

Day1 微服务架构知识介绍

1 打卡任务

作业：

1. 开通ServiceStage与CSE服务，并加入21天转型微服务实战营课堂。

打卡：

1. 将成功开通服务及加入课堂的截图上传完成打卡。

2 准备工作

- 1、已经拥有华为云账户

3 开通方法

- 1、访问<https://www.huaweicloud.com/product/servicestage.html>点击立即使用



- 2、访问<https://www.huaweicloud.com/product/cse.html>点击立即使用



3、访问链接点击加入课堂

<https://classroom.devcloud.huaweicloud.com/diploma/joinclass/41d76a6fea454095964d1cf220f94adb>



4 打卡截图

成功开通CSE服务



成功开通ServiceStage服务，提醒：请选择基础版



成功加入课堂



Day2 微服务入门之编写HelloWorld

1 打卡任务

作业：

- 1、在provider服务中增加一个greeting方法，接受POST请求，请求参数为request body中传递的Person对象，包含name和gender两个字段。name为String类型；gender为枚举Gender类型，有MALE/FEMALE两个值。返回值为GreetingResponse类型，包含msg和timestamp两个字段，msg为根据gender不同生成的问候语 Hello, Mr.xxx (MALE) / Hello, Ms.xxx (FEMALE)，timestamp为java.util.Date类型的时间戳。
- 2、在consumer服务中增加一个greeting方法，接收外部请求的触发，以调用provider服务的greeting方法，并返回provider的应答消息。

打卡：

- 1、调用provider服务的greeting方法成功，并截图
- 2、调用consumer服务的greeting方法成功，并截图

打卡任务基于Day2的demo项目：



Demo-Day2.zip

2 准备工作

- 1、CSEJavaSDK需要在Java 1.8环境运行。请确保本地环境安装了Java 1.8开发环境及maven。
- 2、配置maven setting文件，确保本地开发环境能够下载CSEJavaSDK的依赖包。

配置方法参考 https://support.huaweicloud.com/devg-cse/cse_03_summary.html

如果您之前使用的是默认的maven setting文件的话，也可以直接使用这份配置文件



settings.xml

覆盖它

3、获取AK/SK

参考文档 https://support.huaweicloud.com/usermanual-iam/zh-cn_topic_0079477318.html，获取自己的AK/SK。

AK/SK需要配置在microservice.yaml中，用于连接华为云上的CSE服务时的认证。

配置方式参考Day2课程培训资料。

4、本地成功运行 Day2 的provider、consumer demo工程项目。

3 在 provider 服务中开发 greeting 方法

1、创建作为请求的Person类、作为返回值的GreetingResponse类以及Gender枚举类型

```
C GreetingResponse.java ×
1 package microservice.demo.training21days.provider.service;
2
3 import java.util.Date;
4
5 public class GreetingResponse {
6     private String msg;
7
8     private Date timestamp;
9
10    public String getMsg() { return msg; }
13
14    public void setMsg(String msg) { this.msg = msg; }
17
18    public Date getTimestamp() { return timestamp; }
21
22    public void setTimestamp(Date timestamp) { this.timestamp = timestamp; }
25
26    @Override
27    public String toString() {...}
34 }
```

```

C Person.java x
1 package microservice.demo.training21days.provider.service;
2
3 public class Person {
4     private String name;
5
6     private Gender gender;
7
8     public String getName() { return name; }
11
12     public void setName(String name) { this.name = name; }
15
16     public Gender getGender() { return gender; }
19
20     public void setGender(Gender gender) { this.gender = gender; }
23
24     @Override
25     public String toString() {...}
32 }

E Gender.java x
1 package microservice.demo.training21days.provider.service;
2
3 public enum Gender {
4     MALE,
5     FEMALE
6 }

```

2、在HelloService中创建greeting方法

```

1.. @PostMapping(path = "/greeting")
2.. public GreetingResponse greeting(@RequestBody Person person) {
3..     GreetingResponse greetingResponse = new GreetingResponse();
4..
5..     if (Gender.MALE.equals(person.getGender())) {
6..         greetingResponse.setMsg("Hello, Mr." + person.getName());
7..     } else {
8..         greetingResponse.setMsg("Hello, Ms." + person.getName());
9..     }
10..    greetingResponse.setTimestamp(new Date());
11..    return greetingResponse;
12.. }

```

3、运行provider服务并进行调用

直接启动有可能会因为与服务中心记录的服务契约不一致而报错，此时可以考虑删除服务中心内的provider服务记录，或升级provider服务的版本。

观察服务契约，可以看到greeting方法的请求参数person在body中传递：

```
consumes:
- "application/json"
produces:
- "application/json"
paths:
  .. /greeting:
    .. post:
      .. .. operationId: "greeting"
      .. .. parameters:
      .. .. - in: "body"
      .. ..   name: "person"
      .. ..   required: true
      .. ..   schema:
      .. ..     $ref: "#/definitions/Person"
      .. .. responses:
      .. ..   200:
      .. ..     description: "response of 200"
      .. ..     schema:
      .. ..       $ref: "#/definitions/GreetingResponse"
```

Person的结构如下:

```
.. Person:
  .. type: "object"
  .. properties:
  ..   name:
  ..     type: "string"
  ..   gender:
  ..     type: "string"
  ..     enum:
  ..       - "MALE"
  ..       - "FEMALE"
  ..   x-java-class: "microservice.demo.training21days.provider.service.Gender"
  ..   x-java-class: "microservice.demo.training21days.provider.service.Person"
```

接口接收请求、返回应答的Content-Type都是application/json, 因此, 发送请求时的body中应该拼写一个json串。

4 在 consumer 服务中开发 greeting 方法

- 1、定义Person、GreetingResponse、Gender类, 保持类名、包名、内部属性与provider一致。
- 2、在HelloService代理接口中增加一个greeting方法:

```
public interface HelloService {
  .. String sayHello(String name);

  .. GreetingResponse greeting(Person person);
}
```

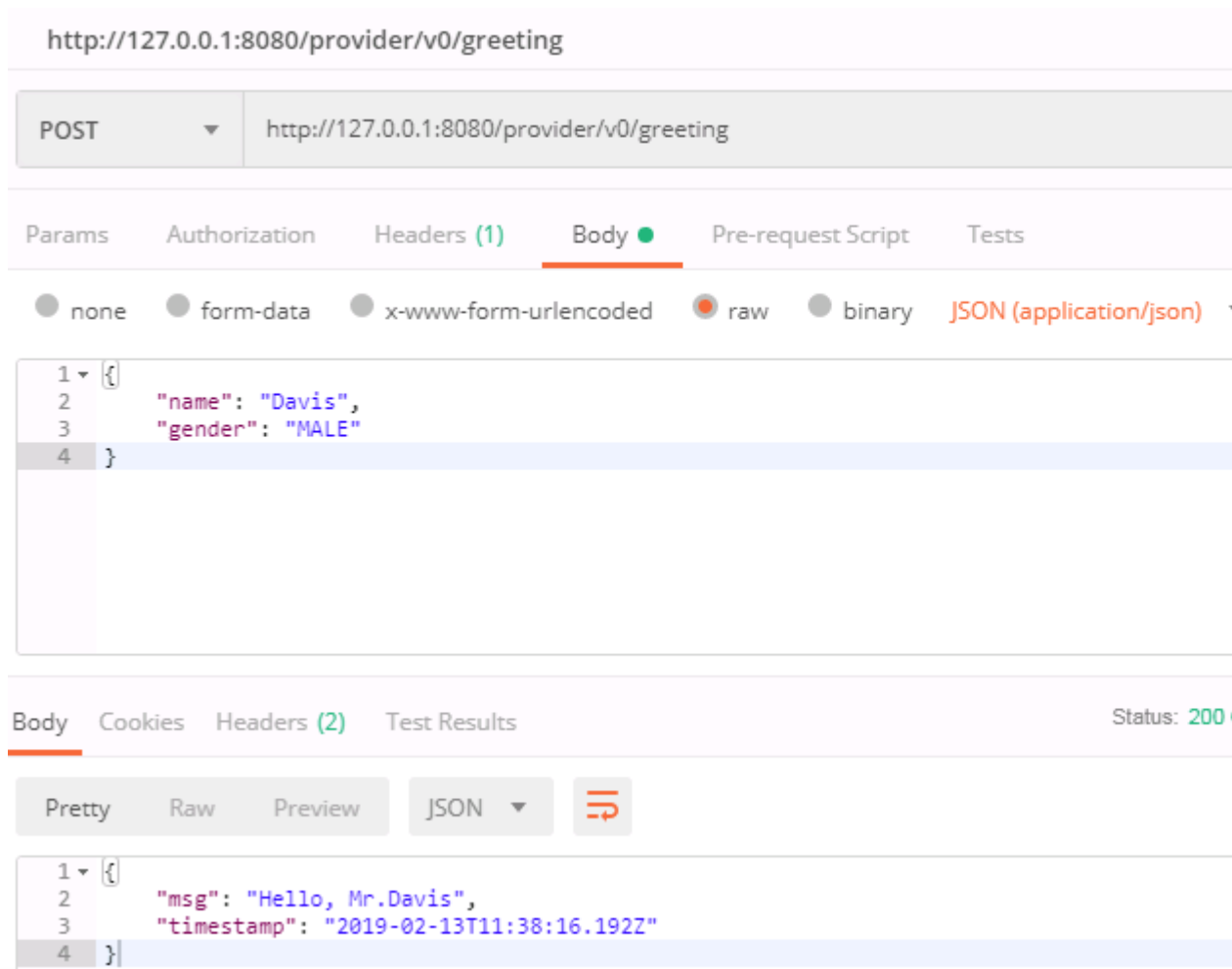
- 3、在HelloConsumerService中增加一个greeting方法，接收请求中的Person参数并通过RPC代理将它发送到provider服务中：

```
..@Path("/greeting")
..@POST
..public GreetingResponse greeting(Person person) {
..    return helloService.greeting(person);
..}
```

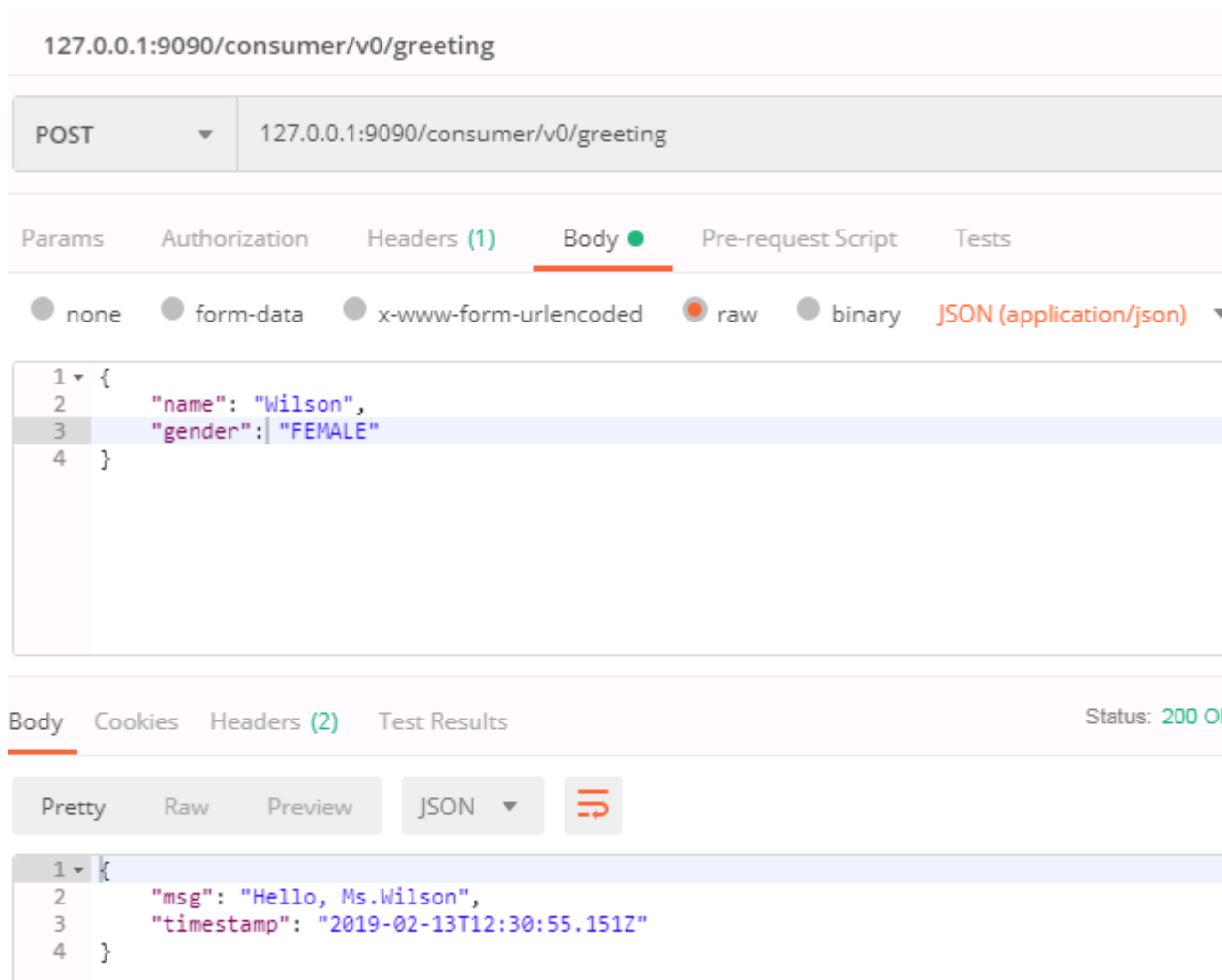
- 4、启动consumer服务，调用它的greeting接口，可以看到正常返回结果。

5 打卡截图

调用provider服务的结果如下：



调用consumer服务的结果如下：



6 总结

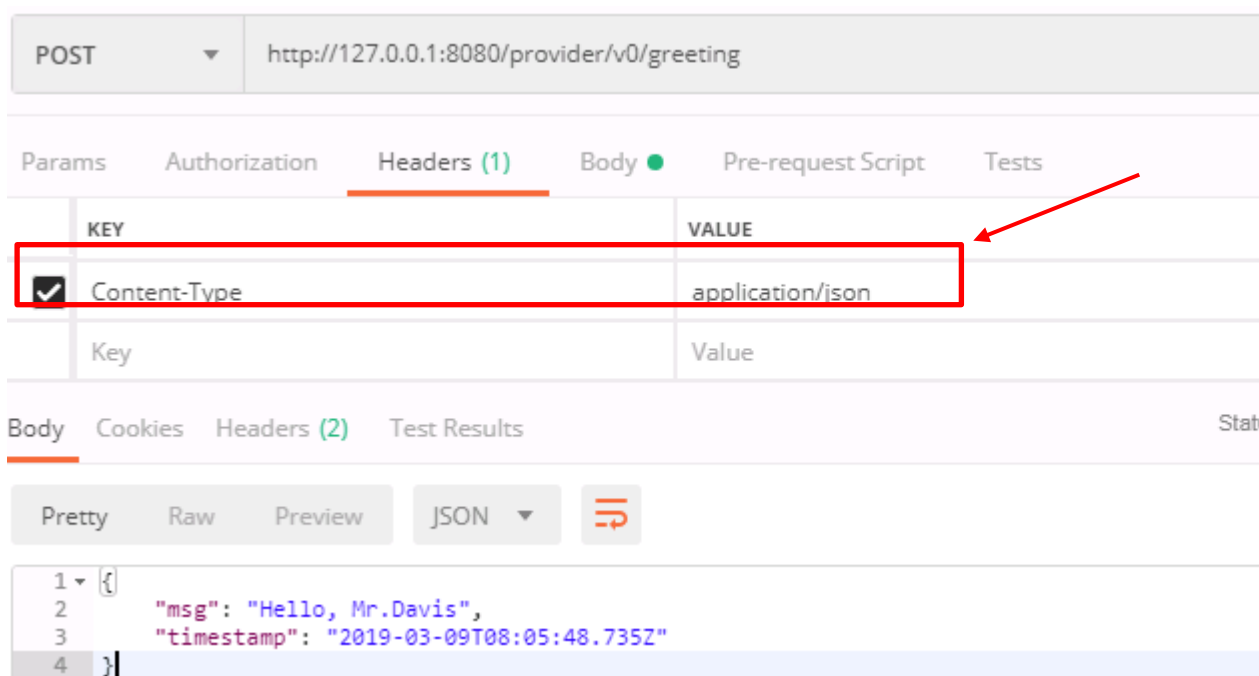
- 1、Java对象类型的参数也会在服务契约中明确描述出来，CSEJavaSDK会根据契约对参数进行序列化和反序列化，用户在业务代码中直接操作Java对象即可，不需要再考虑数据流的序列化问题。因此，参数类型必须是明确的，可以被服务契约声明出来的。例如，Object类型、抽象类型、没有指定具体属性类型的泛型都是不能作为参数的。参考https://docs.servicecomb.io/java-chassis/zh_CN/build-provider/interface-constraints.html。
- 2、consumer端和provider端使用的参数类型不一定要来自于同一个jar包中，但两端的参数的类名和包名最好保持完全一致（如同本次打卡作业）。事实上只要参数类型的完整类名是一致的，CSEJavaSDK就会认为这是同一个Java类型。如果consumer端的参数类型和provider端参数类型不在同一个包中，也可以调用通过，但是中间

多了 契约声明类型 -> json串 -> 代码中的参数类型 的转换过程，增加了运行开销。对于这种情况，CSEJavaSDK在加载provider端契约时会打印提示信息，如“Bad practice, low performance, convert from class

microservice.demo.training21days.provider.service2.Person to class

microservice.demo.training21days.provider.service.Person”。

- 3、当在postman中选择body的类型为application/json时，postman会自动在header中加上Content-Type:application/json。微服务在解析body中承载的数据时，会根据Content-Type来判断应该如何处理数据，因此该header不能去掉，否则将导致微服务解析body失败。



- 4、对于实践过程中碰到的问题，可以参考开源的[ServiceComb-Java-Chassis文档](#)、[CSEJavaSDK文档](#)，可以去[华为云论坛CSE板块](#)搜索资料，有意见和建议也可以去[开源代码库](#)提issue。

善用搜索引擎可以更好地搜索相关信息：



域名解析问题 site:bbs.huaweicloud.com

[全部](#)[新闻](#)[图片](#)[视频](#)[地图](#)[更多](#)

找到约 9,720 条结果（用时 0.23 秒）

[部署任务失败，日志提示DNS解析问题？_云问答_云社区-华为云](#)<https://bbs.huaweicloud.com/ask/1522461230685923> ▼

烟花易冷. 2018-04-01 21:56. 0. 0. **域名解析**不生效的表现是使用ping命令无法解析IP地址，DNS解析失败的原因有很多，具体请参照如下信息：

[服务部署启动后报告域名解析失败_微服务引擎_云论坛_华为云](#)<https://bbs.huaweicloud.com/forum/thread-11806-1-1.html> ▼

2018年10月17日 - 有些时候，在Windows笔记本电脑上做开发的时候也会碰到。这是由于Netty在Windows多网卡环境下做DNS**域名解析**的时候，选错了 ...

参考答案：



Demo-Day2-Homework.zip

Day3 感知微服务和CSE的交互

1 打卡任务

作业：

1、从ServiceStage工具下载页面下载LocalCSE包，在本地安装运行。

基于Day2的demo，修改provider服务和consumer服务的microservice.yaml配置，令两个微服务注册到LocalCSE，并能够正常相互调用。打卡：

- 1、调用consumer服务的greeting方法成功，并截图
- 2、截取consumer服务的日志图片，要求包含consumer服务实例注册成功的日志、没有连接cc/monitor服务报错的日志
- 2、

打卡任务基于Day2的demo项目：



2 准备工作

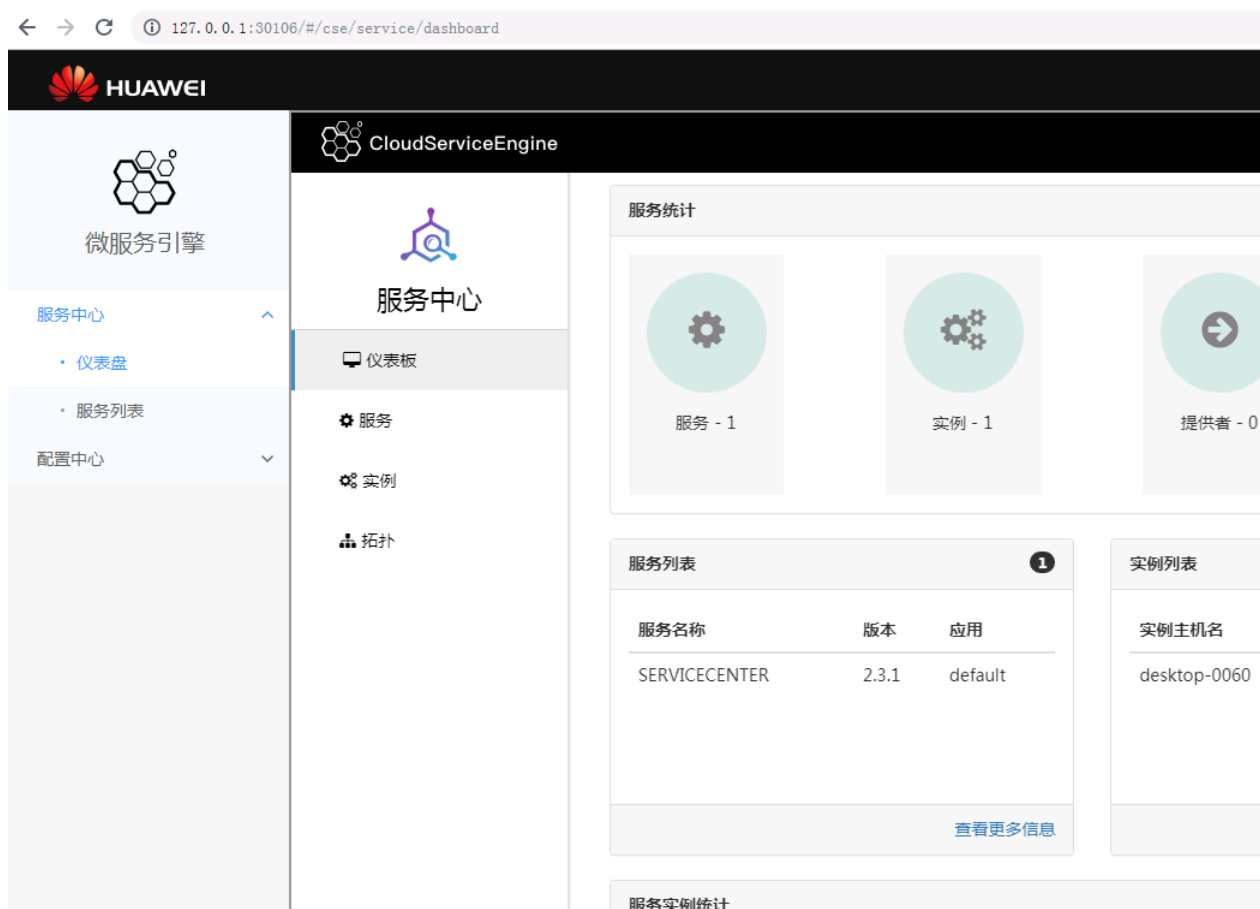
- 1、登录华为云ServiceStage页面，点击进入应用开发 -> 微服务开发 -> 工具下载 -> 本地轻量化微服务引擎，点击下载最新版本。



2、解压下载包，运行其中的start.bat命令，启动LocalCSE

LocalCSE会占用多个端口号，详见 README.md 中的描述，运行时需要保证这些端口没有被其他服务占用。

3、在浏览器输入 <http://127.0.0.1:30106>，能够打开LocalCSE的页面



3 配置微服务连接 LocalCSE

- 1、修改microservice.yaml文件中sc/cc的地址

sc地址改为 <http://127.0.0.1:30100>

cc地址改为 <http://127.0.0.1:30113>

- 2、关闭monitor客户端

仅仅删除monitor的地址不能关闭monitor客户端，微服务会尝试从sc自动发现monitor服务的地址，进而报错。因此需要在 microservice.yaml 文件中配置 servicecomb.monitor.client.enabled=false 显式将其关闭。

- 3、配置完成的microservice.yaml文件如下图所示

```
cse:
  service:
    registry:
      address: http://127.0.0.1:30100
      instance:
        watch: false
    config:
      client:
        serverUri: http://127.0.0.1:30113
        refreshMode: 1
        refresh_interval: 5000
    monitor:
      client:
        serverUri: https://cse.cn-north-1.myhuaweicloud.com:443
        enabled: false
```

4、启动工程，可以看到LocalCSE的服务列表中出现provider和consumer服务的记录

The screenshot shows the Huawei CloudServiceEngine console. On the left is a navigation menu with '微服务引擎' (Microservice Engine) and '服务中心' (Service Center). The main area displays the '服务中心' (Service Center) with a '服务列表' (Service List) table. The table lists three services: 'consumer', 'provider', and 'SERVICECENTER', all with status 'UP'.

