

## 1. 概述

1.1. 为什么会出现这个技术？需要解决哪些问题？

1.2. 是什么

1.3. 解决

## 2. 搭建链路监控步骤

### 2.1. zipkin

2.1.1. 下载/安装

2.1.2. 使用

2.1.3. 运行控制台

2.1.4. 术语

2.1.4.1. 完整的调用链路

2.1.4.2. 上图是什么

2.1.4.3. 名词解释

### 2.2. 服务提供者

2.2.1. 修改Module

2.2.2. POM

2.2.3. YML

2.2.4. 业务类

2.2.4.1. Controller

### 2.3. 服务消费者（调用方）

2.3.1. 修改Module

2.3.2. POM

2.3.3. YML

2.3.4. 业务类

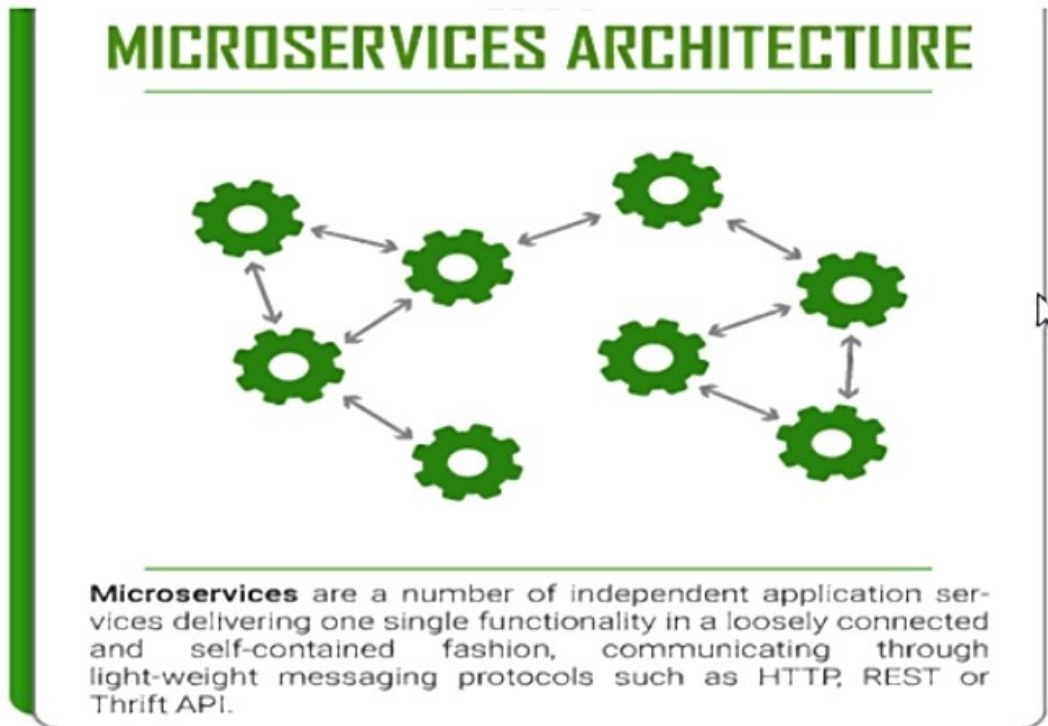
2.3.4.1. Controller

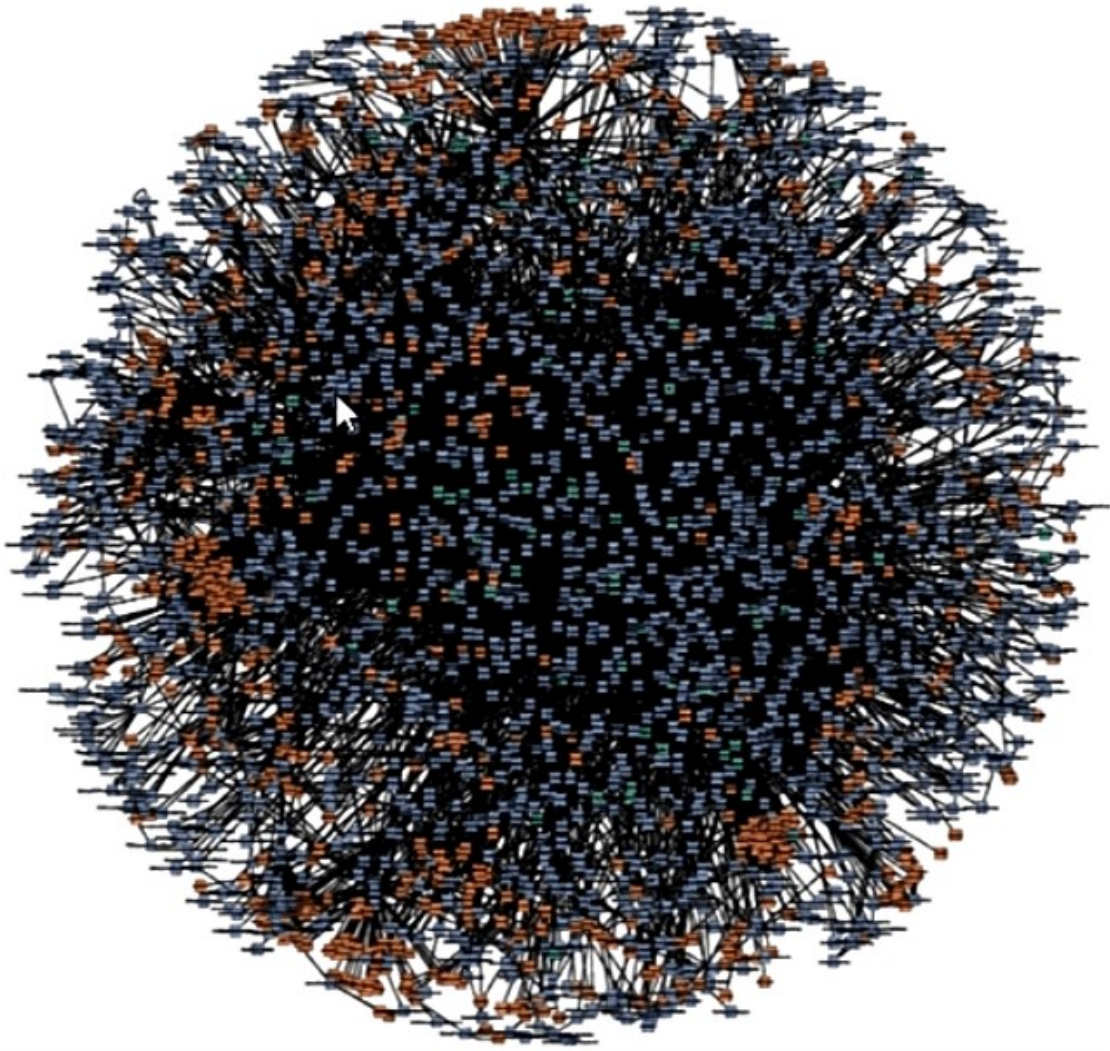
### 2.4. 测试

# 1. 概述

## 1.1. 为什么会出现这个技术？ 需要解决哪些问题？

在微服务框架中，一个由客户端发起的请求在后端系统中会经过多个不同的的服务节点调用来协同产生最后的请求结果，每个前段请求都会形成一条复杂的分布式服务调用链路，链路中的任何一环出现高延时或错误都会引起整个请求最后的失败。





