

东软睿道内部公开

文件编号: D000-

# Spring Cloud微服务架构

版本: 1.0.0

## 第3章 Eureka服务注册与发现

东软睿道教育信息技术有限公司  
(版权所有, 翻版必究)

Copyright © Neusoft Educational Information Technology Co., Ltd  
All Rights Reserved



# 本章教学目标

- ✓ 了解Eureka及服务注册与发现；
- ✓ 了解Eureka的自我保护模式；
- ✓ 了解Actuator与注册微服务信息完善；
- ✓ 掌握Eureka Server的编写；
- ✓ 掌握将微服务注册到Eureka Server上；
- ✓ 掌握消费者从Eureka Server中发现服务；
- ✓ 掌握Eureka集群配置；

# 本章教学内容

节	知识点	掌握程度	难易程度	教学形式	对应在线微课
Eureka简介	服务注册与发现	了解		线下	
	Eureka是什么	了解		线下	
	Eureka架构图	理解		线下	
服务注册与发现实战	流程图	理解		线下	
	编写Eureka Server	掌握		线下	
	将微服务注册到Eureka Server上	掌握		线下	
	Actuator与注册微服务信息完善	了解		线下	
	Eureka的自我保护模式	了解		线下	
	消费者从Eureka Server中发现服务	掌握	难	线下	
Eureka集群配置	原理说明	理解		线下	
	集群配置	掌握		线下	
	微服务发布到集群中	掌握		线下	

# CONTENTS 目录

01

Eureka简介

02

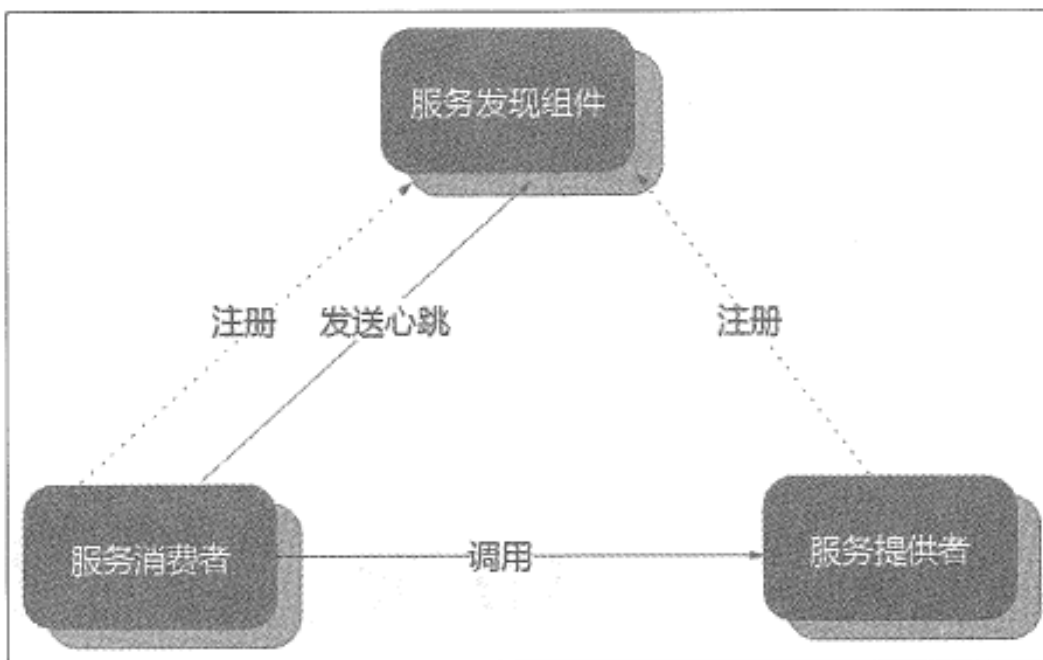
服务注册与发现实战

03

Eureka集群配置

# 服务注册与发现

- ❖ 微服务架构的缺点中最主要的就是由于微服务数量众多导致维护成本巨大，服务注册与发现为解决此问题而产生的。
- ❖ 服务发现架构图



# 服务注册与发现

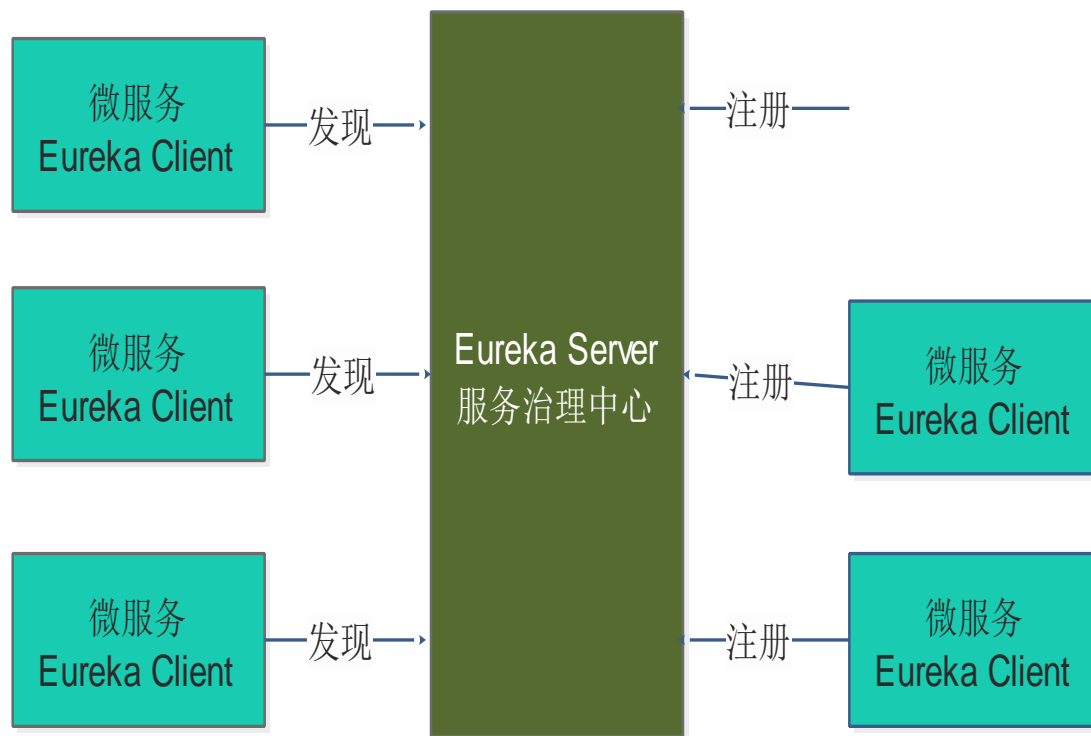
- ❖ Spring Cloud提供了多种注册中心的支持，如：Eureka、ZooKeeper等。推荐使用Eureka。
- ❖ Eureka服务注册与发现的作用是让维护人员从人工维护中解放出来，由服务自维护，微服务作为服务提供方主动向Eureka服务发现与注册中心注册，服务的消费方通过Eureka服务注册与发现中心查询需要的服务并进行调用。