名称	状态	应用	版本	创建时间
consumer	UP	Training21Days-HelloWorld	0.0.1	2019-2-14 16:5
provider	UP	Training21Days-HelloWorld	0.0.1	2019-2-14 16:0
SERVICECENTER	UP	default	2.3.1	2019-2-14 15:37

4 打卡截图

1.调用consumer服务的greeting方法成功，并截图

2.截取consumer服务的日志图片，要求包含consumer服务实例注册成功的日志、没有连接cc/monitor服务报错的日志

调用consumer服务的结果如下：

127.0.0.1:9090/consumer/v0/greeting

POST 127.0.0.1:9090/consumer/v0/greeting

Params Authorization Headers (1) Body Pre-request Script Tests

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON (application/json)

```
1 {
2   "name": "Wilson",
3   "gender": "FEMALE"
4 }
```

Body Cookies Headers (2) Test Results Status: 200 OK

Pretty Raw Preview JSON

```
1 {
2   "msg": "Hello, Ms.Wilson",
3   "timestamp": "2019-02-14T08:07:02.953Z"
4 }
```

consumer日志如下:

```
[INFO] read MicroserviceRegisterTask status is FINISHED org.apache.servicecomb.servicereg
[INFO] running microservice instance register task. org.apache.servicecomb.serviceregistr
[INFO] Register microservice instance success. MicroserviceId=134332a31760cefff9103781f67
[INFO] receive MicroserviceInstanceRegisterTask event, check instance Id... org.apache.se
[INFO] instance registry succeeds for the first time, will send AFTER_REGISTRY event. org
[WARN] keyStore [server.p12] file not exist, please check! org.apache.servicecomb.foundat
[WARN] trustStore [trust.jks] file not exist, please check! org.apache.servicecomb.founda
[INFO] Monitor data sender started. Configured data providers is {com.huawei.paas.monitor
[INFO] ServiceComb is ready. org.apache.servicecomb.core.SCBEEngine$1.afterRegistryInstanc
[INFO] read MicroserviceInstanceRegisterTask status is FINISHED org.apache.servicecomb.se
[INFO] Waiting for status up. timeout: 10000ms org.apache.servicecomb.core.SCBEEngine.wait
[INFO] Status already changed to up. org.apache.servicecomb.core.SCBEEngine.waitForStatusUp(S
[INFO] sc task interval changed from -1 to 30 org.apache.servicecomb.serviceregistry.task
[INFO] Found SPI service javax.ws.rs.core.Response$StatusType, count=0. org.apache.servic
[INFO] Found SPI service org.apache.servicecomb.core.tracing.TraceIdGenerator, count=1. o
[INFO] ... 0. org.apache.servicecomb.core.tracing.BraceTraceIdGenerator. org.apache.service
```

5 小提示

1、关于配置项前缀: 有些同学可能注意到了, 课程资料中写的sc/cc/monitor配置的前

缀是servicecomb，但是示例中的配置项前缀是cse。这是为了开源的ServiceComb-Java-Chassis和CSEJavaSDK相互兼容，框架内部自动将配置项做了映射，cse. 开头的配置项也会映射一份 servicecomb. 开头的。

参考答案：



Day4 微服务实例的生命周期分析

1 打卡任务

作业：

- 1、定义BootListener实现类，监听ServiceComb框架启动事件，在实例注册成功时和实例停机前打印提示日志。
- 2、将provider和consumer服务的环境改为development环境，启动demo调用成功。然后修改provider的sayHello接口，将请求参数name从path参数改为query参数，重启provider/consumer令调用仍然成功。

打卡：

- 1、启动服务进程，然后正常退出，截取服务日志图片，要求包含实例注册成功和实例停机时自定义BootListener扩展打印的提示日志。
- 2、截取provider服务启动日志中关于development环境检测到契约变化后重新注册契约的特征日志。
- 3、截取直接调用provider和通过consumer调用provider成功的截图。

打卡任务基于Day2的demo项目：



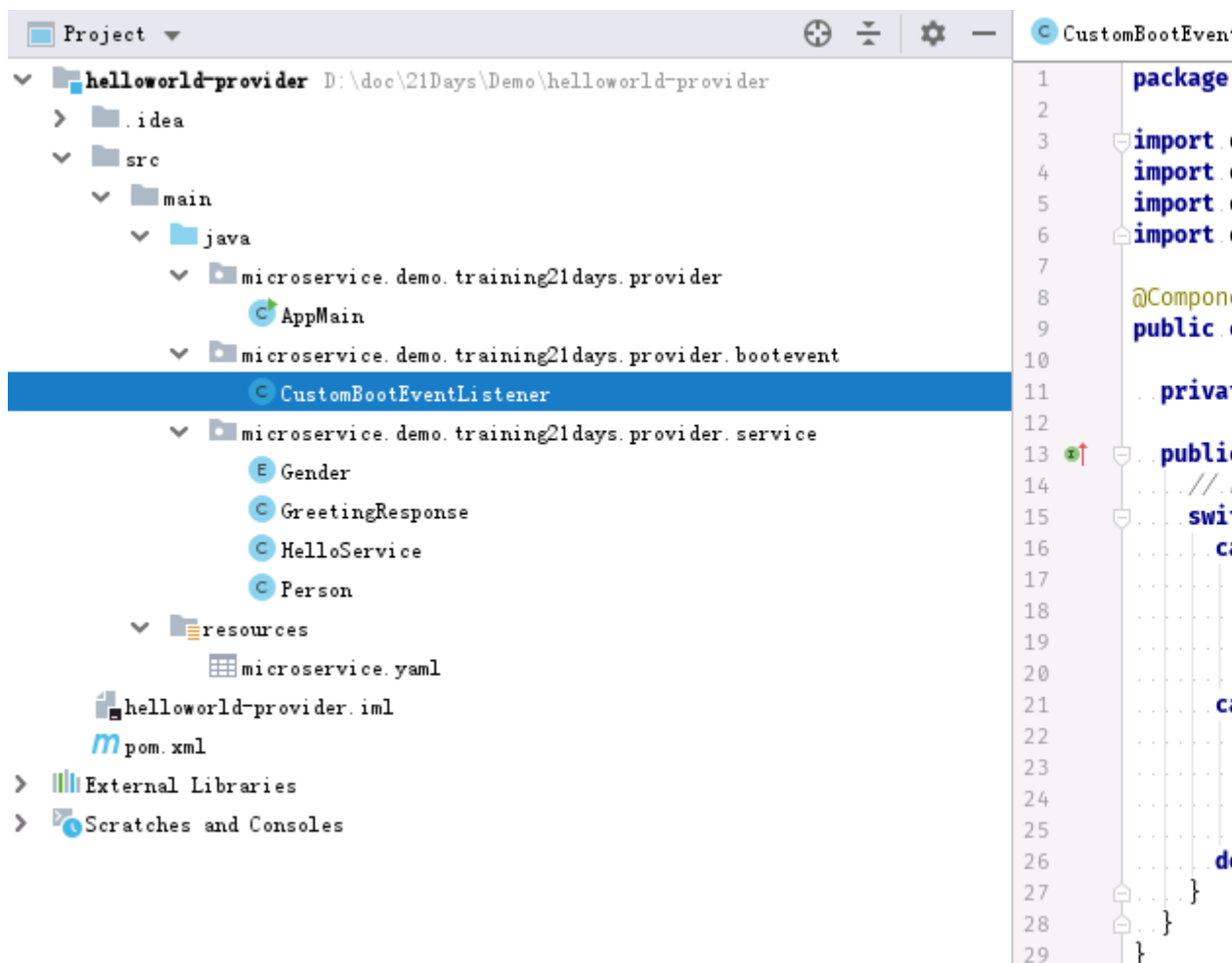
2 准备工作

- 1、正常运行Day2的demo

3 监听 ServiceComb 框架启动事件

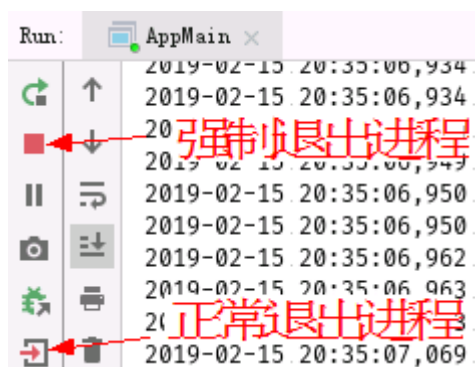
- 1、在provider服务中定义一个BootListener的子类，打上@Component注解使Spring框架

在启动时将其实例化为Spring Bean。



2、启动与停止provider服务进程

注意有些IDE上会提供两种停止方式，以IDEA为例，上面的红色方形按钮是强制退出，不会触发优雅停机。下面的图标是正常退出进程，可以触发优雅停机。大家可以将两种方式都试一下，体会一下不同退出方式下，sc感知实例下线的时间差。



3、 日志如下

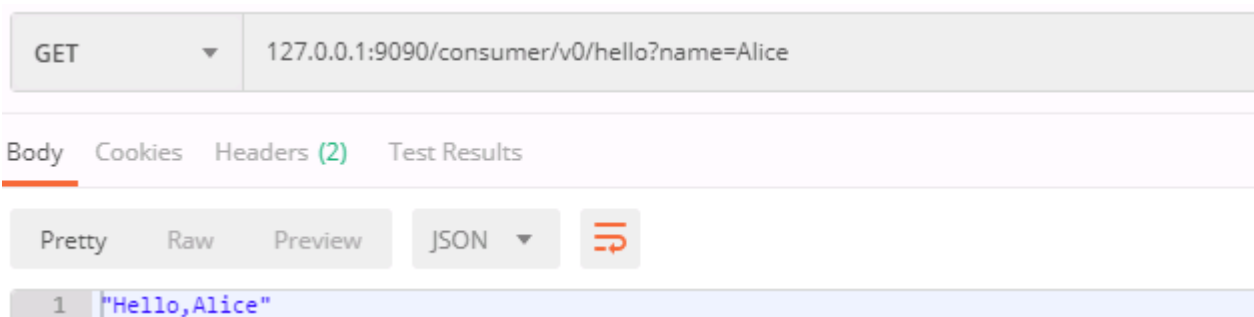
```
===== microservice.demo.training21days.provider.bootevent.CustomB
Service startup completed! microservice.demo.training21days.provider.bootevent.CustomBoot
===== microservice.demo.training21days.provider.bootevent.CustomB
keyStore [server.p12] file not exist, please check! org.apache.servicecomb.foundation.ver
trustStore [trust.jks] file not exist, please check! org.apache.servicecomb.foundation.ve
Monitor data sender started. Configured data providers is {com.huawei.paas.monitor.Health
ServiceComb is ready. org.apache.servicecomb.core.SCBEngine$1.afterRegistryInstance(SCBEn
Waiting for status up. timeout: 10000ms org.apache.servicecomb.core.SCBEngine.waitStatusU
Status already changed to up. org.apache.servicecomb.core.SCBEngine.waitStatusUp(SCBEngin
ServiceComb is closing now... org.apache.servicecomb.core.SCBEngine.destroy(SCBEngine.jav
Closing org.springframework.context.support.ClassPathXmlApplicationContext@2c9f9fb0: star
BootListener org.apache.servicecomb.core.provider.producer.ProducerProviderManager succee
BootListener org.apache.servicecomb.common.rest.RestEngineSchemaListener succeed to proce
BootListener org.apache.servicecomb.AuthHandlerBoot succeed to process BEFORE_CLOSE. org.
===== microservice.demo.training21days.provider.bootevent.CustomB
JVM process is closing! microservice.demo.training21days.provider.bootevent.CustomBootEve
===== microservice.demo.training21days.provider.bootevent.CustomB
```

4 验证 development 环境允许重新注册契约的功能

- 1、在consumer和provider服务的microservice.yaml文件里，配置环境为development

```
APPLICATION_ID: Training21Days-HelloWorld
service_description:
  name: provider
  version: 0.0.1
  environment: development # 设置为开发环境
```

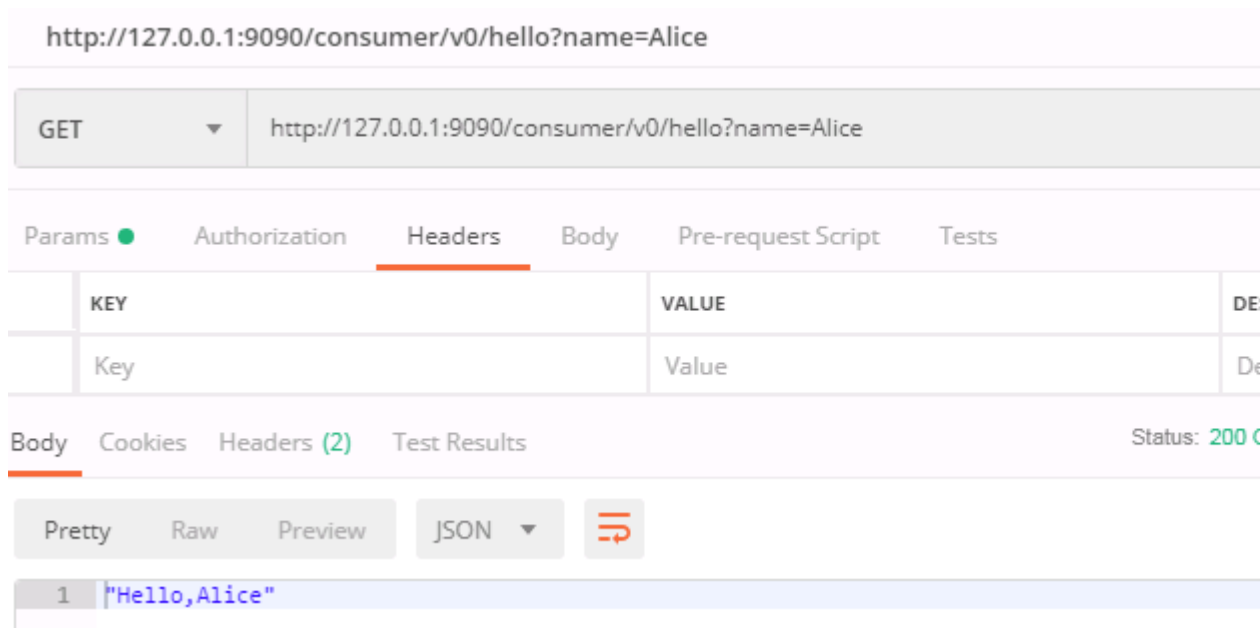
- 2、启动服务，调用consumer，consumer可以正常发现provider服务，并调用成功



- 3、修改provider中sayHello方法的参数，将name变为query参数

```
@RequestMapping(path = "/hello", method = RequestMethod.GET)
public String sayHello(@RequestParam(value = "name") String name) {
    return "Hello," + name;
}
```

- 4、重启provider和consumer服务，调用consumer的sayHello方法，consumer调用provider成功



Provider服务重启时检测到本地契约与sc中已注册的契约不同，会在

`service_description.environment=development`时重新注册契约，启动日志中会打印如

下内容：

```
[INFO] Microservice exists in service center, no need to register. id=[bad44fbabce8b1bb0b  
[INFO] SchemaIds are equals to service center. serviceId=[bad44fbabce8b1bb0be4e237367547c  
[INFO] schemaId [hello] exists [true], summary exists [true] org.apache.servicecomb.servi  
[INFO] schema[hello]'s content is changed and the current environment is [development], s  
[INFO] register schema bad44fbabce8b1bb0be4e237367547c156c7ef8b/hello success. org.apache
```

5 打卡截图

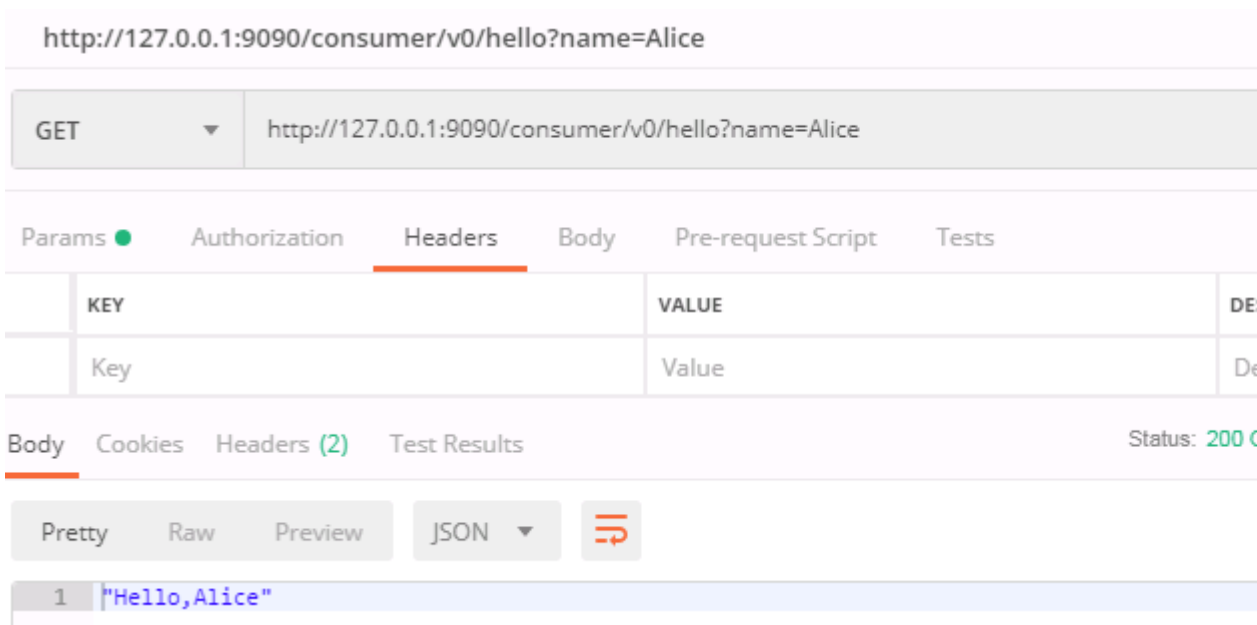
1、启动成功和实例退出的提示日志如下

```
===== microservice.demo.training21days.provider.bootevent.CustomB  
Service startup completed! microservice.demo.training21days.provider.bootevent.CustomBoot  
===== microservice.demo.training21days.provider.bootevent.CustomB  
keyStore [server.p12] file not exist, please check! org.apache.servicecomb.foundation.ver  
trustStore [trust.jks] file not exist, please check! org.apache.servicecomb.foundation.ve  
Monitor data sender started. Configured data providers is {com.huawei.paas.monitor.Health  
ServiceComb is ready. org.apache.servicecomb.core.SCBEngine$1.afterRegistryInstance(SCBE  
Waiting for status up. timeout: 10000ms org.apache.servicecomb.core.SCBEngine.waitStatusU  
Status already changed to up. org.apache.servicecomb.core.SCBEngine.waitStatusUp(SCBEngin  
ServiceComb is closing now... org.apache.servicecomb.core.SCBEngine.destroy(SCBEngine.jav  
Closing org.springframework.context.support.ClassPathXmlApplicationContext@2c9f9fb0: star  
BootListener org.apache.servicecomb.core.provider.producer.ProducerProviderManager succee  
BootListener org.apache.servicecomb.common.rest.RestEngineSchemaListener succeed to proce  
BootListener org.apache.servicecomb.AuthHandlerBoot succeed to process BEFORE_CLOSE. org.  
===== microservice.demo.training21days.provider.bootevent.CustomB  
JVM process is closing! microservice.demo.training21days.provider.bootevent.CustomBootEve  
===== microservice.demo.training21days.provider.bootevent.CustomB
```

2、重新注册契约的特征日志如下

```
Microservice exists in service center, no need to register. id=[bad44fbabce8b1bb0be4e2373  
SchemaIds is different between local and service center. serviceId=[bad44fbabce8b1bb0be4e  
schemaId [hello] exists [true], summary exists [true] org.apache.servicecomb.serviceregis  
schema[hello]'s content is changed and the current environment is [development], so re-re  
register schema bad44fbabce8b1bb0be4e237367547c156c7ef8b/hello success. org.apache.servic
```

3、provider接口改变后consumer调用provider成功



6 小提示

- 1、关于Spring Bean扫描包：有些同学可能注意到了CustomBootEventListener类是作为Spring Bean实例化的（通过@Component注解），但是没有哪里定义扫描包的范围。这是因为AppMain类启动时执行的BeanUtils.init()会找到main类所在的包，并将其加入到Spring扫描包范围里，而CustomBootEventListener在AppMain类所在包的子包里，所以会被自动扫描和加载。
- 2、在修改provider接口后，如果仅重启provider，通过consumer调用provider时会报错。这是因为consumer端服务只会加载一次provider端服务契约，时间点是第一调用provider服务的时候。之后不会再根据契约内容变化刷新本地加载的契约了。如果consumer不重启，则它仍然会把name作为path参数发送出去，导致provider端接受到的请求不符合它的契约要求。
- 3、即使重启consumer后，调用consumer的sayHelloRestTemplate方法仍然会出错。这是

因为RPC调用模式下，CSEJavaSDK框架可以根据服务契约按顺序将请求参数填入REST请求的特定位置发送出去；而RestTemplate调用模式下，name参数的位置是在发起调用的业务代码里决定的，如果不修改代码，name仍然是作为path参数传递的。

```

.. @Path("/helloRT")
.. @GET
.. public String sayHelloRestTemplate(@QueryParam("name") String name) {
..     // RestTemplate 使用方式与原生的Spring.RestTemplate相同，可以直接参考原生Spring的资料
..     // 注意URL不是 http://{IP}:{port}，而是 cse://{provider端服务名}，其他部分如path/query
..     ResponseEntity<String> responseEntity =
..         restTemplate.getForEntity("cse://provider/provider/v0/hello/" + name, String.class);
..     return responseEntity.getBody();
.. }

```

需要修改代码，将name改为从query参数中传递，才能成功调用provider服务，如下图所示：

```

.. @Path("/helloRT")
.. @GET
.. public String sayHelloRestTemplate(@QueryParam("name") String name) {
..     // RestTemplate 使用方式与原生的Spring.RestTemplate相同，可以直接参考原生Spring的资料
..     // 注意URL不是 http://{IP}:{port}，而是 cse://{provider端服务名}，其他部分如path/query
..     ResponseEntity<String> responseEntity =
..         restTemplate.getForEntity("cse://provider/provider/v0/hello?name={1}", String.class);
..     return responseEntity.getBody();
.. }

```

参考答案：



Day5 教你如何配置你的微服务

1 打卡任务

作业：

- 1、将provider服务打成可执行jar包，体验磁盘上的配置文件覆盖jar包内配置和环境变量覆盖配置文件的功能。
- 2、在hello.sayHelloPrefix配置项上加上回调方法，在配置项改变时打印提示日志。
- 3、开启accesslog功能，并将accesslog日志内容输出到业务日志中。

打卡：

- 1、截取provider刷新动态配置时，触发hello.sayHelloPrefix配置项回调方法打印日志的截图。
- 2、截取日志中accesslog打印的内容。

打卡任务基于Day5的demo项目：



Demo-Day5.zip

2 准备工作

- 1、正常运行Day5的demo

3 验证微服务配置优先级关系

- 1、在provider服务的pom文件中加上复制依赖包和打可执行jar包的插件配置

```

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>2.10</version>
      <executions>
        <execution>
          <id>copy-dependencies</id>
          <phase>package</phase>
          <goals>
            <goal>copy-dependencies</goal>
          </goals>
          <configuration>
            <outputDirectory>target/lib</outputDirectory>
          </configuration>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>2.6</version>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <classpathPrefix>./lib/</classpathPrefix>
            <mainClass>${main.class}</mainClass>
            <addDefaultImplementationEntries>true</addDefaultImplementationEntries>
            <addDefaultSpecificationEntries>true</addDefaultSpecificationEntries>
          </manifest>
          <manifestEntries>
            <Class-Path>./</Class-Path>
          </manifestEntries>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>

```

注意在properties中增加一个main.class的配置，指向provider服务的Main类

```

<properties>
  <cse.version>2.3.62</cse.version>
  <main.class>microservice.demo.training21days.provider.AppMain</main.class>
</properties>

```

- 2、执行mvn clean package，在target目录下可以看到打好的jar包和对应的lib目录。实际部署到执行环境的时候需要把jar包和lib目录同时复制过去。

在命令行中执行java -jar helloworld-provider-0.0.1-SNAPSHOT.jar，启动服务。

classes	2019/2/17 17:54	文件夹
lib	2019/2/17 17:55	文件夹
maven-archiver	2019/2/17 17:54	文件夹
maven-status	2019/2/17 17:54	文件夹
helloworld-provider-0.0.1-SNAPSHOT.jar	2019/2/17 17:54	JAR 文件 10 KB

```

C:\Windows\System32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

D:\doc\21Days\Demo\helloworld-provider\target>java -jar helloworld-provider-0.
  
```

通过consumer调用该provider服务的sayHello方法，此时返回的前缀是Hello

```

GET http://127.0.0.1:9090/consumer/v0/hello?name=Alice

Pretty Raw Preview JSON
1 "Hello Alice"
  
```

3、在jar包所在目录下放置一份microservice.yaml配置文件，内容如下

classes	2019/2/17 17:54	文件夹
lib	2019/2/17 17:55	文件夹
maven-archiver	2019/2/17 17:54	文件夹
maven-status	2019/2/17 17:54	文件夹
helloworld-provider-0.0.1-SNAPSHOT.jar	2019/2/17 17:54	JAR 文件
microservice.yaml	2019/2/17 18:08	YAML 文件

```

D:\doc\21Days\Demo\helloworld-provider\target\microservice.yaml -
File Edit Selection Find View Goto Tools Project Preferences Help

microservice.yaml x
1 hello:
2   sayHelloPrefix: "Hello(from disk file)"
  
```

重新启动provider服务的jar包，此时得到的应答如下

```

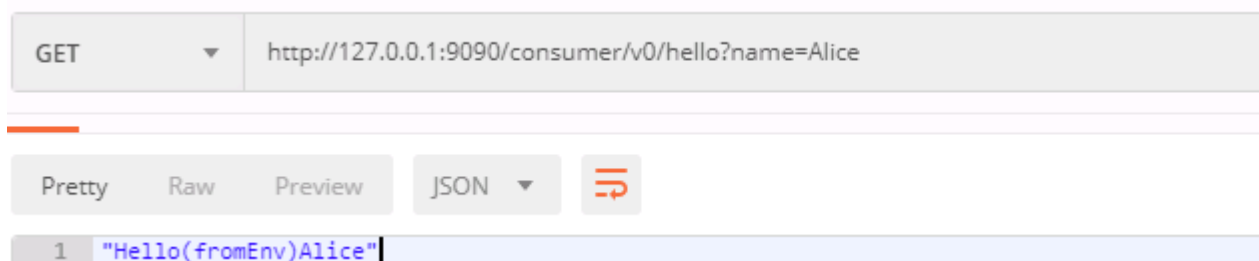
GET http://127.0.0.1:9090/consumer/v0/hello?name=Alice

Pretty Raw Preview JSON
1 "Hello(from disk file) Alice"
  
```

4、设置环境变量hello_sayHelloPrefix=Hello(fromEnv)，再次启动服务

```
D:\doc\21Days\Demo\helloworld-provider\target>set hello_sayHelloPrefix=Hello(f
D:\doc\21Days\Demo\helloworld-provider\target>java -jar helloworld-provider-0.
```

此时调用sayHello方法得到的应答如下



4 给配置项增加回调方法

- 1、在provider服务的HelloService类中增加一个sayHelloPrefix配置项的回调，使其在配置项被刷新时打印一行日志

```
..private static final Logger LOGGER = LoggerFactory.getLogger(HelloService.class);

..private DynamicStringProperty sayHelloPrefix = DynamicPropertyFactory
.....getInstance().getStringProperty( propName: "hello.sayHelloPrefix", defaultValue: ""
....., notifyConfigRefreshed());

..private Runnable notifyConfigRefreshed() {
..    return () -> LOGGER.info("config[hello.sayHelloPrefix] changed to [{}]", sayHelloPre
..}
```

- 2、启动服务，在provider服务的动态配置页面增加hello.sayHelloPrefix的配置

创建配置

作用域

provider@Training21Days-Hello... ▼

* 配置项

hello.sayHelloPrefix

* 值

Hello(fromCC)

确定

取消

待provider服务实例从配置中心刷新配置项后，就会触发回调方法打印日志

```
[INFO] config[hello.sayHelloPrefix] changed to [Hello(fromCC)]! microservice.demo.trainin
[WARN] Config value cache changed: action:create; item:[hello.sayHelloPrefix] org.apache.
[INFO] Updating remote config is done. revision has changed from default94785601 to defau
```

5 启用和配置 accesslog

1、开启accesslog功能

Accesslog功能的说明文档在这里：[https://docs.servicecomb.io/java-](https://docs.servicecomb.io/java-chassis/zh_CN/build-provider/access-log-configuration.html)

[chassis/zh_CN/build-provider/access-log-configuration.html](https://docs.servicecomb.io/java-chassis/zh_CN/build-provider/access-log-configuration.html)。

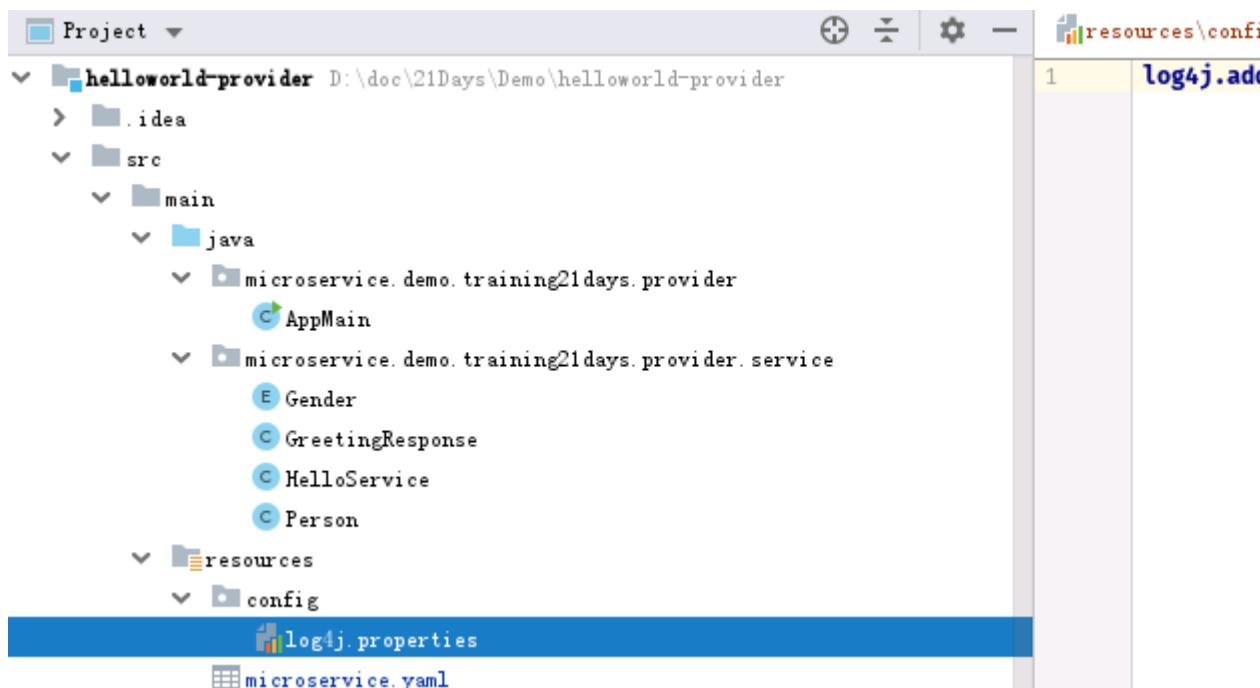
开启accesslog需要在microservice.yaml文件中做如下配置（这里使用了自定义的日志格式，没有用默认配置）：

```
..accesslog:
..  enabled: true
..  pattern: "%h - - %t cs-uri %s %B %D %SCB-traceId"
```

