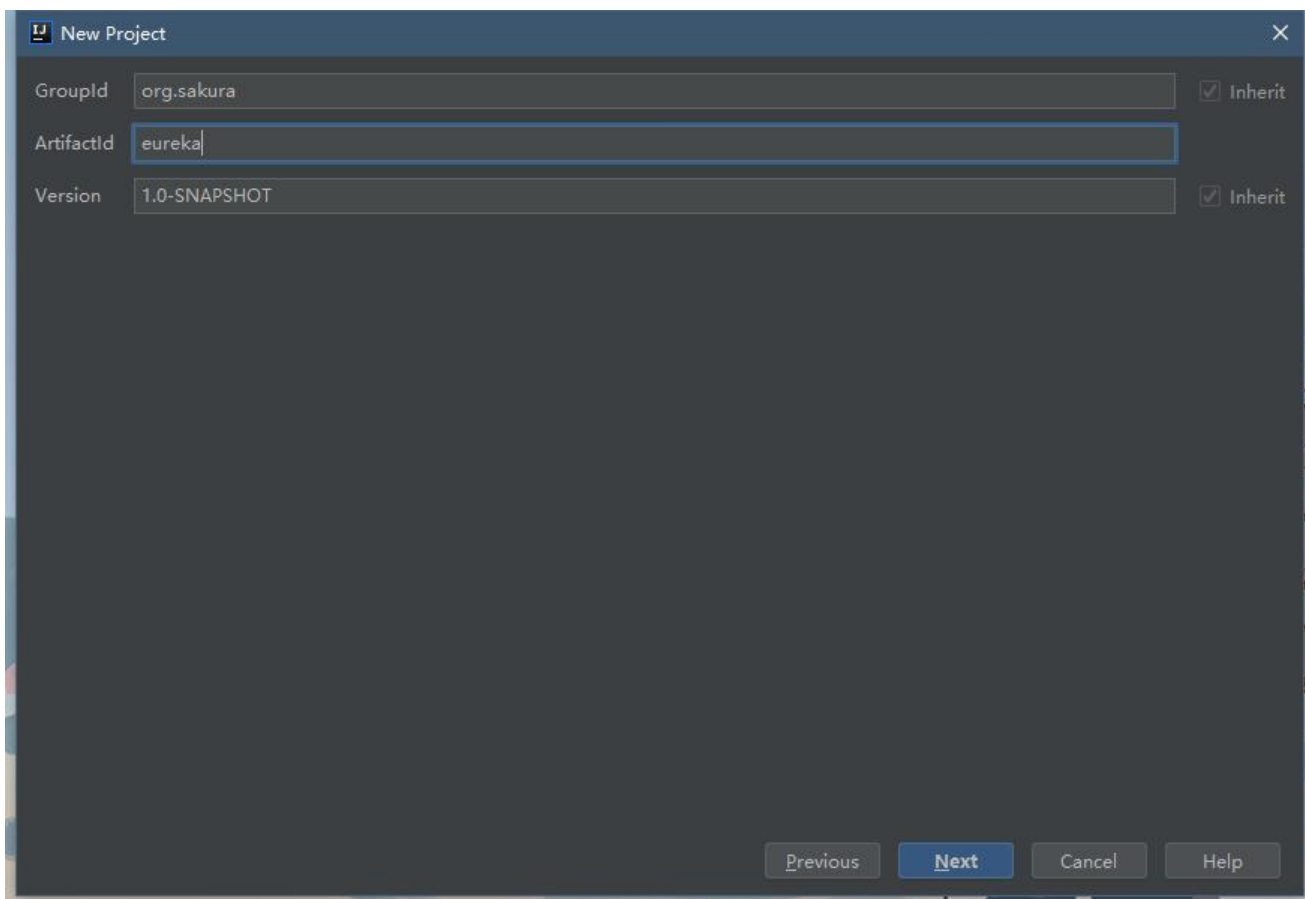
## 服务的注册与发现

# 一、新建主项目

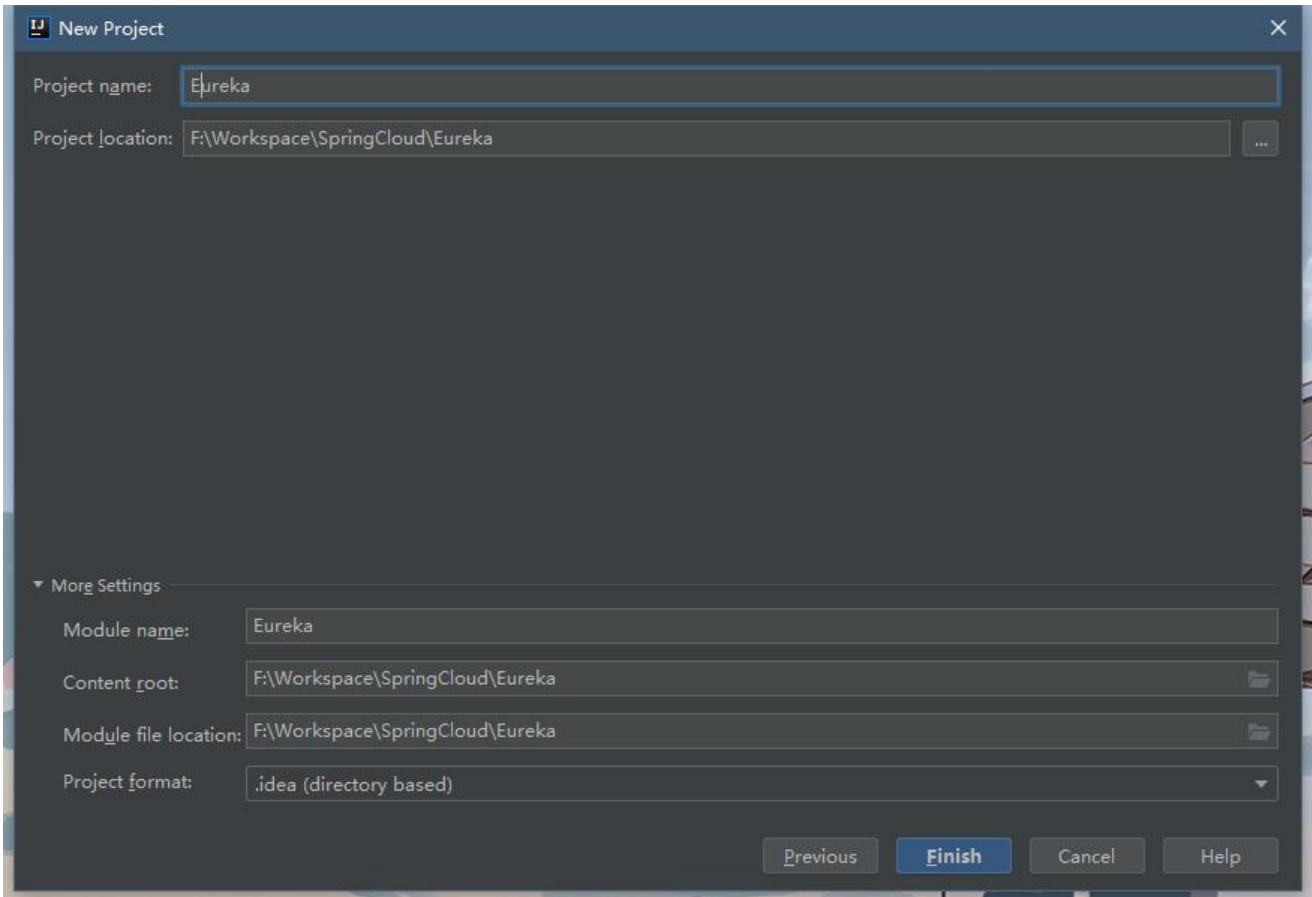
## 1. 选择Maven，点击Next



## 2. 填写相关信息，点击Next



### 3. 确定信息无误后，点击Finish



### 4. 将 `src` 文件夹删除 (如果有的话) 打开 `pom.xml` 文件，添加如下代码。

```
<!--必须指定该父模块，不然后面子模块启动会报错，很麻烦-->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.7.