# Eureka是什么

- ❖ Eureka是Netflix开发的服务发现框架，本身是一个基于REST的服务，主要用于定位服务，以达到负载均衡和中间层服务故障转移的目的。Spring Cloud将它集成在其子项目spring-cloud-netflix中，以实现Spring Cloud的服务发现功能。
- ❖ Eureka项目在github上托管：<https://github.com/Netflix/eureka>
- ❖ Eureka包含两个组件：Eureka Server和Eureka Client。
  - ▶ Eureka Server提供服务注册服务，各个节点启动后，会在Eureka Server中进行注册，这样Eureka Server中的服务注册表中将会存储所有可用服务节点的信息，服务节点的信息可以在界面中直观的看到。
  - ▶ Eureka Client是一个java客户端，用于简化与Eureka Server的交互，客户端同时也具备一个内置的、使用轮询(round-robin)负载算法的负载均衡器。

# Eureka是什么

## ❖ Eureka Server与Eureka Client的关系





# Eureka是什么

- ❖ 在应用启动后，Eureka Client将会向Eureka Server发送心跳，默认周期为30秒，如果Eureka Server在多个心跳周期内没有接收到某个节点的心跳，Eureka Server将会从服务注册表中把这个服务节点移除(默认90秒)。
- ❖ Eureka Server之间通过复制的方式完成数据的同步，Eureka还提供了客户端缓存机制，即使所有的Eureka Server都挂掉，客户端依然可以利用缓存中的信息消费其他服务的API。综上，Eureka通过心跳检查、客户端缓存等机制，确保了系统的高可用性、灵活性和可伸缩性。

