1. SpringCloud：分布式微服务的一站式解决方案，是多种微服务架构落地技术及集合体。
2. 声明父工程步骤：

①<packaging>pom</packaging>

②<dependencyManagement></dependencyManagement>：为了父工程进行统一依赖版本。

1. 问题：创建父工程下的子模块项目（springboot子模块）时，必须要从New Project -> Spring Initializr下创建springboot模块。

然后调整子模块的pom.xml，加入所属父工程的信息：

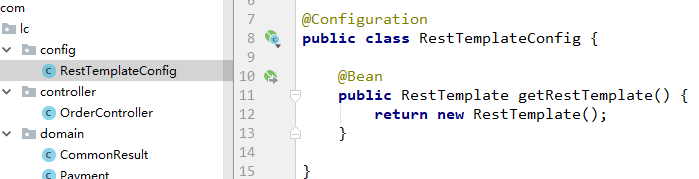
<parent>  
 <groupId>com.lc</groupId>   
 <artifactId>springcloud2021</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 </parent>

最后在父工程的pom.xml下引入子模块的信息：

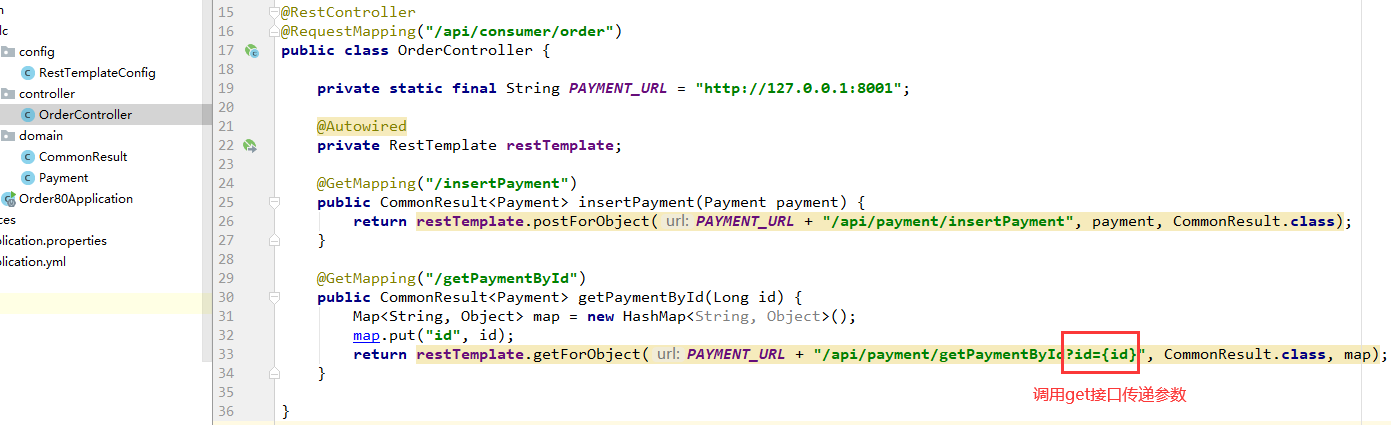
<modules>  
 <module>payment8001</module>  
 </modules>

1. domain当前构造方法调用其他构造方法用this(参数.....)即可。
2. 客户端调用服务端RestTemplate使用：当中封装了HttpClient，提供了访问远程http服务的方法，便捷方位restful服务接口，使用：

①将RestTemplate注入到spring容器中：



②使用RestTemplate调用服务端接口：



注：调用服务端post接口传递对象，参数接口必须要加@RequestBody。

调用服务端get接口传递参数，参数接口必须要加@RequestParam。

1. 扩展（待试）：多个微服务自动调用出idea的Run Dashboard

找到idea项目下 -> .idea文件夹 -> workspace.xml -> 内容在<component name="RunDashboard">下添加内容 -> 关闭idea重启