2、配置accesslog日志打印配置

默认的accesslog是单独打印到一个access.log日志文件中的。这里我们为了实验观察

的方便，在项目的resources/config目录下放置一份log4j.properties配置文件，覆盖默认的配置，令accesslog的内容合并到普通业务日志中输出。



- 3、启动服务，调用接口，可以看到日志中打印了accesslog

```
127.0.0.1 - - Sun, 17 Feb 2019 18:46:10 CST /provider/v0/hello/Bob.200 11.650 5c693b723b9
127.0.0.1 - - Sun, 17 Feb 2019 18:46:13 CST /provider/v0/hello/Bob.200 11.3 5c693b75185db
127.0.0.1 - - Sun, 17 Feb 2019 18:46:14 CST /provider/v0/hello/Bob.200 11.2 5c693b766b0e1
127.0.0.1 - - Sun, 17 Feb 2019 18:46:17 CST /provider/v0/hello/Bob.200 11.2 5c693b79a0d45
```

6 打卡截图

- 1.截取provider刷新动态配置时，触发hello.sayHelloPrefix配置项回调方法打印日志的截图。

- 2.截取日志中accesslog打印的内容。

- 1、Provider动态刷新配置项时的回调日志

```
[INFO] config[hello.sayHelloPrefix] changed to [Hello(fromCC)]! microservice.demo.trainin
[WARN] Config value cache changed: action:create; item:[hello.sayHelloPrefix] org.apache.
[INFO] Updating remote config is done. revision has changed from default94785601 to defau
```

- 2、Accesslog日志

```
127.0.0.1 - - Sun, 17 Feb 2019 18:46:10 CST /provider/v0/hello/Bob.200 11.650 5c693b723b9
127.0.0.1 - - Sun, 17 Feb 2019 18:46:13 CST /provider/v0/hello/Bob.200 11.3 5c693b75185db
127.0.0.1 - - Sun, 17 Feb 2019 18:46:14 CST /provider/v0/hello/Bob.200 11.2 5c693b766b0e1
127.0.0.1 - - Sun, 17 Feb 2019 18:46:17 CST /provider/v0/hello/Bob.200 11.2 5c693b79a0d45
```

7 小提示

- 1、对于CSEJavaSDK而言，框架自身的一些治理配置项和用户定义的业务配置项是同等地位的，即内部处理逻辑是完全一致的。用户也可以遵循一样的配置和使用方法。
- 2、日志配置是一个独立的体系，与业务配置不相关。

参考答案：



Day6 CSE实战之开发网关

1 打卡任务

作业：

- 1、使用DefaultEdgeDispatcher开发一个EdgeService网关服务

打卡：

- 1、通过EdgeService成功调用consumer和provider服务并截图。

打卡任务基于Day5打卡作业的demo项目：



2 准备工作

- 1、正常运行Day5的demo

3 开发 EdgeService 网关服务

- 1、在创建一个EdgeService的maven工程，pom文件配置如下：

```
.. <properties>
..   <cse.version>2.3.62</cse.version>
..   <main.class>microservice.demo.training21days.consumer.AppMain</main.class>
.. </properties>

.. <dependencyManagement>
..   <dependencies>
..     <dependency>
..       <groupId>com.huawei.paas.cse</groupId>
..       <artifactId>cse-dependency</artifactId>
..       <version>${cse.version}</version>
..       <type>pom</type>
..       <scope>import</scope>
..     </dependency>
..   </dependencies>
.. </dependencyManagement>

.. <dependencies>
..   <dependency>
..     <groupId>com.huawei.paas.cse</groupId>
..     <artifactId>cse-solution-service-engine</artifactId>
..   </dependency>
..   <dependency>
..     <groupId>org.apache.servicecomb</groupId>
..     <artifactId>edge-core</artifactId>
..   </dependency>
.. </dependencies>
```

2、定义一个main类

```
package microservice.demo.training21days.edge;

import org.apache.servicecomb.foundation.common.utils.BeanUtils;
import org.apache.servicecomb.foundation.common.utils.Log4jUtils;

public class AppMain {
.. public static void main(String[] args) throws Exception {
..   Log4jUtils.init();
..   BeanUtils.init();
.. }
}
```

3、配置一份microservice.yaml文件

```
APPLICATION_ID: Training21Days-HelloWorld
service_description:
  name: edge
  version: 0.0.1
cse:
  service:
    registry:
      address: https://cse.cn-north-1.myhuaweicloud.com:443
      instance:
        watch: false
    config:
      client:
        serverUri: https://cse.cn-north-1.myhuaweicloud.com:443
        refreshMode: 1
        refresh_interval: 5000
    monitor:
      client:
        serverUri: https://cse.cn-north-1.myhuaweicloud.com:443
  rest:
    address: 0.0.0.0:8000

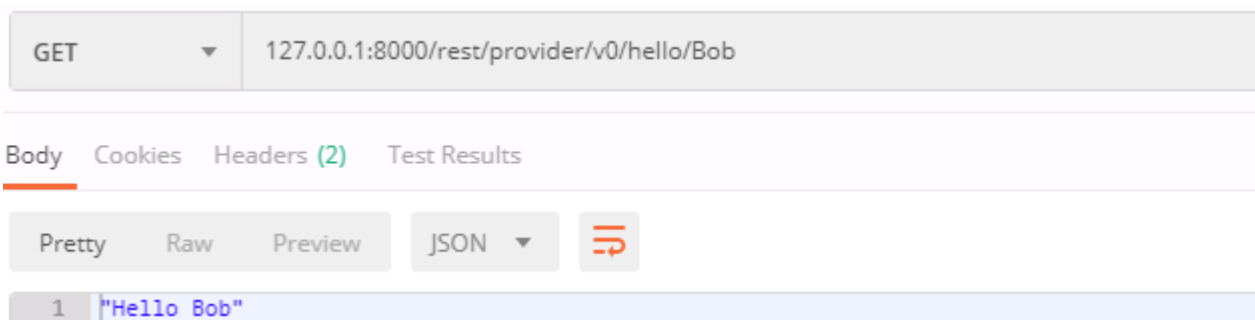
  credentials:
    accessKey: your ak
    secretKey: your sk
    project: cn-north-1
    akskCustomCipher: default

  http:
    dispatcher:
      edge:
        default: ..... # 使用DefaultEdgeDispatcher开发网关服务
        enabled: true ..... # 开启DefaultEdgeDispatcher
        prefix: rest ..... # 匹配请求路径前缀为/rest
        # withVersion默认值就是true, 这里只是展示一下, 实际上可以省略该配置
        withVersion: true ..... # 请求带版本号, 例如v1表示[1.0.0,2.0.0)范围内的微服务版本
        # prefixSegmentCount默认值就是1, 这里只是展示一下, 实际上可以省略该配置
        prefixSegmentCount: 1 ..... # 前缀长度, 例如/rest/provider/v0/hello/Bob, 去掉第一段, 将/p
```

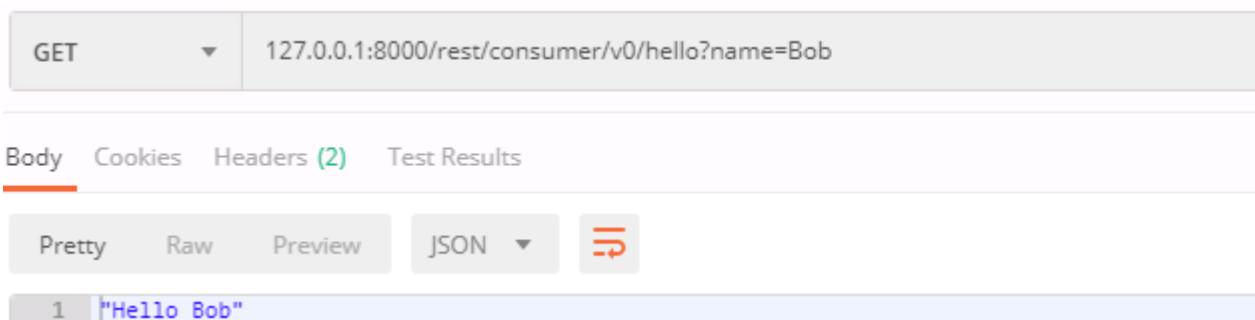
4、启动edge/consumer/provider三个服务，通过edge调用consumer和provider

4 打卡截图

1、通过edge调用provider服务



2、通过edge调用consumer服务



参考答案:



Day7 CSE实战之框架扩展机制

1 打卡任务

作业：

- 1、在edge服务上开发一个HttpServerFilter，该filter从请求中取出名为Username和Password的header存入InvocationContext中，如果没有这两个header则报错，返回401 Unauthorized。
- 2、在edge服务上开发一个Handler，判断InvocationContext中的Username和Password是否相等，如果不相等，则返回401 Unauthorized。并且将该handler配置在edge的consumer端handler链上，只对provider服务生效。

打卡：

- 1、不传Username和Password，调用edge服务，截图报错信息
- 2、通过edge调用provider服务成功并截图。
- 3、通过edge调用consumer的sayHello方法，并且令Username和Password不相等，截图调用成功的结果。

打卡任务基于Day7的demo项目：



Demo-Day7.zip

2 准备工作

- 1、正常运行Day7的demo

3 开发 HttpServerFilter

- 1、在edge服务中创建一个DemoAuthenticationFilter：

```

public class DemoAuthenticationFilter implements HttpServerFilter {

    public static final String USERNAME = "Username";

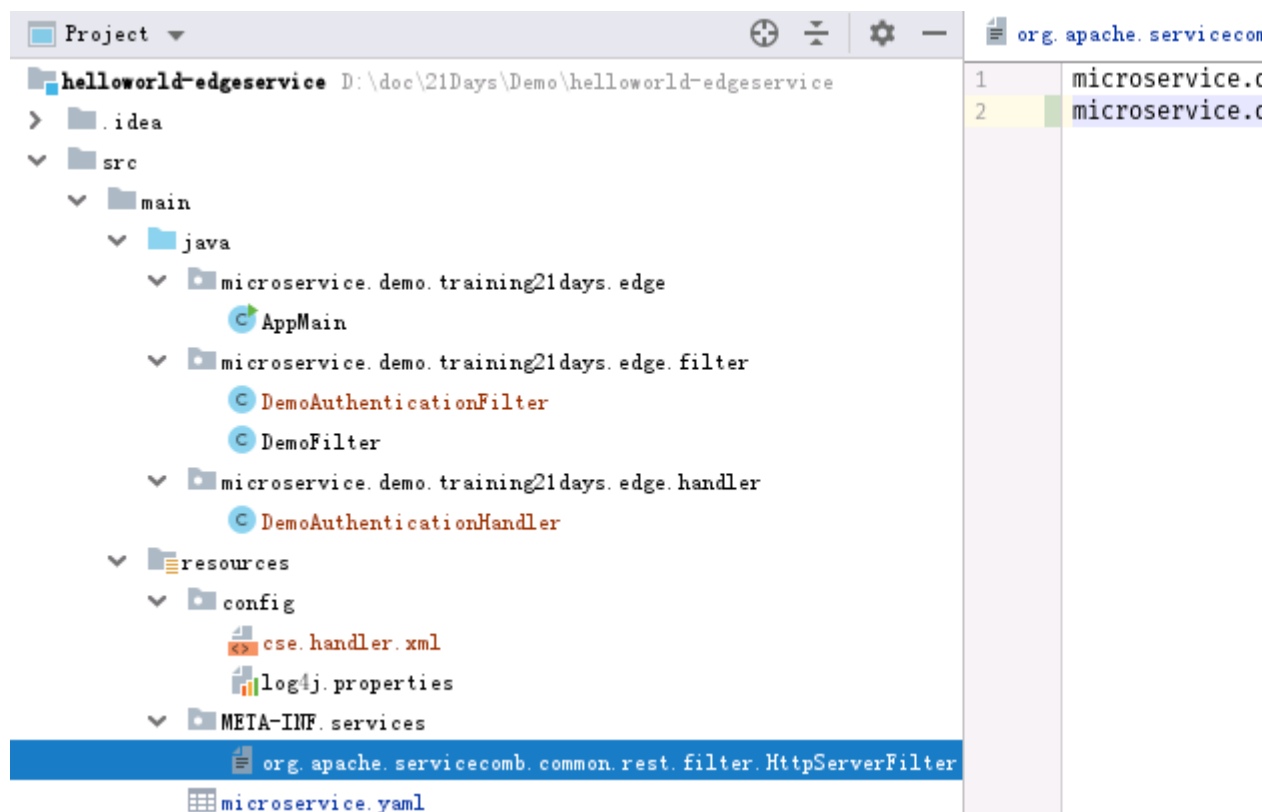
    public static final String PASSWORD = "Password";

    @Override
    public int getOrder() {
        return 0;
    }

    @Override
    public Response afterReceiveRequest(Invocation invocation, HttpServletRequestEx httpServletRequestEx) {
        String username = httpServletRequestEx.getHeader(USERNAME);
        String password = httpServletRequestEx.getHeader(PASSWORD);
        if (StringUtils.isEmpty(username) || StringUtils.isEmpty(password)) {
            return Response.consumerFailResp(
                new InvocationException(Status.UNAUTHORIZED, new CommonExceptionData("Lack of authentication information"))
            );
        }
        invocation.addContext(USERNAME, username);
        invocation.addContext(PASSWORD, password);
        return null;
    }
}

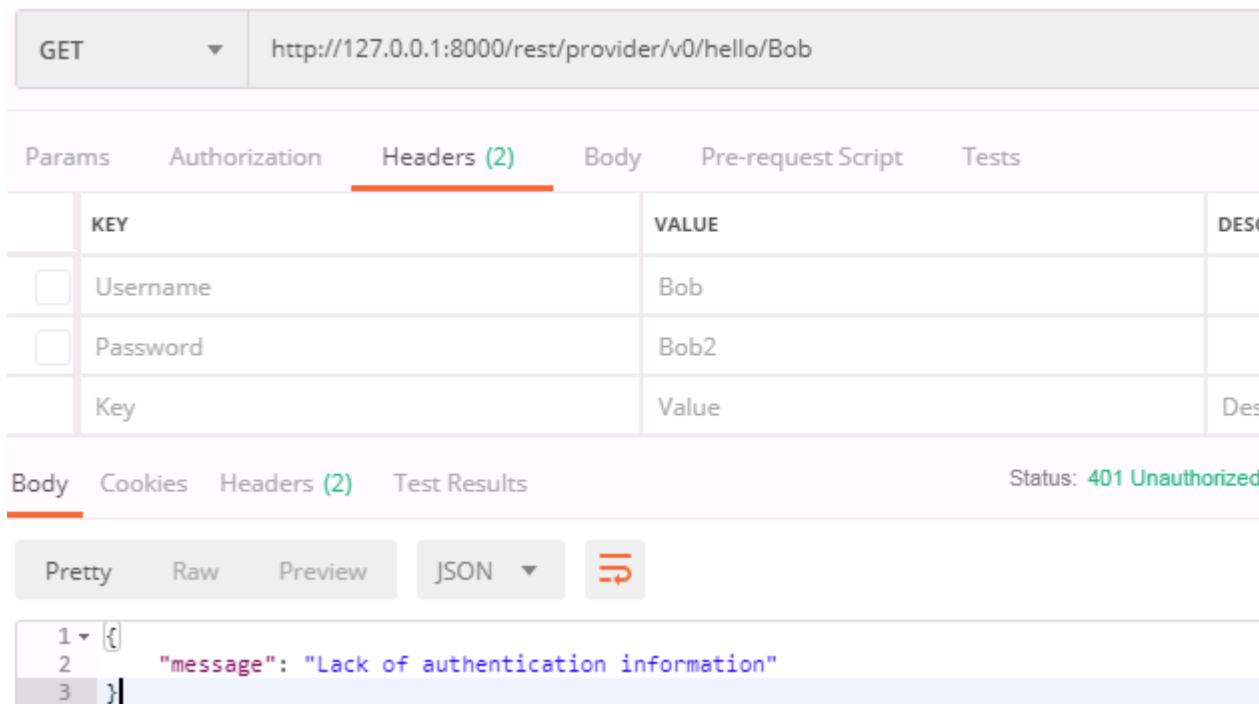
```

- 2、在HttpServerFilter的SPI配置文件中增加DemoAuthenticationFilter的配置：

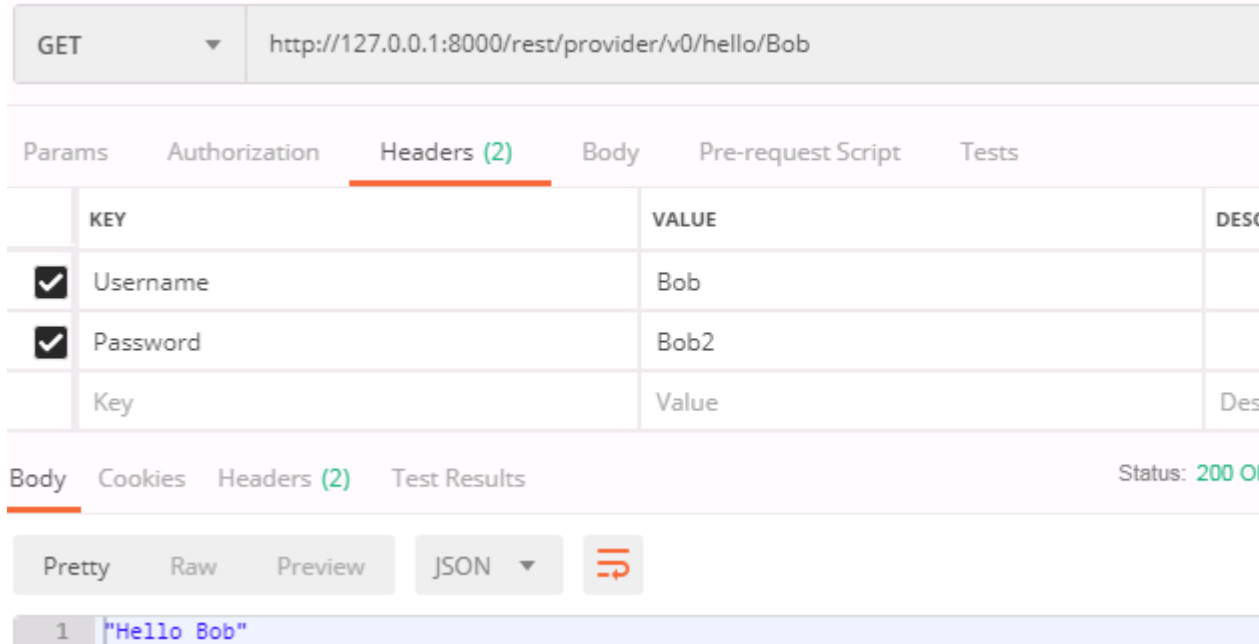


- 3、运行edge服务，此时如果不在header中传Username和Password，是无法调通后端服

务的:



但是如果加入了Username和Password，即使不相等也可以调用通过



4、该限制对于所有服务和接口都生效，即使通过edge调用的是provider

4 开发 Handler

1、在edge服务中定义一个DemoAuthenticationHandler:

```
package microservice.demo.training21days.edge.handler;

import javax.ws.rs.core.Response.Status;

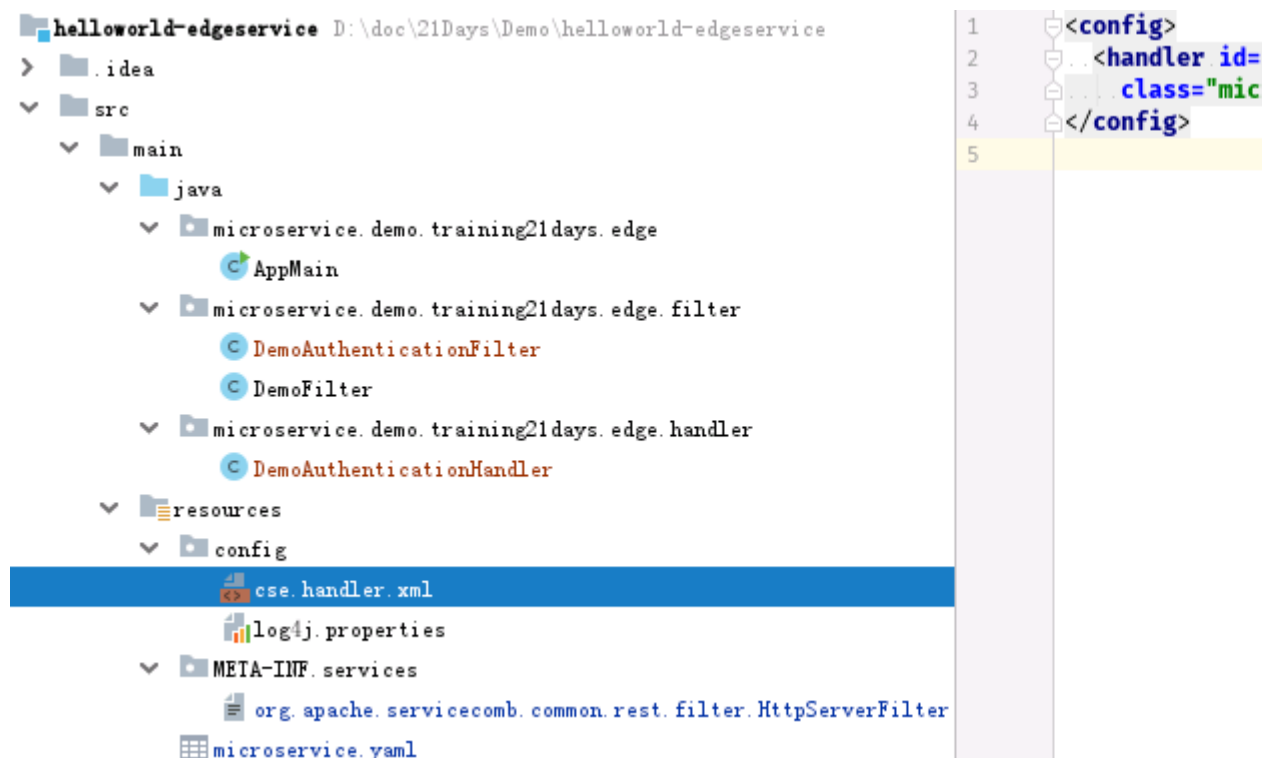
import org.apache.servicecomb.core.Handler;
import org.apache.servicecomb.core.Invocation;
import org.apache.servicecomb.swagger.invocation.AsyncResponse;
import org.apache.servicecomb.swagger.invocation.exception.InvocationException;

import microservice.demo.training21days.edge.filter.DemoAuthenticationFilter;

public class DemoAuthenticationHandler implements Handler {
    @Override
    public void handle(Invocation invocation, AsyncResponse asyncResp) throws Exception {
        String username = invocation.getContext(DemoAuthenticationFilter.USERNAME);
        String password = invocation.getContext(DemoAuthenticationFilter.PASSWORD);
        if (null == username || !username.equals(password)) {
            asyncResp.consumerFail(new InvocationException(Status.UNAUTHORIZED, "Wrong authentication"));
            return;
        }
        invocation.next(asyncResp);
    }
}
```

- 2、在resources/config目录下创建一份cse.handler.xml配置文件，声明

DemoAuthenticationHandler:



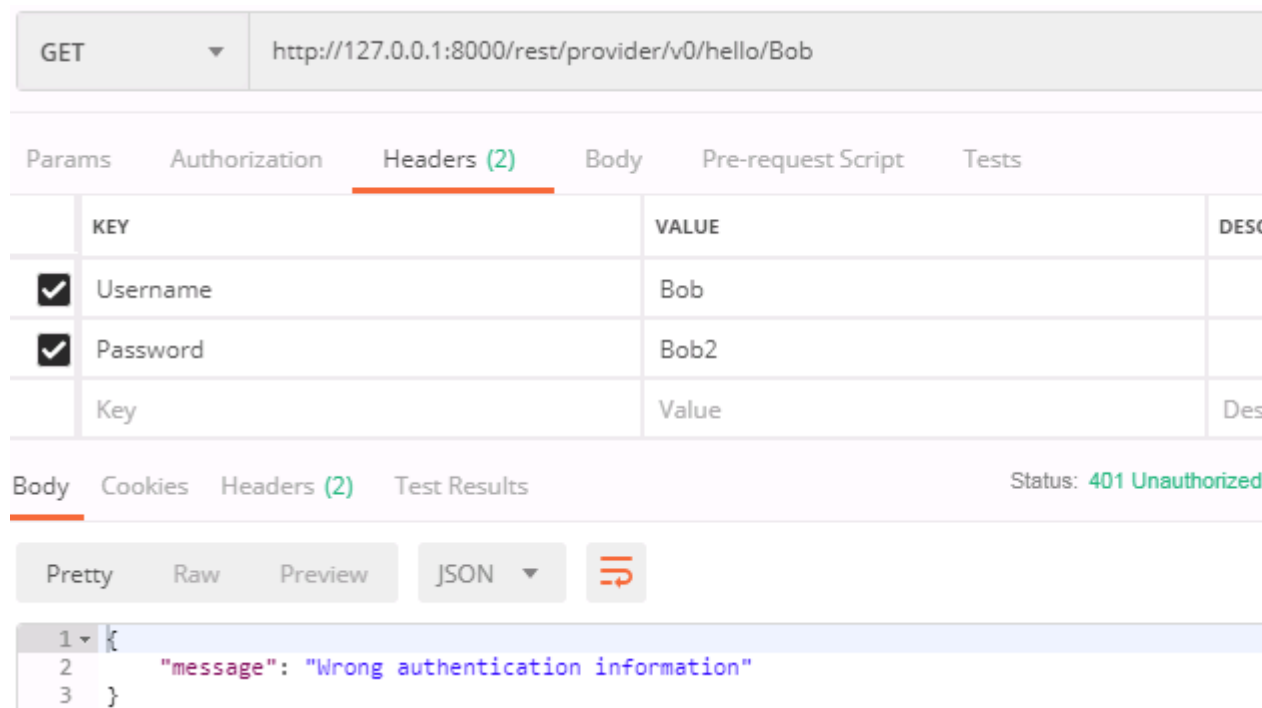
- 3、在microservice.yaml配置文件中定义edge调用provider时的handler链:

```
cse:
  handler:
  chain:
    Consumer:
      service:
        #.该handler链配置只在edge调用provider服务时生效，edge调用其他服务时走的还是默认的handler链
        provider: demo-authentication-handler,qps-flowcontrol-consumer,loadbalance,fault-tolerance
```

注意这里的配置不同于培训PPT中的配置。Day7培训PPT中的配置是provider端默认handler链，而这里的consumer端handler链配置只作用于edge调用provider服务时，如果是edge调用consumer服务则走的是默认的handler链配置。

4、启动edge服务，测试效果

调用provider服务：



KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Username	Bob	
<input checked="" type="checkbox"/> Password	Bob2	
Key	Value	Description

Body Cookies Headers (2) Test Results Status: 401 Unauthorized

Pretty Raw Preview JSON

```
1 {
2   "message": "Wrong authentication information"
3 }
```

调用consumer服务：

GET 127.0.0.1:8000/rest/consumer/v0/hello?name=Alice

Params Authorization Headers (2) Body Pre-request Script Tests

	KEY	VALUE	DESC
<input checked="" type="checkbox"/>	Username	Bob	
<input checked="" type="checkbox"/>	Password	Bob2	
	Key	Value	Des

Body Cookies Headers (2) Test Results Status: 200 OK

Pretty Raw Preview JSON

```
1 "Hello Alice"
```

5 打卡截图

1、不传Username和Password，调用edge服务

GET http://127.0.0.1:8000/rest/provider/v0/hello/Bob

Params Authorization Headers (2) Body Pre-request Script Tests

	KEY	VALUE	DESC
<input type="checkbox"/>	Username	Bob	
<input type="checkbox"/>	Password	Bob2	
	Key	Value	Des

Body Cookies Headers (2) Test Results Status: 401 Unauthorized

Pretty Raw Preview JSON

```
1 {
2   "message": "Lack of authentication information"
3 }
```

2、通过edge调用provider服务成功

GET
http://127.0.0.1:8000/rest/provider/v0/hello/Bob

Params
Authorization
Headers (2)
Body
Pre-request Script
Tests

	KEY	VALUE	DESC
<input checked="" type="checkbox"/>	Username	Bob	
<input checked="" type="checkbox"/>	Password	Bob	
	Key	Value	Des

Body
Cookies
Headers (2)
Test Results
Status: 200 OK

Pretty
Raw
Preview
JSON

1 "Hello Bob"

- 3、通过edge调用consumer的sayHello方法，并且令Username和Password不相等，截图调用成功的结果。

GET
127.0.0.1:8000/rest/consumer/v0/hello?name=Alice

Params
Authorization
Headers (2)
Body
Pre-request Script
Tests

	KEY	VALUE	DESC
<input checked="" type="checkbox"/>	Username	Bob	
<input checked="" type="checkbox"/>	Password	Bob2	
	Key	Value	Des

Body
Cookies
Headers (2)
Test Results
Status: 200 OK

Pretty
Raw
Preview
JSON

1 "Hello Alice"

参考答案：



Day8 CSE实战之负载均衡

1 打卡任务

作业：

- 1、启动edge、consumer、provider实例各一个，通过edge调用consumer。然后启动另一个provider，观察consumer日志中刷新provider实例列表的日志。
- 2、连续调用consumer，退出一个provider，观察调用consumer的结果情况。

打卡：

- 1、截图consumer中刷新provider实例列表的特征日志
- 2、截图provider服务的一个实例退出并且连续调用consumer服务时，edge服务的日志

打卡任务基于Day7的demo项目：



2 准备工作

- 1、正常运行Day7的demo

3 观察 consumer 刷新 provider 实例列表的日志

- 1、启动edge、consumer、provider服务的实例各一个，通过edge调用consumer服务成功：



- 2、启动第二个provider实例，此时sc中有1个edge实例、1个consumer实例、2个provider实



例:

应用列表 微服务列表 实例列表

创建微服务

删除

Training21Days-Hello...

