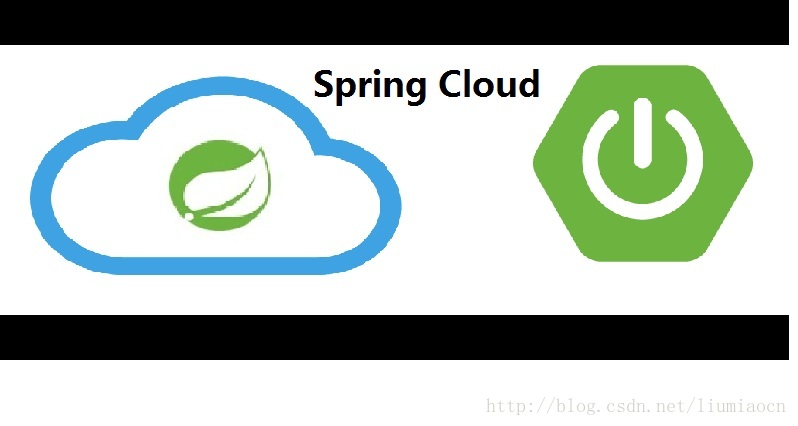
### [Spring基础：快速入门spring cloud（1）：Spring Cloud介绍](http://blog.csdn.net/liumiaocn/article/details/53868315)

分类：**spring**

 （1038）  （0）  举报  收藏



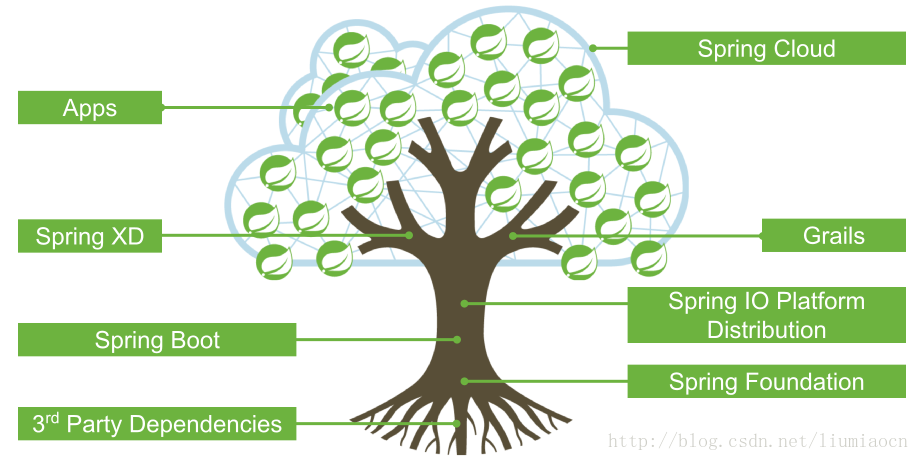
分布式系统, 微服务, Java, 当这三个词放到一起的时候, 很多人自然而然地就会想起Spring Cloud. Spring Cloud是Spring总多的Project中的一个,它提供了一整套的工具帮助系统架构师们在进行分布式设计的时候可以拿来即用, 在创建和发布微服务时极为便捷和有效.

# 基本信息

本系列介绍所使用的spring cloud的Release为Camden SR3。 虽然各个版本可能相差不大。

| **项目** | **详细** |
| --- | --- |
| Release | Camden SR3 |
| Reference | <http://cloud.spring.io/spring-cloud-static/Camden.SR3/> |
| API | <http://cloud.spring.io/spring-cloud-static/spring-cloud.html> |

# Spring Cloud

Spring Cloud提供了一整套的工具帮助创建分布式应用，使用Spring Cloud可以用Spring Cloud的方式创建符合12要素的原生云应用。Spring Cloud与Spring Boot等一起，提供了一整套的微服务架构的原生云应用的解决方案。   


## 分布式应用

分布式应用相比普通应用，因为它的复杂，所以导致很多问题我们在作架构的时候不得不考虑，传统的方式下是通过总多的软件/硬件的集成来解决的。比如至少需要考虑如下问题：

| **项目** | **详细** |
| --- | --- |
| No.1 | 配置管理 |
| No.2 | 控制总线 |
| No.3 | 集群管理 |
| No.4 | 安全机制 |
| No.5 | Session管理 |
| No.6 | Failback |
| No.7 | 智能路由 |
| No.8 | 网关管理 |
| No.9 | 服务管理（服务发现/服务注册等） |
| … | … |

而一旦我们有了Spring Cloud，基本上就是一个All in One的集合。

## 原生云应用

原生云应用Cloud Native Applications是一个被叫的越来越多的概念。Cloud Native到底是什么？从Spring Cloud的角度去理解，Cloud Native是在持续交付和价值驱动的软件开发领域为了鼓励更加方便快捷有效地实施最佳实践的一种软件开发方式。可能这个解释依然不能让大多数人所理解，那么我们换一种方式来理解，它在改变什么？传统的应用是什么？

### 传统应用

关于传统应用下应用相关，这里列出一些我们常见的条目：

| **项目** | **详细** |
| --- | --- |
| 架构 | 3层架构或者N层架构 |
| 耦合 | 紧耦合，有状态 |
| 负载自动扩展 | 不可，或者非常困难 |
| 物理宕机 | 极力避免（钱+精力） |
| 应用恢复 | 检测->确认->手动恢复 |
| DevOps | 开发和运维割裂，持续集成和持续部署困难 |
| … | … |

总结一下，一个词，很娇气，像我们养的宠物。以前有一些关于pets 和 cattle的讨论在这方面能给一些启事，没有读过的建议搜索来读一下。

### 原生云应用

| **项目** | **详细** |
| --- | --- |
| 架构 | 微服务 |
| 耦合 | 松耦合，尽量无状态 |
| 负载自动扩展 | 自动水平扩展 |
| 物理宕机 | 自愈 |
| 应用恢复 | 通过设计和Platform结合，能够自愈 |
| DevOps | 开发和运维无缝集成，持续集成和持续部署方便 |
| … | … |

总结一下，一个词， 很皮实。像养牛场里的牛，有问题了随时换一个，而不是像宠物那样傲娇。以前写过一些文章关于Rancher，Rancher是一个用于部署和管理生产环境的容器的开源平台,它与Kubernetes/Mesos/Docker Swarm进行集成，但是它自己也有一个，它把这个也命名为Cattle，项目的名称也是这样，有兴趣地可以研究一下。

## Twelve-Factor

### URL

| **项目** | **详细** |
| --- | --- |
| 英文URL | <https://12factor.net/> |
| 中文URL | <https://12factor.net/zh_cn/> |

### 摘要

| **项目** | **详细** |
| --- | --- |
| I. 基准代码 | 一份基准代码，多份部署 |
| II. 依赖 | 显式声明依赖关系 |
| III. 配置 | 在环境中存储配置 |
| IV. 后端服务 | 把后端服务当作附加资源 |
| V. 构建，发布，运行 | 严格分离构建和运行 |
| VI. 进程 | 以一个或多个无状态进程运行应用 |
| VII. 端口绑定 | 通过端口绑定提供服务 |
| VIII. 并发 | 通过进程模型进行扩展 |
| IX. 易处理 | 快速启动和优雅终止可最大化健壮性 |
| X. 开发环境与线上环境等价 | 尽可能的保持开发，预发布，线上环境相同 |
| XI. 日志 | 把日志当作事件流 |
| XII. 管理进程 | 后台管理任务当作一次性进程运行 |

# Pom

使用Maven可以非常简单地将Spring Cloud导入工程之中，具体Maven片段如下

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.0.RELEASE</version>

</parent>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>Camden.SR3</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

</dependencies>

# 子项目

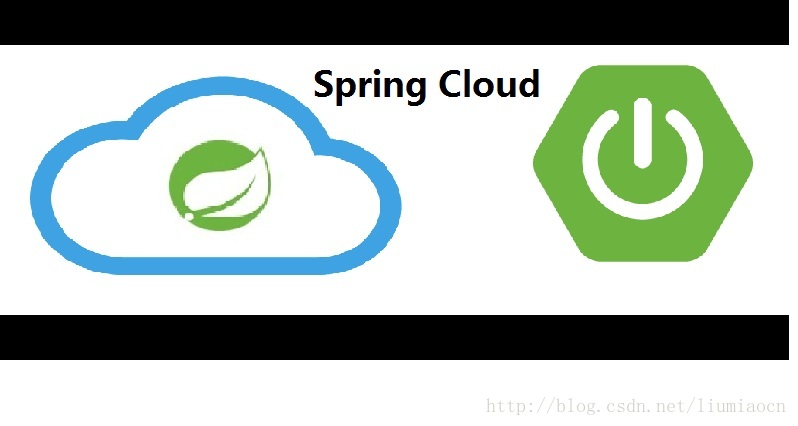
| **项目** | **详细** |
| --- | --- |
| Spring Cloud Config | 统一配置管理,支持本地/Git/SVN. |
| Spring Cloud Netflix | Netflix提供的多种组件包括Eureka, Hystrix, Zuul, Archaius等。 |
| Spring Cloud Bus | 使用分布式的消息机制将服务和服务实例连接起来的事件总线，在跨集群进行状态通知时非常有效，比如配置变更事件。 |
| Spring Cloud for Cloud Foundry | 与CloudFoundry进行集成 |
| Spring Cloud Cloud Foundry Service Broker | 提供Service Broker用来管理相关服务 |
| Spring Cloud Cluster | 通过对Zookeeper, Redis, Hazelcast, Consul等进行抽象和实现所提供的关于Leadership election以及common stateful patterns的支持 |
| Spring Cloud Consul | 使用Hashicorp Consul进行服务发现和配置管理 |
| Spring Cloud Security | 为应用程序添加OAuth2的安全控制 |
| Spring Cloud Sleuth | 提供与Zipkin/HTrace/ELK等兼容的Tracing机制 |
| Spring Cloud Data Flow | 简单易用的DSL与扩拽可用的GUI，加之REST-API一起提供了一种简化的微服务数据流管道机制。 |
| Spring Cloud Stream | 使用Apache Kafka或者RabbitMQ收发消息，提供了一种轻量级的事件驱动的微服务框架用于和外系统进行连接。 |
| Spring Cloud Stream App Starters | 基于spring boot，与外系统进行整合 |
| Spring Cloud Zookeeper | 使用Apache Zookeeper进行服务发现和配置管理 |
| Spring Cloud for Amazon Web Services | 用于简化与Amazon Web Service更好的集成 |
| Spring Cloud Connectors | 使得Paas应用更容易与后端服务（比如数据库）进行交互 |
| Spring Cloud Starters | 更简单的进行以来管理（同其他starter类似） |
| Spring Cloud CLI | 用CLI方式创建Spring Cloud的各种组件。 |

总结：本文走马观花地过了一下Spring Cloud相关的知识和概念，从下篇文章开始，我们将使用具体的例子，逐步深入研究是如何使用Spring Cloud的各种组件非常简单的进行服务发现/负载均衡/配置管理/网关管理等。

