Spring Cloud 系列——Eureka

作者：figo2young
原文链接：https://ld246.com/article/1567564766812
来源网站：链滴

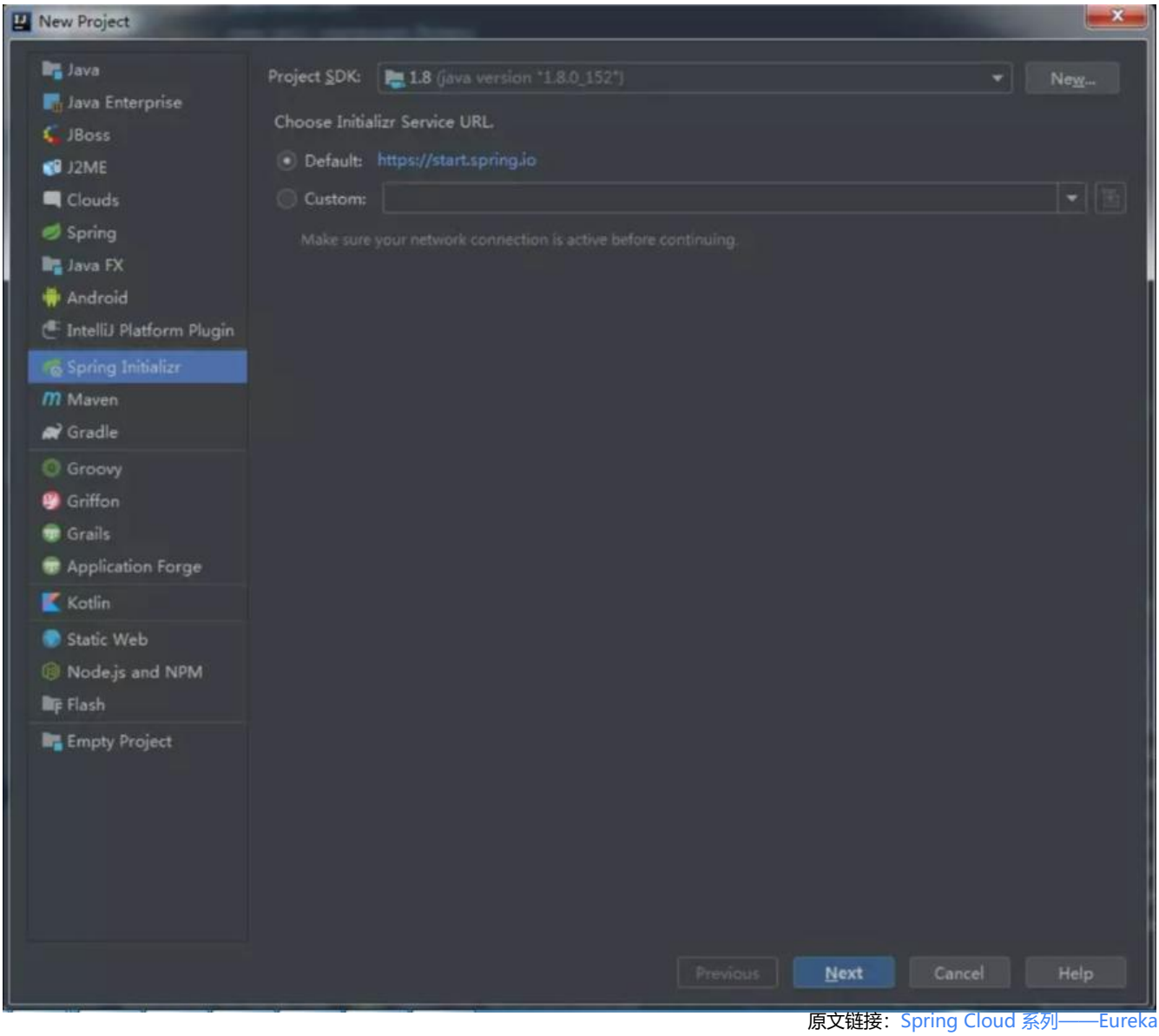
从这篇文章开始我们共同走进Spring Cloud的世界，共同了解目前互联网微服务的利器——Spring loud是如何帮助我们更好的构建微服务体系。如果您还没有接触过微服务，或者没有接触过Spring Cl ud，这个系列能帮助你对Spring Cloud有一个全面和整体的认识。

