1. 概述

1.1. 是什么

1.1.1. Github官网

Github官网: https://github.com/spring-cloud/spring-cloud-openfeign

1.1.2. 简要说明

Feign是一个声明式的Web服务客户端,让编写Web服务客户端变得非常容易,只需创建一个接口并在接口上添加注解即可

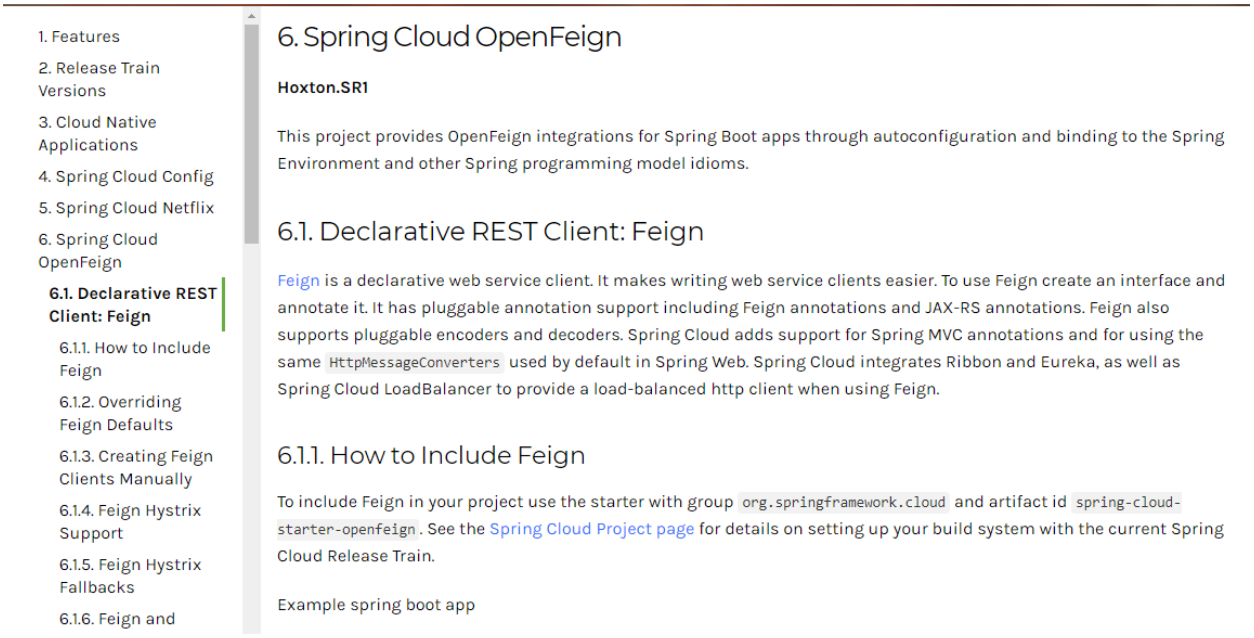
1.1.3. 官网解释

https://cloud.spring.io/spring-cloud-static/Hoxton.SR1/reference/htmlsingle/#spring-cloud-openfeign

Feign是一个声明式 WebService客户端。使用 Feign能让编写 Web service客户端更加简单。

它的使用方法是定义一个服务接口然后在上面添加注解。Feign也支持可拔插式的编码器和解码器。

Spring Cloud对 Feign进行了封装使其支持了 Spring MVC标准注解和 HttpMessageConverters。Feign可以与 Eureka和 Ribbon组合使用以支持负载均衡



1.2. 能干啥

Feign能干什么

- Feign旨在使编写 Java Http客户端变得更容易。
- 前面在使用 Ribbon + RestTemplate时，利用 RestTemplate对http请求的封装处理，形成了一套模版化的调用方法。但是在实际开发中，由于对服务依赖的调用可能不止一处，往往一个接口会被多处调用，所以通常都会针对每个微服务自行封装一些客户端类来包装这些依赖服务的调用。所以，Feign在此基础上做了进一步封装，由他来帮助我们定义和实现依赖服务接口的定义。在 Feign的实现下我们只需创建一个接口并使用注解的方式来配置它（以前是Dao接口上面标注 Mappe注解现在是一个微服务接口上面标注一个Feign注解即可），即可完成对服务提供方的接口绑定，简化了使用 Spring Cloud Ribbon时，自动封装服务调用客户端的开发量。

Feign集成了Ribbon

- 利用Rbom维护了 Payment的服务列表信息，并且通过轮询实现了客户端的负载均衡。而与Ribbon不同的是，通过Feign只需要定义服务绑定接口且以声明式的方法，优雅而简单的实现了服务调用

1.3. Feign和OpenFeign的区别

Feign	OpenFeign
Feign是Spring Cloud组件中的一个轻量级RESTful的HTTP服务客户端 Feign内置了Ribbon，用来做客户端负载均衡，去调用服务注册中心的服务。Feign的使用方式是：使用Feign的注解定义接口，调用这个接口，就可以调用服务注册中心的服务	OpenFeign是Spring Cloud 在Feign的基础上支持了SpringMVC的注解，如@RequesMapping等等。OpenFeign的@FeignClient可以解析SpringMVC的@RequestMapping注解下的接口，并通过动态代理的方式产生实现类，实现类中做负载均衡并调用其他服务。

