**[Spring Cloud 入门教程(一): 服务注册](https://www.cnblogs.com/chry/p/7248947.html)**

**1.  什么是Spring Cloud？**

*Spring提供了一系列工具，可以帮助开发人员迅速搭建分布式系统中的公共组件（比如：配置管理，服务发现，断路器，智能路由，微代理，控制总线，一次性令牌，全局锁，主节点选举， 分布式session, 集群状态）。协调分布式环境中各个系统，为各类服务提供模板性配置。使用Spring Cloud, 开发人员可以搭建实现了这些样板的应用，并且在任何分布式环境下都能工作得非常好，小到笔记本电脑， 大到数据中心和云平台。*

Spring Cloud官网的定义比较抽象，我们可以从简单的东西开始。Spring Cloud是基于Spring Boot的， 最适合用于管理Spring Boot创建的各个微服务应用。要管理分布式环境下的各个Spring Boot微服务，必然存在服务的注册问题。所以我们先从服务的注册谈起。既然是注册，必然有个管理注册中心的服务器，各个在Spring Cloud管理下的Spring Boot应用就是需要注册的client

Spring Cloud使用erureka server,  然后所有需要访问配置文件的应用都作为一个erureka client注册上去。eureka是一个高可用的组件，它没有后端缓存，每一个实例注册之后需要向注册中心发送心跳，在默认情况下erureka server也是一个eureka client ,必须要指定一个 server。

**2.  创建Eureka Server**

1）.创建一个Maven工程helloworld.eureka.server， pom.xml内容如下：

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

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

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

4 <modelVersion>4.0.0</modelVersion>

5 <groupId>com.chry</groupId>

6 <artifactId>springcloud.helloworld.eureka.server</artifactId>

7 <version>0.0.1-SNAPSHOT</version>

8 <packaging>jar</packaging>

9 <name>springcloud.helloworld.Eureka.server</name>

10 <description>Demo Spring Eureka Server</description>

11

12 <parent>

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

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

15 <version>1.5.3.RELEASE</version>

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

17 </parent>

18

19 <properties>

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

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

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

23 </properties>

24

25 <dependencies>

26 <!--eureka server -->

27 <dependency>

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

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

30 </dependency>

31 <dependency>

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

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

34 </dependency>

35 <dependency>

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

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

38 </dependency>

39 <!-- spring boot test-->

40 <dependency>

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

42 <artifactId>spring-boot-starter-test</artifactId>

43 <scope>test</scope>

44 </dependency>

45 </dependencies>

46

47 <dependencyManagement>

48 <dependencies>

49 <dependency>

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

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

52 <version>Dalston.RC1</version>

53 <type>pom</type>

54 <scope>import</scope>

55 </dependency>

56 </dependencies>

57 </dependencyManagement>

58

59 <build>

60 <plugins>

61 <plugin>

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

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

64 </plugin>

65 </plugins>

66 </build>

67

68 <repositories>

69 <repository>

70 <id>spring-milestones</id>

71 <name>Spring Milestones</name>

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

73 <snapshots>

74 <enabled>false</enabled>

75 </snapshots>

76 </repository>

77 </repositories>

78

79 </project>

 2）. 用Spring Boot创建一个服务类EurekaServerApplication，需要一个注解@EnableEurekaServer加在springboot工程的启动类上

1 package springcloud.helloworld.eureka.server;

2

3 import org.springframework.boot.SpringApplication;

4 import org.springframework.boot.autoconfigure.SpringBootApplication;

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

6

7 @EnableEurekaServer

8 @SpringBootApplication

9 public class EurekaServerApplication {

10

11 public static void main(String[] args) {

12 SpringApplication.run(EurekaServerApplication.class, args);

13 }

14 }

3).eureka server的配置文件appication.yml，其中registerWithEureka：false和fetchRegistry：false表明自己是一个eureka server

1 server:

2 port: 8761

3

4 eureka:

5 instance:

6 hostname: localhost

7 client:

8 registerWithEureka: false

9 fetchRegistry: false

10 serviceUrl:

11 defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

4) eureka server的工程结构如下

5）启动eureka server，然后访问http://localhost:8761, 界面如下， "No instances available" 表示无client注册

