# 简易Spring Cloud工程搭建-初篇

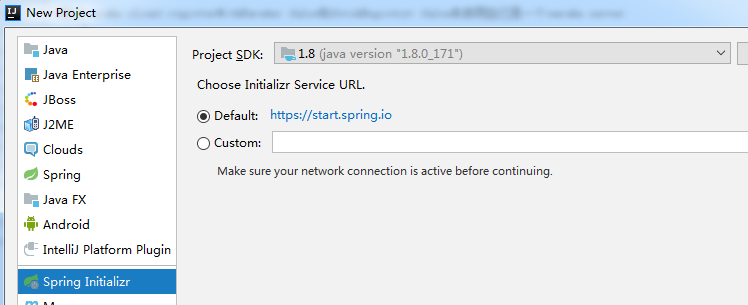
序:

看了半天的springcloud简介,越看越晕,这东西光看理论不上手练习,理解起来还真费劲。所以整理了下网上的例子，本文的例子大多参考了csdn的<<史上最简单的SpringCloud教程>>。吐槽一下此博文的内容确实很好,有些介绍的内容还太全，容易踩坑，故整理下实例方便学习。个人建议把实例都做一遍，再回头看理论就豁然开朗。

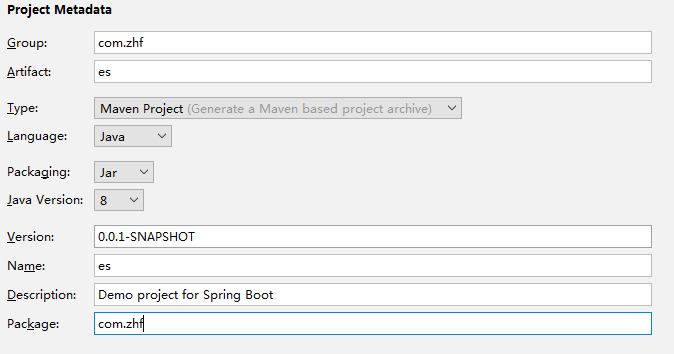
代码下载地址：https://github.com/kaifuu/cloud.git