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

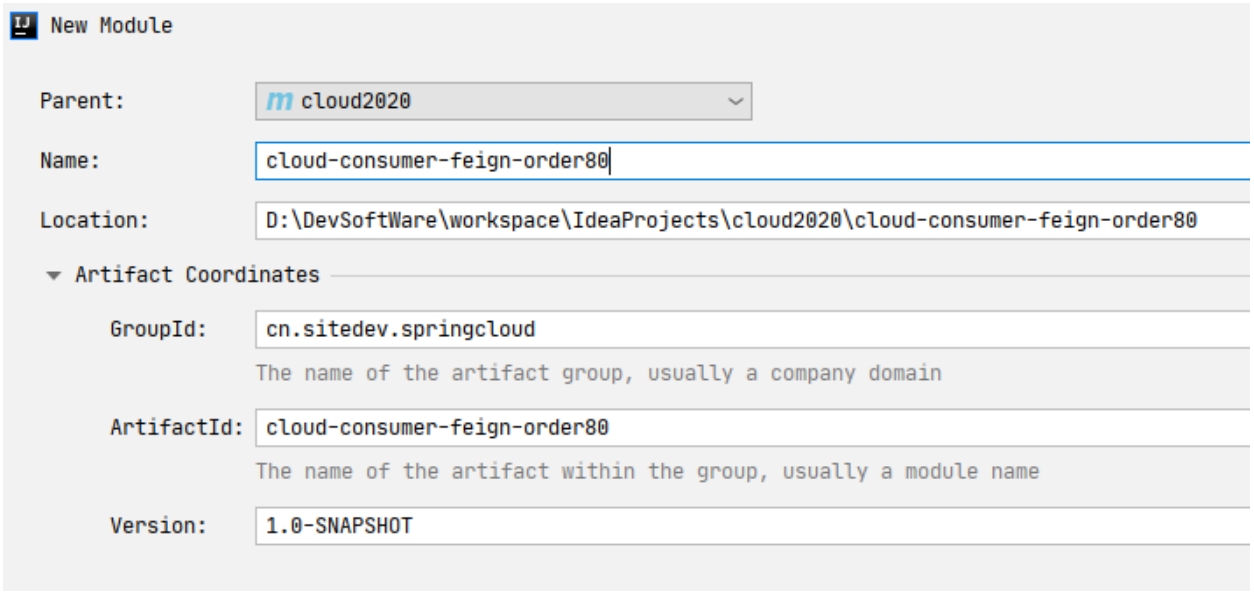
2. OpenFeign使用步骤

2.1. 使用方式: 接口 + 注解

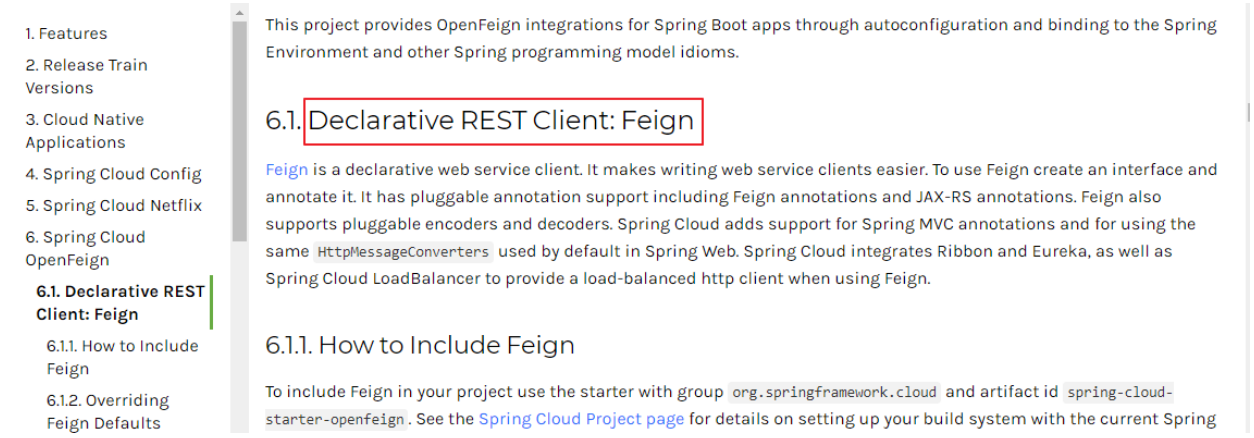
微服务调用接口 + @FeignClient

2.2. 新建cloud-consumer-feign-order80

新建模块cloud-consumer-feign-order80



> 注意: Feign在消费端使用



2.3. POM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
```

```
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-consumer-feign-order80</artifactId>
    <description>订单消费者之feign</description>

    <dependencies>
        <!--openfeign-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
        <!--eureka client-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--监控-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <!--热部署-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
```

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

</project>
```

2.4. YML

```
server:
  port: 80
eureka:
  client:
    register-with-eureka: false
    fetch-registry: true
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka,http://eureka7002.com:7002/eure
```

