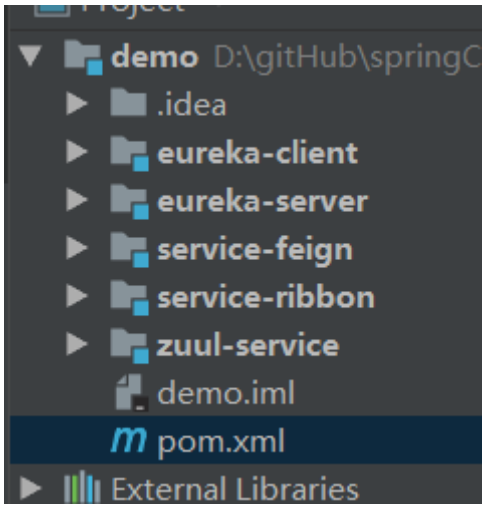
10-28



demo：主Maven项目 管理依赖。springboot 项目。

eureka-client: 服务提供者 。

eureka-server：注册中心，管理可用服务并维系一张可服务表。有web界面。

ribbon：实现负载均衡。

feign：更好的实现负载均衡。

zuul：路由，路由过滤

基于springcloud (Finchley.RELEASE) 版本

demo maven依赖：

```
<?xml version="1.0" encoding="UTF-8"?><project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>cn.mengpeng</groupId>
<artifactId>demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>pom</packaging>
<name>demo</name>
<description>Demo project for Spring Boot</description>
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
<version>2.0.3.RELEASE</version>
    <relativePath/>
</parent>
<modules>
    <module>eureka-server</module>
    <module>eureka-client</module>
```

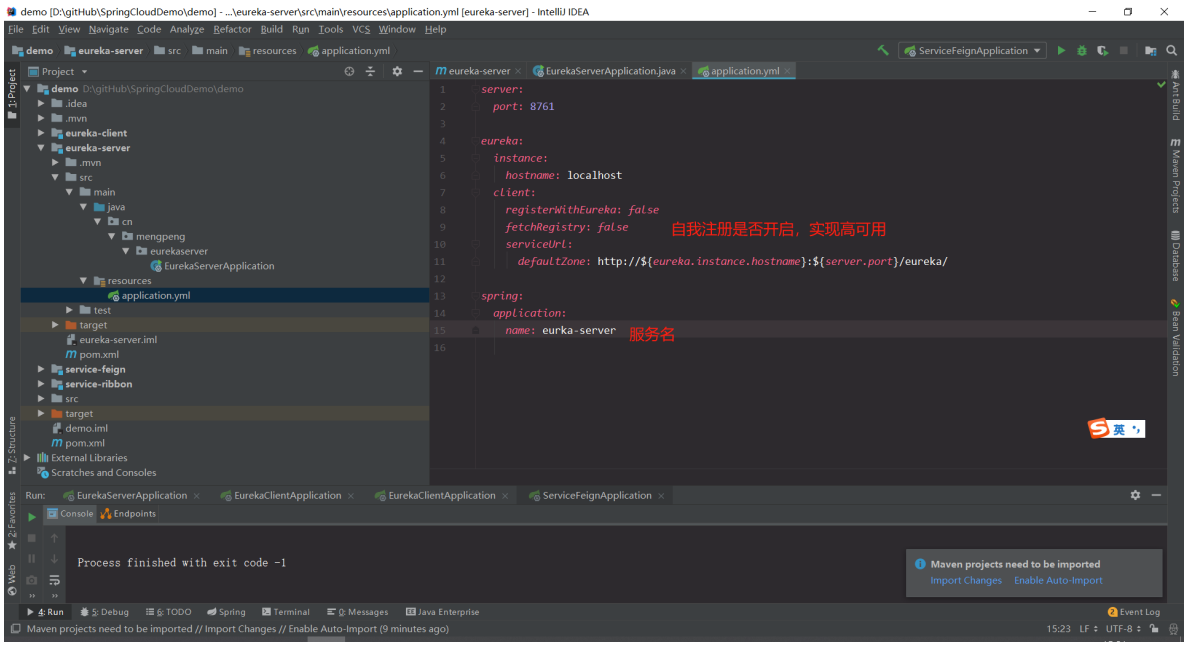
```
    <module>service-ribbon</module>
    <module>service-feign</module>
</modules>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
<java.version>1.8</java.version>
<spring-cloud.version>Finchley.RELEASE</spring-cloud.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>5.0.4.RELEASE</version>
</dependency>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.12</version>        </dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
<exclusions>
<exclusion>

<groupId>org.junit.vintage</groupId>
<artifactId>junit-vintage-engine</artifactId>
</exclusion>
</exclusions>
</dependency>
</dependencies>
<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>Finchley.RELEASE</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
```

