SpringCloud 完整案例第二季

一、开发环境准备

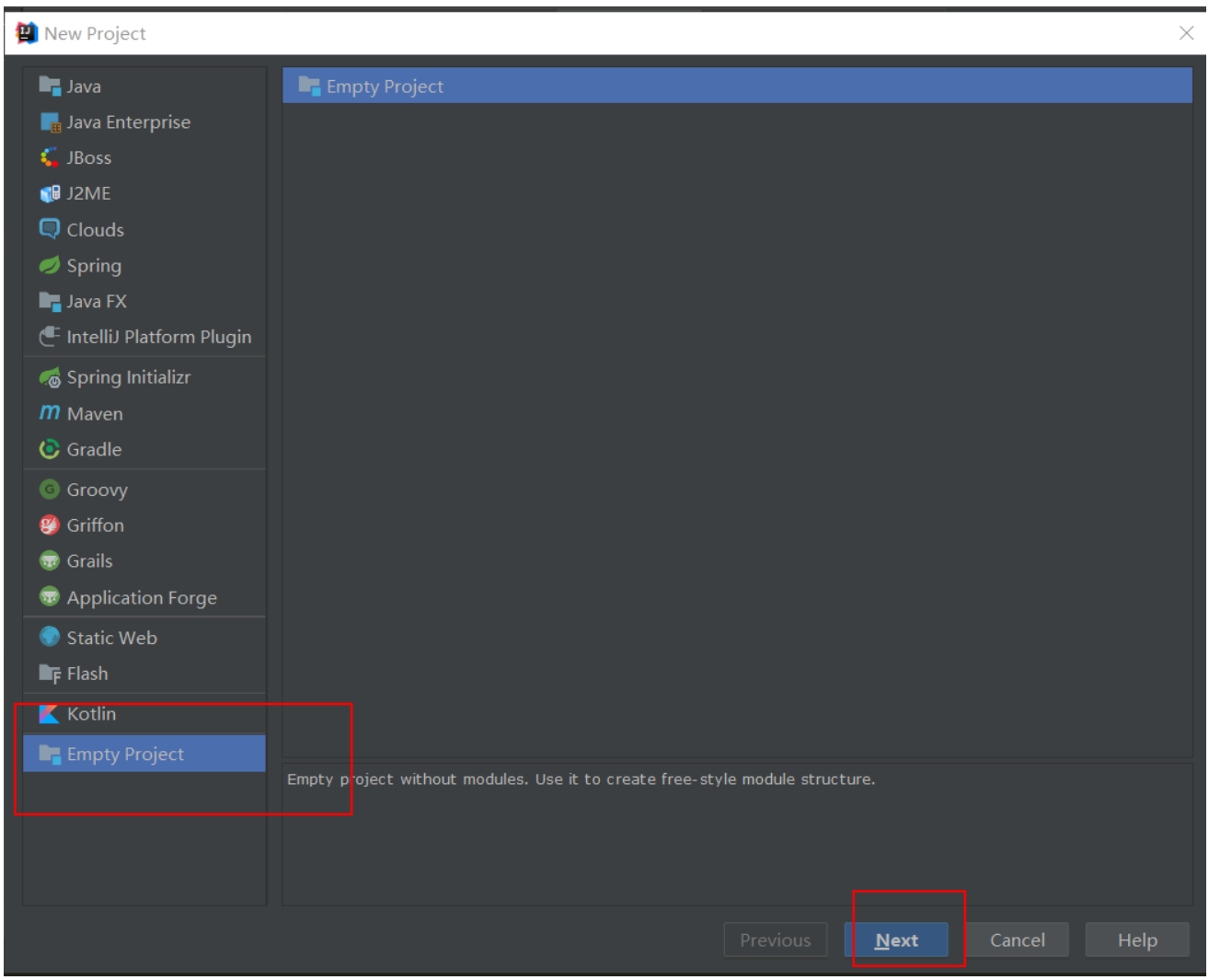
idea: IDEA ULTIMATE 2018.1

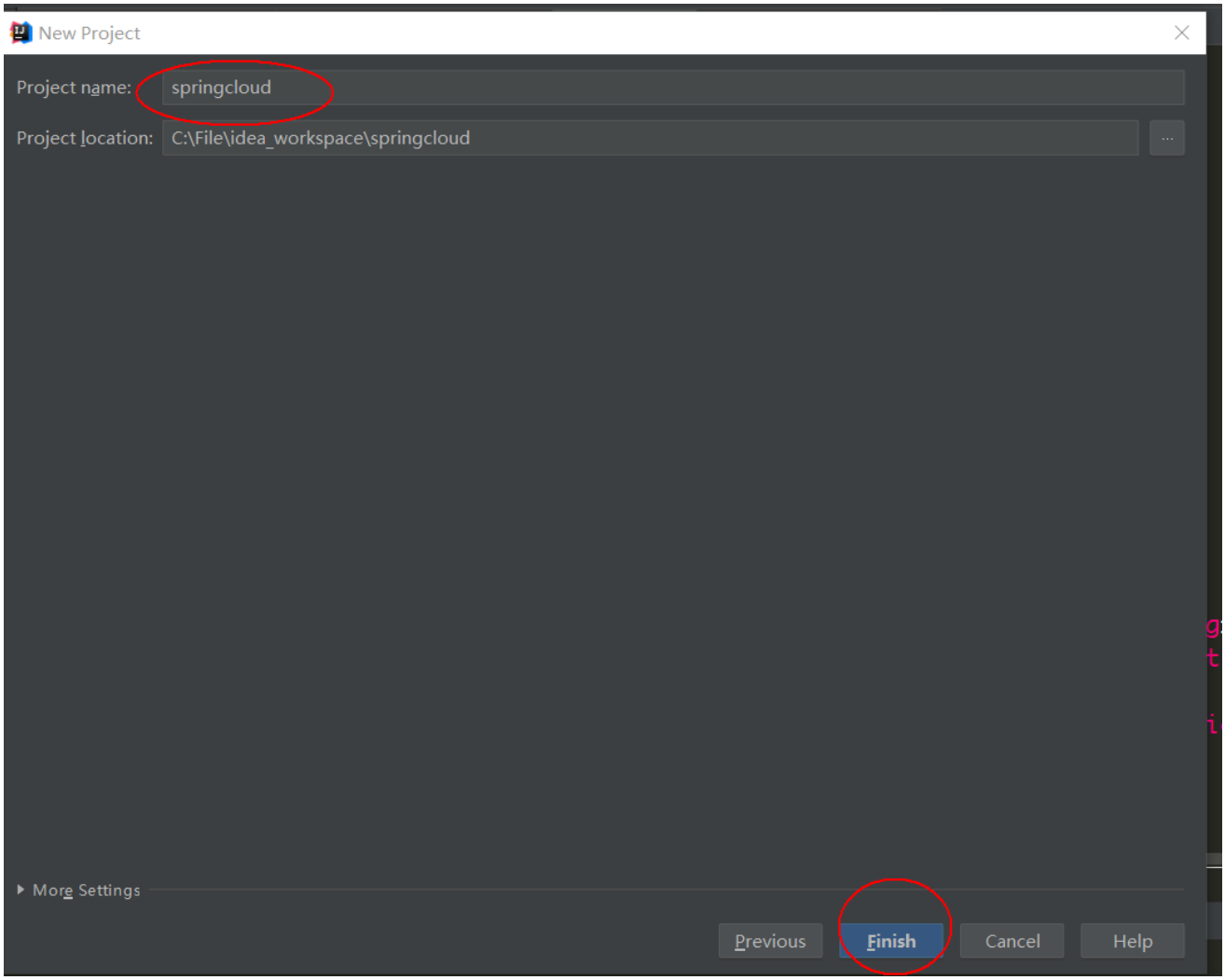
java: 1.8

maven: 3.5

二、创建工程（注意有坑）

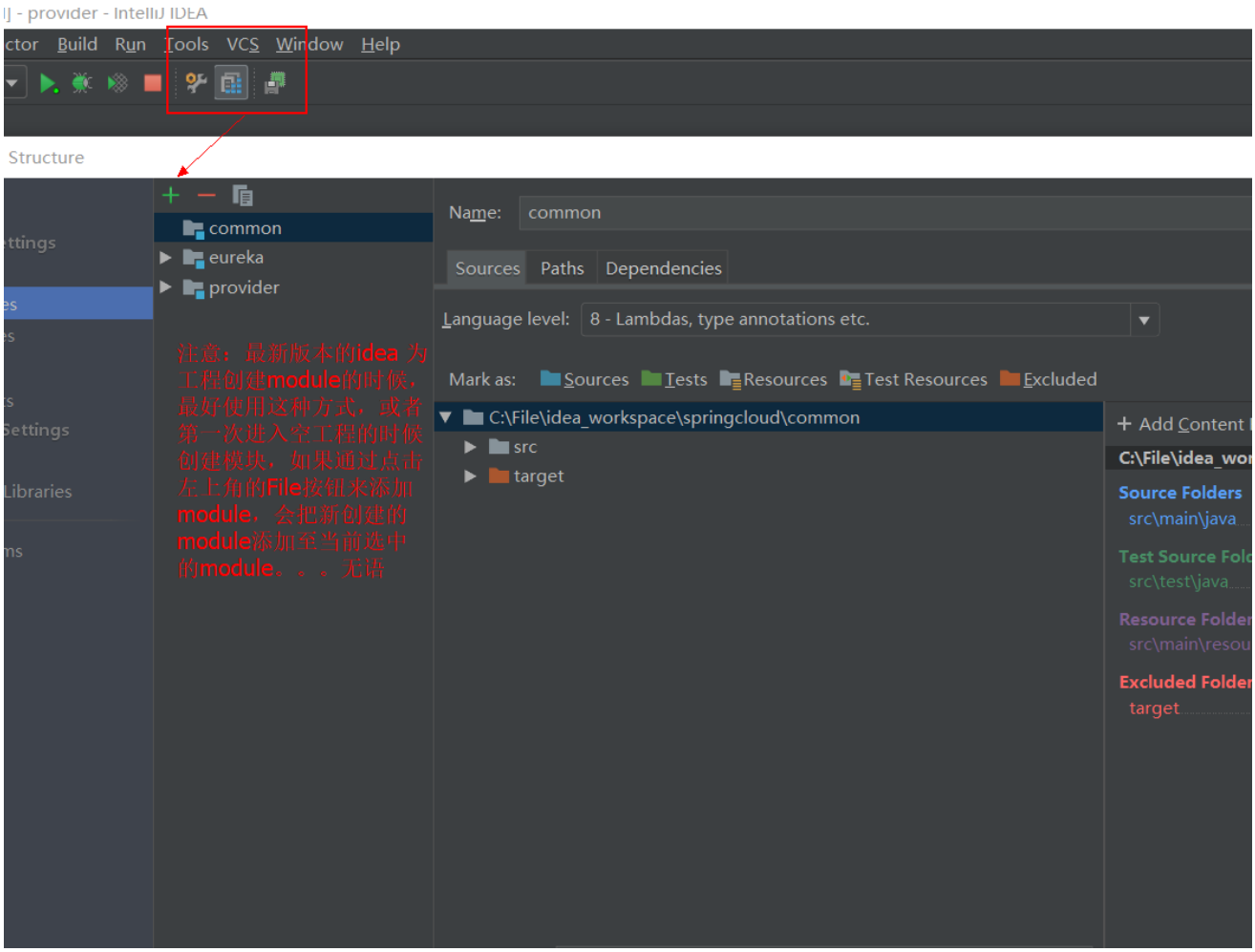
1. 首先创建一个空工程

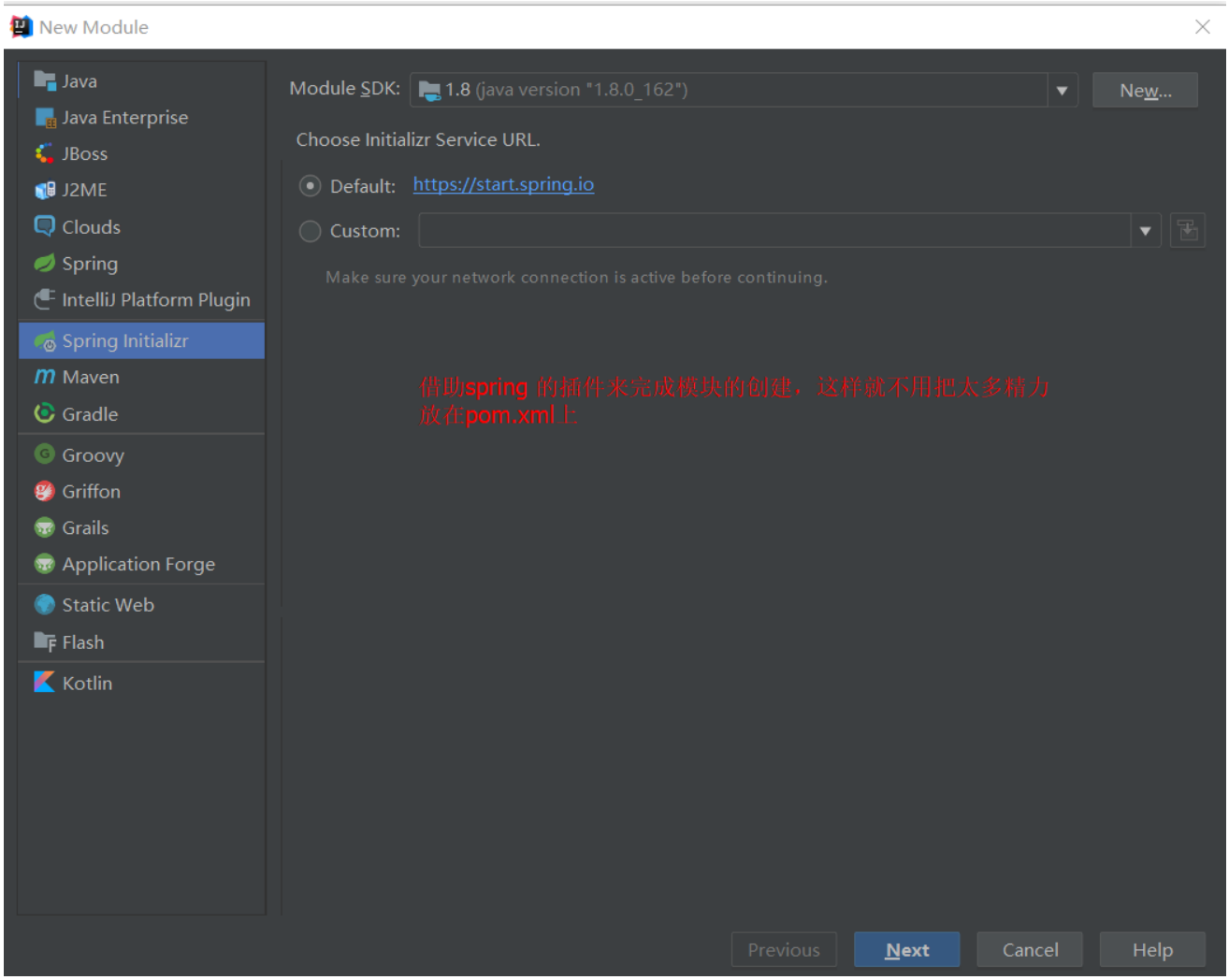


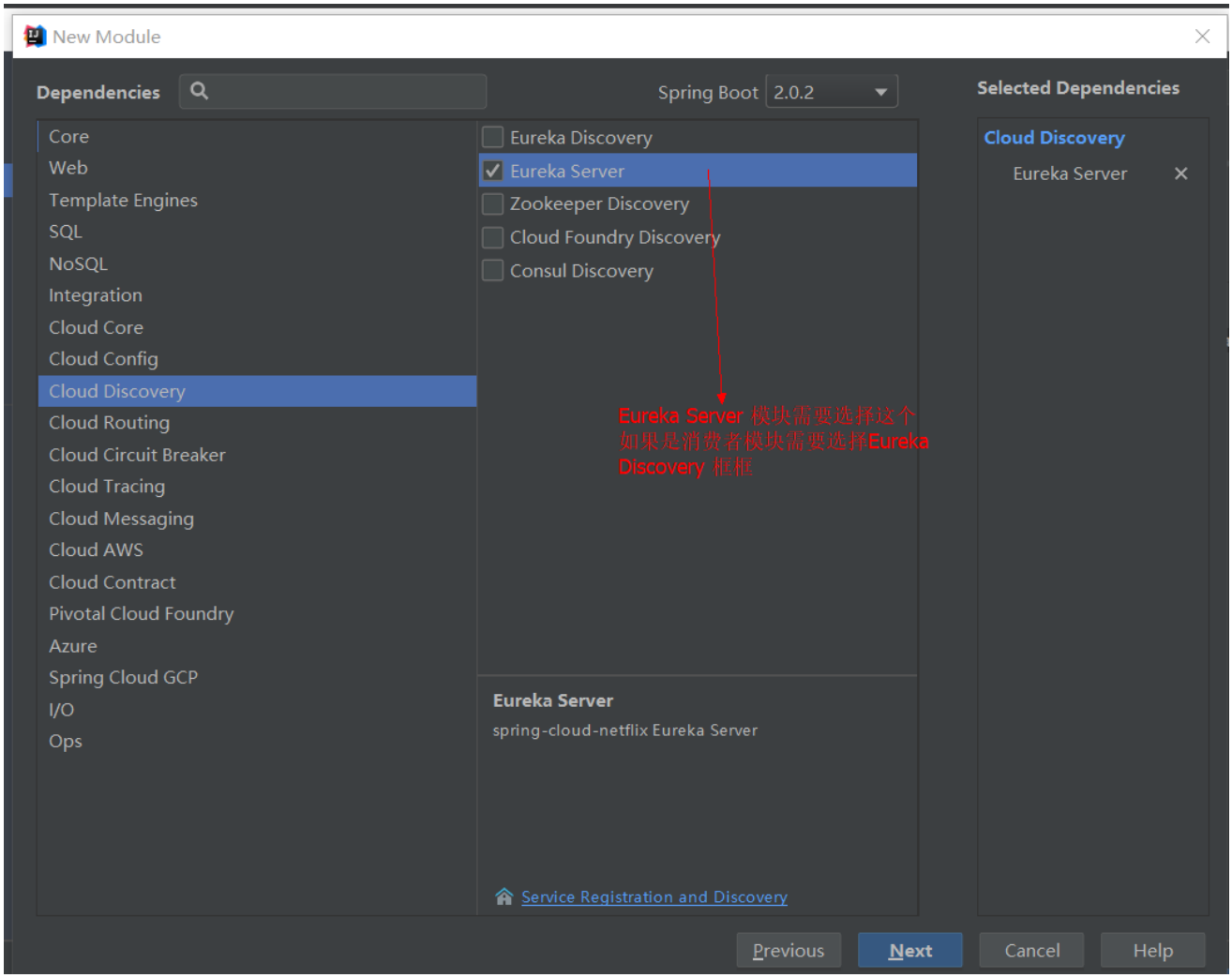


2. 创建并测试Eureka Server 的module

2.1 创建module


IJ - provider - IntelliJ IDEA
ctor Build Run Tools VCS Window Help
Structure
common
eureka
provider
注意：最新版本的idea 为工程创建module的时候，最好使用这种方式，或者第一次进入空工程的时候创建模块，如果通过点击左上角的File按钮来添加module，会把新创建的module添加至当前选中的module。。。无语
Name: common
Sources Paths Dependencies
Language level: 8 - Lambdas, type annotations etc.
Mark as: Sources Tests Resources Test Resources Excluded
C:\File\idea_workspace\springcloud\common
src
target
Add Content
Source Folders
src\main\java
Test Source Fold
src\test\java
Resource Folder
src\main\resou
Excluded Folder
target


New Module
Java
Java Enterprise
JBoss
J2ME
Clouds
Spring
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Griffon
Grails
Application Forge
Static Web
Flash
Kotlin
Module SDK:
1.8 (java version "1.8.0_162")
New...
Choose Initializr Service URL.
Default:
https://start.spring.io
Custom:
Make sure your network connection is active before continuing.
借助spring 的插件来完成模块的创建，这样就不用把太多精力
放在pom.xml上
Previous
Next
Cancel
Help



2.2 yml

```yml
server:
  port: 8761

eureka:
  instance:
    hostname: localhost
  client:
    registerWithEureka: false
    fetchRegistry: false
    serviceUrl:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

2.3 pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
```

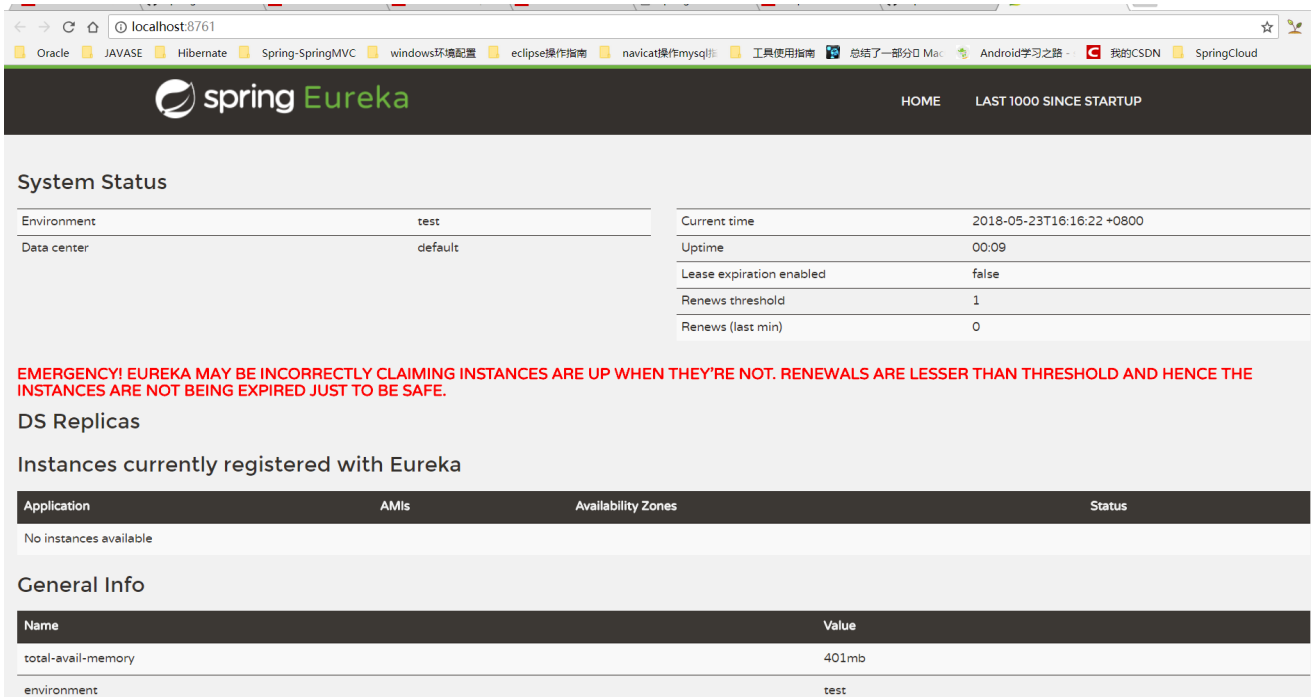
```
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>eureka</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>eureka</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <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.BUILD-SNAPSHOT</spring-cloud.version>
    </properties>