在微服务架构出现前，我们的软件系统都使用单体架构，随着互联网的发展，微服务架构成为了越越多互联网公司的首选架构体系，由于篇幅有限，本文不详述微服务与单体系统之间的异同。仅以一简单易懂的类比故事来阐述微服务与单体系统的区别：我们以一个样样精通的厨师类比为一个单体系，这个厨师需要提供完成一道菜所需要的每一项功能——选材、洗碗、洗菜、切菜、炒菜、洗锅、端等等，每次客人点菜，厨师完成了所有工作，然后给客人端上菜，这些功能就好比一个单体系统的功，样样俱全。微服务架构就好比上面的每个功能都有一个专门负责这些工作的人来做，每个人之间通特定的方式来交流。这样做的好处是每个人都各自干好自己负责的活，每个人可以独立的去进行培训升级、修改），不会对其他人造成影响。这个小故事只是大概说明了微服务和单体应用之间的区别，于初次接触微服务的同学，能有初步的认识。

从单体系统架构往微服务架构发展，中间自然会有非常多的问题，Spring Cloud会为我们解决微服架构中产生的各种问题，下面进入我们的Spring Cloud全家桶时间——Eureka。

Eureka的主要功能：搭建服务的注册中心，提供服务的注册和服务的发现。下面介绍如何从0搭建个Eureka Server（后面的搭建都是基于Spring Boot 2.1.0和Spring Cloud 2.0.0——Finchley.RELEAE版，并使用intellij idea）。

1.创建一个基础的Spring Boot项目



2.引入Eureka Server包，并配置Spring Cloud的版本

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

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

<!-- dependencyManagement只管理版本，不会实际下载jar包 -->
<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>
```

3.在resources目录下添加配置文件，并如下图添加配置（本文使用yml文件）

```
server:
  port: 2222
eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
  instance:
    hostname: localhost
```

1.server.port:Erueka Server的端口

2.eureka.client.register-with-eureka:表示是否向注册中心注册自己（注册中心不需要注册自己）

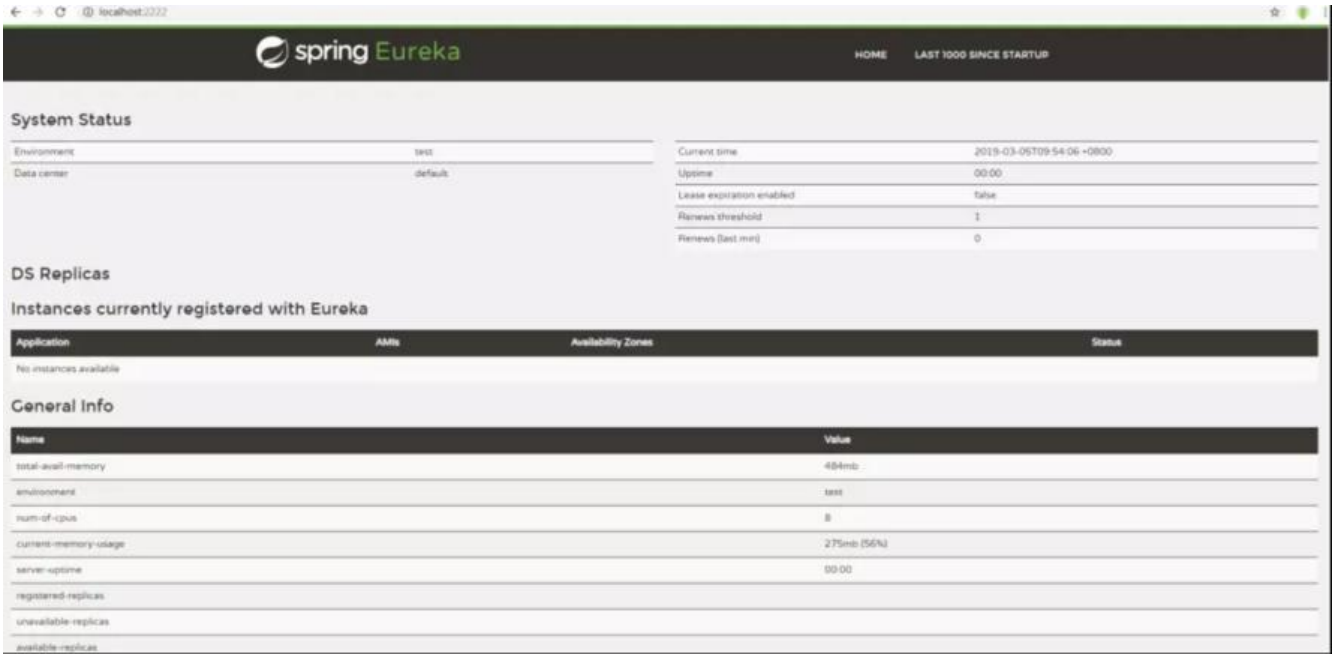
3.eureka.client.fetch-registry:表示是否检索服务（注册中心不需要检索服务，所以设为false）

4.eureka.client.service-url.defaultZone:服务中心的默认空间（服务注册的时候会用到）

4.启动类中添加注解@EnableEurekaServer