### [Spring基础：快速入门spring cloud（2）：服务发现之eureka](http://blog.csdn.net/liumiaocn/article/details/53886164)

分类：**spring**

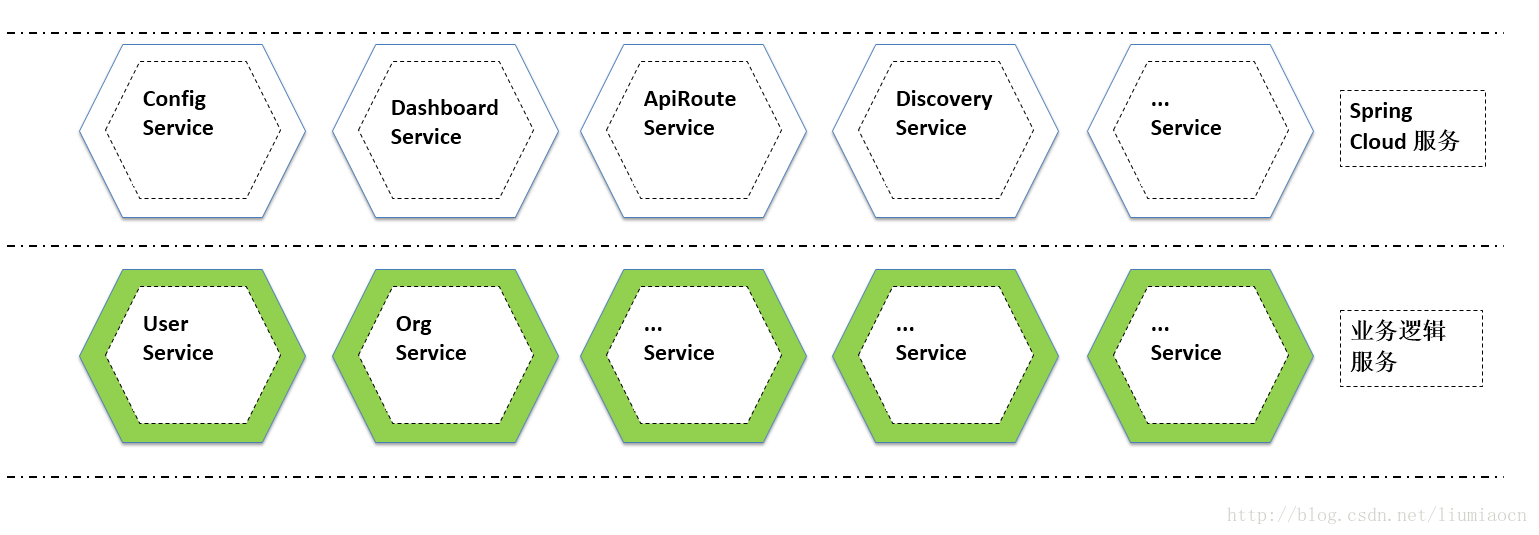
 （1323）  （0）  举报  收藏



Spring Cloud是Spring总多的Project中的一个,它提供了一整套的工具帮助系统架构师们在进行分布式设计的时候可以拿来即用, 在创建和发布微服务时极为便捷和有效。本系列文章将会使用最简单的例子和最为容易的方式来学习Spring Cloud。本文将会介绍如何使用Spring Cloud的Eureka实现服务发现。

# 构成

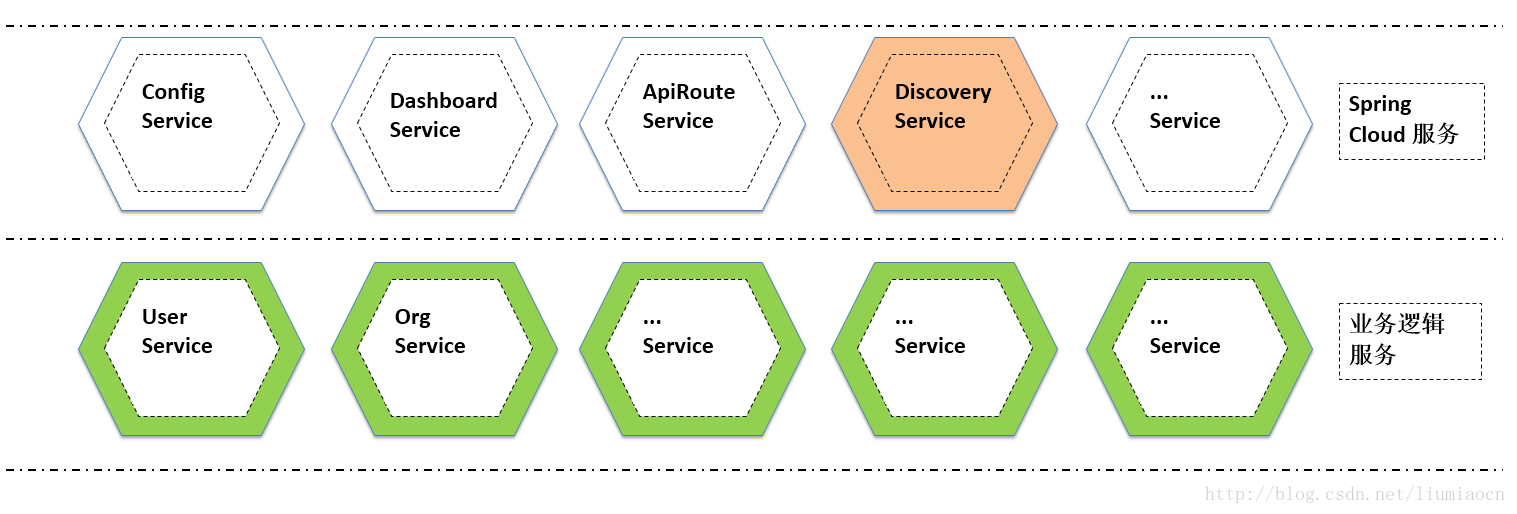
Spring Cloud由很多子项目构成，为了介绍方便，挑出一些项目中常用的进行如下构成。



具体各个Service之间的关系不再一一展示，将其粗暴地分为框架类服务（Spring Cloud服务）也业务逻辑服务两种，各服务功能和实现简单如下说明。

| **项目** | **详细** |
| --- | --- |
| Config Service | Spring Cloud Config:统一配置管理服务 |
| Dashboard Service | Hystrix Dashboard |
| Api Route Service | Zuul:Api Gateway |
| Discovery Service | Eureka：服务发现 |
| User Service | RESTFUL的用户相关的服务 |
| Org Service | RESTFUL的组织相关的服务 |

# Discovery Service



本文将会用最简单的方式来介绍如何使用Eureka进行服务发现的，以及Spring Cloud中使用Eureka是如何方便。

# Sprint Boot项目

创建一个Spring Boot项目，详细不再介绍， 集体方法可以参看如下文档。

| **项目** | **详细** |
| --- | --- |
| SPRING INITIALIZR | <http://blog.csdn.net/liumiaocn/article/details/53442364> |

# Eureka Server

## Pom设定

需要将 <artifactId>spring-cloud-starter-eureka-server</artifactId>加入到POM中，可以参照如下Pom示例。

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.liumiaocn.demo.springcloud</groupId>

<artifactId>discoveryservice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>discoveryservice</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-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>Camden.BUILD-SNAPSHOT</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-snapshots</id>

<name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

## 注解

加入注解EnableEurekaServer，在Spring boot的应用中只需这样一行就将EurekaServer引入其中。

package com.liumiaocn.demo.springcloud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class DiscoveryserviceApplication {

public static void main(String[] args) {

SpringApplication.run(DiscoveryserviceApplication.class, args);

}

}

## 设定文件

使用bootstrap和application两个设定文件来进行相关设定，入门时为了更加快速，使用最少的设定文件和设定语句，基本上只有不可或缺的才会加上，入门之后详细内容参看spring cloud官方文档即可。application.properties文件的最少作如下设定即可。

server.port=8801

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

eureka.client.serviceUrl.defaultZone=http://localhost:${server.port}/eureka/

| **项目** | **详细** |
| --- | --- |
| server.port | Eureka Server的Dashboard所使用的port |
| eureka.client.register-with-eureka | 是否要注册到服务器端，因为此处为Eureka Server，所以设定为false |
| eureka.client.fetch-registry | 是否从服务器端取得注册信息，因为此处为Eureka Server，所以设定为false |
| eureka.client.serviceUrl.defaultZone | 设定为为<http://localhost:8801/eureka/>，缺省为8761端口。 |

## 启动

2016-12-28 07:59:33.200 INFO 5228 --- [ main] s.c.a.AnnotationConfigApplicationContext : Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@c86b9e3: startup date [Wed Dec 28 07:59:33 CST 2016]; root of context hierarchy

2016-12-28 07:59:33.434 INFO 5228 --- [ main] f.a.AutowiredAnnotationBeanPostProcessor : JSR-330 'javax.inject.Inject' annotation found and supported for autowiring

2016-12-28 07:59:33.466 INFO 5228 --- [ main] trationDelegate$BeanPostProcessorChecker : Bean 'configurationPropertiesRebinderAutoConfiguration' of type [class org.springframework.cloud.autoconfigure.ConfigurationPropertiesRebinderAutoConfiguration$$EnhancerBySpringCGLIB$$fb56a29e] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v1.4.3.RELEASE)

2016-12-28 07:59:33.902 INFO 5228 --- [ main] c.l.d.s.DiscoveryserviceApplication : No active profile set, falling back to default profiles: default

2016-12-28 07:59:33.902 INFO 5228 --- [ main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@35229f85: startup date [Wed Dec 28 07:59:33 CST 2016]; parent: org.springframework.context.annotation.AnnotationConfigApplicationContext@c86b9e3

2016-12-28 07:59:34.796 WARN 5228 --- [ main] o.s.c.a.ConfigurationClassPostProcessor : Cannot enhance @Configuration bean definition 'refreshScope' since its singleton instance has been created too early. The typical cause is a non-static @Bean method with a BeanDefinitionRegistryPostProcessor return type: Consider declaring such methods as 'static'.

2016-12-28 07:59:34.967 INFO 5228 --- [ main] o.s.cloud.context.scope.GenericScope : BeanFactory id=37576f2c-11e3-3d56-80ff-2435c6a59fa3

2016-12-28 07:59:35.030 INFO 5228 --- [ main] f.a.AutowiredAnnotationBeanPostProcessor : JSR-330 'javax.inject.Inject' annotation found and supported for autowiring

2016-12-28 07:59:35.139 INFO 5228 --- [ main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.cloud.autoconfigure.ConfigurationPropertiesRebinderAutoConfiguration' of type [class org.springframework.cloud.autoconfigure.ConfigurationPropertiesRebinderAutoConfiguration$$EnhancerBySpringCGLIB$$fb56a29e] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)

2016-12-28 07:59:35.562 INFO 5228 --- [ main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8801 (http)

2016-12-28 07:59:35.578 INFO 5228 --- [ main] o.apache.catalina.core.StandardService : Starting service Tomcat

2016-12-28 07:59:35.578 INFO 5228 --- [ main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.6

2016-12-28 07:59:35.687 INFO 5228 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

2016-12-28 07:59:35.687 INFO 5228 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1785 ms

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'metricsFilter' to: [/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.c.embedded.FilterRegistrationBean : Mapping filter: 'webRequestTraceFilter' to: [/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.c.embedded.FilterRegistrationBean : Mapping filter: 'servletContainer' to urls: [/eureka/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'applicationContextIdFilter' to: [/\*]

2016-12-28 07:59:36.248 INFO 5228 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]

2016-12-28 07:59:36.342 INFO 5228 --- [ost-startStop-1] c.s.j.s.i.a.WebApplicationImpl : Initiating Jersey application, version 'Jersey: 1.19.1 03/11/2016 02:08 PM'

2016-12-28 07:59:36.404 INFO 5228 --- [ost-startStop-1] c.n.d.provider.DiscoveryJerseyProvider : Using JSON encoding codec LegacyJacksonJson

2016-12-28 07:59:36.404 INFO 5228 --- [ost-startStop-1] c.n.d.provider.DiscoveryJerseyProvider : Using JSON decoding codec LegacyJacksonJson

2016-12-28 07:59:36.514 INFO 5228 --- [ost-startStop-1] c.n.d.provider.DiscoveryJerseyProvider : Using XML encoding codec XStreamXml

2016-12-28 07:59:36.514 INFO 5228 --- [ost-startStop-1] c.n.d.provider.DiscoveryJerseyProvider : Using XML decoding codec XStreamXml

2016-12-28 07:59:37.230 INFO 5228 --- [ main] o.s.c.n.eureka.InstanceInfoFactory : Setting initial instance status as: STARTING

2016-12-28 07:59:37.292 INFO 5228 --- [ main] com.netflix.discovery.DiscoveryClient : Client configured to neither register nor query for data.

