注册中心:

server.port=8001  
eureka.instance.hostname=localhost  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false  
eureka.client.service-url.defaultZone=http://${eureka.instance.hostname}:${server.port}/eureka/

服务提供者:

#注册中心  
eureka:  
 client:  
 serviceUrl:  
 defaultZone: <http://localhost:8001/eureka/>

//注册中心的地址  
server:  
 port: 8770

//服务端口号  
spring:  
 application:  
 name: recite-admin

//服务名，与消费者中的feign对应

保持原有的MVC架构 进行访问就可以了

注意

Controller层的@RequestMapping的值需要与消费者中feign接口里面的相对应

@EnableEurekaClient //消费者注意打上这个注解  
@SpringBootApplication  
public class AdminApplication {

消费者:

#注册中心  
eureka:  
 client:  
 serviceUrl:  
 defaultZone: <http://localhost:8001/eureka/>

//注册中心的地址  
server:  
 port: 8770 //服务端口号  
spring:  
 application:  
 name: recite-admin

//服务名，与消费者中的feign对应

配置文件与提供者配置相似

@SpringBootApplication  
@EnableDiscoveryClient //消费者注意打上这两个注解  
@EnableFeignClients  
public class ConsumerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(ConsumerApplication.class, args);  
 }

}

Feign为连接消费者与提供者的桥梁

访问消费者的Controller，调用对应的feign的对应接口

Controller:

@RestController  
public class ReaderController {  
  
 @Autowired  
 ReaderFeign readerFeign;  
  
 //课本信息  
 //获取全部课本信息  
 @GetMapping(value = "/v0.1/classbook")  
 Map<String, Object> findClassbook(@RequestParam("pageNumber") int pageNumber){  
 return readerFeign.findClassbook(pageNumber);  
 }

}

Feign:

@FeignClient(value = "recite-reader")  
public interface ReaderFeign {  
 @GetMapping(value = "/v0.1/classbook")

//这个Mapping一定要与提供者的controller里面的方法对应

//RequestParam一定要给name，不然提供者不能识别

//如果参数是实体类需要打上@RequestBody  
 Map<String, Object> findClassbook(@RequestParam("pageNumber") int pageNumber);

}