# CONTENTS

## 目录

01

Eureka简介

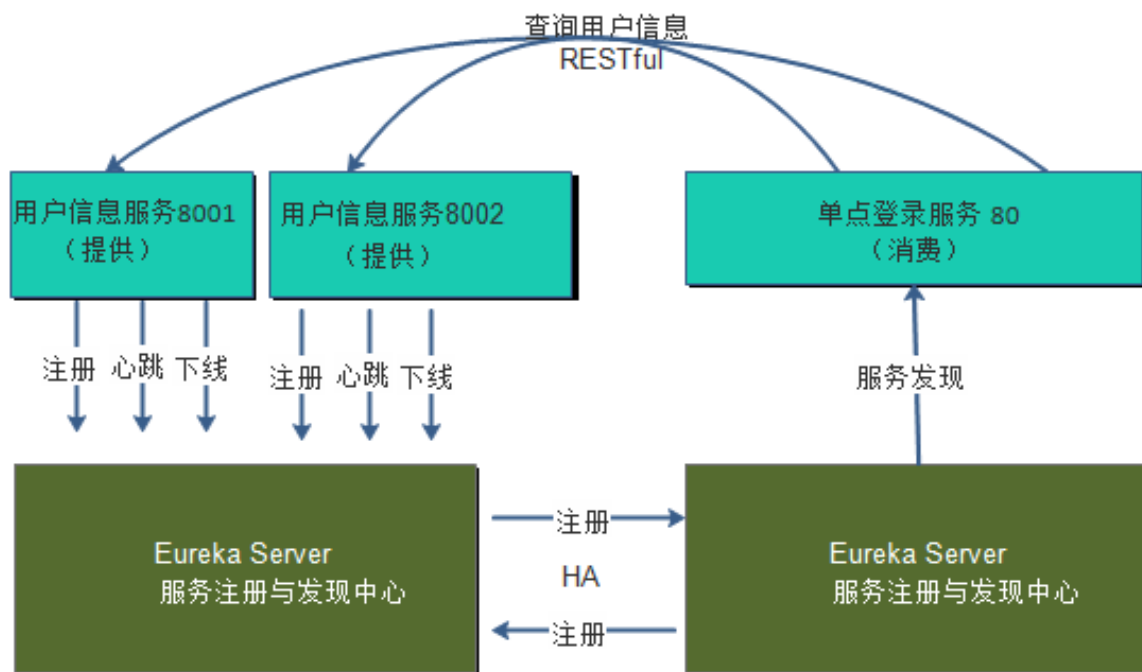
02

服务注册与发现实战

03

Eureka集群配置

# 流程图

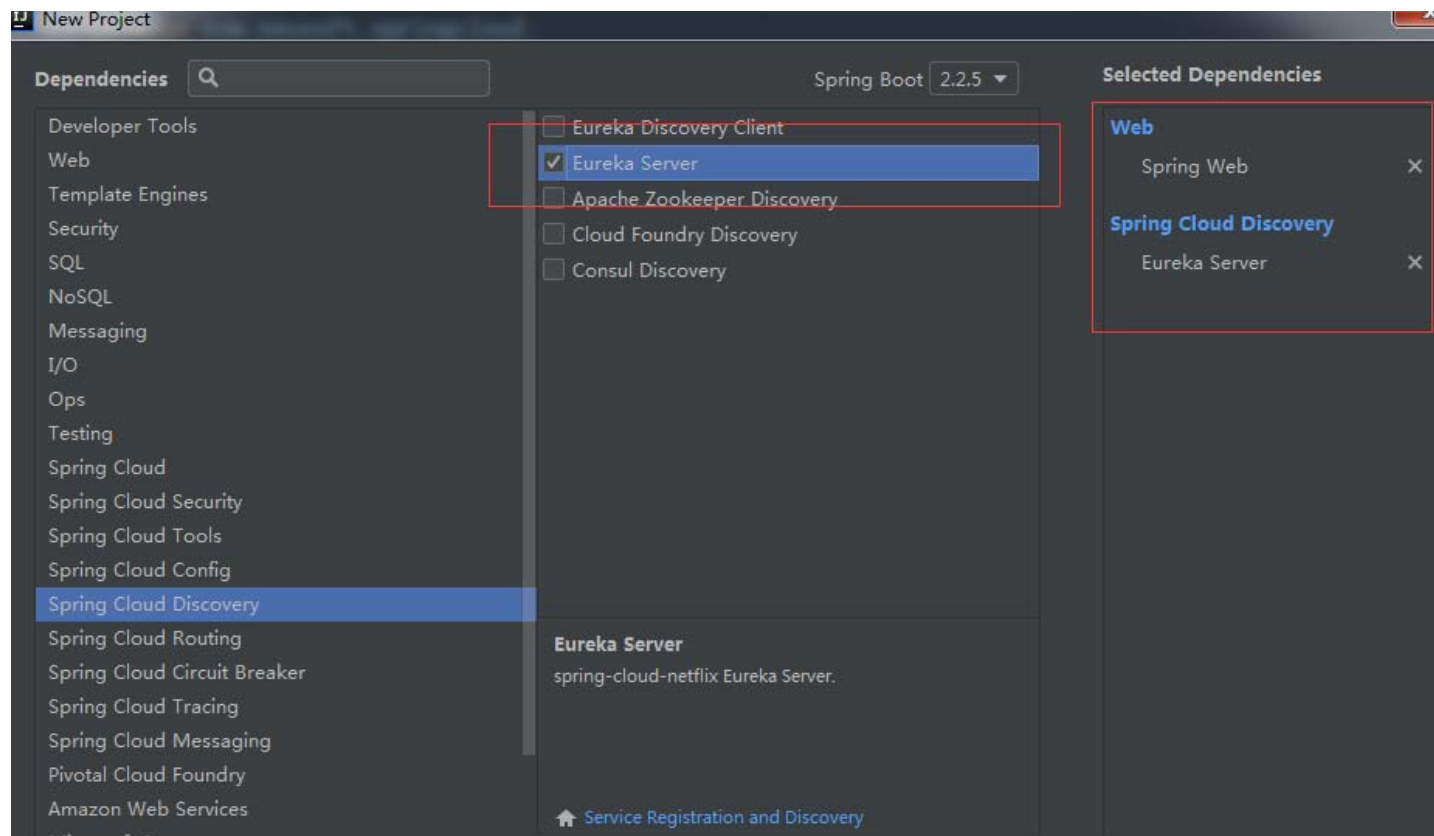


# 编写Eureka Server

- ❖ 新建eureka工程为Eureka 服务注册中心
  - ▶ 新建eureka工程
  - ▶ 编写全局配置文件
  - ▶ 编写主启动类
  - ▶ 测试

# 编写Eureka Server

## ❖ 1、新建eureka工程



# 编写Eureka Server

## ❖ 3、编写全局配置文件

```
spring:
  application:
    name: regist-center #注册中心的名字
server:
  port: 7001 #服务端口
#注册中心相关配置
eureka:
  # 配置关闭自我保护, 并按需配置Eureka Server清理无效节点的时间间隔 (5000ms) 。
  server:
    enable-self-preservation: false
    eviction-interval-timer-in-ms: 5000
  client:
    register-with-eureka: false # 不将自己注册到注册中心
    fetch-registry: false # 因为自己是注册中心, 因此不用检索服务信息
    # 注册中心的地址
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
instance:
  prefer-ip-address: true #用机器名标识服务名
  hostname: localhost
```

# 编写Eureka Server

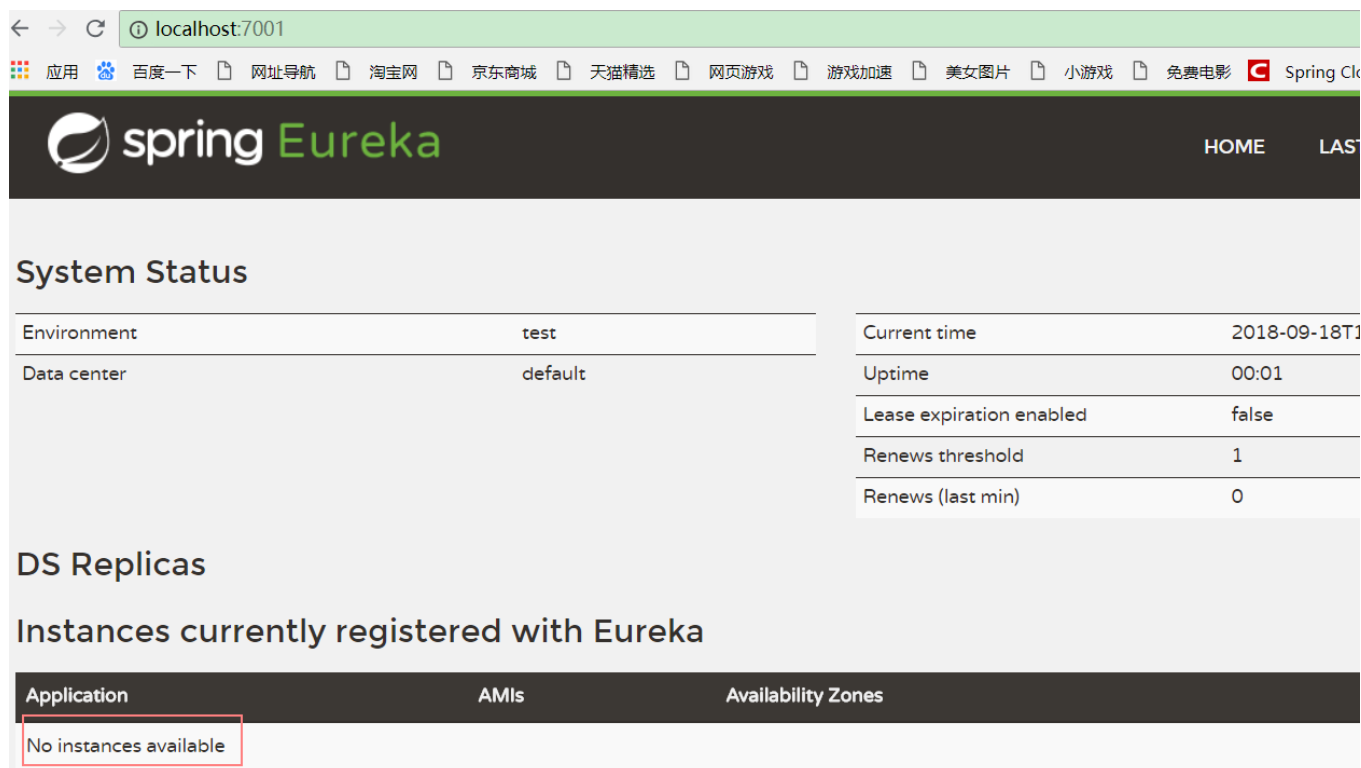
## ❖ 4、编写主启动类

@SpringBootApplication

@EnableEurekaServer

# 编写Eureka Server

- ❖ 5、测试
  - ▶ 启动主启动类
  - ▶ 浏览器 <http://localhost:7001/>
- ❖ No application available 没有服务被发现，因为还没有注册服务



The screenshot displays the Spring Eureka Server web interface. The browser address bar shows `localhost:7001`. The page header includes the Spring Eureka logo and navigation links for HOME and LAST. The main content area is divided into two sections: System Status and DS Replicas.