微服务名称

Q

C

<input type="checkbox"/>	微服务名称	所属应用	版本号	实例数	创建时间	操作
<input type="checkbox"/>	provider	Training21Days-HelloWorld	1	2	2019/03/02 10:05:40 GMT+08:00	删除
<input type="checkbox"/>	consumer	Training21Days-HelloWorld	1	1	2019/03/02 10:05:37 GMT+08:00	删除
<input type="checkbox"/>	edge	Training21Days-HelloWorld	1	1	2019/03/02 10:05:33 GMT+08:00	删除

3、当consumer的定时查询微服务实例任务从sc查询到provider的新实例时，会打印如下日志：

```
[INFO] find instances[2] from service center success. service=Training21Days-HelloWorld/provider/0.0.0, old revision=90131437abf15221a355628215105f8f85bc05c9, new revision=bb751686ea4772b0c92a012e7404de46c6d6b2f1 org.a
[INFO] service id=8a3eefaca25baeb05ebf98737d678b68bf50c92, instance id=adb613e83c8f11e9900e0255ac105166, endpoints=[rest://192.168.0.45:8080] org.apache.servicecomb.serviceregistry.registry.AbstractServiceRegistry.find
[INFO] service id=8a3eefaca25baeb05ebf98737d678b68bf50c92, instance id=bcc3fbbac3c8f11e9900e0255ac105166, endpoints=[rest://192.168.0.45:8081] org.apache.servicecomb.serviceregistry.registry.AbstractServiceRegistry.find
```

TIPS: 启动第一个provider实例后，一定要先调用一次consumer，让它去查询provider服务，再启动第二个provider服务的实例。

由于consumer服务查询provider服务信息是懒加载的，如果一直不触发consumer调用provider的话，则consumer一直不会去sc查询和刷新provider服务信息。

4 观察重试和实例隔离

1、完成上一步的操作后，现在应该有1个edge实例、1个consumer实例、2个provider实例在线。我们一边连续调用consumer服务，一边停止一个provider服务的实例，观察edge服务的日志：

```
2019-03-02 10:36:18,318 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:18 CST /rest/consumer/v0/hello?name=Alice 200 13 9 5c79ec22aa832fb6 org...
2019-03-02 10:36:18,458 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:18 CST /rest/consumer/v0/hello?name=Alice 200 13 8 5c79ec22efdd3294 org...
2019-03-02 10:36:18,595 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:18 CST /rest/consumer/v0/hello?name=Alice 200 13 9 5c79ec22efdd3294 org...
2019-03-02 10:36:18,734 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:18 CST /rest/consumer/v0/hello?name=Alice 200 13 9 5c79ec2281824229 org...
2019-03-02 10:36:18,874 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:18 CST /rest/consumer/v0/hello?name=Alice 200 13 11 5c79ec221e738369 org...
2019-03-02 10:36:19,009 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:19 CST /rest/consumer/v0/hello?name=Alice 200 13 7 5c79ec233926822f org...
2019-03-02 10:36:19,151 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:19 CST /rest/consumer/v0/hello?name=Alice 200 13 14 5c79ec233926822f org...
2019-03-02 10:36:20,328 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:20 CST /rest/consumer/v0/hello?name=Alice 200 13 1043 5c79ec23c02dc5d6 org...
2019-03-02 10:36:21,486 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:21 CST /rest/consumer/v0/hello?name=Alice 200 13 1030 5c79ec24700e3b61 org...
2019-03-02 10:36:22,635 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:22 CST /rest/consumer/v0/hello?name=Alice 200 13 1022 5c79ec256f8b1837 org...
2019-03-02 10:36:23,814 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:23 CST /rest/consumer/v0/hello?name=Alice 200 13 1039 5c79ec26ac3b7395 org...
2019-03-02 10:36:24,992 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:24 CST /rest/consumer/v0/hello?name=Alice 200 13 1041 5c79ec272a4e9518 org...
2019-03-02 10:36:25,145 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:25 CST /rest/consumer/v0/hello?name=Alice 200 13 13 5c79ec2904c87588 org...
2019-03-02 10:36:25,281 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:25 CST /rest/consumer/v0/hello?name=Alice 200 13 8 5c79ec292aebc0b6 org...
2019-03-02 10:36:25,500 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:25 CST /rest/consumer/v0/hello?name=Alice 200 13 8 5c79ec292aebc0b6 org...
2019-03-02 10:36:25,638 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:25 CST /rest/consumer/v0/hello?name=Alice 200 13 8 5c79ec29b55cea60 org...
2019-03-02 10:36:25,778 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:25 CST /rest/consumer/v0/hello?name=Alice 200 13 8 5c79ec29b0847ee22 org...
2019-03-02 10:36:25,922 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:25 CST /rest/consumer/v0/hello?name=Alice 200 13 8 5c79ec29e9ca2e2a org...
2019-03-02 10:36:26,061 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:26 CST /rest/consumer/v0/hello?name=Alice 200 13 8 5c79ec2af7117725 org...
2019-03-02 10:36:26,212 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:26 CST /rest/consumer/v0/hello?name=Alice 200 13 20 5c79ec2a3fed1295 org...
2019-03-02 10:36:26,364 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 10:36:26 CST /rest/consumer/v0/hello?name=Alice 200 13 10 5c79ec2ae2a46f44 org...
```

请求返回的状态码始终是200，说明调用都成功了

有五次请求的时延明显大于其他请求，这是consumer调用到已退出的provider实例，导致报错，在重试一次请求后才调用成功的，因此花费时间更长一些

后续调用的时延又恢复到较短的水平，此时consumer已经将有问题的provider实例隔离了，不会再调用它

2、观察consumer服务的日志

当已下线的provider实例还没有被consumer隔离时，consumer会调用到有问题的provider实例上出错，然后在另一个provider实例上重试才成功，有类似于下面的日志内容：

```
2019-03-02 10:36:24,974 [ERROR] Invoke server failed. Operation CONSUMER rest provider.hello.sayHello; server rest://192.168.0.45:8080; 0-0 msg cause:InvocationException,message:InvocationException: code=490;msg=CommonExceptionData [message=C
You can add fallback logic by catching this exception.
info: operation-provider.hello.sayHello;cause:InvocationException,message:InvocationException: code=490;msg=CommonExceptionData [message=Cse Internal Bad Request];cause:AnnotatedConnectException,message:Connection refused: no further inform
2019-03-02 10:36:24,989 [ERROR] Invoke server success. Operation CONSUMER rest provider.hello.sayHello; server rest://192.168.0.45:8080; org.apache.servicecomb.loadbalancer.LoadBalancerHandler$3.onExecutionSuccess(LoadBalancerHandler.java:306)
```

之后consumer将provider问题实例隔离时会打印如下日志：


```
[INFO] 192.168.0.45 - - Sat, 02 Mar 2019 10:36:23 CST /consumer/v0/hello?name=Alice 200 13 1037 5c79ec27a2a4a910
[WARN] Isolate service provider's instance bcc3fbba3c8f11e9900e0255ac105166. org.apache.servicecomb.loadbalance.f
[INFO] 192.168.0.45 - - Sat, 02 Mar 2019 10:36:25 CST /consumer/v0/hello?name=Alice 200 13 9 5c79ec290d4c87588 org
[INFO] 192.168.0.45 - - Sat, 02 Mar 2019 10:36:25 CST /consumer/v0/hello?name=Alice 200 13 5 5c79ec290d32e07d org
[INFO] 192.168.0.45 - - Sat, 02 Mar 2019 10:36:25 CST /consumer/v0/hello?name=Alice 200 13 5 5c79ec292aecb0b6 org
```

5 打卡截图

1、Consumer刷新provider实例记录的日志:

[M10]	26A1ATC	78-B9364	19C9532D960026PAB0314Q318PAB081268-05	72U25UC6	74-PC314P093C8177B40000055229C70210E	64QB07U2-16E2-11705'108'0'42-00081	01E'9B9C6'26A1ATC6C00P'26A1ATC6E121L'16E121L'VPR138C'26A1ATC6E121L'17UQ
[M10]	26A1ATC	78-B9364	19C9532D960026PAB0314Q318PAB081268-05	72U25UC6	74-PC314P093C8177B40000055229C70210E	01E'9B9C6'26A1ATC6C00P'26A1ATC6E121L'16E121L'VPR138C'26A1ATC6E121L'17UQ	01E'9B9C6'26A1ATC6C00P'26A1ATC6E121L'16E121L'VPR138C'26A1ATC6E121L'17UQ
[M10]	74UQ	72U25UC6	17UQ	26A1ATC	6E121L	16E121L	01E'9B9C6'26A1ATC6C00P'26A1ATC6E121L'16E121L'VPR138C'26A1ATC6E121L'17UQ

2、通过edge连续调用consumer，并且下线一个provider实例时，edge服务的日志：

2019-03-02	10:36:18,318	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:18 CST	/rest/consumer/v0/hello?name=Alice	200	13	9	5c79ec22aa832fb6	org..
2019-03-02	10:36:18,458	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:18 CST	/rest/consumer/v0/hello?name=Alice	200	13	8	5c79ec22fedd3294	org..
2019-03-02	10:36:18,595	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:18 CST	/rest/consumer/v0/hello?name=Alice	200	13	9	5c79ec225fa5a057	org..
2019-03-02	10:36:18,734	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:18 CST	/rest/consumer/v0/hello?name=Alice	200	13	9	5c79ec2281824229	org..
2019-03-02	10:36:18,874	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:18 CST	/rest/consumer/v0/hello?name=Alice	200	13	11	5c79ec221e738369	org..
2019-03-02	10:36:19,009	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:19 CST	/rest/consumer/v0/hello?name=Alice	200	13	7	5c79ec233926827f	org..
2019-03-02	10:36:19,151	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:19 CST	/rest/consumer/v0/hello?name=Alice	200	13	14	5c79ec231f4722f5	org..
2019-03-02	10:36:20,328	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:19 CST	/rest/consumer/v0/hello?name=Alice	200	13	1043	5c79ec2304cd5d6	o
2019-03-02	10:36:21,486	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:20 CST	/rest/consumer/v0/hello?name=Alice	200	13	1030	5c79ec24700e3b61	o
2019-03-02	10:36:22,635	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:21 CST	/rest/consumer/v0/hello?name=Alice	200	13	1022	5c79ec256f8b1037	o
2019-03-02	10:36:23,814	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:22 CST	/rest/consumer/v0/hello?name=Alice	200	13	1039	5c79ec26ac3b7395	o
2019-03-02	10:36:24,992	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:23 CST	/rest/consumer/v0/hello?name=Alice	200	13	1041	5c79ec27a2a4a980	o
2019-03-02	10:36:25,145	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:25 CST	/rest/consumer/v0/hello?name=Alice	200	13	13	5c79ec2904c85110	o
2019-03-02	10:36:25,281	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:25 CST	/rest/consumer/v0/hello?name=Alice	200	13	8	5c79ec29db32e07d	org..
2019-03-02	10:36:25,500	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:25 CST	/rest/consumer/v0/hello?name=Alice	200	13	8	5c79ec292a5eb0b6	org..
2019-03-02	10:36:25,638	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:25 CST	/rest/consumer/v0/hello?name=Alice	200	13	8	5c79ec29b55cea60	org..
2019-03-02	10:36:25,778	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:25 CST	/rest/consumer/v0/hello?name=Alice	200	13	8	5c79ec29b847ee22	org..
2019-03-02	10:36:25,922	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:25 CST	/rest/consumer/v0/hello?name=Alice	200	13	8	5c79ec29e9ca2e24	org..
2019-03-02	10:36:26,061	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:26 CST	/rest/consumer/v0/hello?name=Alice	200	13	8	5c79ec2af7117725	org..
2019-03-02	10:36:26,212	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:26 CST	/rest/consumer/v0/hello?name=Alice	200	13	20	5c79ec2a3fed1295	org..
2019-03-02	10:36:26,364	[INFO]	127.0.0.1	- - Sat, 02 Mar 2019 10:36:26 CST	/rest/consumer/v0/hello?name=Alice	200	13	10	5c79ec2a264a6f4c	org..

Day9 CSE实战之服务治理

1 打卡任务

作业：

- 1、在服务治理页面配置provider对consumer服务限流，通过edge分别调用consumer和provider，观察效果

打卡：

- 1、截图consumer调用provider时报429状态码的日志

打卡任务基于Day9的demo项目：



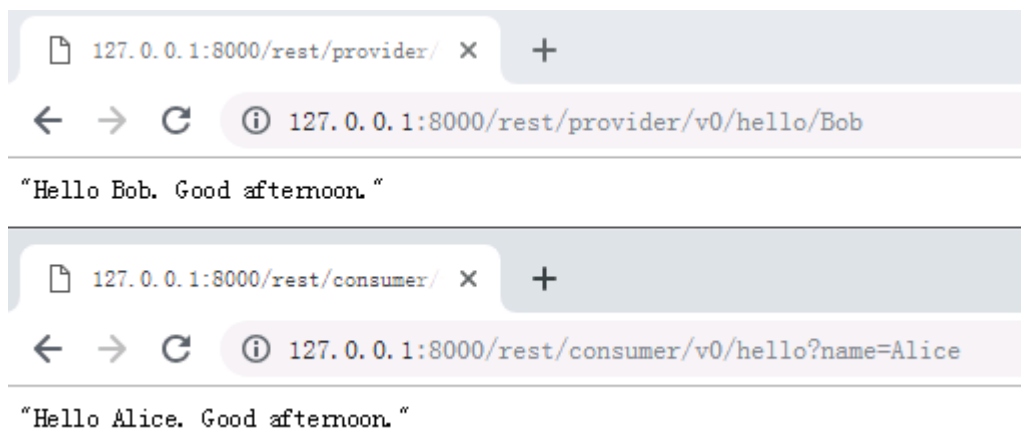
Demo-Day9.zip

2 准备工作

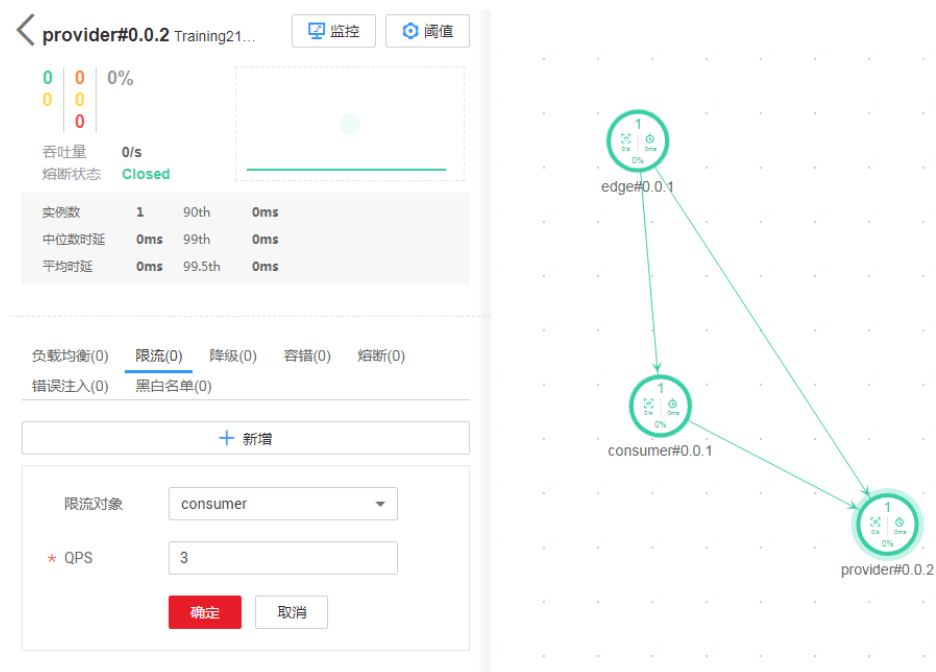
- 1、正常运行Day9的demo

3 体验限流策略

- 1、启动edge、consumer、provider服务，通过edge调用consumer、provider成功



2、在治理页面上配置provider服务对consumer服务限流，限制流量为3QPS



3、通过edge连续调用consumer，观察consumer的日志

```
2019-03-02 15:39:13,396 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 200 30 7 5c7a332190613095
2019-03-02 15:39:13,533 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a332107056834
2019-03-02 15:39:13,672 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 200 30 5 5c7a3321f9513fe5
2019-03-02 15:39:13,838 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 429 41 14 5c7a3321bf6c0586
2019-03-02 15:39:13,977 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a332159743988
2019-03-02 15:39:14,114 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a3322ef14349c
2019-03-02 15:39:14,257 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a33226061ce9c
2019-03-02 15:39:14,403 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a33224186f06b
2019-03-02 15:39:14,533 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a3322bbced913
2019-03-02 15:39:14,686 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 200 30 3 5c7a3322cf40f0b4
2019-03-02 15:39:14,846 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 5 5c7a3322ec944db3
2019-03-02 15:39:14,975 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a33223f0867b8
2019-03-02 15:39:15,103 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a332314cf59b
2019-03-02 15:39:15,270 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a332384fecacc
2019-03-02 15:39:15,424 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 200 30 8 5c7a3323e9128287
2019-03-02 15:39:15,572 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 200 30 7 5c7a332337cb3795
2019-03-02 15:39:15,725 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 200 30 9 5c7a332357ca336c
2019-03-02 15:39:15,857 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a33232a369099
2019-03-02 15:39:15,997 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a33238ec8050a
2019-03-02 15:39:16,163 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a3324d06a84df
2019-03-02 15:39:16,309 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 429 41 5 5c7a3324bb317ab6
2019-03-02 15:39:16,443 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a3324eb2390b8
2019-03-02 15:39:16,613 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 200 30 8 5c7a3324d5bb8e63
2019-03-02 15:39:16,765 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 200 30 9 5c7a3324fef2b6ef
2019-03-02 15:39:16,914 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 429 41 13 5c7a3324fbc88ee1
2019-03-02 15:39:17,070 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:17 CST /consumer/v0/hello?name=Alice 429 41 5 5c7a3325d050a8aa
```

通过日志可以看出每秒钟只能有三次请求成功，其他调用都会返回429 Too Many

Requests错误

← → ↺ ⓘ 127.0.0.1:8000/rest/consumer/v0/hello?name=Alice

["message": "rejected by qps flowcontrol"]

The screenshot shows the Chrome DevTools Network tab. The top toolbar includes buttons for Elements, Console, Sources, Network, Performance, Memory, Application, Security, and Audits. The Network tab is active, showing a list of requests. The first request, 'hello?name=Alice /rest/consumer/v0', is selected. The 'Headers' pane is open, showing the 'General' tab. The status is '429 Too Many Requests'. The request URL is 'http://127.0.0.1:8000/rest/consumer/v0/hello?name=Alice'. The request method is 'GET'. The remote address is '127.0.0.1:8000'. The referrer policy is 'no-referrer-when-downgrade'.

4、通过edge连续调用provider，观察edge的日志

```
2019-03-02 15:46:17,909 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:17 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34c90f96a099
2019-03-02 15:46:18,071 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:18 CST /rest/provider/v0/hello/Bob 200 28 3 5c7a34ca652e58b2
2019-03-02 15:46:18,247 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:18 CST /rest/provider/v0/hello/Bob 200 28 8 5c7a34ca61330330
2019-03-02 15:46:18,417 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:18 CST /rest/provider/v0/hello/Bob 200 28 2 5c7a34ca0474b07f
2019-03-02 15:46:18,608 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:18 CST /rest/provider/v0/hello/Bob 200 28 5 5c7a34ca0af01fa1
2019-03-02 15:46:18,805 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:18 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34cab8e7bba1
2019-03-02 15:46:18,962 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:18 CST /rest/provider/v0/hello/Bob 200 28 8 5c7a34ca71ea23be
2019-03-02 15:46:19,122 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:19 CST /rest/provider/v0/hello/Bob 200 28 3 5c7a34cb34261d77
2019-03-02 15:46:19,307 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:19 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34cbaff34189
2019-03-02 15:46:19,464 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:19 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34cb3029c1ff
2019-03-02 15:46:19,656 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:19 CST /rest/provider/v0/hello/Bob 200 28 5 5c7a34cbd49dacc4
2019-03-02 15:46:19,853 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:19 CST /rest/provider/v0/hello/Bob 200 28 12 5c7a34cb8038972c
2019-03-02 15:46:20,014 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:20 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34ccaf946af7
2019-03-02 15:46:20,217 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:20 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34cc8d2b6fde
2019-03-02 15:46:20,414 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:20 CST /rest/provider/v0/hello/Bob 200 28 7 5c7a34cc801704a2
2019-03-02 15:46:20,588 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:20 CST /rest/provider/v0/hello/Bob 200 28 3 5c7a34cc1604465b
2019-03-02 15:46:20,752 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:20 CST /rest/provider/v0/hello/Bob 200 28 10 5c7a34ccdd1a9251
2019-03-02 15:46:20,920 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:20 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34cc1b219a81
2019-03-02 15:46:21,108 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:21 CST /rest/provider/v0/hello/Bob 200 28 18 5c7a34cd765bb77c
2019-03-02 15:46:21,274 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:21 CST /rest/provider/v0/hello/Bob 200 28 4 5c7a34cd536a0f9d
2019-03-02 15:46:21,457 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:21 CST /rest/provider/v0/hello/Bob 200 28 5 5c7a34cdd82773ea
2019-03-02 15:46:21,639 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:21 CST /rest/provider/v0/hello/Bob 200 28 3 5c7a34cdf2a25777
2019-03-02 15:46:21,809 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:21 CST /rest/provider/v0/hello/Bob 200 28 6 5c7a34cdc697aeb
2019-03-02 15:46:21,970 [INFO] 127.0.0.1 - - Sat, 02 Mar 2019 15:46:21 CST /rest/provider/v0/hello/Bob 200 28 3 5c7a34cde641aa22
```

从以上日志可以看出通过edge直接调用provider时，即使流量大于3QPS也不会触发限流，说明我们在治理页面配置的provider服务限流策略只对consumer生效，不对edge起作用。

4 打卡截图

1、Consumer服务的日志展示的限流策略效果如下图：

```
2019-03-02 15:39:13,396 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 200 30 7 5c7a332190613095
2019-03-02 15:39:13,533 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a332107056834
2019-03-02 15:39:13,672 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 200 30 5 5c7a3321f9513fe5
2019-03-02 15:39:13,838 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 429 41 14 5c7a3321bf6c0586
2019-03-02 15:39:13,977 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:13 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a332159743988
2019-03-02 15:39:14,114 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a3322ef14349c
2019-03-02 15:39:14,257 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a33226061ce9c
2019-03-02 15:39:14,403 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a33224186f06b
2019-03-02 15:39:14,533 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a3322bbced913
2019-03-02 15:39:14,686 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 200 30 3 5c7a3322cf40f0b4
2019-03-02 15:39:14,846 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 5 5c7a3322ec944db3
2019-03-02 15:39:14,975 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:14 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a33223f0867b8
2019-03-02 15:39:15,103 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a332314cfc59b
2019-03-02 15:39:15,270 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a332384fecacc
2019-03-02 15:39:15,424 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 200 30 8 5c7a3323e9128287
2019-03-02 15:39:15,572 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 200 30 7 5c7a332337cb3795
2019-03-02 15:39:15,725 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 200 30 9 5c7a332357ca336c
2019-03-02 15:39:15,857 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 6 5c7a33232a369099
2019-03-02 15:39:15,997 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:15 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a33238ec8050a
2019-03-02 15:39:16,163 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 429 41 4 5c7a3324d06a84df
2019-03-02 15:39:16,309 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 429 41 5 5c7a3324bb317ab6
2019-03-02 15:39:16,443 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 200 30 4 5c7a3324eb2390b8
2019-03-02 15:39:16,613 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 200 30 8 5c7a3324d5bb8e63
2019-03-02 15:39:16,765 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 200 30 9 5c7a3324fef2b6ef
2019-03-02 15:39:16,914 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:16 CST /consumer/v0/hello?name=Alice 429 41 13 5c7a3324fbc88ee1
2019-03-02 15:39:17,070 [INFO] 192.168.0.45 - - Sat, 02 Mar 2019 15:39:17 CST /consumer/v0/hello?name=Alice 429 41 5 5c7a3325d050a8aa
```

Day10 CSE实战之微服务线程模型和性能统计

1 打卡任务

作业：

- 1、开启metrics模块查看服务的性能数据。
- 2、在consumer服务中增加一个reactive的REST方法，通过reactive模式调用provider服务

打卡：

- 1、截图consumer日志中的metrics统计日志
- 2、调用consumer的reactive方法成功并截图

打卡任务基于Day10的demo项目：



2 准备工作

- 1、正常运行Day10的demo

3 开启 Metrics

此处以consumer服务为例，其他服务也是用同样的方法开启metrics

- 1、在pom文件中引入metrics的依赖：


```
<dependencies>
  <dependency>
    <groupId>com.huawei.paas.cse</groupId>
    <artifactId>cse-solution-service-engine</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.servicecomb</groupId>
    <artifactId>metrics-core</artifactId>
  </dependency>
</dependencies>
```

2、在microservice.yaml文件中配置开启metrics日志：

```
cse:
  metrics:
    publisher:
      defaultLog:
        enabled: true # 是否在默认的日志中打印metrics日志
        window_time: 10000 # metrics日志打印周期
```

