

由于Spring Cloud为服务治理做了一层抽象接口，所以在Spring Cloud应用中可以支持多种不同的服务治理框架，比如：Netflix Eureka、Consul、Zookeeper。在Spring Cloud服务治理抽象层的作用下，我们可以无缝地切换服务治理实现，并且不影响任何其他的服务注册、服务发现、服务调用等逻辑。

所以，下面我们通过介绍两种服务治理的实现来体会Spring Cloud这一层抽象所带来的好处。

Spring Cloud Eureka

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。通过一些简单的注解，开发者就可以快速的在应用中配置一下常用模块并构建庞大的分布式系统。它主要提供的模块包括：服务发现（Eureka），断路器（Hystrix），智能路由（Zuul），客户端负载均衡（Ribbon）等。

下面，就来具体看看如何使用Spring Cloud Eureka实现服务治理。

创建“服务注册中心”

在主工程下面创建maven module，命名为eureka-server，并在pom.xml中引入需要的依赖内容：

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

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。这一步非常的简单，只需要在一个普通的Spring Boot应用中添加这个注解就能开启此功能，比如下面的例子：

```
package lizzy.springcloud;

import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;


@EnableEurekaServer
@SpringBootApplication
public class EurekaServerApplication {
    public static void main(String[] args) {
        new
        SpringApplicationBuilder(EurekaServerApplication.class).web(true).run(args);
    }
}
```

在默认设置下，该服务注册中心也会将自己作为客户端来尝试注册它自己，所以我们需要禁用它的客户端注册行为，只需要在`application.properties`配置文件中增加如下信息：

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

```
eureka.instance.hostname=localhost
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

为了与后续要进行注册的服务区分，这里将服务注册中心的端口通过`server.port`属性设置为1001。启动工程后，访问：<http://localhost:1001/>，可以看到下面的页面，其中还没有发现任何服务。

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2019-04-15T15:30:51 +0800
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

Name	Value
total-avail-memory	517mb
environment	test
num-of-cpus	8
current-memory-usage	78mb (15%)
server-uptime	00:00
registered-replicas	
unavailable-replicas	

创建“服务提供方”

下面我们创建提供服务的客户端，并向服务注册中心注册自己。本文我们主要介绍服务的注册与发现，所以我们不妨在服务提供方中尝试着提供一个接口来获取当前所有的服务信息。首先，创建maven module，命名为`eureka-client`，在`pom.xml`中，加入如下配置：

```
<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>
```

其次，实现`/dc`请求处理接口，通过`DiscoveryClient`对象，在日志中打印出服务实例的相关内容。

```
package lizzy.springcloud.controller;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class DcController {
    @Autowired
    DiscoveryClient discoveryClient;

    @Value("${server.port}")
    String port;

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

```

最后在主类中通过加上@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。

```
package lizzy.springcloud;
```

```

import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@EnableDiscoveryClient
@SpringBootApplication
public class EurekaClientApplication {
    public static void main(String[] args) {
        new
        SpringApplicationBuilder(EurekaClientApplication.class).web(true).run(args);
    }
}

```

我们在完成了服务内容的实现之后，再继续对application.properties做一些配置工作，具体如下：

```

spring.application.name=eureka-client
server.port=2001

```

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。为了在本机上测试区分服务提供方和服务注册中心，使用server.port属性设置不同的端口。

启动该工程后，再次访问：http://localhost:1001/。可以如下图内容，我们定义的服务被成功注册了。

DS Replicas			
Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
EUREKA-CLIENT	n/a (1)	(1)	UP (1) - DESKTOP-FL1GF59:eureka-client:2001

当然，我们也可以通过直接访问eureka-client服务提供的/dc接口来获取当前的服务清单，只需要访问：http://localhost:2001/dc，我们可以得到如下输出返回：

Services: [eureka-client]

具体工程说明如下：

- eureka的服务注册中心：eureka-server
- eureka的服务提供方：eureka-client