2.5. 主启动

```
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients
public class OrderFeignMain80 {
    public static void main(String[] args) {
        SpringApplication.run(OrderFeignMain80.class, args);
    }
}
```

2.6. 业务类

2.6.1. 使用方式: 业务逻辑接口+@FeignClient配置调用provider服务

注意主启动类上的@EnableFeignClients注解要和业务接口上的@FeignClient配合使用

2.6.2. 新建PaymentFeignService接口并新增注解@FeignClient

```
package cn.sitedev.springcloud.service;

import cn.sitedev.springcloud.entities.CommonResult;
import cn.sitedev.springcloud.entities.Payment;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@FeignClient(value = "CLOUD-PAYMENT-SERVICE")
public interface PaymentFeignService {
    @GetMapping(value = "/payment/get/{id}")
    CommonResult<Payment> getPaymentById(@PathVariable("id") Long id);
}
```

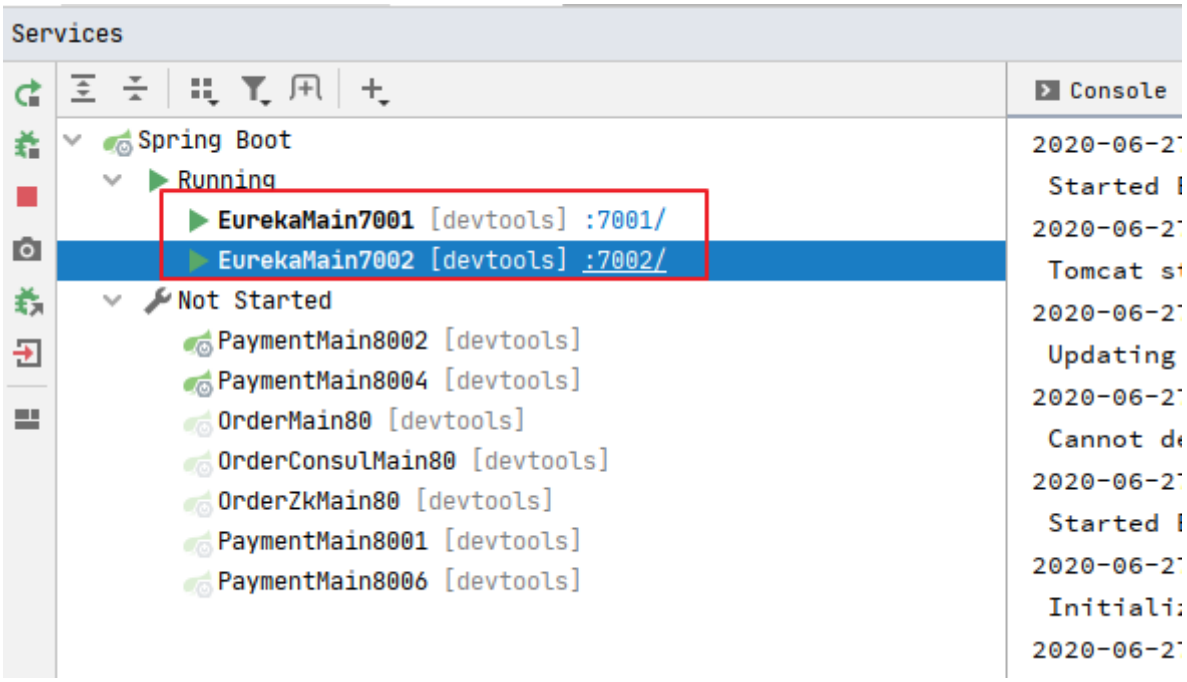
2.6.3. 控制层Controller

```
package cn.sitedev.springcloud.controller;

import cn.sitedev.springcloud.entities.CommonResult;
import cn.sitedev.springcloud.entities.Payment;
import cn.sitedev.springcloud.service.PaymentFeignService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;

@RestController
@Slf4j
public class OrderFeignController {
    @Resource
```

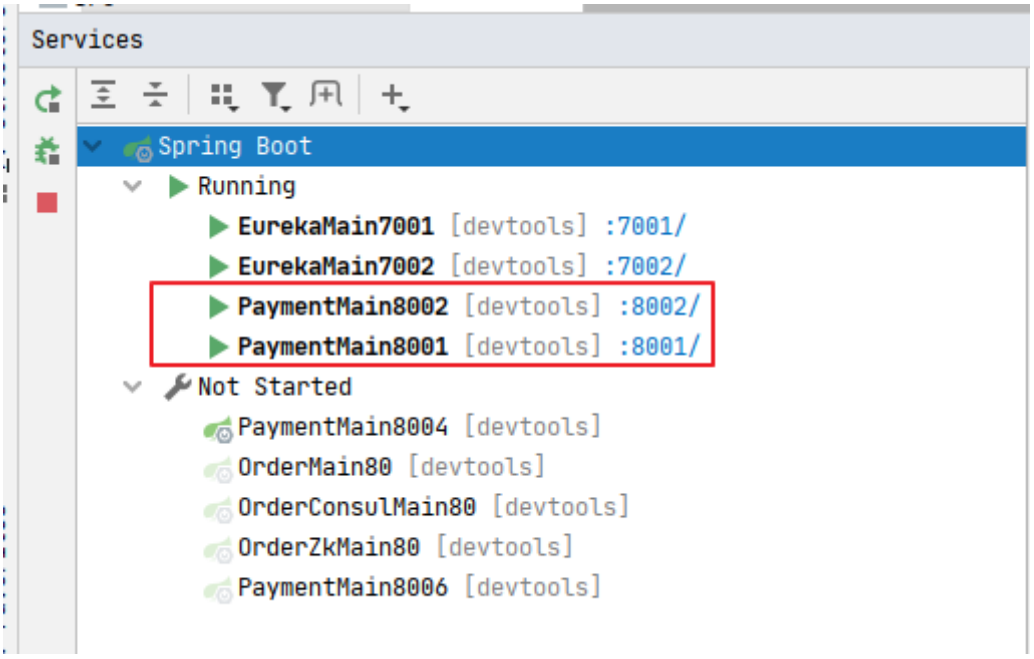
```
17      private PaymentFeignService paymentFeignService;
18
19      @GetMapping(value = "/consumer/payment/get/{id}")
20      public CommonResult<Payment> getPaymentById(@PathVariable("id") Long id) {
21          return paymentFeignService.getPaymentById(id);
22      }
23  }
```