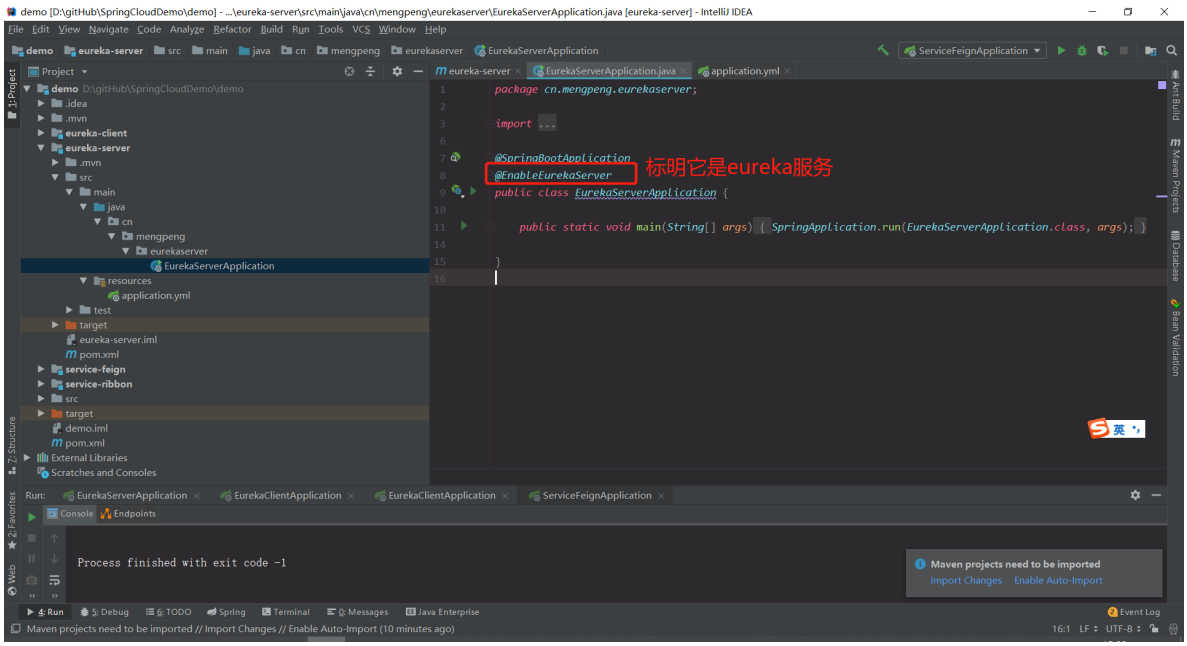
```
</build></project>
```

eureka-server：

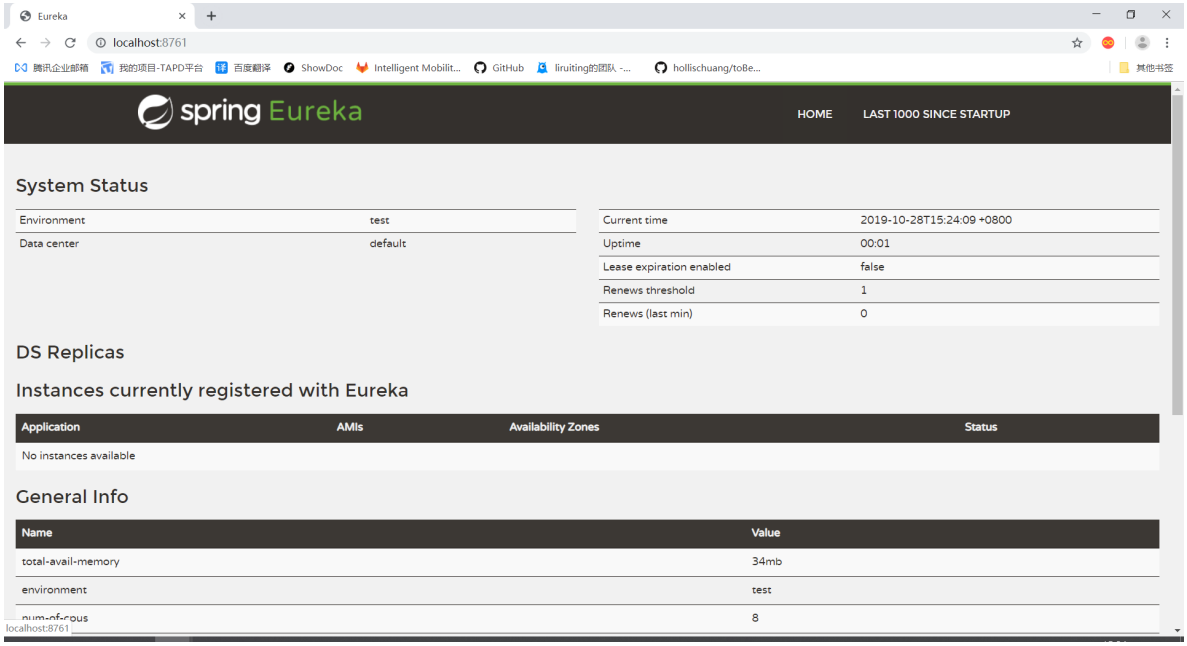
所需依赖：

```
<parent>
    <groupId>cn.mengpeng</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
    </dependency>
</dependencies>
```