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

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

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</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>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>


</project>
```

2.4 主启动类

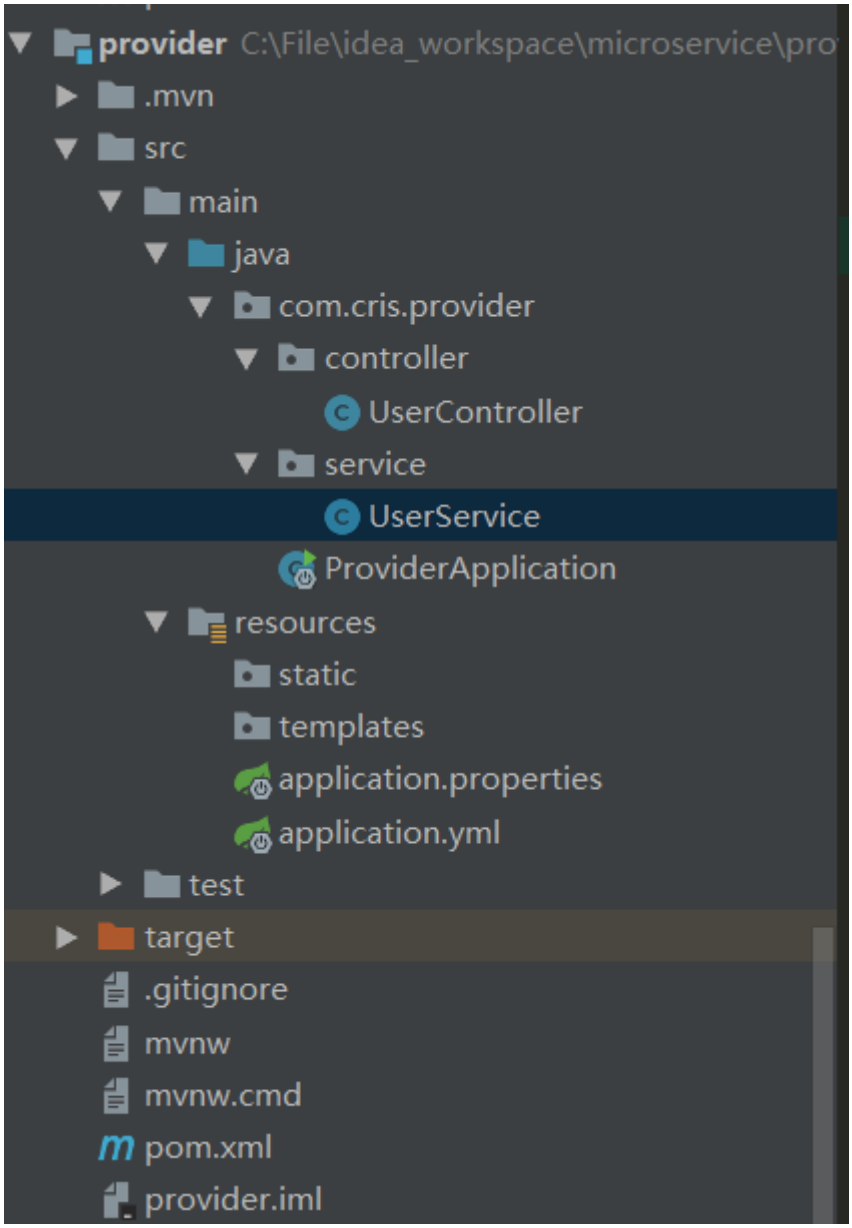
```
@SpringBootApplication
@EnableEurekaServer
public class EurekaApplication {

    public static void main(String[] args) {
        SpringApplication.run(EurekaApplication.class, args);
    }
}
```

2.5 启动测试



- [IDEA 安装lombok插件]

4. 创建provider 模块（添加web和eureka discovery 依赖）



4.1 application.yml

```yaml
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/     # 注册到指定的eureka server 服务器上
server:
  port: 8762
spring:
  application:
    name: service-provider                  # 服务名（很重要）
```

4.2 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>provider</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>provider</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <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.BUILD-SNAPSHOT</spring-cloud.version>
    </properties>

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

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

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</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>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>


</project>
```

4.3 service

```
/**
 * @ClassName UserService
 * @Description TODO
 * @Author zc-cris
 * @Version 1.0
 **/
@Service
public class UserService {
    public String get(){
        return "cris";
    }
}
```

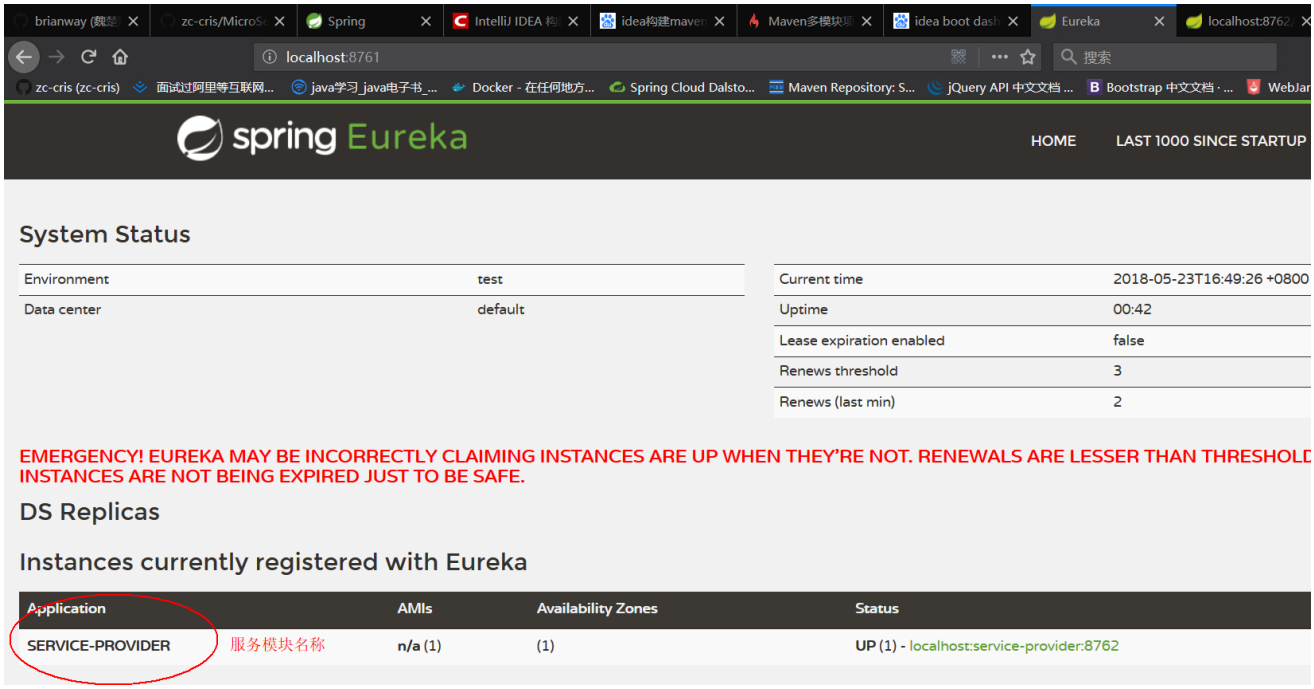
4.4 controller

```java
/**
 * @ClassName UserController
 * @Description TODO
 * @Author zc-cris
 * @Version 1.0
 **/
@RestController
public class UserController {

    @Autowired
    private UserService service;

    @Value("${server.port}")
    private String port;