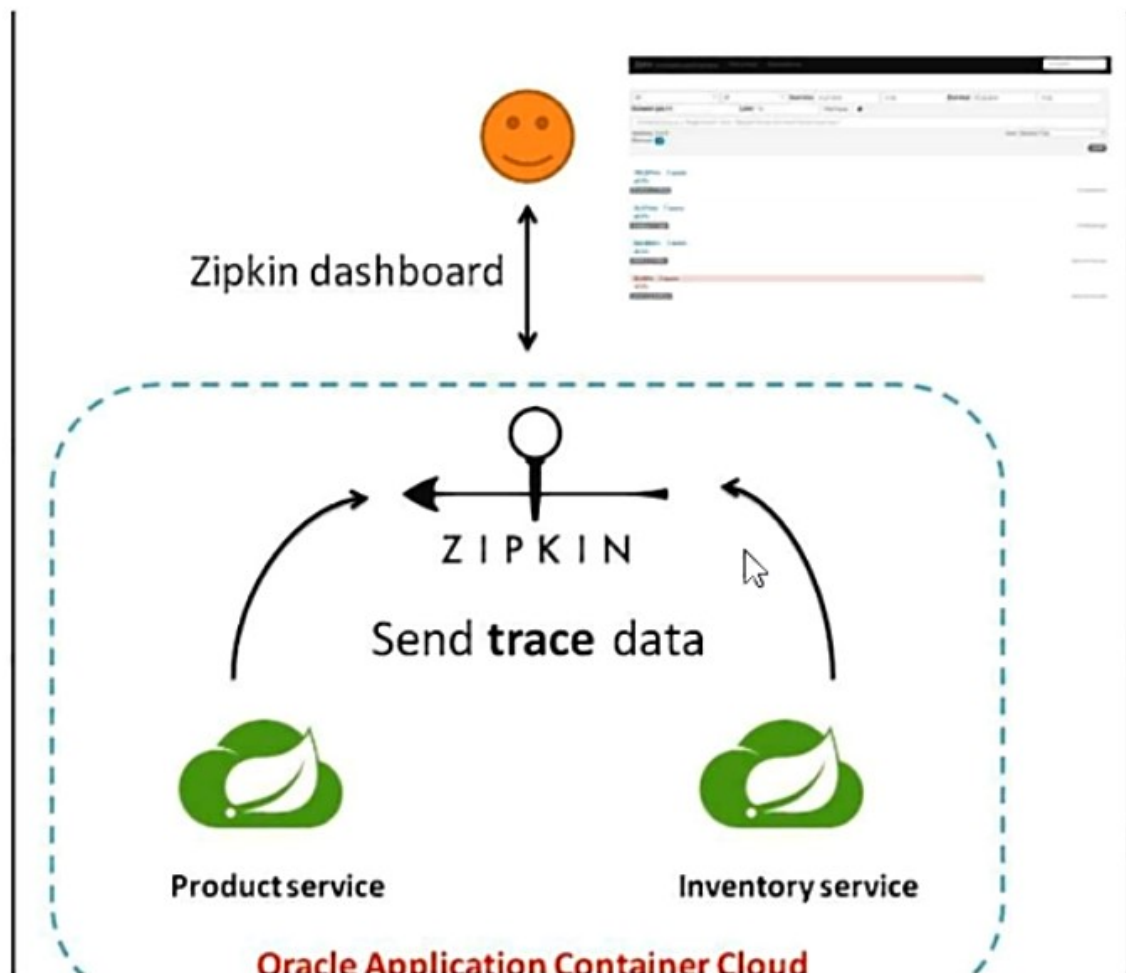
## 1.2. 是什么

官网: <https://github.com/spring-cloud/spring-cloud-sleuth>

Spring Cloud Sleuth提供了一套完整的服务跟踪的解决方案

在分布式系统中提供追踪解决方案并且兼容支持了zipkin

## 1.3. 解决



## 2. 搭建链路监控步骤

### 2.1. zipkin

#### 2.1.1. 下载/安装

SpringCloud从F版起已不需要自己构建Zipkin server了，只需要调用jar包即可

下载地址: <https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/>

这里使用的版本为: zipkin-server-2.12.9.exec.jar

此电脑 > 软件 (D:) > DevSoftWare > zipkin		
	名称	修改日期
	zipkin-server-2.12.9-exec.jar	2020/3/24 23:

#### 2.1.2. 使用

命令行执行:

```
1 D:\DevSoftWare\zipkin>java -jar D:\DevSoftWare\zipkin\zipkin-server-2.12.9-exec.jar
```

[illegible]

### 2.1.3. 运行控制台

浏览器访问<http://localhost:9411/zipkin/>

Zipkin

查找

已保存

依赖

Try Lens UI

请输入traceid

搜索

服务名

Span名称

时间

all

▼

Span Name

▼

1 小时

▼

根据Annotation查询

持续时间 (μs) >=

数量

排序

For example: http.path=/foo/bar/ and cluster=foo and cache.miss

Ex: 100ms or 5s

10

耗时降序

▼

查找

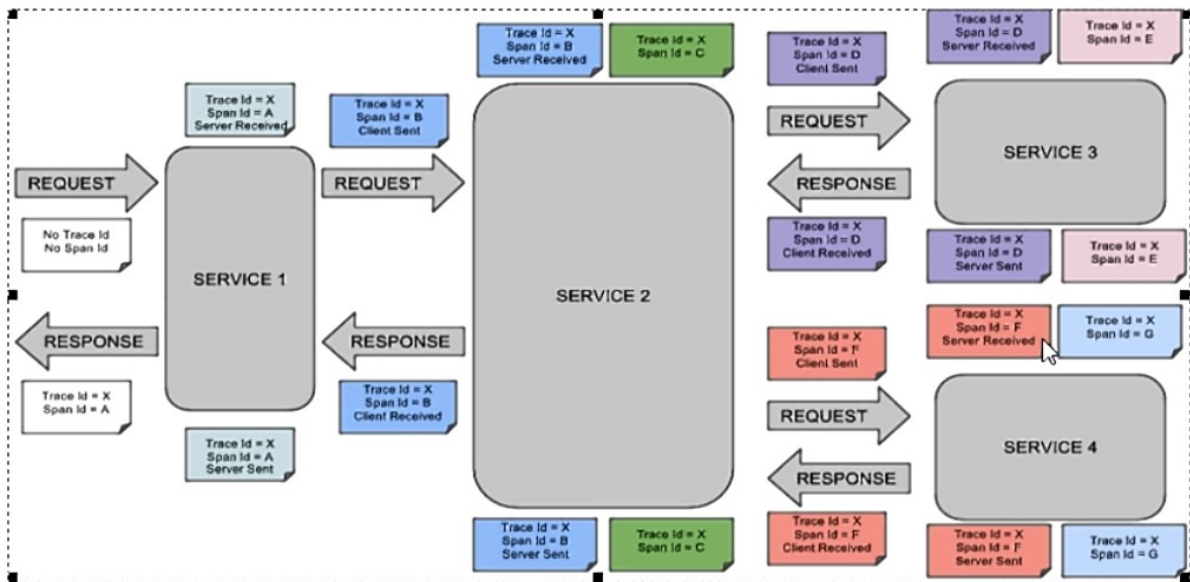
?