启动，并访问服务端界面，此时未有任何可用的服务注册进来。



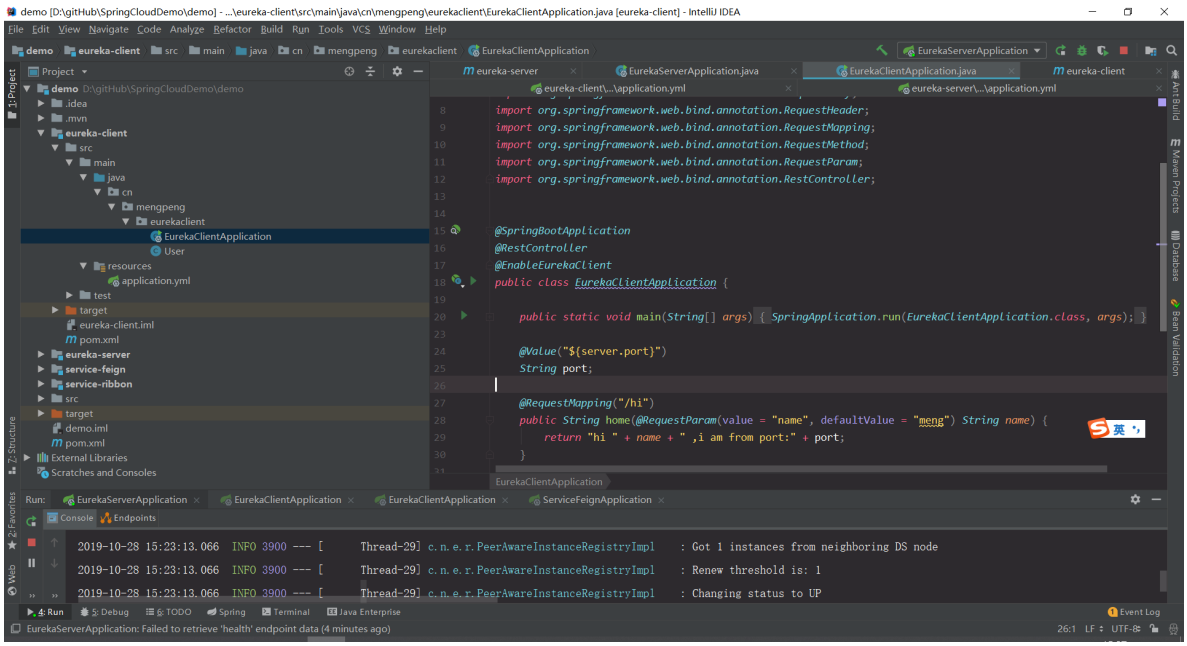
创建 服务提供方：

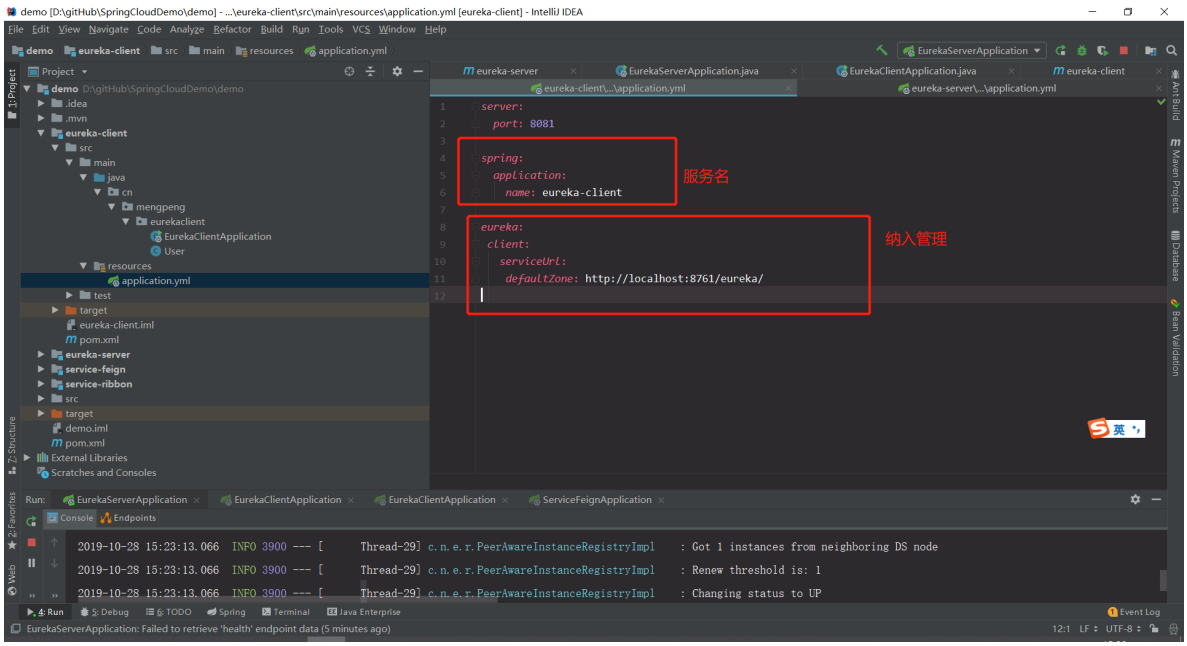
新建一个springboot工程

添加依赖： 父依赖 第一个里有

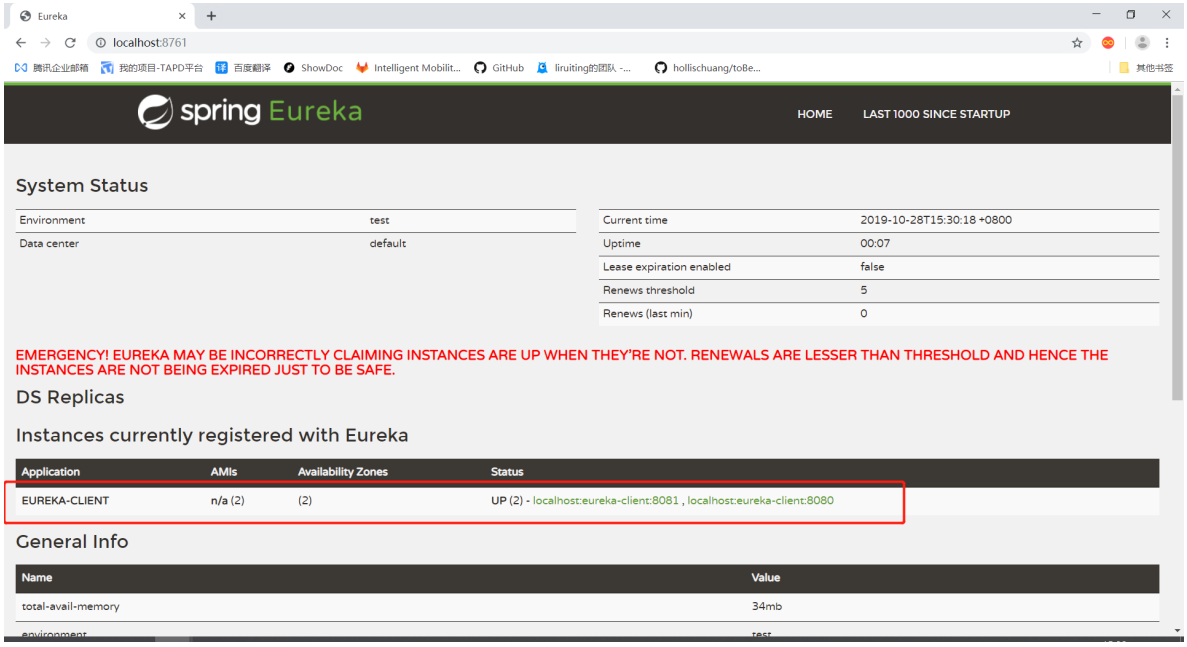
```
    <dependencies>
      <dependency>
          <groupId>org.springframework.cloud</groupId>
          <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
      </dependency>
  </dependencies>
```