2016-12-28 07:59:37.292 INFO 5228 --- [ main] com.netflix.discovery.DiscoveryClient : Discovery Client initialized at timestamp 1482883177292 with initial instances count: 0

2016-12-28 07:59:37.370 INFO 5228 --- [ main] c.n.eureka.DefaultEurekaServerContext : Initializing ...

2016-12-28 07:59:37.370 INFO 5228 --- [ main] c.n.eureka.cluster.PeerEurekaNodes : Adding new peer nodes [http://localhost:8761/eureka/]

2016-12-28 07:59:37.682 INFO 5228 --- [ main] c.n.d.provider.DiscoveryJerseyProvider : Using JSON encoding codec LegacyJacksonJson

2016-12-28 07:59:37.682 INFO 5228 --- [ main] c.n.d.provider.DiscoveryJerseyProvider : Using JSON decoding codec LegacyJacksonJson

2016-12-28 07:59:37.682 INFO 5228 --- [ main] c.n.d.provider.DiscoveryJerseyProvider : Using XML encoding codec XStreamXml

2016-12-28 07:59:37.682 INFO 5228 --- [ main] c.n.d.provider.DiscoveryJerseyProvider : Using XML decoding codec XStreamXml

2016-12-28 07:59:37.791 INFO 5228 --- [ main] c.n.eureka.cluster.PeerEurekaNodes : Replica node URL: http://localhost:8761/eureka/

2016-12-28 07:59:37.807 INFO 5228 --- [ main] c.n.e.registry.AbstractInstanceRegistry : Finished initializing remote region registries. All known remote regions: []

2016-12-28 07:59:37.807 INFO 5228 --- [ main] c.n.eureka.DefaultEurekaServerContext : Initialized

2016-12-28 07:59:38.103 INFO 5228 --- [ main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@35229f85: startup date [Wed Dec 28 07:59:33 CST 2016]; parent: org.springframework.context.annotation.AnnotationConfigApplicationContext@c86b9e3

2016-12-28 07:59:38.228 INFO 5228 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/],methods=[GET]}" onto public java.lang.String org.springframework.cloud.netflix.eureka.server.EurekaController.status(javax.servlet.http.HttpServletRequest,java.util.Map<java.lang.String, java.lang.Object>)

2016-12-28 07:59:38.228 INFO 5228 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/lastn],methods=[GET]}" onto public java.lang.String org.springframework.cloud.netflix.eureka.server.EurekaController.lastn(javax.servlet.http.HttpServletRequest,java.util.Map<java.lang.String, java.lang.Object>)

2016-12-28 07:59:38.228 INFO 5228 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)

2016-12-28 07:59:38.228 INFO 5228 --- [ main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)

2016-12-28 07:59:38.259 INFO 5228 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]

2016-12-28 07:59:38.259 INFO 5228 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]

2016-12-28 07:59:38.290 INFO 5228 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/mappings || /mappings.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/health || /health.json],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.HealthMvcEndpoint.invoke(java.security.Principal)

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/beans || /beans.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/info || /info.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/restart || /restart.json],methods=[POST]}" onto public java.lang.Object org.springframework.cloud.context.restart.RestartMvcEndpoint.invoke()

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/refresh || /refresh.json],methods=[POST]}" onto public java.lang.Object org.springframework.cloud.endpoint.GenericPostableMvcEndpoint.invoke()

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/dump || /dump.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/configprops || /configprops.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.836 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/resume || /resume.json],methods=[POST]}" onto public java.lang.Object org.springframework.cloud.endpoint.GenericPostableMvcEndpoint.invoke()

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/trace || /trace.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/env],methods=[POST]}" onto public java.lang.Object org.springframework.cloud.context.environment.EnvironmentManagerMvcEndpoint.value(java.util.Map<java.lang.String, java.lang.String>)

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/env/reset],methods=[POST]}" onto public java.util.Map<java.lang.String, java.lang.Object> org.springframework.cloud.context.environment.EnvironmentManagerMvcEndpoint.reset()

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/metrics/{name:.\*}],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.MetricsMvcEndpoint.value(java.lang.String)

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/metrics || /metrics.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/heapdump || /heapdump.json],methods=[GET],produces=[application/octet-stream]}" onto public void org.springframework.boot.actuate.endpoint.mvc.HeapdumpMvcEndpoint.invoke(boolean,javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse) throws java.io.IOException,javax.servlet.ServletException

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/pause || /pause.json],methods=[POST]}" onto public java.lang.Object org.springframework.cloud.endpoint.GenericPostableMvcEndpoint.invoke()

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/autoconfig || /autoconfig.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/archaius || /archaius.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/env/{name:.\*}],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EnvironmentMvcEndpoint.value(java.lang.String)

2016-12-28 07:59:38.852 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/env || /env.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.868 INFO 5228 --- [ main] o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/features || /features.json],methods=[GET],produces=[application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()

2016-12-28 07:59:38.977 INFO 5228 --- [ main] o.s.ui.freemarker.SpringTemplateLoader : SpringTemplateLoader for FreeMarker: using resource loader [org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@35229f85: startup date [Wed Dec 28 07:59:33 CST 2016]; parent: org.springframework.context.annotation.AnnotationConfigApplicationContext@c86b9e3] and template loader path [classpath:/templates/]

2016-12-28 07:59:38.977 INFO 5228 --- [ main] o.s.w.s.v.f.FreeMarkerConfigurer : ClassTemplateLoader for Spring macros added to FreeMarker configuration

2016-12-28 07:59:39.070 WARN 5228 --- [ main] o.s.c.n.a.ArchaiusAutoConfiguration : No spring.application.name found, defaulting to 'application'

2016-12-28 07:59:39.070 WARN 5228 --- [ main] c.n.c.sources.URLConfigurationSource : No URLs will be polled as dynamic configuration sources.

2016-12-28 07:59:39.070 INFO 5228 --- [ main] c.n.c.sources.URLConfigurationSource : To enable URLs as dynamic configuration sources, define System property archaius.configurationSource.additionalUrls or make config.properties available on classpath.

2016-12-28 07:59:39.086 WARN 5228 --- [ main] c.n.c.sources.URLConfigurationSource : No URLs will be polled as dynamic configuration sources.

2016-12-28 07:59:39.086 INFO 5228 --- [ main] c.n.c.sources.URLConfigurationSource : To enable URLs as dynamic configuration sources, define System property archaius.configurationSource.additionalUrls or make config.properties available on classpath.

2016-12-28 07:59:39.164 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup

2016-12-28 07:59:39.164 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'refreshScope' has been autodetected for JMX exposure

2016-12-28 07:59:39.164 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'environmentManager' has been autodetected for JMX exposure

2016-12-28 07:59:39.164 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'configurationPropertiesRebinder' has been autodetected for JMX exposure

2016-12-28 07:59:39.164 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'refreshEndpoint' has been autodetected for JMX exposure

2016-12-28 07:59:39.164 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'restartEndpoint' has been autodetected for JMX exposure

2016-12-28 07:59:39.164 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Located managed bean 'environmentManager': registering with JMX server as MBean [org.springframework.cloud.context.environment:name=environmentManager,type=EnvironmentManager]

2016-12-28 07:59:39.180 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Located managed bean 'restartEndpoint': registering with JMX server as MBean [org.springframework.cloud.context.restart:name=restartEndpoint,type=RestartEndpoint]

2016-12-28 07:59:39.195 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Located managed bean 'refreshScope': registering with JMX server as MBean [org.springframework.cloud.context.scope.refresh:name=refreshScope,type=RefreshScope]

2016-12-28 07:59:39.195 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Located managed bean 'configurationPropertiesRebinder': registering with JMX server as MBean [org.springframework.cloud.context.properties:name=configurationPropertiesRebinder,context=35229f85,type=ConfigurationPropertiesRebinder]

2016-12-28 07:59:39.212 INFO 5228 --- [ main] o.s.j.e.a.AnnotationMBeanExporter : Located managed bean 'refreshEndpoint': registering with JMX server as MBean [org.springframework.cloud.endpoint:name=refreshEndpoint,type=RefreshEndpoint]

2016-12-28 07:59:39.369 INFO 5228 --- [ main] o.s.c.support.DefaultLifecycleProcessor : Starting beans in phase 0

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] o.s.c.n.e.server.EurekaServerBootstrap : Setting the eureka configuration..

2016-12-28 07:59:39.384 INFO 5228 --- [ main] c.n.e.EurekaDiscoveryClientConfiguration : Registering application unknown with eureka with status UP

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] o.s.c.n.e.server.EurekaServerBootstrap : Eureka data center value eureka.datacenter is not set, defaulting to default

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] o.s.c.n.e.server.EurekaServerBootstrap : Eureka environment value eureka.environment is not set, defaulting to test

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] o.s.c.n.e.server.EurekaServerBootstrap : isAws returned false

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] o.s.c.n.e.server.EurekaServerBootstrap : Initialized server context

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl : Got 1 instances from neighboring DS node

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl : Renew threshold is: 1

2016-12-28 07:59:39.384 INFO 5228 --- [ Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl : Changing status to UP

2016-12-28 07:59:39.400 INFO 5228 --- [ Thread-10] e.s.EurekaServerInitializerConfiguration : Started Eureka Server

2016-12-28 07:59:39.462 INFO 5228 --- [ main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8801 (http)

