

## Spring Cloud简介

Spring Cloud是一个基于Spring Boot实现的云应用开发工具，它为基于JVM的云应用开发中的配置管理、服务发现、断路器、智能路由、微代理、控制总线、全局锁、决策竞选、分布式会话和集群状态管理等操作提供了一种简单的开发方式。

Spring Cloud包含了多个子项目（针对分布式系统中涉及的多个不同开源产品），比如：

Spring Cloud Config、Spring Cloud Netflix、Spring Cloud CloudFoundry、Spring Cloud AWS、Spring Cloud Security、Spring Cloud Commons、Spring Cloud Zookeeper、Spring Cloud CLI等项目。

## 服务发布与消费

### 创建服务注册中心

创建一个基础的Spring Boot工程，这里命名为springcloud-eureka，并在 `pom.xml` 文件中添加以下依赖：

```
1. <parent>
2.     <groupId>org.springframework.boot</groupId>
3.     <artifactId>spring-boot-starter-parent</artifactId>
4.     <version>1.3.5.RELEASE</version>
5.     <relativePath />
6. </parent>
7. <properties>
8.     <!-- spring-cloud版本号 -->
9.     <spring.cloud.version>Brixton.SR5</spring.cloud.version>
10. </properties>
11. <dependencyManagement>
12.     <dependencies>
13.         <dependency>
14.             <groupId>org.springframework.cloud</groupId>
15.             <artifactId>spring-cloud-dependencies</artifactId>
16.             <version>${spring.cloud.version}</version>
17.             <type>pom</type>
18.             <scope>import</scope>
19.         </dependency>
20.     </dependencies>
21. </dependencyManagement>
22. <dependencies>
23.     <dependency>
24.         <groupId>org.springframework.boot</groupId>
25.         <artifactId>spring-boot-starter-test</artifactId>
26.         <scope>test</scope>
27.     </dependency>
28.     <dependency>
29.         <groupId>org.springframework.cloud</groupId>
30.         <artifactId>spring-cloud-starter-eureka-server</artifactId>
31.     </dependency>
32. </dependencies>
33. <build>
34.     <finalName>springcloud-eureka</finalName>
35.     <plugins>
36.         <!-- 设定maven所用jre版本避免maven update project时项目jre变为其它版
37.         <plugin>
38.             <groupId>org.apache.maven.plugins</groupId>
39.             <artifactId>maven-compiler-plugin</artifactId>
40.             <configuration>
41.                 <source>1.7</source>
42.                 <target>1.7</target>
43.                 <encoding>UTF-8</encoding>
44.             </configuration>
45.         </plugin>
46.         <plugin>
47.             <groupId>org.springframework.boot</groupId>
48.             <artifactId>spring-boot-maven-plugin</artifactId>
49.         </plugin>
50.     </plugins>
51. </build>
```

启动一个服务注册中心，只需在Spring Boot的启动类中添加 `@EnableEurekaServer` 注解：

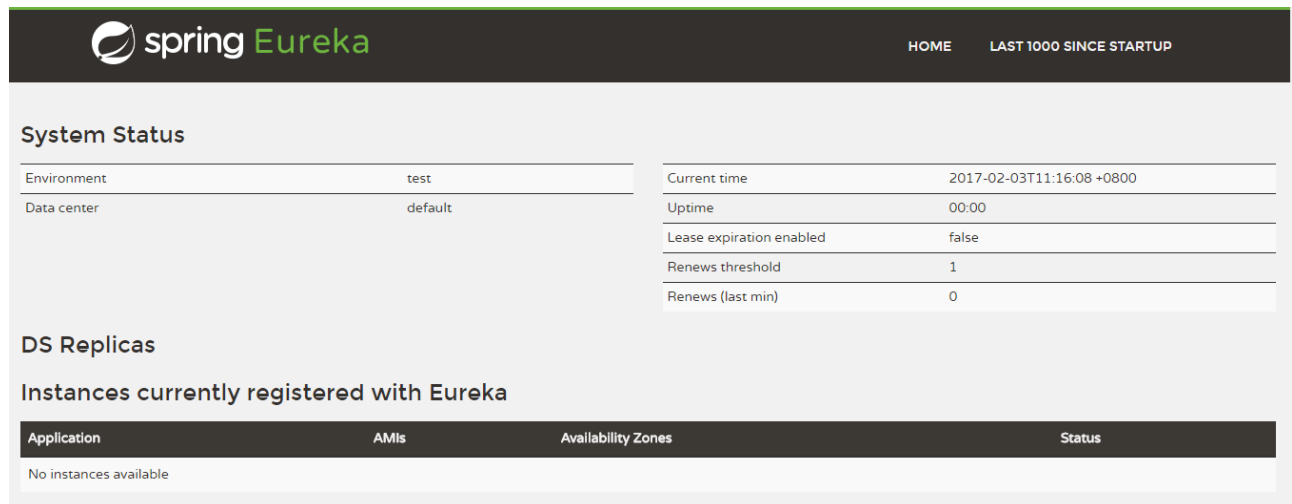
```
1. @EnableEurekaServer
2. @SpringBootApplication
3. public class EurekaApplication {
4.     public static void main(String[] args) {
5.         new SpringApplicationBuilder(EurekaApplication.class).web(true).run(args);
6.     }
7. }
```