请选择查找的条件。

### 2.1.4. 术语

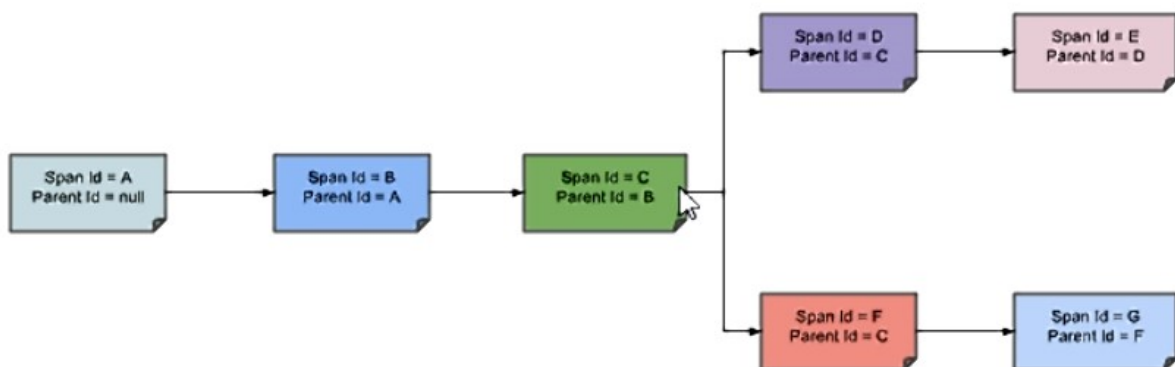
### 2.1.4.1. 完整的调用链路

表示请求链路，一条链路通过 Trace Id唯标识， Span标识发起的请求信息，各span通过 parent id关联起来

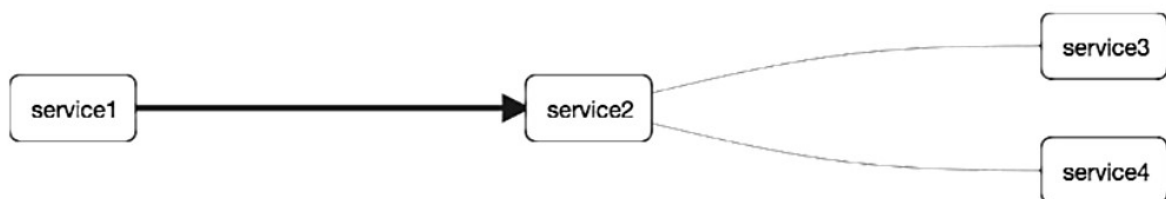


### 2.1.4.2. 上图是什么

一条链路通过 Trace Id 唯标识，Span 标识发起的请求信息，各span通过 parent id 关联起来



整个链路的依赖关系如下：



### 2.1.4.3. 名词解释

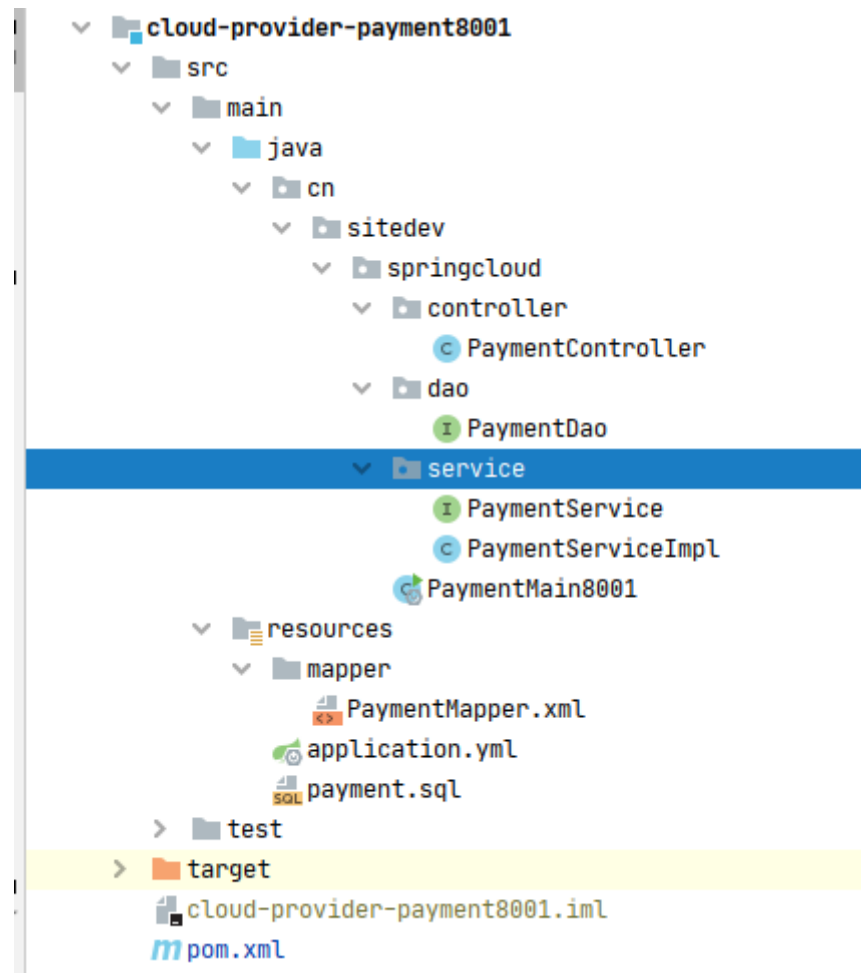
Trace:类似于树结构的Span集合，表示一条调用链路，存在唯一标识

span:表示调用链路来源，通俗的理解span就是一次请求信息

## 2.2. 服务提供者

### 2.2.1. 修改Module

修改cloud-provider-payment8001



## 2.2.2. POM

修改内容:

```
1      <!--包含了sleuth+zipkin-->
2      <dependency>
3          <groupId>org.springframework.cloud</groupId>
4          <artifactId>spring-cloud-starter-zipkin</artifactId>
5      </dependency>
```

完整内容:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
5     <parent>
6         <artifactId>cloud2020</artifactId>
7         <groupId>cn.sitedev.springcloud</groupId>
```