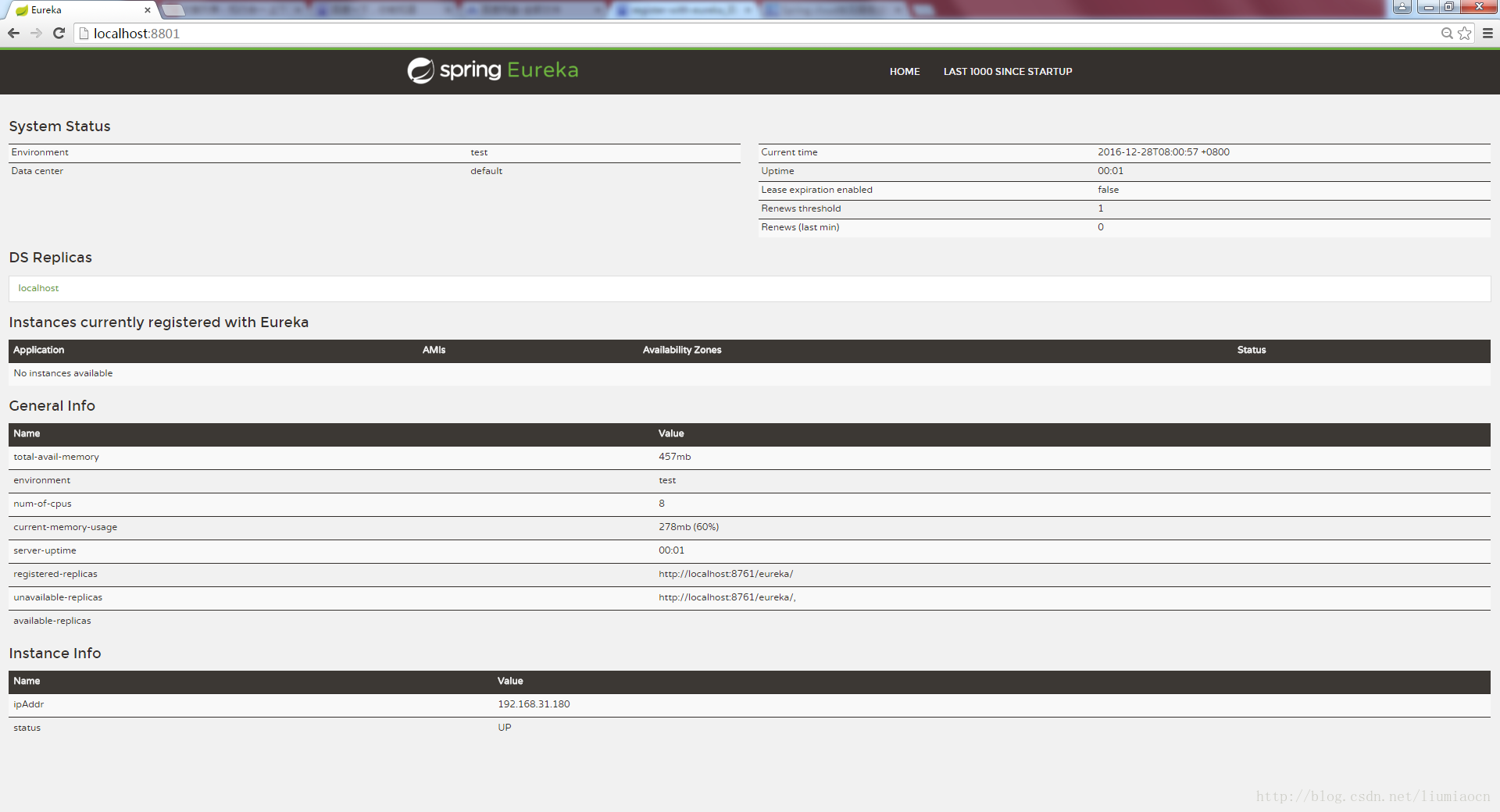
2016-12-28 07:59:39.462 INFO 5228 --- [ main] c.n.e.EurekaDiscoveryClientConfiguration : Updating port to 8801

2016-12-28 07:59:39.478 INFO 5228 --- [ main] c.l.d.s.DiscoveryserviceApplication : Started DiscoveryserviceApplication in 7.073 seconds (JVM running for 7.555)

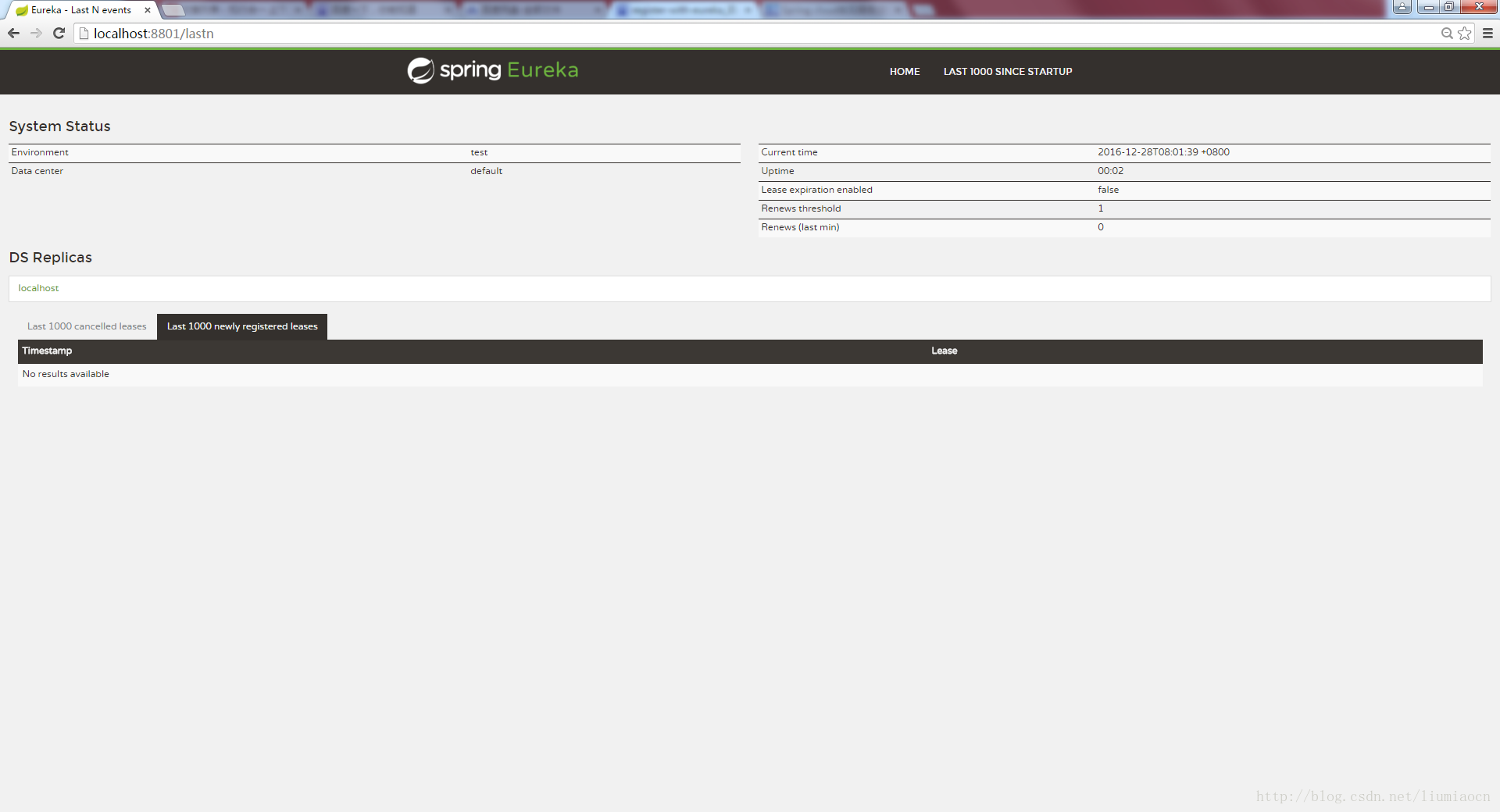
## Eureka Server

使用8081port就可以访问到Eureka Server的Dashboard了，从这里可以清楚地看到[Instances currently registered with Eureka]下还没有任何服务注册到这里。

## home



## lastn



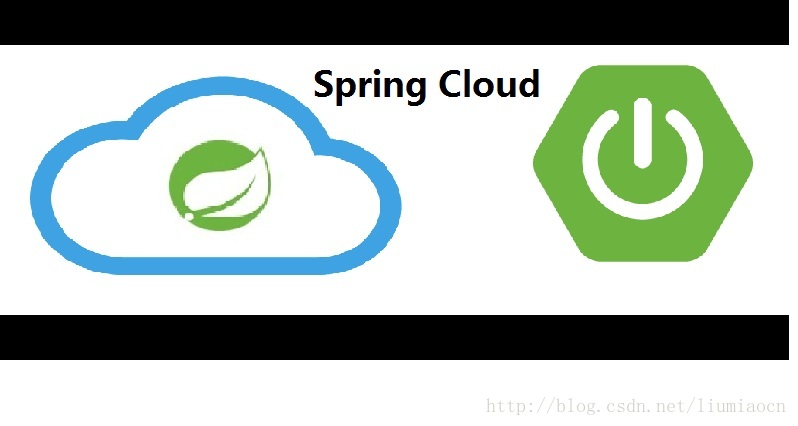
# 总结

这篇文章中我们了解到了如何简单的通过1行注解和3行设定即可启动服务发现功能的Eureka的Server端，在接下来的文章中，你将会看到Client端如何注册到Eureka的Server以及其他各种组件的使用方法。

### [Spring基础：快速入门spring cloud（3）：业务逻辑服务](http://blog.csdn.net/liumiaocn/article/details/53922951)

分类：**spring**

 （851）  （0）  举报  收藏

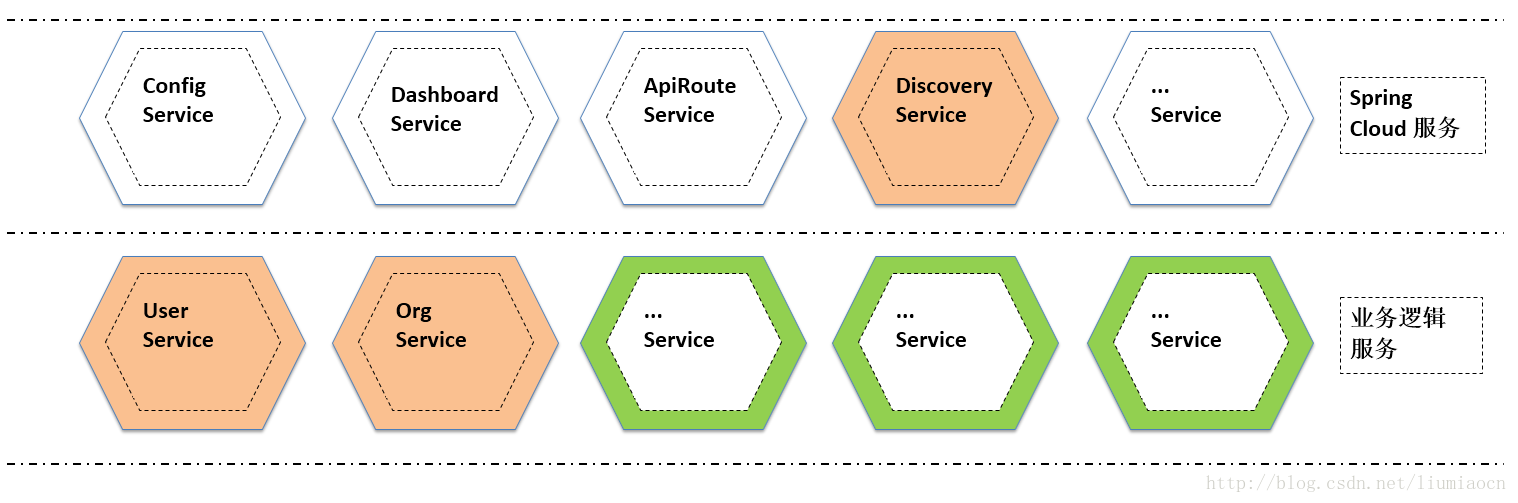


Spring Cloud是Spring总多的Project中的一个,它提供了一整套的工具帮助系统架构师们在进行分布式设计的时候可以拿来即用, 在创建和发布微服务时极为便捷和有效。本系列文章将会使用最简单的例子和最为容易的方式来学习Spring Cloud。本文种将会创建两个非常简单的service以, 并在此基础上介绍如何通过注解和设定向Eureka Server进行服务注册。

# 构成

| **项目** | **详细** |
| --- | --- |
| Config Service | Spring Cloud Config:统一配置管理服务 |
| Dashboard Service | Hystrix Dashboard |
| Api Route Service | Zuul:Api Gateway |
| Discovery Service | Eureka：服务发现 |
| User Service | RESTFUL的用户相关的服务 |
| Org Service | RESTFUL的组织相关的服务 |

