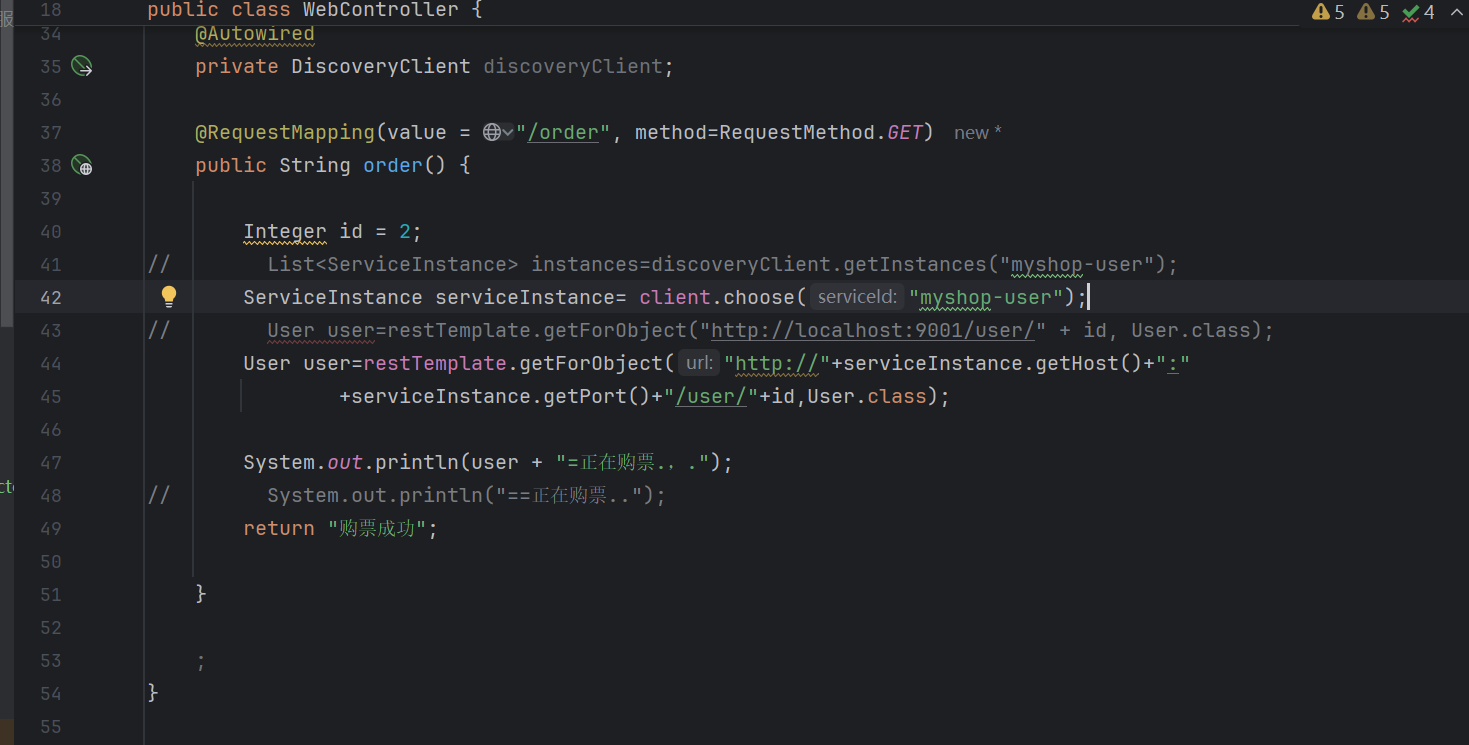
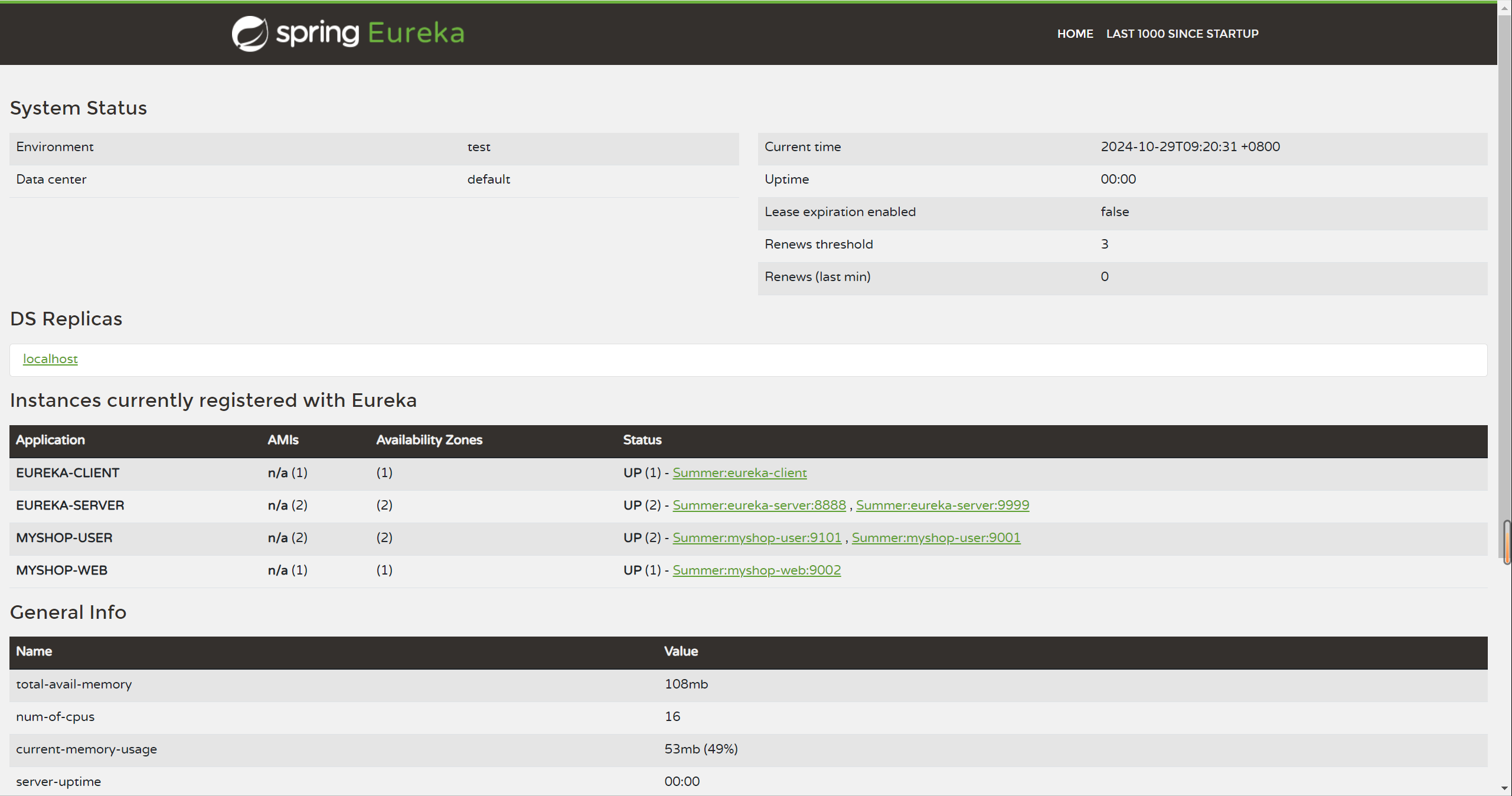
# Lab4 Spring Cloud服务调用与负载均衡