2.7. 测试

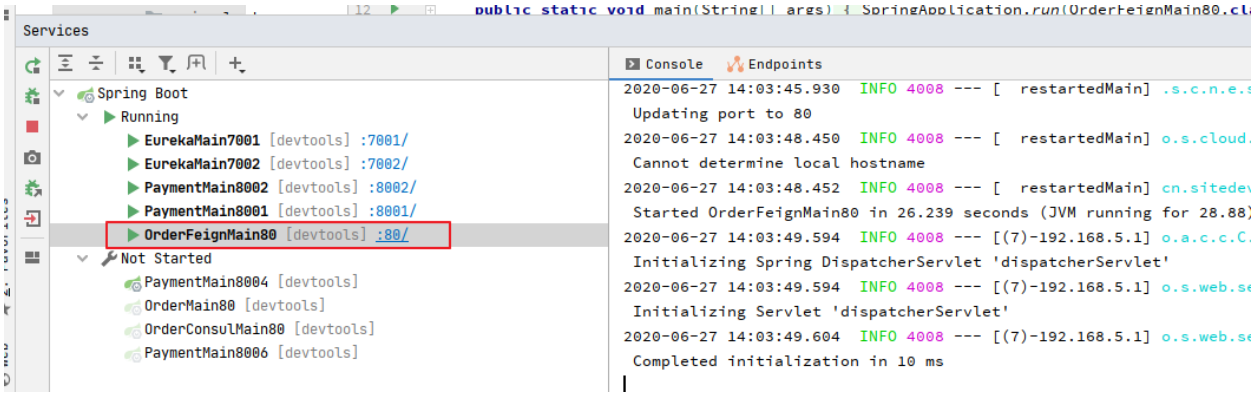
先启动2个eureka集群7001/7002



再启动2个微服务8001/8002



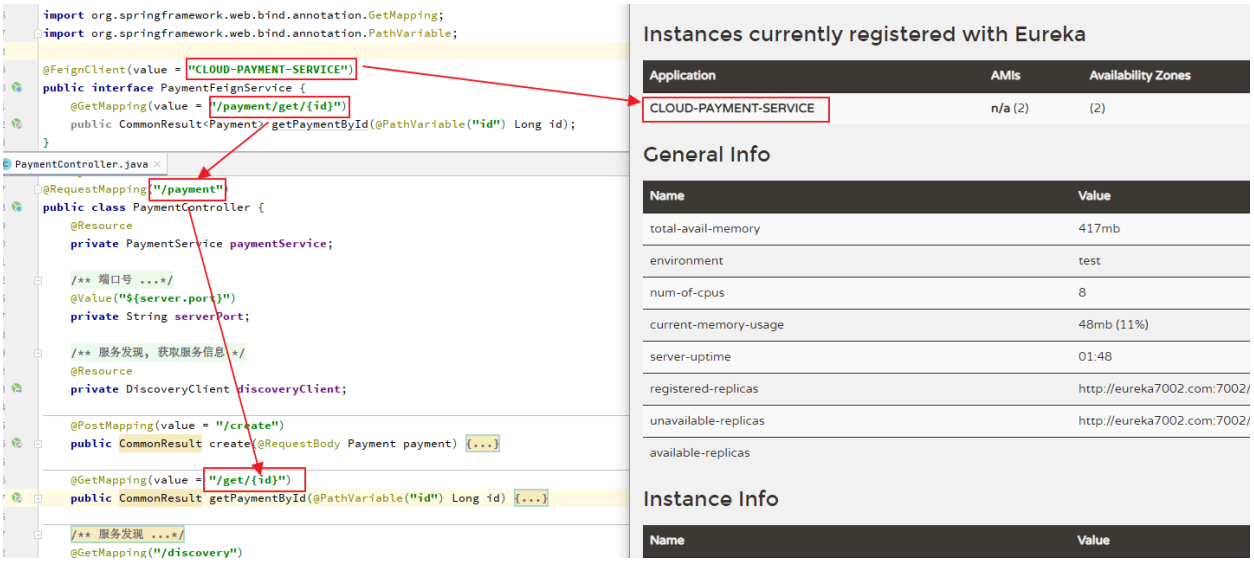
启动集成了OpenFeign功能的微服务80



浏览器访问 http://localhost/consumer/payment/get/1

> Feign自带负载均衡配置项

2.8. 小总结



3. OpenFeign超时控制

3.1. 超时设置, 故意设置超时演示出错情况

3.1.1. 服务提供方8001故意写暂停程序

Controller层添加一个超时方法paymentFeignTimeout:

```
package cn.sitedev.springcloud.controller;

import cn.sitedev.springcloud.entities.CommonResult;
import cn.sitedev.springcloud.entities.Payment;
import cn.sitedev.springcloud.service.PaymentService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.client.ServiceInstance;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.web.bind.annotation.*;

import javax.annotation.Resource;
import java.util.List;
import java.util.concurrent.TimeUnit;

@RestController
@Slf4j
@RequestMapping("/payment")
public class PaymentController {
    @Resource
    private PaymentService paymentService;
```

```
22
23      /**
24       * 端口号
25       * 查看负载均衡效果
26       */
27      @Value("${server.port}")
28      private String serverPort;
29
30      /**
31       * 服务发现，获取服务信息
32       */
33      @Resource
34      private DiscoveryClient discoveryClient;
35
36      @PostMapping(value = "/create")
37      public CommonResult create(@RequestBody Payment payment) {
38          int result = paymentService.create(payment);
39          log.info("*****插入结果: " + result);
40          if (result > 0) {
41              return new CommonResult(200, "插入数据库成功，serverPort: " + serverPort
42          } else {
43              return new CommonResult(444, "插入数据库失败", null);
44          }
45      }
46
47      @GetMapping(value = "/get/{id}")
48      public CommonResult getPaymentById(@PathVariable("id") Long id) {
49          Payment payment = paymentService.getPaymentById(id);
50          log.info("*****查询结果: " + payment);