**System Status**

Environment	test	Current time	2018-09-18T10:00:01.000Z
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

**DS Replicas**

Instances currently registered with Eureka

Application	AMIs	Availability Zones
No instances available		



# 将微服务注册到Eureka Server上

- ❖ Eureka将已有的用户微服务注册进Eureka服务中心
- ❖ 新建provider模块（参考第二章 编写微服务提供者模块）
  - ▶ 修改全局配置文件
  - ▶ 修改主启动类
  - ▶ 测试

# 将微服务注册到Eureka Server上

```
❖ application.yml
spring:
  application:
    name: provider
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
      jdbc:mysql://localhost:3306/mybatis?useUnicode=true&characterEncoding=utf-8&useSSL=false&allowMultiQueries=true
    username: root
    password: 123456
  server:
    port: 8001
  eureka:
    client:
      service-url:
        defaultZone: http://localhost:7001/eureka/
    instance:
      prefer-ip-address: false
```

# 将微服务注册到Eureka Server上

## ❖ 测试

- ▶ 先启动EurekaServer
- ▶ 再启动8001主启动类
- ▶ 浏览器 <http://localhost:7001/>
- ▶ 看到有服务注册进来

The screenshot shows the Eureka Server web interface at <http://localhost:7001/>. The browser's address bar and tabs are visible at the top. The main content area is divided into two sections: 'System Status' and 'DS Replicas'.

**System Status**

Environment	test	Current time	2018-09-18T17:05:03 +08:00
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	3
		Renews (last min)	0

**DS Replicas**

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA-CLIENT-USER	n/a (1)	(1)	UP (1) - localhost:eureka-client-user:8001

Red annotations highlight key information: a yellow box labeled '应用名' (Application Name) points to the 'EUREKA-CLIENT-USER' application name in the table. Red boxes also highlight the application name 'EUREKA-CLIENT-USER' and the status 'UP (1) - localhost:eureka-client-user:8001'.

# Actuator与注册微服务信息完善

- ❖ 当前问题：服务信息中含有主机名称 actuator 英['æktju:etə] 马达

renews (last min) U

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA-CLIENT-USER	n/a (1)	(1)	UP (1) - localhost:eureka-client-user:8001

- ❖ 主机名称:服务名称修改

- ▶ 修改provider全局配置文件

eureka:

client:

service-url:

defaultZone: http://localhost:7001/eureka/

instance:

instance-id: provider1

- ▶ 修改后测试

## Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
PROVIDER	n/a (1)	(1)	UP (1) - provider1

# Actuator与注册微服务信息完善

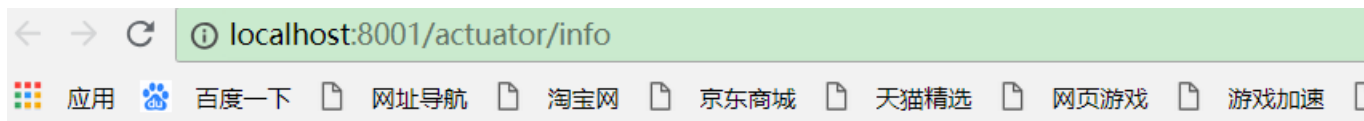
❖ 当前问题：点击status项后，显示错误

renews (last min) U

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA-CLIENT-USER	n/a (1)	(1)	UP (1) - <a href="http://localhost:eureka-client-user:8001">localhost:eureka-client-user:8001</a>

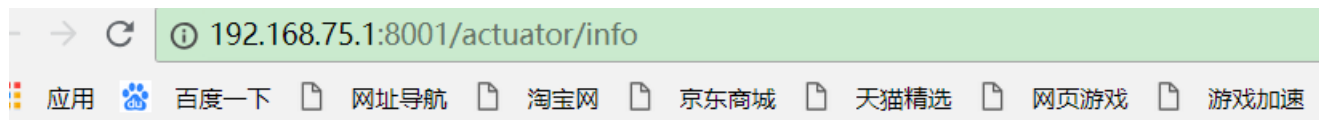


# Actuator与注册微服务信息完善

- ❖ 将localhost修改为IP地址
  - ▶ 修改provider全局配置文件

#访问路径可以显示IP地址

```
eureka.instance.prefer-ip-address: true
```



## Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Sep 19 09:34:51 CST 2018

There was an unexpected error (type=Not Found, status=404).

