# SpringCloud微服务分享说明

该分享是基于SpringBoot2.1基础之上

代码开发工具：idea2018，

maven环境，jdk1.8及以上，

网络联通状态。

# 网站架构演变过程

## 传统架构

传统的SSH架构，分为三层架构 web控制层、业务逻辑层、数据库访问层。

传统架构也就是单点应用，就是大家在刚开始初学JavaEE技术的时候SSH架构或者SSM架构，业务没有进行拆分，都写同一个项目工程里面，一般是适合于个人或者是小团队开发。

这种架构模式，一旦有一个模块导致服务不可用，可能会影响整个项目。

## 分布式架构

分布式架构基于传统架构演变过来，将传统的单体项目以项目模块进行拆分，拆分为会员项目、订单项目、支付项目、优惠券项目等，从而降低耦合度，这种项目架构模式慢慢开始适合于互联网公司规模人数开发。

## SOA架构

SOA架构代表面向与服务架构，俗称服务化，通俗的理解为面向与业务逻辑层开发，将共同的业务逻辑抽取出来形成一个服务，提供给其他服务接口进行调用，服务与服务之间调用使用rpc远程技术。

SOA架构特点:

1.SOA架构中通常使用XML方式实现通讯，在高并发情况下XML比较冗余会带来极大的影响，所以最后微服务架构中采用JSON替代xml方式

2.SOA架构的底层实现通过WebService和ESB（xml与中间件混合物），Web Service技术是SOA服务化的一种实现方式，WebService底层采用soap协议进行通讯，soap协议就是Http或者是Https通道传输XML数据实现的协议。

## 微服务架构

### 微服务架构产生的原因

微服务架构基于SOA架构演变过来的

在传统的WebService架构中有如下问题:

1. 依赖中心化服务发现机制
2. 使用Soap通讯协议，通常使用XML格式来序列化通讯数据，xml格式非常喜欢重，比较占宽带传输。
3. 服务化管理和治理设施不完善

### 漫谈微服务架构

## 什么是微服务

微服务架是从SOA架构演变过来，比SOA架构粒度会更加精细，让专业的人去做专业的事情（专注），目的提高效率，每个服务于服务之间互不影响，微服务架构中，每个服务必须独立部署，互不影响，微服务架构更加体现轻巧、轻量级，是适合于互联网公司敏捷开发。

## 微服务架构特征

微服务架构倡导应用程序设计程多个独立、可配置、可运行和可微服务的子服务。

服务与服务通讯协议采用Http协议，使用restful风格API形式来进行通讯，数据交换格式轻量级json格式通讯，整个传输过程中，采用二进制，所以http协议可以跨语言平台，并且可以和其他不同的语言进行相互的通讯，所以很多开放平台都采用http协议接口。

## 微服务架构如何拆分

1.微服务把每一个职责单一功能存放在独立的服务中

2.每个服务运行在单独的进程中

3.每个服务有自己独立数据库存储、实际上有自己独立的缓存、数据库、消息队列等资源。

### 微服务架构与SOA架构区别

1.微服务架构基于 SOA架构 演变过来，继承 SOA架构的优点，在微服务架构中去除 SOA 架构中的 ESB 消息总线，采用 http+json（restful）进行传输。

2.微服务架构比 SOA 架构粒度会更加精细，让专业的人去做专业的事情（专注），目的提高效率，每个服务于服务之间互不影响，微服务架构中，每个服务必须独立部署，微服务架构更加轻巧，轻量级。

3.SOA 架构中可能数据库存储会发生共享，微服务强调独每个服务都是单独数据库，保证每个服务于服务之间互不影响。

4.项目体现特征微服务架构比 SOA 架构更加适合与互联网公司敏捷开发、快速迭代版本，因为粒度非常精细。

# SpringCloud微服务框架

## 为什么选择SpringCloud

因为SpringCloud出现,对微服务技术提供了非常大的帮助，因为SpringCloud 提供了一套完整的微服务解决方案，不像其他框架只是解决了微服务中某个问题。