    @GetMapping("/provider/get")
    public String get(){
        return service.get() + port;
    }
}
```

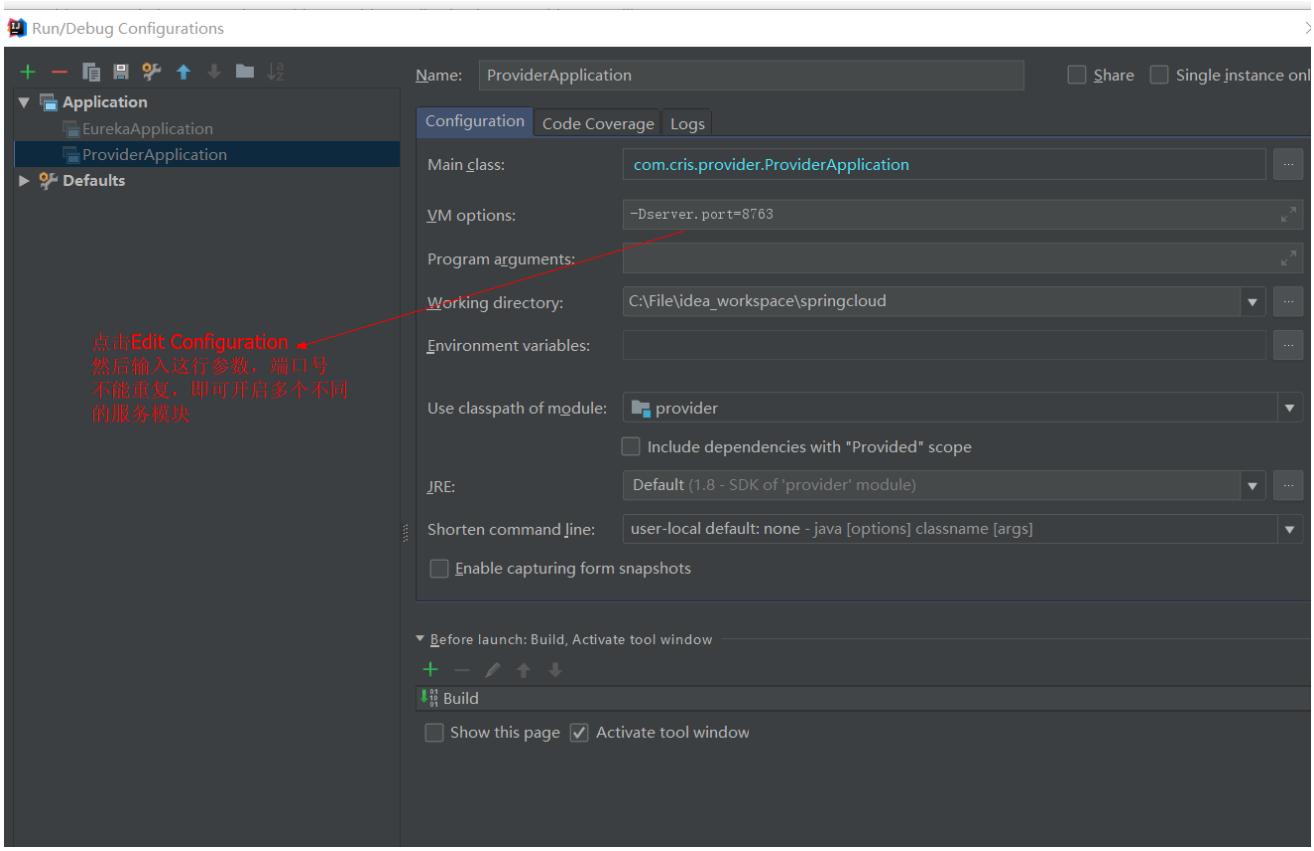
4.5 主启动类

```java
@SpringBootApplication
@EnableEurekaClient
public class ProviderApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProviderApplication.class, args);
    }
}
```

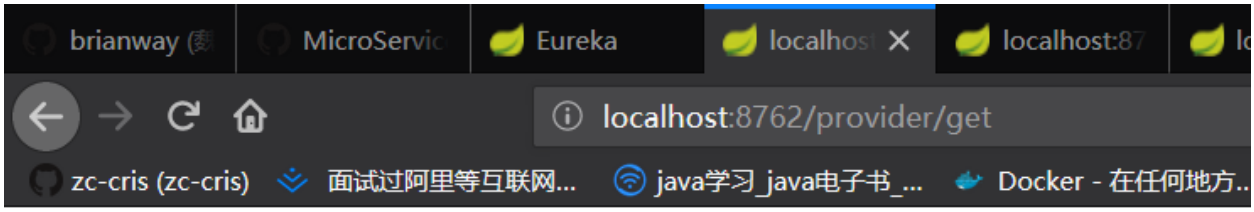
4.5 启动并测试



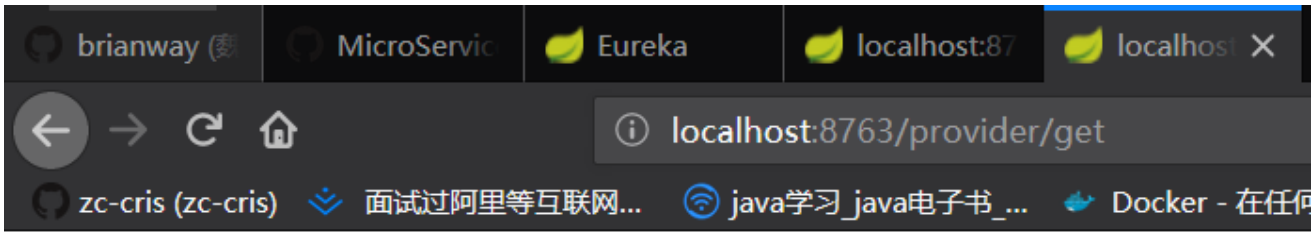
5. 快速启动多个不同端口的Provider 模块



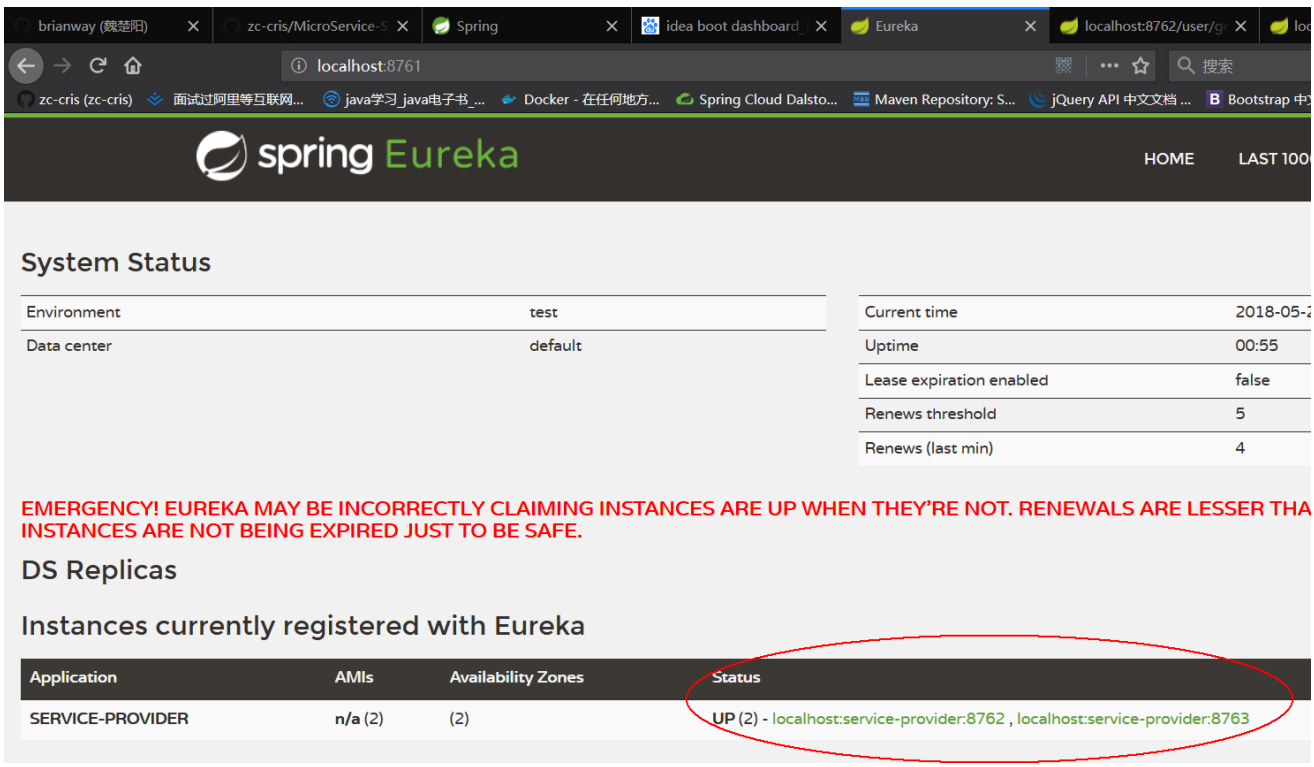
- 启动并测试



cris8762



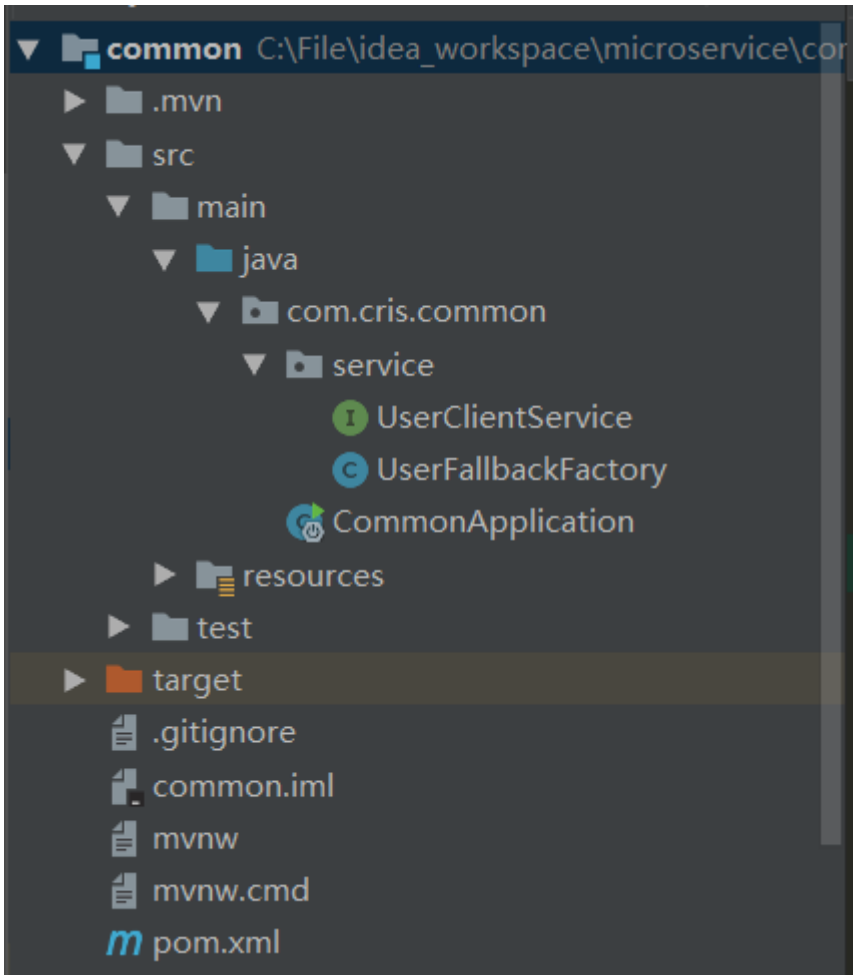
cris8763



6. 引入Feign + hystrix组件的公共模块

> 注意：这里不建议使用Ribbon 组件+RestTemplate 的方式来构建消费模块，原因在第一季已经阐述，这里直接上手Feign 组件以及hystrix

6.1 创建公共模块（添加Feign 依赖）



6.2 pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>common</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>common</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
```

```
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <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.BUILD-SNAPSHOT</spring-cloud.version>
    </properties>

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

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

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</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>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>

            </snapshots>
```

```
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>


</project>

```

- 注意，这里创建公共模块的时候直接使用spring initial工具创建，其他的模块（消费模块，生产模块）需要引用公共模块直接导入GAV 即可，而不是传统意义上的maven模块

6.2 新建映射到服务模块controller 的client并且进行hystrix 处理

```
/**
 * @ClassName UserClientService
 * @Description TODO
 * @Author zc-cris
 * @Version 1.0
 **/
@FeignClient(value = "SERVICE-PROVIDER", fallbackFactory = UserFallbackFactory.class)
public interface UserClientService {

    @GetMapping("/provider/get")
    public String get();

}
```

6.3 hystrix 处理逻辑(AOP)

```
/**
 * @ClassName UserFallbackFactory
 * @Description TODO
 * @Author zc-cris
 * @Version 1.0
 **/
@Component
public class UserFallbackFactory implements FallbackFactory<UserClientService> {

    @Override
    public UserClientService create(Throwable throwable) {
        return new UserClientService() {
            @Override
            public String get() {

                return "当前服务模块不可用";
```

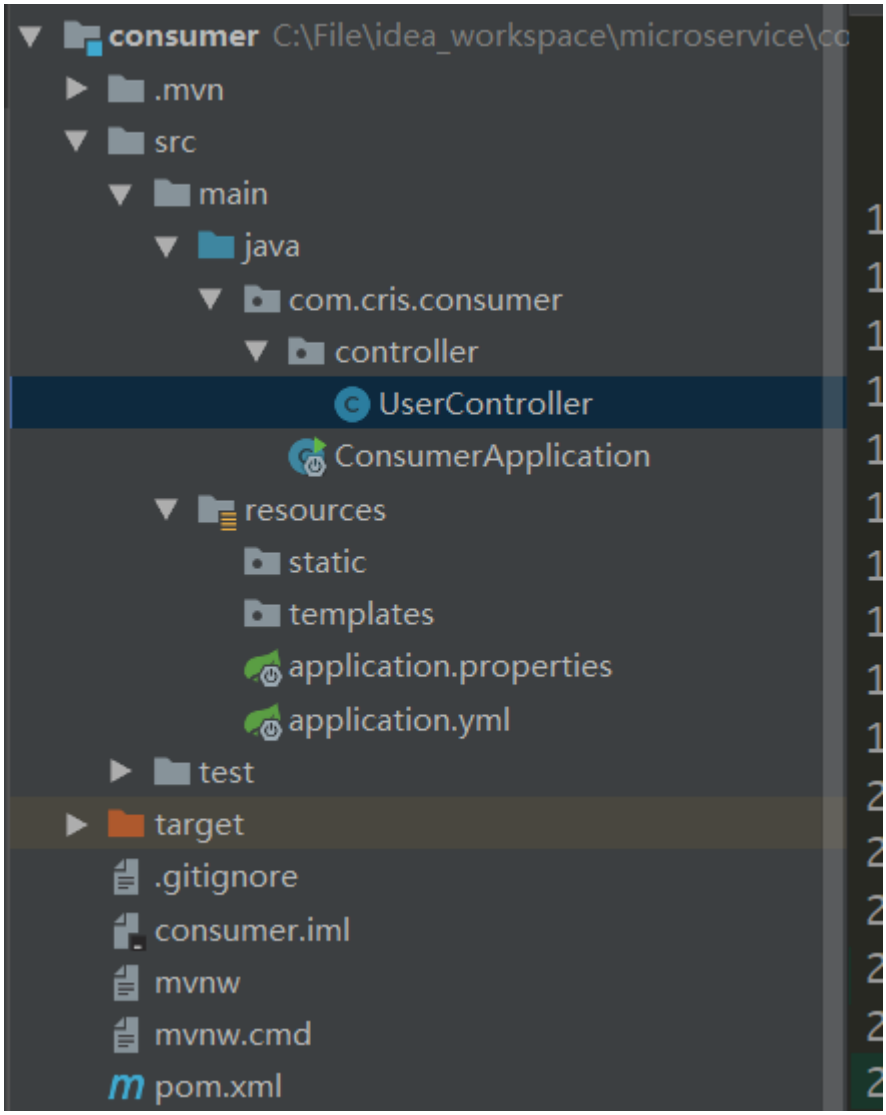
```
            }
        };
    }
}
```

