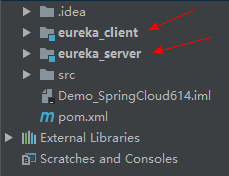
SpringCloud学习体会

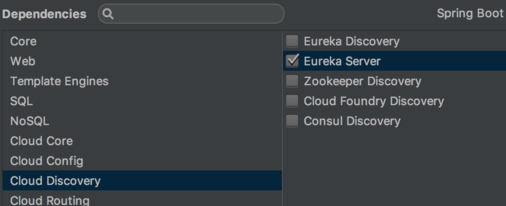
1. SpringCloud是什么

SpringCloud是 构建分布式系统的一套工具，其运行环境简单，且基于SpringBoot构建。

1. 创建SpringCloud的服务注册中心（Eureka）
2. 在一个主Maven中创建两个Module，分别做服务的服务端和客户端



2、创建时都选中如下所示。创建完两个分Module会自动下载依赖，但一直下载不下来，最后利用翻墙软件才顺利获得所有依赖库



* 1. **启动一个服务注册中心**，只需要一个注解@EnableEurekaServer，写在配置类的类上面。在默认情况下erureka\_server也是一个client ,

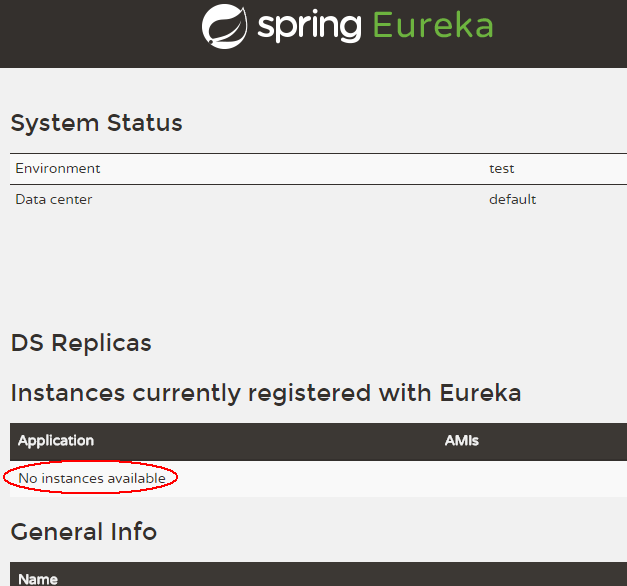
@EnableEurekaServer  
@SpringBootApplication  
public class EurekaServerApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaServerApplication.class, args);

4、还必须要修改其配置文件。eureka server的配置文件application.properties如下：

server.port= 8761  
eureka.instance.hostname=localhost  
eureka.client.registerWithEureka=**false**eureka.client.fetchRegistry=**false**eureka.client.serviceUrl.defaultZone=http://${eureka.instance.hostname}:${server.port}/eureka/

其中eureka.client.registerWithEureka=false和eureka.client.fetchRegistry=false来表明自己是一个server.

5、eureka\_server启动后,浏览器访问：[http://localhost:8761](http://localhost:8761/) ,界面如下：



此时显示没有服务被发现

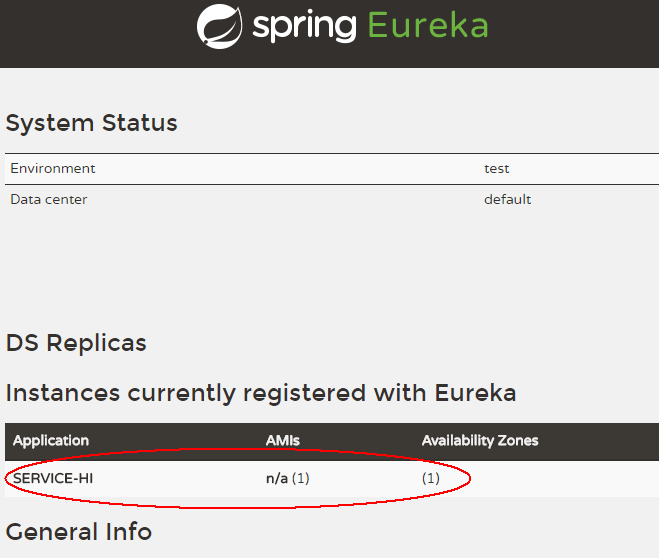
6、通过注解@EnableEurekaClient 表明自己是一个client,同样需要写在eureka\_client的配置类的类上面

@EnableEurekaClient  
@SpringBootApplication  
@RestController  
public class EurekaClientApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaClientApplication.class, args);  
 }  
 @Value("${server.port}")  
 String port;  
 @RequestMapping("/hi")  
 public String home(@RequestParam String name) {  
 return "hi "+name+",i am from port:" +port;  
 }  
}

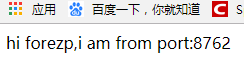
7、同时需要修改其配置文件application.properties

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/  
server.port=8762  
spring.application.name=service-hi

8、启动eureka\_client，并刷新eureka\_server的页面，端口为7862的服务已经注册在服务中了，服务名为SERVICE-HI ,



9、访问 <http://localhost:8762/hi?name=forezp> ，就会看到



10、总结：

使用Eureka是因为在一个完整的系统架构中，任何单点的服务都不能保证不会中断，因此我们需要服务发现机制，在某个节点中断后，其它的节点能够继续提供服务，从而保证整个系统是高可用的。