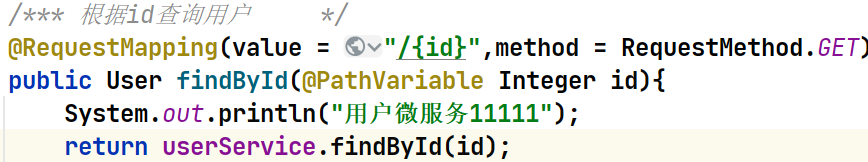
1. 实现RestTemplate（服务调用）+Ribbon（负载均衡）
2. 修改购票微服务Webcontroller类代码：1）注入负载均衡客户端LoadBalancerClient；2）使用Ribbon的choose方法选择合适的服务





1. 启动两个端口分别为9001和9101的myshop-user用户微服务（参考lab3启动两个eureka-server步骤）。
2. 修改用户微服务9001的controller代码的findById方法，增加println（“用户微服务11111”）

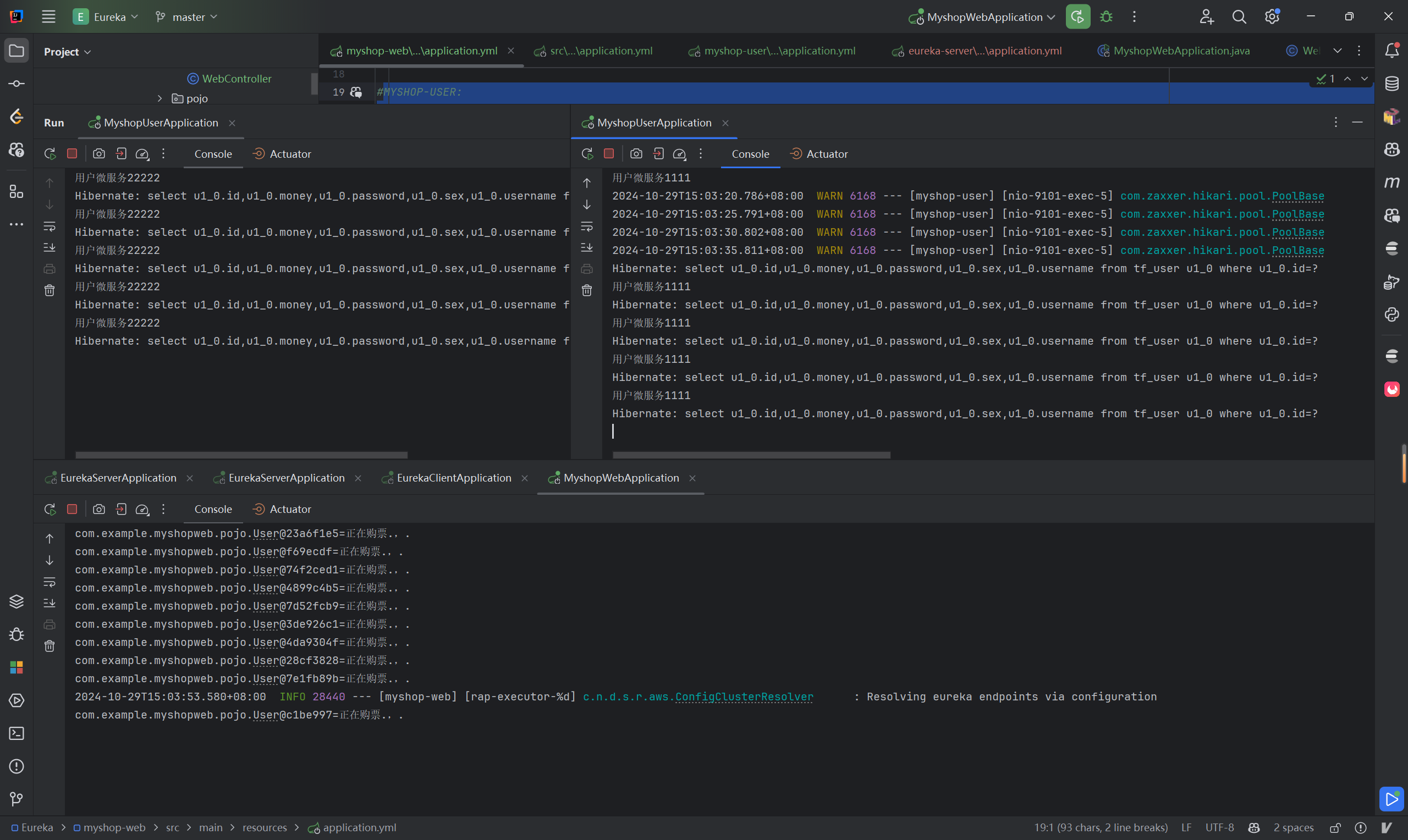


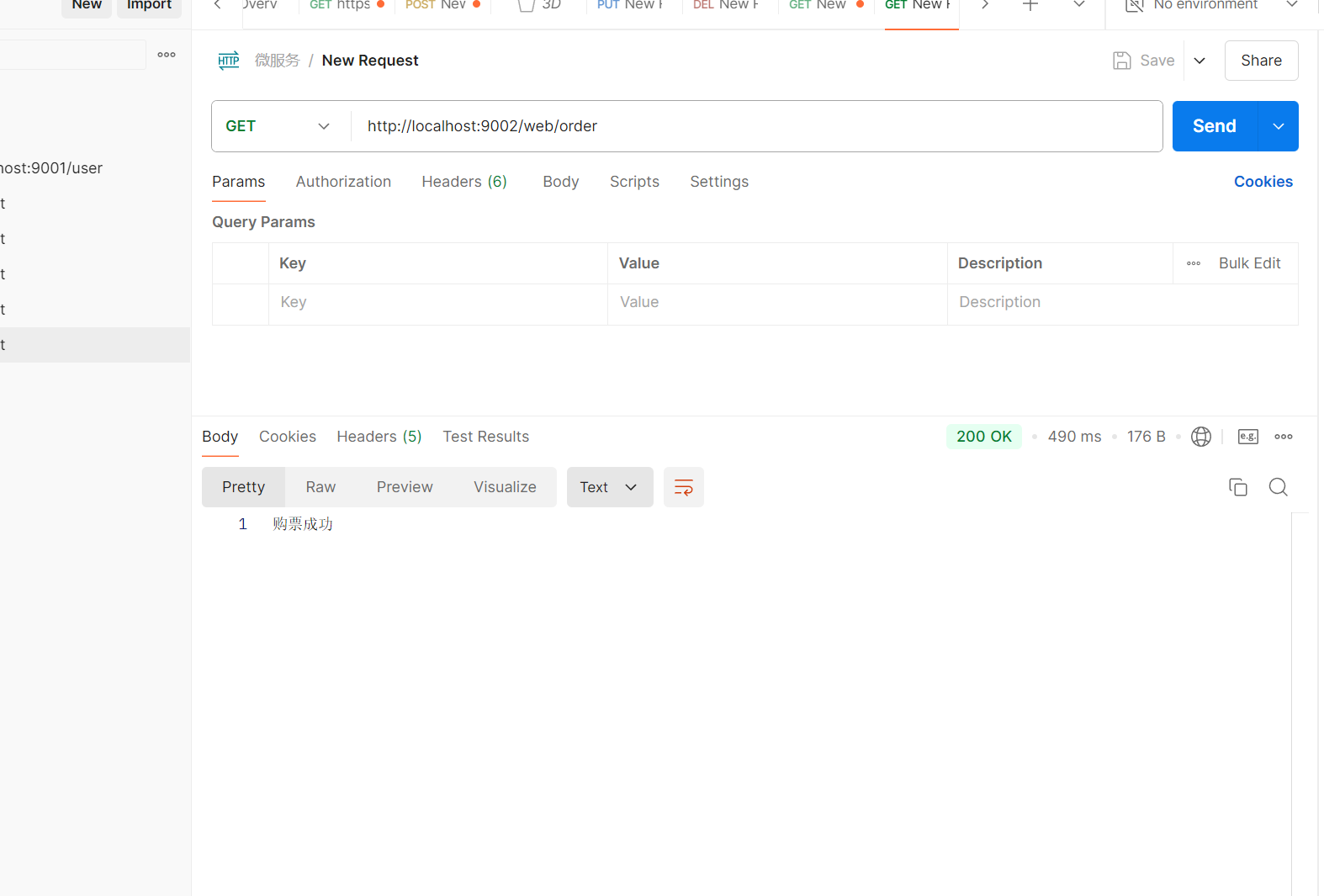


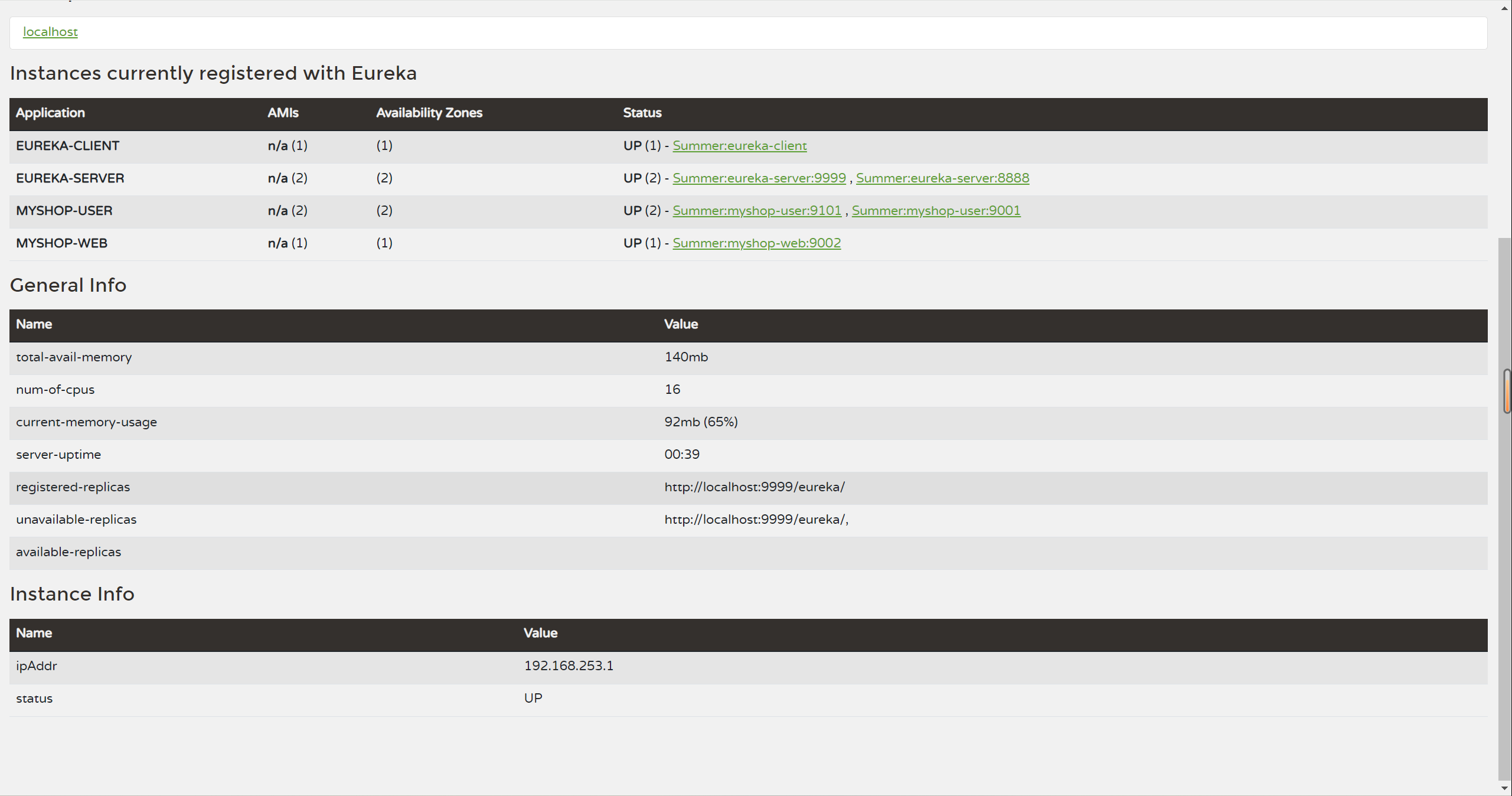
2）修改用户微服务9101的UserController代码的findById方法，增加println（“用户微服务22222”）

1. 重启购票微服务，并多次请求该服务order（）方法，观察调用的是哪个用户微服务？并记录请求结果、eureka界面及IDEA后台信息，此时ribbon采用的是何种负载均衡算法？