```java
/**
 * @author 关注微信公众号：Java学习交流社区
 * （更多技术干货、原创文章、面试资料，群聊吹水一应俱全）
 */

@EnableEurekaServer
@SpringBootApplication(exclude= {DataSourceAutoConfiguration.class})
public class EurekaServerApplication {

    public static void main(String[] args) { SpringApplication.run(EurekaServerApplication.class, args); }

}
```

启动Spring boot应用，访问http://localhost:2222，能看到以下页面即为配置成功



现在可以看到中间的 Instances currently registered with Eureka是No instances available，表示前服务中心并没有服务注册，下面我们建立一个简单的Eureka Client注册到服务中心当中

1.同样新建一个Spring Boot项目，引入Spring boot的web模块和eureka-client块

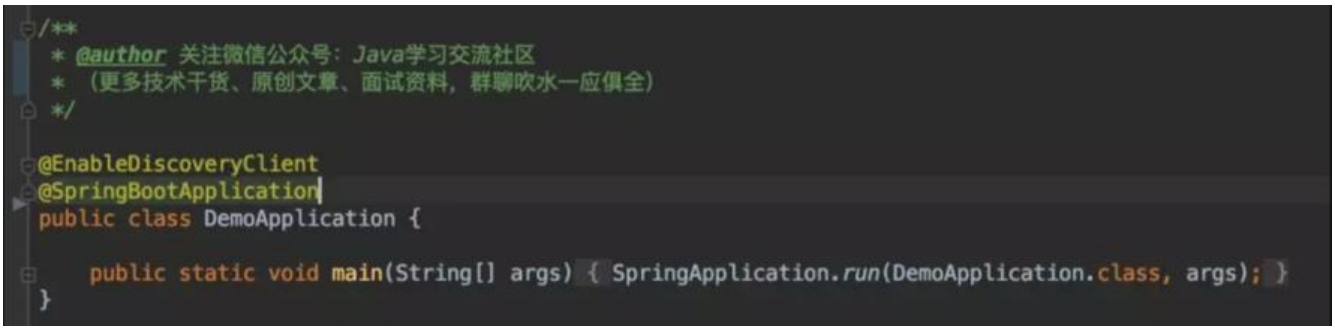
```xml
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
   </dependency>
</dependencies>
```

2.同样在resources目录下的配置文件中，新增配置

```yaml
spring:
  application:
    name: test-client-service
```

```
#eureka注册中心服务器地址
eureka:
  client:
    service-url:
      defaultZone: http://localhost:2222/eureka/
```

3.启动类中添加注解@EnableDiscoveryClient



```
/**
 * @author 关注微信公众号：Java学习交流社区
 *  （更多技术干货、原创文章、面试资料，群聊吹水一应俱全）
 */

@EnableDiscoveryClient
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) { SpringApplication.run(DemoApplication.class, args); }
}
```

4.任意添加一个RestController

```
/**
 * @author 关注微信公众号：Java学习交流社区
 *  （更多技术干货、原创文章、面试资料，群聊吹水一应俱全）
 */

@RestController
public class DemoController {

    @Autowired
    private IpConfiguration ipConfiguration;

    @RequestMapping("/")
    public String test(){
        return ipConfiguration.getPort()+"";
    }

}
```

IpConfiguration用于获取当前端口（为了测试负载均衡时获取不同端口，非必须）

```
@Component
public class IpConfiguration implements ApplicationListener<WebServerInitializedEvent> {
    private int serverPort;

    @Override
    public void onApplicationEvent(WebServerInitializedEvent event) {
        this.serverPort = event.getWebServer().getPort();
    }

    /**
     * @return 获取本微服务端口号
```

```
     */
    public int getPort() {
        return this.serverPort;
    }

}
```

配置完成，运行client，可以看到以下画面

Instances currently registered with Eureka中显示有注册的client了，状态为Up

为了验证注册到服务中心的客户端都能被访问，把client项目修改一下端口为8089，8090分别导出成jr包格式,同时运行两个端口的包。

再另外起一个Eureka Client项目，端口为9002，服务名称为ribbon-service，与上面介绍的Client配一样，不同点在启动类中添加一下代码：