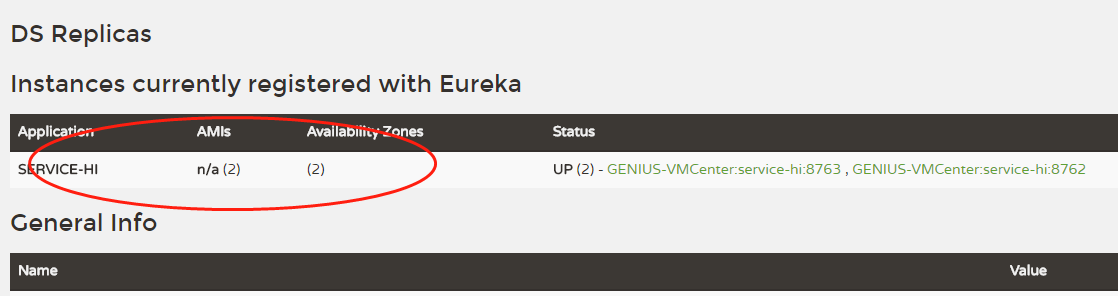
服务发现有两种模式：一种是客户端发现模式，一种是服务端发现模式。Erueka采用的是客户端发现模式。

Eureka通过心跳检测、健康检查、客户端缓存等机制，保证了系统具有高可用和灵活性。

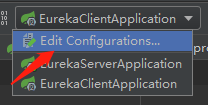
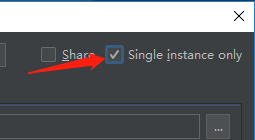
1. 服务消费者（rest+ribbon）

1、在微服务架构中，业务都会被拆分成一个独立的服务，服务与服务的通讯是基于http restful的。ribbon是一个负载均衡客户端，可以很好的控制htt和tcp的一些行为。Feign默认集成了ribbon。

2、启动eureka\_client工程，它的端口为8762；将eureka\_client的配置文件的端口改为8763并启动，这时你会发现：eureka\_client在eureka-server注册了2个实例，这就相当于一个小的集群。



Idea里一个工程要启动多个实例是指如下，需要把勾去掉

1. 重新新建一个spring-boot工程，取名为：service-ribbon；在它的pom.xml文件分别添加如下起步依赖：

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

<version>1.4.4.RELEASE</version>  
</dependency>

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>

1. 在工程的配置文件指定服务的注册中心地址为http://localhost:8761/eureka/，程序名称为 service-ribbon，程序端口为8764。配置文件application.xml如下：

eureka.client.serviceUrl.defaultZone= http://localhost:8761/eureka/  
server.port=8764  
spring.application.name= service-ribbon

1. 在工程的启动类中,通过@EnableDiscoveryClient向服务中心注册；并且向程序的ioc注入一个bean: restTemplate;并通过@LoadBalanced注解表明这个restRemplate开启负载均衡的功能。

@SpringBootApplication  
@EnableDiscoveryClient  
public class ServiceRibbonApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ServiceRibbonApplication.class, args);  
 }  
   
 @Bean  
 @LoadBalanced  
 public RestTemplate restTemplate(){  
 return new RestTemplate();  
 }  
}

1. 写一个测试类HelloService，通过之前注入ioc容器的restTemplate来消费service-hi服务的“/hi”接口，在这里直接用的程序名替代了具体的url地址，在ribbon中它会根据服务名来选择具体的服务实例，根据服务实例在请求的时候会用具体的url替换掉服务名，代码如下：

@Service  
public class HelloService {  
 @Autowired  
 RestTemplate restTemplate;  
  
 public String hiService(String name){  
 return restTemplate.getForObject("http://service-hi/hi?name="+name,String.class);   
 }  
}

1. 写一个controller，在controller中用调用HelloService 的方法，代码如下：

@RestController  
public class HelloController {  
 @Autowired  
 HelloService helloService;  
 @RequestMapping(value = "/hi")  
 public String hi(@RequestParam String name){  
 return helloService.hiService(name);  
 }  
}

1. 在浏览器上多次访问http://localhost:8764/hi?name=forezp，浏览器交替显示：

8、这说明当通过调用restTemplate.getForObject(“ [service-hi /hi?name=](http://service-hi/hi?name=)“

+name,String.class)方法时，已经做了负载均衡，访问了不同的端口的服务实例。此时工程架构为：

<1>一个服务注册中心，eureka\_server,端口为8761

<2>service-hi工程跑了两个实例，端口分别为8762,8763，分别向服务注册中心注册

<3>sercvice-ribbon端口为8764,向服务注册中心注册

<4>当sercvice-ribbon通过restTemplate调用service-hi的hi接口时，因为用ribbon进行了负载均衡，会轮流的调用service-hi：8762和8763 两个端口的hi接口；

1. 服务消费者（Feign）

1、Feign 采用的是基于接口的注解，并且其也整合了ribbon

2、启动eureka-server，端口为8761; 启动service-hi 两次，端口分别为8762 、8773.

3、新建一个spring-boot工程，取名为serice-feign，在它的pom文件引入Feign的起步依赖spring-cloud-starter-feign、Eureka的起步依赖spring-cloud-

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

<version>1.4.4.RELEASE</version>  
</dependency>

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>

4、引入这些依赖之后，在工程的配置文件 application.xml做相关的配置，包括配置程序名为 eureka-feign-client，端口号为 8765，服务注册地址为 http://localhost:876l/eureka/，代码如下：

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/  
server.port=8765  
spring.application.name= service-feign

5、在程序的启动类ServiceFeignApplication ，加上@EnableFeignClients注解开启Feign的功能：

@SpringBootApplication  
@EnableDiscoveryClient  
@EnableFeignClients  
public class SericeFeignApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(SericeFeignApplication.class, args);  
 }  
}

此学习体会是根据网上一篇帖子学习的，因为此贴存在很多问题，需要占用太多学习时间寻找解决方案，所以放弃根据此贴的学习。另外已经找到此贴作者的书籍，将根据书籍进一步学习