服务治理: 阿里巴巴开源的Dubbo和当当网在其基础上扩展的Dubbox、Eureka、Apache 的Consul等

分布式配置中心: 百度的disconf、Netfix的Archaius、360的QConf、SpringCloud、携程的阿波罗等。

分布式任务:xxl-job、elastic-job、springcloud的task等。

服务跟踪:京东的hyra、springcloud的sleuth等

## SpringCloud简介

SpringCloud是基于**SpringBoot基础之上开发的微服务框架**，SpringCloud是一套目前非常完整的微服务解决方案框架，其内容包含服务治理、注册中心、配置管理、断路器、智能路由、微代理、控制总线、全局锁、分布式会话等。

SpringCloud包含众多的子项目

SpringCloud config 分布式配置中心

SpringCloud netflix 核心组件

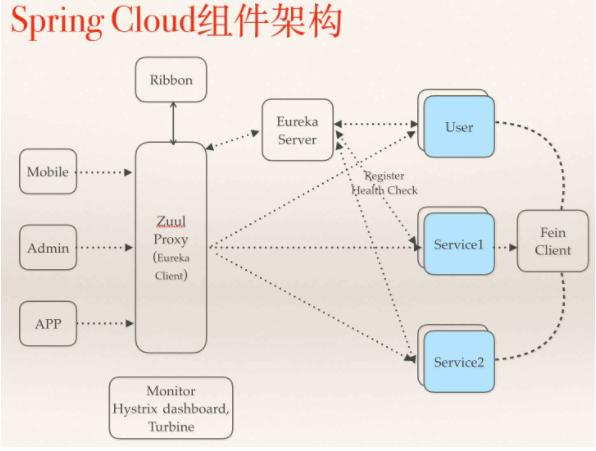
Eureka:服务治理 注册中心

Hystrix:服务保护框架

Ribbon:客户端负载均衡器

Feign：基于ribbon和hystrix的声明式服务调用组件

Zuul: 网关组件,提供智能路由、访问过滤等功能。



# 服务治理SpringCloud Eureka

## 什么是服务治理

在传统rpc远程调用中，服务与服务依赖关系，管理比较复杂，所以需要使用服务治理，管理服务与服务之间依赖关系，可以实现服务调用、负载均衡、容错等，实现服务发现与注册。

## 服务注册与发现

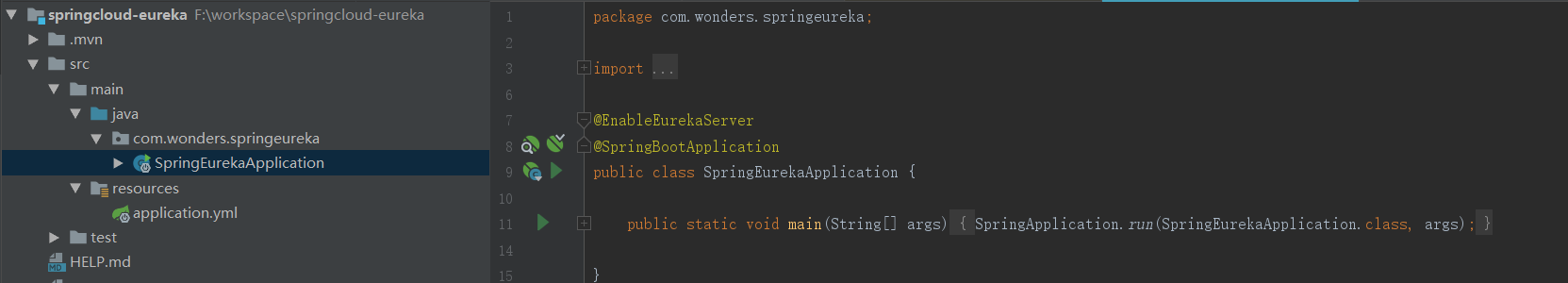
在服务注册与发现中，有一个注册中心，当服务器启动的时候，会把当前自己服务器的信息 比如 服务地址通讯地址等以别名方式注册到注册中心上。