内容：<option name="configurationTypes">

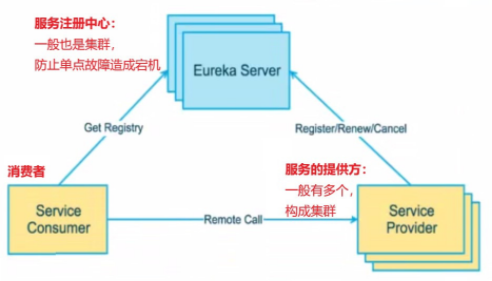
<set>

<option value="SpringBootApplicationConfigurationType" />

</set>

</option>

1. 抽离公共部分代码生成项目 -> 编译打包 -> 其他项目中删除公共代码并引入依赖即可。
2. Eureka：服务注册中心，集群模式解决单点故障。

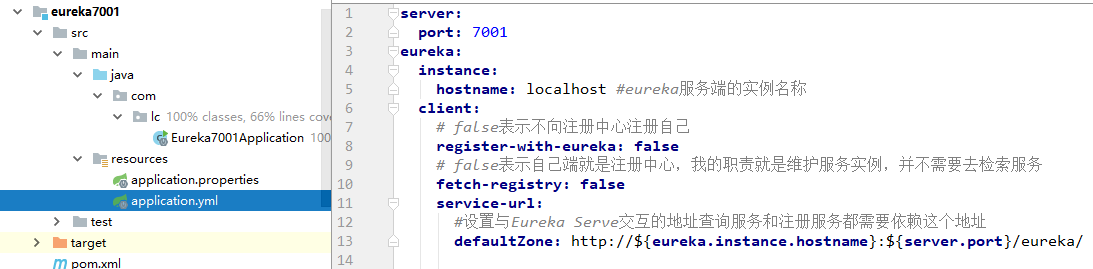


1. Eureka：开启新模块引入注册中心：

① 导入注册端jar包

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

② 设置单台Eureka的application.yml:



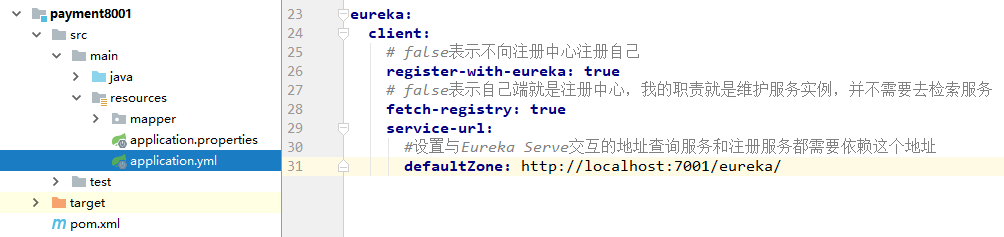
③ 主启动类上添加注解：@EnableEurekaServer

1. Eureka：提供端需注册服务配置：

① 导入提供端jar包：

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

② 设置单台提供者的application.yml:



③ 主启动类上添加注解：@EnableEurekaClient

④ 访问地址：<http://localhost:7001> 不带/eureka，查看服务注册内容

1. Eureka：配置注册中心集群，相互注册，相互守望

① 因多个注册中心配置的hostname都是都是localhost，所以需要在hosts文件中配置：

127.0.0.1 eureka7001.com

127.0.0.1 eureka7002.com

② 配置注册中心application.yml文件



③ 配置提供端application.yml集群模式



12. Eureka：配置服务端集群模式，除端口号不同外，都一致，客户端访问时，需要将地址路径改为服务端的服务名称，配置RestTemplate时需要加负载均衡注解，获取指定服务器提供服务。



13. Eureka：修改提供端在注册中心的名称及点击显示ip信息；

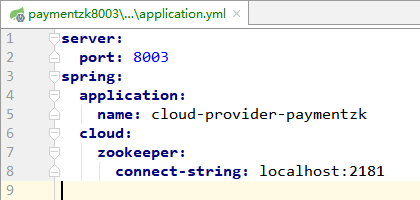
**eureka:  
 instance:  
 instance-id:** payment8001  
 **prefer-ip-address: true**