采取轮询算法，请求依次分配（左右各五次请求）







1. 修改Ribbon负载均衡算法为随机算法

1）购票微服务application.yml中增加：

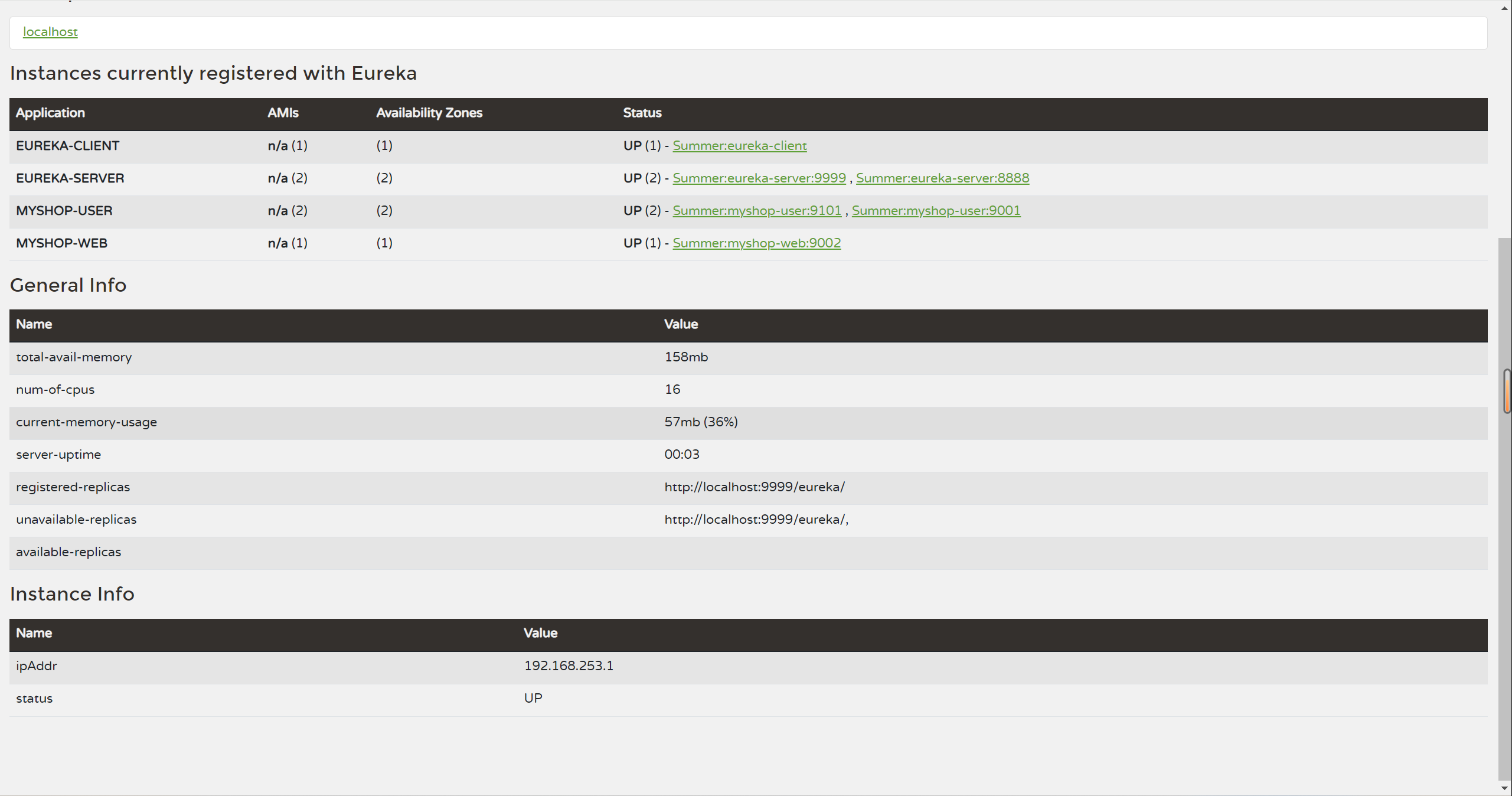
# 修改Ribbon的负载均衡算法

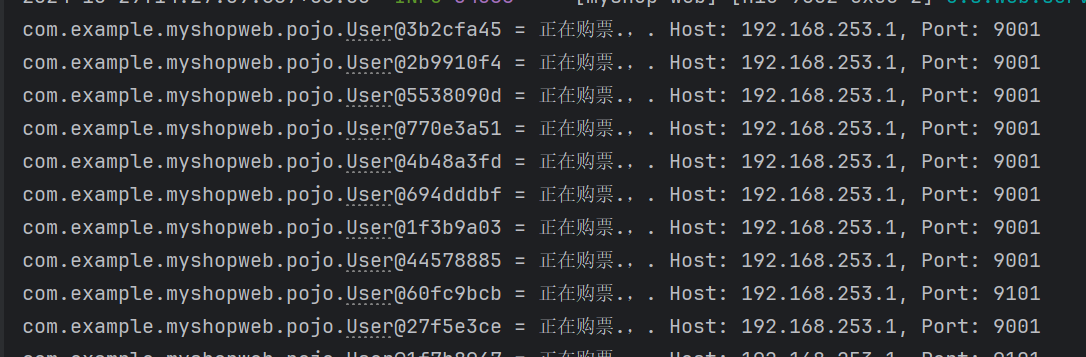
myshop-user:

ribbon:

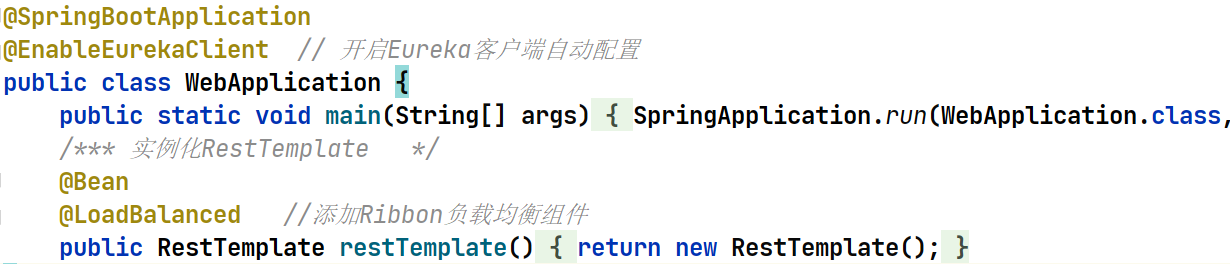
NFLoadBalancerRuleClassName: com.netflix.loadbalancer.RandomRule