51          if (payment != null) {
52              return new CommonResult(200, "查询成功，serverPort: " + serverPort, paym
53          } else {
54              return new CommonResult(444, "没有对应记录，查询id: " + id, null);
55          }
56      }
57
58      /**
59       * 服务发现
60       *
61       * @return
62       */
63      @GetMapping("/discovery")
64      public Object discovery() {
```

```
        List<String> services = discoveryClient.getServices();
        for (String element : services) {
            log.info("*****element: " + element);
        }
        // 一个微服务下的全部实例
        List<ServiceInstance> instances = discoveryClient.getInstances("CLOUD-PAYMEN
        for (ServiceInstance instance : instances) {
            log.info(instance.getServiceId() + "\t" + instance.getHost() + "\t" + in
        }
        return discoveryClient;
    }

    @GetMapping(value = "/lb")
    public String getPaymentLB() {
        return serverPort;
    }

    @GetMapping(value = "/feign/timeout")
    public String paymentFeignTimeout() {
        try {
            TimeUnit.SECONDS.sleep(3);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return serverPort;
    }
}
```

3.1.2. 服务消费方80添加超时方法PaymentFeignService

添加一个超时方法paymentFeignTimeout

```
package cn.sitedev.springcloud.service;

import cn.sitedev.springcloud.entities.CommonResult;
import cn.sitedev.springcloud.entities.Payment;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@FeignClient(value = "CLOUD-PAYMENT-SERVICE")
public interface PaymentFeignService {
```

```
    @GetMapping(value = "/payment/get/{id}")
    CommonResult<Payment> getPaymentById(@PathVariable("id") Long id);

    @GetMapping(value = "/payment/feign/timeout")
    String paymentFeignTimeout();

}
```

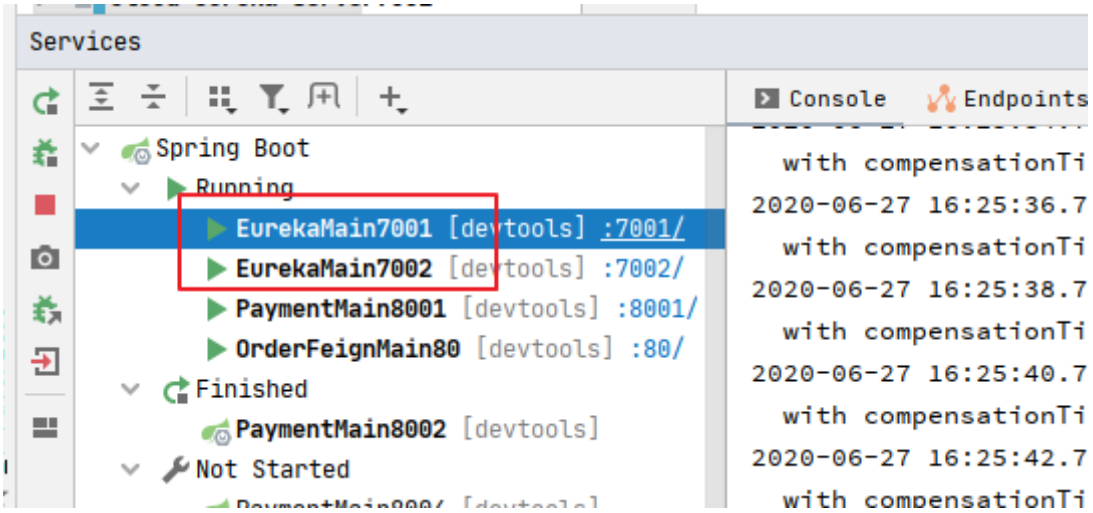
3.1.3. 服务消费方80添加超时方法OrderFeignController