此处配置的是每10秒钟打印一次metrics日志

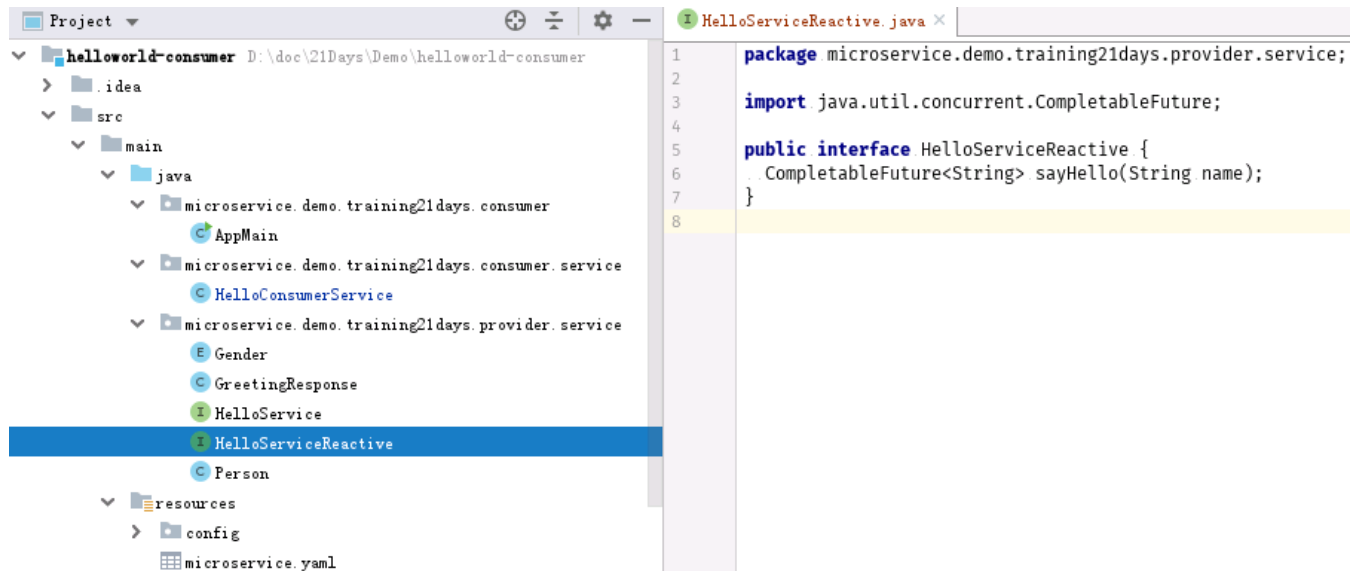
3、运行edge、consumer、provider服务，调用consumer服务，观察日志输出的信息

```
vertx:
  instances:
    name: eventLoopContext-created
    registry: 2
    transport: 10
    monitor-center: 2
    config-center: 2
  transport:
  client.endpoints:
    remote:
      connectCount: 0
      disconnectCount: 0
      connections: 1
      send(Bps): 0
      receive(Bps): 756
    (summary):
      connectCount: 0
      disconnectCount: 0
      connections: 1
      send(Bps): 0
      receive(Bps): 756
    server.endpoints:
      listen:
        connectCount: 0
        disconnectCount: 0
        rejectByLimit: 0
        connections: 1
        send(Bps): 756
        receive(Bps): 0
      (summary):
        connectCount: 0
        disconnectCount: 0
        rejectByLimit: 0
        connections: 1
        send(Bps): 756
        receive(Bps): 0
  threadPool:
    corePoolSize: 8
    maxThreads: 8
    poolSize: 8
    currentThreadsBusy: 0
    queueSize: 0
    taskCount: 0.0
    completedTaskCount: 0.0
    name: cse.executor.groupThreadPool-group1
    corePoolSize: 8
    maxThreads: 8
    poolSize: 8
    currentThreadsBusy: 0
    queueSize: 0
    taskCount: 25.6
    completedTaskCount: 25.6
    name: cse.executor.groupThreadPool-group0
  consumer:
    simple:
      status: tps latency operation
      rest.200: 25 2.429/6.217 provider.hello.sayHello
      25 2.429/6.217 (summary)
    details:
      rest.200:
        provider.hello.sayHello:
          prepare: 0.009/0.021 handlersReq: 0.506/4.238 cFiltersReq: 0.014/0.035 sendReq: 0.398/0.674
          getConnect: 0.224/0.493 writeBuf: 0.174/0.301 waitResp: 1.250/1.746 wakeConsumer: 0.049/0.372
          cFiltersResp: 0.109/0.209 handlersResp: 0.091/0.162
  producer:
    simple:
      status: tps latency operation
      rest.200: 25 3.024/6.802 consumer.helloConsumer.sayHello
      25 3.024/6.802 (summary)
    details:
      rest.200:
        consumer.helloConsumer.sayHello:
          prepare: 0.049/0.206 queue: 0.134/0.204 filtersReq: 0.040/0.082 handlersReq: 0.111/0.222
          execute: 2.532/6.328 handlersResp: 0.019/0.086 filtersResp: 0.084/0.505 sendResp: 0.048/0.195
```

从consumer日志中可以看到如上图所示的metrics信息

4 在 consumer 服务中开发一个 reactive 模式的 REST 方法

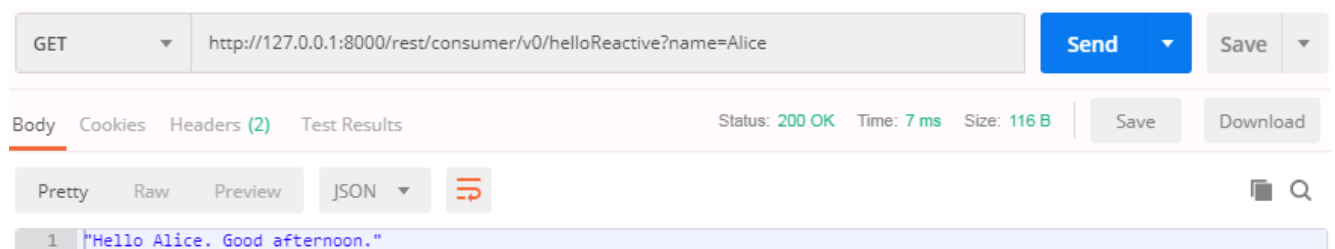
- 1、为了让consumer能够以reactive模式调用provider服务，我们先在consumer服务中定义一个Reactive风格的RPC接口类，供consumer服务发请求时使用



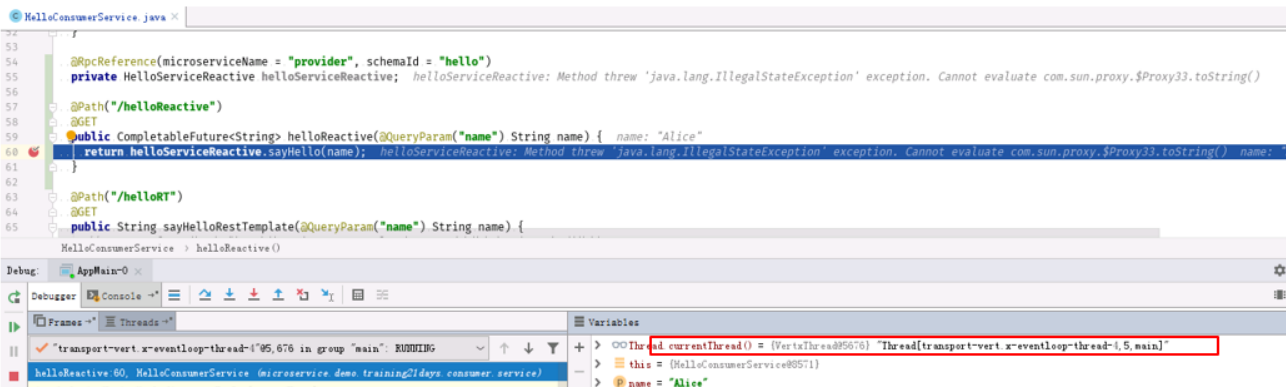
- 2、在HelloConsumerService类中定义reactive风格的REST方法

```
1 @RpcReference(microserviceName = "provider", schemaId = "hello")
2 private HelloServiceReactive helloServiceReactive;
3
4 @Path("/helloReactive")
5 @GET
6 public CompletableFuture<String> helloReactive(@QueryParam("name") String name) {
7     return helloServiceReactive.sayHello(name);
8 }
```

- 3、启动consumer服务，通过edge服务调用consumer的helloReactive方法，调用成功



- 4、如果以debug模式启动consumer服务，在helloReactive方法上打断点，调用consumer的helloReactive方法，可以看到该方法的执行线程是网络线程。即reactive模式的方法默认是执行在网络线程里的。



5 打卡截图

1、Consumer服务的metrics统计日志截图：

```
vertx:
..instances:
... name ..... eventLoopContext-created
... registry 2
... transport 10
... monitor-center 2
... config-center 2
..transport:
... client.endpoints:
... remote ..... connectCount ..... disconnectCount ..... connections ..... send(Bps) ..... receive(Bps)
... (summary) ..... 0 ..... 0 ..... 1 ..... 0 ..... 756
... server.endpoints:
... listen ..... connectCount ..... disconnectCount ..... rejectByLimit ..... connections ..... send(Bps) ..... receive(Bps)
... 0.0.0.0:9090 ..... 0 ..... 0 ..... 0 ..... 1 ..... 756 ..... 0
... (summary) ..... 0 ..... 0 ..... 0 ..... 1 ..... 756 ..... 0
threadPool:
..corePoolSize maxThreads poolSize currentThreadsBusy queueSize taskCount completedTaskCount name
..8 ..... 8 ..... 0 ..... 0 ..... 0 ..... 0.0 ..... 0.0 ..... cse.executor.groupThreadPool-group1
..8 ..... 8 ..... 8 ..... 0 ..... 0 ..... 25.6 ..... 25.6 ..... cse.executor.groupThreadPool-group0
consumer:
..simple:
... status ..... tps ..... latency ..... operation
... rest.200 ..... 25 ..... 2.429/6.217 ..... provider.hello.sayHello
... ..... 25 ..... 2.429/6.217 ..... (summary)
..details:
... rest.200:
... provider.hello.sayHello:
... prepare ..... : 0.009/0.021 ..... handlersReq : 0.506/4.238 ..... cFiltersReq: 0.014/0.035 ..... sendReq ..... : 0.398/0.674
... getConnect ..... : 0.224/0.493 ..... writeBuf ..... : 0.174/0.301 ..... waitResp ..... : 1.250/1.746 ..... wakeConsumer: 0.049/0.372
... cFiltersResp: 0.109/0.209 ..... handlersResp: 0.091/0.162
producer:
..simple:
... status ..... tps ..... latency ..... operation
... rest.200 ..... 25 ..... 3.024/6.802 ..... consumer.helloConsumer.sayHello
... ..... 25 ..... 3.024/6.802 ..... (summary)
..details:
... rest.200:
... consumer.helloConsumer.sayHello:
... prepare: 0.049/0.206 ..... queue ..... : 0.134/0.204 ..... filtersReq : 0.040/0.082 ..... handlersReq: 0.111/0.222
... execute: 2.532/6.328 ..... handlersResp: 0.019/0.086 ..... filtersResp: 0.084/0.505 ..... sendResp ..... : 0.048/0.195
```

2、调用consumer的reactive方法成功的截图：



GET

http://127.0.0.1:8000/rest/consumer/v0/helloReactive?name=Alice

Send

Save

Body

Cookies

Headers (2)

Test Results

Status: 200 OK

Time: 7 ms

Size: 116 B

Save

Download

Pretty

Raw

Preview

JSON

1

"Hello Alice. Good afternoon."

参考答案:



Demo-Day10-Ho
mework.zip

Day11 微服务入门之编写HelloWorld

1 打卡任务

作业：

1. 在provider服务中增加一个greeting方法，接受POST请求，请求参数为request body中传递的Person对象，包含name和gender两个字段。name为String类型；gender为枚举Gender类型，有MALE/FEMALE两个值。返回值为GreetingResponse类型，包含msg和timestamp两个字段，msg为根据gender不同生成的问候语 Hello, Mr.xxx (MALE) / Hello, Ms.xxx (FEMALE)，timestamp为java.util.Date类型的时间戳。
2. 在consumer服务中增加一个greeting方法，接收外部请求的触发，以调用provider服务的greeting方法，并返回provider的应答消息。
 - i. 打卡：
3. 调用provider服务的greeting方法成功，并截图
4. 调用consumer服务的greeting方法成功，并截图

开发基于 Day11 的 demo

2 准备工作

1. 本地成功运行 Day11 的provider、consumer demo工程项目。

3 在 provider 服务中开发 greeting 方法

1. 创建 GreetingResponse 和 Person，以及MALE和FEMALE的声明

```

const Male string = "MALE"
const Female string = "FEMALE"

type GreetingResponse struct {
    Msg string `json:"msg"`
    Timestamp time.Duration `json:"timestamp"`
}

type Person struct {
    Name string `json:"name"`
    Gender string `json:"gender"`
}

```

2, 为 service 创建 Greeting 方法

```

func (*Service) Greeting(ctx *restful.Context) {
    param := Person{}
    ctx.ReadEntity(&param)
    r := GreetingResponse{}
    if param.Gender == Male {
        r.Msg = fmt.Sprintf(format: "Hello, Mr.%s", param.Name)
    } else if param.Gender == Female {
        r.Msg = fmt.Sprintf(format: "Hello, Ms.%s", param.Name)
    }
    r.Timestamp = time.Duration(time.Now().UnixNano() / 1e6)
    ctx.WriteJSON(r, contentType: "application/json")
}

// URLPatterns
func (*Service) URLPatterns() []restful.Route {
    return []restful.Route{
        {Method: http.MethodPost, Path: "/greeting", ResourceFuncName: "Greeting"},
    }
}

```

3, 启动 provider 服务, 调用它的 greeting 接口, 可以看到正常返回结果。

4 在 consumer 服务中开发 greeting 方法

1, 为 consumer 创建 Greeting 方法, 在方法中我们需要将请求中 body 参数传递至 provider

```

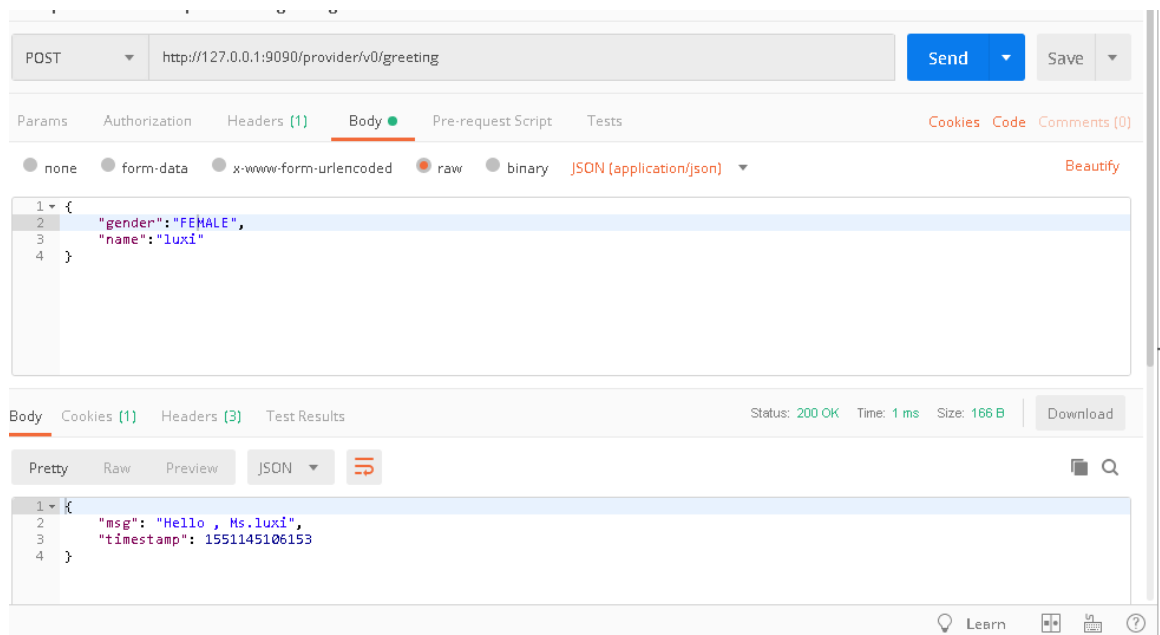
8
9 func (*Client) Greeting(ctx *restful.Context) {
10
11     req, err := rest.NewRequest(http.MethodPost, .. urlStr: "cse://server-demo/greeting", .. body: nil)
12     if err != nil {
13         ctx.WriteError(http.StatusInternalServerError, err)
14         return
15     }
16     req.Body = ctx.ReadRequest().Body
17     resp, err := core.NewRestInvoker().ContextDo(context.TODO(), req)
18     if err != nil {
19         ctx.WriteError(http.StatusInternalServerError, err)
20         return
21     }
22     ctx.Write(httputil.ReadBody(resp))
23 }
24
25 // URLPatterns
26 func (*Client) URLPatterns() []restful.Route {
27     return []restful.Route{
28         {Method: http.MethodPost, Path: "/greeting", ResourceFuncName: "Greeting"},
29     }
30 }
31

```

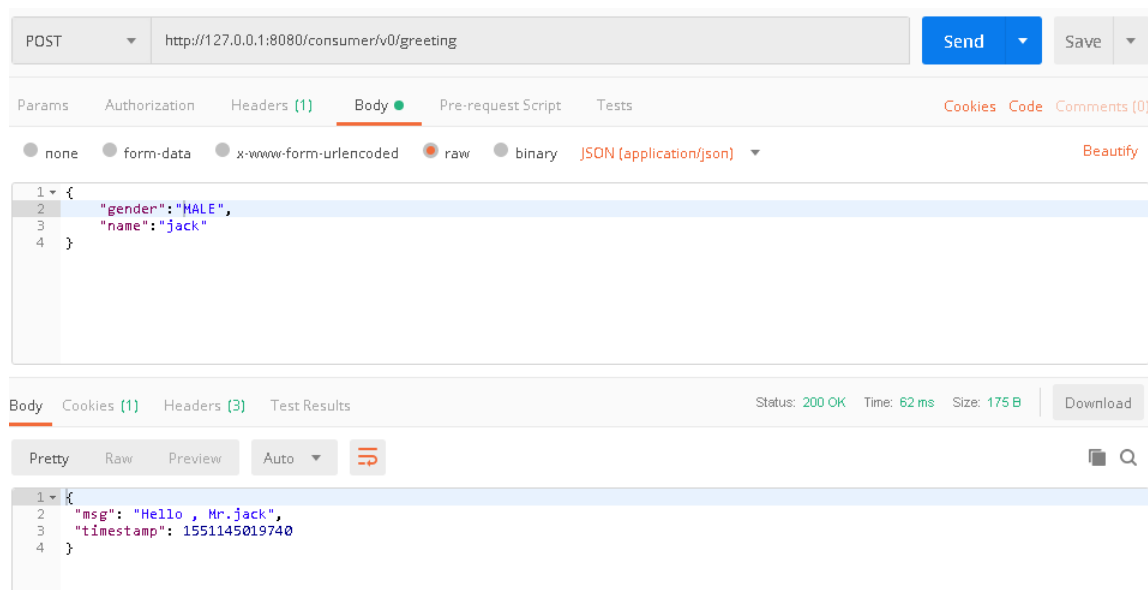
2，启动 consumer 服务，调用它的 greeting 接口，可以看到正常返回结果。

5 打卡截图

1，调用provider服务的结果如下：



2，调用consumer服务结果如下



6 总结

- 1, 在该demo中, 我们仅介绍了CseGoSdk的rest协议的开发, 除了支持rest协议之外同时也支持grpc协议, 如果需要了解更多相关文档请参考, [CseGoSdk文档](#)。
- 2, 在使用过程中如遇到问题, 欢迎在华为云社区进行提问, 你也可以到[开源代码库](#)提issue

Day12 微服务入门之实战

1 打卡任务

作业：

1. 修改consumer端的熔断，容错配置。
2. 调用consumer端方法，测试熔断，并打卡截图
3. 调用consumer端方法，测试容错，并打卡截图

开发基于 Day12 的 demo

2 准备工作

1. 本地成功运行 Day12 的provider、consumer demo工程项目。

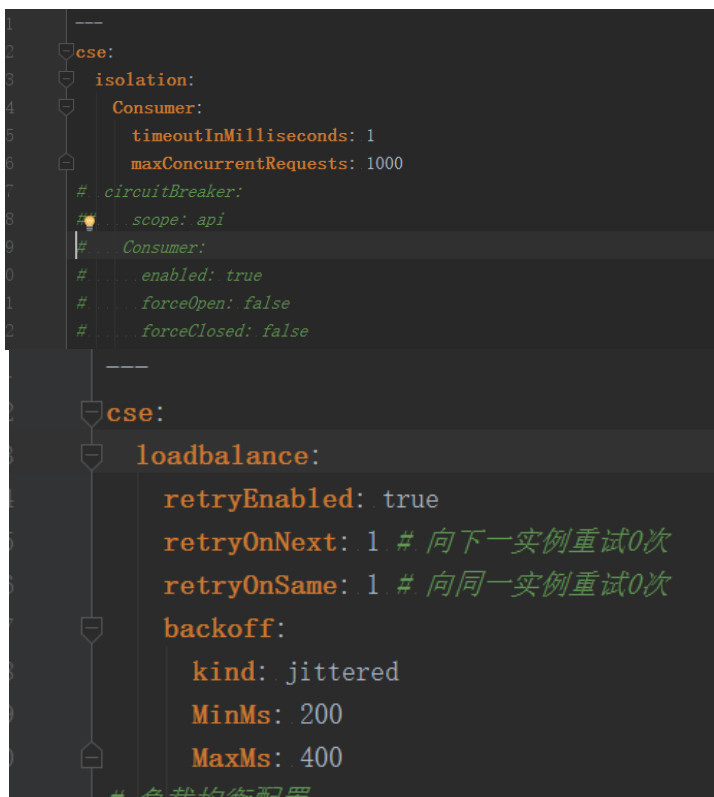
3 在 consumer 服务中修改熔断相关配置

1. 修改consumer的熔断配置。配置要求：请求达到10后计算熔断，熔断在错误率超过30%

开启熔断，参考配置如下：

```
cse:
  isolation:
    Consumer:
      timeoutInMilliseconds: 30
      maxConcurrentRequests: 1000
  circuitBreaker:
    # scope: api
    Consumer:
      enabled: true
      forceOpen: false
      forceClosed: false
      sleepWindowInMilliseconds: 10000
      requestVolumeThreshold: 10 # 请求次数
      errorThresholdPercentage: 30 # 70
```

2. 为容错配置，要求统一服务访问 1 次，非统一服 1 次，并保留超时设置

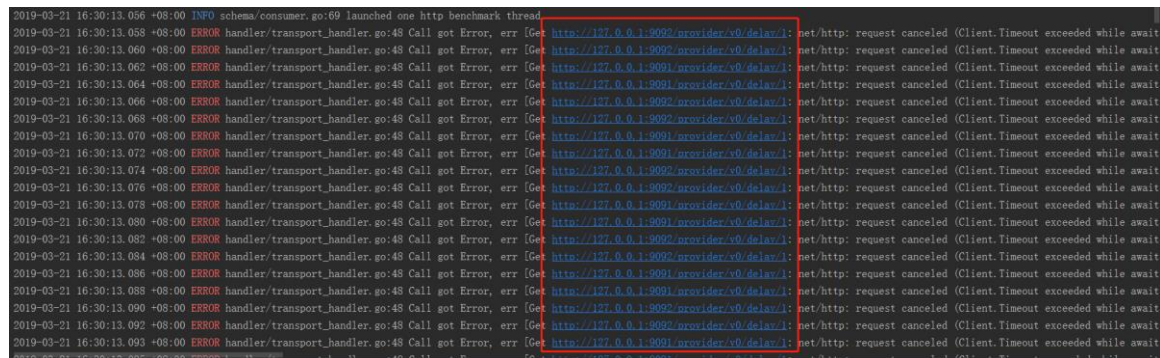


4 打卡截图

1, 容错结果如下:

开启一个线程进程访问, 多个线程难于看出效果:

url: <http://127.0.0.1:8080/consumer/v0/delay?t=1s&delay=1&c=1>



2, 熔断结果如下

GET

http://127.0.0.1:8080/consumer/v0/delay?t=1s&delay=40&c=5

Send

Save

Params

Authorization

Headers (1)

Body

Pre-request Script

Tests

Cookies

Code

Comments (0)

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	t	1s			
<input checked="" type="checkbox"/>	delay	40			
<input checked="" type="checkbox"/>	c	5			
	Key	Value	Description		

Body

Cookies (1)

Headers (3)

Test Results

Status: 200 OK

Time: 1045 ms

Size: 1.87 MB

Save

Download

Pretty

Raw

Preview

JSON

```
36      "Error": "Get http://127.0.0.1:9092/provider/v0/delay/40: net/http: request canceled (Client.Timeout exceeded\n37      },\n38      {\n39      },\n40      "Error": "Get http://127.0.0.1:9092/provider/v0/delay/40: net/http: request canceled (Client.Timeout exceeded\n41      },\n42      {\n43      },\n44      "Error": "API provider:rest:/provider/v0/delay/40 is isolated because of error: circuit open"\n45      },\n46      {\n47      },\n48      "Error": "API provider:rest:/provider/v0/delay/40 is isolated because of error: circuit open"\n49      },\n50      {
```

Learn

Day13 CSE实战之异构技术栈相互调用

1 打卡任务

作业：

1. 运行CSEGoSDK的provider。并调用任意接口查看接口返回(Day11)
2. 运行CSEJavaSDK的consumer，并调用任意接口并查看返回(Day6)。
 - i. 打卡：
3. 调用consumer服务的方法成功，并截图

开发基于 Day11 和 Day6 的 demo

2 准备工作

1. 本地成功运行 Day6 的provider、consumer demo工程项目。
2. 本地成功运行 Day11 的provider、consumer demo工程项目。

3 运行 CSEGoSDK 的 provider 端服务

1. 请按照Day11的课程进行启动

4 运行 CSEJavaSDK 的 consumer 服务

2. 请按照Day6的课程进行启动

5 打卡截图

- 1, 调用edge服务的结果以及日志如下：

```
11:53:35.941 +08:00 DEBUG configcenter-client/configcenter_client.go:338
11:53:35.941 +08:00 DEBUG configcenter-client/configcenter_client.go:256
11:53:35.993 +08:00 DEBUG configcenter-source/configcentersource.go:253 e
11:53:36.258 +08:00 INFO schema/service.go:28 access provider hello
11:53:43.008 +08:00 DEBUG servicecenter/servicecenter.go:173 Heartbeat su
```

http://192.168.0.45:8000/rest/consumer/v0/hello?name=Alice

GET

http://192.168.0.45:8000/rest/consumer/v0/hello?name=Alice

Params

Authorization

Headers

Body

Pre-request Script

Tests

	KEY	VALUE
<input checked="" type="checkbox"/>	name	Alice
	Key	Value

Body

Cookies

Headers (3)

Test Results

Pretty

Raw

Preview

JSON

1

"hello , Alice"

Day14-CSE实战之其他服务何如接入之

helloworld

1 打卡任务

作业：

1. 在springboot服务中增加两个个greeting方法，分别为providerGreeting和consumerGreeting接受POST请求，请求参数为request body中传递的Person对象，包含name和gender两个字段。name为String类型；gender为枚举Gender类型，有MALE/FEMALE两个值。返回值为GreetingResponse类型，包含msg和timestamp两个字段，msg为根据gender不同生成的问候语 Hello, Mr.xxx (MALE) / Hello, Ms.xxx (FEMALE)，timestamp为java.util.Date类型的时间戳。

打卡：

2. 调用provider服务的greeting方法成功，并截图
3. 调用consumer服务的greeting方法成功，并截图

开发基于Day14 demo进行开发

2 准备任务

1. 能正常运行Day14的provider和consumer

3 为 springboot 服务增加 greeting 方法

1. 创建作为请求的Person类、作为返回值的GreetingResponse类以及Gender枚举类型

```

public class GreetingResponse {
    private String msg;
    private Date timestamp;

    public String getMsg() { return msg; }

    public void setMsg(String msg) { this.msg = msg; }

    public Date getTimestamp() { return timestamp; }

    public void setTimestamp(Date timestamp) { this.timestamp = timestamp; }
}

```

```

public enum Gender {
    MALE,
    FEMALE
}

```

```

2
3 public class person {
4     private Gender gender;
5     private String name;
6
7
8     public Gender getGender() { return gender; }
9     public void setGender(Gender gender) { this.gender = gender; }
0     public String getName() { return name; }
1     public void setName(String name) { this.name = name; }
2

```

2. helloController中创建consumerGreeting和providerGreeting方法

```

@RestController
@RequestMapping("/consumer/v0")
public class ConsumerController {
    @RequestMapping("/hello/{name}")
    public String ProviderHello(@PathVariable(value = "name") String name) {
        RestTemplate restTemplate = new RestTemplate();
        String s = restTemplate.getForObject(url: "http://provider/provider/v0/hello/" + name, String.class);
        return s;
    }

    @RequestMapping("/greeting")
    public GreetingResponse ConsumerGreeting(@RequestBody Person p) {
        RestTemplate restTemplate = new RestTemplate();
        GreetingResponse response = restTemplate.postForObject(url: "http://provider/provider/v0/greeting", p, GreetingResponse.class);
        return response;
    }
}

