

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：



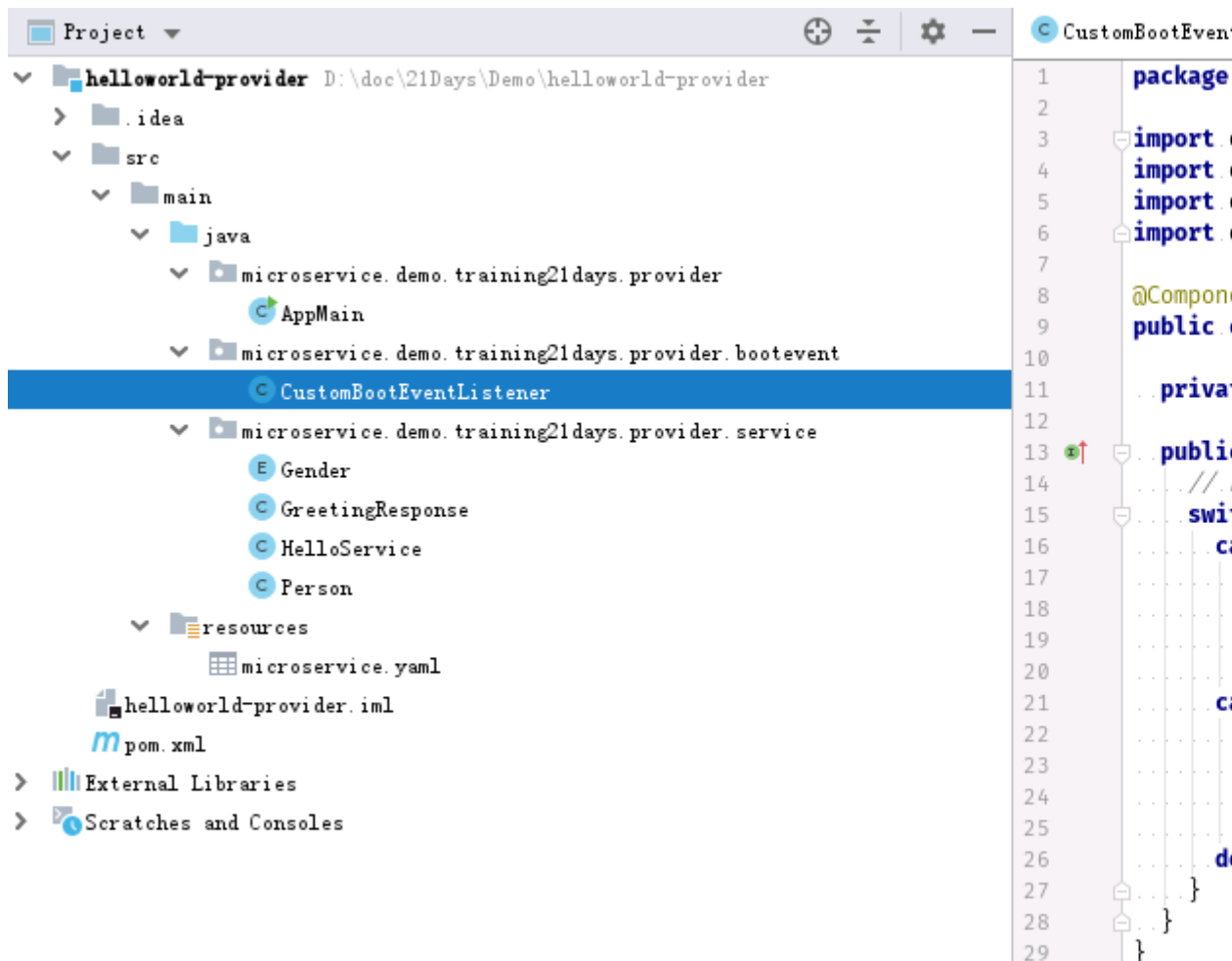
2 准备工作

- 1、正常运行Day2的demo

3 监听 ServiceComb 框架启动事件

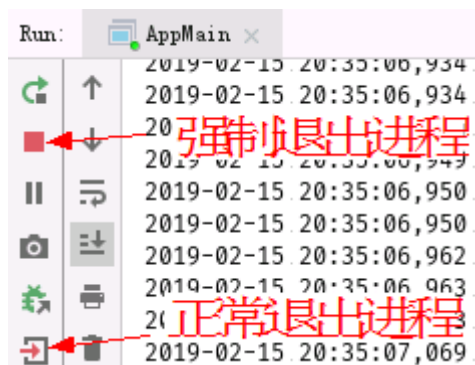
- 1、在provider服务中定义一个BootListener的子类，打上@Component注解使Spring框架

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

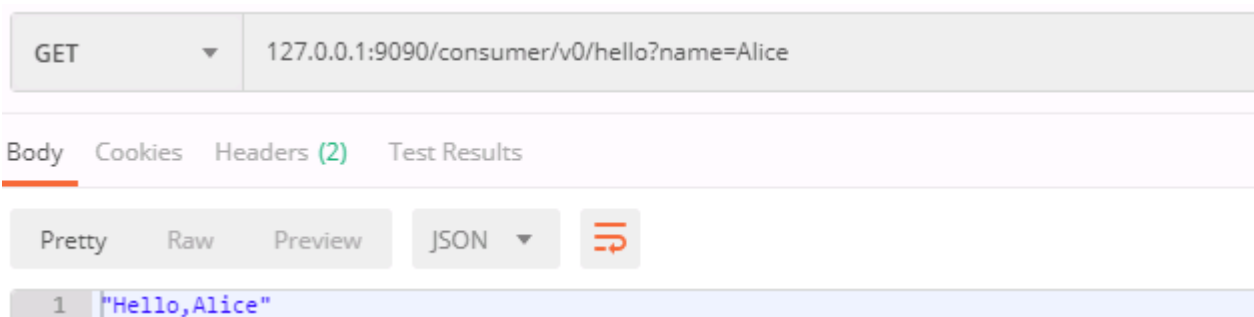
```
===== microservice.demo.training21days.provider.bootevent.CustomB