新建可访问访问url。 申明这是一个@EnableEurekaClient





分别以8080，8081端口多实例启动它。



有两个服务注册进来了。

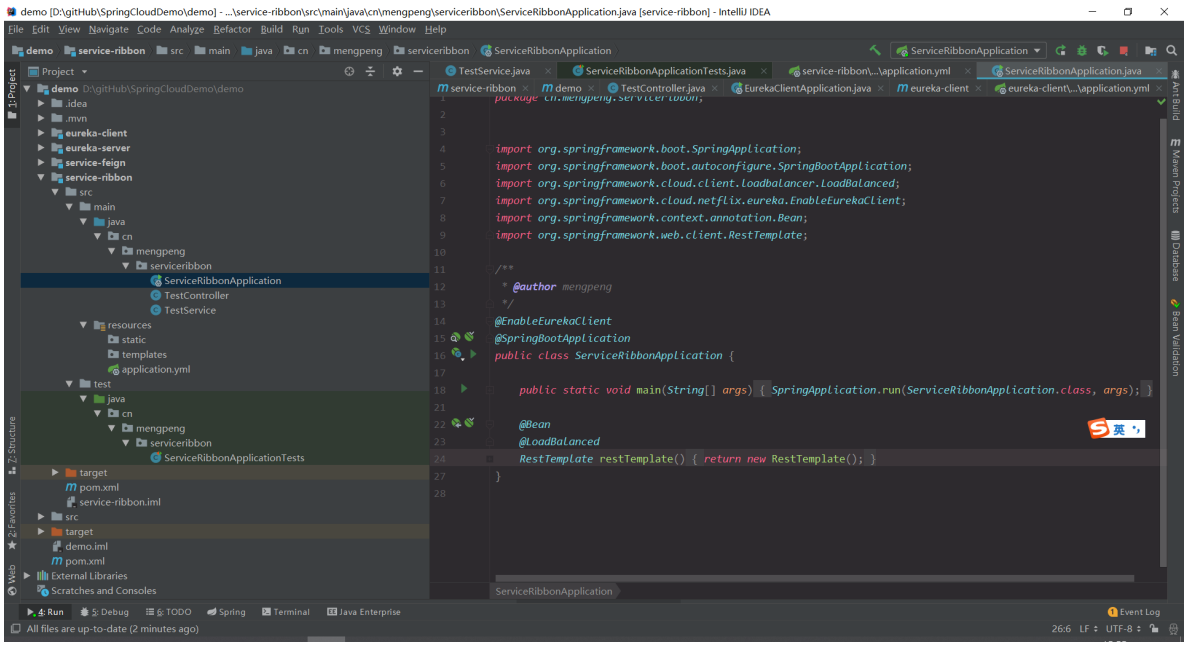
构建service-ribbon：

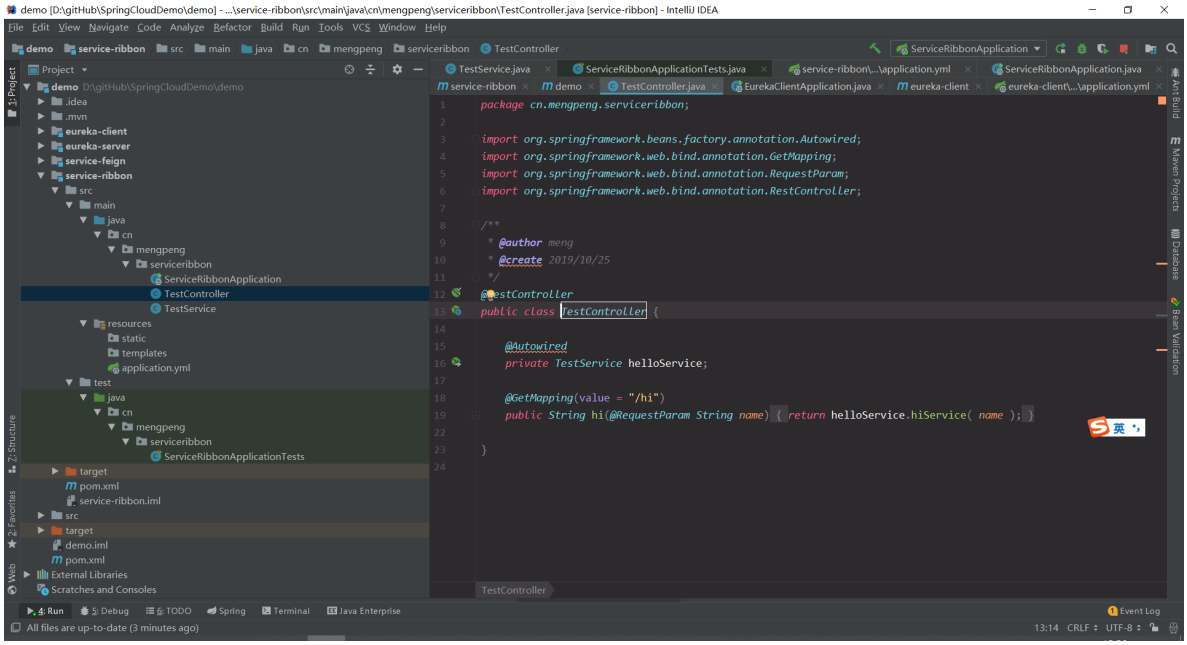
新建一个springboot项目；

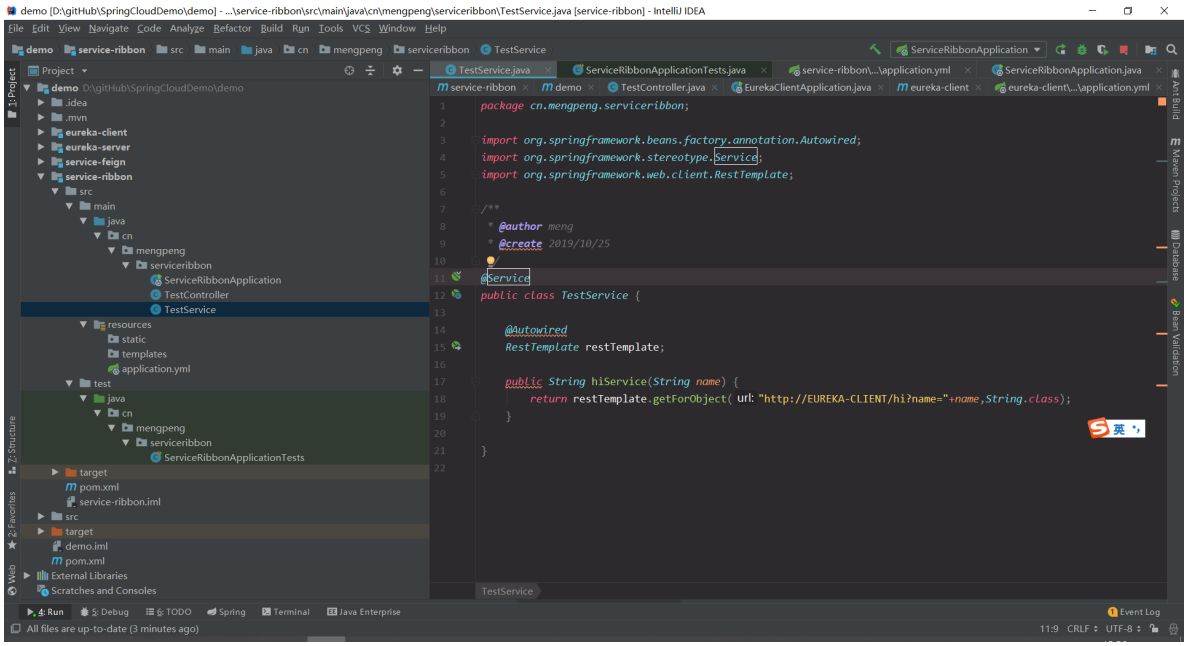
依赖：父依赖 第一个里有

```
<dependencies>

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
        </dependency>


        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
        </dependency>
    </dependencies>
```