另一方（消费者|服务提供者），以该别名的方式去注册中心上获取到实际的服务通讯地址，让后在实现本地rpc调用远程。

## 搭建注册中心

### 常用注册中心框架

### 注册中心环境搭建



#### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.12.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <groupId>com.wonders</groupId>  <artifactId>springcloud-eureka</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>springcloud-eureka</name>  <description>Demo project for Spring Boot</description>  <properties>  <java.version>1.8</java.version>  <spring-cloud.version>Greenwich.SR5</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement> |

#### application.yml

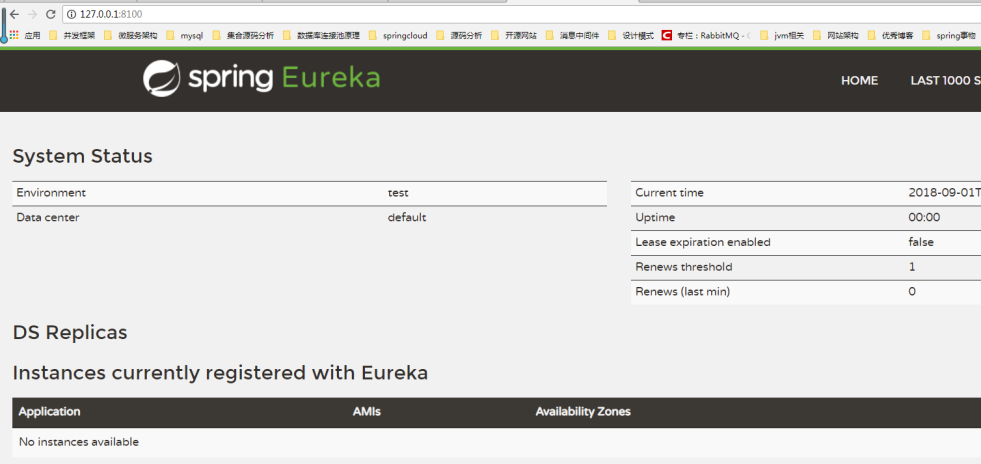
|  |
| --- |
| ###服务端口号  server:  port: 8100  ###eureka 基本信息配置  eureka:  instance:  ###注册到eurekaip地址  hostname: 127.0.0.1  client:  service-url:  defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/  ###因为项目是注册中心，不需要自己注册自己  register-with-eureka: **false**  ###因为项目是注册中心，不需要检索服务  fetch-registry: **false** |

#### 启动Eureka服务

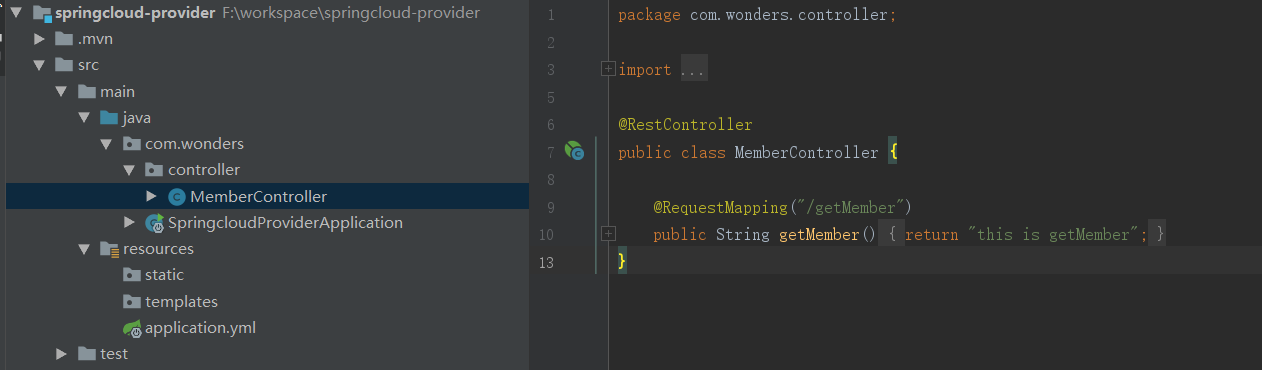
|  |
| --- |
| @EnableEurekaServer  @SpringBootApplication  public class SpringEurekaApplication {  public static void main(String[] args) {  SpringApplication.run(SpringEurekaApplication.class, args);  }  } |