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>

<!--父模块类型必须为pom-->
<packaging>pom</packaging>

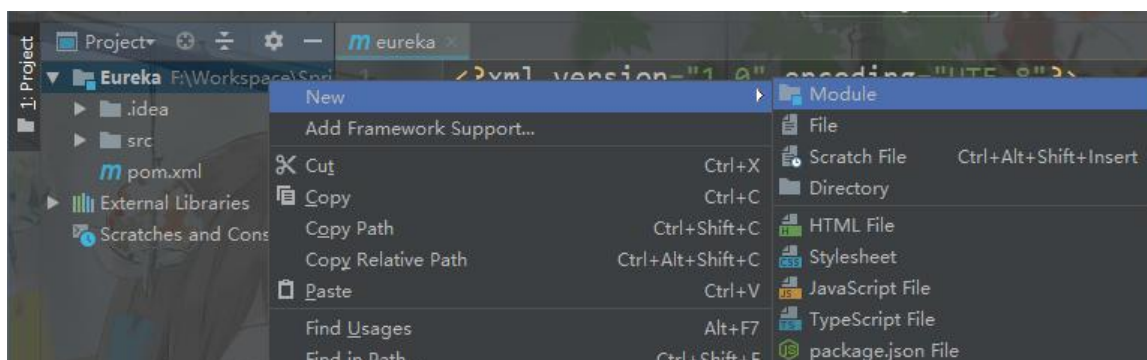
<!--包含子模块-->
<modules>

</modules>

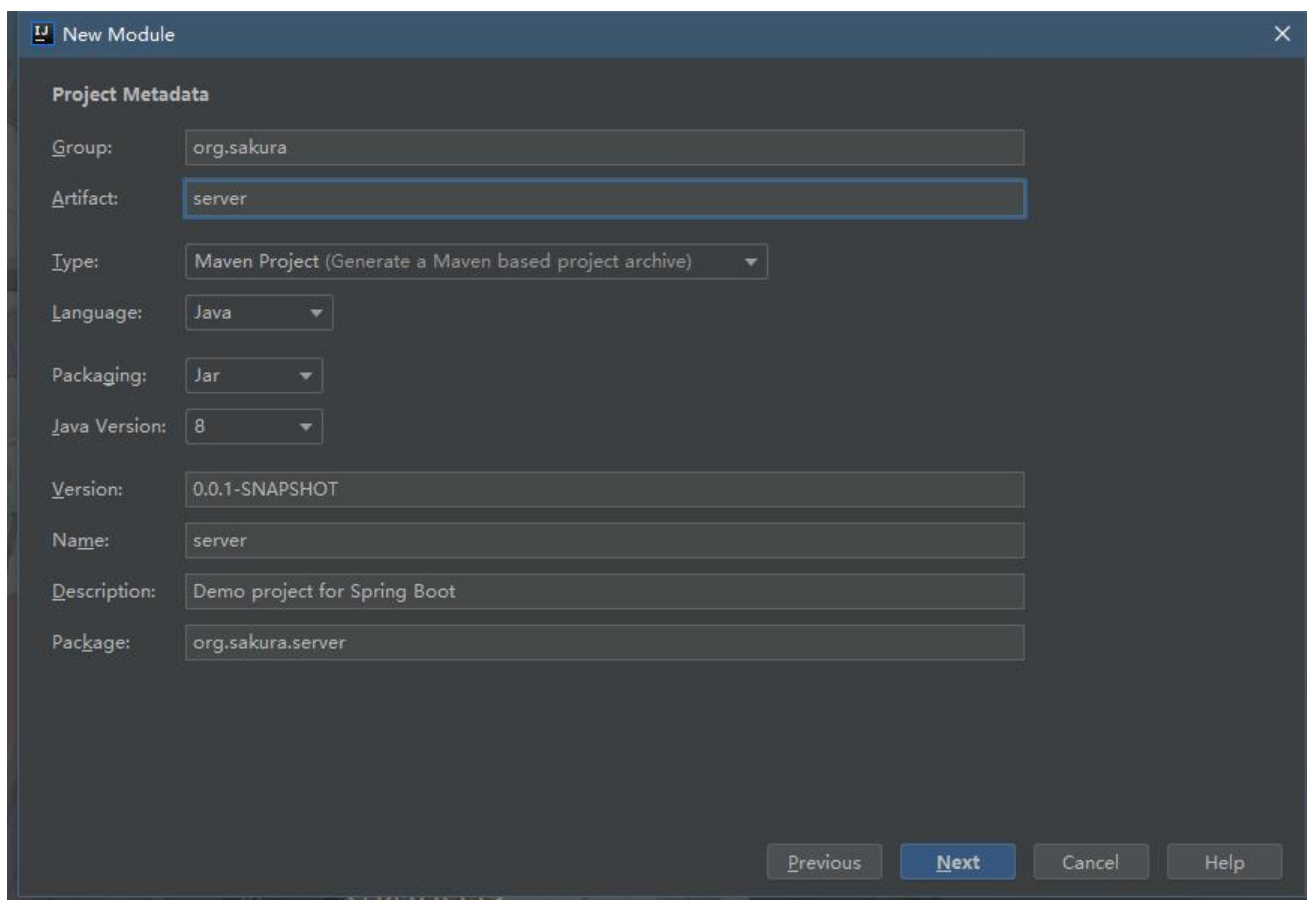
<!--在父模块添加web依赖，子模块可继承该依赖-->