添加一个超时方法paymentFeignTimeout

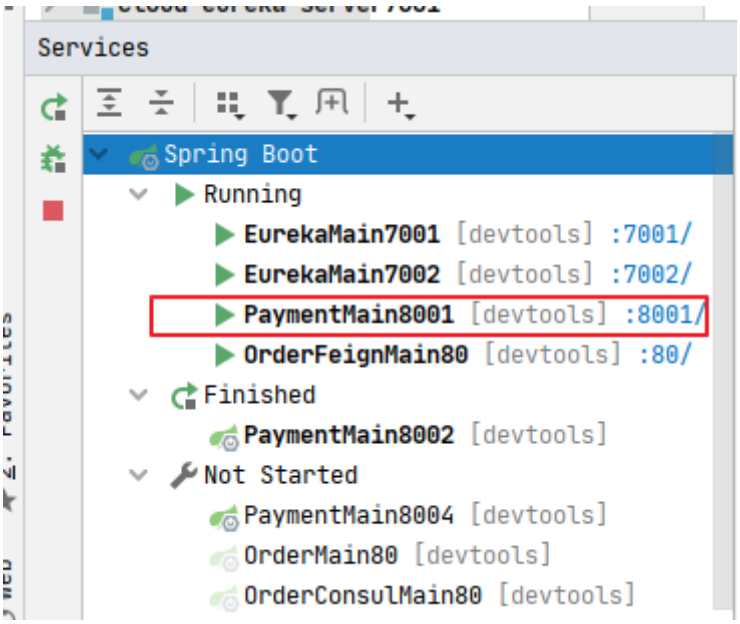
```
package cn.sitedev.springcloud.controller;

import cn.sitedev.springcloud.entities.CommonResult;
import cn.sitedev.springcloud.entities.Payment;
import cn.sitedev.springcloud.service.PaymentFeignService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;

@RestController
@Slf4j
public class OrderFeignController {
    @Resource
    private PaymentFeignService paymentFeignService;

    @GetMapping(value = "/consumer/payment/get/{id}")
    public CommonResult<Payment> getPaymentById(@PathVariable("id") Long id) {
        return paymentFeignService.getPaymentById(id);
    }

    @GetMapping(value = "/consumer/payment/feign/timeout")
    public String paymentFeignTimeout() {
        // openFeign-ribbon, 客户端一般默认等待1s
        return paymentFeignService.paymentFeignTimeout();
    }
}
```