@EnableEurekaServer作用:开启eureka server服务。

启动服务，输入http://127.0.0.1:8100,可打开eureka界面。



### 注册服务提供者



#### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.12.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <groupId>com.wonders</groupId>  <artifactId>springcloud-provider</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>springcloud-provider</name>  <description>Demo project for Spring Boot</description>  <properties>  <java.version>1.8</java.version>  <spring-cloud.version>Greenwich.SR5</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement> |

#### application.yml

|  |
| --- |
| ###服务启动端口号  server:  port: 8000  ###服务名称(服务注册到eureka名称)  spring:  application:  name: app-springcloud-provider  ###服务注册到eureka地址  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:8001/eureka  ###注册服务  register-with-eureka: true  ###检索服务  fetch-registry: true |

#### 服务接口

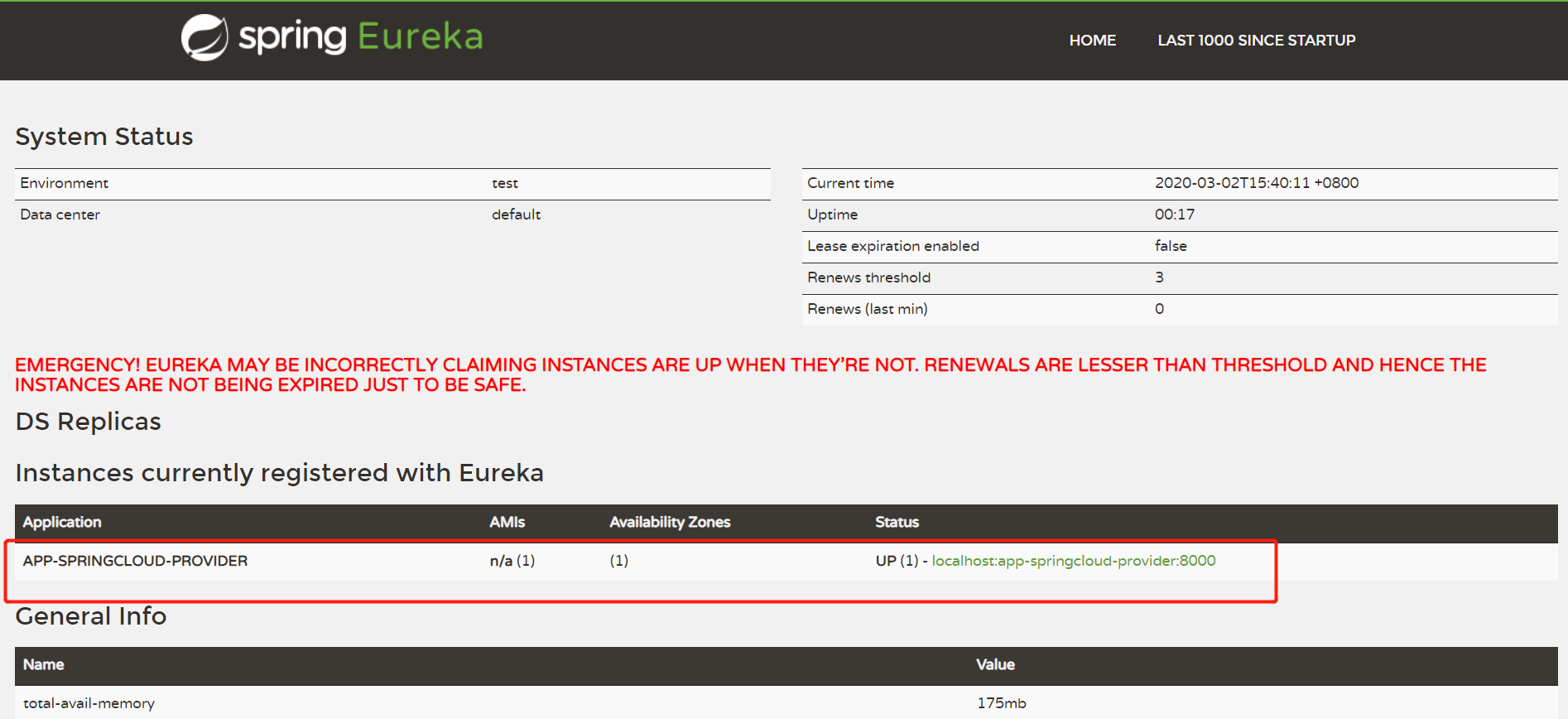
|  |
| --- |
| @RestController  **public** **class** MemberController {  @RequestMapping("/getMember")  **public** String getMember() {  **return** "this is getMember";  }  } |