1. 重启购票微服务，并多次请求购票微服务，观察并记录IDEA后台被调用微服务信息





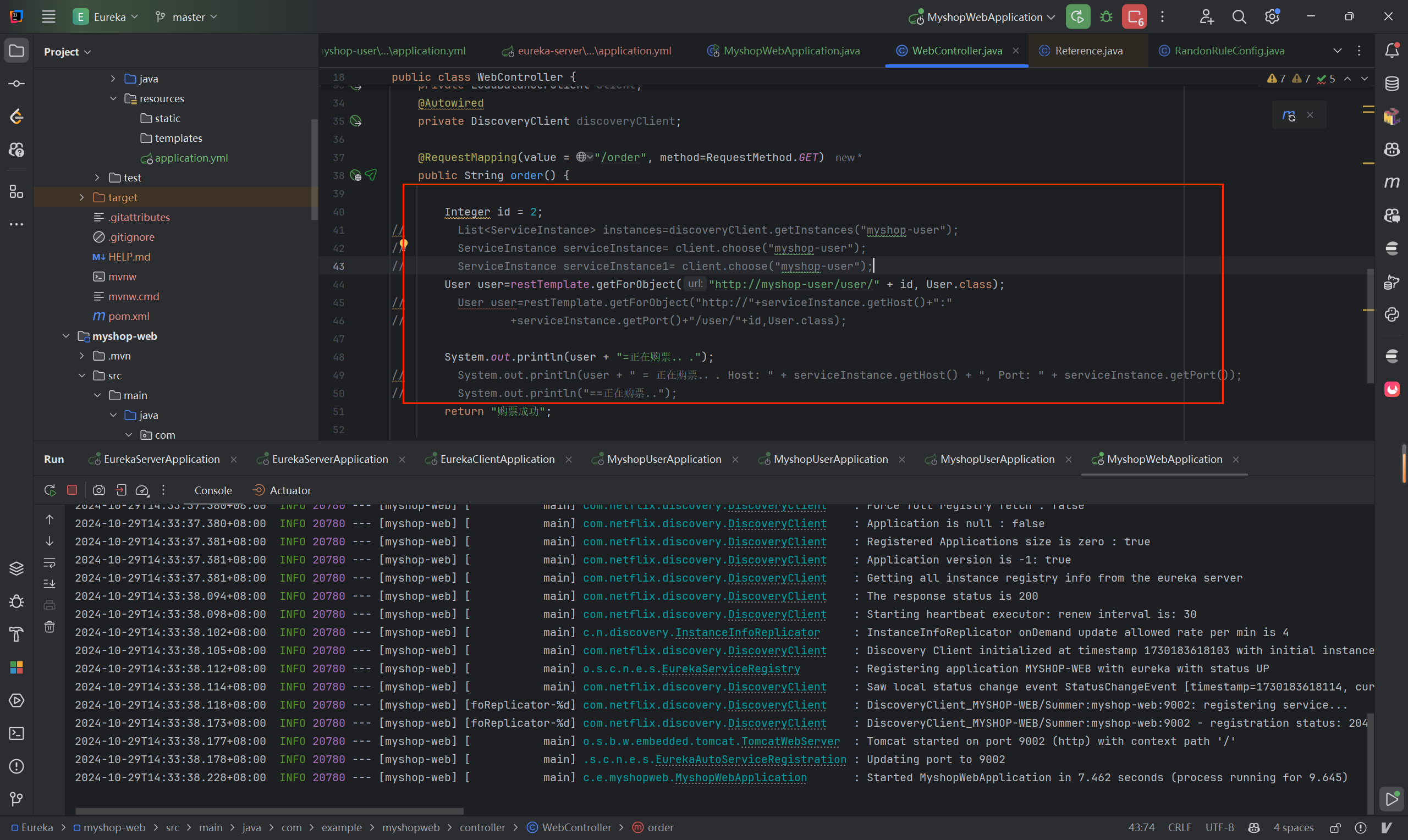
1. 简化购票微服务
2. 在启动类的RestTemplate实例方法上添加@LoadBalanced注解