```
8      <version>1.0-SNAPSHOT</version>
9  </parent>
10 <modelVersion>4.0.0</modelVersion>
11
12 <artifactId>cloud-provider-payment8001</artifactId>
13
14 <dependencies>
15     <!--包含了sleuth+zipkin-->
16     <dependency>
17         <groupId>org.springframework.cloud</groupId>
18         <artifactId>spring-cloud-starter-zipkin</artifactId>
19     </dependency>
20     <!--eureka-client-->
21     <dependency>
22         <groupId>org.springframework.cloud</groupId>
23         <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
24     </dependency>
25     <!--引入自己定义的api通用包，可以使用Payment支付Entity-->
26     <dependency>
27         <groupId>cn.sitedev.springcloud</groupId>
28         <artifactId>cloud-api-commons</artifactId>
29         <version>${project.version}</version>
30     </dependency>
31     <dependency>
32         <groupId>org.springframework.boot</groupId>
33         <artifactId>spring-boot-starter-web</artifactId>
34     </dependency>
35
36     <dependency>
37         <groupId>org.springframework.boot</groupId>
38         <artifactId>spring-boot-starter-actuator</artifactId>
39     </dependency>
40
41     <dependency>
42         <groupId>org.mybatis.spring.boot</groupId>
43         <artifactId>mybatis-spring-boot-starter</artifactId>
44     </dependency>
45
46     <dependency>
47         <groupId>com.alibaba</groupId>
48         <artifactId>druid-spring-boot-starter</artifactId>
49         <version>1.1.10</version>
50     </dependency>
```



```

51     <dependency>
52         <groupId>mysql</groupId>
53         <artifactId>mysql-connector-java</artifactId>
54     </dependency>
55
56     <dependency>
57         <groupId>org.springframework.boot</groupId>
58         <artifactId>spring-boot-starter-jdbc</artifactId>
59     </dependency>
60
61     <dependency>
62         <groupId>org.springframework.boot</groupId>
63         <artifactId>spring-boot-devtools</artifactId>
64         <scope>runtime</scope>
65         <optional>true</optional>
66     </dependency>
67     <dependency>
68         <groupId>org.projectlombok</groupId>
69         <artifactId>lombok</artifactId>
70         <optional>true</optional>
71     </dependency>
72     <dependency>
73         <groupId>org.springframework.boot</groupId>
74         <artifactId>spring-boot-starter-test</artifactId>
75         <scope>test</scope>
76     </dependency>
77
78 </dependencies>
79
80 </project>

```

### 2.2.3. YML

修改内容:

```

1 spring:
2   zipkin:
3     base-url: http://localhost:9411
4   sleuth:
5     sampler:
6       probability: 1 # 采样率值介于0到1之间, 1表示全部采集

```

完整内容:

```
1 server:
2   port: 8001
3
4 spring:
5   application:
6     name: cloud-payment-service
7   datasource:
8     type: com.alibaba.druid.pool.DruidDataSource # 当前数据源操作类型
9     driver-class-name: org.gjt.mm.mysql.Driver # mysql驱动包
10    url: jdbc:mysql://localhost:3306/db2019?useUnicode=true&characterEncoding=utf-8
11    username: root
12    password: root
13  zipkin:
14    base-url: http://localhost:9411
15  sleuth:
16    sampler:
17      probability: 1 # 采样率值介于0到1之间, 1表示全部采集
18
19 mybatis:
20   mapperLocations: classpath:mapper/*.xml
21   type-aliases-package: cn.sitedev.springcloud.entities # 所有Entity别名类所在包
22
23 eureka:
24   client:
25     # 表示是否将自己注册进EurekaServer, 默认为true
26     register-with-eureka: true
27     # 是否从EurekaServer抓取已有的注册信息, 默认为true
28     # 单节点无所谓, 集群必须设置为true才能配合ribbon使用负载均衡
29     fetch-registry: true
30     service-url:
31       defaultZone: http://eureka7001.com:7001/eureka,http://eureka7002.com:7002/eureka
32   instance:
33     instance-id: payment8001
34     prefer-ip-address: true # 访问路径可以显示IP地址
35     # Eureka客户端向服务端发送心跳的时间间隔, 单位:s.默认30s
36     lease-renewal-interval-in-seconds: 1
37     # Eureka服务端在收到最后一次心跳后等待时间上限.单位:s.默认90s.超时剔除服务
38     lease-expiration-duration-in-seconds: 2
```

## 2.2.4. 业务类

### 2.2.4.1. Controller

```
1 package cn.sitedev.springcloud.controller;
2
3 import cn.sitedev.springcloud.entities.CommonResult;
4 import cn.sitedev.springcloud.entities.Payment;
5 import cn.sitedev.springcloud.service.PaymentService;
6 import lombok.extern.slf4j.Slf4j;
7 import org.springframework.beans.factory.annotation.Value;
8 import org.springframework.cloud.client.ServiceInstance;
9 import org.springframework.cloud.client.discovery.DiscoveryClient;
10 import org.springframework.web.bind.annotation.*;
11
12 import javax.annotation.Resource;
13 import javax.servlet.http.HttpServletRequest;
14 import java.util.List;
15 import java.util.concurrent.TimeUnit;
16
17 @RestController
18 @Slf4j
19 @RequestMapping("/payment")
20 public class PaymentController {
21     @Resource
22     private PaymentService paymentService;
23
24     /**
25      * 端口号
26      * 查看负载均衡效果
27      */
28     @Value("${server.port}")
29     private String serverPort;
30
31     /**
32      * 服务发现，获取服务信息
33      */
34     @Resource
35     private DiscoveryClient discoveryClient;
36
37     @PostMapping(value = "/create")
38     public CommonResult create(@RequestBody Payment payment) {
```