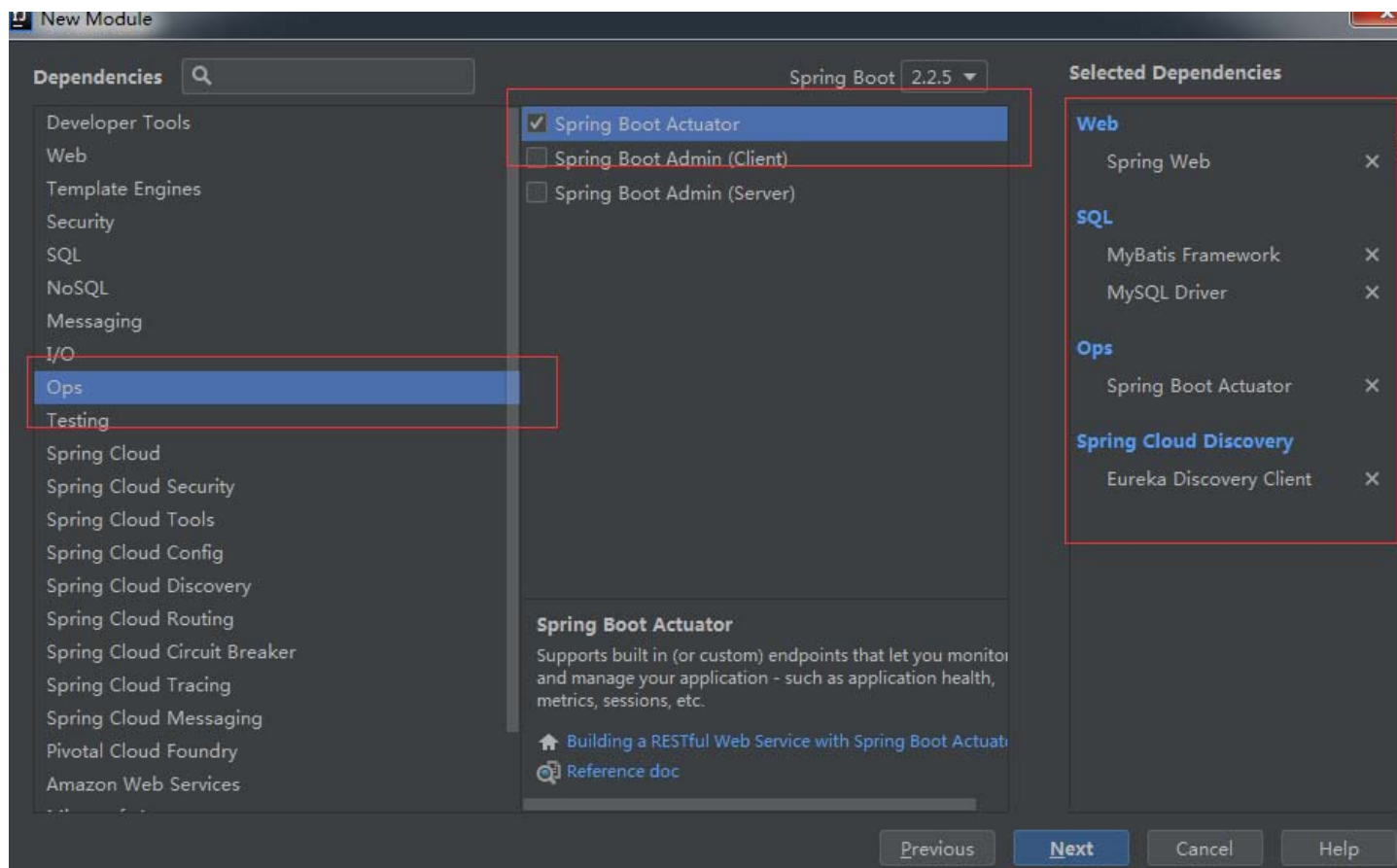
No message available

# Actuator与注册微服务信息完善

- ❖ 错误信息的修正
- ❖ 微服务info内容详细信息
  - ▶ 1、新建模块provider2
  - ▶ 2、修改m全局配置文件

# Actuator与注册微服务信息完善

## ❖ 1、新建provider2





# Actuator与注册微服务信息完善

❖ 2、修改mscloud-provider-user-8001 全局配置文件

`info.app.name: provider`

`info.company.name: neusoft`

.....

`info.author.name: xxx`

# Eureka的自我保护模式

- ❖ 演示案例
  - ▶ 故障现象
  - ▶ 导致原因



# Eureka的自我保护模式

## ❖ 故障现象

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

### DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA-CLIENT-USER	n/a (1)	(1)	UP (1) - <a href="#">mscloud-user8001</a>

## ❖ 故障原因

- ▶ 某时刻某一个微服务不可用了，Eureka不会立刻清理，依旧会对该微服务的信息进行保存

# Eureka的自我保护模式

- ❖ 默认情况下，如果Eureka Server在一定时间内没有接收到某个微服务实例的心跳，Eureka Server将会注销该实例（默认90秒）。但是当网络分区故障发生时，微服务与Eureka Server之间无法正常通信，以上行为可能变得非常危险了——因为微服务本身其实是健康的，此时本不应该注销这个微服务。
- ❖ Eureka通过“自我保护模式”来解决这个问题
  - ▶ 当Eureka Server节点在短时间内丢失过多客户端时（可能发生了网络分区故障），那么这个节点就会进入自我保护模式。一旦进入该模式，Eureka Server就会保护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）。当网络故障恢复后，该Eureka Server节点会自动退出自我保护模式。
  - ▶ 在Spring Cloud中，可以使用`eureka.server.enable-self-preservation = false` 禁用自我保护模式。

# Eureka的自我保护模式

- ❖ 修改Eureka server的全局配置文件，可以禁用，**但是不建议**
  - ▶ 修改eureka全局配置文件

```
#禁用自我保护模式  
eureka.server.enable-self-preservation=false
```

THE SELF PRESERVATION MODE IS TURNED OFF.THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.

DS Replicas

Instances currently registered with Eureka

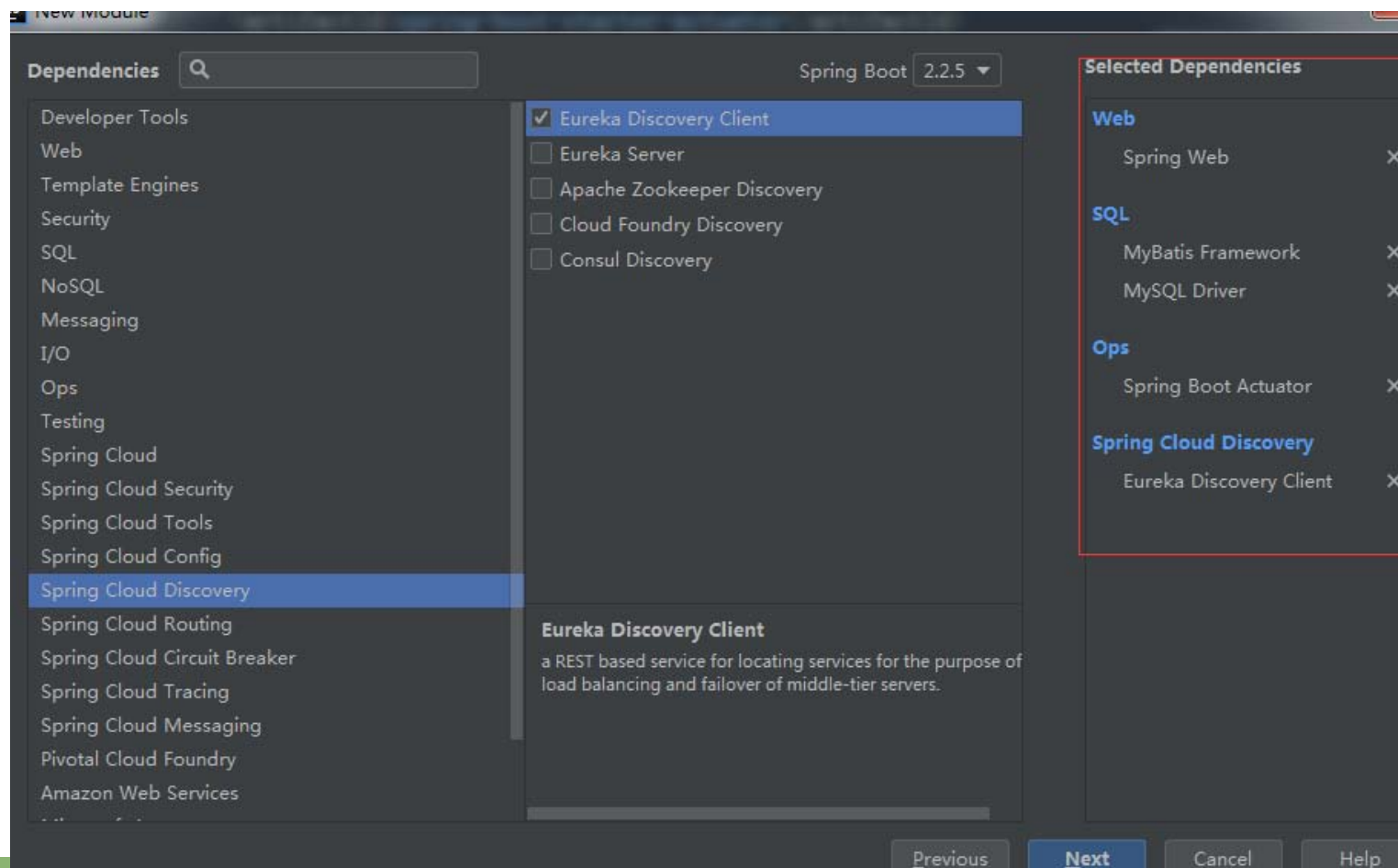
Application	AMIs	Availability Zones	Status
EUREKA-CLIENT-USER	n/a (1)	(1)	UP (1) - msccloud-user8001