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

## 二、新建Eureka Server

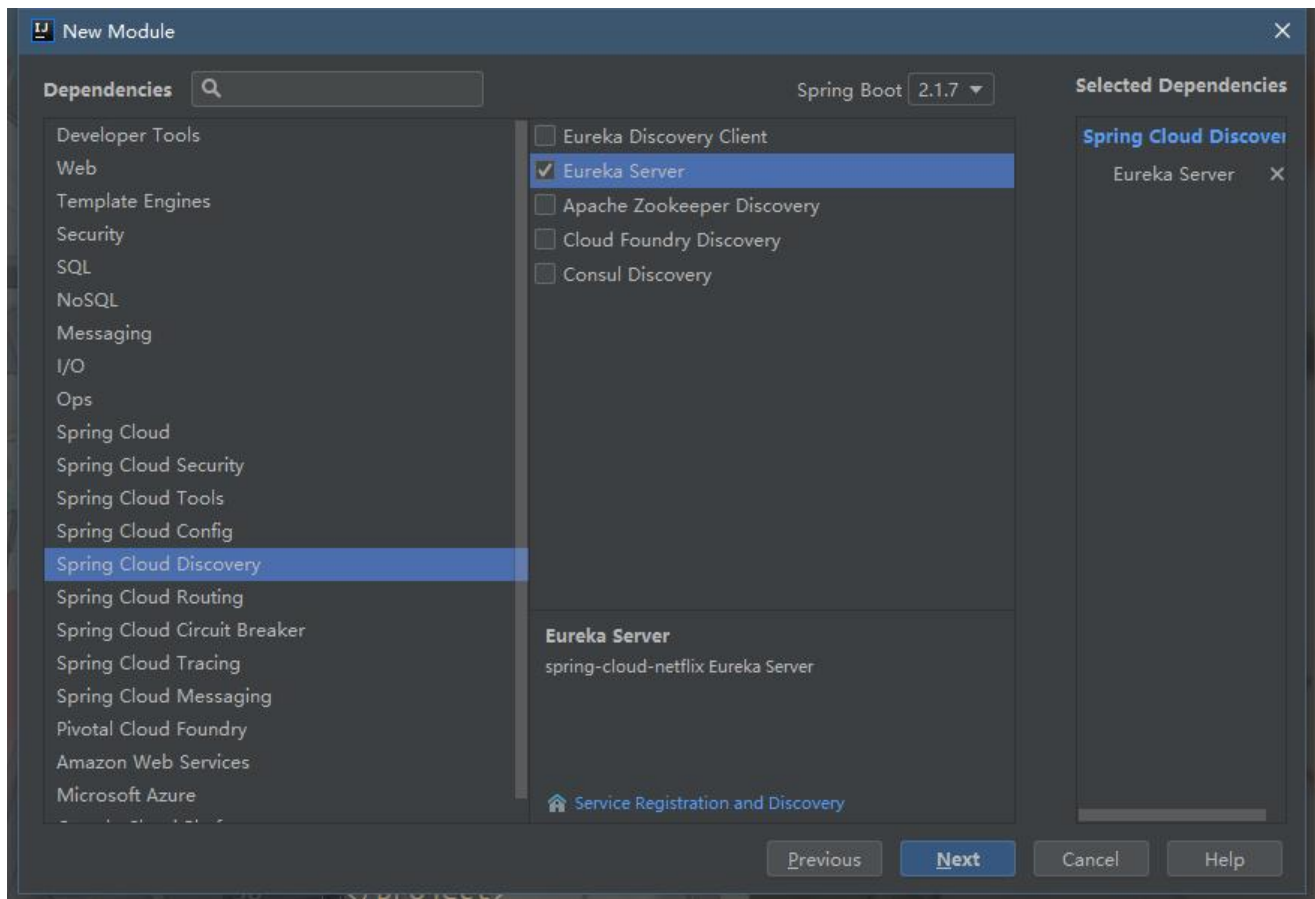
1. 在项目上右键新建Module



2. 选择Spring Initializr, 点击Next, 填写相关信息, Next



3. 选择导入Eureka Server 依赖, Next, 确认信息, 点击Finish



4. 打开Server模块的 `pom.xml` 文件，修改 `<parent>` 标签