6.4 主启动类

```
@SpringBootApplication
public class CommonApplication {

    public static void main(String[] args) {
        SpringApplication.run(CommonApplication.class, args);
    }
}
```

6.5 mvn install

7. 新建消费模块（添加Feign, web, eureka discovery 依赖）



7.1 pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>consumer</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>consumer</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
```

```
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <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.BUILD-SNAPSHOT</spring-cloud.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>com.cris</groupId>
            <artifactId>common</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

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

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</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>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>


</project>
```

7.2 application.yml

```yml
eureka:
  client:
    register-with-eureka: false   # 试试不把自己注册到eureka server 中，是否可以拿到eureka server上的服务信息列表（答案是可以的）
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
server:
  port: 8765
spring:
  application:
    name: service-feign
feign:
  hystrix:
      enabled: true   // 支持熔断处理
```

7.3 controller

```
/**
 * @ClassName UserController
 * @Description TODO
 * @Author zc-cris

 * @Version 1.0
```

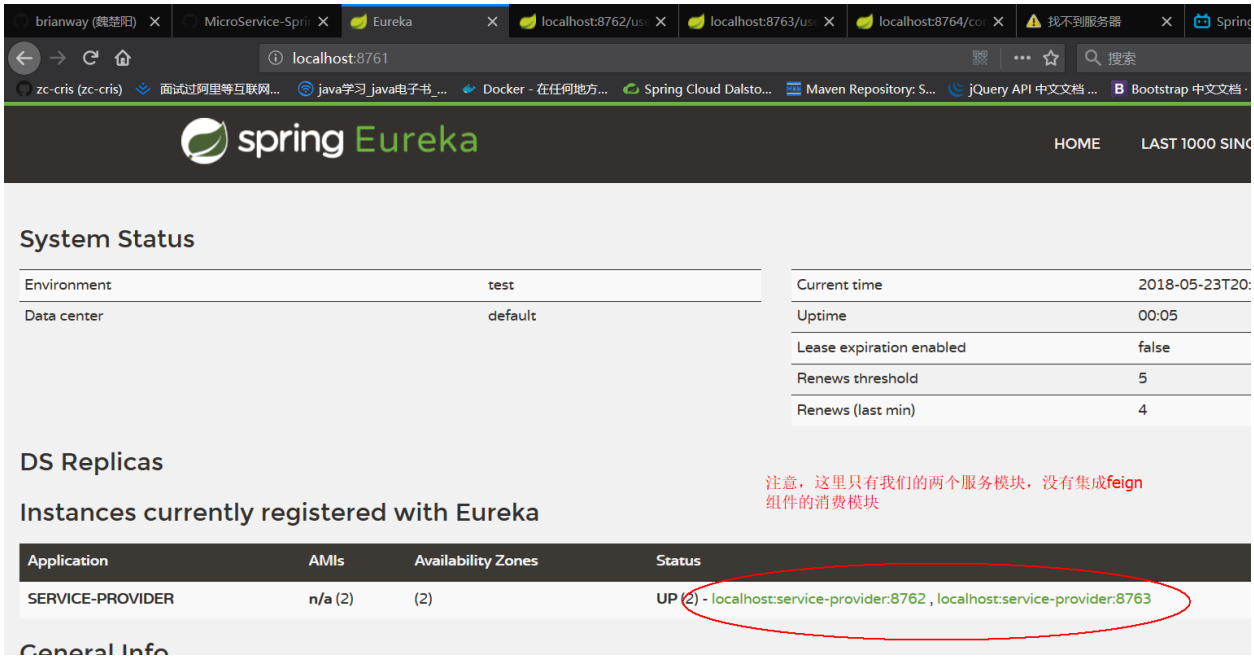
```
 **/
@RestController
public class UserController {

    @Autowired
    private UserClientService service;

    @GetMapping("/consumer/user/get")
    public String get(){
        return service.get();
    }

    @GetMapping("/consumer/get")
    public String getUser(){
        return "cris";
    }

}
```

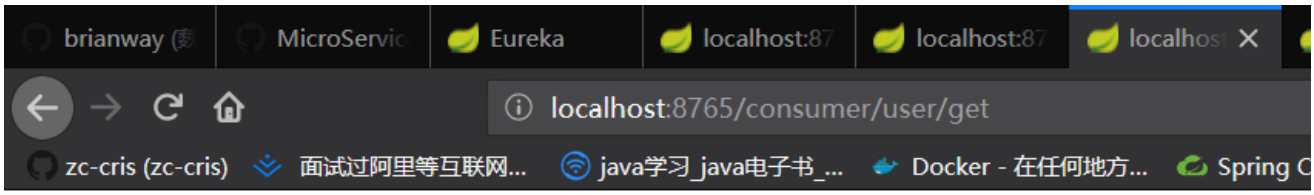
7.4 主启动类

```
@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients(basePackages = {"com.cris"})
@ComponentScan("com.cris")
public class ConsumerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConsumerApplication.class, args);
    }
}
```

7.5 启动并测试

- 先启动eureka server，然后启动我们集成了feign 组件的消费模块

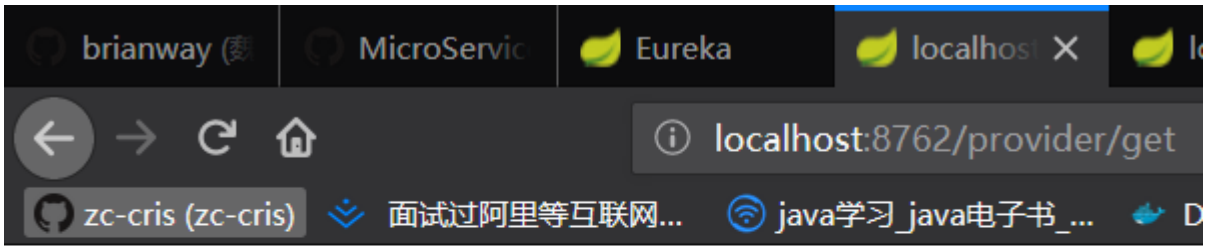




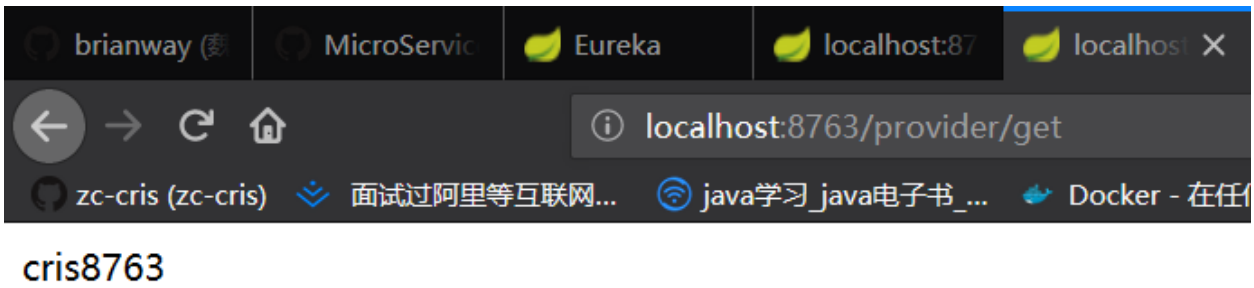
当前服务模块不可用

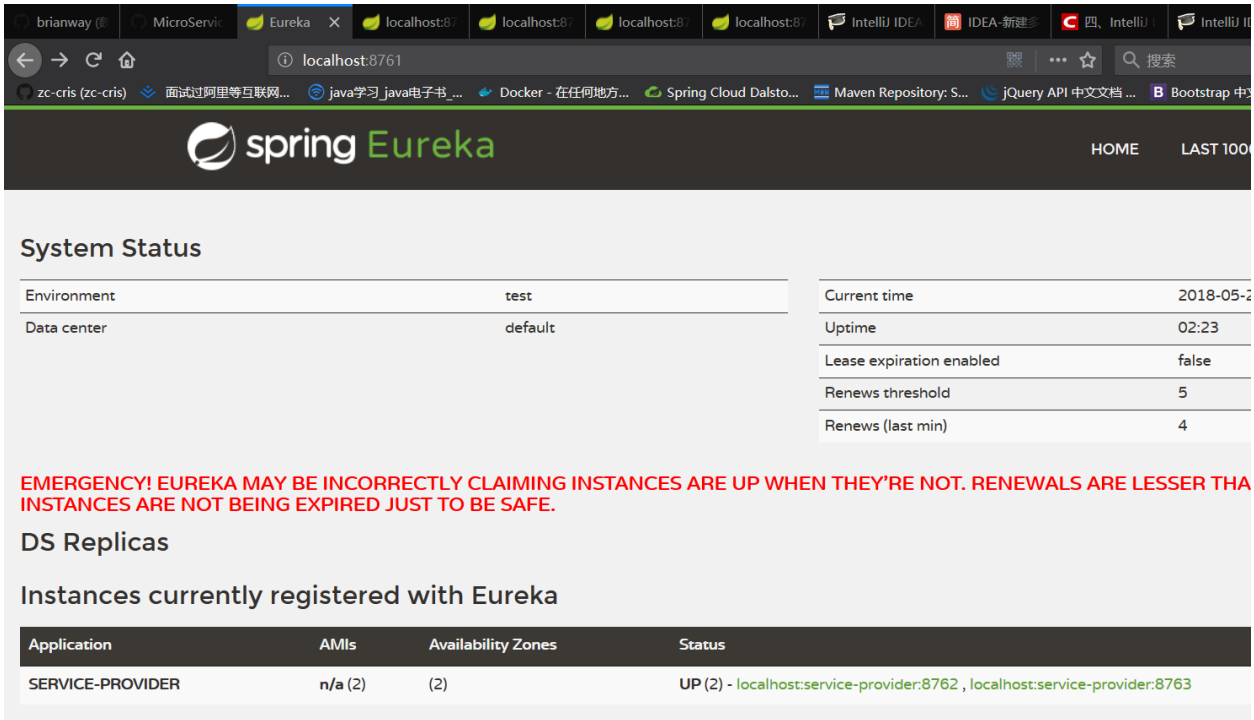
可以看到我们的hystrix 处理单元起作用了，因为服务模块并没有启动，模拟降级或者熔断处理

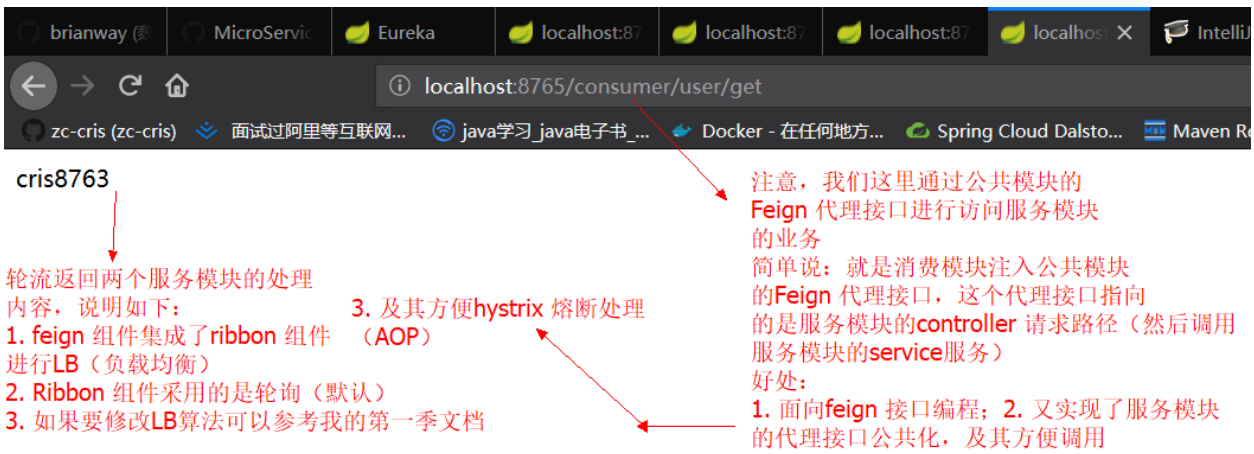
- 启动我们的服务模块（多端口）再测试



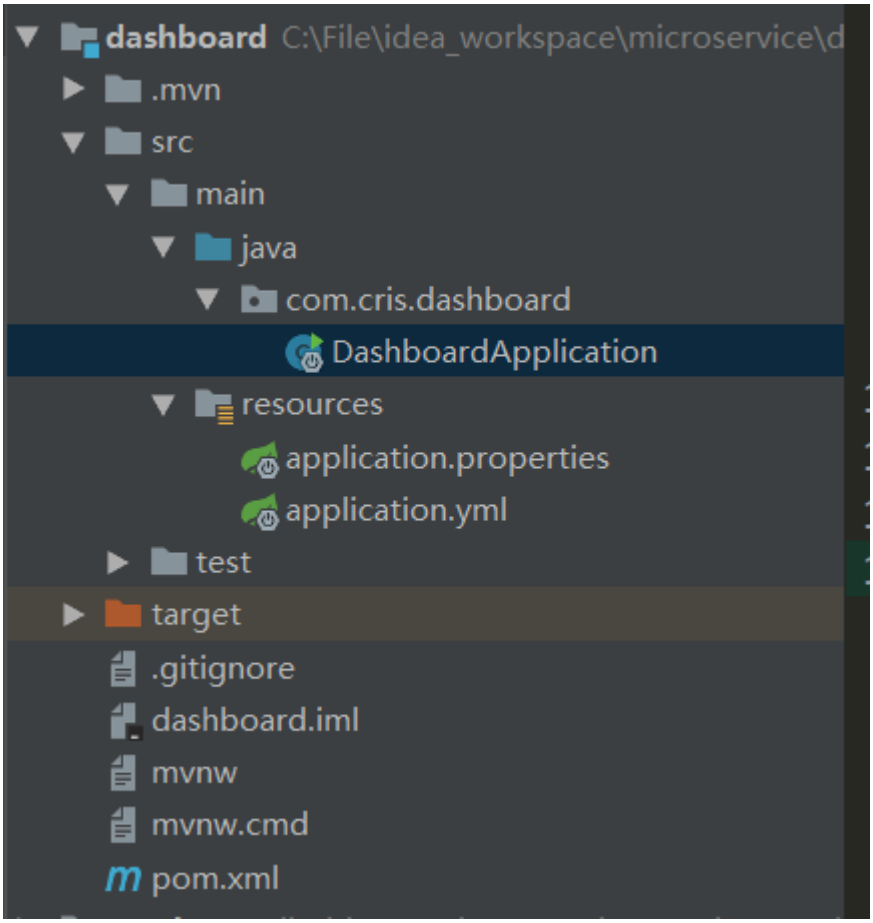
cris8762







8. 新建监控模块（引入hystrix Dashboard依赖）



8.1 pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>dashboard</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>dashboard</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
```

```
22          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
23          <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
24          <java.version>1.8</java.version>
25          <spring-cloud.version>Finchley.BUILD-SNAPSHOT</spring-cloud.version>
26      </properties>
27
28      <dependencies>
29
30          <dependency>
31              <groupId>org.springframework.cloud</groupId>
32              <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
33          </dependency>
34          <dependency>
35              <groupId>org.springframework.cloud</groupId>
36              <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
37          </dependency>
38          <dependency>
39              <groupId>org.springframework.boot</groupId>
40              <artifactId>spring-boot-starter-web</artifactId>
41          </dependency>
42          <dependency>
43              <groupId>org.springframework.cloud</groupId>
44              <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
45          </dependency>
46
47          <dependency>
48              <groupId>org.springframework.boot</groupId>
49              <artifactId>spring-boot-starter-test</artifactId>
50              <scope>test</scope>
51          </dependency>
52      </dependencies>
53
54      <dependencyManagement>
55          <dependencies>
56              <dependency>
57                  <groupId>org.springframework.cloud</groupId>
58                  <artifactId>spring-cloud-dependencies</artifactId>
59                  <version>${spring-cloud.version}</version>
60                  <type>pom</type>
61                  <scope>import</scope>
62              </dependency>
63          </dependencies>
64      </dependencyManagement>
65
66      <build>
67          <plugins>
68              <plugin>
69                  <groupId>org.springframework.boot</groupId>
70                  <artifactId>spring-boot-maven-plugin</artifactId>
71              </plugin>
72          </plugins>
73      </build>

74
```

```
    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>


</project>
```