# 消费者从Eureka Server中发现服务

- ❖ 对于注册进Eureka Server里面的微服务，可以通过服务发现来获得该服务的信息
  - ▶ 新建provider3模块（参考provider2）
  - ▶ 修改主启动类
  - ▶ 自测

# 消费者从Eureka Server中发现服务

- ▶ 新建provider3模块（参考provider2）



# 消费者从Eureka Server中发现服务

- ❖ 在provider3的controller中添加discovery方法

@Autowired

private DiscoveryClient client;

@RequestMapping(value = "/discovery", method = RequestMethod. *GET*)

public Object discovery()

{

    List<String> list = client.getServices();

    System. *out*.println("\*\*\*\*\*" + list);

    List<ServiceInstance> srvList =

    client.getInstances("provider");

    for (ServiceInstance element : srvList) {

        System. *out*.println(element.getServiceId() + "\t" +  
        element.getHost() + "\t" + element.getPort() + "\t"  
                + element.getUri());

    }

    return this.client;

}



# 消费者从Eureka Server中发现服务

- ▶ 在主启动类添加@EnableDiscoveryClient  
@SpringBootApplication  
@EnableEurekaClient  
@EnableDiscoveryClient

# 消费者从Eureka Server中发现服务

## ❖ 测试

- ▶ 启动8003的主启动类
- ▶ <http://localhost:8003/country/discovery>

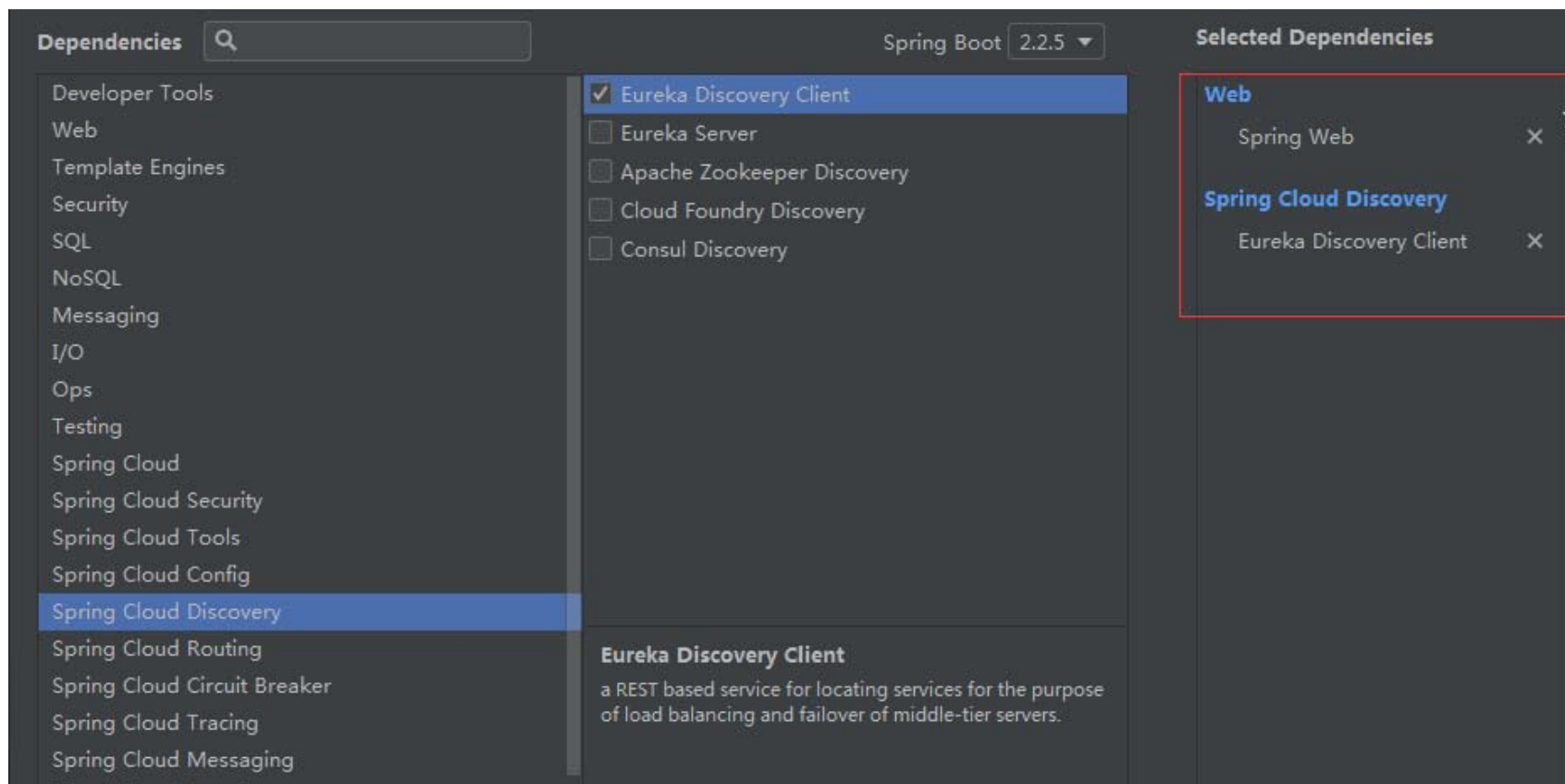
```
← → ↻ localhost:8003/country/discovery  
{"discoveryClients":[{"services":["provider"],"order":0}, {"services":["provider"],"order":0}]}
```