```
<parent>
  <groupId>org.sakura</groupId>
  <artifactId>eureka</artifactId>
  <version>1.0-SNAPSHOT</version>
</parent>
```

5. 打开主模块的 `pom.xml` 文件，在 `<modules>` 标签添加相应的子模块

```
<!--包含子模块-->
<modules>
  <module>Server</module>
</modules>
```

6. 将Server模块中的 `application.properties` 重命名为 `application.yml`，并添加如下信息，也可直接使用 `properties` 文件类型（需修改如下代码）

```
# Eureka 服务注册与发现的组件
server:
  port: 8080
```

```
spring:
  application:
    #服务名，很重要
    name: server
```

```
eureka:
  instance:
    hostname: localhost
```

```

#将prefer-ip-address设为开启时，将默认显示服务的地址，而非主机名
# prefer-ip-address: true #以IP地址注册到服务中心，相互注册使用IP地址
# prefer-ip: 127.0.0.1 #显式设置服务的地址

client:
# 下面两个 false 表明自己是 server，而非 client
register-with-eureka: false # 不要使用 eureka 服务进行注册，即在管理界面不可见
fetch-registry: false # 不要在本机缓存注册表信息
service-url:
  defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
# defaultZone: http://127.0.0.1:${server.port}/eureka/

server:
#开启自我保护模式
enable-self-preservation: false
#清理无效节点,默认60*1000毫秒,即60秒
eviction-interval-timer-in-ms: 5000

```

7. 修改Server模块的启动类，添加 `@EnableEurekaServer`注解即可

```

@EnableEurekaServer
@SpringBootApplication
public class ServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServerApplication.class, args);
    }