## 一、创建服务注册中心-Eureka Server

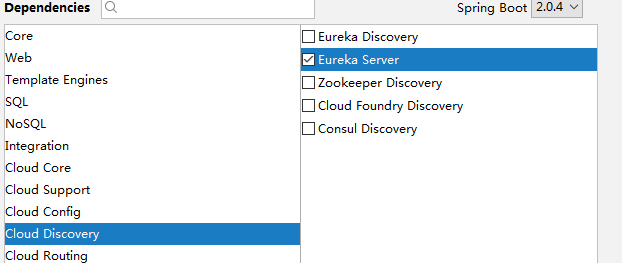
1.新建工程es



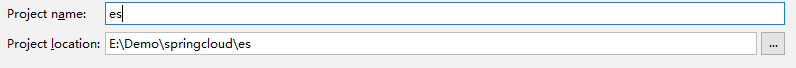
2.点击下一步输入包名



3.选择Eureka Server



4.选择工程路径，并选择完成。



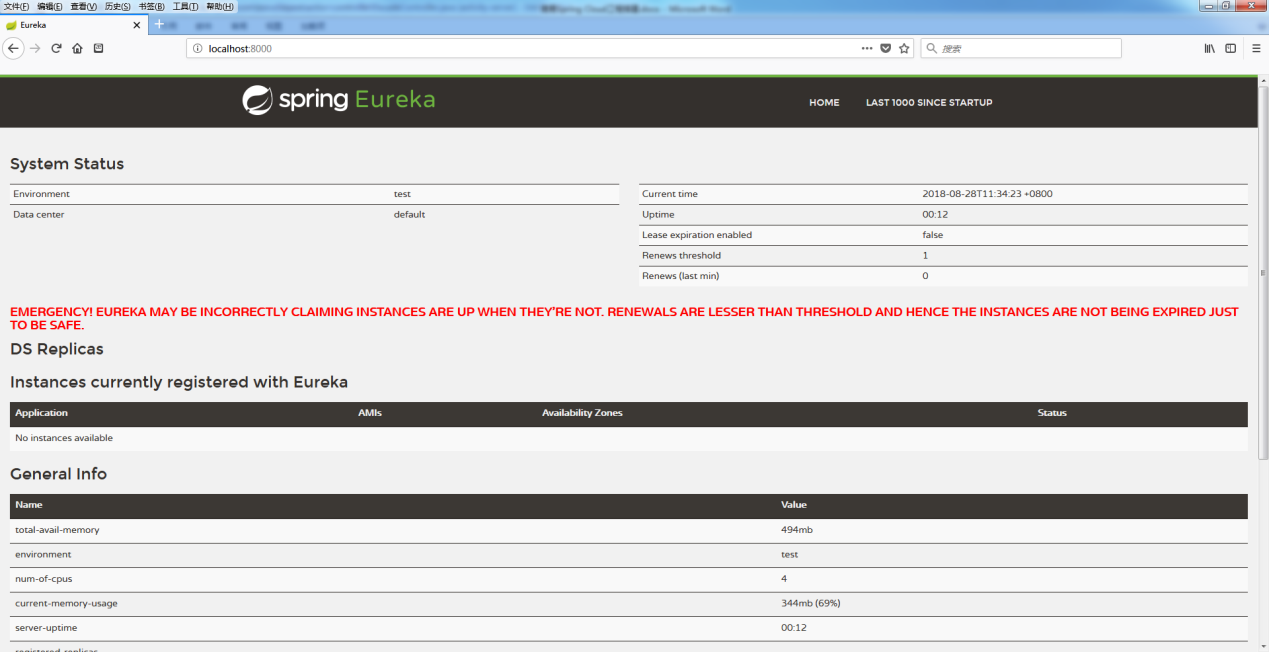
5.设置yml文件

|  |
| --- |
| **server:  port:** 8000  ***#通过registerWithEureka：false和fetchRegistry：false来表明自己是一个eureka server.*****eureka:  instance:  hostname:** localhost  **client:  registerWithEureka: false  fetchRegistry: false  serviceUrl:  defaultZone:** http://${**eureka.instance.hostname**}:${**server.port**}/eureka/  **spring:  application:  name:** es |

6. 配置服务注册中心，只需要一个注解@EnableEurekaServer

|  |
| --- |
| **package** com.zhf.es;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.netflix.eureka.server.EnableEurekaServer; *// 服务注册中心注解@EnableEurekaServer* @SpringBootApplication @EnableEurekaServer **public class** EsApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(EsApplication.**class**, args);  } } |

7.启动工程，访问http://localhost:8000/

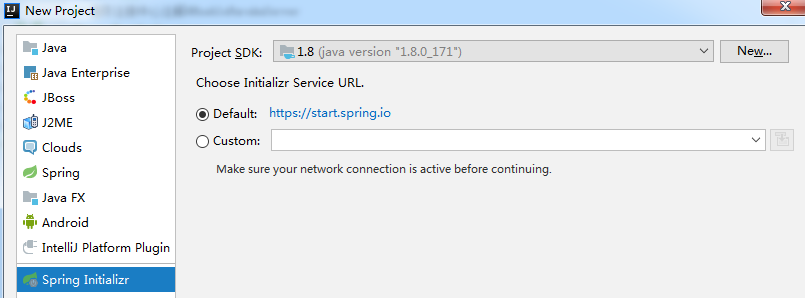


8.pom.xml如下

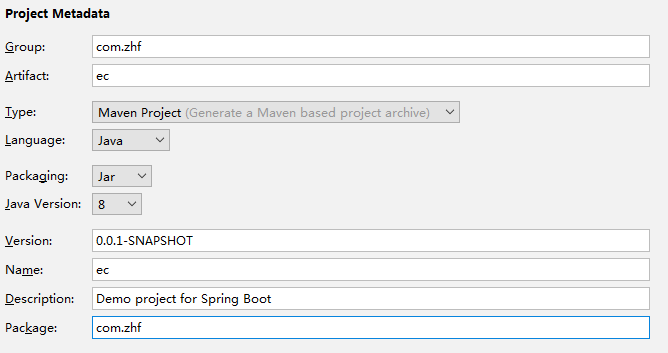
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>es</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>es</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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**> </**project**> |