Service startup completed! microservice.demo.training21days.provider.bootevent.CustomBoot
===== microservice.demo.training21days.provider.bootevent.CustomB
keyStore [server.p12] file not exist, please check! org.apache.servicecomb.foundation.ver
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
```

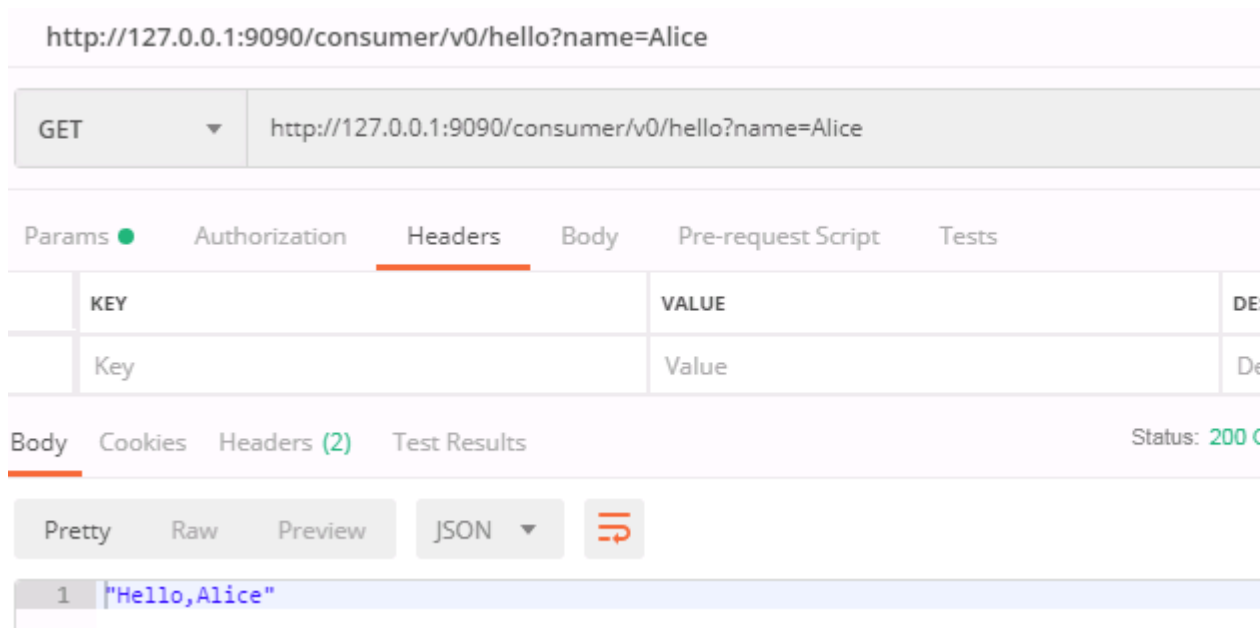
- 2、启动服务，调用consumer，consumer可以正常发现provider服务，并调用成功