}

```

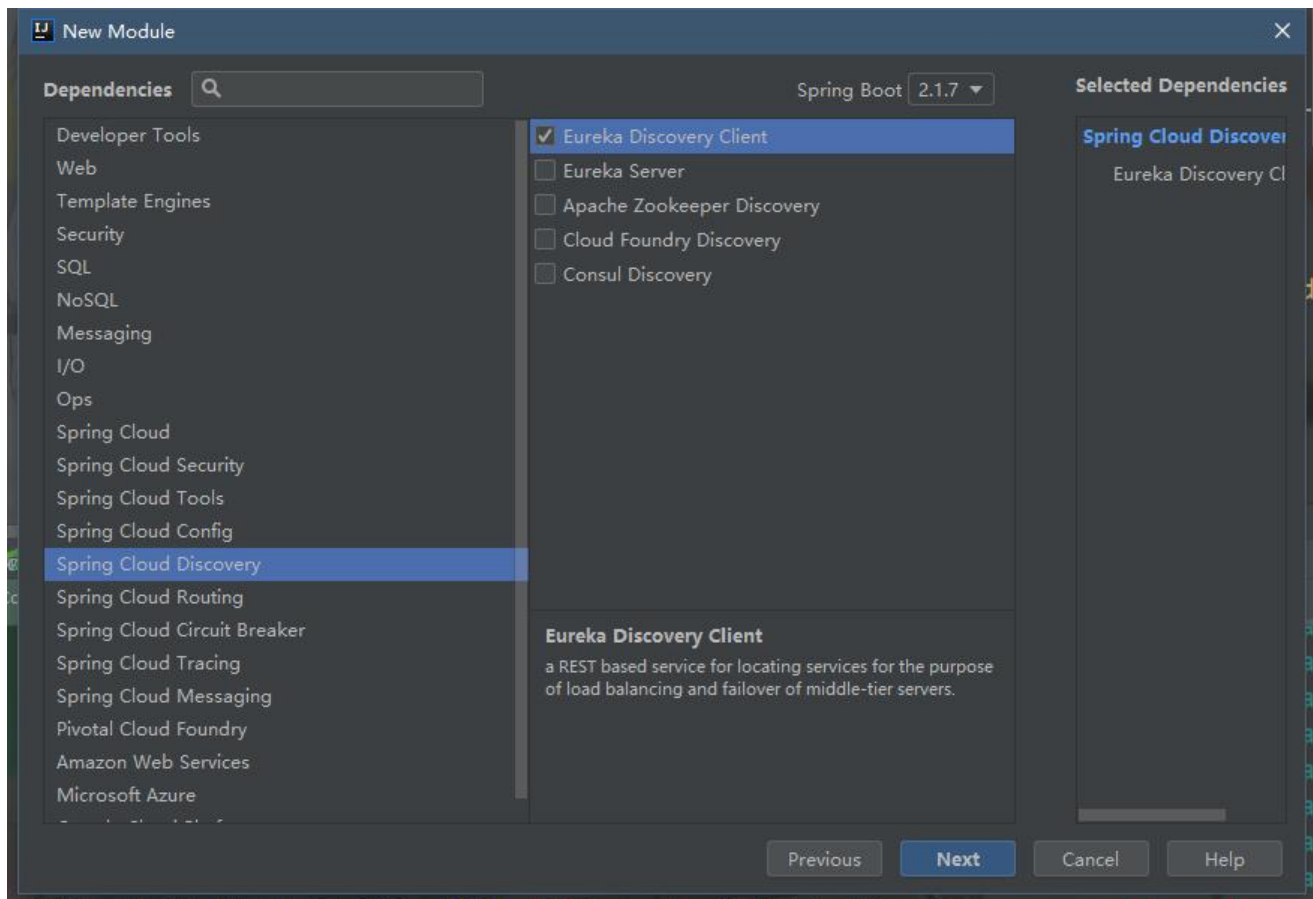
8. 启动Server服务，打开 <http://localhost:8080>

The screenshot shows the Spring Eureka web interface. At the top, there's a navigation bar with 'HOME' and 'LAST 1000 SINCE STARTUP'. Below this is the 'System Status' section, which contains two tables. The first table shows 'Environment: test' and 'Data center: default'. The second table shows 'Current time: 2019-08-27T20:31:12 +0800', 'Uptime: 00:00', 'Lease expiration enabled: true', 'Renews threshold: 1', and 'Renews (last min): 0'. Below these tables, a red warning message states: 'THE SELF PRESERVATION MODE IS TURNED OFF. THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.' Under the 'DS Replicas' section, it says 'Instances currently registered with Eureka'. A table below this shows 'Application', 'AMIs', 'Availability Zones', and 'Status'. The table is empty, with the text 'No instances available' and '当前无服务实例' (No service instances currently available) displayed. At the bottom, the 'General Info' section shows a table with 'Name' and 'Value' columns, containing 'total-avail-memory: 314mb', 'environment: test', and 'num-of-cpus: 8'.

## 三、新建Eureka Client

1. 新建Client模块，前两步和之前一样，只有导入依赖那里不一样





2. 修改子父模块的 `pom.xml`，与Server模块一样

3. 修改 `application.yml`

```
server:
  port: 8081
```

```
spring:
  application:
    # 服务名，很重要
    name: client
```

```
eureka:
  instance:
    hostname: localhost
    #以IP地址注册到服务中心，相互注册使用IP地址
    # prefer-ip-address: true
```

```
client:
  service-url:
    #服务注册地址
    defaultZone: http://${eureka.instance.hostname}:8080/eureka
```

4. 修改启动类，添加 `@EnableEurekaClient`注解（`@EnableDiscoveryClient`或不加都可以）

```
@EnableEurekaClient
@SpringBootApplication
public class Client1Application {
```