默认情况下，该服务注册中心也会将自己作为客户端来注册它自己，所以需要禁用它的注册行为，只需要在 `application.properties` 中添加以下配置：

```
1. server.port=1111
2. eureka.client.register-with-eureka=false
3. eureka.client.fetch-registry=false
4. eureka.client.serviceUrl.defaultZone=http://localhost:${server.port}/eureka/
```

第二、第三行把默认注册行为设置为false，第四行设置注册中心的地址。

启动工程后，访问：<http://localhost:1111/>



The screenshot shows the Spring Eureka web interface. At the top, there's a header with the Spring Eureka logo and navigation links for HOME and LAST 1000 SINCE STARTUP. Below the header, the 'System Status' section contains two tables. The left table shows 'Environment' as 'test' and 'Data center' as 'default'. The right table shows 'Current time' as '2017-02-03T11:16:08 +0800', 'Uptime' as '00:00', 'Lease expiration enabled' as 'false', 'Renews threshold' as '1', and 'Renews (last min)' as '0'. Below the system status, the 'DS Replicas' section is empty. The 'Instances currently registered with Eureka' section shows a table with headers 'Application', 'AMIs', 'Availability Zones', and 'Status'. The table body contains the text 'No instances available'.

可以看到目前还没有发现任何服务。

## 创建服务提供方

创建一个提供服务的客户端，并向服务注册中心注册，即对外发布服务。

假设我们有一个提供计算功能的微服务模块，通过传入两个参数a和b，计算a+b的结果并返回。

首先，创建一个基础的Spring Boot工程，这里命名为springcloud-service，并在 `pom.xml` 文件中添加以下依赖：

```

1. <parent>
2.   <groupId>org.springframework.boot</groupId>
3.   <artifactId>spring-boot-starter-parent</artifactId>
4.   <version>1.3.5.RELEASE</version>
5.   <relativePath/> <!-- lookup parent from repository -->
6. </parent>
7. <dependencies>
8.   <dependency>
9.     <groupId>org.springframework.boot</groupId>
10.    <artifactId>spring-boot-starter-test</artifactId>
11.    <scope>test</scope>
12.  </dependency>
13.  <dependency>
14.    <groupId>org.springframework.cloud</groupId>
15.    <artifactId>spring-cloud-starter-eureka</artifactId>
16.  </dependency>
17. </dependencies>
18. <dependencyManagement>
19.   <dependencies>
20.     <dependency>
21.       <groupId>org.springframework.cloud</groupId>
22.       <artifactId>spring-cloud-dependencies</artifactId>
23.       <version>Brixton.RELEASE</version>
24.       <type>pom</type>
25.       <scope>import</scope>
26.     </dependency>
27.   </dependencies>
28. </dependencyManagement>

```

其次，创建一个 `controller`，简单实现 `/add` 接口：

```

1. @RestController
2. public class ComputeController {
3.     @RequestMapping(value = "/add", method = RequestMethod.GET)
4.     public Integer add(@RequestParam Integer a, @RequestParam Integer b) {
5.         Integer r = a + b;
6.         return r;
7.     }
8. }

```

要向注册中心注册服务，只需在启动类中添加 `@EnableDiscoveryClient` 注解：

```

1. @EnableDiscoveryClient
2. @SpringBootApplication
3. public class ComputeServiceApplication {
4.     public static void main(String[] args) {
5.         new SpringApplicationBuilder(ComputeServiceApplication.class).web(true)
6.         .run(args);
7.     }
}