- 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

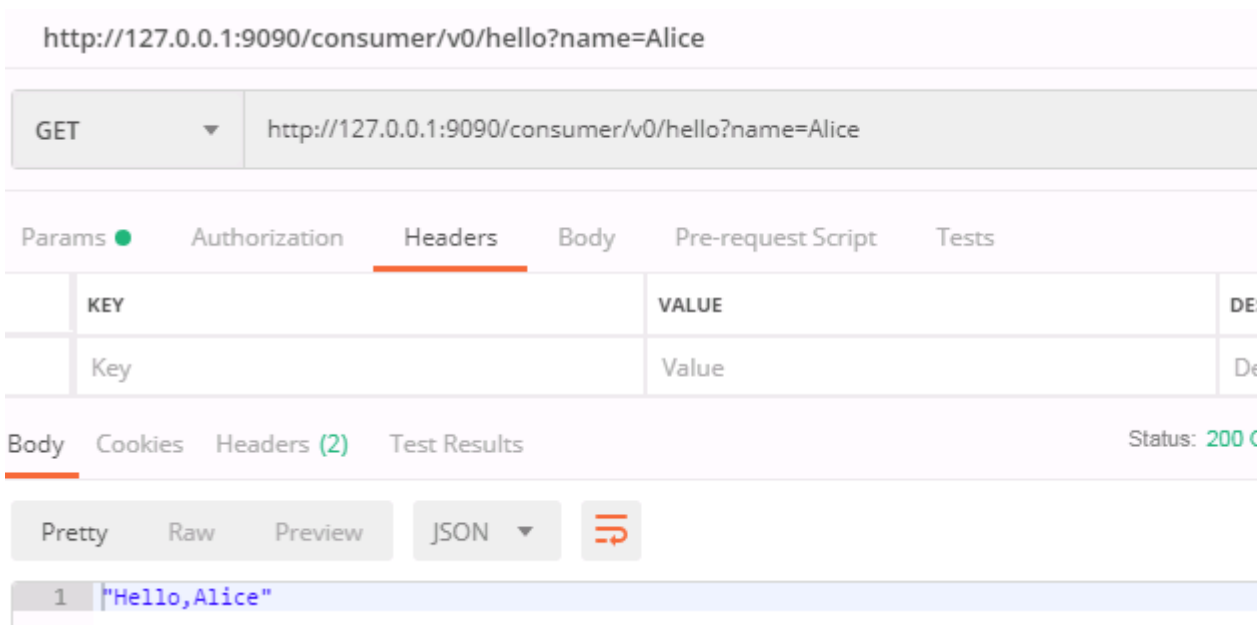
1、启动成功和实例退出的提示日志如下

```
===== 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



The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:9090/consumer/v0/hello?name=Alice`
- Method:** GET
- Headers:** The 'Headers' tab is selected, showing a table with columns 'KEY' and 'VALUE'. The table is currently empty.
- Body:** The 'Body' tab is selected, showing the response content: `"Hello,Alice"`.
- Status:** 200 OK

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：

