

Spring Cloud Eureka是Spring Cloud Netflix项目下的服务治理模块。而Spring Cloud Netflix项目是Spring Cloud的子项目之一，主要内容是对Netflix公司一系列开源产品的包装，它为Spring Boot应用提供了自配置的Netflix OSS整合。

eureka区别于之前用的比较多的zookeeper和后面要讲的consul，它没有提供服务端，所以得我们自己搭建一个。

所以我们先简单的搭建一个eureka服务端，首先我们创建一个SpringBoot的工程，并在pom中进行如下配置。

```
<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>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.10.RELEASE</version>
  </parent>

  <groupId>com.xyz</groupId>
  <artifactId>eureka-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>eureka-server</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <jdk.version>1.8</jdk.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-eureka-server</artifactId>
    </dependency>
  </dependencies>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
```

```

        <artifactId>spring-cloud-dependencies</artifactId>
        <version>Dalston.SR1</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.3</version>
            <configuration>
                <!-- 指定source和target的版本 -->
                <source>${jdk.version}</source>
                <target>${jdk.version}</target>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

```

可以看见主要就是引入了spring-cloud-starter-eureka-server这个jar包，他就提供了eureka服务端运行所需要的东西了。

接下来我们需要创建一个启动类，区别于之前我们创建boot的启动类的区别就是这里。

```

@EnableEurekaServer
@SpringBootApplication
public class Application {

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

}

```

通过@EnableEurekaServer注解启动一个服务注册中心提供给其他应用进行对话。

然后还得进行相关的配置，直接在application.properties中配置即可。

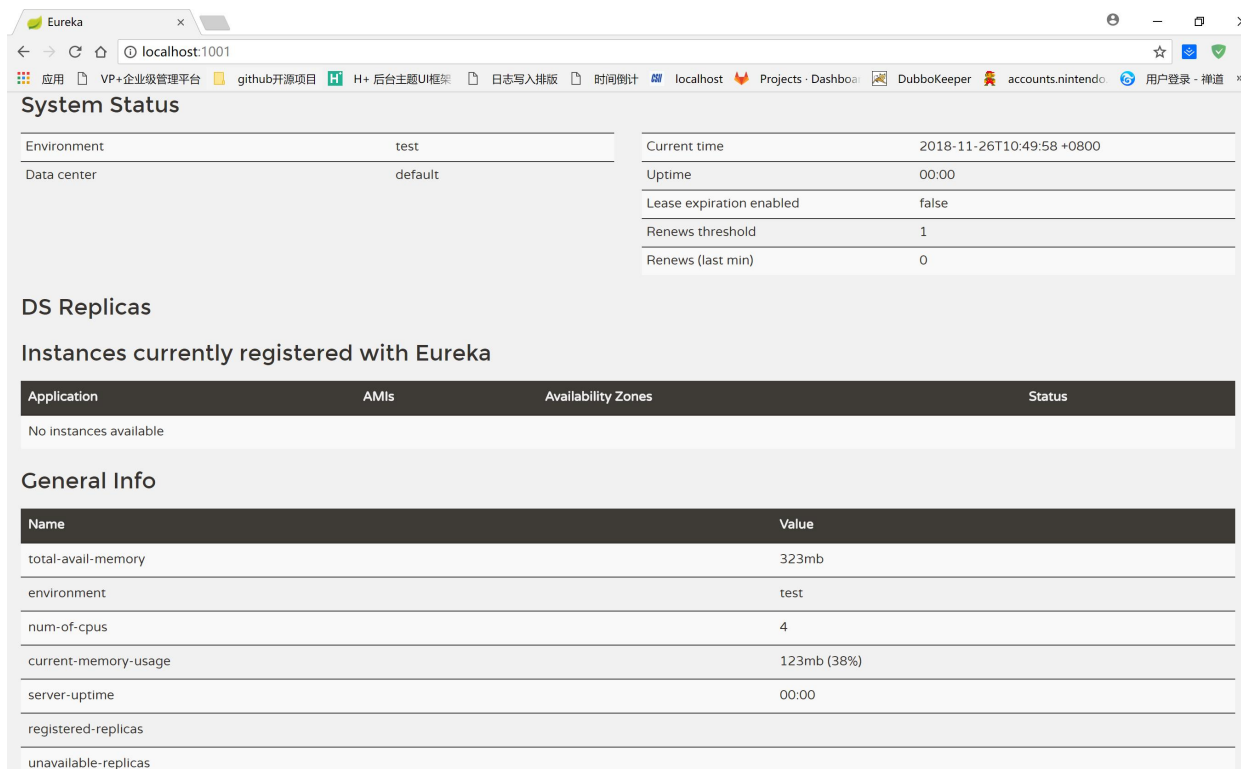
```

spring.application.name=eureka-server
server.port=1001