3.1.4. 测试

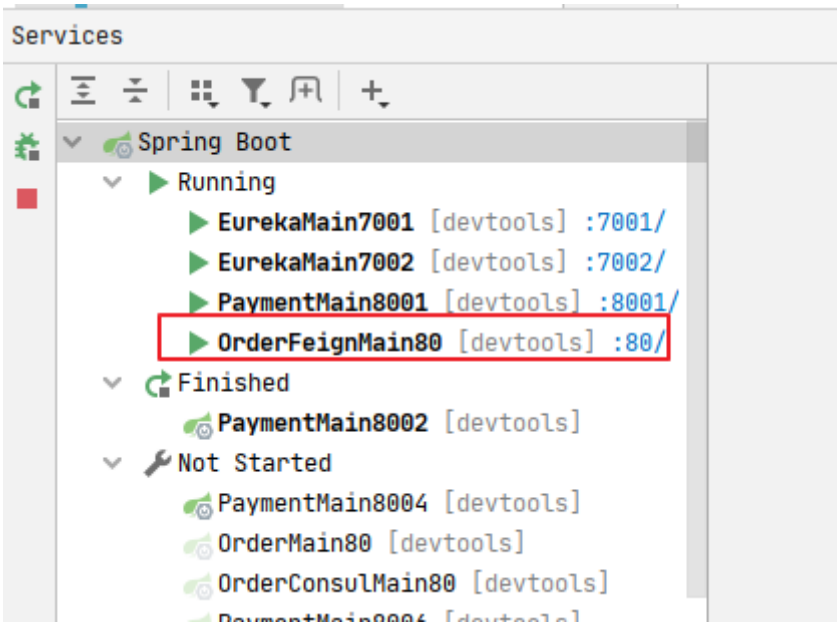
启动eureka集群7001/7002



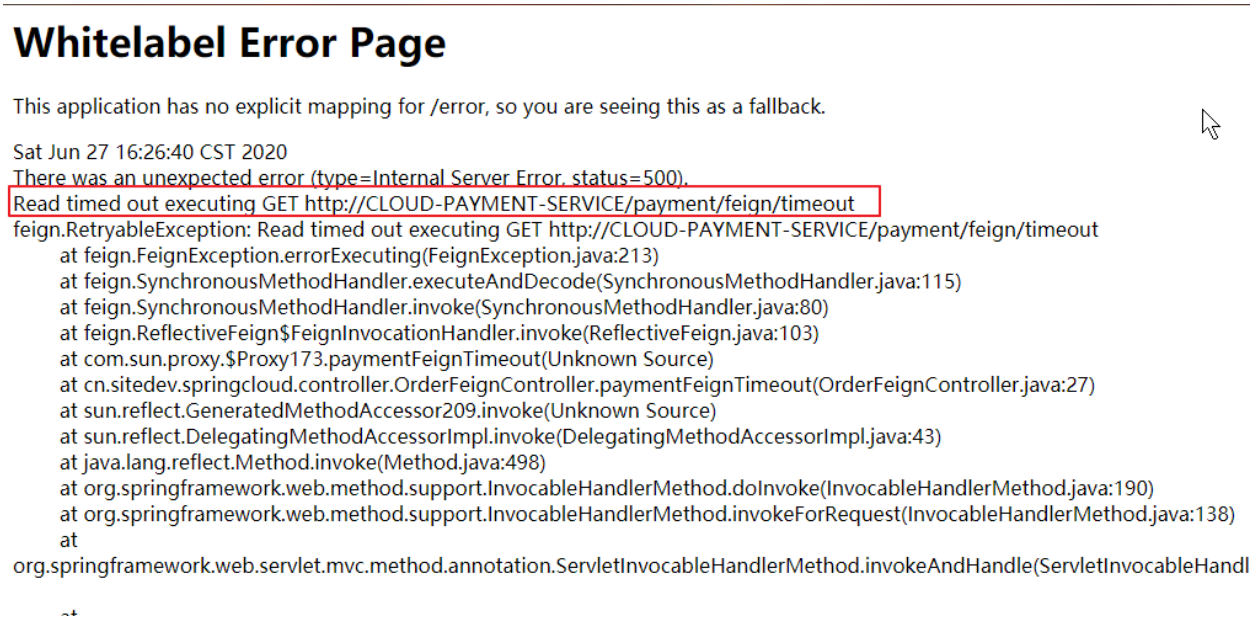
启动支付服务8001



启动订单服务80



浏览器访问 http://localhost/consumer/payment/feign/timeout



3.2. 现象原因: OpenFeign 默认等待1s, 超过后报错

3.3. 是什么

默认Feign客户端只等待一秒钟，但是服务端处理需要超过1秒钟，导致 Feign客户端不想等待了，直接返回报错。

为了避免这样的情况，有时候我们需要设置Feign客户端的超时控制。

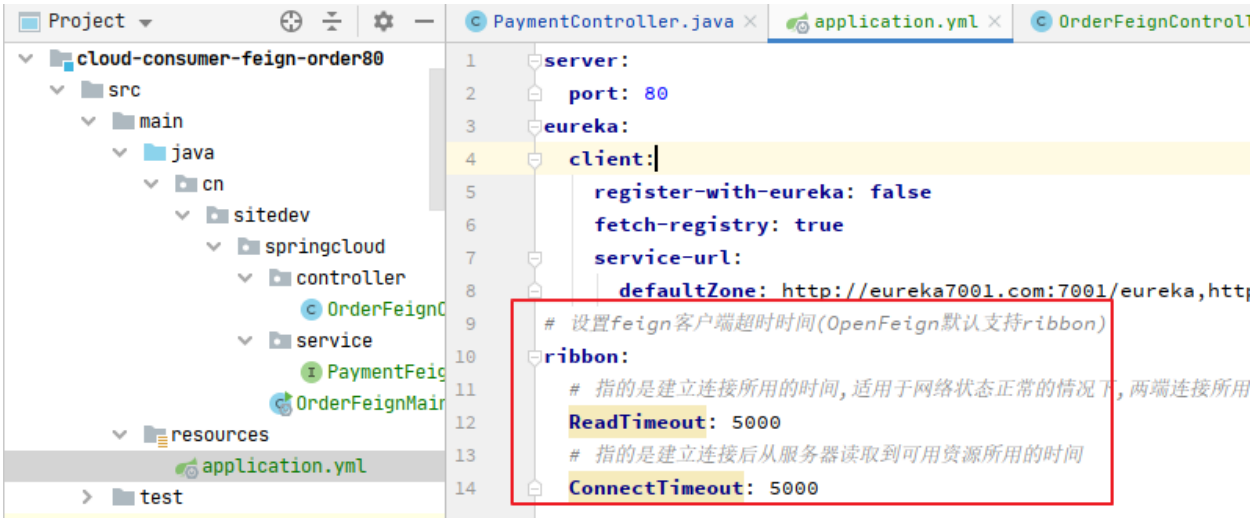
需要在YML文件中开启配置

OpenFeign默认支持Ribbon

3.4. YML配置超时时间

消费端YML文件里需要开启OpenFeign客户端超时控制

```yaml
server:
  port: 80
eureka:
  client:
    register-with-eureka: false
    fetch-registry: true
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka,http://eureka7002.com:7002/eure
# 设置feign客户端超时时间(OpenFeign默认支持ribbon)
ribbon:
  # 指的是建立连接所用的时间,适用于网络状态正常的情况下,两端连接所用的时间
  ReadTimeout: 5000
  # 指的是建立连接后从服务器读取到可用资源所用的时间
  ConnectTimeout: 5000
```