```

public static void main(String[] args) {
    SpringApplication.run(Client1Application.class, args);
}
}

```

5. 启动Client服务（启动Client前，需保证Server正在运行，不然会报错），打开刚才的链接

The screenshot shows the Spring Eureka dashboard. At the top, there's a navigation bar with 'HOME' and 'LAST 1000 SINCE STARTUP'. Below this, the 'System Status' section displays two tables. The left table shows 'Environment: test' and 'Data center: default'. The right table shows 'Current time: 2019-08-27T21:30:21 +0800', 'Uptime: 00:59', 'Lease expiration enabled: true', 'Renews threshold: 0', and 'Renews (last min): 2'. A red warning message states: 'THE SELF PRESERVATION MODE IS TURNED OFF. THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.' Below this, the 'DS Replicas' section shows 'Instances currently registered with Eureka'. A table lists one instance: 'CLIENT' with 'n/a (1)' AMIs and '(1)' Availability Zones, with a status of 'UP (1) - localhost:client:8081'. A red box highlights this instance. The 'General Info' section at the bottom shows 'Name' and 'Value' for 'total-avail-memory: 314mb' and 'environment: test'.

可以看到，Client服务已经注册到服务中心了。这里可能有小伙伴会发现点击这个服务的链接是会出现04的，这是因为项目没有使用到Actuator。

## 服务的调用

SpringCloud有两种服务调用的方式

- Ribbon
- Feign

### 一、Ribbon

SpringCloud 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。乎存在于每一个SpringCloud构建的微服务和基础设施中。因为微服务间的调用，API网关的请求转等内容，实际上都是通过Ribbon来实现的。

1. 再创建一个Client2服务，配置文件中除端口与之前的Client不同外，其它都一致，**服务名也一样**这是为了实现负载均衡。

```

server:
  port: 8082

```

```

spring:
  application:
    name: client

```

```
eureka:
  instance:
    hostname: localhost
    #以IP地址注册到服务中心，相互注册使用IP地址
    # prefer-ip-address: true

  client:
    service-url:
      #服务注册地址
      defaultZone: http://${eureka.instance.hostname}:8080/eureka
```

## 2. 为两个Client各添加一个API接口

@RestController

```
public class HelloController {
```

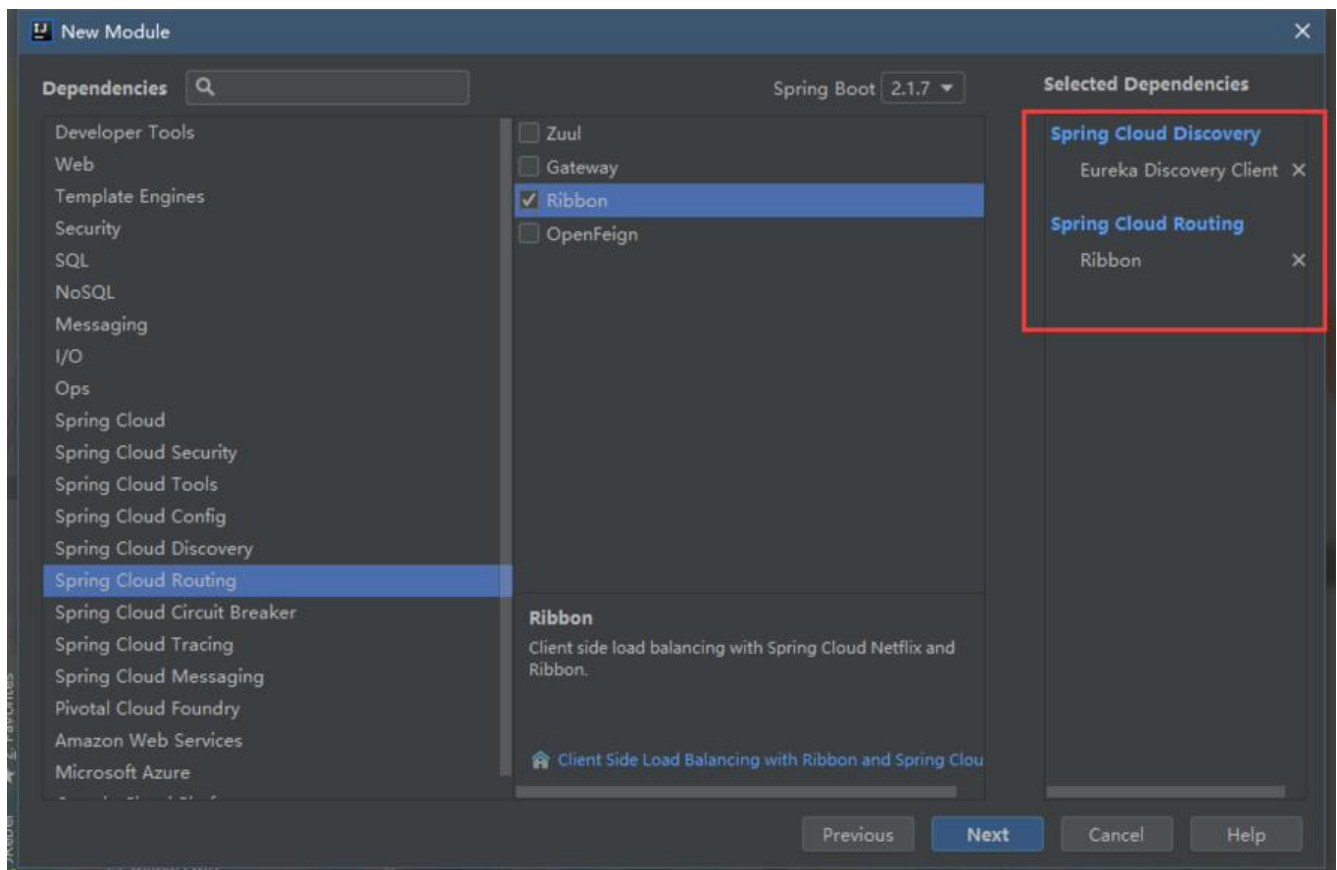
```
    @GetMapping("/hello")
    public String sayHello(@RequestParam(required = true,name = "name") String name){
        return "Hello " + name + ", 8081";
    }
}
```

@RestController