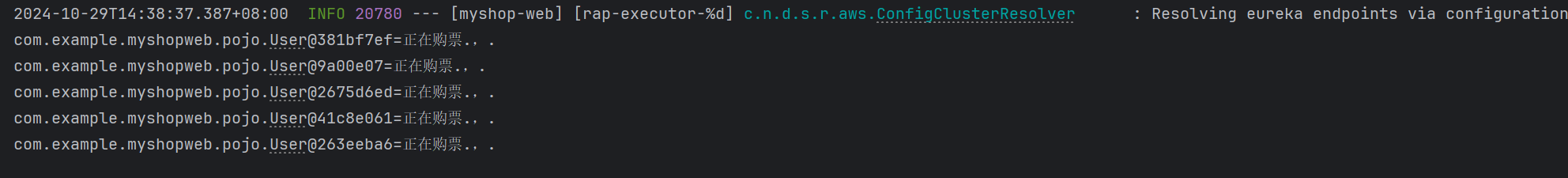


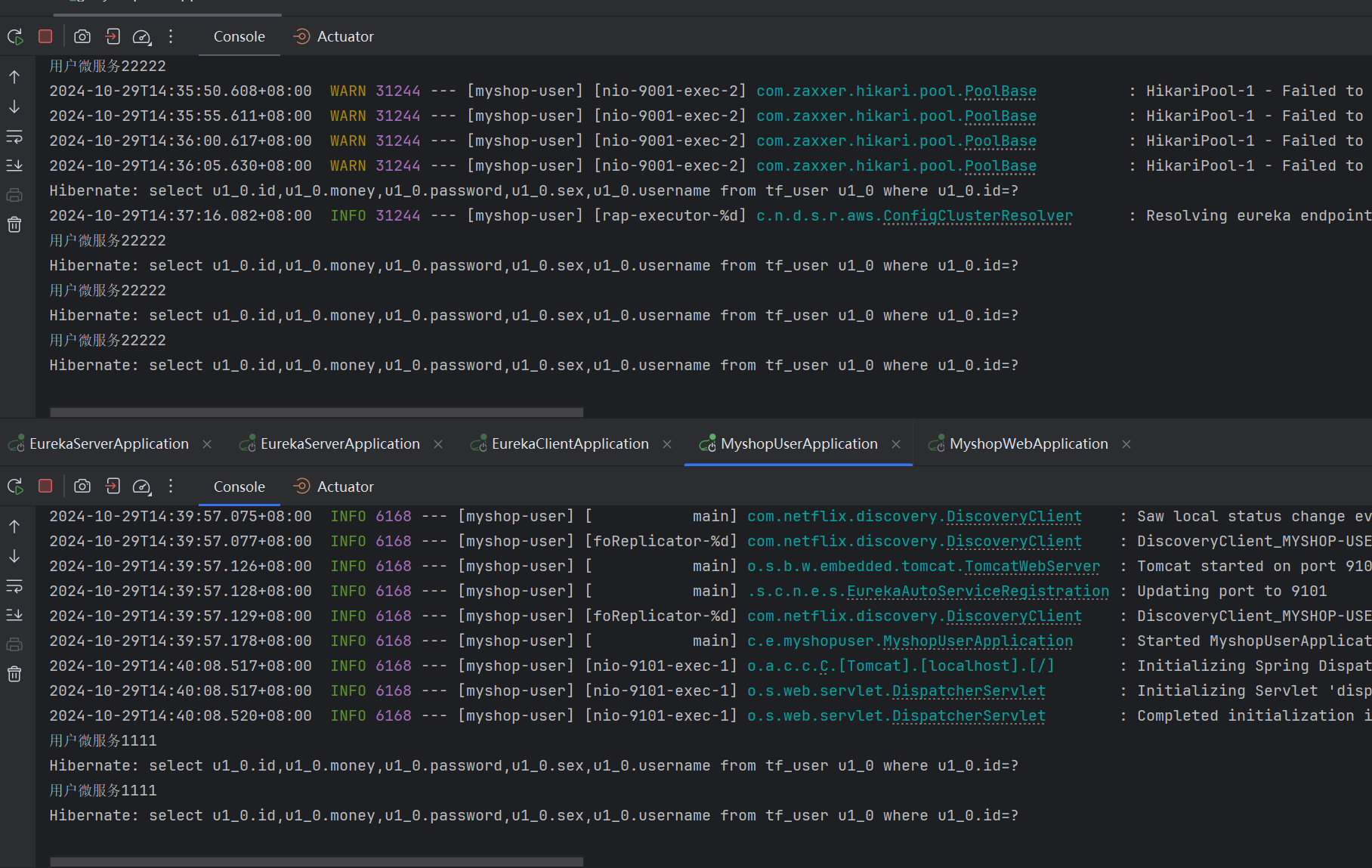
1. 修改WebController类的order方法代码

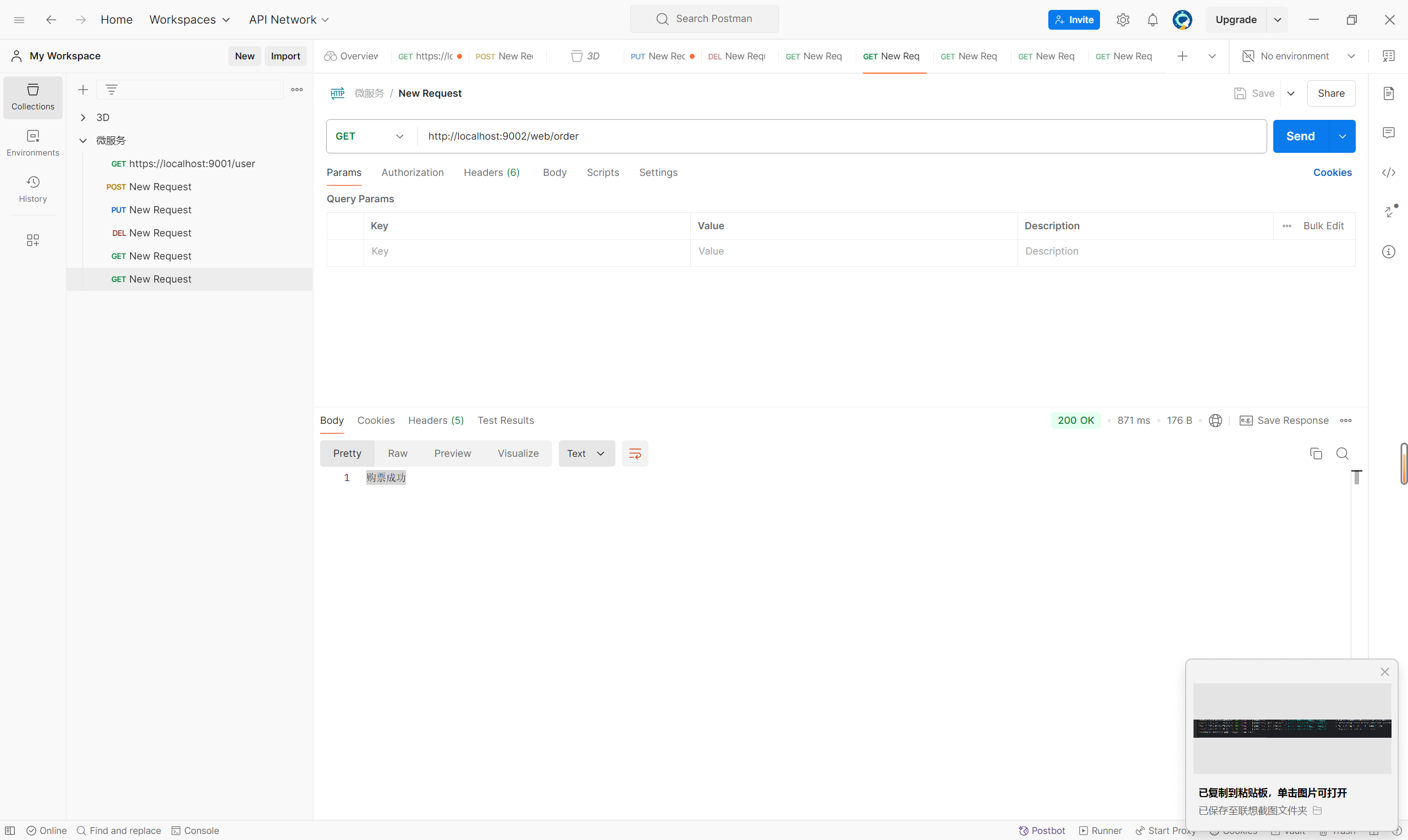


3）重启购票微服务，观察是否能正常调用用户微服务。









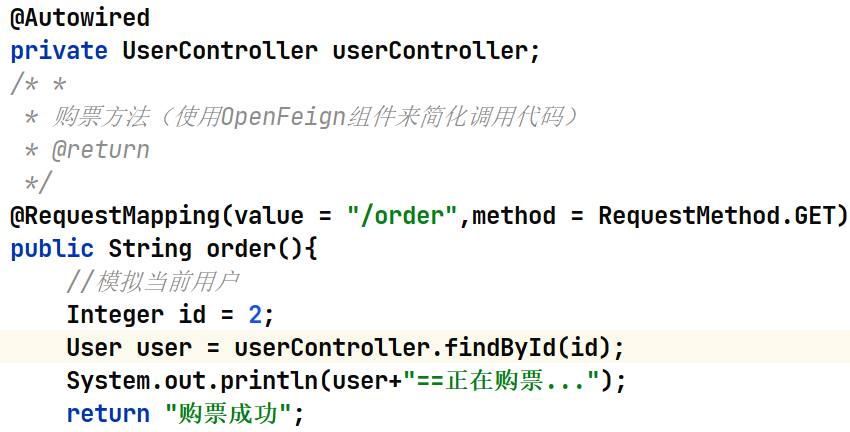
1. OpenFeign（服务调用）+ 内置Ribbon（负载均衡）
2. 服务调用者-购票微服务的pom.xml中导入OpenFeign的依赖