# 业务逻辑服务



为了更加容易入门，此处使用在入门Spring boot的时候使用的Hello World级别的例子，Spring boot熟悉的可以直接跳过，只需直接确认注解和设定部分即可。

## User Service

### POM设定

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.liumiaocn.demo.springcloud</groupId>

<artifactId>userservice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>userservice</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</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>Camden.BUILD-SNAPSHOT</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-snapshots</id>

<name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

### 示例代码

package com.liumiaocn.demo.springcloud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication

@RestController

@RequestMapping("/user")

public class UserserviceApplication {

@RequestMapping("/detail")

public String Detail(){

return "## Detail information about : User ";

}

public static void main(String[] args) {

SpringApplication.run(UserserviceApplication.class, args);

}

}

### 注解

加入注解EnableEurekaClient，加入后如下所示。

package com.liumiaocn.demo.springcloud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication

@RestController

@RequestMapping("/user")

@EnableEurekaClient

public class UserserviceApplication {

@RequestMapping("/detail")

public String Detail(){

return "## Detail information about : User ";

}

public static void main(String[] args) {

SpringApplication.run(UserserviceApplication.class, args);

}

}

### 设定文件

使用bootstrap和application两个设定文件来进行相关设定，入门时为了更加快速，使用最少的设定文件和设定语句，基本上只有不可或缺的才会加上，入门之后详细内容参看spring cloud官方文档即可。application.properties文件的最少作如下设定即可。

server.port=9001

spring.application.name=UserService

eureka.client.serviceUrl.defaultZone=http://localhost:8801/eureka/

| **项目** | **详细** |
| --- | --- |
| server.port | User Service提供服务所用Port |
| spring.application.name | 向Eureka Server进行注册时使用的服务名，在后面我们将会在Dashboard上确认到 |
| eureka.client.serviceUrl.defaultZone | <http://localhost:8801/eureka/> 注意此处的８８０１端口号需要跟Server端一致。 |

### 生成Package

| **项目** | **详细** |
| --- | --- |
| 命令 | mvn clean package |
| 执行场所 | pom所在目录 |
| 目标文件所在目录 | 工程根目录下Target |
| 目标文件名称 | userservice-0.0.1-SNAPSHOT.jar |

## Org Service

### POM设定

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.liumiaocn.demo.springcloud</groupId>

<artifactId>orgservice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orgservice</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</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>Camden.BUILD-SNAPSHOT</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-snapshots</id>

<name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

### 示例代码

package com.liumiaocn.demo.springcloud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication

@RestController

@RequestMapping("/org")

@EnableEurekaClient

public class OrgserviceApplication {

@RequestMapping("/detail")

public String Detail(){

return "## The detail information about: Organization";

}

public static void main(String[] args) {

SpringApplication.run(OrgserviceApplication.class, args);

}

}

### 设定文件

server.port=9002

spring.application.name=OrgService

eureka.client.serviceUrl.defaultZone=http://localhost:8801/eureka/

### 生成Package

| **项目** | **详细** |
| --- | --- |
| 命令 | mvn clean package |
| 执行场所 | pom所在目录 |
| 目标文件所在目录 | 工程根目录下Target |
| 目标文件名称 | orgservice-0.0.1-SNAPSHOT.jar |

# 生成Eureka Server的Package

| **项目** | **详细** |
| --- | --- |
| 命令 | mvn clean package |
| 执行场所 | pom所在目录 |
| 目标文件所在目录 | 工程根目录下Target |
| 目标文件名称 | discoveryservice-0.0.1-SNAPSHOT.jar |

# 启动

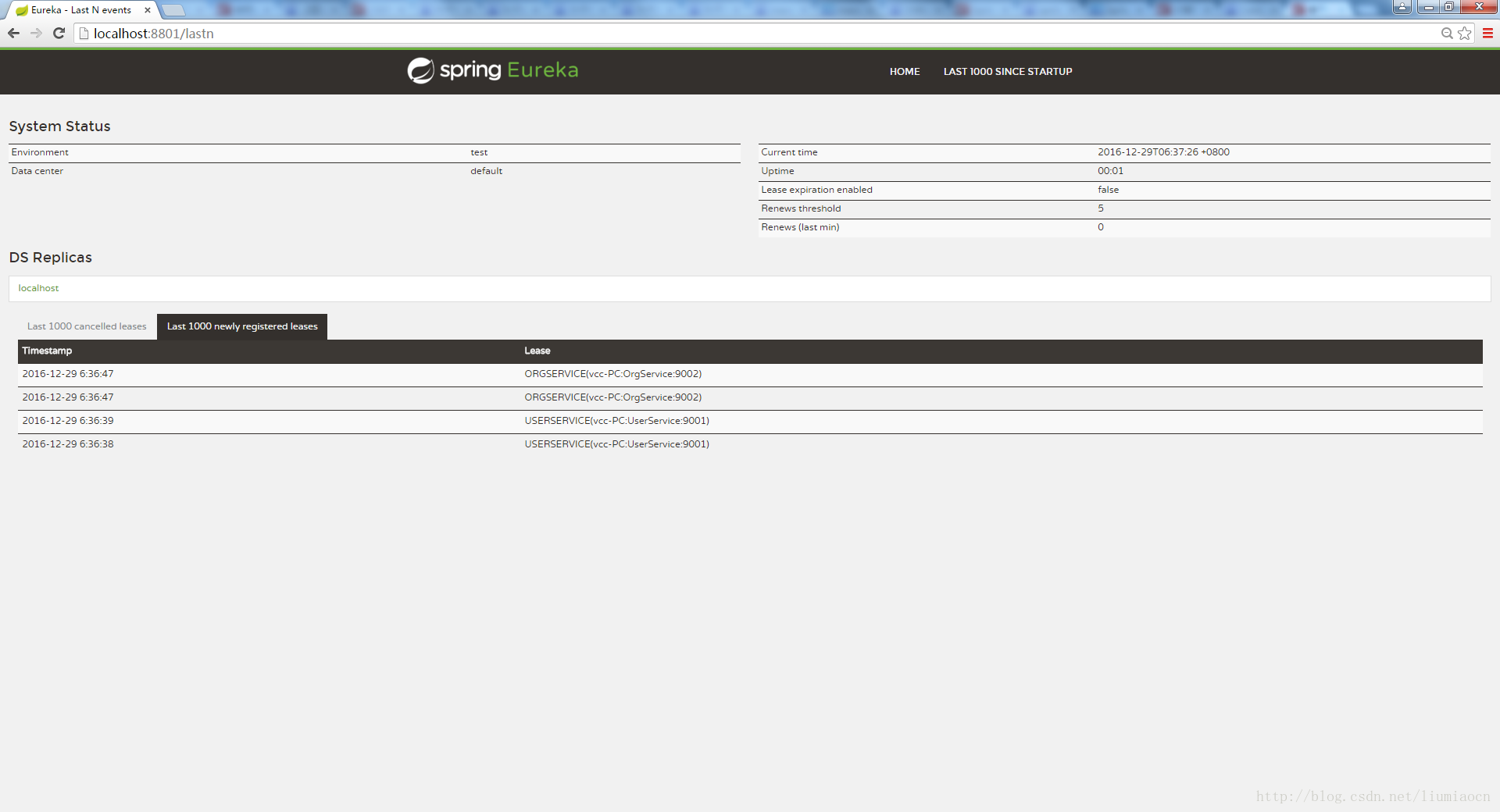
使用java -jar启动各个Service

| **项目** | **详细** |
| --- | --- |
| No.1 | java -jar discoveryservice-0.0.1-SNAPSHOT.jar |
| No.2 | java -jar userservice-0.0.1-SNAPSHOT.jar |
| No.3 | java -jar orgservice-0.0.1-SNAPSHOT.jar |

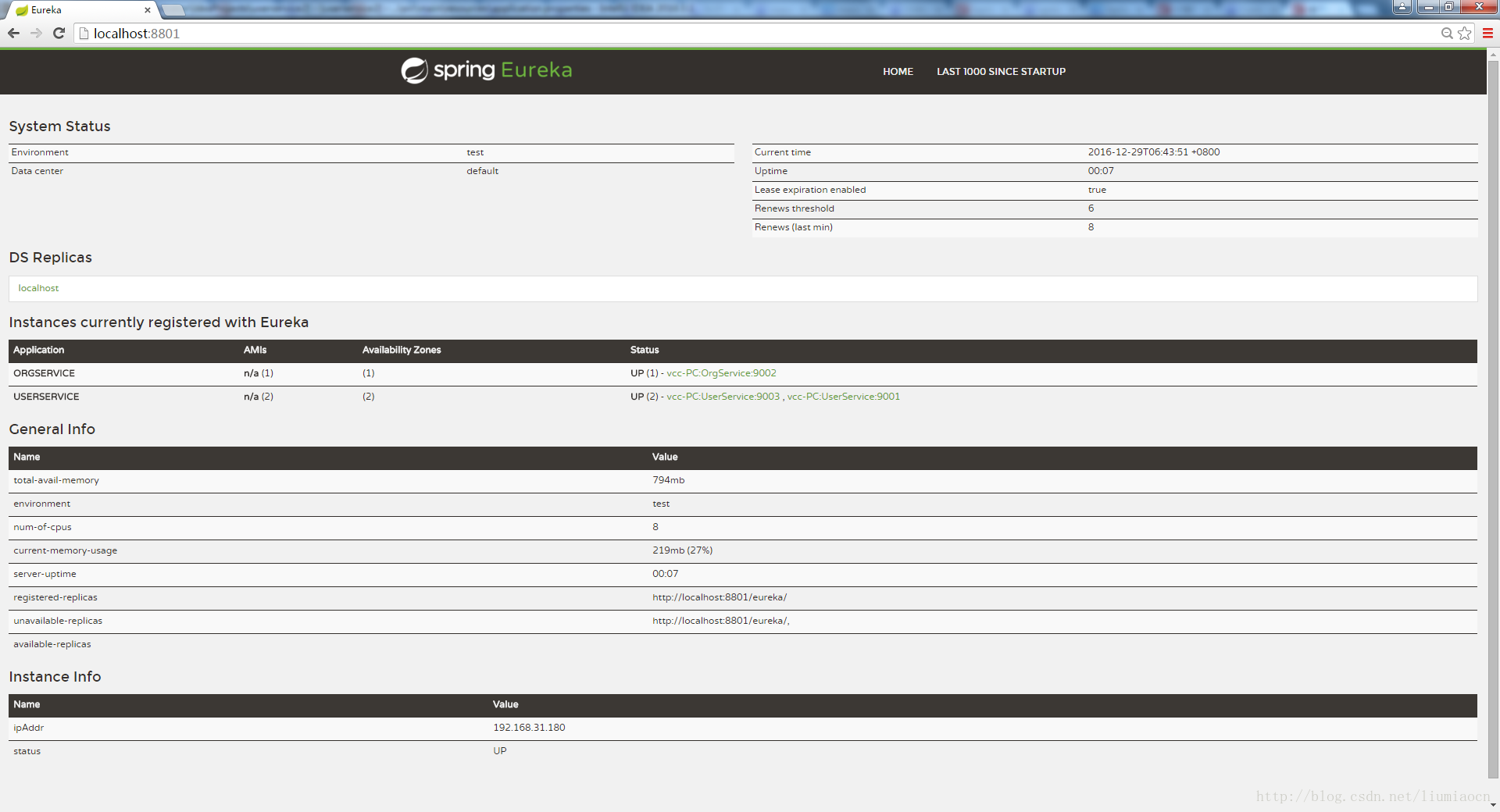
# 确认结果

＞我们再次确认一下服务注册之后Eureka Server的Dashboard

## HOME



## LASTN



# 服务的多重实例

＞过去我们在使用面向服务的架构时，服务的多重启动管理和负债的均衡都是需要考虑的事情，而现在流行的微服务和容器的结合更是诗的服务多重启动等管理变得无比简单，本来我们只需要将UserService在不同的容器中启动即可，但是考虑到那样至少在Spring Cloud的教程中必须安装Docker或者多台机器或者虚拟机才可以试验。为了简单摹拟，我们将UserService的工程重新创建一边，内容完全一样，只修改port，这样也可以为后面的负载均衡的试验提供一个简单的基础。