```

3. 运行provider服务并进行调用

运行springboot服务后，访问/provider/v0/greeting接口

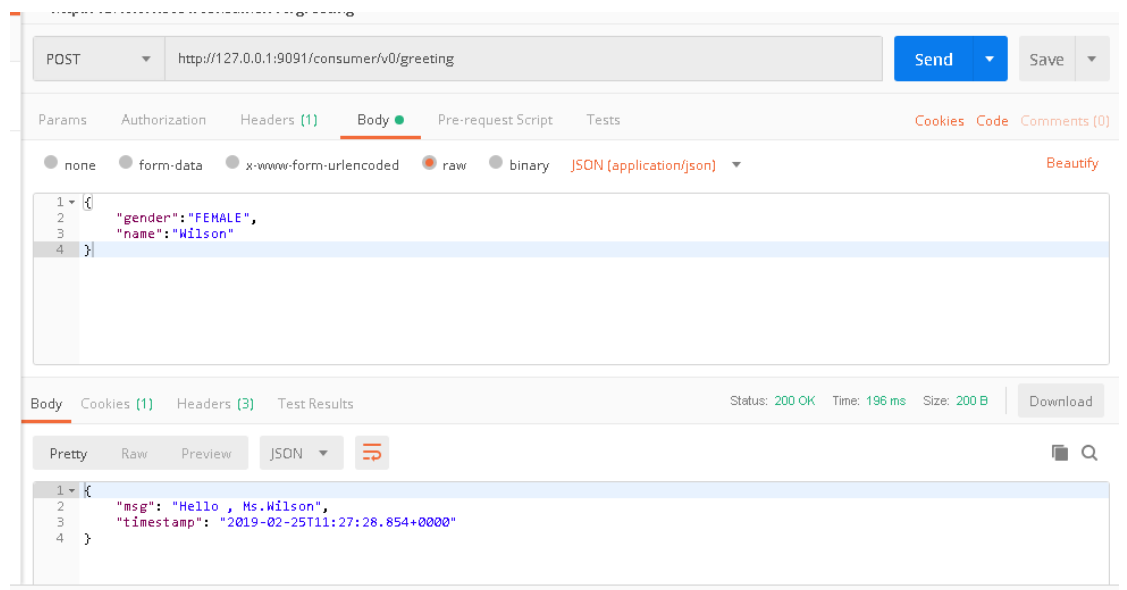
提示 :请启动 Day11 的 consumer 和 provider ,在 Day11demo 中 ,springboot 的 consumer

访问 Day11 的 provider , Day11 的 consumer 访问 springboot 的 provider 接口

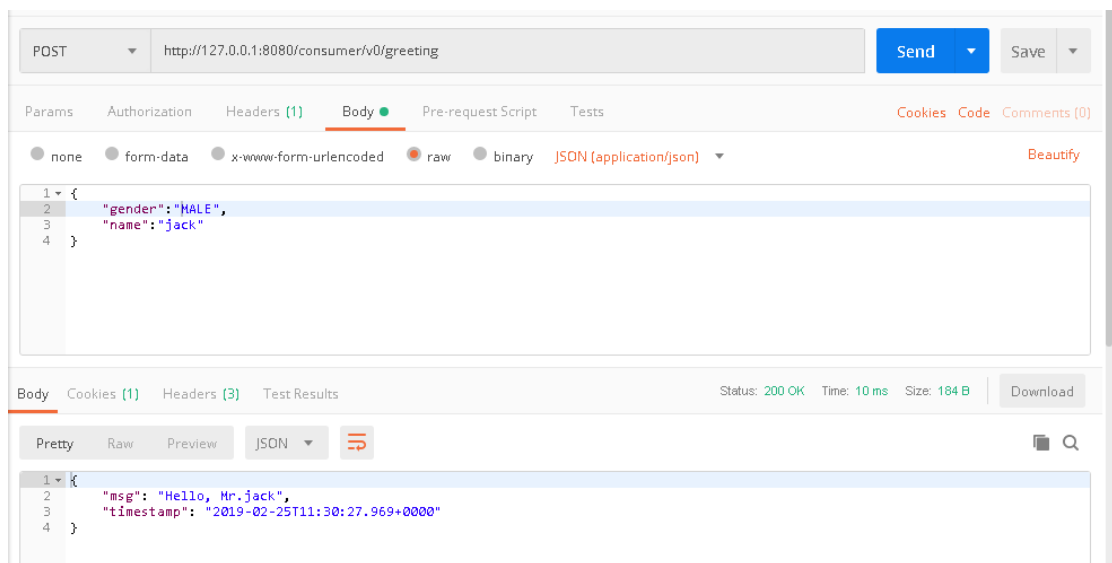
4 打卡截图

在 Day11 和修改后的 spring 全部启动后

1. 访问springboot的consumer接口



2. 访问Day11中的consumer端接口



5 总结

1. 在springboot时，需要在springboot中设置http_proxy。
2. 启动mesher前，我们需要将mesher的监听地址修改为运行机器的外部地址(非127.0.0.1)，如外部地址为：192.168.0.1，这里不能使用127.0.0.1
3. 在接入cse后，通过服务名进行访问非直接使用ip+port访问

DAY15 微服务云应用平台介绍

1 任务介绍

本日对微服务云应用平台进行了整体介绍，为考量学习效果，请大家完成问答题。

2 任务执行

2.1 点击链接加入课堂：

<https://classroom.devcloud.huaweicloud.com/diploma/joinclass/41d76a6fea454095964d1cf220f94adb>



2.2 点击作业




点击进入作业



点击文件查看问答题目

Class Room
Powered by DevCloud

←


DAY15 微服务云应用... 未提交
所属课时：Day 15

提交作业 留言

> 通用习题

DAY15 微服务云应用平台介绍.d...

Class Room
Powered by DevCloud

我的工作台 资源市场

① 华北区1 Hi zoeey2017

←

DAY15 微服务云应用... 未提交

截止日期
2019-03-31 23:59:59

评分
暂未评分

评语
暂未点评

提交作业 留言

> 通用习题

DAY15 微服务云应用平台介绍.d...

DAY15 微服务云应用平台介绍.docx

HUAWEI 文档名称 文档密级

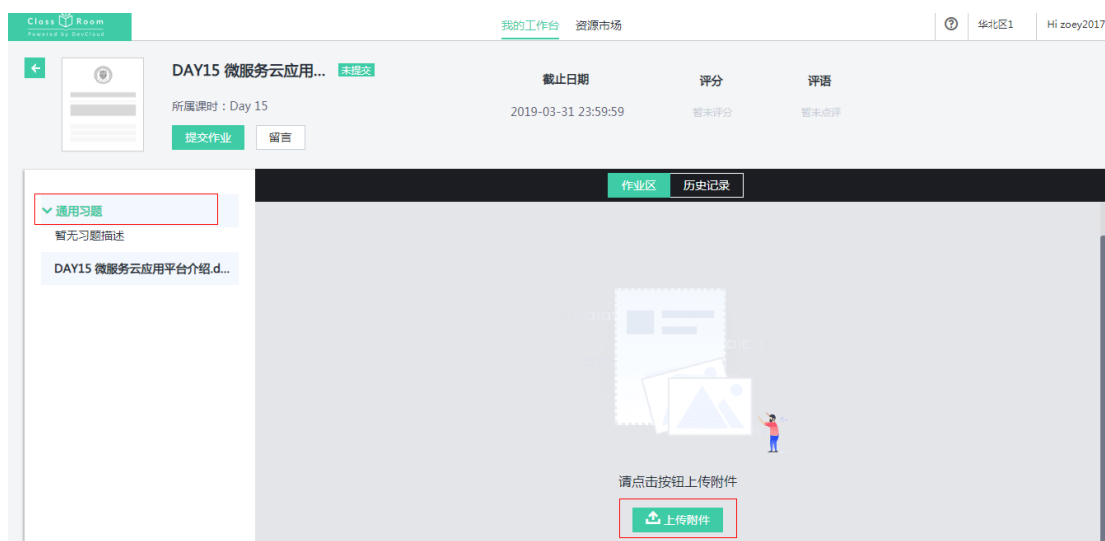
判断题

在每一道题目的括号内用√（表示正确）或×（表示错误）来标识答案。

- ServiceStage 提供了完备的 Service Mesh 商业版本。（ ）
- ServiceStage 支持 ServiceComb 商业版，但不支持 Spring Cloud 商业版。（ ）
- ServiceStage 提供的微服务治理能力包括但不限于服务注册、发现、通信、服务

2.3 本地新建一份Word文档，将答案写于文档中。

点击通用习题，点击上传附件，将答案上传。



点击提交作业并确定，即可完成作业提交



2.4 将已提交的结果进行截图



3 打卡任务

将以下页面截图进行打卡



DAY16 微服务应用开发之持续交付

1 打卡任务

作业：

基于模板创建云上工程，并访问应用接口；更新代码后，执行流水线，访问新版本应用接口。

打卡：

访问新版应用接口，并截图。

2 准备工作

- 1、切换区域为“华东-上海二”。
- 2、确认选择的ServiceStage版本为“基础版”（可免费使用10个实例，**超过10个实例或选择的是专业版则会按小时进行计费**）。
- 3、已拥有DevCloud账号。
- 4、已拥有可运行的CCE集群、某一节点绑定弹性IP。

3 创建云上工程

本步骤可查阅ServiceStage用户指南，https://support.huaweicloud.com/usermanual-servicestage/servicestage_user_0011.html。

- 1、登录ServiceStage，选择 持续交付 > 工程。
- 2、单击“创建云上工程”，单击“基于模板创建”下的“创建云上工程”。
- 3、选择语言及框架
本次实践选择语言“Java 8”，选择框架“CSE-Java（SpringMVC）”。
- 4、选择部署系统
选择“云容器引擎CCE”
- 5、配置工程参数。

a. 输入项目名称。

b. 配置一个源码仓库

本次实践配置DevCloud“源码仓库”参数。

i. 在“代码源来源”中选择DevCloud，单击“绑定帐户”，输入密码(该密码是DevCloud仓库的密码)，获取DevCloud源码仓库授权。

ii. 选择默认命名空间Demo，并输入源码仓库名称：hellocloud。

c. 选择仓库组织（如果没有则通过“创建仓库组织”进行创建，创建成功后返回刷新进行选择即可）。

d. 选择部署集群：云容器引擎CCE。

e. 高级设置

i. 设置访问方式

勾选“把应用发布到主机网络上，用于website、LB等开放页面，接口给外部访问的场景。”

ii. 添加服务

1)设置服务名称

输入“hellocloud”

2)设置访问方式

“访问方式”选择“公网访问”

“访问类型”选择“弹性IP”

3)设置端口映射

容器端口输入“8080”

访问端口“30069”

* 项目名称

* 代码来源

DevCloud

GitHub

Gitee

Bitbucket

GitLab

已绑定账户:

命名空间 仓库名称

* 仓库组织 [创建仓库组织](#)

* 部署集群 [创建集群](#)

* 集群命名空间 [创建命名空间](#)

* 实例个数

高级设置 (资源配额, 设置访问方式) ^

资源配额

设置访问方式 ☒ 把应用发布到主机网络上,用于website、LB等开放页面,接口给外部访问的场景。

[添加服务](#)

内部域名访问地址	访问方式	容器端口	访问端口	协议	操作
hellocloud.default.svc.cluster.local:30069	公网访问>弹性IP:49.4.4.250	8080	30069	TCP	删除

- 6、单击“部署”，返回“持续交付 > 工程”查看已创建工程清单中查看刚创建的工程，“状态”栏显示正常图标，表示工程创建成功。

您可以创建基于微服务SDK的微服务工程并下载，快速进入业务代码开发阶段。 [如何创建微服务？](#)

云上工程 本地工程

[+ 创建云上工程](#)

名称	状态	创建时间	描述	操作
project-w6x7	正常	2019/03/02 17:11:08 GMT+08:00		源码地址 跳转到流水线 删除工程

4 访问云上工程应用

- 1、选择 持续交付 > 工程。
- 2、选择 [步骤3](#)创建的云上工程项目，单击“跳转到流水线”。
- 3、等待流水线运行完成。

微服务云应用平台

流水线: 流水线详情

项目-w6x7

状态: 成功

创建时间: 2019/03/27 09:44:49 GMT+08:00

最近一次执行时间: 2019/03/27 09:44:49 GMT+08:00

流水线: 流水线详情

1 基本信息

流水线名称: project-w6x7

描述:

2 Build

build

构建系统 Assembling

成功 详情 2分钟前

3 Deploy

deploy

部署系统 CFE

成功 详情 刚刚

- 4、选择 应用管理 > 应用列表，找到对应的应用，查看状态为“运行中”，如下所示：

应用名称	类型	状态	部署系统	外部访问地址	实例总数	创建时间	操作
project-w6x7	容器应用	运行中	容器引擎 CCE	49.4.4.250:30069	1	2019/03/02 17:42:49 GMT+08:00	编辑 升级 更多

通过chrome浏览器访问应用接口（将下面的IP地址替换为上图中对应的外部访问

地址）：<http://49.4.4.250:30069/rest/helloworld?name=apply>

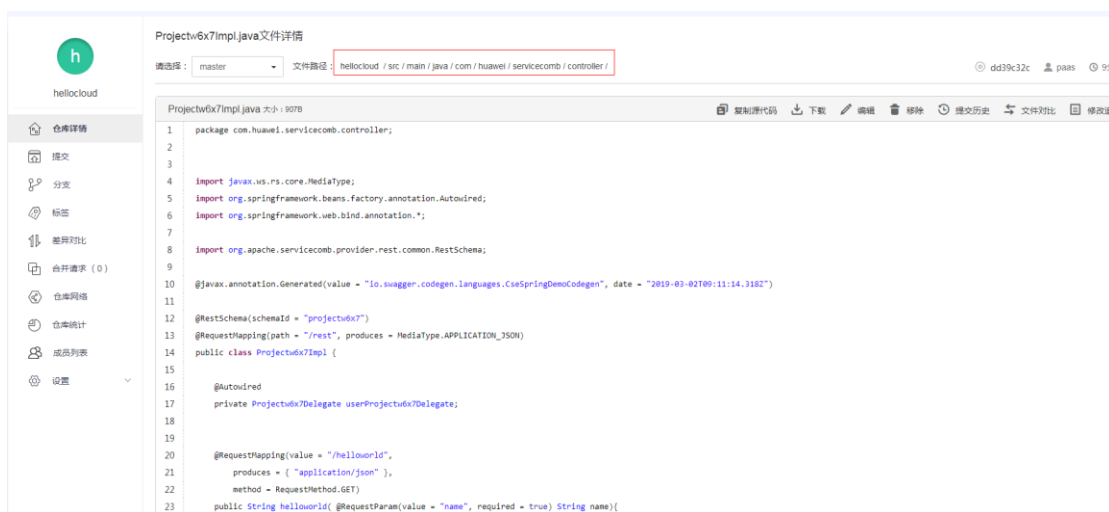
返回如下结果：

← → ↻ ⓘ 不安全 | 49.4.4.250:30069/rest/helloworld?name=apply

“apply”

5 通过流水线升级应用

- 1、登录ServiceStage，选择 持续交付 > 工程。
- 2、选择步骤3创建的云上工程项目，单击“源码地址”，跳转到DevCloud源码仓库，“服务”-->“代码托管”，找到刚刚的代码仓库 Demo / hellocloud。
- 3、修改应用接口返回内容，



在hellocloud/src/main/java/com/huawei/servicecomb/controller/Projectw6x7Impl.java文件（注：名称Projectw6x7Impl会根据您输入的工程名有所不同）中将：

```
public class Projectw6x7Impl {

    @Autowired
    private Projectw6x7Delegate userProjectw6x7Delegate;

    @RequestMapping(value = "/helloworld",
        produces = { "application/json" },
        method = RequestMethod.GET)
    public String helloworld( @RequestParam(value = "name", required = true) String name){

        return userProjectw6x7Delegate.helloworld(name);
    }

}
```

中的 userProjectw6x7Delegate.helloworld(name) 在线编辑修改为：

userProjectw6x7Delegate.helloworld("HI: " + name);

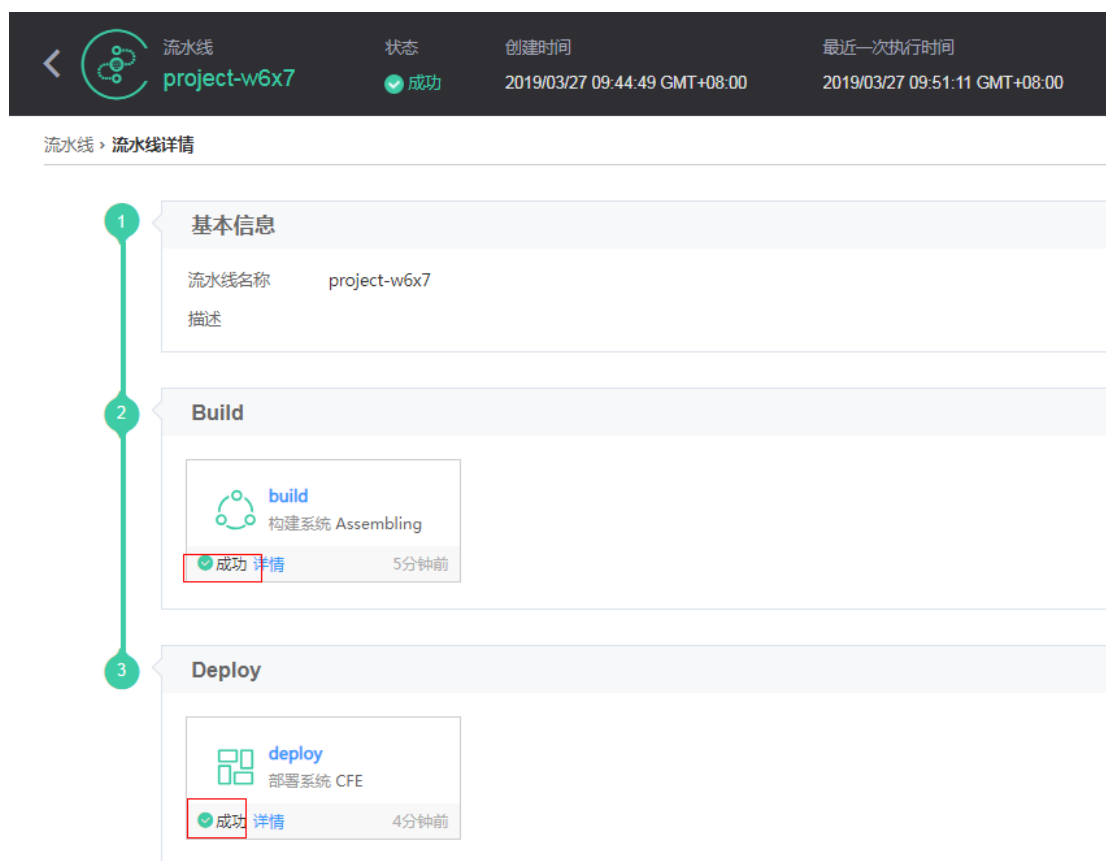
然后保存提交。

```

11
12 @RestSchema(schemaId = "projectw6x7")
13 @RequestMapping(path = "/rest", produces = MediaType.APPLICATION_JSON)
14 public class Projectw6x7Impl {
15
16     @Autowired
17     private Projectw6x7Delegate userProjectw6x7Delegate;
18
19
20     @RequestMapping(value = "/helloworld",
21         produces = { "application/json" },
22         method = RequestMethod.GET)
23     public String helloworld( @RequestParam(value = "name", required = true) String name){
24
25         return userProjectw6x7Delegate.helloworld("HI: " + name);
26     }
27
28 }

```

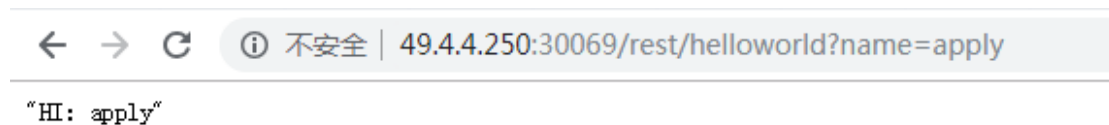
- 4、修改后在ServiceStage界面选择 持续交付 > 发布，选择上述流水线，单击“启动”



- 5、等待流水线运行完成。
- 6、点击 应用管理 > 应用列表，找到对应的应用，通过chrome浏览器访问应用接口（将下面的IP地址替换为应用实际对应的外部访问地址）：

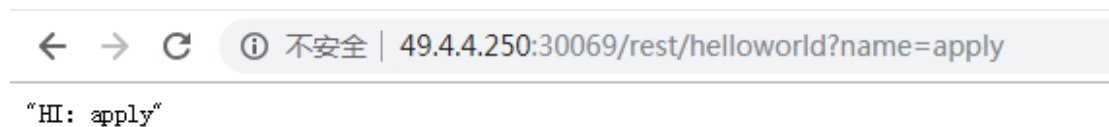
<http://49.4.4.250:30069/rest/helloworld?name=apply>

返回如下结果:



6 打卡截图

- 1、通过浏览器访问应用接口: <http://49.4.4.250:30069/rest/helloworld?name=apply>



DAY17 微服务应用开发之本地工具

1 打卡任务(Eclipse ServiceStage 插件)

作业：

使用Eclipse的ServiceStage开发插件，将本地的项目部署到ServiceStage公有云上，并能访问成功。

打卡：

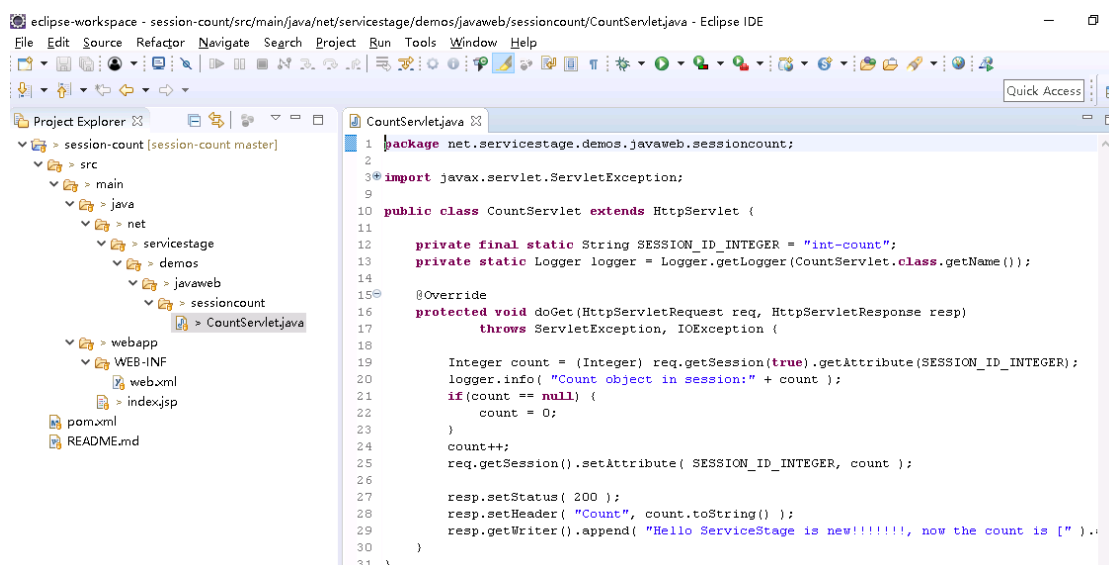
Eclipse插件配置正确且项目部署成功并能访问。

2 准备工作

- 1、切换区域为“华东-上海二”。
- 2、确认选择的ServiceStage版本为“基础版”（可免费使用10个实例，**超过10个实例或选择的是专业版则会按小时进行计费**）。
- 3、已拥有可运行的CCE集群，并购买了负载均衡ELB，绑定了弹性IP。
- 4、已安装eclipse。
- 5、已有能编译成jar/war的简单demo（注意：目前该插件暂只支持Tomcat和Node.js应用），或直接使用示例: <https://github.com/StoneComes/session-count>。

3 操作步骤

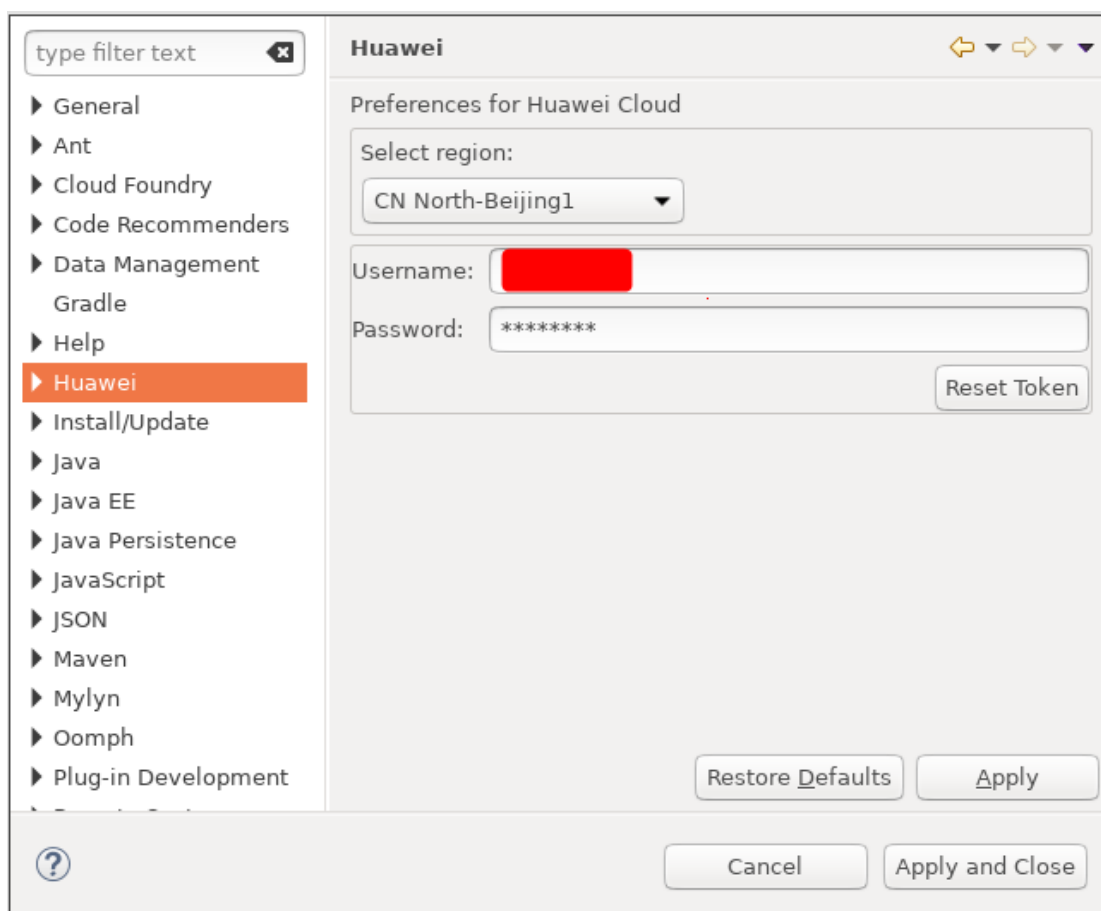
- 1、下载Eclipse ServiceStage插件，下载链接: <https://cse-bucket.obs.myhwclouds.com/Plugins/servicestage-eclipse-plugin-1.0.jar>
- 2、将该jar包放在你eclipse的安装目录下的\${ECLIPSE_INSTALLATION}/dropins，重启eclipse以使其生效。
- 3、在eclipse中导入已准备好的demo（例如: <https://github.com/StoneComes/session-count>），如下图：