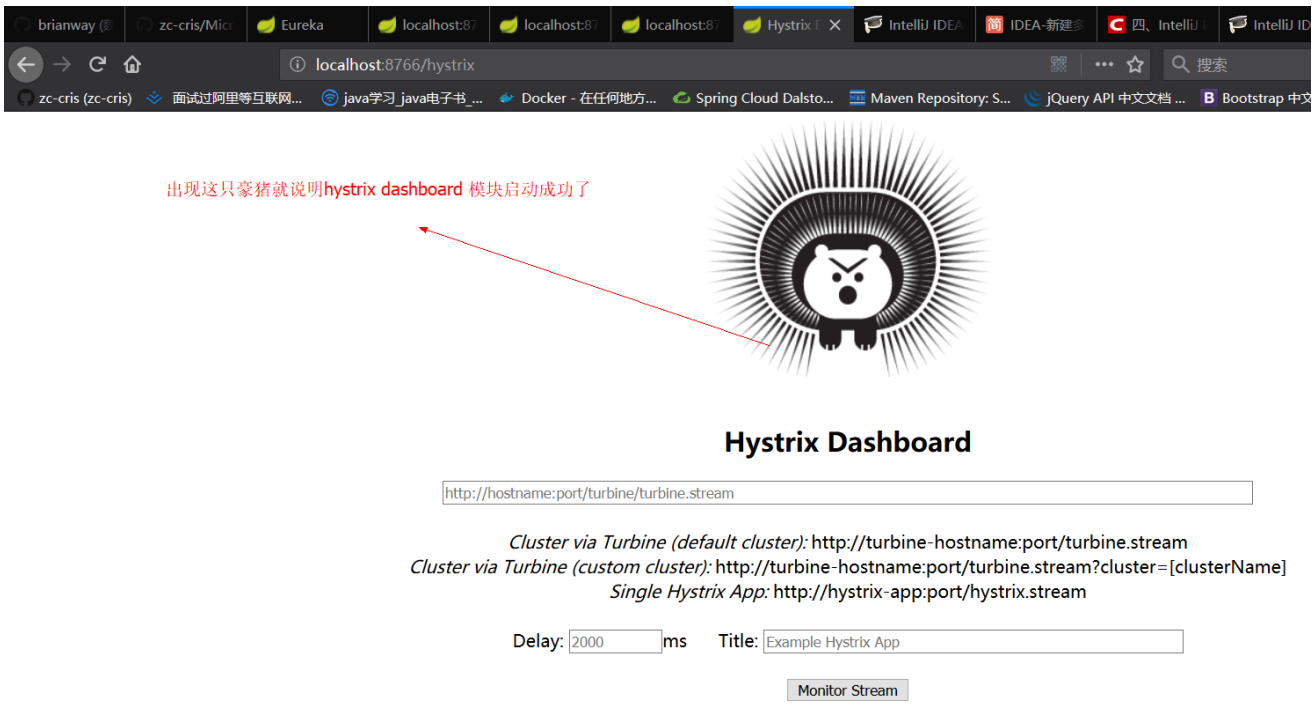
8.2 主启动类

```
@SpringBootApplication
@EnableHystrixDashboard
@EnableDiscoveryClient  // 其实和@EnableEurekaClient 功能相似，但是多用于其他服务中心
public class DashboardApplication {

    public static void main(String[] args) {
        SpringApplication.run(DashboardApplication.class, args);
    }
}
```

8.3 yml文件

```
server:
  port: 8766
spring:
  application:
    name: service-dashboard
```

8.4 启动并自测

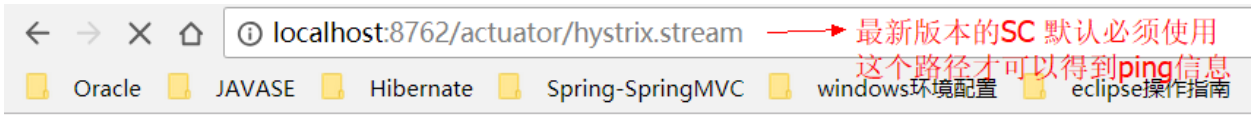


8.5 服务模块添加actuator 监控组件和hystrix 依赖

```
        <!-- actuator 监控信息完善 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <!-- 引入hystrix 熔断器组件 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
        </dependency>
```

8.6 启动服务模块测试hystrix dashboard 组件（有坑）

- 因为使用的是最新版本的SpringCloud（Finchley.BUILD-SNAPSHOT）和SpringBoot（2.0.2.RELEASE）和之前使用1.5版本的时候不一样，所以填坑如下




```
data:
{"type":"HystrixCommand","name":"get","group":"UserController","currentTime":1527162182071,"isCircuitBre
:0,"rollingCountEmit":0,"rollingCountExceptionsThrown":0,"rollingCountFailure":0,"rollingCountFallbackEm
ccess":0,"rollingCountResponsesFromCache":0,"rollingCountSemaphoreRejected":0,"rollingCountShortCircuite
llingMaxConcurrentExecutionCount":1,"latencyExecute_mean":1,"latencyExecute":{"0":1,"25":1,"50":1,"75":1
{"0":1,"25":1,"50":1,"75":1,"90":1,"95":1,"99":1,"99.5":1,"100":1},"propertyValue_circuitBreakerRequestV
entage":50,"propertyValue_circuitBreakerForceOpen":false,"propertyValue_circuitBreakerForceClosed":false
onThreadTimeoutInMilliseconds":1000,"propertyValue_executionTimeoutInMilliseconds":1000,"propertyValue_e
cutionIsolationSemaphoreMaxConcurrentRequests":10,"propertyValue_fallbackIsolationSemaphoreMaxConcurrent
ertyValue_requestLogEnabled":true,"reportingHosts":1,"threadPool":"UserController"}

data:
```

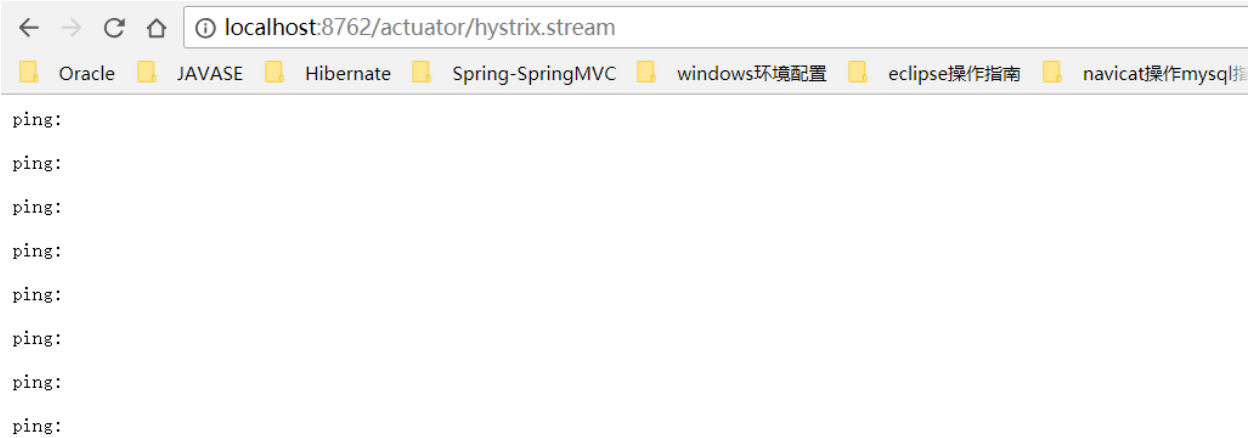

  - 如果说访问以上路径出现404，[请参考这篇文章]

- 我的解决方案是：

  服务模块的yml文件

```yaml
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
server:
  port: 8762
spring:
  application:
    name: service-provider

management:
  endpoints:
    web:
      exposure:
        include: "*"
```

- 然后如果出现进入dashboard 管理界面一直loading 的情况，请参考这篇文章
- 但是问题又来了，因为我是把熔断，降级处理全部抽取出来放在公共的feign 代理接口中进行处理的，所以我配置完以后还是一直loading... --!



- 解决方案：其实很简单，直接修改服务模块的服务方法，将其标记为@HystrixCommand(),因为我们的hystrix dashboard 不是所有业务方法都会监控的，只有标记了需要熔断或是降级处理的方法才会被监控起来

```
/**
 * @ClassName UserController
 * @Description TODO
 * @Author zc-cris
 * @Version 1.0
 **/
@RestController
public class UserController {

    @Autowired
    private UserService service;

    @Value("${server.port}")
    private String port;

    @GetMapping("/provider/get")
    @HystrixCommand()
    public String get(){
        return service.get() + port;
    }
}
```

- 然后修改provider 模块的主启动类

```java
@EnableEurekaClient // 这个服务模块启动后自动注册进Eureka Server
@SpringBootApplication
@EnableCircuitBreaker // 对hystrix 熔断器的支持
@EnableHystrix
public class ProviderApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProviderApplication.class, args);
    }
}
```

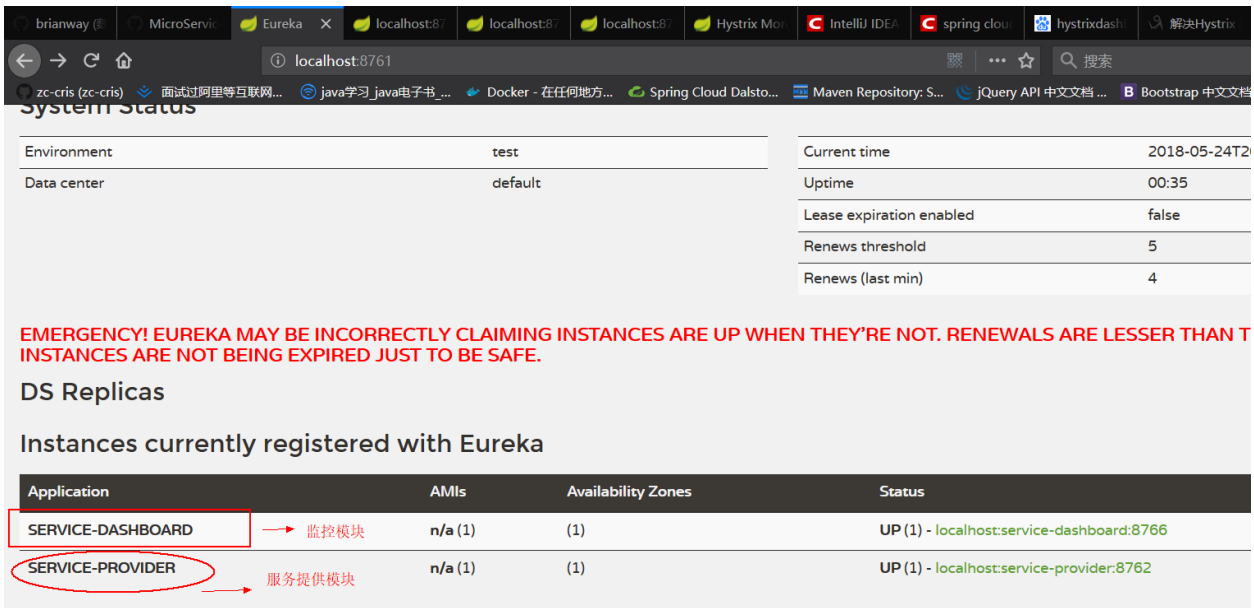
- 再修改provider 模块的pom.xml

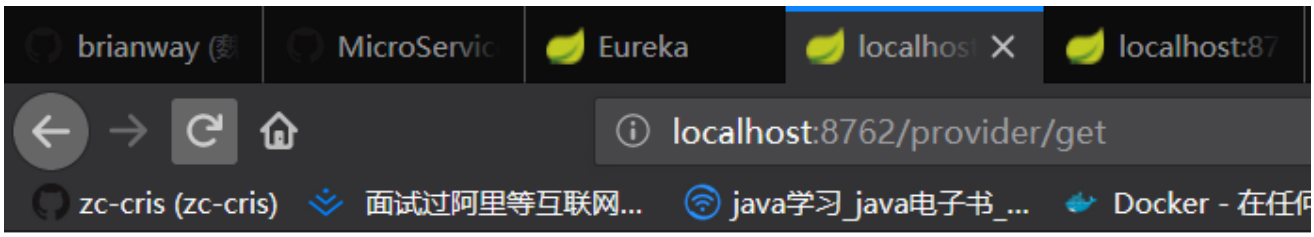
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>provider</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>provider</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <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.BUILD-SNAPSHOT</spring-cloud.version>
    </properties>

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

```
        <!-- 引入hystrix 熔断器组件 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
        </dependency>
        <!-- actuator 监控信息完善 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>

    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</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>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>

                <enabled>true</enabled>
```

```
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
</project>
```

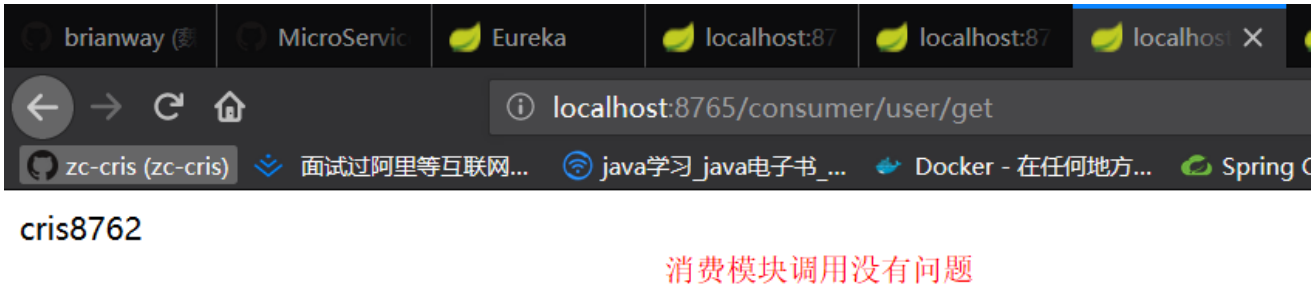
- 最后启动eureka server 模块，customer 模块，provider 模块，dashboard 模块