```
@Bean
@LoadBalanced
RestTemplate restTemplate(){
    return new RestTemplate();
}
```

这里使用了Ribbon这个Spring Cloud的工具包，后面的文章会详细讲解Ribbon，此处可以先使用，解为提供一个http请求的服务，并负载均衡地请求一个服务的多个提供者

添加一个RestController用于此服务的访问

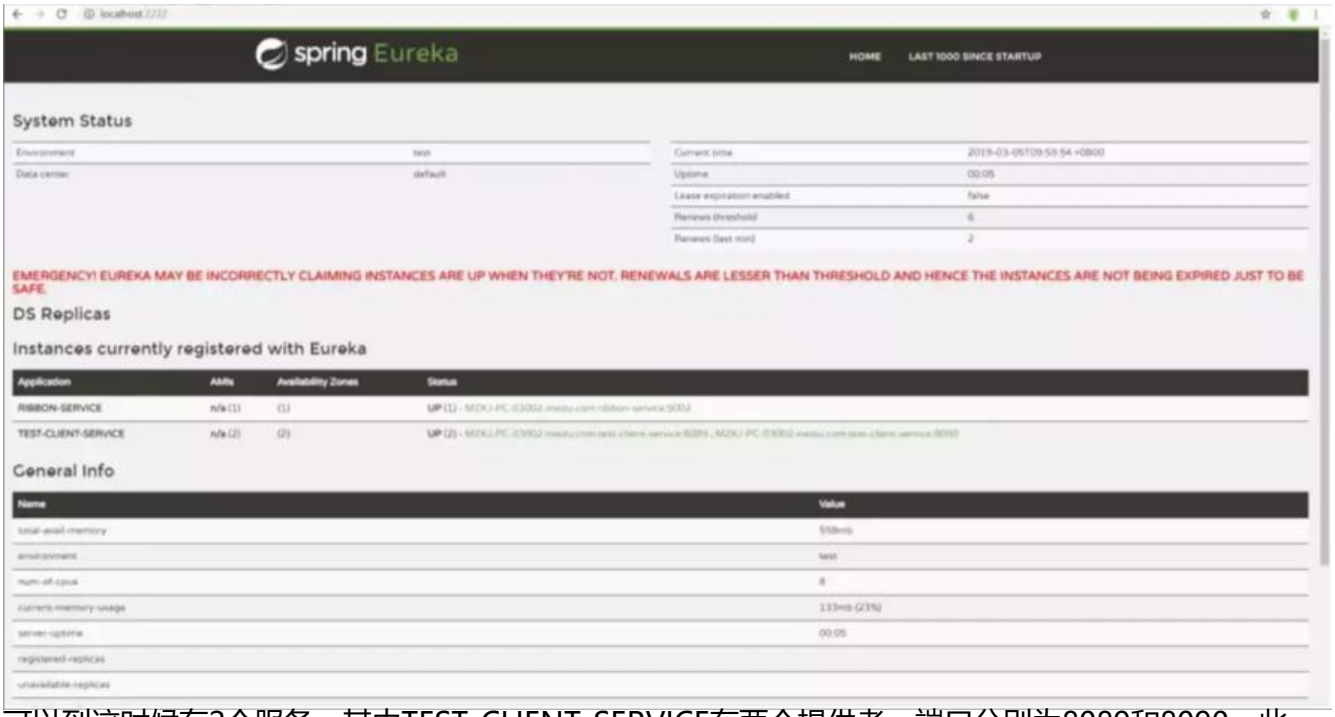
```
@RestController
@RequestMapping("/ribbon")
public class ConsumerController {

    @Autowired
    private TestClientService testClientService;

    @RequestMapping
    public String helloConsumer(){
        return testClientService.testClientService();
    }

}
```

创建Service：

```
@Service
public class TestClientService{

    @Autowired
    private RestTemplate restTemplate;

    public String testClientService(){
        String testClientService = restTemplate.getForEntity("http://TEST-CLIENT-SERVICE/",Stri
g.class).getBody();
```

```
        return testClientService;
    }

}
```

启动后看到界面如下：



可以到这时候有2个服务，其中TEST-CLIENT-SERVICE有两个提供者，端口分别为8089和8090，此我们访问http://localhost:9002/ribbon，可以看到这时候能够负载均衡地访问到2个TEST-CLIENT-SRVICE

结束语：本文讲解了Spring Cloud中Eureka的注册中心的搭建，服务的注册，以及一个简单的负载衡访问Eureka Client的实例，希望能对读者对Spring Cloud的应用能有所帮助。

下一篇将会对Spring Cloud中的Ribbon和Feign的配置及应用进行讲解，想要和我学习Spring Clou系列的同学，记得关注我的微信公众号《Java学习交流社区》

```
<hr>
<div style="text-align:center">
<img src="https://b3logfile.com/file/2019/08/Java学习交流社区推广二维码-fdaf0ead.gif" style
```

"margin-bottom:15px"></img>

扫码关注微信公众号《Java学习交流社区》，即可获取原创《Java并发编程与高并发解决方》+《Redis》思维导图

作者：Java学习交流社区
地址：https://www.jvscc.cn/articles/2019/08/22/1566441527094.html