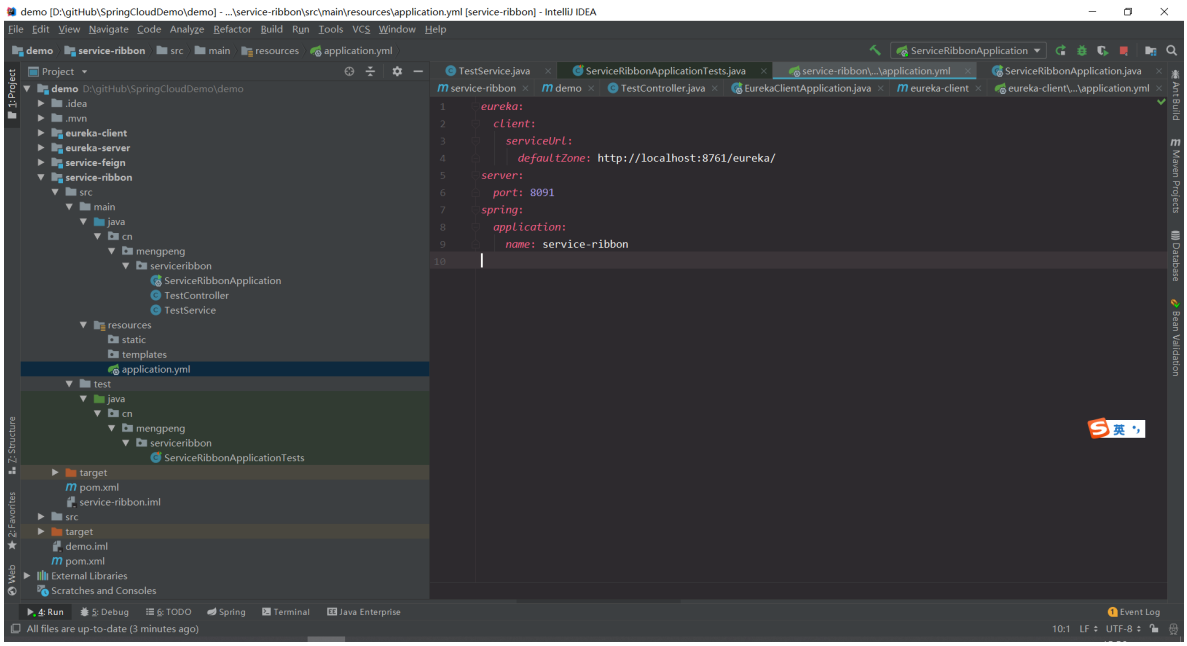
启动项：

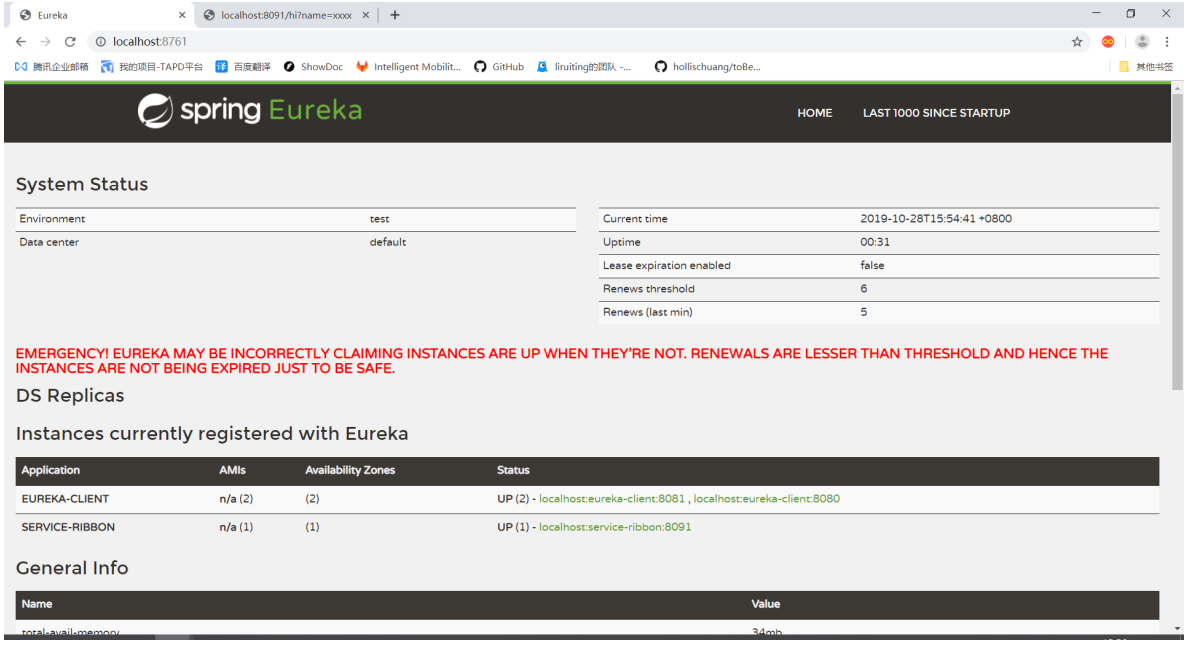

demo [D:\gitHub\SpringCloudDemo\demo] - ...\service-ribbon\src\main\java\cn\mengpeng\serviceribbon\ServiceRibbonApplication.java [service-ribbon] - IntelliJ IDEA
package cn.mengpeng.serviceribbon;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;
/**
* @author mengpeng
*/
@EnableEurekaClient
@SpringBootApplication
public class ServiceRibbonApplication {
public static void main(String[] args) { SpringApplication.run(ServiceRibbonApplication.class, args); }
@Bean
@LoadBalanced
RestTemplate restTemplate() { return new RestTemplate(); }
}