## POM

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.liumiaocn.demo.springcloud</groupId>

<artifactId>userservice2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>userservice2</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</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>Camden.BUILD-SNAPSHOT</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-snapshots</id>

<name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

## Demo代码

package com.liumiaocn.demo.springcloud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication

@RequestMapping("/user")

@EnableEurekaClient

@RestController

public class Userservice2Application {

@RequestMapping("/detail")

public String Detail(){

return "## The detail information about: User";

}

public static void main(String[] args) {

SpringApplication.run(Userservice2Application.class, args);

}

}

## 设定

server.port=9003

spring.application.name=UserService

eureka.client.serviceUrl.defaultZone=http://localhost:8801/eureka/

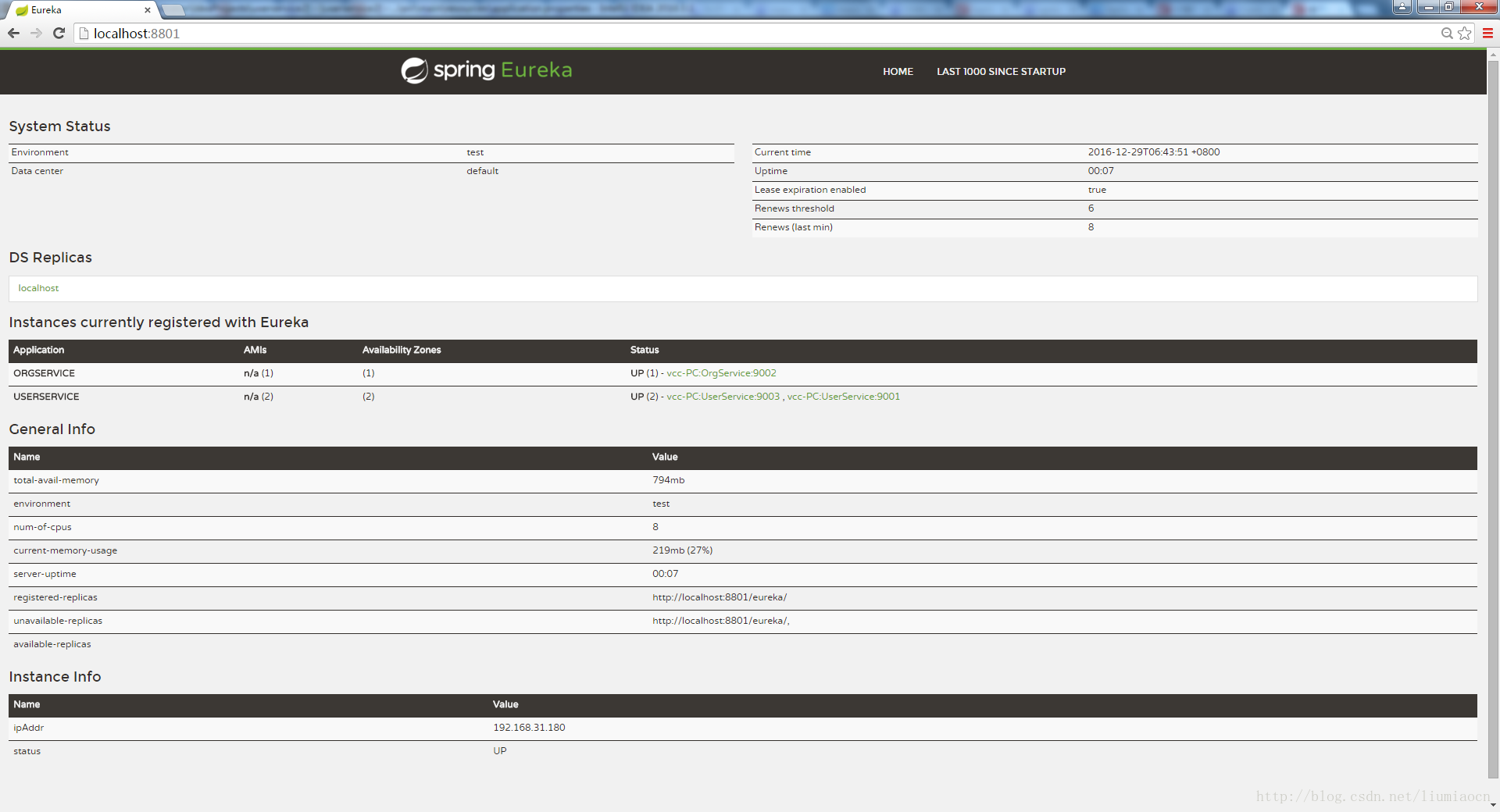
**启动**

| **项目** | **详细** |
| --- | --- |
| 命令 | mvn clean package |
| 执行场所 | pom所在目录 |
| 目标文件所在目录 | 工程根目录下Target |
| 目标文件名称 | userservice2-0.0.1-SNAPSHOT.jar |
| 启动命令 | java -jar userservice2-0.0.1-SNAPSHOT.jar |

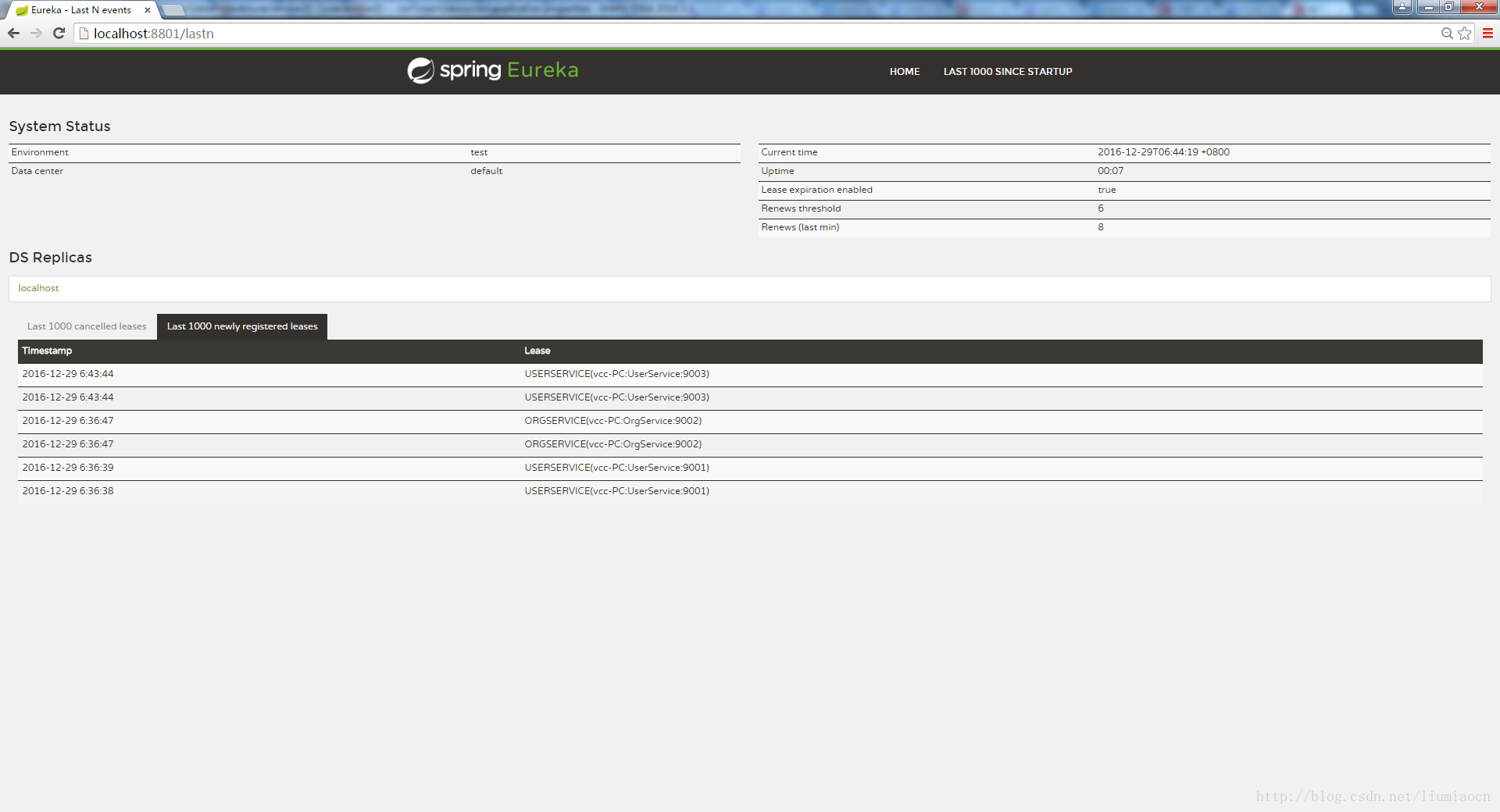
## 结果确认

### HOME

注册的USERSERVICE可以清楚地看到与ORGSERVICE的不同，它将9002和9003提供的同一服务都列了出来。



### LASTN



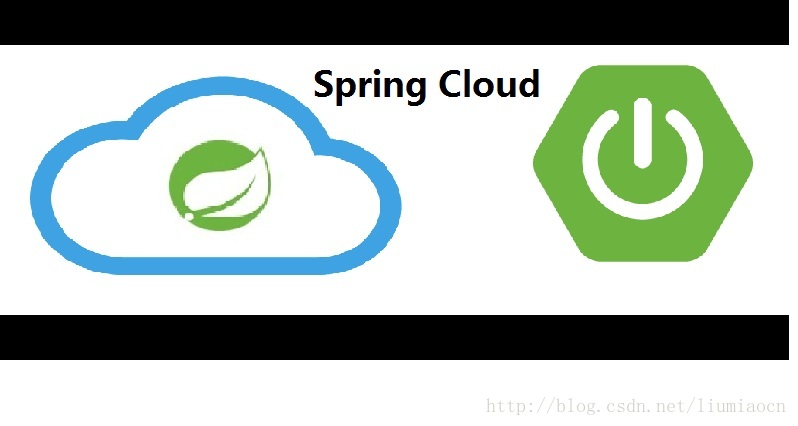
# 总结

＞在这篇文章中，我们继续了解了服务注册以及多个服务同时注册到Eureka上的场景，在接下来的文章中我们将继续学习如何使用这些方便好用的Spring Cloud的各种功能。