修改重启后,浏览器访问 http://localhost/consumer/payment/feign/timeout

4. OpenFeign日志打印功能

4.1. 是什么

Feign提供了日志打印功能，我们可以通过配置来调整日志级别，从而了解 Feign中Http请求的细节

说白了就是对 Feign接口的调用情况进行监控和输出

4.2. 日志级别

NONE：默认的，不显示任何日志；

BASIC：仅记录请求方法、URL、响应状态码及执行时间；

HEADERS：除了BASIC中定义的信息之外，还有请求和响应的头信息；

FULL：除了 HEADERS中定义的信息之外，还有请求和响应的正文及元数据。

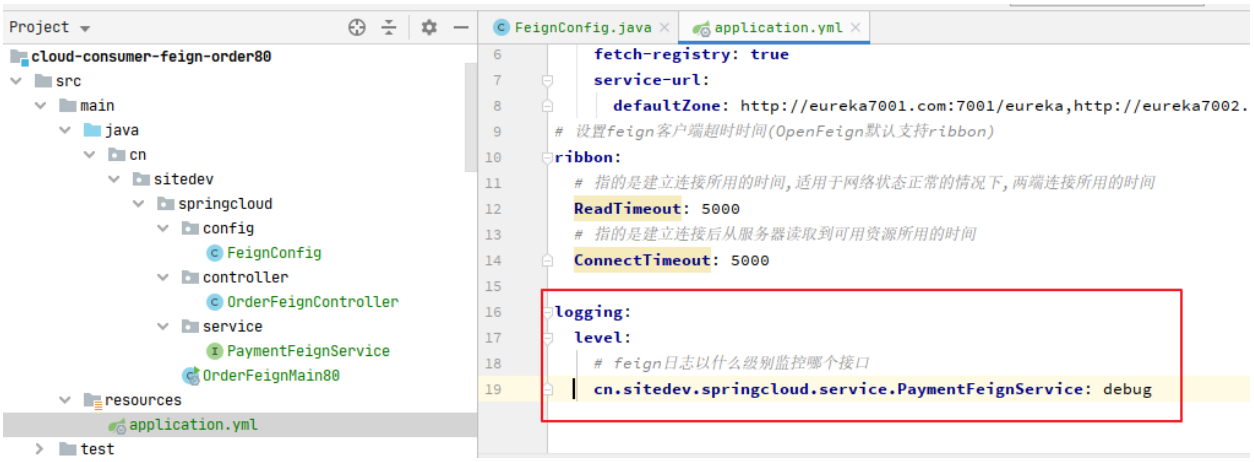
4.3. 配置日志Bean

```
package cn.sitedev.springcloud.config;

import feign.Logger;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * OpenFeignClient配置
 **/
@Configuration
public class FeignConfig {

    /**
     * feignClient配置日志级别
     *
     * @return
     */
    @Bean
    public Logger.Level feignLoggerLevel() {
        // 请求和响应的头信息,请求和响应的正文及元数据
        return Logger.Level.FULL;
    }
}
```

4.4. YML开启日志的Feign客户端

```
server:
  port: 80
eureka:
```

```yaml
  client:
    register-with-eureka: false
    fetch-registry: true
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka,http://eureka7002.com:7002/eure
# 设置feign客户端超时时间(OpenFeign默认支持ribbon)
ribbon:
  # 指的是建立连接所用的时间,适用于网络状态正常的情况下,两端连接所用的时间
  ReadTimeout: 5000
  # 指的是建立连接后从服务器读取到可用资源所用的时间
  ConnectTimeout: 5000

logging:
  level:
    # feign日志以什么级别监控哪个接口
    cn.sitedev.springcloud.service.PaymentFeignService: debug
```



4.5. 后台日志查看

浏览器访问 http://localhost/consumer/payment/get/1

查看控制台输出


Spring Boot
Running
EurekaMain7001 [devtools] :7001/
EurekaMain7002 [devtools] :7002/
PaymentMain8001 [devtools] :8001/
OrderFeignMain80 [devtools] :80/
Finished
PaymentMain8002 [devtools]
Not Started
PaymentMain8004 [devtools]
OrderMain80 [devtools]
OrderConsulMain80 [devtools]
Console
Endpoints
e#getPaymentById] ---> GET http://CLOUD-PAYMENT-SERVICE/payment/get/1 HTTP/1.1
e#getPaymentById] ---> END HTTP (0-byte body)
e#getPaymentById] <--- HTTP/1.1 200 (8ms)
e#getPaymentById] connection: keep-alive
e#getPaymentById] content-type: application/json
e#getPaymentById] date: Sat, 27 Jun 2020 08:39:12 GMT
e#getPaymentById] keep-alive: timeout=60
e#getPaymentById] transfer-encoding: chunked
e#getPaymentById]
e#getPaymentById] {"code":200,"message":"查询成功, serverPort: 8001","data":{"id":1,"serial":"测试支付模块接口"}}
e#getPaymentById] <--- END HTTP (107-byte body)