## 二、创建一个服务提供者-Eureka Client

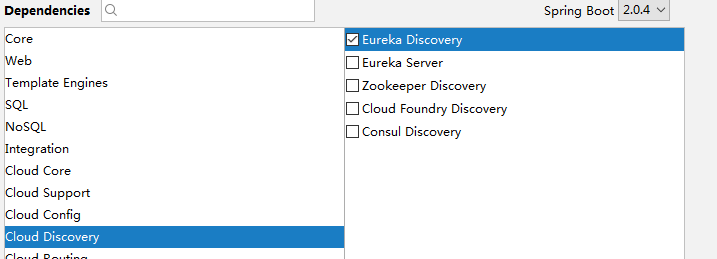
1.新建工程



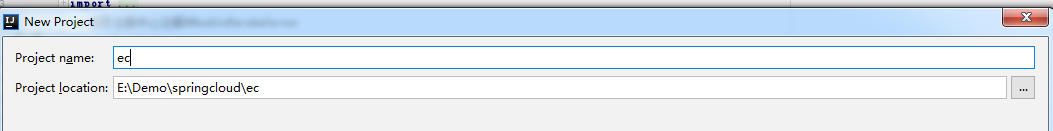
2.填写包路径



3.选择Eureka Discovery



4.选择保存路径



5.配置@EnableEurekaClient表明是一个eurekaclient工程

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient;  @SpringBootApplication @EnableEurekaClient **public class** EcApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(EcApplication.**class**, args);  } } |

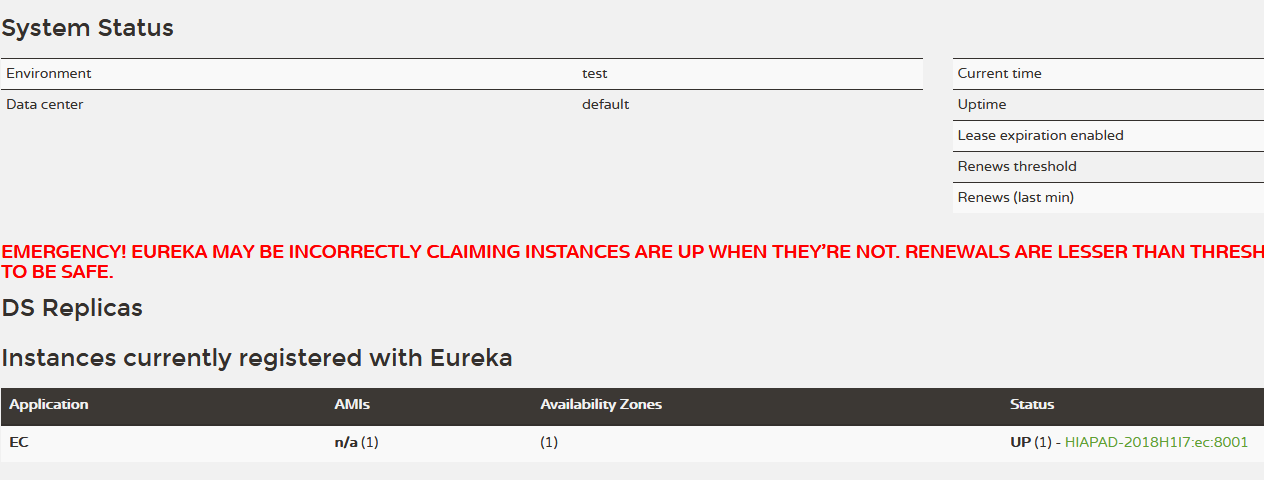
新建一个controller（后面ribbin要用）

|  |
| --- |
| **package** com.zhf.controller;  **import** org.springframework.beans.factory.annotation.Value; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  @RestController **public class** IndexController {  @Value(**"${server.port}"**)  String **port**;   @RequestMapping(**"/index"**)  **public** String home() {  **return "hello world i'm port:"** + **port**;  } } |

6.配置yml

|  |
| --- |
| **server:  port:** 8001  **spring:  application:  name:** ec  **eureka:  client:  serviceUrl: *# 注明自己的服务注册中心的地址*****defaultZone:** http://localhost:8000/eureka/ |