```

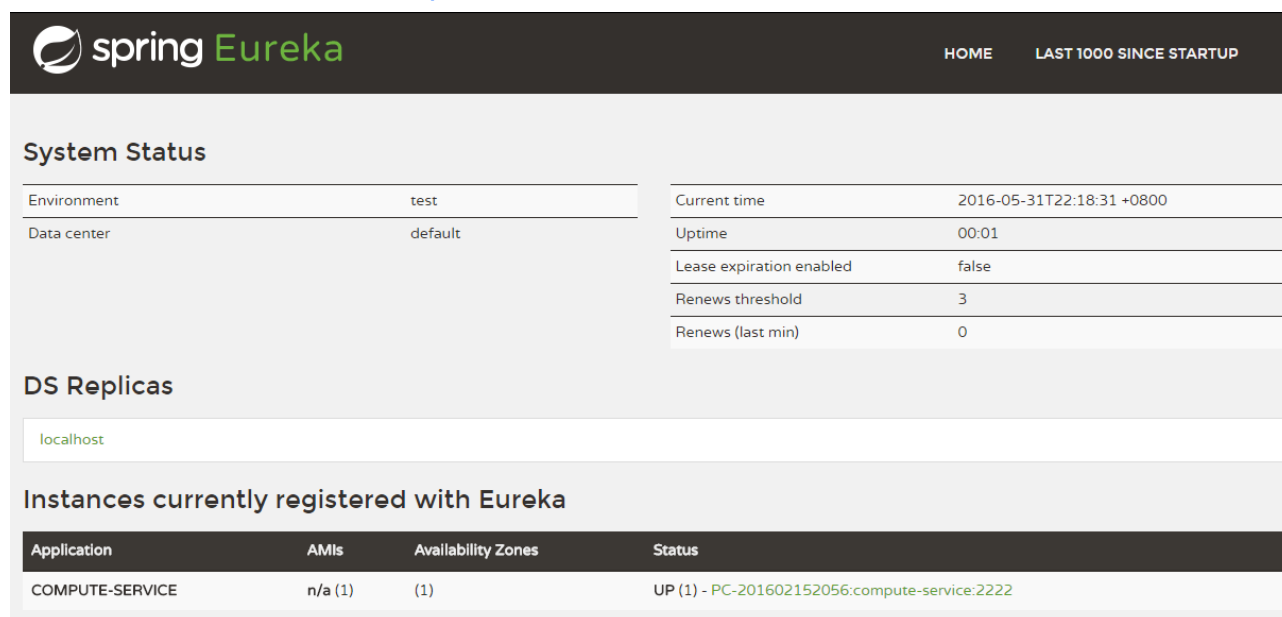
最后，需要配置微服务的名称和注册中心的地址：

1. `spring.application.name=compute-service`
2. `server.port=2222`
3. `eureka.client.serviceUrl.defaultZone=http://localhost:1111/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。

启动该工程后，再次访问：<http://localhost:1111/>



The screenshot displays the Spring Eureka web interface. At the top, there's a navigation bar with the Spring Eureka logo and links for HOME and LAST 1000 SINCE STARTUP. The main content area is divided into three sections:

- System Status:** A table showing environment (test), data center (default), current time (2016-05-31T22:18:31 +0800), uptime (00:01), lease expiration enabled (false), renew threshold (3), and renews (last min) (0).
- DS Replicas:** A section showing the local host (localhost).
- Instances currently registered with Eureka:** A table with columns Application, AMIs, Availability Zones, and Status. It shows one instance: COMPUTE-SERVICE, n/a (1), (1), and UP (1) - PC-201602152056:compute-service:2222.

可以看到，COMPUTE-SERVICE的服务已经被注册了。

## 创建服务消费方

Spring Cloud中有两种方式可以消费服务，Ribbon和Feign。

### 使用Ribbon实现消费者

Ribbon是一个基于HTTP和TCP客户端的负载均衡器，可以在通过客户端中配置的 `ribbonServerList` 服务端列表去轮询访问以达到负载均衡的作用。

当Ribbon与Eureka联合使用时，`ribbonServerList` 会被 `DiscoverEnabledNIWSServerList` 重写，扩展成从Eureka注册中心中获取服务端列表，因此在Spring Cloud中，我们只需简单配置即可使用Ribbon消费服务。

创建一个基础的Spring Boot工程，并在 `pom.xml` 文件中添加以下依赖：

```

1. <parent>
2.   <groupId>org.springframework.boot</groupId>
3.   <artifactId>spring-boot-starter-parent</artifactId>
4.   <version>1.3.5.RELEASE</version>
5.   <relativePath/> <!-- lookup parent from repository -->
6. </parent>
7. <dependencies>
8.   <dependency>
9.     <groupId>org.springframework.cloud</groupId>
10.    <artifactId>spring-cloud-starter-ribbon</artifactId>
11.  </dependency>
12.  <dependency>
13.    <groupId>org.springframework.cloud</groupId>
14.    <artifactId>spring-cloud-starter-eureka</artifactId>
15.  </dependency>
16.  <dependency>
17.    <groupId>org.springframework.boot</groupId>
18.    <artifactId>spring-boot-starter-web</artifactId>
19.  </dependency>
20.  <dependency>
21.    <groupId>org.springframework.boot</groupId>
22.    <artifactId>spring-boot-starter-test</artifactId>
23.    <scope>test</scope>
24.  </dependency>
25. </dependencies>
26. <dependencyManagement>
27.   <dependencies>
28.     <dependency>
29.       <groupId>org.springframework.cloud</groupId>
30.       <artifactId>spring-cloud-dependencies</artifactId>
31.       <version>Brixton.RELEASE</version>
32.       <type>pom</type>
33.       <scope>import</scope>
34.     </dependency>
35.   </dependencies>
36. </dependencyManagement>