```

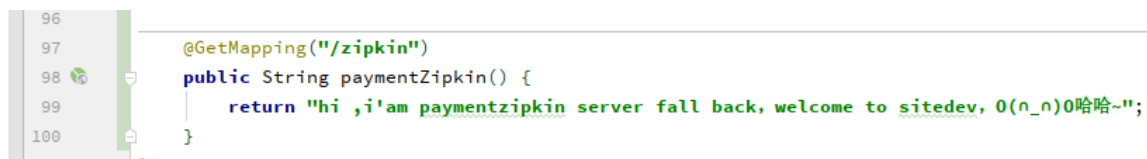
39     int result = paymentService.create(payment);
40     log.info("*****插入结果: " + result);
41     if (result > 0) {
42         return new CommonResult(200, "插入数据库成功, serverPort: " + serverPort);
43     } else {
44         return new CommonResult(444, "插入数据库失败", null);
45     }
46 }
47
48 @GetMapping(value = "/get/{id}")
49 public CommonResult getPaymentById(@PathVariable("id") Long id) {
50     Payment payment = paymentService.getPaymentById(id);
51     log.info("*****查询结果: " + payment);
52     if (payment != null) {
53         return new CommonResult(200, "查询成功, serverPort: " + serverPort, payment);
54     } else {
55         return new CommonResult(444, "没有对应记录, 查询id: " + id, null);
56     }
57 }
58
59 /**
60  * 服务发现
61  *
62  * @return
63  */
64 @GetMapping("/discovery")
65 public Object discovery() {
66     List<String> services = discoveryClient.getServices();
67     for (String element : services) {
68         log.info("*****element: " + element);
69     }
70     // 一个微服务下的全部实例
71     List<ServiceInstance> instances = discoveryClient.getInstances("CLOUD-PAYMENT-SERVICE");
72     for (ServiceInstance instance : instances) {
73         log.info(instance.getServiceId() + "\t" + instance.getHost() + "\t" + instance.getPort() + "\t" + instance.getInstanceId());
74     }
75     return discoveryClient;
76 }
77
78 @GetMapping(value = "/lb")
79 public String getPaymentLB(HttpServletRequest request) {
80     String header = request.getHeader("MyHeader");
81     System.out.println("来自gateway9527的header: " + header);

```

```

82     String parameter = request.getParameter("MyParameter");
83     System.out.println("来自gateway9527parameter: " + parameter);
84     return serverPort;
85 }
86
87 @GetMapping(value = "/feign/timeout")
88 public String paymentFeignTimeout() {
89     try {
90         TimeUnit.SECONDS.sleep(3);
91     } catch (InterruptedException e) {
92         e.printStackTrace();
93     }
94     return serverPort;
95 }
96
97 @GetMapping("/zipkin")
98 public String paymentZipkin() {
99     return "hi ,i'am paymentzipkin server fall back, welcome to sitedev, 0(n_n)0哈哈~";
100 }
101 }

```



```

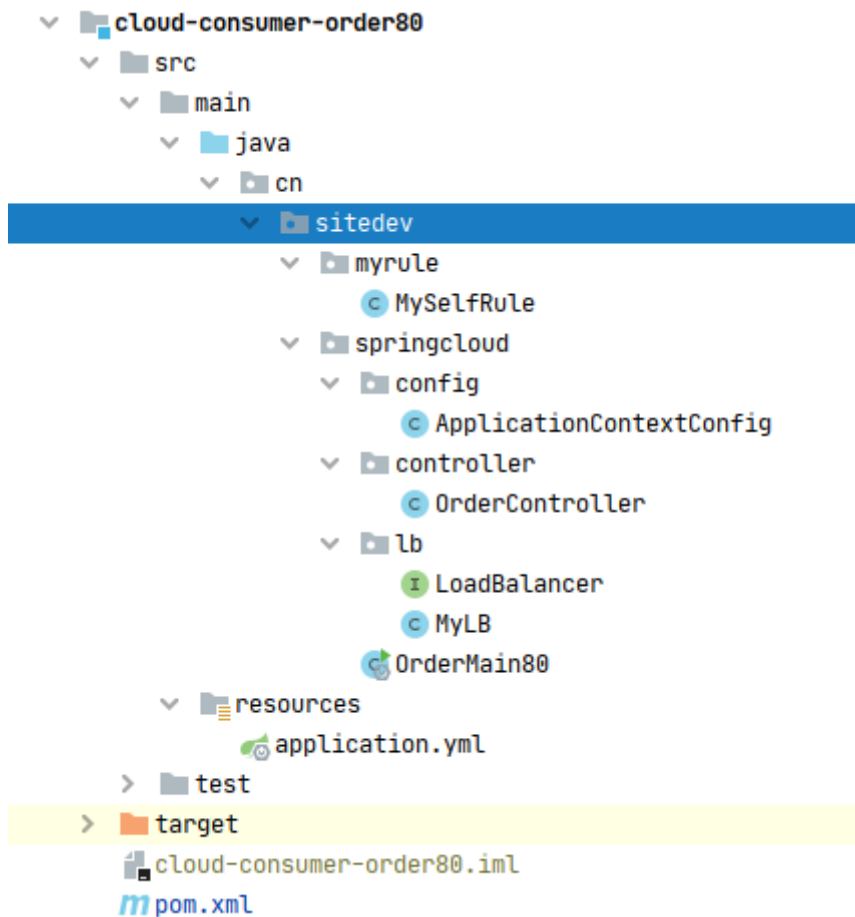
96
97 @GetMapping("/zipkin")
98 public String paymentZipkin() {
99     return "hi ,i'am paymentzipkin server fall back, welcome to sitedev, 0(n_n)0哈哈~";
100 }

```

## 2.3. 服务消费者（调用方）

### 2.3.1. 修改Module

修改cloud-consumer-order80



## 2.3.2. POM

修改内容:

```
1      <!--包含了sleuth+zipkin-->
2      <dependency>
3          <groupId>org.springframework.cloud</groupId>
4          <artifactId>spring-cloud-starter-zipkin</artifactId>
5      </dependency>
```

完整内容:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
5     <parent>
6         <artifactId>cloud2020</artifactId>
7         <groupId>cn.sitedev.springcloud</groupId>
8         <version>1.0-SNAPSHOT</version>
9     </parent>
```

```
10 <modelVersion>4.0.0</modelVersion>
11
12 <artifactId>cloud-consumer-order80</artifactId>
13
14 <dependencies>
15     <!--包含了sleuth+zipkin-->
16     <dependency>
17         <groupId>org.springframework.cloud</groupId>
18         <artifactId>spring-cloud-starter-zipkin</artifactId>
19     </dependency>
20     <!--eureka-client-->
21     <dependency>
22         <groupId>org.springframework.cloud</groupId>
23         <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
24     </dependency>
25     <!--引入自己定义的api通用包，可以使用Payment支付Entity-->
26     <dependency>
27         <groupId>cn.sitedev.springcloud</groupId>
28         <artifactId>cloud-api-commons</artifactId>
29         <version>${project.version}</version>
30     </dependency>
31     <dependency>
32         <groupId>org.springframework.boot</groupId>
33         <artifactId>spring-boot-starter-web</artifactId>
34     </dependency>
35
36     <dependency>
37         <groupId>org.springframework.boot</groupId>
38         <artifactId>spring-boot-starter-actuator</artifactId>
39     </dependency>
40
41     <dependency>
42         <groupId>org.springframework.boot</groupId>
43         <artifactId>spring-boot-devtools</artifactId>
44         <scope>runtime</scope>
45         <optional>true</optional>
46     </dependency>
47     <dependency>
48         <groupId>org.projectlombok</groupId>
49         <artifactId>lombok</artifactId>
50         <optional>true</optional>
51     </dependency>
52     <dependency>
```



```

53         <groupId>org.springframework.boot</groupId>
54         <artifactId>spring-boot-starter-test</artifactId>
55         <scope>test</scope>
56     </dependency>
57
58 </dependencies>
59 </project>

```

### 2.3.3. YML

修改内容:

```

1 spring:
2   zipkin:
3     base-url: http://localhost:9411
4   sleuth:
5     sampler:
6       probability: 1 # 采样率

```

完整内容:

```

1 server:
2   port: 80
3 spring:
4   application:
5     name: cloud-order-service
6   zipkin:
7     base-url: http://localhost:9411
8   sleuth:
9     sampler:
10      probability: 1 # 采样率
11 eureka:
12   client:
13     # 表示是否将自己注册进EurekaServer, 默认为true
14     register-with-eureka: true
15     # 是否从EurekaServer抓取已有的注册信息, 默认为true
16     # 单节点无所谓, 集群必须设置为true才能配合ribbon使用负载均衡
17     fetch-registry: true
18     service-url:
19       defaultZone: http://eureka7001.com:7001/eureka,http://eureka7002.com:7002/eureka

```

## 2.3.4. 业务类

### 2.3.4.1. Controller

```
1 package cn.sitedev.springcloud.controller;
2
3 import cn.sitedev.springcloud.entities.CommonResult;
4 import cn.sitedev.springcloud.entities.Payment;
5 import cn.sitedev.springcloud.lb.LoadBalancer;
6 import lombok.extern.slf4j.Slf4j;
7 import org.springframework.cloud.client.ServiceInstance;
8 import org.springframework.cloud.client.discovery.DiscoveryClient;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.*;
11 import org.springframework.web.client.RestTemplate;
12
13 import javax.annotation.Resource;
14 import java.net.URI;
15 import java.util.List;
16
17 @RestController
18 @RequestMapping("/consumer/payment")
19 @Slf4j
20 public class OrderController {
21
22     // public static final String PAYMENT_URL = "http://localhost:8001";
23
24     // 通过在 eureka上注册过的微服务名称调用
25     public static final String PAYMENT_URL = "http://CLOUD-PAYMENT-SERVICE";
26
27     @Resource
28     private RestTemplate restTemplate;
29
30     @Resource
31     private LoadBalancer loadBalancer;
32
33     @Resource
34     private DiscoveryClient discoveryClient;
35
36     @PostMapping(value = "/create")
37     public CommonResult<Payment> create(Payment payment) {
```

```

38         return restTemplate.postForObject(PAYMENT_URL + "/payment/create", payment,
39     }
40
41     @PostMapping(value = "/createForEntity")
42     public CommonResult<Payment> create2(Payment payment) {
43         return restTemplate.postForEntity(PAYMENT_URL + "/payment/create", payment,
44     }
45
46     @GetMapping(value = "/get/{id}")
47     public CommonResult<Payment> getPayment(@PathVariable("id") Long id) {
48         return restTemplate.getForObject(PAYMENT_URL + "/payment/get/" + id, Common
49     }
50
51     @GetMapping(value = "/getForEntity/{id}")
52     public CommonResult<Payment> getPayment2(@PathVariable("id") Long id) {
53         ResponseEntity<CommonResult> entity = restTemplate.getForEntity(PAYMENT_URL
54             CommonResult.class);
55         if (entity.getStatusCode().is2xxSuccessful()) {
56             return entity.getBody();
57         } else {
58             return new CommonResult<>(444, "操作失败");
59         }
60     }
61
62     @GetMapping(value = "/lb")
63     public String getPaymentLB() {
64         List<ServiceInstance> instances = discoveryClient.getInstances("CLOUD-PAYMEN
65         if (instances == null || instances.size() <= 0) {
66             return null;
67         }
68         ServiceInstance serviceInstance = loadBalancer.instances(instances);
69         URI uri = serviceInstance.getUri();
70         return restTemplate.getForObject(uri + "/payment/lb", String.class);
71     }
72
73     // =====> zipkin+sleuth
74     @GetMapping("/zipkin")
75     public String paymentZipkin() {
76         String result = restTemplate.getForObject("http://localhost:8001" + "/paymer
77         return result;
78     }
79 }