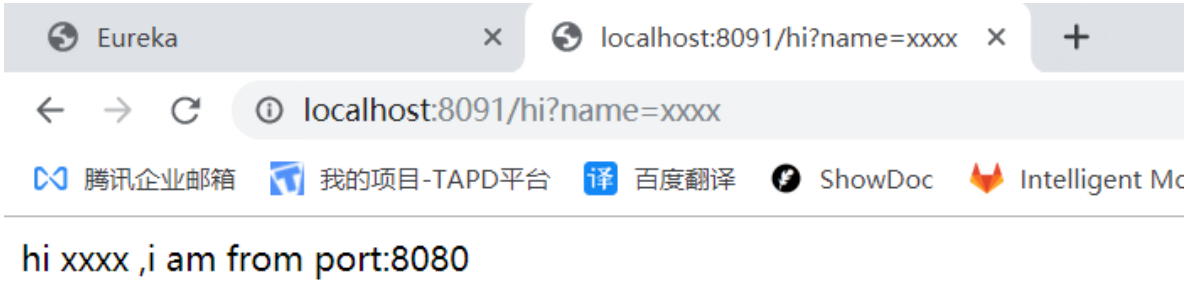

demo [D:\gitHub\SpringCloudDemo\demo] - ...\service-ribbon\src\main\java\cn\mengpeng\serviceribbon\TestController.java [service-ribbon] - IntelliJ IDEA
package cn.mengpeng.serviceribbon;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
/**
* @author meng
* @create 2019/10/25
*/
@RestController
public class TestController {
@Autowired
private TestService helloService;
@GetMapping(value = "/hi")
public String hi(@RequestParam String name) { return helloService.hiService(name); }
}

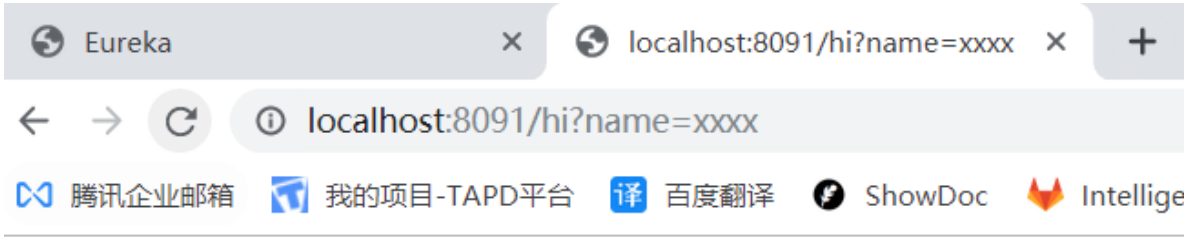

demo [D:\gitHub\SpringCloudDemo\demo] - ...\service-ribbon\src\main\java\cn\mengpeng\serviceribbon\TestService.java [service-ribbon] - IntelliJ IDEA
package cn.mengpeng.serviceribbon;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
/**
* @author meng
* @create 2019/10/25
*/
@Service
public class TestService {
@Autowired
RestTemplate restTemplate;
public String hiService(String name) {
return restTemplate.getForObject(url: "http://EUREKA-CLIENT/hi?name="+name,String.class);
}
}

纳入服务。


demo [D:\gitHub\SpringCloudDemo\demo] - ...\service-ribbon\src\main\resources\application.yml [service-ribbon] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ServiceRibbonApplication
Project
demo D:\gitHub\SpringCloudDemo\demo
.idea
.mvn
eureka-client
eureka-server
service-feign
service-ribbon
src
main
java
cn
mengpeng
serviceribbon
ServiceRibbonApplication
TestController
TestService
resources
static
templates
application.yml
test
java
cn
mengpeng
serviceribbon
ServiceRibbonApplicationTests
target
pom.xml
service-ribbon.iml
src
target
demo.iml
pom.xml
External Libraries
Scratches and Consoles
TestService.java
ServiceRibbonApplicationTests.java
service-ribbon\...\application.yml
ServiceRibbonApplication.java
service-ribbon
demo
TestController.java
EurekaClientApplication.java
eureka-client
eureka-client\...\application.yml
eureka:
client:
serviceUrl:
defaultZone: http://localhost:8761/eureka/
server:
port: 8091
spring:
application:
name: service-ribbon
4: Run
5: Debug
6: TODO
Spring
Terminal
Java Enterprise
Event Log
All files are up-to-date (3 minutes ago)
10:1 LF UTF-8


Eureka
localhost:8091/hi?name=xxxx
localhost:8761
spring Eureka
HOME
LAST 1000 SINCE STARTUP
System Status
Environment test
Data center default
Current time 2019-10-28T15:54:41 +0800
Uptime 00:31
Lease expiration enabled false
Renews threshold 6
Renews (last min) 5
EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.
DS Replicas
Instances currently registered with Eureka
Application AMIs Availability Zones Status
EUREKA-CLIENT n/a (2) (2) UP (2) - localhost:eureka-client:8081 , localhost:eureka-client:8080
SERVICE-RIBBON n/a (1) (1) UP (1) - localhost:service-ribbon:8091
General Info
Name Value


Eureka
localhost:8091/hi?name=xxxx
localhost:8091/hi?name=xxxx
腾讯企业邮箱
我的项目-TAPD平台
百度翻译
ShowDoc
Intelligent Mc
hi xxxx ,i am from port:8080



hi xxxx ,i am from port:8081

负载均衡。

构建Feign

ribbon实现的负载均衡所需要的参数须在请求的URL中进行拼接，如果参数少的话或许我们还可以忍受，一旦有多个参数的话，这时拼接请求字符串就会效率低下，并且显得好傻。

feign：

https://www.cnblogs.com/huangjuncong/p/9053576.html

熔断机制：

微服务（a）中一个服务可能依赖另一个服务(b)， 然 ，当b服务出现故障或网络异常时，大量的请求可能会积压在 a 和 b服务中， 导致线程阻塞，线程的资源快速消耗，从而引起'雪崩效应' 整个系统都变得不可用，为了避免雪崩效应， 开发出了熔断机制， 熔断机制是指：当请求失败次数超过所设置的 阀指时，程序不再进行逻辑操作，快速失败任务，返回失败信息。如此便不会导致大量进程阻塞。