4、配置华为Preferences（Window > Preferences> Huawei）

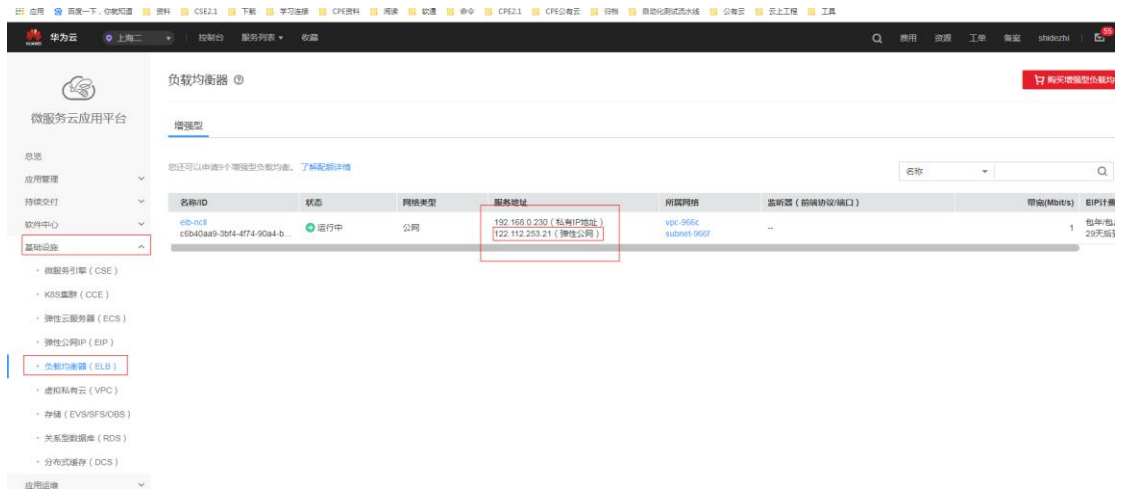
a.选择区域为 CN East-Shanghai2。

b.输入用户名和密码：

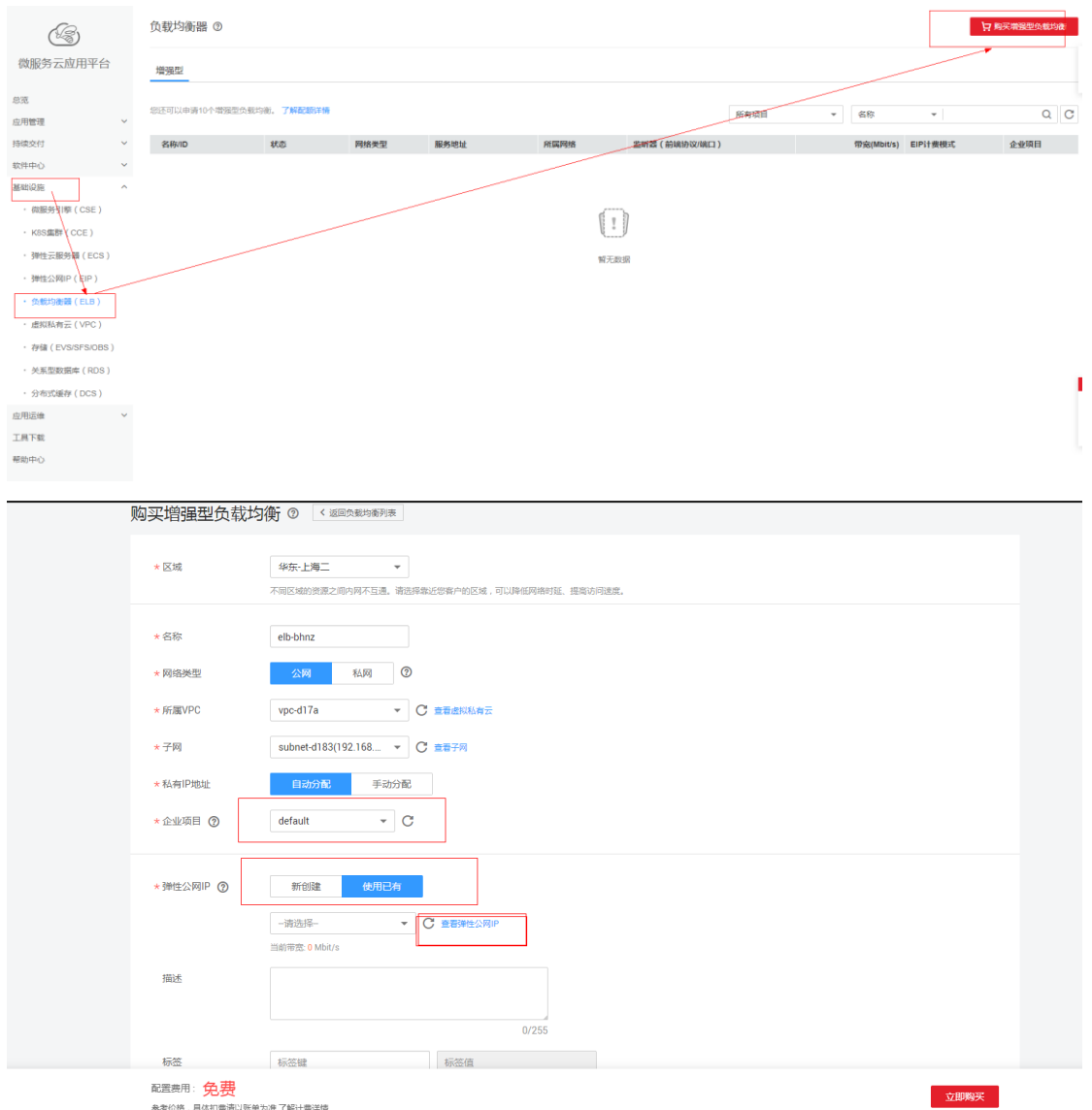


5、登录servicestage页面，查看自己的账户是否已经有ELB，且绑定了弹性IP（如有则

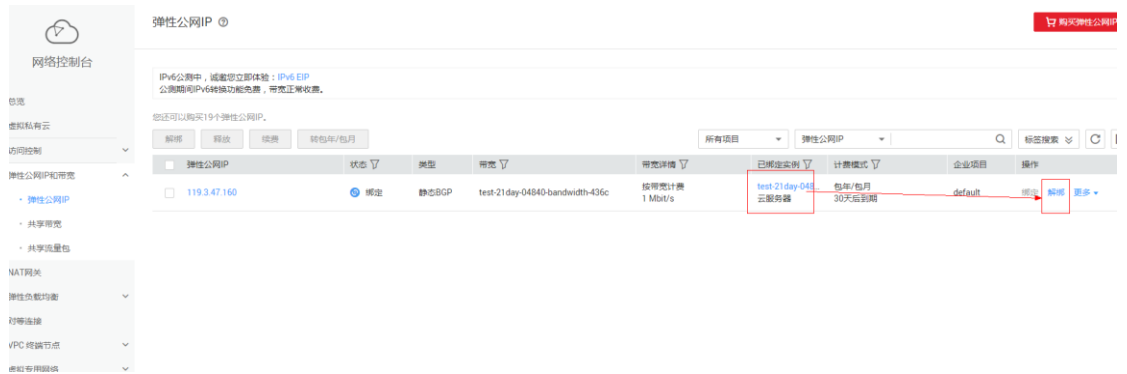
跳过该步骤，直接到步骤6)



如果没有则购买增强型负载均衡，如下图所示进行购买：



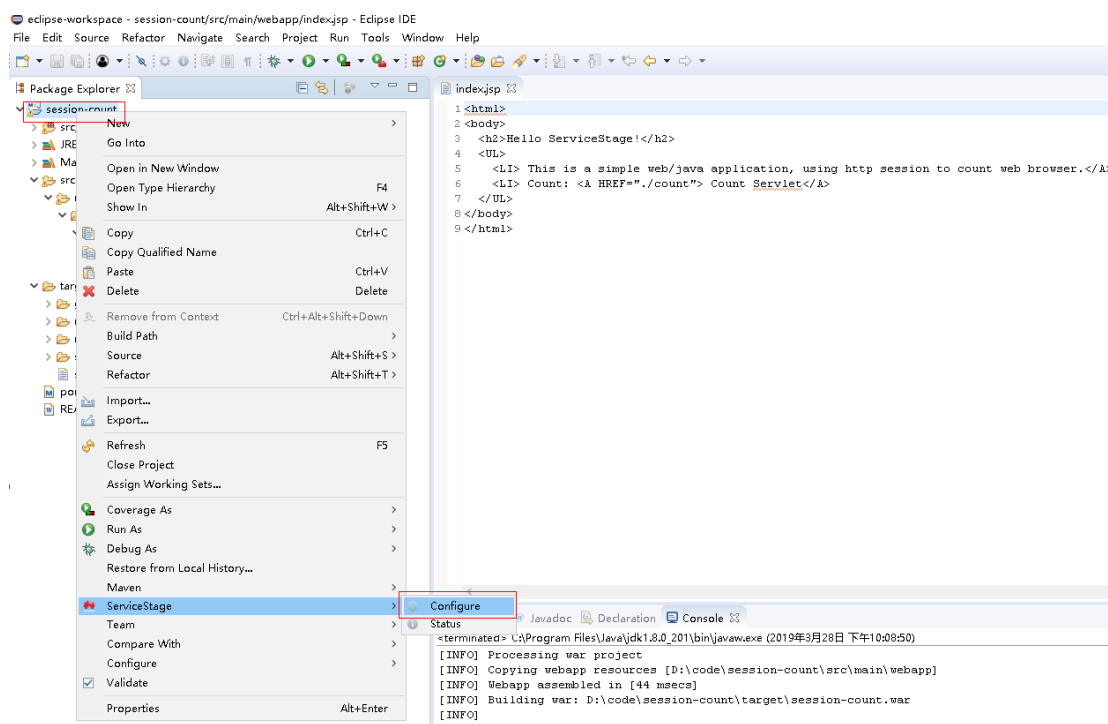
点击“查看弹性公网IP”，然后解绑掉当前不用的弹性IP：



解绑成功后，返回到刚才购买页面进行增强型负载均衡的购买：




6、右键单击“项目”选中，选择“ServiceStage> Configure”，如图下图所示：



- 填写应用信息（注意：Application Name只能是小写和数字，且以小写开头）。
- 填写用于存储待上传部署文件的SWR仓库信息。
- 选择集群ID、ELB ID、VPC ID等平台信息。
- 选择应用所需的服务信息。
- 单击“Finish”创建或更新项目的servicestage.xml文件。

注意：下图中CCE集群和ELB要是同一个VPC下的，三者要相匹配，相当于是同一个网络下的，如果没有ELB，则不用填下图中ELB对应的值。



Describe your application

Describe your application that will be deployed to ServiceStage

Service Instance

ID:*

64c08696-0d80-4c61-b7e2-38f29c4f194d

Re...

Application

Name:*

test-eclipse

Display Name:*

test-eclipse

Description:

test-eclipse

Version:*

1.0

Type:*

Tomcat 8

Category:*

Webapp

Port:*

8080

SWR Upload Info

Repository:*

test-eclipse

Platform Information

CCE Cluster:*

test-eclipse

CCE Namespace:*

default

ELB:*

elb-zz01

VPC:*

vpc-test

Subnet:*

192.168.20.0/24

?

< Back

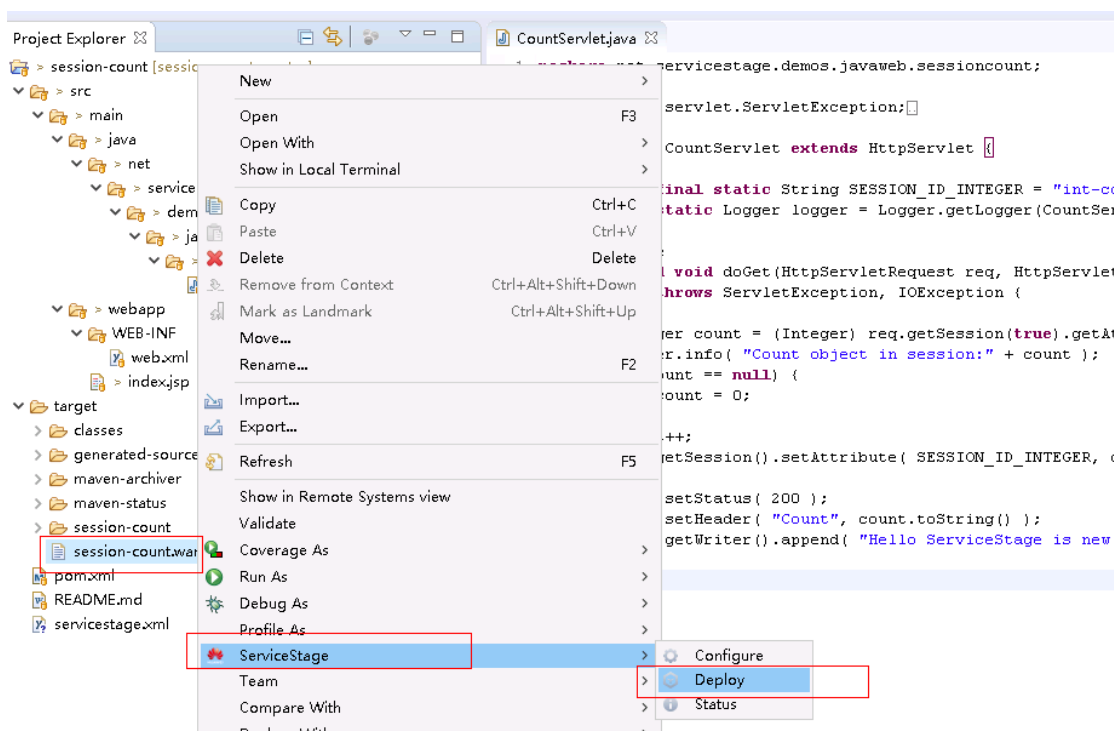
Next >

Finish

Cancel

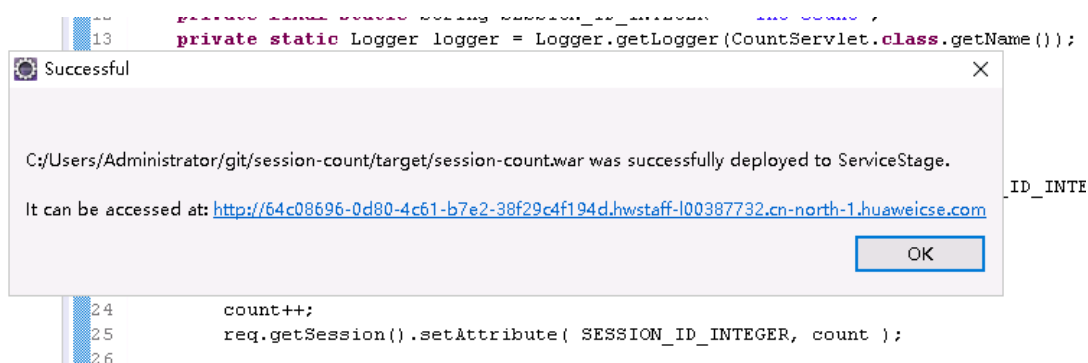
7、将该项目部署起来

- 如有必要，构建项目，生成war文件。(示例工程演示的是java的demo，所以先编译，编译后会在target目录下产生一个war文件)
- 在项目窗口，右键单击war文件或Node.js项目，选择“ServiceStage > Deploy”。



Eclipse的右下角会有部署进程并显示进程信息。

- 8、如果部署成功，最后会在eclipse弹出类似如下的提示框：



- 9、可以在ServiceStage云应用平台，点击 应用管理 > 应用列表，查看刚才部署成功的应用：



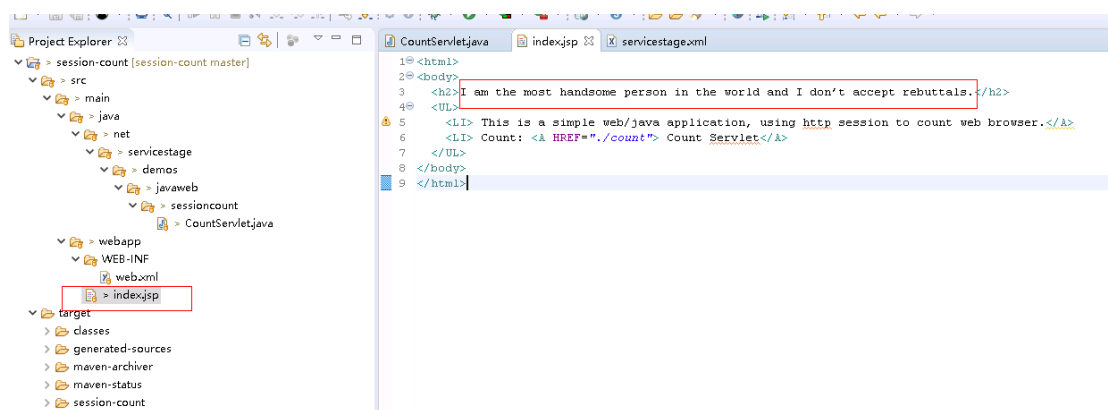
10、 点击上图中外部访问地址，测试是否访问成功。

Hello ServiceStage update!

- This is a simple web/java application, using http session to count web browser.
- Count: [Count Servlet](#)

11、 验证更新代码，更新应用是否会成功。

a. 修改代码



b. 重新编译打包

c. 重复步骤6~9的操作，进行应用更新，然后在ServiceStage界面可以查看到应用正在更新：



d. 等待更新完，再次访问该应用



成功访问到

I am the most handsome person in the world and I don't accept rebuttals.

- This is a simple web/java application, using http session to count web browser.
- Count: [Count Servlet](#)

4 打卡截图

截图1:

Describe your application

Describe your application that will be deployed to ServiceStage

Service Instance

ID:*

64c08696-0d80-4c61-b7e2-38f29c4f194d

Refresh

Application

Name:*

test-eclipse

Display Name:*

test-eclipse

Description:

test-eclipse

Version:*

1.0

Type:*

Tomcat 8

Category:*

Webapp

Port:*

8080

SWR Upload Info

Repository:*

test-eclipse

Platform Information

CCE Cluster:*

test-eclipse

CCE Namespace:*

default

ELB:*

elb-zz01

VPC:*

vpc-test

Subnet:*

192.168.20.0/24

?

< Back

Next >

Finish

Cancel

截图二:

微服务云应用平台

应用管理

应用列表

应用配置

持续交付

软件中心

资源中心

您可以创建自己的应用，并且可以根据需要对应用进行部署、配置、监控、扩容、升级、卸载、服务发现、负载均衡等操作。 [如何创建应用？](#)

+ 创建应用

应用

集群: test-demo

全部状态 (1)

请输入应用名称

Q

+

应用名称	类型	状态	部署系统	外部访问地址	实例总数	创建时间	操作
servicestage408mm	Tomcat	运行中	云容器引擎CCE		1	2019/03/27 10:05:10 GMT+08:00	伸缩 升级 更多

截图三:



I am the most handsome person in the world and I donâ€™t accept rebuttals.

- This is a simple web/java application, using http session to count web browser.
- Count: [Count Servlet](#)

DAY18 微服务应用实战之快速上线

1 打卡任务

作业：

基于模板创建一个ServiceComb微服务应用。本作业操作步骤较多，请按操作指导一步步执行。

打卡：

访问创建的微服务接口成功并截图。

2 准备工作

- 1、切换区域为“华东-上海二”。
- 2、确认选择的ServiceStage版本为“基础版”（可免费使用10个实例，超过10个实例或选择的是专业版则会按小时进行计费）。
- 3、已拥有DevCloud账号和密码。
- 4、已有CCE集群，集群有1个节点绑定弹性IP。

3 基于模板创建 ServiceComb 微服务应用

- 1、登录ServiceStage，点击 总览 > 快捷操作 > 创建ServiceComb应用。
- 2、运行环境选择Java8，输入应用名称和应用显示名称hello，应用来源选择“模板”，代码源来源选择DevCloud，输入密码进行授权（如已授权则这一步可省略）。

创建应用

返回应用管理

1 基本信息

2 应用配置

3 规格确认

应用类型

ServiceComb

更换

运行环境

Tomcat8

Java8

应用名称

hello

应用显示名称

hello

描述

0/100

应用来源

Jar包

源代码仓库

模板

代码源来源

DevCloud

GitHub

Gitee

Bitbucket

GitLab

租户名

用户名

HTTPS密码

如何获取HTTPS密码, 请查看

详情

确定

取消

3、选择生成代码归档的命名空间，输入归档的仓库名称，如：hello。

代码源来源

DevCloud

GitHub

Gitee

Bitbucket

GitLab

已绑定账户:

命名空间

Demo

仓库名称

hello

框架

ServiceComb (SpringMVC)

2.3.56

基于ServiceComb框架, 支持SpringM...

ServiceComb (JAX-RS)

2.3.56

基于ServiceComb框架, 支持JAX-RS...

ServiceComb (POJO)

2.3.56

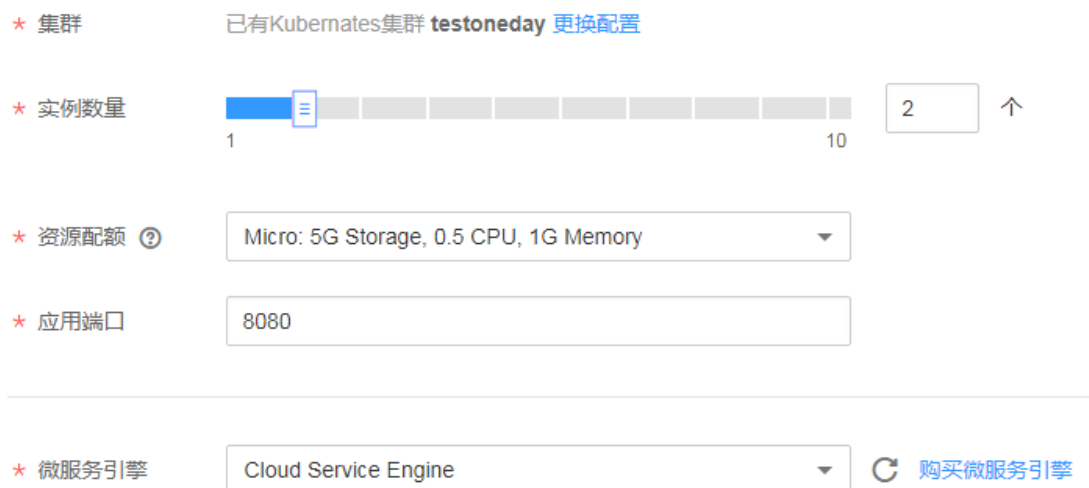
基于ServiceComb框架, 支持接口和接...

4、框架选择ServiceComb(SpringMVC)后，点击下一步。

5、部署系统选择CCE，并选择已有集群（如果没有可以在更换配置，选择新创建）。



6、选择微服务引擎，默认用Cloud Service Engine，其他默认即可。



7、下一步，确认提交，等待应用部署成功



4 打卡截图

选择左边菜单 基础设施 -> 微服务引擎，进入Cloud Service Engine的控制台。查看刚才部署成功的hello是否注册到服务中心：

应用管理

持续交付

软件中心

基础设施

K8S集群

弹性伸缩组

CCI命名空间

弹性云服务器

微服务引擎

您还可以购买 3 个微服务引擎专享版。

所有企业项目

所有状态(3)

名称

Q

C

I

名称	状态	版本	已部署实例数	服务中心连接地址	企业项目	创建时间	操作
Cloud Service Engine	可用	专业版	1/100 (1%)	https://cse.cn-north-1.myhuaweicloud.com	--	--	控制台 更多
cse-hupengchao	可用	专享版	4/100 (4%)	https://192.168.20.81:30100,https://192.1	default	2019/02/13 10:29:31 GMT+08:00	控制台 更多
cui/rush-test	可用	专享版	3/100 (3%)	https://192.168.0.91:30100	default	2019/02/11 06:47:16 GMT+08:00	控制台 更多

微服务控制台

Cloud Service Engine

仪表盘

服务目录

服务治理

全局配置

事务看板

仪表盘

您可以通过仪表盘实时查看微服务运行时的指标，以对微服务做出合适的治理动作。

排序方式： 吞吐量 平均时延 请求数

hello#0.0.1 springmvc

0 | 0 | 0%

0 | 0 | 0

吞吐量 0/s

熔断状态 Closed

实例数 2 90th 0ms

中位数时延 0ms 99th 0ms

平均时延 0ms 99.5th 0ms

DAY19 微服务应用实战之服务治理

1 打卡任务

作业：

创建两个微服务应用：webapp和helloworld，webapp用来访问helloworld，然后再创建helloworld的灰度版本，进行灰度发布。本作业操作步骤较多，请按操作指导一步步执行。

打卡：

验证灰度发布成功，并截图。

2 准备工作

- 1、切换区域为“华东-上海二”。
- 2、确认选择的ServiceStage版本为“基础版”（可免费使用10个实例，**超过10个实例或选择的是专业版则会按小时进行计费**）。

- 3、已拥有DevCloud账号和密码，并导入下面两个工程到自己的DevCloud账号下：

<https://github.com/servicestage-demo/helloclient.git>

<https://github.com/servicestage-demo/helloworld.git>

步骤1：登录[DevCloud代码托管](#)，点击“新建仓库”，进入新建代码仓库页面。

步骤2：在新建代码仓库页面，点击“导入仓库”tab页，在“源仓库路径”中输入“<https://github.com/servicestage-demo/helloclient.git>”，然后点击“确定”按钮，导入成功。

步骤3：重复步骤1和2，导入<https://github.com/servicestage-demo/helloworld.git>工程。

- 4、已有CCE集群

3 创建 ServiceComb 微服务应用 helloworld

- 1、登录ServiceStage，选择 总览 > 快捷操作 > 创建ServiceComb应用。
- 2、运行环境选择Java8，修改应用显示名称为helloworld（自动生成的应用名称不要修改），应用来源选择“源码仓库”，代码源来源选择DevCloud，输入密码进行授权（如已授权则这一步可省略），仓库名称选择准备工作导入的helloworld项目，分支选择master，版本为1.0.0。

* 应用类型 ServiceComb 更换

* 运行环境 Tomcat8 Java8

* 应用名称 servicestagecdsl

* 应用显示名称 helloworld

描述

* 应用来源 Jar包 源码仓库 模板

* 代码源来源 DevCloud GitHub Gitee Bitbucket GitLab

已绑定账户: [redacted]

命名空间 Demo 仓库名称 helloworld 分支 master

* 版本 1.0.0

- 3、下一步，部署系统选择CCE，并选择已有集群（如果没有可以在更换配置，选择新创建）

2 应用配置

云容器引擎CCE 云容器实例CCI Beta!

已有Kubernetes集群 testoneday 更换配置

* 配置方式 使用已有 新创建

* 部署集群 testoneday

* 集群命名空间 default

- 4、选择微服务引擎，默认用Cloud Service Engine，实例数量选择1个（节省资源），应用端口8081（与代码监听一致）

部署系统

云容器引擎CCE

Beta!

云容器实例CCI

集群

已有Kubernetes集群 **testoneday** [更换配置](#)

实例数量

1

10

1

个

资源配额 ?

Micro: 5G Storage, 0.5 CPU, 1G Memory

应用端口

8081

微服务引擎

Cloud Service Engine

[购买微服务引擎](#)

5、下一步，确认提交，等待应用部署成功



应用helloworld正在创建中...

[返回应用管理](#)
[查看应用详情](#)

就绪

2019/02/16 09:46:07 GMT+08:00 查询 DCS 实例信息 完成
2019/02/16 09:46:07 GMT+08:00 查询 RDS 实例信息 完成

构建

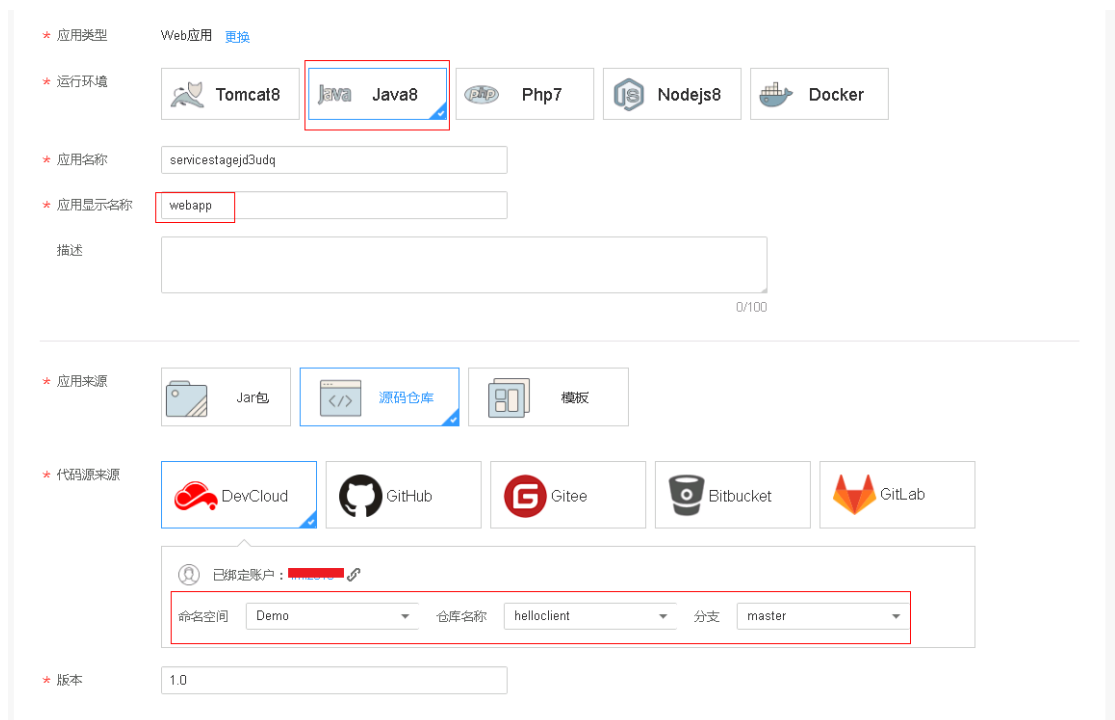
2019/02/16 09:46:07 GMT+08:00 构建应用 执行中
2019/02/16 09:46:07 GMT+08:00 创建构建任务 完成
2019/02/16 09:46:07 GMT+08:00 执行构建任务 完成
2019/02/16 09:46:07 GMT+08:00 查询构建状态 执行中
2019/02/16 09:46:07 GMT+08:00 查询构建状态 完成
2019/02/16 09:46:07 GMT+08:00 等待构建完成 执行中

4 创建 Web 应用 webapp 用于访问刚才的 helloworld

- 1、在ServiceStage菜单点击 应用管理 > 创建应用，选择 Web应用 > 创建Web应用。



- 2、运行环境选择Java8，修改应用显示名称为webapp（自动生成的应用名称不要修改），应用来源选择“源码仓库”，代码源来源选择DevCloud，选择准备工作导入的helloclient项目，分支选择master。



- 3、下一步，部署系统：CCE，集群选择和上面相同的集群，实例数改为1，端口8080

（与代码监听保持一致）

* 部署系统

云容器引擎CCE

Beta! 云容器实例CCI

* 集群

已有Kubernetes集群 **testoneday** 更换配置

* 实例数量

1

10

1

* 资源配额 ?