1. Zookeeper：使用zk替代Eureka作为注册中心

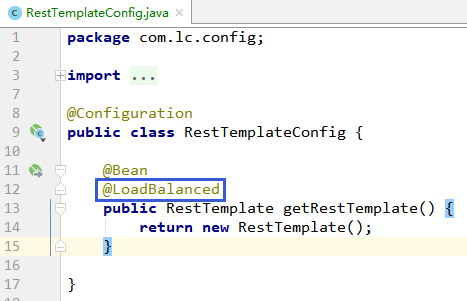
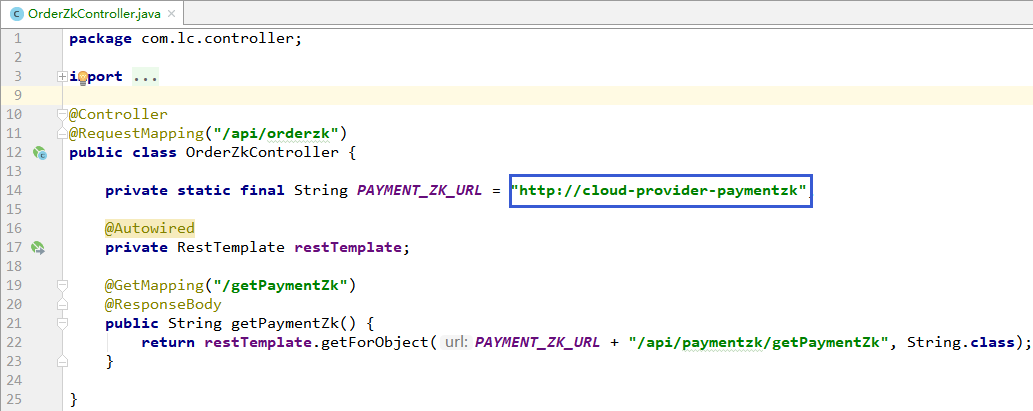
① 启动zk控制台，项目导入jar包依赖，需要注意控制版本；

<**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-zookeeper-discovery</**artifactId**>  
 *<!--先排除自带的zookeeper3.5.3-->* <**exclusions**>  
 <**exclusion**>  
 <**groupId**>org.apache.zookeeper</**groupId**>  
 <**artifactId**>zookeeper</**artifactId**>  
 </**exclusion**>  
 </**exclusions**>  
</**dependency**>  
*<!--添加zookeeper3.4.9版本-->*<**dependency**>  
 <**groupId**>org.apache.zookeeper</**groupId**>  
 <**artifactId**>zookeeper</**artifactId**>  
 <**version**>3.4.9</**version**>  
</**dependency**>

② 调用端和服务提供端配置application.yml，注入到zk中



③ 调用端再需要配置RestTemplate，远程进行调用服务端接口。



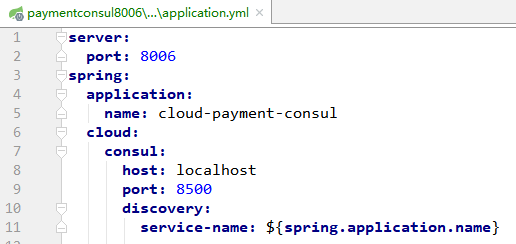
1. Consul：使用consul注册中心

①下载并执行consul.exe文件，在解压目录下，cmd->consul --version查看版本，使用开发模式启动consul：consul agent -dev，地址localhost:8500可以查看consul注册管理页面。

②调用端和服务提供端导入jar包依赖：

<**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-consul-discovery</**artifactId**>  
</**dependency**>

③调用端和服务提供端配置application.yml：



④调用端配置RestTemplate进行远程调用服务提供端，服务名称调用注意添加注解@LoadBalanced。

1. Eureka、Zookeeper和Consul的区别：

Eureka保证ap，保证可用性和分区容错性；

Zookeeper保证cp，保证一致性和分区容错性；

Consul保证cp，保证一致性和分区容错性；

1. Ribbon：功能服务端调用，负载均衡（默认轮询方式） + RestTemplate服务调用；
2. Ribbon：使用ribbon，详细介绍使用RestTemplate服务调用：

①如果存在Eureka并自带ribbon的jar包可以使用现有的，不存在引入jar包：

**<dependency>**

**<groupld>org.springframework.cloud</groupld> <artifactld>spring-cloud-starter-netflix-ribbon</artifactid>**

**</dependency>**

②RestTemplate方法getForObject和getForEntity的区别：getForObject返回的json串，getForEntity返回ResponseEntity对象，包含响应头，响应体等具体信息；