7.在ES服务界面可以看到EC

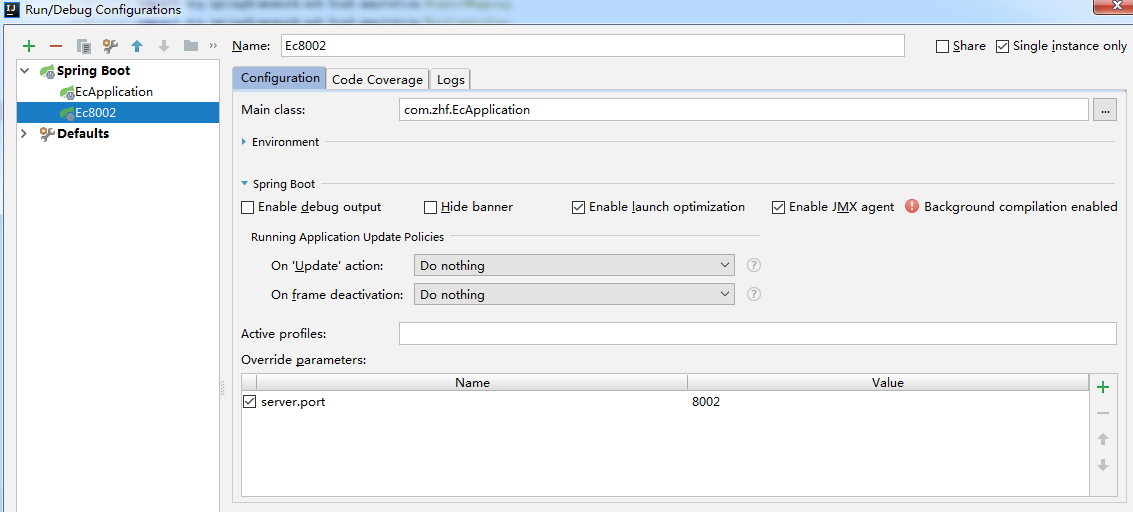


8.pom.xml如下

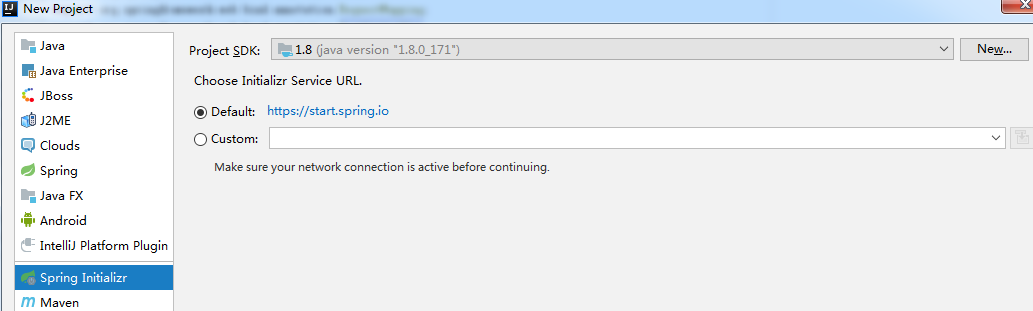
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>ec</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>ec</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**spring-cloud.version**>  </**properties**>   <**dependencies**>  <**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**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-web</**artifactId**>  </**dependency**>   *<!--<dependency>-->  <!--<groupId>org.springframework.boot</groupId>-->  <!--<artifactId>spring-boot-starter-web</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**>  </**project**> |