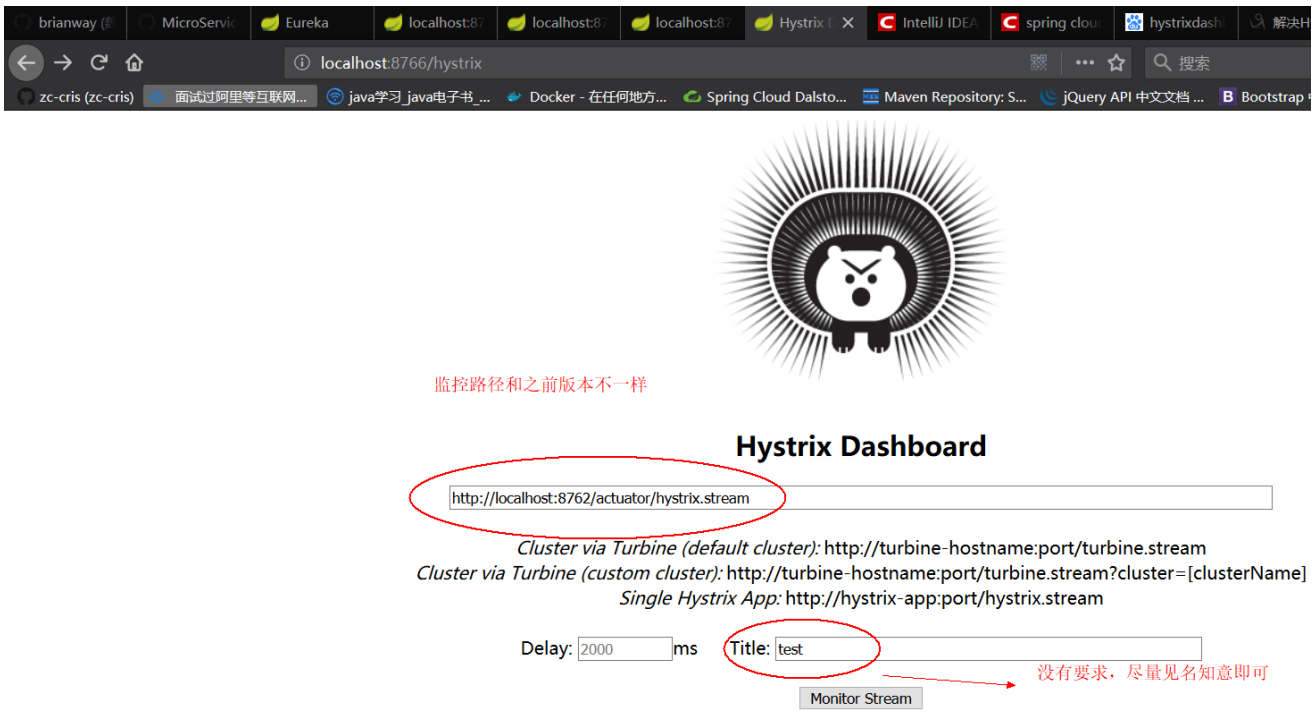




cris8762

服务模块自测没有问题





- 进入dashboard 管理界面，如果出现loading... ，刷新消费模块调用服务模块的url即可

9. 总结

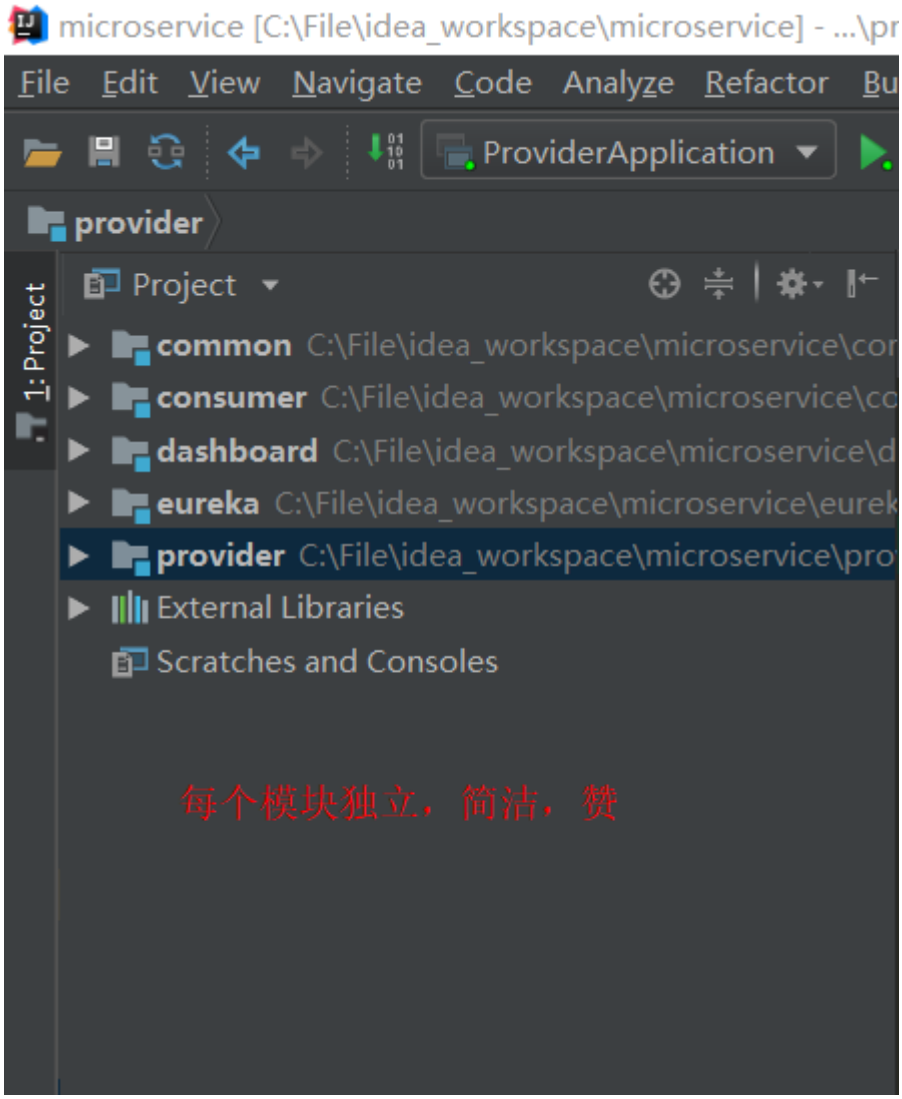
至此，我们完成了公共模块（feign代理接口 + hystrix集中处理）+ 服务模块（hystrix标记） + eureka server（集群） + 消费模块（注入feign代理接口）+ hystrix dashboard 监控模块的整体搭建；踩的坑如实记录下来，方便以后查看。

多说一句，最开始使用IDEA 搭建这个项目的时候，本来是按照eclipse 上的模式（Maven 多模块构建的方式）进行搭建，但是始终有问题，不是配置文件读取失败，就是classNotFind。。。测试了好久，最后使用如下方式进行SP 的项目搭建

I. 新建一个空工程（顶级工程）

II. 添加模块（全部使用spring initialzr插件）

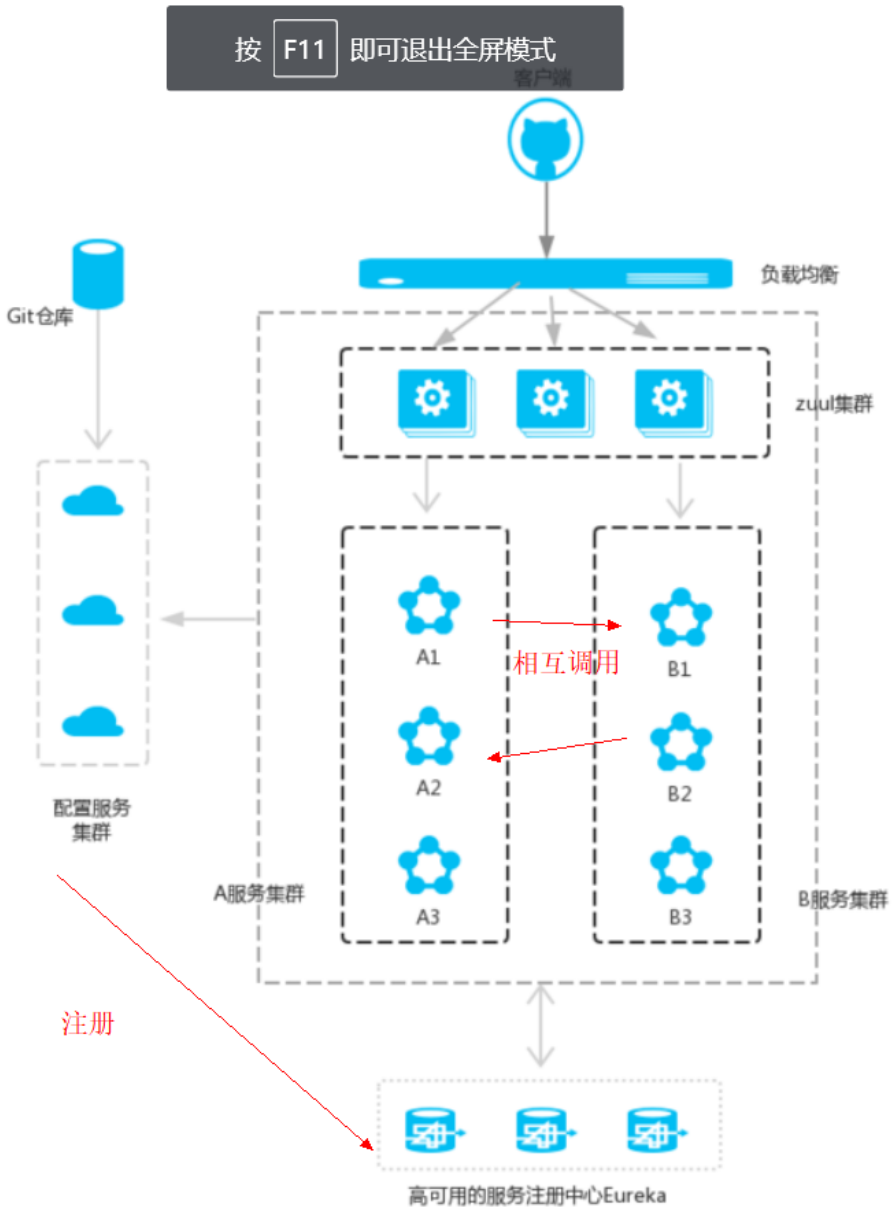
III. 公共组件等放在common 模块，其他模块如果需要可以引用这个common 模块（消费模块或者服务模块等），每个模块各自独立，而引用jar包全部放在空工程下，特别方便开发



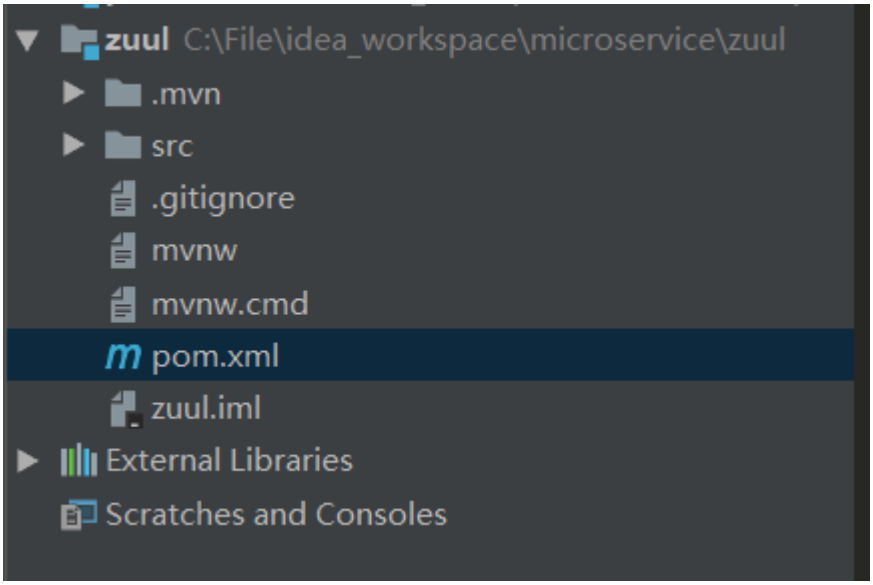
> 最后，本节的第三点没有，因为排版失误 –

三、其他组件添加和完善

1. Zull（网关组件）



1.1 新建一个Zuul 模块（引入web, zuul, eureka discovery 依赖）



1.2 pom.xml(亮点：添加actuator 依赖和maven-resource 处理插件，完成该模块的具体信息展示)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>zuul</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>zuul</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <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.BUILD-SNAPSHOT</spring-cloud.version>
    </properties>

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

```
        </dependency>
        <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-zuul</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!-- actuator 监控信息完善 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <finalName>zuul</finalName>
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <filtering>true</filtering>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <configuration>
                    <delimiters>
                        <delimiter>$</delimiter>
                    </delimiters>
                </configuration>
            </plugin>

```

```
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>

</project>
```

1.3 yml文件

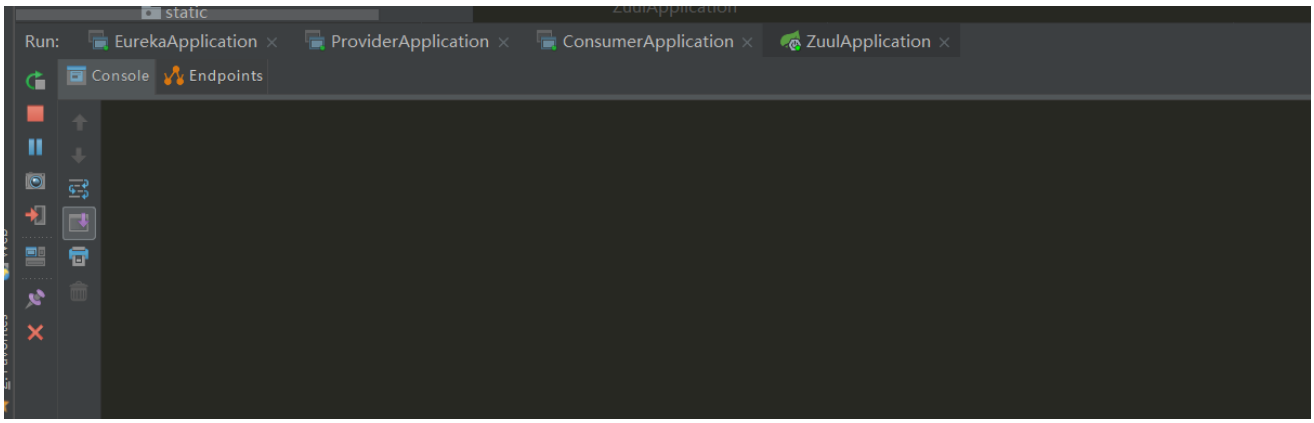
```yml
server:
  port: 8766    # 端口设置

spring:
  application:
    name: microservicecloud-zuul-gateway                # 注册到Eureka 以及对外暴露的微服务名字(重要)

eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/    # 将我们的zuul服务注册到eureka server 的地址
  instance:
    prefer-ip-address: true                   #注册到服务中心使用ip进行注册（服务名称显示的ip详情）


info:
  app.name: cris-microservicecloud
  company.name: www.cris.com

  app.programmer: zc-cris
```