## ▶ 控制台输出

```
*****[provider]  
PROVIDER 192.168.0.103 8001 http://192.168.0.103:8001  
PROVIDER 192.168.0.103 8003 http://192.168.0.103:8003  
PROVIDER 192.168.0.103 8002 http://192.168.0.103:8002
```

# 消费者从Eureka Server中发现服务

## ❖ 新建consumer模块



# 消费者从Eureka Server中发现服务

❖ 修改全局配置文件

```
server:
```

```
  port: 80
```

```
eureka:
```

```
  client:
```

```
    service-url:
```

```
      defaultZone: http://localhost:7001/eureka/
```

```
instance:
```

```
  prefer-ip-address: true
```

# 消费者从Eureka Server中发现服务

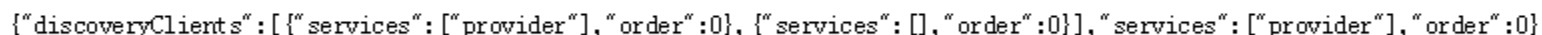
❖ consumer模块的controller里添加提供者调用方法

@RestController

```
public class ConsumerController {  
    private static final String REST_URL_PREFIX =  
    "http://localhost:8003";  
  
    /**  
     * 使用 使用restTemplate访问restful接口非常的简单。  
     */  
    @Autowired  
    private RestTemplate restTemplate;  
  
    // 测试@EnableDiscoveryClient, 消费端可以调用服务发现  
    @RequestMapping(value = "/consumer/discovery")  
    public Object discovery() {  
        return restTemplate.getForObject(REST_URL_PREFIX +  
        "/country/discovery", Object.class);  
    }  
}
```

▶ 启动服务消费者80主启动类

- ▶ `http://localhost/consumer/discovery`



# CONTENTS 目录

01

Eureka简介

02

服务注册与发现实战

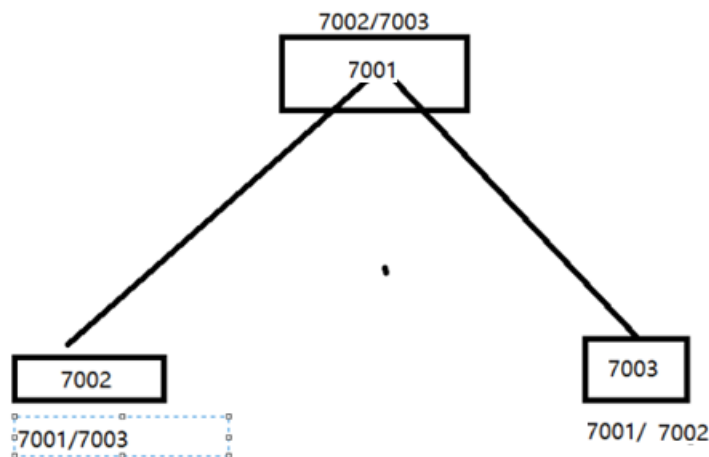
03

Eureka集群配置

# Eureka集群配置

## ❖ 原理说明

- ▶ Eureka服务是一个单点服务，在生产环境就会出现单点故障，为了确保Eureka服务的高可用，需要搭建Eureka服务的集群。
- ▶ 搭建Eureka集群非常简单，只要启动多个Eureka服务并且让这些服务之间彼此进行注册即可实现。