Micro: 5G Storage, 0.5 CPU, 1G Memory

* 应用端口

8080

- 4、选择负载均衡，如果已存在则直接跳过；如果没有负载均衡，则点击“去配置”，然后选择“新创建”，弹性公网IP选择“使用已有”，如果没有可以通过查看弹性公网IP去进行解绑，解绑后返回刷新选择后点击“确定”即可。

* 资源配额 ?

Micro: 5G Storage, 0.5 CPU, 1G Memory

* 应用端口

8080

* 访问方式

新建负载均衡 您有待完善的配置信息，去配置>>

* 域名

自动生成

自动生成的域名仅有7天有效期，您可以选择绑定域名或应用创建后绑定域名。

JVM参数

比如-Xms256m -Xmx1024m，多个参数以空格分隔，不填则使用默认值

更换访问方式配置

* 配置方式

使用已有 新创建

* 负载均衡

elb-o84p 查看负载均衡

* 弹性公网IP ?

新创建 使用已有

119.3.27.65 查看弹性公网IP

HTTPS

配置费用

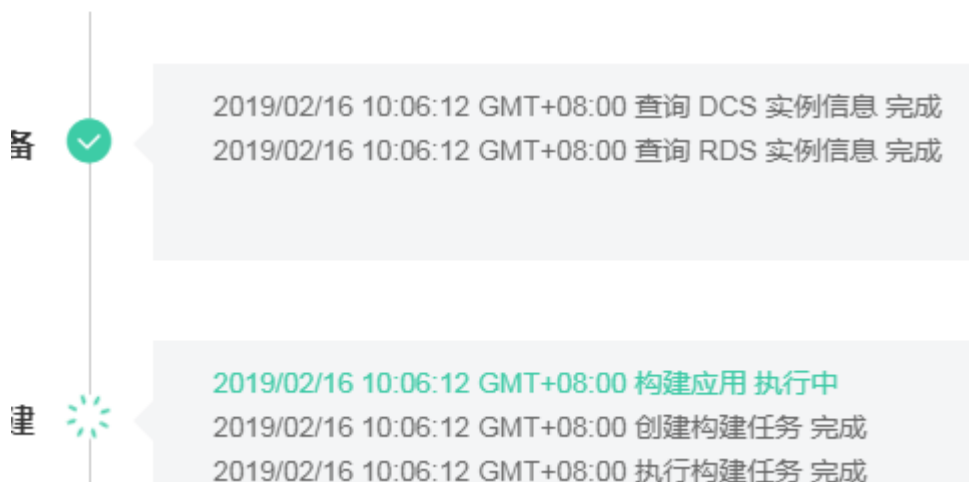
¥ 0.00/小时

- 5、下一步，确认提交，等待应用部署完成

应用webapp正在创建中...

返回应用管理

查看应用详情



6、部署完成后，点击查看应用详情，点击访问地址就可以访问刚才部署的服务

应用管理 > webapp

概览	实例列表	访问方式	更新升级	伸缩	事件	运行日志	阈值告警	运维配置	调度策略	基础设施
应用名称	webapp								类型	
状态	运行中								部署系统	
实例个数（正常/全部）	1/1								所在集群	
外部访问地址	86e220a7-2c10-4969-b724-cac5c6e6f648.hwstaff-I00387732.cn-north-1.huaweicse.com								命名空间	
创建时间	2019/02/16 10:06:12 GMT+08:00								标签	

在外部访问地址后增加接口url: /cse/helloworld?name=friends，效果类似如下：

浏览器地址栏显示：
不安全 | 86e220a7-2c10-4969-b724-cac5c6e6f648.hwstaff-I00387732.cn-north-1.huaweicse.com/cse/helloworld?name=friends
应用 | CICD | OLD | MicroService | 备忘 | K8S | golang | 原型 | oauth | FUXI | 环境 | ServiceStage · Co...
"hi, friends"

5 创建 ServiceComb 微服务应用 helloworldgray

- 1、在ServiceStage菜单，点击 总览 > 快捷操作 > 创建ServiceComb应用。
- 2、运行环境选择Java8，修改应用显示名称为helloworldgray（自动生成的应用名称不

要修改），应用来源选择“源码仓库”，代码源来源选择DevCloud，输入密码进行授权（如已授权则这一步可省略），仓库名称选择准备工作导入的helloworld项目，分支选择gray，版本填写2.0.0

应用类型: ServiceComb 更换

运行环境: Tomcat8, Java8

应用名称: servicestageff9k3

应用显示名称: helloworldgray

描述: 0/100

应用来源: Jar包, 源码仓库, 模板

代码源来源: DevCloud, GitHub, Gitee, Bitbucket, GitLab

已绑定账户: [redacted]

命名空间: Demo, 仓库名称: helloworld, 分支: gray

版本: 2.0.0

3、剩下操作与上面步骤3后的完全一样（注意端口号要改为8081），等待部署完成。

6 灰度发布

当前 helloworld 存在两个版本, master 分支的版本 1.0.0 返回的是 hi, xxx; gray 分支的 2.0.0 返回的是 hello, xxx。下面我们添加灰度策略

- 1、选择左边菜单 基础设施 -> 微服务引擎，进入Cloud Service Engine的控制台。在服务目录，点击进入helloworld微服务

微服务控制台
Cloud Service Engine

仪表盘
服务目录
服务治理
全局配置
事务看板

应用列表 微服务列表 实例列表

创建微服务 删除

微服务名称	所属应用
webapp	springmvc
helloworld	springmvc

- 2、左边菜单 灰度发布。添加发布规则：对于参数name=world的导流到2.0.0版本

创建新规则

★ 发布规则

权重 自定义

★ 规则名称

作用域

★ 版本 ☐ 1.0.0 ☒ 2.0.0

☐ 是否添加自定义版本

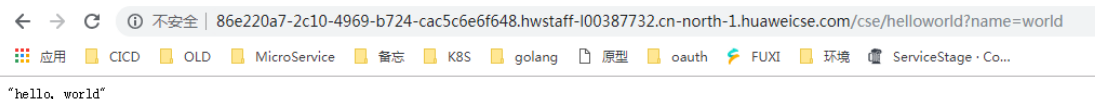
规则配置

★ 参数名

★ 规则

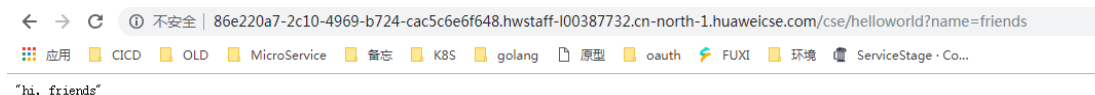
• 参数符合【name=world】条件的请求将会分配到【2.0.0】微服务版本中

- 3、再访问接口`http://{服务域名}/cse/helloworld?name=friends`,可以看到返回一直都是“hi, friends”; 如果把name改为world, 结果返回一直是“hello, world”



7 打卡截图

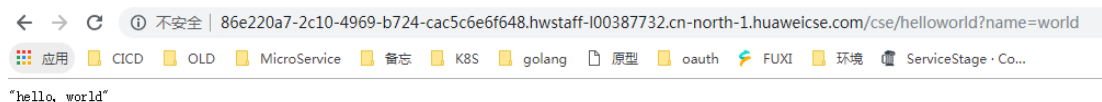
- 1、访问接口url, `http://{服务域名}/cse/helloworld?name=friends`, 返回“hi, friends”



- 2、选择左边菜单 基础设施 -> 微服务引擎, 进入Cloud Service Engine的控制台。查看服务治理



3、访问接口`http://{服务域名}/cse/helloworld?name=world`,可以看到返回一直都是“hello, world”



DAY20 微服务应用运维之应用监控

1 打卡任务

作业：

通过提供并已安装好的[示例应用](#)进行应用监控体验，以进一步了解应用监控提供的各种能力。

打卡：

查询服务对应实例的CPU使用率和事务拓扑图并截图作为打卡。

2 准备工作

已获得体验需要用到相关账号信息：

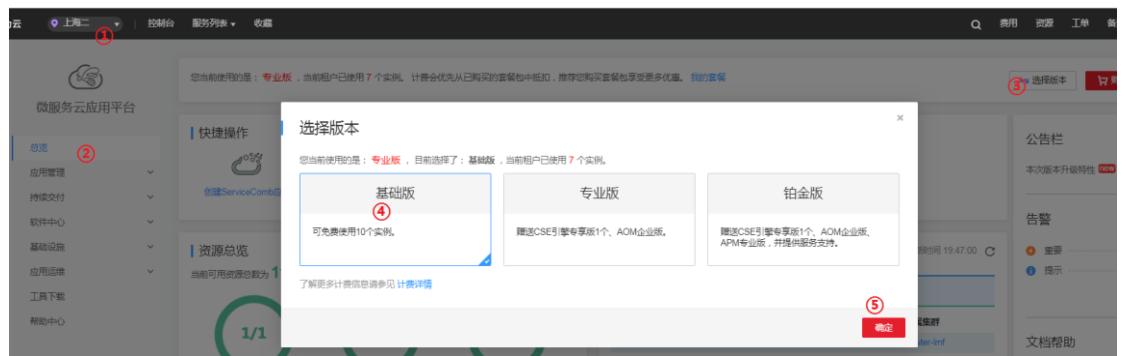
- 1、华为云IAM用户登录账号（账号名：apmdemo，用户名：apmdemo1，密码：apm1234）
- 2、示例应用网站登录账号（账号名：apm，密码：123456）

3 体验过程

- 1、登录ServiceStage，选择总览，查看当前使用的ServiceStage版本是否为“基础版”，如果之前选择的是“专业版”，如下图所示：



请立即重新选择版本为“基础版”，选择方式如下图所示：



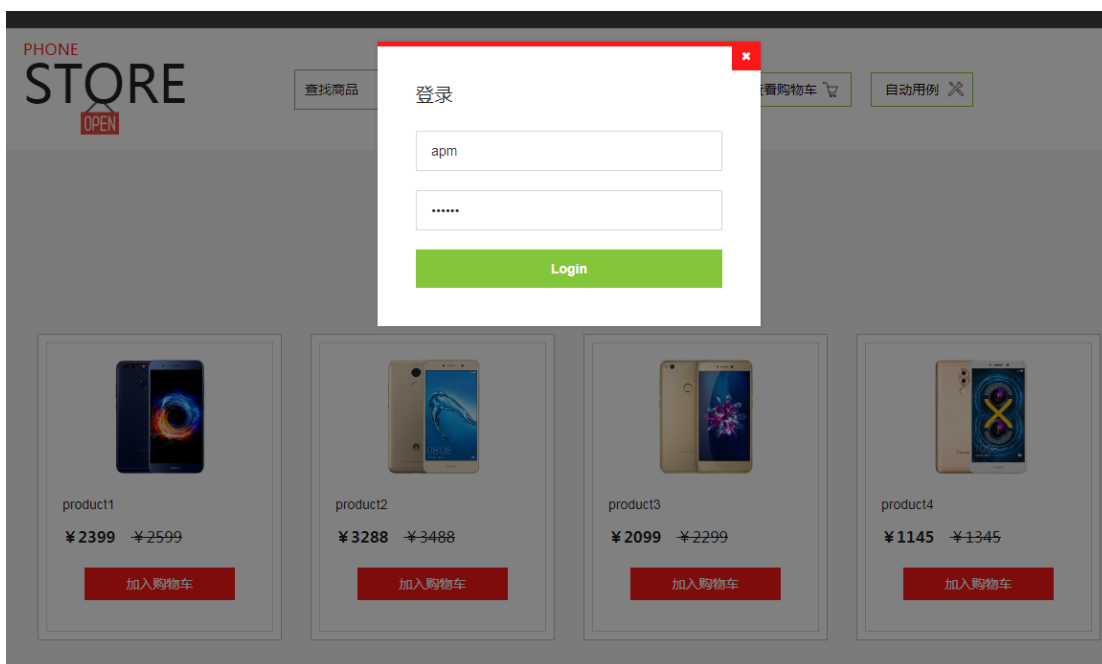
- 2、点击菜单 应用管理 > 应用列表，如果列表不为空，则选中所有应用，点击“删除应用”按钮，删除所有前面课程创建的应用（注意：如果不删除，保留的应用实例可能会导致账号扣费，请务必执行此操作）。



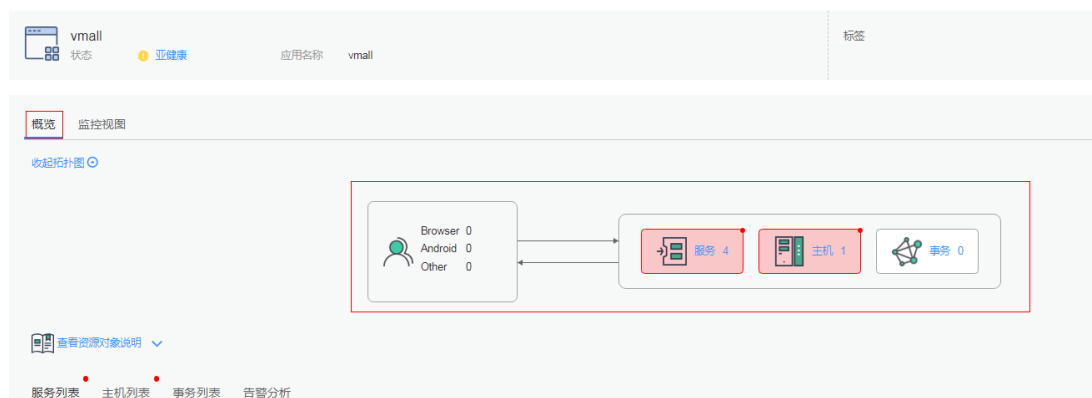
- 3、退出当前账号，返回华为云登录界面，点击“IAM用户登录”切换至IAM用户账号登录界面，使用IAM用户账号（账号名：apmdemo，用户名：apmdemo1，密码：apm1234）登录。



- 4、打开[示例应用网站](#)，点击“登录”按钮，输入应用登录账号（账号名：apm，密码：123456），点击“Login”登录。

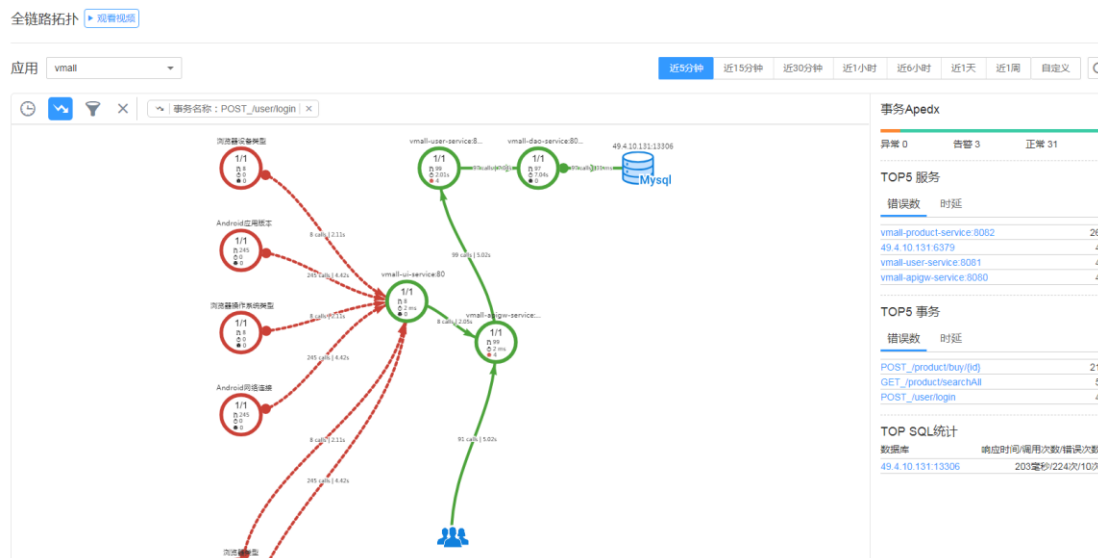


- 5、在华为云网站打开[ServiceStage应用运维管理](#)，依次点击左侧手风琴菜单“应用监控” > “应用列表”，打开应用列表展示界面。
- 6、在应用列表中单击名称为vmall的应用，在“概览”页签中查看该应用包含的服务、主机、事务等概览信息。



7、在服务列表的几个服务中，点击“服务列表”页签，选择其中一个服务对应实例查看CPU使用率，并截取指标图表的放大效果图作为打卡截图。

8、点击选择“事务列表”页，在事务列表中点击任意的事务名称，跳转到事务总览页面，在搜索框中输入login查询登录事务，查看用户登录事务的拓扑图。



9、在事务列表搜索框输入search进行过滤，查看对应事务拓扑图，截取拓扑信息作为打卡截图。

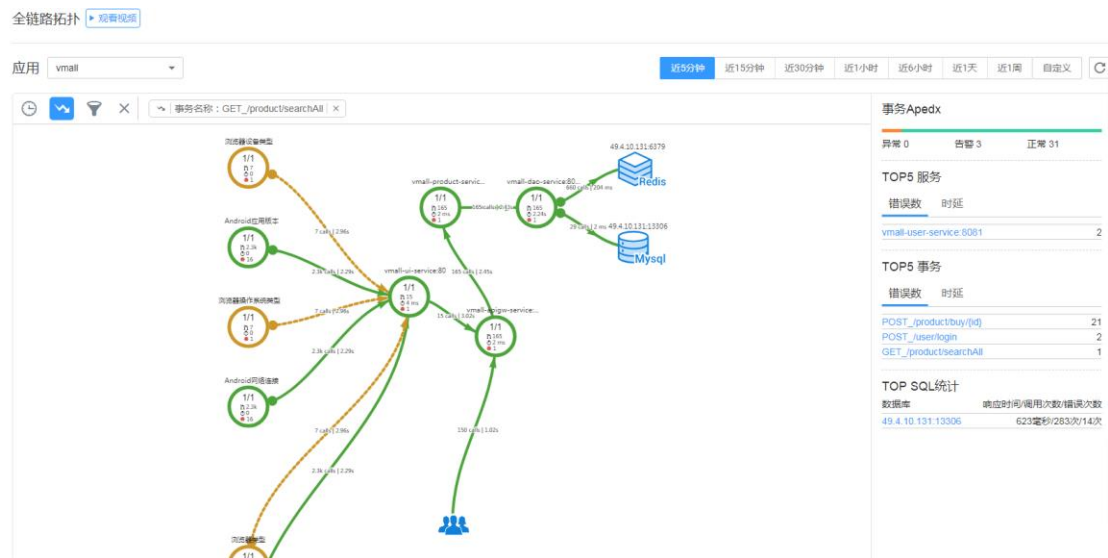
10、应用监控还包括很多其他高级功能，可以自行摸索浏览。

4 打卡截图

1、vmall-ui-service实例CPU使用率指标图表放大图效果，类似如下：



2、在事务列表搜索框输入search，查看对应事务拓扑图，类似如下：



DAY21 微服务应用运维之调用追踪

1 打卡任务

作业：

通过提供并已安装好的[示例应用](#)进行应用调用追踪体验，以便进一步了解调用追踪的功能。

打卡：

将某一失败事务的最底层原因进行截图作为打卡。

2 准备工作

已获得体验需要用到的相关账号信息：

- 1、华为云IAM用户登录账号（账号名：apmdemo，用户名：apmdemo1，密码：apm1234）
- 2、示例应用网站登录账号（账号名：apm，密码：123456）

3 体验过程

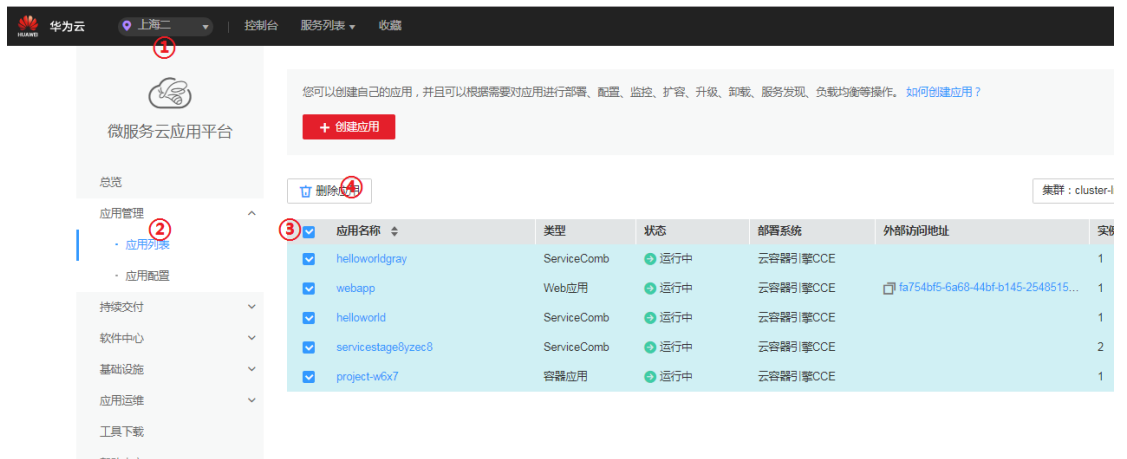
- 1、登录ServiceStage，选择总览，查看当前使用的ServiceStage版本是否为“基础版”，如果之前选择的是“专业版”，如下图所示：



请立即重新选择版本为“基础版”，选择方式如下图所示：



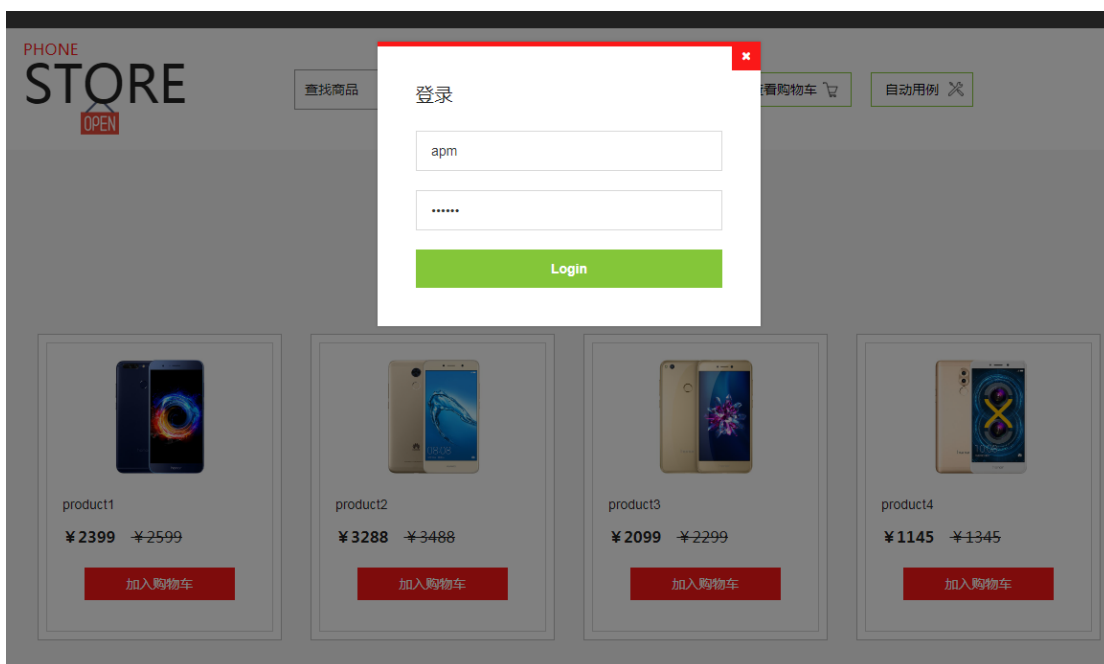
- 2、点击菜单 应用管理 > 应用列表，如果列表不为空，则选中所有应用，点击“删除应用”按钮，删除所有前面课程创建的应用（注意：如果不删除，保留的应用实例可能会导致账号扣费，请务必执行此操作）。



- 3、退出当前账号，返回华为云登录界面，点击“IAM用户登录”切换至IAM用户账号登录界面，使用IAM用户账号（账号名：apmdemo，用户名：apmdemo1，密码：apm1234）登录。



- 4、打开[示例应用网站](#)，点击“登录”按钮，输入应用登录账号（账号名：apm，密码：123456），点击“Login”登录。



- 5、在华为云网站打开[ServiceStage应用运维管理](#)，依次点击左侧手风琴菜单“性能管理” > “调用链”，打开调用链展示界面。
- 6、在调用链界面，应用选择为vmall后点击“搜索”按钮，在搜索结果中会将vmall应用的所有服务对应所有事务都查询出来，选取其中一个服务的事务来查询调用关系，比如下图即选择的是vmall-apigw-service的“POST_/user/login”事务调用关


系:

调用链 - 调用关系

服务数: 3 调用深度: 17 总Span数: 22

请输入关键词

服务	方法	参数	状态	时间线(ms)	操作
vmall-apigw-service	invoke	/user/user/login	成功	21	详情
vmall-user-service	invoke	/user/login	成功	19	详情
vmall-dao-service	invoke	/persistence/user	成功	15	详情

点击  进一步查看详细信息。

- 使用高级搜索查询所有失败的事务，选取其中的一个失败事务进行调用链分析，查看详情中的失败原因，并截取失败原因界面作为打开截图。

4 打卡截图

将失败事务的最底层原因进行截图作为打卡，截图类似如下：

调用链 - 调用关系

服务数: 4 调用深度: 18 总Span数: 22

请输入关键词

服务	方法	参数	状态	时间线(ms)	操作
vmall-ui-service	invoke	/product/searchAll	失败	3279	详情
vmall-apigw-service	invoke	/product/product/searchAll	失败	3273	详情
vmall-product-service	invoke	/product/searchAll	失败	0	详情
vmall-product-service	doget		失败	0	详情 参数采集
vmall-product-service	getItems		失败	1	详情 参数采集
vmall-product-service	getInputStream	http://localhost:8083/persistence/products	失败	0	详情 参数采集
vmall-dao-service	invoke	/persistence/products	失败	2	详情
vmall-dao-service	doget		失败	0	详情 参数采集
vmall-dao-service	findProducts		失败	1265	详情 参数采集
vmall-dao-service	exists		失败	2001	详情 参数采集
vmall-dao-service			成功	1	详情 参数采集
vmall-product-service			成功	1	详情 参数采集

详情

类型	描述
TX-TYPE	GET_product/searchAll
clusterId	UnknownCluster
destinationId	REDIS
exception class	redis.clients.jedis.exceptions.JedisTimeoutException
exception msg	java.net.SocketTimeoutException: connect timed out
monitorGroup	vmall
namespace	default
result	1
root	false
serviceType	REDIS