#### 启动服务提供者项目

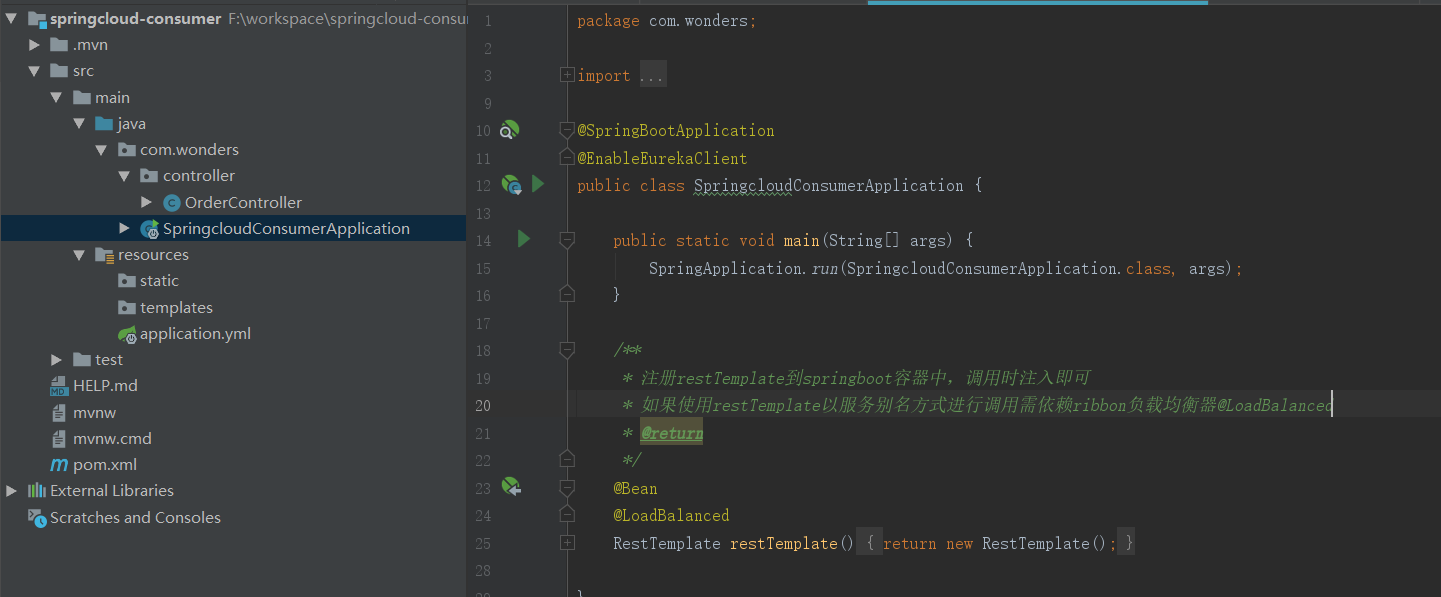
|  |
| --- |
| @EnableEurekaClient  @SpringBootApplication  public class SpringcloudProviderApplication {  public static void main(String[] args) {  SpringApplication.run(SpringcloudProviderApplication.class, args);  }  } |