```
public class HelloController {
```

```
    @GetMapping("/hello")
    public String sayHello(@RequestParam(required = true,name = "name") String name){
        return "Hello " + name + ", 8082";
    }
}
```

## 3. 再新建一个Robbon模块，前面基本一样，依赖不同



#### 4. 修改配置文件和pom

```
server:
  port: 8083

spring:
  application:
    name: ribbon

eureka:
  instance:
    hostname: localhost
    #以IP地址注册到服务中心, 相互注册使用IP地址
    # prefer-ip-address: true

  client:
    service-url:
      #服务注册地址
      defaultZone: http://${eureka.instance.hostname}:8080/eureka
```

#### 5. 修改启动类, 需添加一个RestTemplate bean

```
@EnableDiscoveryClient
@SpringBootApplication

public class RibbonApplication {

    public static void main(String[] args) {
        SpringApplication.run(RibbonApplication.class, args);
    }

    /**
     * 负载均衡配置
     * @return
     */
    @Bean
    @LoadBalanced
    RestTemplate restTemplate(){
        return new RestTemplate();
    }
}
```

#### 6. 在Ribbon模块中加入一个Service和一个Controller

```
@Service
public class HelloService {

    @Autowired
    RestTemplate restTemplate;

    public String helloService(String name){
        return restTemplate.getForObject("http://client/hello?name="+name,String.class);
    }
}

@RestController
```

```

public class HelloController {

    @Autowired
    HelloService helloService;

    @GetMapping("/hello")
    public String hello(String name){
        return helloService.helloService(name);
    }
}

```

7. 依次启动Server模块和其它各模块

The screenshot shows the Spring Eureka dashboard. At the top, there's a navigation bar with 'HOME' and 'LAST 1000 SINCE STARTUP'. Below this, the 'System Status' section displays various metrics in a table:

Environment	test	Current time	2019-08-27T22:48:41 +0800
Data center	default	Uptime	00:18
		Lease expiration enabled	true
		Renews threshold	0
		Renews (last min)	6

Below the status table, a red warning message states: 'THE SELF PRESERVATION MODE IS TURNED OFF. THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.'

The 'DS Replicas' section shows 'Instances currently registered with Eureka' in a table:

Application	AMIs	Availability Zones	Status
CLIENT	n/a (2)	(2)	UP (2) - localhost:client:8081, localhost:client:8082
RIBBON	n/a (1)	(1)	UP (1) - localhost:ribbon:8083

The 'General Info' section at the bottom shows a table with system details:

Name	Value
total-avail-memory	222mb
environment	test

可以看到，两个Client和一个Ribbon都已经注册上去了

8. 打开 <http://localhost:8083/hello?name=sakura>

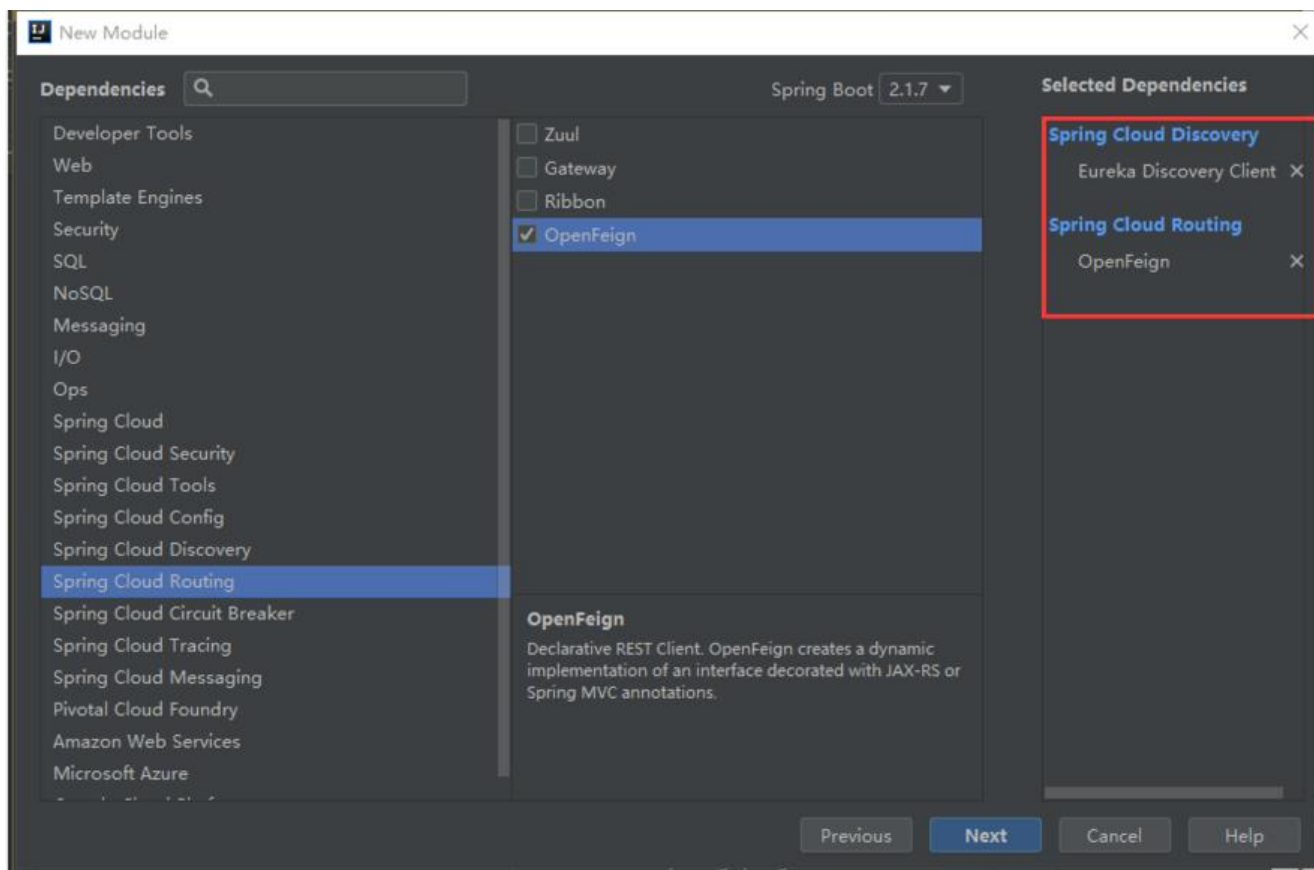
The screenshot shows a web browser window with the address bar set to 'localhost:8083/hello?name=sakura'. The page content displays 'Hello sakura, 8081'. A second browser window is overlaid on top, showing the same address bar but with the page content displaying 'Hello sakura, 8082'. This demonstrates that the service being called (8082) is different from the one initially shown (8081), indicating successful load balancing.

每次刷新调用的服务都不同，证明客户端负载均衡成功了

## 二、Feign

Feign是基于Ribbon实现的工具，采用基于接口的注解

1. 新建Feign模块，引入依赖



## 2. 修改配置文件和pom

server:  
port: 8084

spring:  
application:  
name: feign

eureka:  
instance:  
hostname: localhost  
#以IP地址注册到服务中心，相互注册使用IP地址  
# prefer-ip-address: true

client:  
service-url:  
#服务注册地址  
defaultZone: http://\${eureka.instance.hostname}:8080/eureka

## 3. 修改启动类，添加 `@EnableEurekaClient`和`@EnableFeignClients`