```

```

73 // =====> zipkin+sleuth
74 @GetMapping("/zipkin")
75 public String paymentZipkin() {
76     String result = restTemplate.getForObject( url: "http://localhost:8001" + "/payment/zipkin/",
77     String.class);
78     return result;
79 }

```

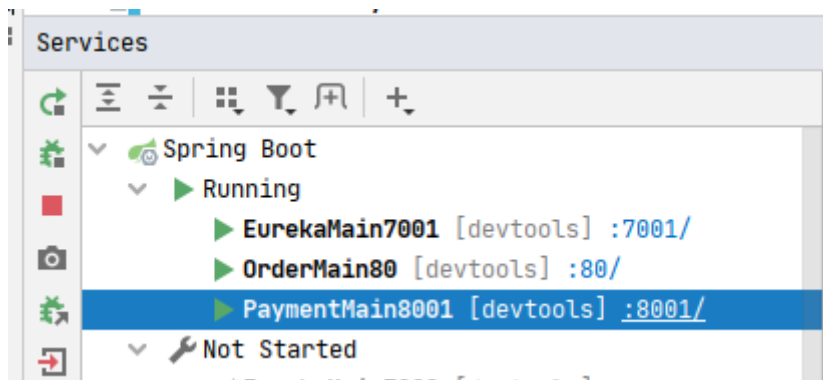
## 2.4. 测试

1) 依次启动以下服务

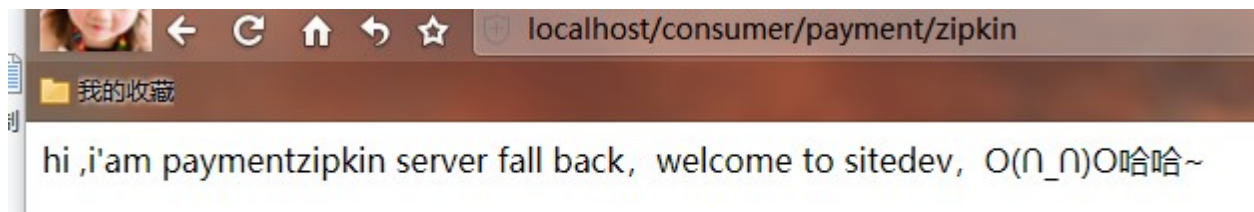
启动EurekaMain7001

启动OrderMain80

启动PaymentMain8001



2) 80调用8001(浏览器访问<http://localhost/consumer/payment/zipkin>)几次测试下

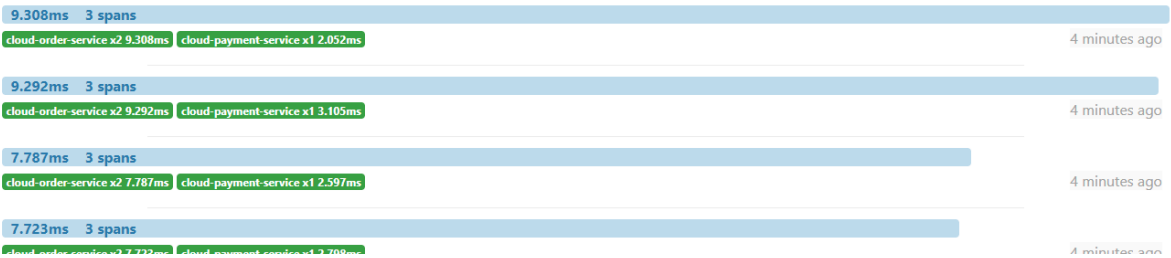


3) 打开浏览器访问 <http://localhost:9411> , 会出现以下界面



4) 查看依赖关系

服务名	Span名称	时间
all	all	1 小时
根据Annotation查询		持续时间 (μs) >=
For example: http.path=/foo/bar/ and cluster=foo and cache.miss		Ex: 100ms or 5s
		数量
		10
		排序
		耗时降序
<a href="#">查找</a> <a href="#">?</a>		
Showing: 10 of 10		
Services: <a href="#">all</a>		
<a href="#">JSON</a> <a href="#">📄</a>		



持续时间: 9.308ms	服务: 2	深度: 2	Span总数: 2	<a href="#">JSON</a> <a href="#">📄</a>
<a href="#">全部展开</a> <a href="#">全部折叠</a>				
cloud-order-service x2 cloud-payment-service x1				