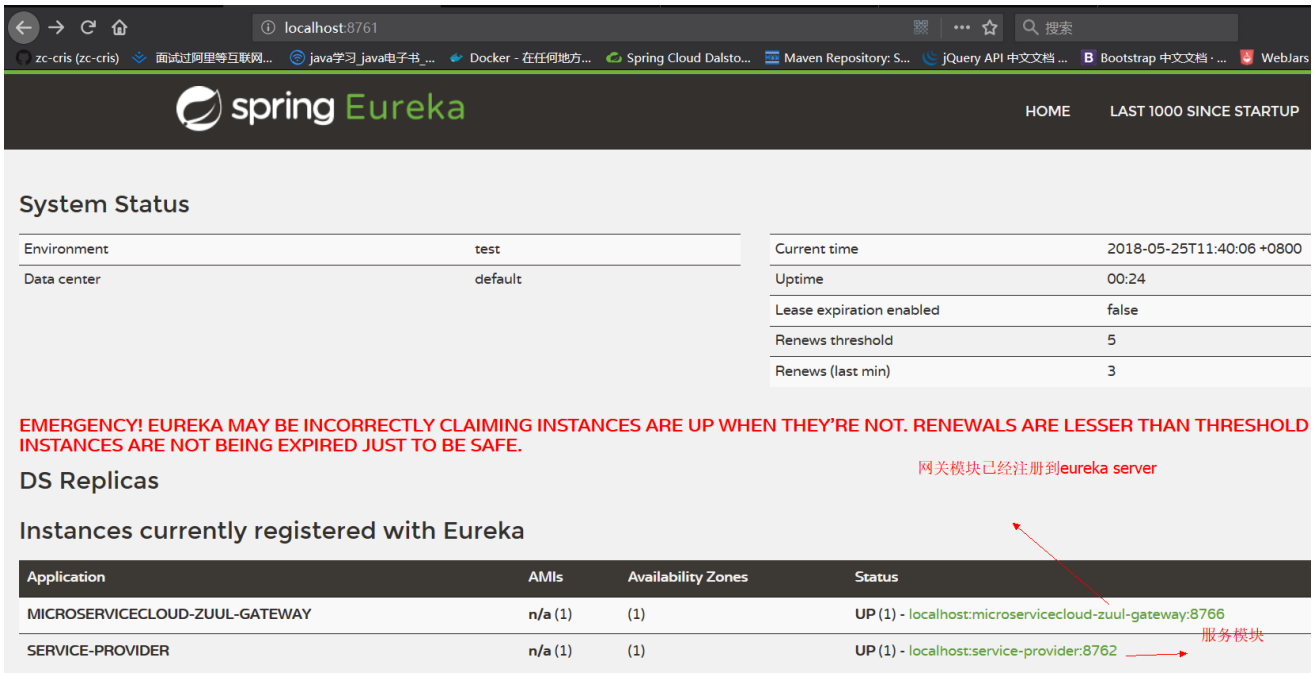
```
build.artifactId: $project.artifactId$
build.version: $project.version$
```

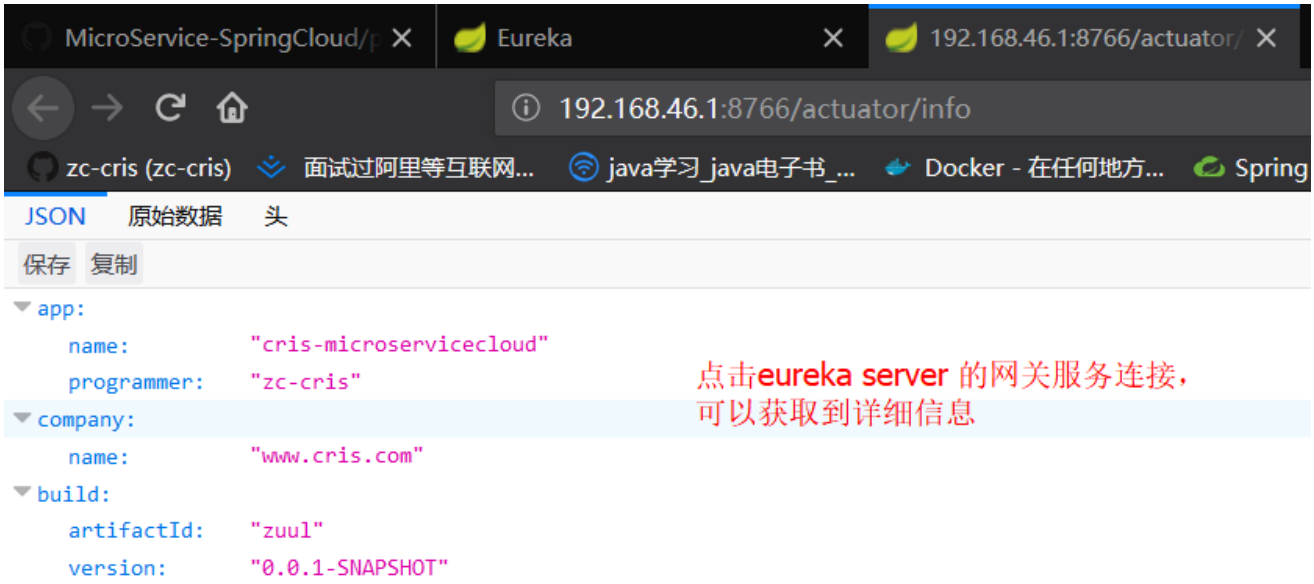
1.4 主启动类

```
@SpringBootApplication
@EnableZuulProxy            // 开启zuul 代理
@EnableEurekaClient     // 将服务注册到eureka server
public class ZuulApplication {

    public static void main(String[] args) {
        SpringApplication.run(ZuulApplication.class, args);
    }
}
```

1.5 启动测试



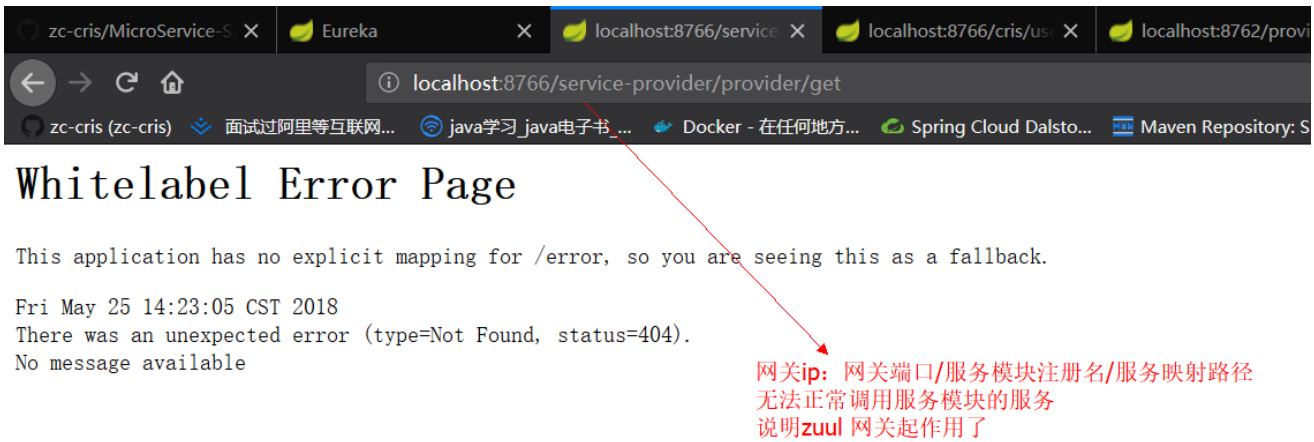


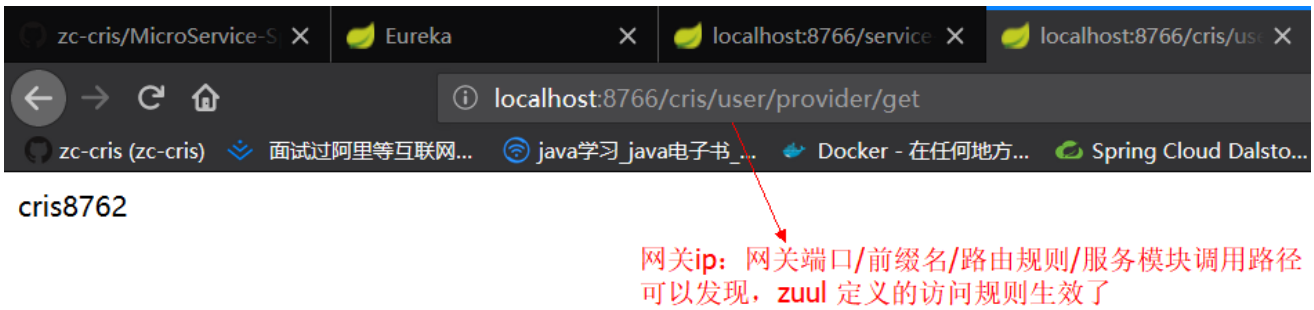


1.6 zuul 的路由功能

- 修改yml文件，定义服务访问规则

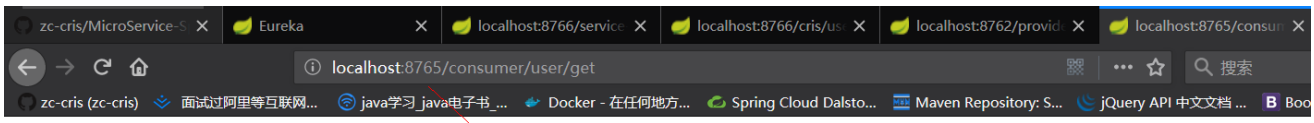
```yml
# zuul代理指定服务模块，将其对外暴露的服务名隐藏换个马甲
zuul:
  prefix: /cris   # 设置公共的访问前缀
  ignored-services: "*"   # 隐藏所有服务路径
  routes:
    user.serviceId: service-provider  # 指定注册在eureka server 上的服务名
    user.path: /user/**     # 必须通过以下路径访问
```



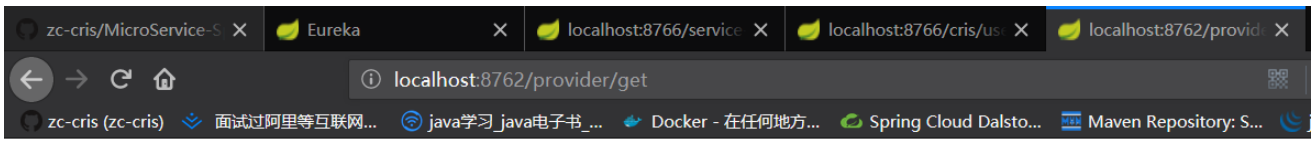


网关ip：网关端口/前缀名/路由规则/服务模块调用路径
可以发现，zuul 定义的访问规则生效了

注意：zuul主要针对的是外界访问我们的微服务架构，及其进行访问控制和过滤，但是我们自己微服务架构内的各个模块之间的调用不受影响



消费模块调用服务模块不受zuul的影响，照常调用



服务模块的服务不受任何影响

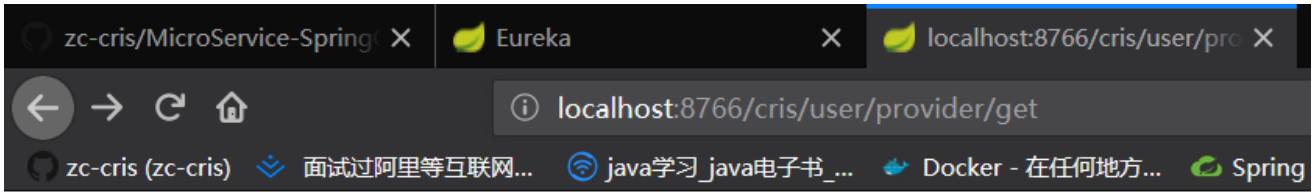
1.7 zuul 的过滤功能

针对用户的请求，我们还可以使用zuul 做权限认证

- 新建一个过滤器做认证

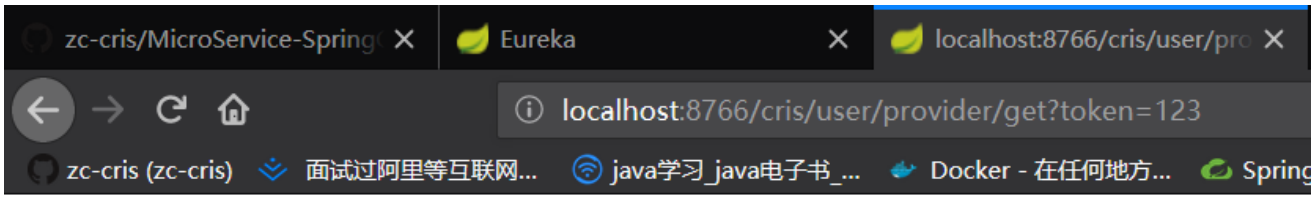
```
/**
 * @ClassName MyFilter
 * @Description TODO
 * @Author zc-cris
 * @Version 1.0
 **/

@Component
```

```java
public class MyFilter extends ZuulFilter {

    // 过滤规则
    @Override
    public String filterType() {
        return "pre";
    }

    // 过滤顺序
    @Override
    public int filterOrder() {
        return 0;
    }

    // 是否需要过滤，一般开启
    @Override
    public boolean shouldFilter() {
        return true;
    }

    // 认证规则
    @Override
    public Object run() throws ZuulException {
        RequestContext currentContext = RequestContext.getCurrentContext();
        HttpServletRequest request = currentContext.getRequest();
        String token = request.getParameter("token");
        if (token == null){
            currentContext.setSendZuulResponse(false);
            currentContext.setResponseStatusCode(401);
            try {
                currentContext.getResponse().getWriter().write("token is missing!,please login or call administrator");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
}
```



token is missing!,please login or call administrator

外界访问就必须提供凭证了



cris8762

拥有凭证，外界才可以访问我们的整个微服务体系

- 注意：
  - `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，越小优先级越高

- shouldFilter：这里可以写逻辑判断，是否要过滤
- 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。

2. config 分布式配置组件（结合git/github）

2.1 pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.cris</groupId>
  <artifactId>config-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>config-server</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.2.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <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.BUILD-SNAPSHOT</spring-cloud.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-config-server</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>

    <dependency>
```

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

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>${spring-cloud.version}</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>

  <repositories>
    <repository>
      <id>spring-snapshots</id>
      <name>Spring Snapshots</name>
      <url>https://repo.spring.io/snapshot</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </repository>
    <repository>
      <id>spring-milestones</id>
      <name>Spring Milestones</name>
      <url>https://repo.spring.io/milestone</url>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

2.2 yml

```yml
server:
  port: 8767

```

```yaml
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/    # 将我们的zuul服务注册到eureka server 的地址
  instance:
    prefer-ip-address: true                  #注册到服务中心使用ip进行注册（服务名称显示的ip详情）

spring:
  application:
    name: microservicecloud-config
  cloud:
    config:
      server:
        git:
          uri: git@github.com:zc-cris/springcloud_server_config_demo.git   #github上的仓库名字
```

2.3 主启动类