### [Spring基础：快速入门spring cloud（4）：API网关之Zuul](http://blog.csdn.net/liumiaocn/article/details/53941354)

分类：**spring**

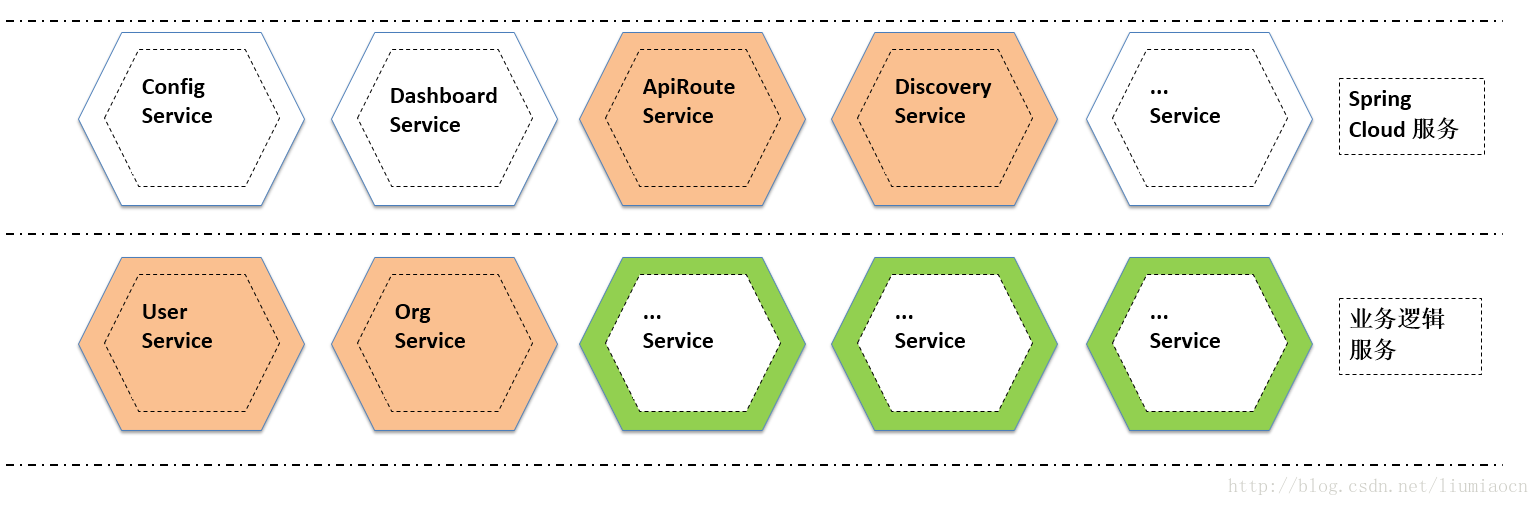
 （1692）  （1）  举报  收藏



Spring Cloud是Spring总多的Project中的一个,它提供了一整套的工具帮助系统架构师们在进行分布式设计的时候可以拿来即用, 在创建和发布微服务时极为便捷和有效。   
本系列文章将会使用最简单的例子和最为容易的方式来学习Spring Cloud。本文将会介绍如何引入Zuul在微服务的架构中简单实现API网关。

# 构成

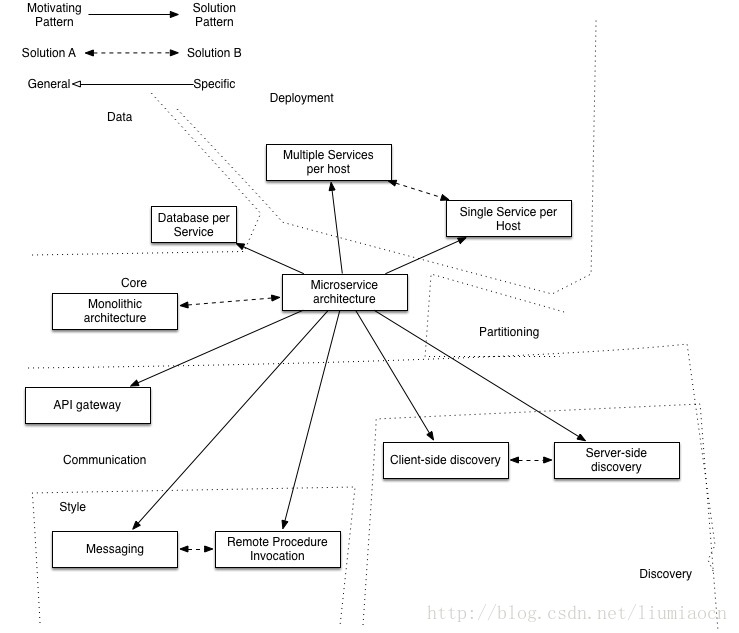
| **项目** | **详细** |
| --- | --- |
| Config Service | Spring Cloud Config:统一配置管理服务 |
| Dashboard Service | Hystrix Dashboard |
| Api Route Service | Zuul:Api Gateway |
| Discovery Service | Eureka：服务发现 |
| User Service | RESTFUL的用户相关的服务 |
| Org Service | RESTFUL的组织相关的服务 |



# 微服务架构

＞Api Gateway是微服务架构中需要考虑的情况之一，详细信息此处不再展开有兴趣的可以查看下面这篇文章

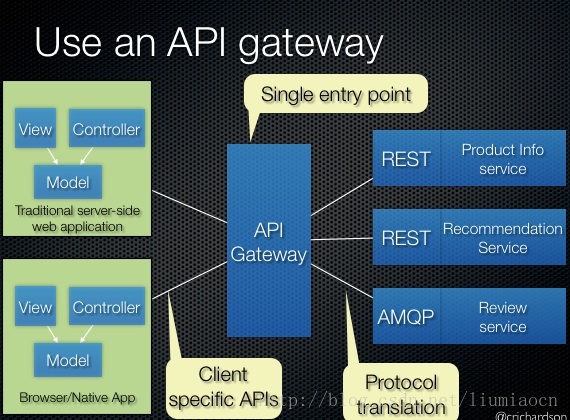
| **URL** | <http://microservices.io/patterns/microservices.html> |
| --- | --- |



# Api Gateway

在很多的场景下，尽量少的修改代码或者不修改代码而将不同服务的入口进行统一管理，将API网关作为用户的唯一入口，可以参看Richardson的文章里面的例子来对此概念的展开进行进一步地理解。

| **URL** | <http://microservices.io/patterns/apigateway.html> |
| --- | --- |

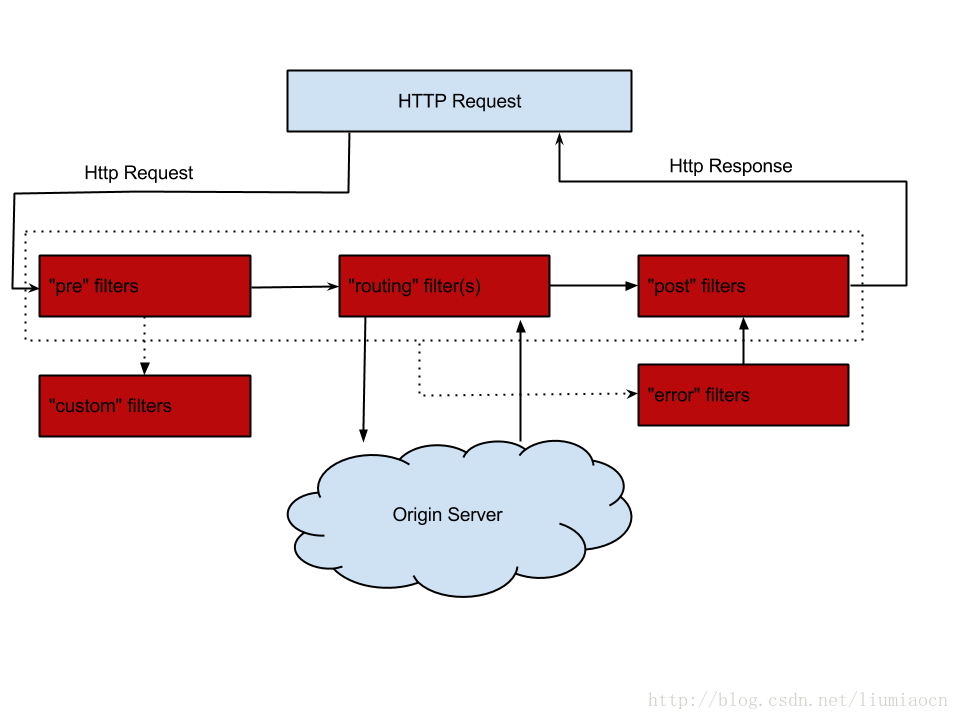


# Zuul介绍

＞Zuul 是Spring Cloud 子项目Spring Cloud Netflix的一个组件，它是Netflix对ApiGateway实现的一份答卷，应用非常广泛。常见的功能如下

* 身份验证
* 压力测试
* Canary Testing
* 动态路由
* 安全控制   
  ．．．

＞Zuul作为中间这一层，如何处理相关请求。



# 场景

＞使用我们前面创建的例子，我们有user和 org两个服务，分别在９００１和９００２端口提供服务，我们如果想要查询用户和组织的详细信息，一般是通过如下方式进行访问的。

| **项目** | **访问方法（ＵＲＬ）** |
| --- | --- |
| 用户详细 | <http://localhost:9001/user/detail> |
| 组织详细 | <http://localhost:9002/org/detail> |

# Pom详细

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.liumiaocn.demo.springcloud</groupId>

<artifactId>apirouteservice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>apirouteservice</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-ribbon</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-zuul</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</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>Camden.BUILD-SNAPSHOT</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-snapshots</id>

<name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

# apirouteservice

只需要加入EnableZuulProxy注解。

package com.liumiaocn.demo.springcloud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication

@EnableEurekaClient

@EnableZuulProxy

public class ApirouteserviceApplication {

public static void main(String[] args) {

SpringApplication.run(ApirouteserviceApplication.class, args);

}

}

# 设定文件

server.port=8001

spring.application.name=apirouteservice

eureka.client.serviceUrl.defaultZone=http://localhost:8801/eureka/

zuul.routes.userservice.path=/user/\*\*

zuul.routes.userservice.strPrefix=true

zuul.routes.userservice.url=http://localhost:9001/user

zuul.routes.orgservice.path=/org/\*\*

zuul.routes.orgservice.strPrefix=true

zuul.routes.orgservice.url=http://localhost:9002/org

| **项目** | **详细** |
| --- | --- |
| server.port | apirouteService提供服务所用Port,将作为用户统一端口访问各个服务 |
| spring.application.name | 向Eureka Server进行注册时使用的服务名 |
| eureka.client.serviceUrl.defaultZone | <http://localhost:8801/eureka/> 注意此处的８８０１端口号需要跟Server端一致。 |
| zuul.routes.服务名.path | 注意此处服务名要与注册的服务名一致 |
| zuul.routes.服务名.strPrefix | 是否清除前缀（false |
| zuul.routes.服务名.url | 注意此处服务名要与注册的服务名一致，同时ｕｒｌ的层次也需要注意 |

# 生成Package

| **项目** | **详细** |
| --- | --- |
| 命令 | mvn clean package |
| 执行场所 | pom所在目录 |
| 目标文件所在目录 | 工程根目录下Target |
| 目标文件名称 | apirouteservice-0.0.1-SNAPSHOT.jar |