```

在启动类中，通过 `@EnableDiscoveryClient` 注解来添加发现服务的能力。创建 `RestTemplate` 实例，并通过 `@LoadBalanced` 注解来开启负载均衡。

```

1. @SpringBootApplication
2. @EnableDiscoveryClient
3. public class RibbonApplication {
4.   @Bean
5.   @LoadBalanced
6.   RestTemplate restTemplate() {
7.     return new RestTemplate();
8.   }
9.   public static void main(String[] args) {
10.    SpringApplication.run(RibbonApplication.class, args);
11.  }
12. }

```

创建 `ConsumerController` 来消费 `COMPUTE-SERVICE` 的add服务。通过注入 `RestTemplate` 来调用服务：

```
1. @RestController
2. public class ConsumerController {
3.     @Autowired
4.     RestTemplate restTemplate;
5.     @RequestMapping(value = "/add", method = RequestMethod.GET)
6.     public String add() {
7.         return restTemplate.getForEntity("http://COMPUTE-SERVICE/add?a=10&b=2", String.class).getBody();
8.     }
9. }
```

最后，在 `application.properties` 中配置Eureka注册中心：

```
1. spring.application.name=ribbon-consumer
2. server.port=3333
3. eureka.client.serviceUrl.defaultZone=http://localhost:1111/eureka/
```

访问消费端的add接口，即可看到返回结果。

## 使用Feign实现消费者

Feign是一个声明式的Web Service客户端，它使得编写Web Service客户端变得更加简单。我们只需要使用Feign来创建一个接口并用注解来配置它既可完成。它具备可插拔的注解支持，包括Feign注解和JAX-RS注解。

Feign也支持可插拔的编码器和解码器。Spring Cloud为Feign增加了对Spring MVC注解的支持，还整合了Ribbon和Eureka来提供均衡负载的HTTP客户端实现。

创建一个基础的Spring Boot工程，并在 `pom.xml` 文件中添加以下依赖：

```

1. <parent>
2.   <groupId>org.springframework.boot</groupId>
3.   <artifactId>spring-boot-starter-parent</artifactId>
4.   <version>1.3.5.RELEASE</version>
5.   <relativePath/> <!-- lookup parent from repository -->
6. </parent>
7. <dependencies>
8.   <dependency>
9.     <groupId>org.springframework.cloud</groupId>
10.    <artifactId>spring-cloud-starter-feign</artifactId>
11.  </dependency>
12.  <dependency>
13.    <groupId>org.springframework.cloud</groupId>
14.    <artifactId>spring-cloud-starter-eureka</artifactId>
15.  </dependency>
16.  <dependency>
17.    <groupId>org.springframework.boot</groupId>
18.    <artifactId>spring-boot-starter-web</artifactId>
19.  </dependency>
20.  <dependency>
21.    <groupId>org.springframework.boot</groupId>
22.    <artifactId>spring-boot-starter-test</artifactId>
23.    <scope>test</scope>
24.  </dependency>
25. </dependencies>
26. <dependencyManagement>
27.   <dependencies>
28.     <dependency>
29.       <groupId>org.springframework.cloud</groupId>
30.       <artifactId>spring-cloud-dependencies</artifactId>
31.       <version>Brixton.RELEASE</version>
32.       <type>pom</type>
33.       <scope>import</scope>
34.     </dependency>
35.   </dependencies>
36. </dependencyManagement>

```

在启动类中通过 `@EnableFeignClients` 注解来开启Feign功能：

```

1. @SpringBootApplication
2. @EnableDiscoveryClient
3. @EnableFeignClients
4. public class FeignApplication {
5.     public static void main(String[] args) {
6.         SpringApplication.run(FeignApplication.class, args);
7.     }
8. }

```

定义 `compute-service` 服务提供的接口：



```
1. @FeignClient("compute-service")
2. public interface ComputeClient {
3.     @RequestMapping(method = RequestMethod.GET, value = "/add")
4.     Integer add(@RequestParam(value = "a") Integer a, @RequestParam(value = '
5. }
```

- 使用 `@FeignClient("compute-service")` 注解来绑定该接口对应的服务；
- 通过Spring MVC的注解来配置服务的具体实现；  
在Controller中注入 `ComputeClient` ，即可消费服务：

```
1. @RestController
2. public class ConsumerController {
3.     @Autowired
4.     ComputeClient computeClient;
5.     @RequestMapping(value = "/add", method = RequestMethod.GET)
6.     public Integer add() {
7.         return computeClient.add(10, 20);
8.     }
9. }
```

`application.properties` 的配置与使用Ribbon时相同，指定Eureka注册中心即可。