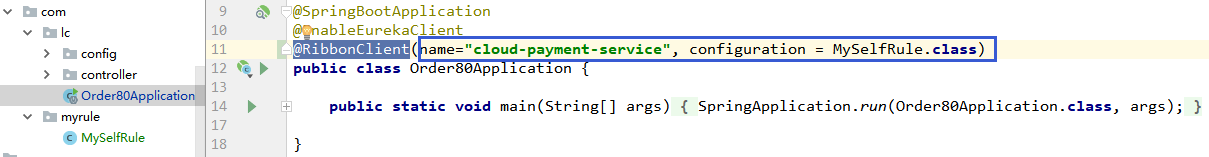
1. Ribbon：将默认的轮询算法改为随机算法服务调用：

①在服务调用端实现，切记更改的负载算法不能在@ComponentScan所扫描的当前包下以及子包下；

②改为随机算法方式：



③在主启动类上使用注解@RibbonClient并表明调用的服务提供端及使用的算法：

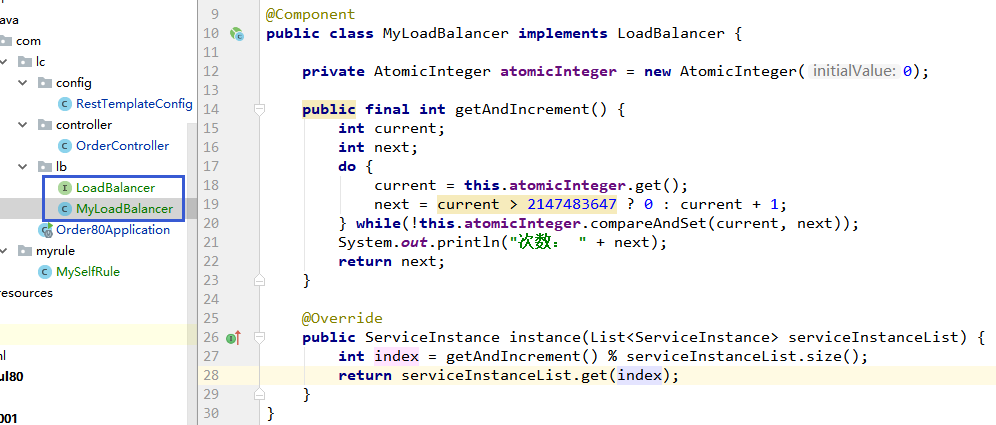


1. Ribbon：手写轮询算法

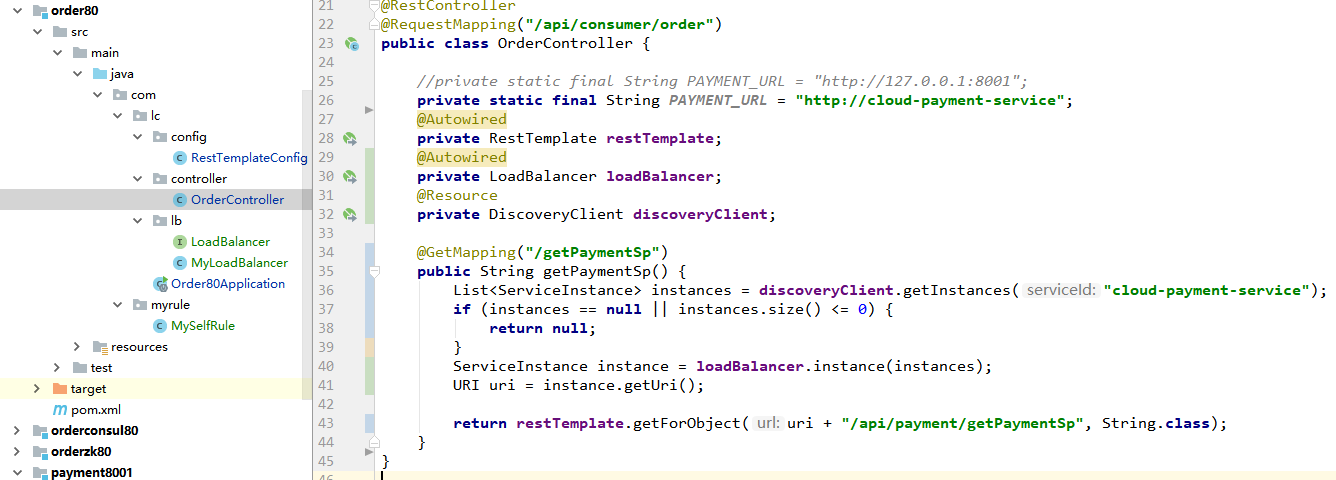
①原理：rest接口第几次请求数 % 服务器集群总数量 = 实际调用服务器位置下标；

②去除Ribbon自带的@LoadBalanced负载注解；

③定义接口和实现类，构造自定义轮询算法；



④调用服务提供端接口：



1. OpenFeign：功能服务端调用，服务接口绑定器，接口 + 注解；
2. OpenFeign：