熔断还有另一个机制： 自我修复。

当服务b熔断后， 经过一段时间，会半打开熔断器，检查一部分请求是否正常，其他请求快速失败，如果检查的这部分请求正常了，则可以认为服务b已经正常，就会关闭熔断器。

feign开启熔断一直失败：

```
feign:
    hystrix:
        enabled: true
```

添加 开启后依然是失败，

坑： 添加一系列注册列表后熔断器开启成功。

```
eureka:
    client:
        register-with-eureka: true
        fetch-registry: true
```

ribbon 和 feign

Ribbon和Feign都是用于调用其他服务的，不过方式不同。

1.启动类使用的注解不同，Ribbon用的是@RibbonClient，Feign用的是@EnableFeignClients。

2.服务的指定位置不同，Ribbon是在@RibbonClient注解上声明，Feign则是在定义抽象方法的接口中使用@FeignClient声明。

3.调用方式不同，Ribbon需要自己构建http请求，模拟http请求然后使用RestTemplate发送给其他服务，步骤相当繁琐。

Feign则是在Ribbon的基础上进行了一次改进，采用接口的方式，将需要调用的其他服务的方法定义成抽象方法即可，

不需要自己构建http请求。不过要注意的是抽象方法的注解、方法签名要和提供服务的方法完全一致。

构建zuul

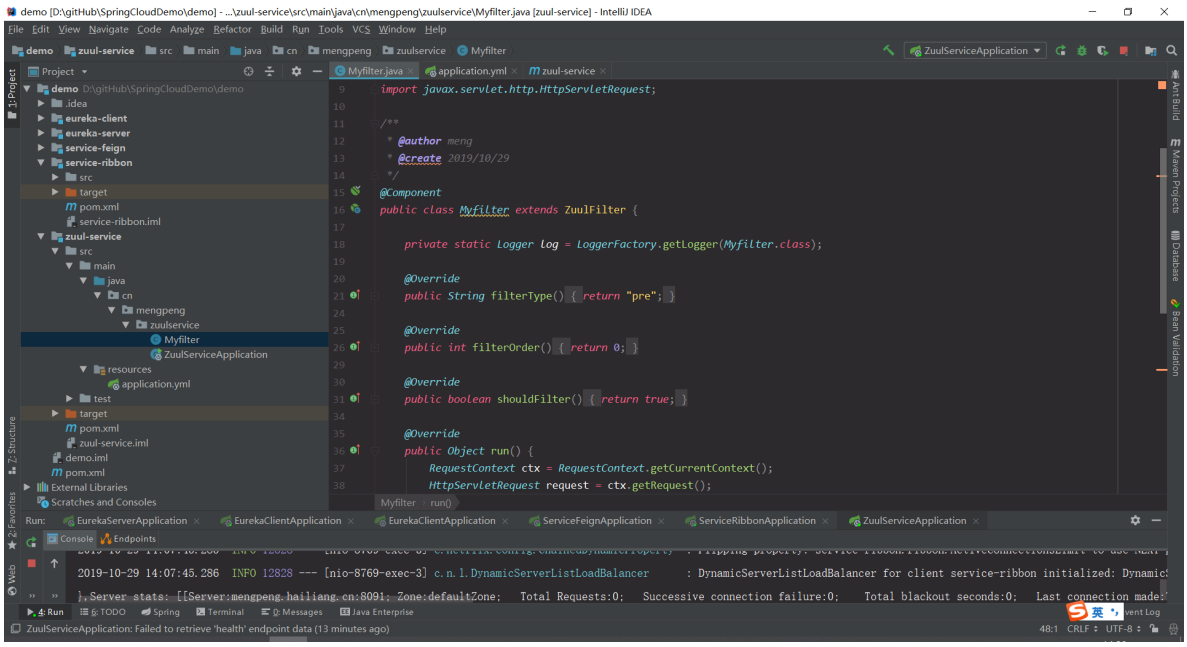
```
client:
  serviceUrl:
    defaultZone: http://localhost:8761/eureka/
server:
  port: 8769
spring:
  application:
    name: service-zuul
zuul:
  routes:
    api-a:
      path: /api-a/**
      serviceId: service-ribbon
    api-b:
      path: /api-b/**
      serviceId: SERVICE-FEIGN
```

主要是 设置路由的转发。

/api-a/开头的所有路由交给 service-ribbon 处理 ***

坑： 有些命名好像必须要大写。小写无法识别，注册中心的服务一律都会识别成大写不论你是否小写注册。

过滤器：



filterType：返回一个字符串代表过滤器的类型，在zuul中定义了四种不同生命周期的过滤器类型，具体如下：
pre：路由之前
routing：路由之时
post： 路由之后
error：发送错误调用
filterOrder：过滤的顺序
shouldFilter：这里可以写逻辑判断，是否要过滤，本文true,永远过滤。
run：过滤器的具体逻辑。可用很复杂，包括查sql，nosql去判断该请求到底有没有权限访问。

问题：

1.熔断降级处理失败了。

zuul设置路由成功了。

路由过滤成功。

负载均衡成功了。

单独的熔断降级处理是 通过的。

但是通过路由访问已经关闭了的服务 熔断降级失败了。可能是设置还没到那一步。

2.访问貌似有成功率， 就算服务是正常的 也不会说一直访问成功， 偶尔会失败。

而且失败的概率不小。