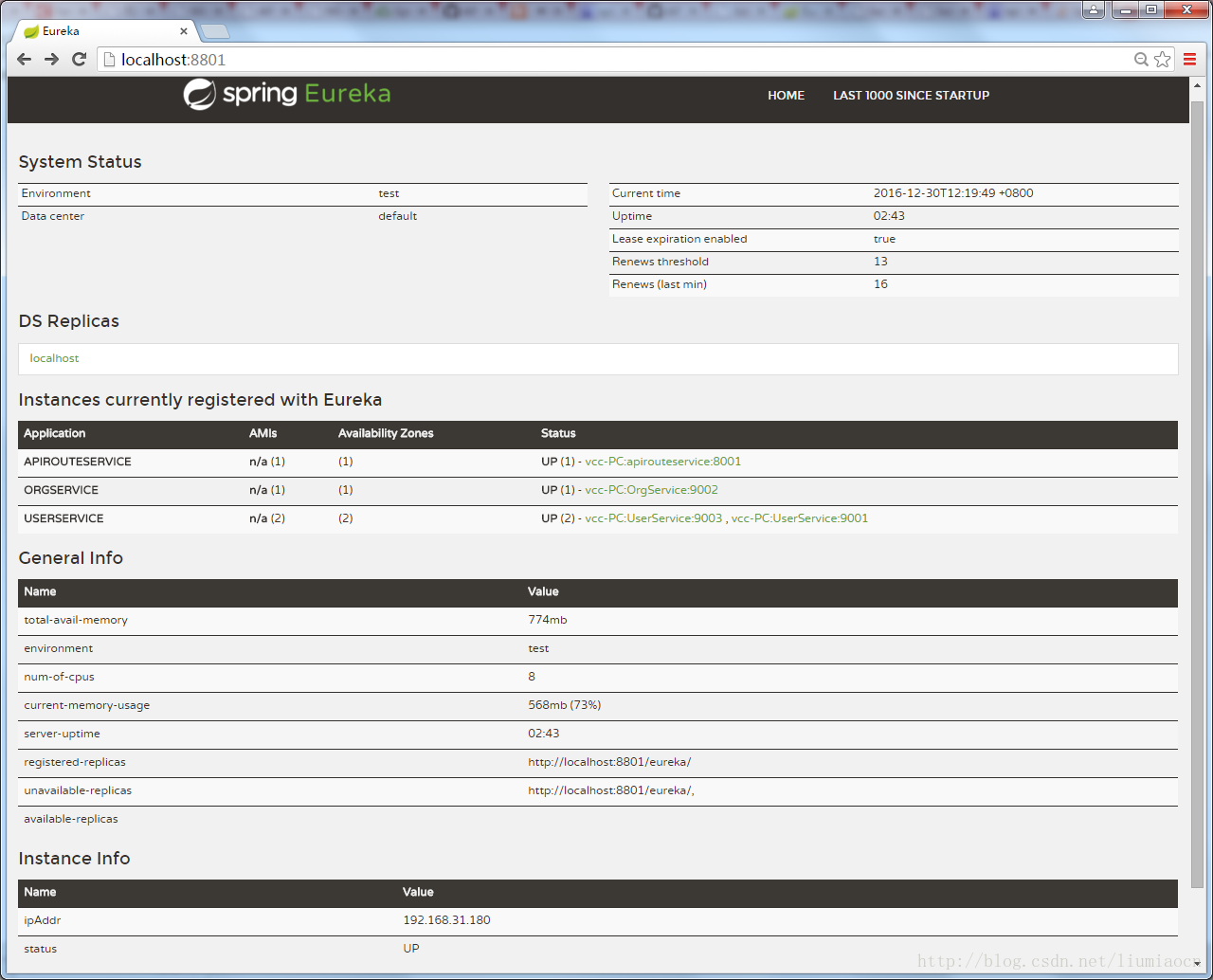
# 统一访问方式

| **项目** | **访问方法（ＵＲＬ）** |
| --- | --- |
| 用户详细 | <http://localhost:8001/user/detail> |
| 组织详细 | <http://localhost:8001/org/detail> |

# 启动

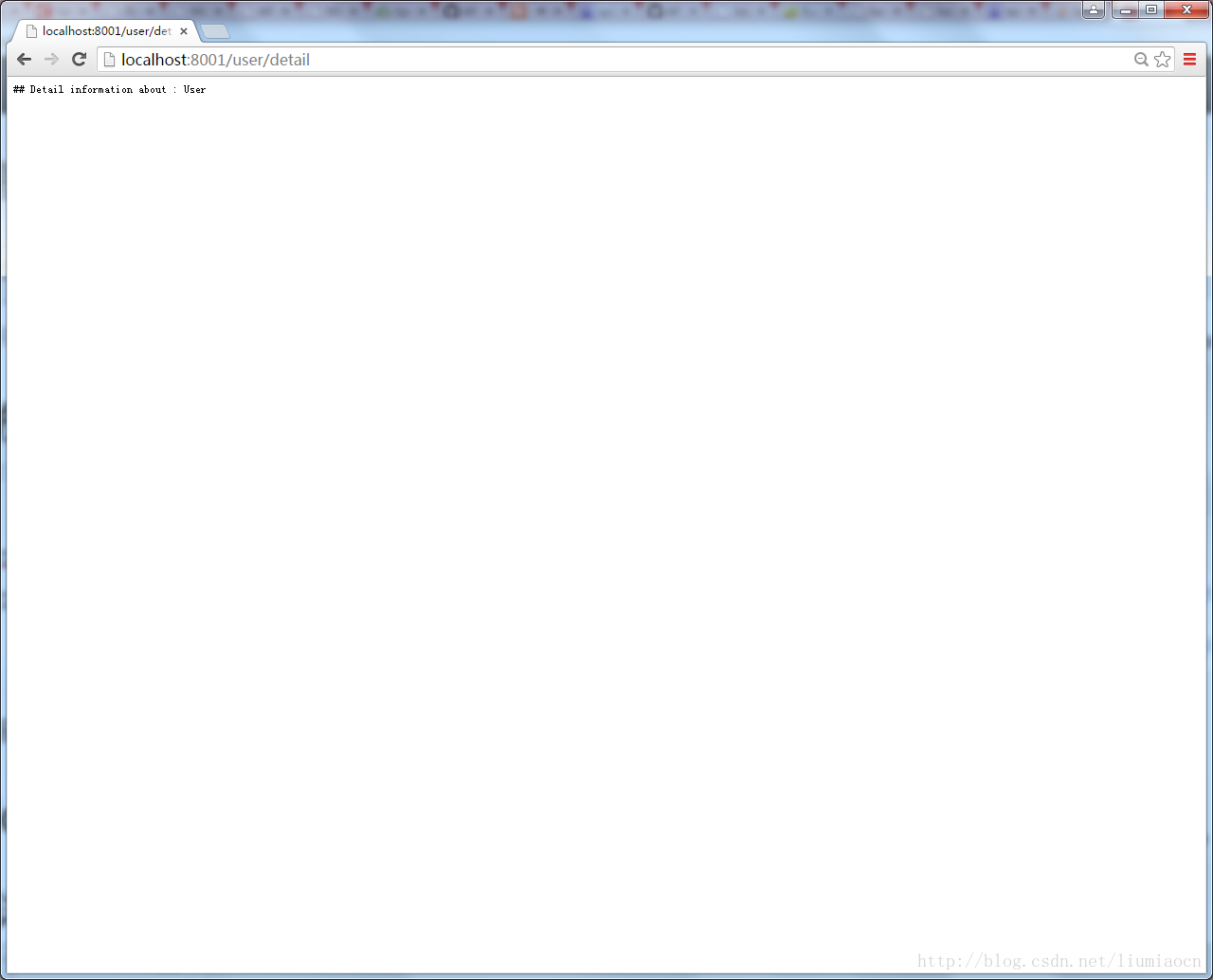
＞启动各个服务后，也启动apirouteservice

# Eureka确认



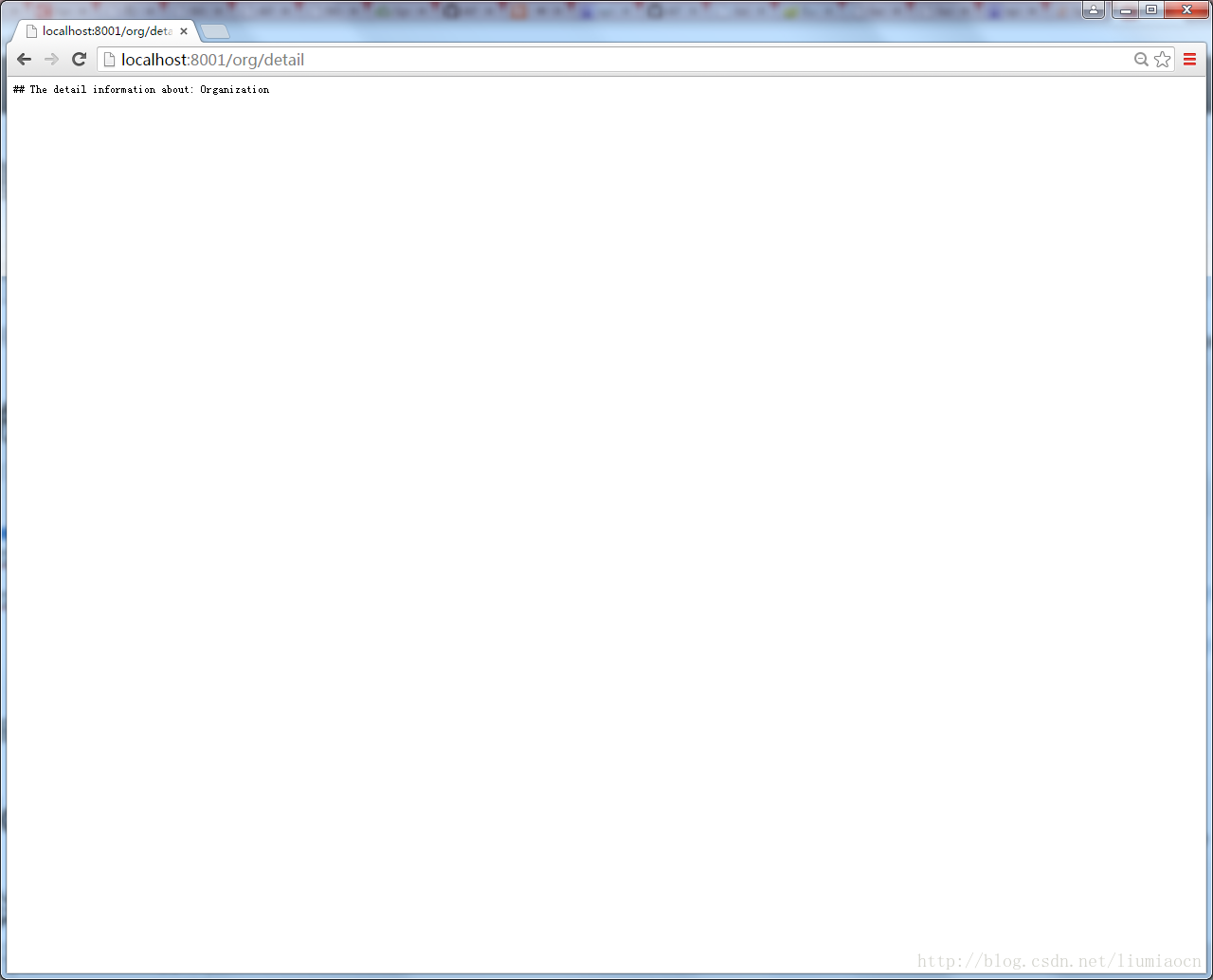
# User详细信息

＞通过Zuul统一访问用户详细信息



# Org详细信息、

＞通过Zuul统一访问组织详细信息



# 总结

＞本文使用zuul实现了Api网关最简单的功能，但是很重要的是我们了解到了如何获取封装这一层的方法，可以分发页面就可以控制负载等，在接下来的文章里我们会继续学习Spring Cloud的各种其他强大功能模块。