```

```
#实例名称
eureka.instance.hostname=localhost
#是否向注册中心注册自己
eureka.client.register-with-eureka=false
#是否需要检索服务
eureka.client.fetch-registry=false
```

然后我们要做的就是启动啦，启动完在浏览器输入localhost:1001即可访问了。



The screenshot shows the Eureka web interface in a browser window. The address bar shows 'localhost:1001'. The page has a header with 'Eureka' and a navigation bar with various links. The main content area is divided into several sections:

- System Status**: A table showing system information.

Environment	test	Current time	2018-11-26T10:49:58 +0800
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0
- DS Replicas**: A section for distributed storage replicas.
- Instances currently registered with Eureka**: A table showing registered instances.

Application	AMIs	Availability Zones	Status
No instances available			
- General Info**: A table showing general system information.

Name	Value
total-avail-memory	323mb
environment	test
num-of-cpus	4
current-memory-usage	123mb (38%)
server-uptime	00:00
registered-replicas	
unavailable-replicas	

至此，eureka的服务端（注册中心）搭建也就完成了，下面就要介绍如何注册和发现服务了。

紧接着我们创建服务提供者，同样的创建一个SpringBoot的工程，看看这里的pom。

```
<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>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.10.RELEASE</version>
    </parent>
```

```
<groupId>com.xyz</groupId>
<artifactId>eureka-client</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>eureka-client</name>
<url>http://maven.apache.org</url>

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <jdk.version>1.8</jdk.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>
</dependencies>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Dalston.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.3</version>
      <configuration>
        <!-- 指定source和target的版本 -->
        <source>${jdk.version}</source>
        <target>${jdk.version}</target>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

然后看一下启动类。

```
@EnableDiscoveryClient
@SpringBootApplication
@ComponentScan("com.xyz.controller")
public class Application {

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

}
```

@EnableDiscoveryClient注解，该注解能激活Eureka中的DiscoveryClient实现，这样才能实现Controller中对服务信息的输出。

然后还得进行相关的配置。

```
spring.application.name=eureka-client
server.port=1002

eureka.client.serviceUrl.defaultZone=http://localhost:1001/eureka/
```

通过spring.application.name属性，我们可以指定微服务的名称后续在调用的时候只需要使用该名称就可以进行服务的访问。

eureka.client.serviceUrl.defaultZone属性对应服务注册中心的配置内容，指定服务注册中心的位置。

下面列出测试用的controller。

```
@RestController
public class DemoController {

    @Autowired
    private DiscoveryClient discoveryClient;

    @GetMapping("/test")
    public String dc() {
        String services = "Services: " + discoveryClient.getServices();
        System.out.println(services);
        return services;
    }

}
```

我们再把该工程启动。看一下eureka的管理界面。

Eureka

localhost:1001

应用 VP+企业级管理平台 github开源项目 H+ 后台主题UI框架 日志写入排版 时间倒计时 localhost Projects - Dashboa DubboKeeper accounts.nintendo 用户登录 - 禅道

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2018-11-26T11:23:54 +0800
Data center	default	Uptime	00:00
		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	n/a (1)	(1)	UP (1) - 192.168.1.112:eureka-client:1002

General Info

Name	Value
total-avail-memory	321mb
environment	test
num-of-cpus	4
current-memory-usage	51mb (15%)

可以看到我们的提供者已经注册过来了，因为提供者本身就有 controller，我们直接访问<http://localhost:1002/test>看看结果。

localhost:1002/test

localhost:1002/test

应用 VP+企业级管理平台 github开源项目

Services: [eureka-client]