```java
@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConfigServerApplication.class, args);
    }
}
```

2.4 启动并测试

zc-cris/springcloud_server_c | Eureka | localhost:8766/cris/user/pro | localhost:8762/provider/get | localhost:8765/consumer/us

localhost:8761

zc-cris (zc-cris) | 面试过阿里等互联网... | java学习_java电子书_... | Docker - 在任何地方... | Spring Cloud Dalsto... | Maven Repository: S... | jQuery API 中文文档 ... | Bootstrap 中文文档 · ...

spring Eureka

HOME LAST 1000 SINCE S

System Status

Environment	test	Current time	2018-05-25T15:42:
Data center	default	Uptime	04:26
		Lease expiration enabled	false
		Renews threshold	6
		Renews (last min)	6

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRE
INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
MICROSERVICECLOUD-CONFIG → config server 模块	n/a (1)	(1)	UP (1) - localhost:microservicecloud-config:8767
MICROSERVICECLOUD-ZUUL-GATEWAY	n/a (1)	(1)	UP (1) - localhost:microservicecloud-zuul-gateway:8766
SERVICE-PROVIDER	n/a (1)	(1)	UP (1) - localhost:service-provider:8762

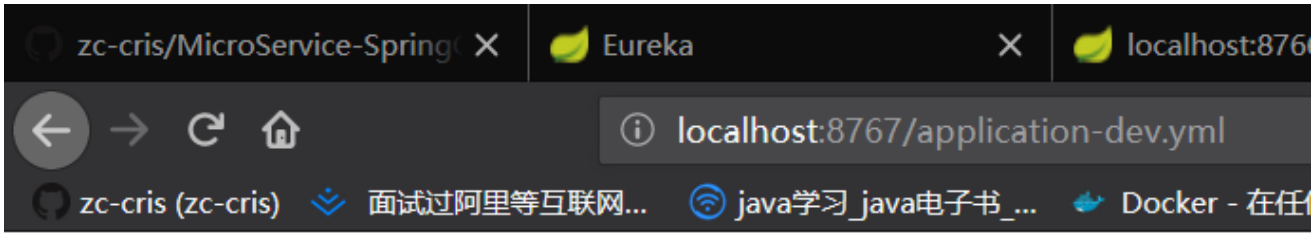
zc-cris/MicroService-Spring | Eureka | localhost:8766/cris/user/pro | localhost:8762/provi

localhost:8767/application-test.yml

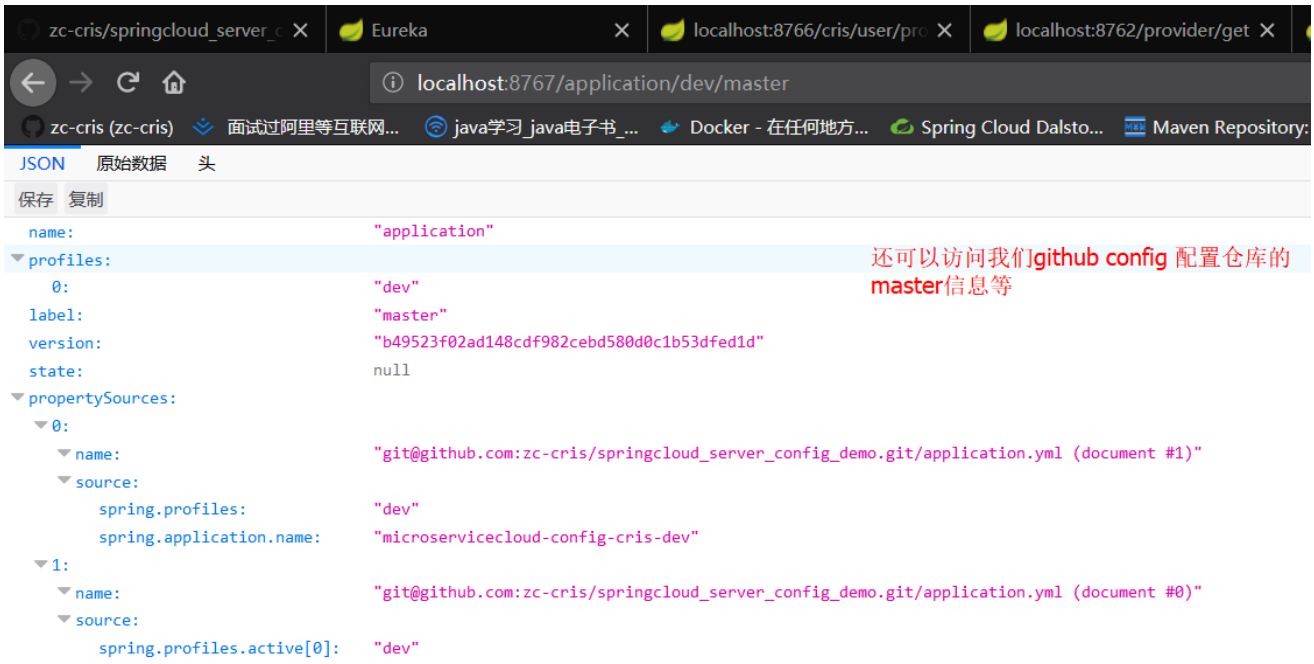
zc-cris (zc-cris) | 面试过阿里等互联网... | java学习_java电子书_... | Docker - 在任何地方... | Spring Cloud Dalsto... | Mav

```
spring:
  application:
    name: microservicecloud-config-cris-test
  profiles:
    active:
    - dev
```

通过我们新建的config server模块成功访问到了我的github上的配置文件信息

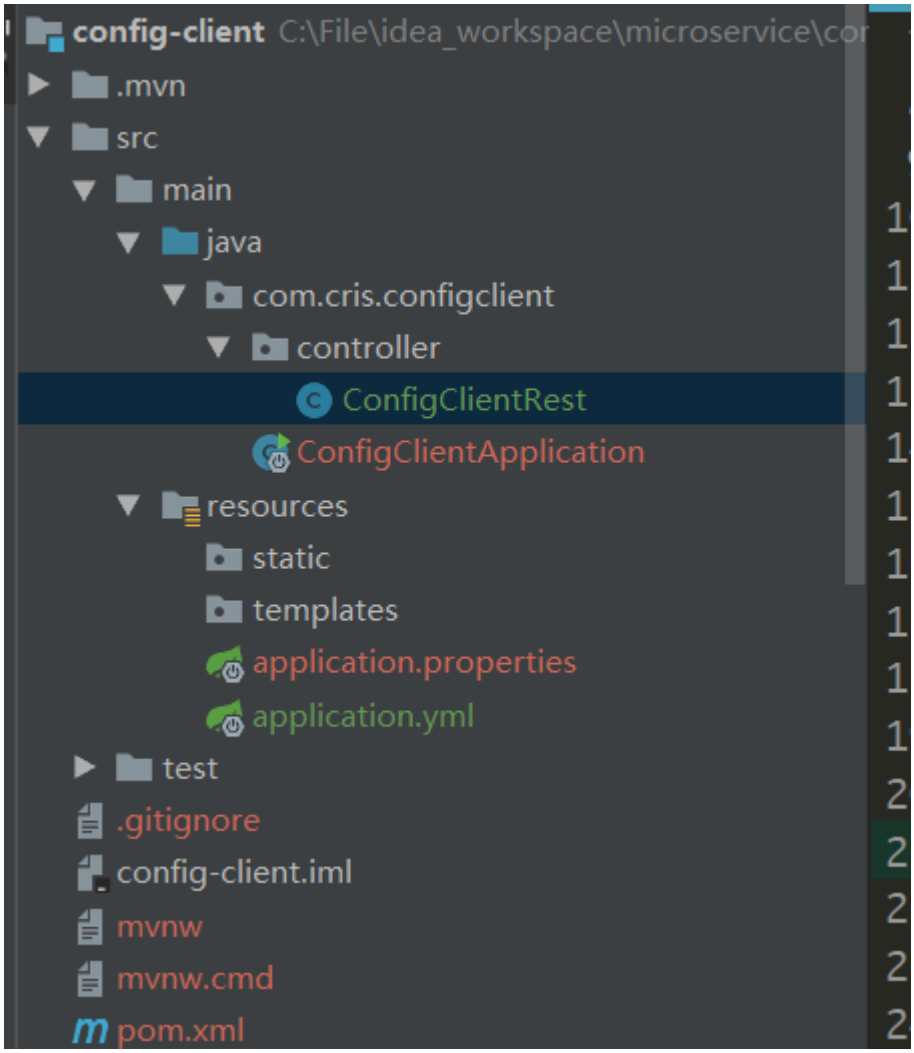



```
spring:
  application:
    name: microservicecloud-config-cris-dev
  profiles:
    active:
    - dev
```



2.5 测试config client 模块能否通过config server端读取到github上的配置信息

1、新建一个config 版的customer 模块（引入eureka discovery, web, config client 依赖）



2、pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cris</groupId>
    <artifactId>config-client</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>config-client</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->

    </parent>
```

```
20
21      <properties>
22          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
23          <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
24          <java.version>1.8</java.version>
25          <spring-cloud.version>Finchley.BUILD-SNAPSHOT</spring-cloud.version>
26      </properties>
27
28      <dependencies>
29          <dependency>
30              <groupId>org.springframework.boot</groupId>
31              <artifactId>spring-boot-starter-web</artifactId>
32          </dependency>
33          <dependency>
34              <groupId>org.springframework.cloud</groupId>
35              <artifactId>spring-cloud-starter-config</artifactId>
36          </dependency>
37          <dependency>
38              <groupId>org.springframework.cloud</groupId>
39              <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
40          </dependency>
41
42          <dependency>
43              <groupId>org.springframework.boot</groupId>
44              <artifactId>spring-boot-starter-test</artifactId>
45              <scope>test</scope>
46          </dependency>
47      </dependencies>
48
49      <dependencyManagement>
50          <dependencies>
51              <dependency>
52                  <groupId>org.springframework.cloud</groupId>
53                  <artifactId>spring-cloud-dependencies</artifactId>
54                  <version>${spring-cloud.version}</version>
55                  <type>pom</type>
56                  <scope>import</scope>
57              </dependency>
58          </dependencies>
59      </dependencyManagement>
60
61      <build>
62          <plugins>
63              <plugin>
64                  <groupId>org.springframework.boot</groupId>
65                  <artifactId>spring-boot-maven-plugin</artifactId>
66              </plugin>
67          </plugins>
68      </build>
69
70      <repositories>
71          <repository>

72              <id>spring-snapshots</id>
```

```
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>


</project>
```

3、yml

```
spring:
  cloud:
    config:
      name: microservicecloud-config-client # 需要从github上读取的资源名称，注意没有yml后缀名
      profile: dev # 本次访问的配置环境
      label: master
      uri: http://localhost:8767    #客户端微服务启动后，先去8767端口寻找服务端，通过服务端获取github的repository 地址
  application:
    name: config-client

eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/    # 将我们的zuul服务注册到eureka server 的地址
  instance:
    prefer-ip-address: true   #注册到服务中心使用ip进行注册（服务名称显示的ip详情）
server:
  port: 8768
```

4、主启动类

```java
@SpringBootApplication
@EnableEurekaClient
public class ConfigClientApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConfigClientApplication.class, args);
    }
}
```

5、测试用的controller

```java
/**
 * @ClassName ConfigClientrest
 * @Description TODO
 * @Author zc-cris
 * @Version 1.0
 **/
@RestController
public class ConfigClientRest {

    @Value("${spring.application.name}")
    private String applicationName;

    @Value("${eureka.client.service-url.defaultZone}")
    private String eurekaServers;

    @Value("${server.port}")
    private String port;

    @GetMapping("/config")
    public String getConfig() {
        String string = "applicationName:" + applicationName + ",eurekaServers:" + eurekaServers + ",port:" + port;
        System.out.println(string);
        return string;
    }
}
```

6、往github上传配置文件

zc-cris / springcloud_server_config_demo

Watch 0 Star

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Branch: master springcloud_server_config_demo / config-client.yml

zc-cris 第二个sc案例的config client 模块配置

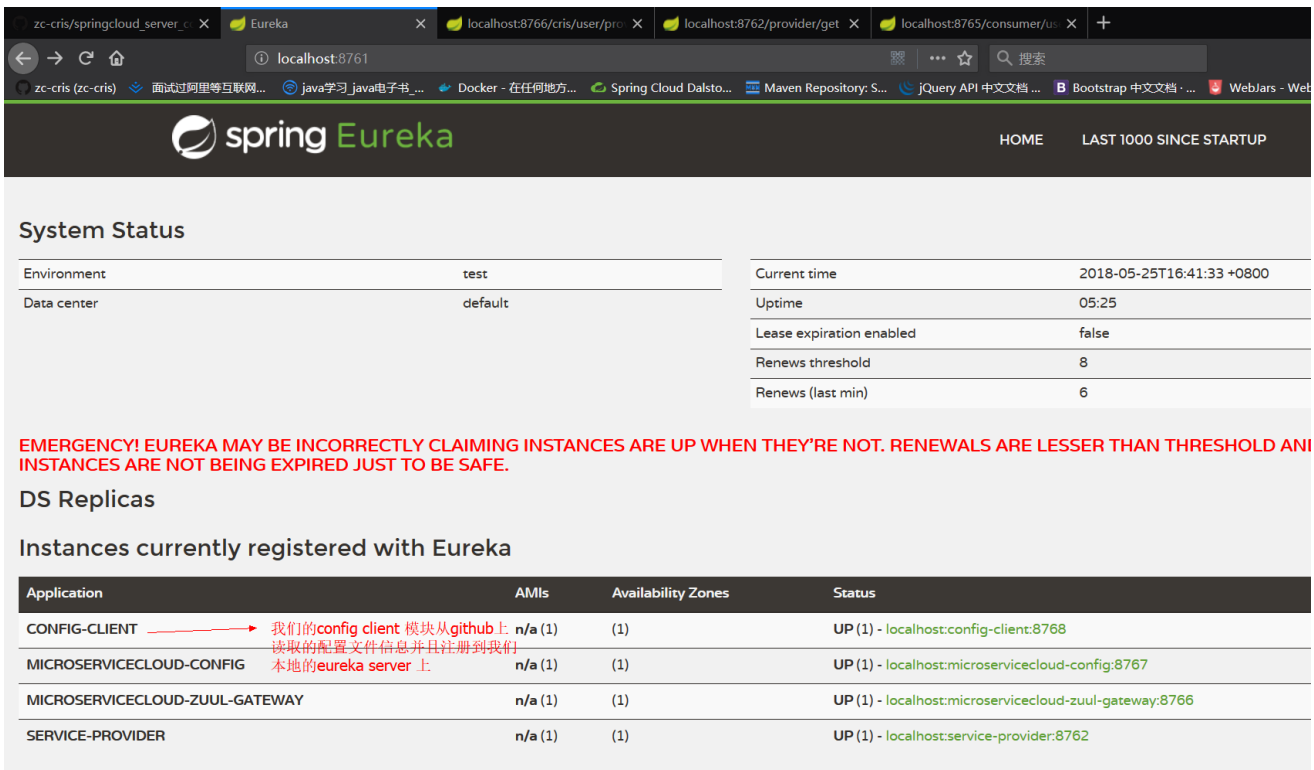
1 contributor

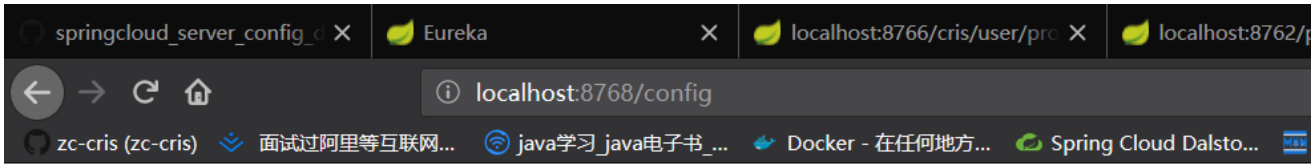
27 lines (26 sloc) | 393 Bytes

Raw | Blame | Hist

```yaml
spring:
  profiles:
    active:
    - dev

---
server:
  port: 8768
spring:
  profiles: dev
  application:
    name: config-client
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
---
```

7. 启动测试





applicationName:config-client,eurekaServers:http://localhost:8761/eureka/,port:8768

这就是我们的config 版 的微服务模块通过 config server模块从github上读取到的配置信息

8. 如果对SpringCloud 的config 组件还想有个综合的实战练习，可以参考我的第一季SC整合案例，这里就不在进行重复性操作了，只要config server 可以从github上读取配置，我们的微服务模块可以通过server 端读取到各自的配置信息（配置集中化管理），即可