<!-- 导入OpenFeign依赖 -->  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
</dependency>

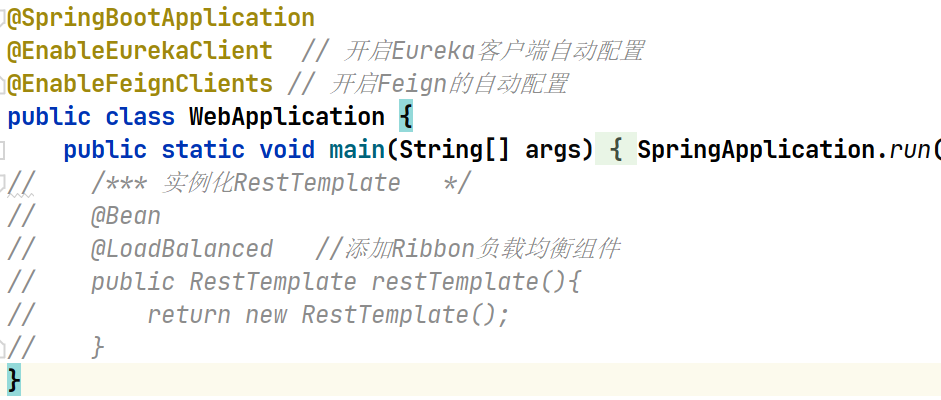
1. 编写调用方代理接口（创建在client包下），使用@FeignClient指定调用的服务名

/\*\* \* 用户微服务远程接口  
 \* 三个注意事项：  
 \* 1）使用@FeignClient，声明需要调用的微服务  
 \* 2) 检查@RequesetMapping注解，value值（路径）是否完整  
 \* 3) @PathVariable注解的value不能省略的  
 \*/  
@FeignClient(value = "myshop-user")  
public interface UserController {  
 @RequestMapping(value = "/user/{id}", method = RequestMethod.GET)  
 public User findById(@PathVariable(value = "id") Integer id);  
}

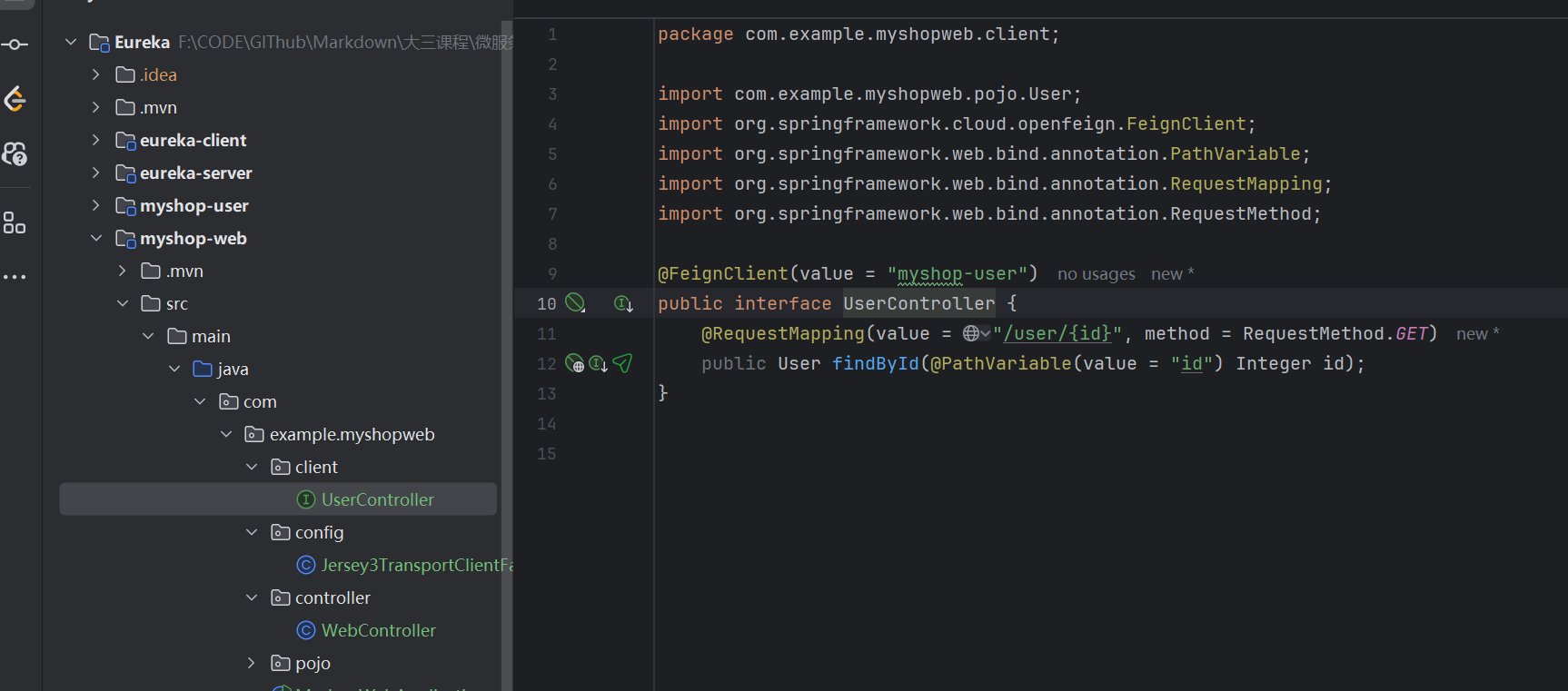
1. 购票微服务WebController类中使用代理接口调用远程服务方法，替换掉以前的restTemplate.getForObject方法

