# 标题一

## 标题二

正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文

### 标题三

正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文

#### 标题四

正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文正文

表格

|  |  |  |
| --- | --- | --- |
| 分类 | 具体依赖 | 作用 |
|  | DevTools | Springboot开发工具包，热部署需要使用 |
|  | Security | Spring Security，spring提供的安全服务 |
|  | Lomok | Java Annotation，annotation注解支持 |

代码块

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <optional>true</optional>  
</dependency>

# Springcloud实战篇

Springcloud分布式框架是基于Springboot框架的，在以下的每一个服务中都要在Springboot工程的pom.xml中引入Springcloud的依赖。

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

## 服务注册与发现

Springcloud中服务注册与发现功能一般使用Eureka作为服务注册与发现组件，该组件主要分为服务端和客户端。服务端充当注册中心，统一管理分布式系统中的各个服务；而客户端则是作为微服务端，要把自身注册到服务端，通过服务端去访问分布式系统中其他服务。

### EurekaServer

1、引入Eureka服务端依赖：在pom.xml文件中引入eureka-server依赖。

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

2、配置EurekaServer：在application.yml配置文件中对EurekaServer进行配置。

**服务名：**是该服务在分布式系统中的别名（非常重要），分布式系统中都是使用服务名进行服务请求的，但对于EurekaServer而言，若不对EurekaServer进行集群的话，该服务名可以随便取（但建议养成良好习惯，不乱取名）。

**服务开启端口：**该服务在主机上开启的端口。

**服务主机名：**服务所在主机的ip，若本地访问可以设置localhsot

**EurekaServer特殊设置：**register-with-eureka（是否注册自身到注册中心）、fetch-registry（是否从注册中心获取注册信息）。EurekaServer其实本质也是客户端，这两个属性就决定了该服务的服务端性质。

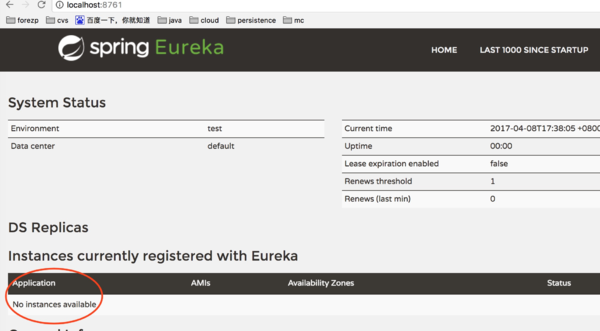
**服务路径：**在EurekaServer没有集群的情况下，EurekaClient都通过该路径访问EurekaServer服务，进行服务注册等操作。

**spring:  
 application:  
 name:** eureka-server *#服务名***server:  
 port:** 9001 *#服务开启端口***eureka:  
 instance:  
 hostname:** localhost *#服务主机名* **client:  
 register-with-eureka: false** *#是否注册自身到注册中心* **fetch-registry: false** *#是否从注册中心获取注册信息* **service-url:  
 defaultZone:** http://${**eureka.instance.hostname**}:${**server.port**}/eureka/ *#服务路径*

3、开启服务：将@EnableEurekaServer注解标注在Springboot启动类上，就可开启EurekaServer服务。

@SpringBootApplication  
@EnableEurekaServer //开启EurekaServer服务  
public class EurekaServerApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaServerApplication.class, args);  
 }  
}

最后通过访问http://localhost:9001进入EurekaServer的控制台，其中DS Replicas处可以查看服务注册中心中已经注册的服务。并且可以看到服务的简略信息（端口，服务进程数等）。



### EurekaClient

1、引入Eureka客户端依赖：在pom.xml文件中引入，eureka-client依赖。

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>

2、配置EurekaClient：在application.yml配置文件中配置下相关信息。其中defaultZone配置注册中心的服务路径，而不是自身服务路径（客户端服务不需要通过服务路径访问，只需要通过服务名访问服务）。

**spring:  
 application:  
 name:** service-hi *#服务名***server:  
 port:** 9102 *#服务开启端口***eureka:  
 client:  
 serviceUrl:  
 defaultZone:** http://localhost:9001/eureka/ *#注册中心服务路径*

3、启动客户端服务：将@EnableEurekaClient注解标注在Springboot启动类上。其中@EnableEurekaClient注解可以用@EnableDiscoveryClient注解代替，但是两者有区别，详见原理篇。

@EnableEurekaClient //表示该项目为eureka注册中心客户端  
@SpringBootApplication  
public class EurekaClient01Application {  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaClient01Application.class, args);  
 }  
}

## 客户端负载均衡

Springcloud中一般使用ribbon组件实现客户端负载均衡的功能，也可以用Feign组件（该组件集成了ribbon组件）。

### Ribbon+restTemplate

1、引入ribbon依赖：负载均衡要配合分布式系统使用，也就是该服务要注册到分布式系统中，因此不仅要引入ribbon依赖，还得引入EurekaClient依赖。由于EurekaClient依赖中已经集成了ribbon组件，因此不需要另外引入ribbon，只需要引入EurekaClient依赖即可。

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>

2、配置EurekaClient：步骤和服务注册与发现中一致。

3、配置Bean—>RestTemplate：配置一个RestTemplate Bean到ioc容器。

4、开启负载均衡：用@LoadBalance注解标注RestTemplate Bean以开启负载均衡。

@Bean  
@LoadBalanced //开启负载均衡  
RestTemplate restTemplate(){  
 return new RestTemplate();  
}

5、请求服务：利用RestTemplate的方法getForObject方法获取http请求的body，还可用getForEntity获取整个http请求的信息，还有post、put、delete等请求方式的方法。

public String hiService(String name) {  
 System.*out*.println("进入HtlloService.hiService()!");  
 return restTemplate.getForObject(

"http://SERVICE-HI/hi?name="+name,//请求路径，把主机名改为服务名，否则不可负载均衡

String.class //想要获取的返回值类型

);  
}

### Feign

Feign组件集成了ribbon组件，但Feign简化了我们的工作，只需要编写接口就可以实现负载据均衡的功能。

1、引入Feign依赖：在pom.xml文件中引入openfeign依赖。除此之外还需要引入EurekaClient

依赖，让该服务注册到分布式系统中。

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
</dependency>

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>

2、

## 分布式配置

## 断路器

## 服务网关

# Springcloud原理篇

## 服务注册与发现

### Eureka

## 客户端负载均衡

## 分布式配置

## 断路器

## 服务网关