**3.  创建Eureka Client**

1). 创建一个Maven工程helloworld.eureka.client， pom.xml内容如下：

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

2 <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">

3 <modelVersion>4.0.0</modelVersion>

4 <groupId>com.chry</groupId>

5 <artifactId>springcloud.helloworld.eureka.client</artifactId>

6 <version>0.0.1-SNAPSHOT</version>

7 <name>springcloud.helloworld.eureka.client</name>

8 <packaging>jar</packaging>

9 <description>Demo Spring Boot Client</description>

10

11 <parent>

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

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

14 <version>1.5.3.RELEASE</version>

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

16 </parent>

17

18 <properties>

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

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

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

22 </properties>

23

24 <dependencies>

25 <dependency>

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

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

28 </dependency>

29 <dependency>

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

31 <artifactId>spring-boot-starter-web</artifactId>

32 </dependency>

33

34 <dependency>

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

36 <artifactId>spring-boot-starter-test</artifactId>

37 <scope>test</scope>

38 </dependency>

39 </dependencies>

40

41 <dependencyManagement>

42 <dependencies>

43 <dependency>

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

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

46 <version>Dalston.RC1</version>

47 <type>pom</type>

48 <scope>import</scope>

49 </dependency>

50 </dependencies>

51 </dependencyManagement>

52

53 <build>

54 <plugins>

55 <plugin>

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

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

58 </plugin>

59 </plugins>

60 </build>

61

62 <repositories>

63 <repository>

64 <id>spring-milestones</id>

65 <name>Spring Milestones</name>

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

67 <snapshots>

68 <enabled>false</enabled>

69 </snapshots>

70 </repository>

71 </repositories>

72

73

74 </project>

[复制代码](javascript:void(0);)

2).  创建主类EurekaClientApplication

使用@EnableEurekaClient注解表明是client

[复制代码](javascript:void(0);)

1 package springcloud.helloworld.eureka.client;

2

3 import org.springframework.beans.factory.annotation.Value;

4 import org.springframework.boot.SpringApplication;

5 import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

8 import org.springframework.web.bind.annotation.RequestParam;

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

10

11 @SpringBootApplication

12 @EnableEurekaClient

13 @RestController

14

15 public class EurekaClientApplication {

16

17 public static void main(String[] args) {

18 SpringApplication.run(EurekaClientApplication.class, args);

19 }

20

21 @Value("${server.port}")

22 String port;

23 @RequestMapping("/")

24 public String home() {

25 return "hello world from port " + port;26 }

27

28 }

[复制代码](javascript:void(0);)

3) eureka client的配置文件appication.yml

[复制代码](javascript:void(0);)

1 eureka:

2 client:

3 serviceUrl:

4 defaultZone: http://localhost:8761/eureka/

5 server:

6 port: 8762

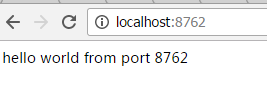
7 spring:

8 application:

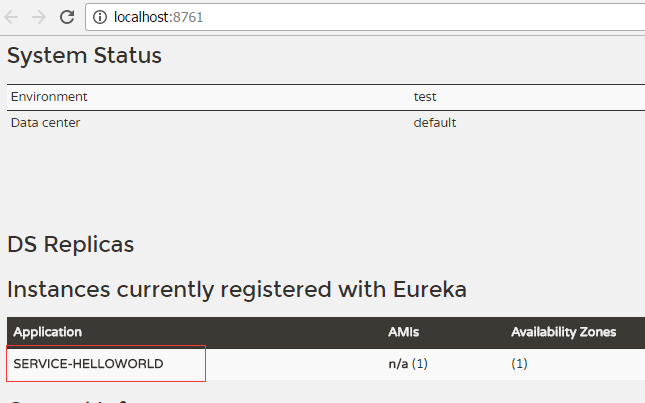
9 name: service-helloworld

[复制代码](javascript:void(0);)

4). Client启动后， 可以访问http://localhost:8762



5). 再次访问服务器端口, 可以看到Service Helloworld已经自动注册到之前的server中

****