@EnableEurekaClient 作用:开启eureka client服务。

启动成功，可在eureka界面看到服务在注册列表中



### 服务消费者



#### Maven依赖信息

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.12.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <groupId>com.wonders</groupId>  <artifactId>springcloud-consumer</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>springcloud-consumer</name>  <description>Demo project for Spring Boot</description>  <properties>  <java.version>1.8</java.version>  <spring-cloud.version>Greenwich.SR5</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement> |

#### Application.yml配置

|  |
| --- |
| ###服务启动端口号  server:  port: 8001  ###服务名称(服务注册到eureka名称)  spring:  application:  name: app-springcloud-consumer  ###服务注册到eureka地址  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:8001/eureka  ###注册服务  register-with-eureka: true  ###检索服务  fetch-registry: true |

#### 使用rest方式调用服务

|  |
| --- |
| @RestController  public class OrderController {  @Autowired  private RestTemplate restTemplate;  @RequestMapping("/getOrder")  public String getOrder() {  // order 使用rpc 远程调用技术 调用注册服务  String memberUrl = "http://app-springcloud-provider/getMember";  String result = restTemplate.getForObject(memberUrl, String.class);  System.out.println("调用服务,result:" + result);  return result;  }  } |

#### 启动消费者服务

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  public class SpringcloudConsumerApplication {  public static void main(String[] args) {  SpringApplication.run(SpringcloudConsumerApplication.class, args);  }  /\*\*  \* 注册restTemplate到springboot容器中，调用时注入即可  \* 如果使用restTemplate以服务别名方式进行调用需依赖ribbon负载均衡器@LoadBalanced  \* @return  \*/  @Bean  @LoadBalanced  RestTemplate restTemplate() {  return new RestTemplate();  }  } |

@LoadBalanced就能让这个RestTemplate在请求时拥有客户端负载均衡的能力

## 高可用注册中心

在微服务中，注册中心非常核心，可以实现服务治理，如果一旦注册出现故障的时候，可能会导致整个微服务无法访问，在这时候就需要对注册中心实现高可用集群模式。

### Eureka高可用原理

默认情况下Eureka是让服务注册中心，不注册自己

|  |
| --- |
| ###因为该应用为注册中心，不会注册自己（集群需开启）  register-with-eureka: **true**  ###不需要去注册中心上检索服务（集群需开启）  fetch-registry: **true** |