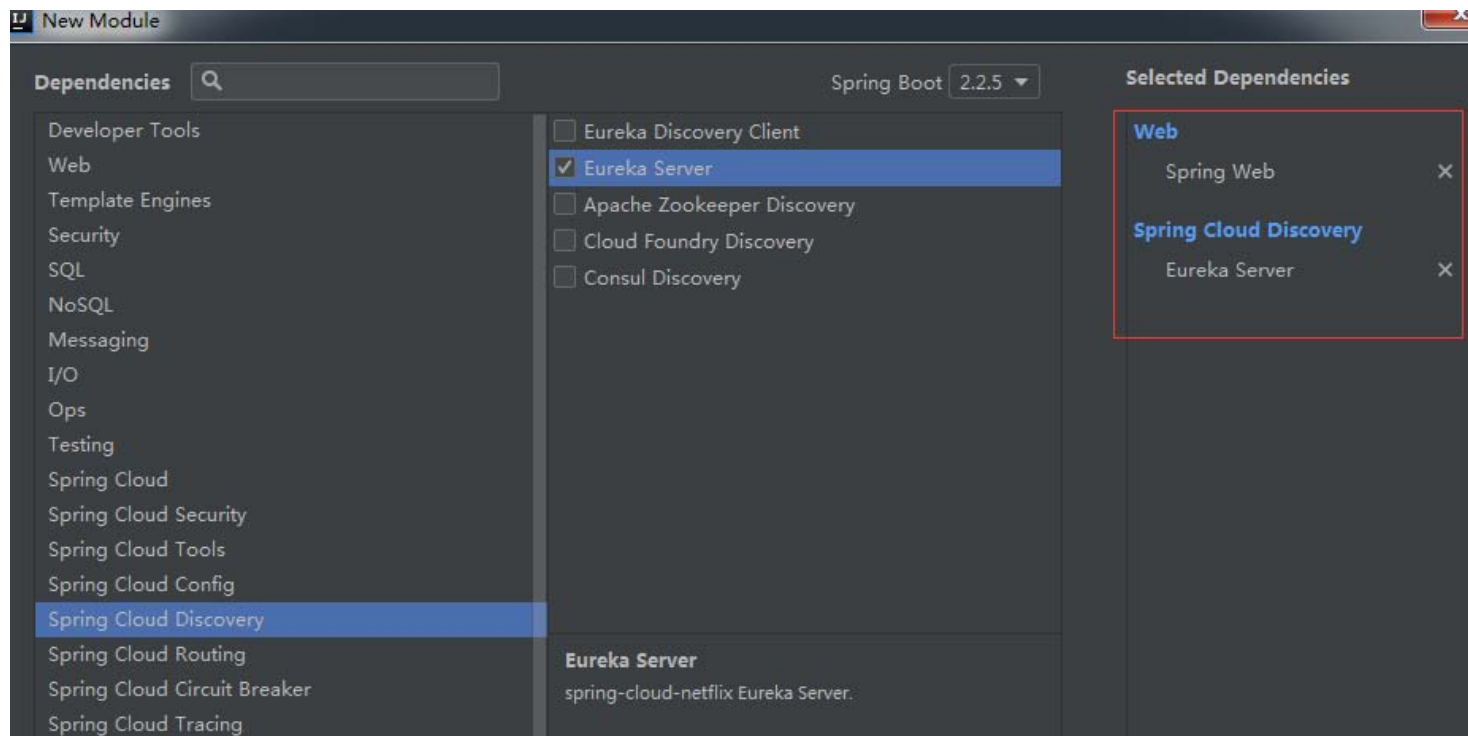


# Eureka集群配置

- ❖ 集群配置案例：
  - ▶ 建立3台Eureka Server
- ❖ 需要再建立2台Eureka Server
  - ▶ 1、新建eureka2、eureka3
  - ▶ 2、按照eureka选择依赖
  - ▶ 3、修改eureka2、eureka3主启动类
  - ▶ 4、修改hosts映射配置
  - ▶ 5、修改eureka服务器的全局配置文件

# Eureka集群配置

- ❖ 1、新建Module模块eureka2、eureka3
- ❖ 2、按照eureka选择依赖
- ❖ 3、修改eureka2、eureka3主启动类



# Eureka集群配置

## ❖ 4、修改映射配置

- ▶ 修改域名映射
- ▶ 修改C:\Windows\System32\drivers\etc路径中的hosts文件

127.0.0.1 eureka1.com

127.0.0.1 eureka2.com

127.0.0.1 eureka3.com

# Eureka集群配置

## ❖ 5、修改Eureka服务器的全局配置文件

- ▶ Eureka Server7001修改如下:
- ▶ 其他7002、7003两个工程需要修改端口号与defaultZone（配置另外两个服务地址）

spring:

application:

name: **regist-center** #注册中心的名字, **不变**

server:

port: **7001** #服务端口

#注册中心相关配置

eureka:

# 配置关闭自我保护, 并按需配置Eureka Server清理无效节点的时间间隔 (5000ms)。

server:

enable-self-preservation: false

eviction-interval-timer-in-ms: 5000

client:

register-with-eureka: false # 不将自己注册到注册中心

fetch-registry: false # 因为自己是注册中心, 因此不用检索服务信息

# 注册中心的地址

service-url:

defaultZone: **http://eureka2:7002/eureka/, http://eureka3:7003/eureka/**

instance:

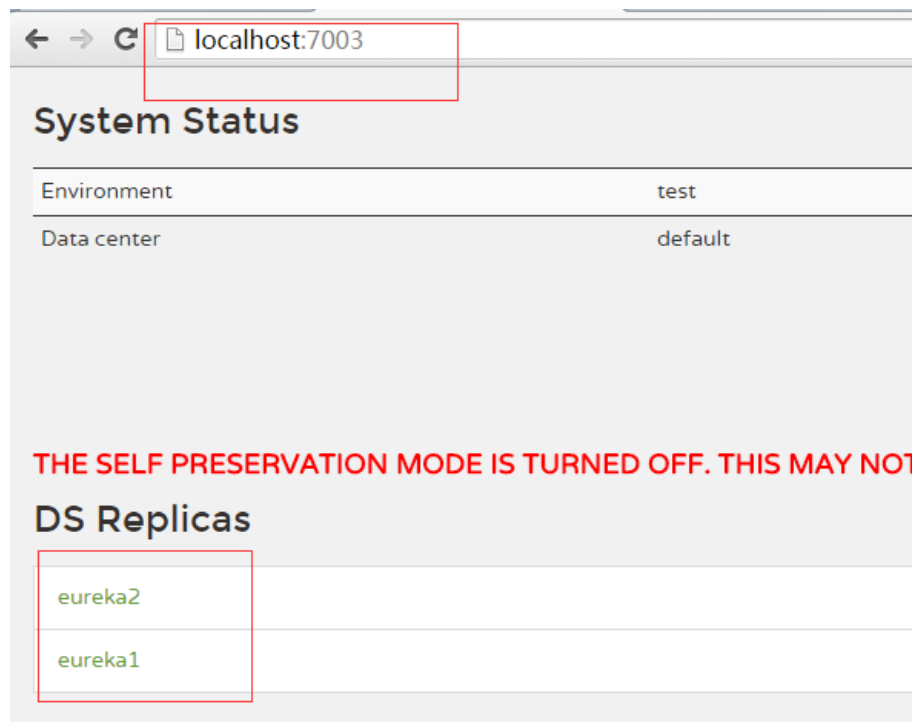
prefer-ip-address: true #用机器名标识服务名

hostname: localhost

# Eureka集群配置

## ❖ 测试

- ▶ 依次启动Eureka Server 7001、7002、7003
- ▶ 浏览器的访问地址修改为域名：端口号
  - ★ `http://eureka:7003`



# 微服务发布到集群配置中

## ❖ 案例：

- ▶ 将provider微服务发布到前面建立的集群配置中
- ▶ 测试 `http://eureka3:7003`

← → ↻ localhost:7003

### System Status

Environment	test	Current time	2020-03-14T16:55:46 +0800
Data center	default	Uptime	00:01
		Lease expiration enabled	true
		Renews threshold	3
		Renews (last min)	0

THE SELF PRESERVATION MODE IS TURNED OFF. THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.

### DS Replicas

- eureka2
- eureka1

### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
PROVIDER	n/a (1)	(1)	UP (1) - provider1

# 本章重点总结

- ❖ 了解Eureka及服务注册与发现；
- ❖ 了解Eureka的自我保护模式；
- ❖ 了解Actuator与注册微服务信息完善；
- ❖ 掌握Eureka Server的编写；
- ❖ 掌握将微服务注册到Eureka Server上；
- ❖ 掌握消费者从Eureka Server中发现服务；
- ❖ 掌握Eureka集群配置；

# 课后作业【必做任务】

- ❖ 1、独立完成课件中的示例