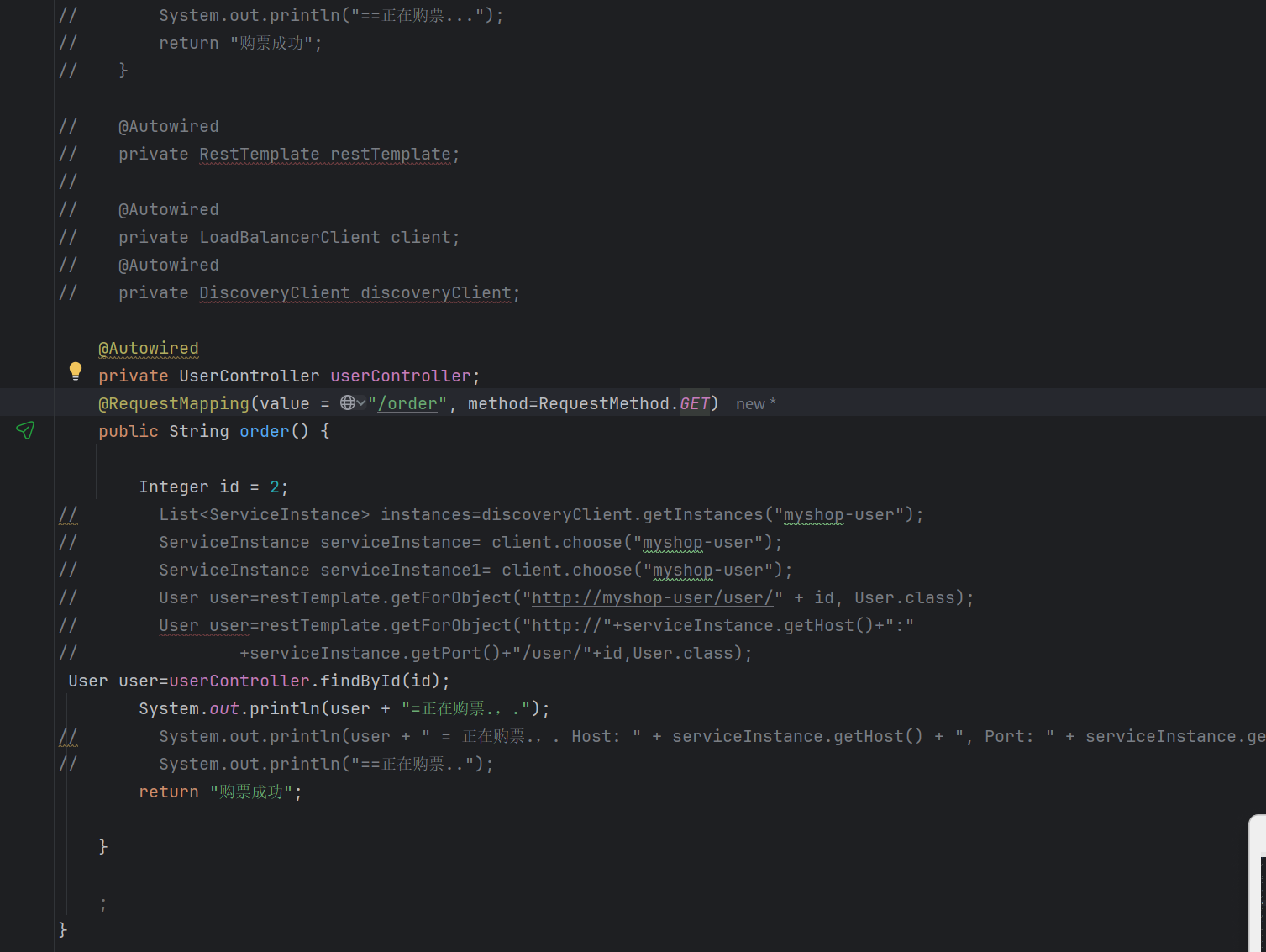
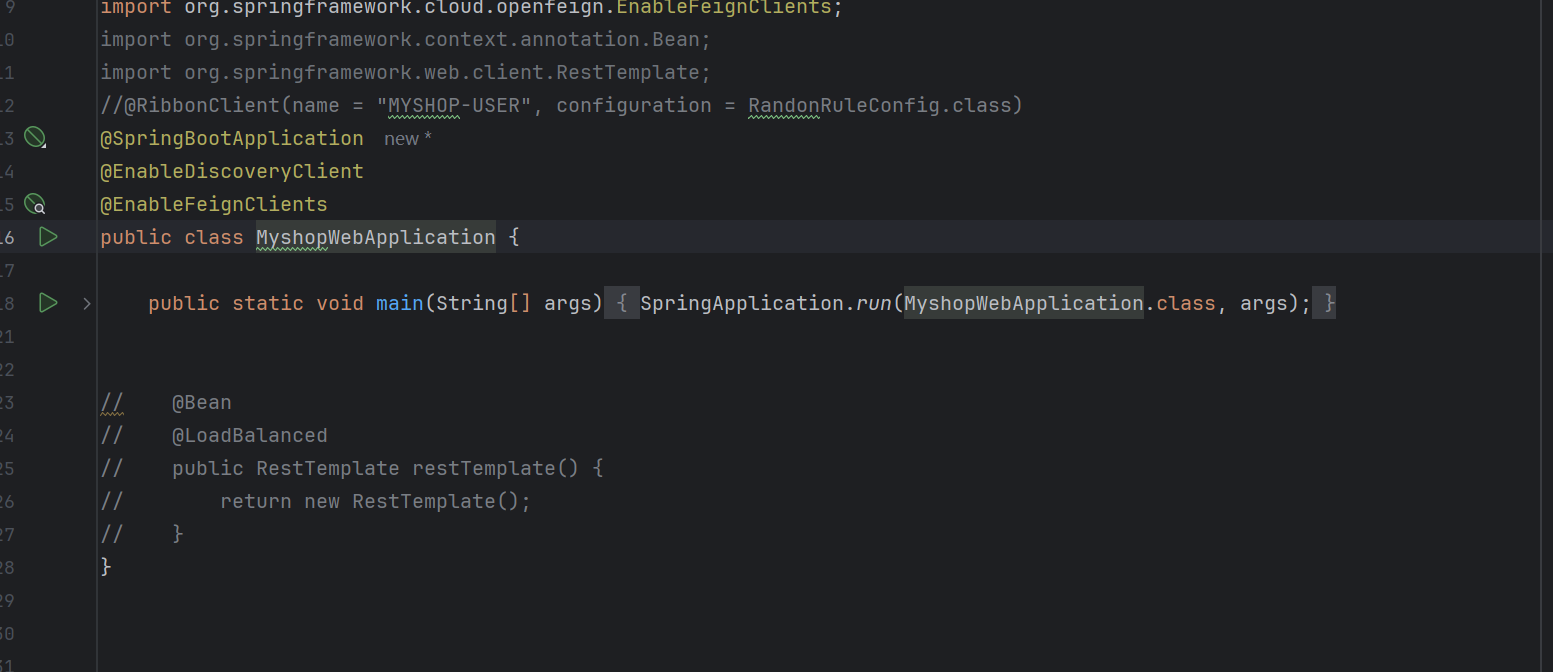


1. 购票微服务启动类添加@EnableFeignClients注解，注释掉restTemplate



1. 重启购票微服务，并多次请求该服务order（）方法，观察如何调用用户微服务的。并记录请求结果及IDEA后台信息，此时内置的ribbon采用的是何种负载均衡算法？



轮询算法