```
@EnableFeignClients
@EnableEurekaClient
@SpringBootApplication
public class FeignApplication {

    public static void main(String[] args) {
        SpringApplication.run(FeignApplication.class, args);
    }
}
```

```
}
```

#### 4. 创建一个Service接口

```
@FeignClient(value = "client") //value值为需要调用的服务名
public interface IFeignService {

    @GetMapping("/hello") //这里的地址为需要调用的服务里相应的接口地址
    String hello(@RequestParam(value = "name")String name);
}
```

5. 使用上面的接口（声明完上面Feign接口后，其他Spring管理的类，如Service、Controller都可以接注入使用,IDEA可能会提示不能注入，可忽略）

```
@RestController
public class HelloController {

    @Autowired
    private IFeignService iFeignService;

    @GetMapping("/feign/hello")
    public String sayHello(String name){
        return iFeignService.hello(name);
    }
}
```

#### 6. 依次启动Server模块和Feign模块和Client模块

The screenshot shows the Spring Eureka dashboard. At the top, there's a navigation bar with 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content area is divided into several sections:

- System Status:** A table showing environment details (test, default) and system metrics (Current time: 2019-08-27T23:25:38 +0800, Uptime: 00:54, Lease expiration enabled: true, Renewal threshold: 0, Renewal (last min): 4).
- DS Replicas:** A section indicating that the self-preservation mode is turned off.
- Instances currently registered with Eureka:** A table with columns: Application, AMIs, Availability Zones, and Status. It lists two instances: CLIENT (n/a (2), (2), UP (2) - localhost:client:8081, localhost:client:8082) and FEIGN (n/a (1), (1), UP (1) - localhost:feign:8084).
- General Info:** A table showing system information (Name, Value) such as total-avail-memory (222mb).

#### 7. 访问 <http://localhost:8084/feign/hello?name=sakura>

The screenshot shows a web browser window with the address bar set to 'localhost:8084/feign/hello?name=sakura'. The page content displays 'Hello sakura, 8081' in the main area and 'Hello sakura, 8082' in a separate section, indicating a successful API call and response.

以上相对全面简洁的介绍了SpringCloud中服务的注册、发现与调用。代码已上传至[Github](#)