Eureka高可用实际上将自己作为服务向其他服务注册中心注册自己，这样就可以形成一组相互注册的服务注册中心，从而实现服务清单的互相同步，达到高可用效果。

### Eureka集群环境搭建

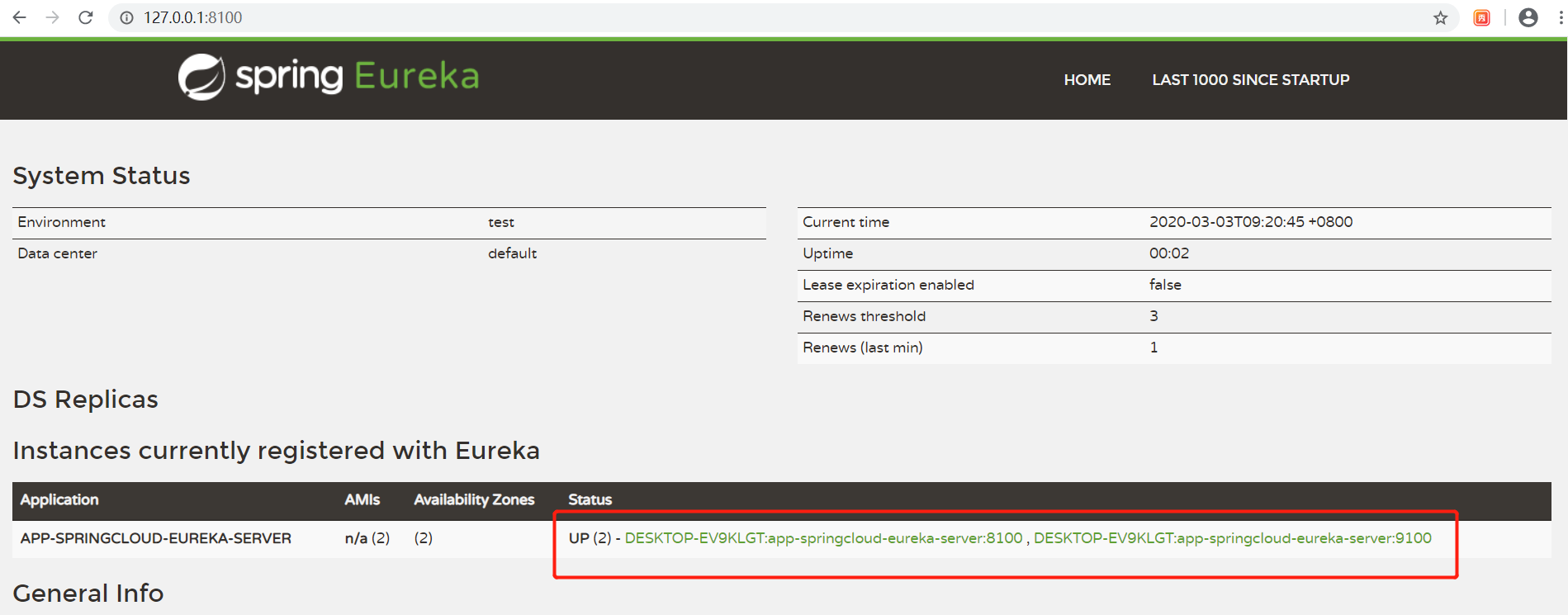
#### Eureka01配置

|  |
| --- |
| ###服务端口号  server:  port: 8100  ###eureka 基本信息配置  spring:  application:  name: app-springcloud-eureka-server  eureka:  instance:  ###注册到eurekaip地址  hostname: 127.0.0.1  client:  service-url:  defaultZone: http://127.0.0.1:9100/eureka/  ###因为项目是注册中心，将自己作为服务向其他服务注册中心注册  register-with-eureka: **true**  ###开启检索服务  fetch-registry: **true** |

#### Eureka02配置

|  |
| --- |
| ###服务端口号  server:  port: 9100  ###eureka 基本信息配置  spring:  application:  name: app-springcloud-eureka-server  eureka:  instance:  ###注册到eurekaip地址  hostname: 127.0.0.1  client:  service-url:  defaultZone: http://127.0.0.1:8100/eureka/  ###因为项目是注册中心，将自己作为服务向其他服务注册中心注册  register-with-eureka: **true**  ###开启检索服务  fetch-registry: **true** |

集群时，注册中心相互注册，实现你中有我，我中有你。Spring.application.name应用名称要相同,如有多台是，在eureka.client.serviceUrl.defaultZone属性后面用逗号隔开添加注册。



#### 客户端集成Eureka集群

添加集群地址即可。

|  |
| --- |
| ###服务启动端口号  server:  port: 8000  ###服务名称(服务注册到eureka名称)  spring:  application:  name: app-springcloud-provider  ###服务注册到eureka地址  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:8100/eureka,http://127.0.0.1:9100/eureka  ###注册服务  register-with-eureka: true  ###检索服务  fetch-registry: true |

启动服务提供者和消费者，在eureka界面可看到服务注册在某一台机器上，相当于主服务器，另外一台则是从，关闭主服务连接，可观察服务会注册到从服务器，界面要一点时间更新才可以看到，耐心等待刷新即可。