## 三、消费服务之-负载均衡客户端-ribbon

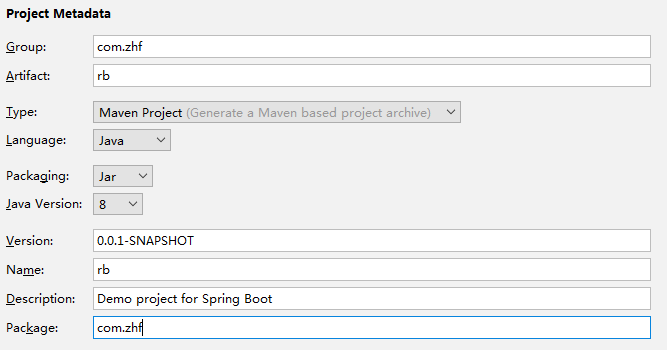
准备：需要启动两个ec实例，端口8001和8002



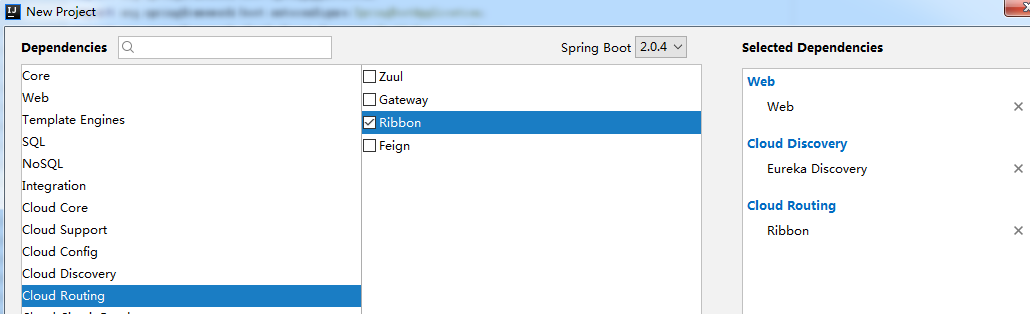
1.创建工程



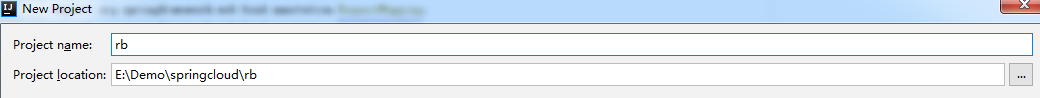
2.创建包



3.选择Web、Discovery和Ribbon



4.指定工程目录



5.设置yml

|  |
| --- |
| **eureka:  client:  serviceUrl: *# 服务注册中心的地址*****defaultZone:** http://localhost:8000/eureka/ **server:  port:** 8100 **spring:  application:  name:** rb |

6.设置负载均衡的功能@LoadBalanced

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.client.loadbalancer.LoadBalanced; **import** org.springframework.context.annotation.Bean; **import** org.springframework.web.client.RestTemplate;  *//@EnableDiscoveryClient:向服务注册中心Eureka Sever注册* @SpringBootApplication @EnableDiscoveryClient **public class** RbApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(RbApplication.**class**, args);  }   @Bean  **@LoadBalanced**  RestTemplate restTemplate() {  **return new** RestTemplate();  } } |

7.编写controller

|  |
| --- |
| **package** com.zhf.action;  **import** com.zhf.service.HelloService; **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;  @RestController **public class** HelloController {  @Autowired  HelloService **helloService**;   @GetMapping(value = **"/get"**)  **public** String getHello(@RequestParam String name) {  **return helloService**.getService( name );  } } |

8.编写service

|  |
| --- |
| **package** com.zhf.service;  **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Service; **import** org.springframework.web.client.RestTemplate;  @Service **public class** HelloService {  @Autowired  RestTemplate **restTemplate**;   **public** String getService(String name) {  **return restTemplate**.getForObject(**"http://ec/index?name="** + name, String.**class**);  } } |

9. 在浏览器上多次访问http://localhost:8100/get?name=111，因为用ribbon进行了负载均衡，会轮流的调用ec的8001和8002端口，所以浏览器交替显示：

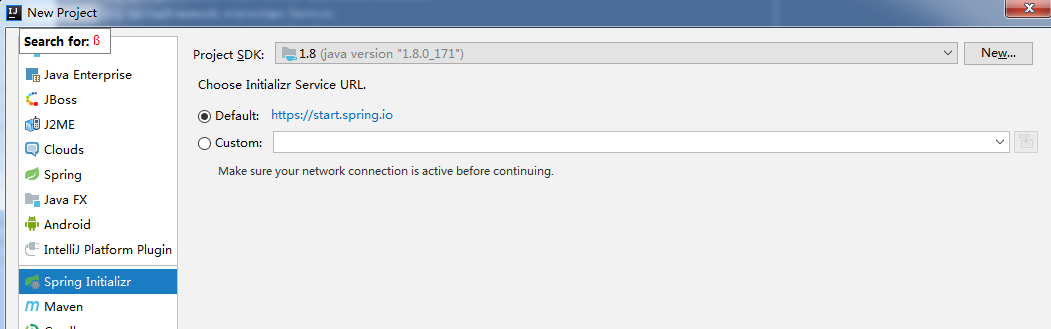
|  |
| --- |
| hello world i'm port:8001  hello world i'm port:8002 |

10.pom.xml如下

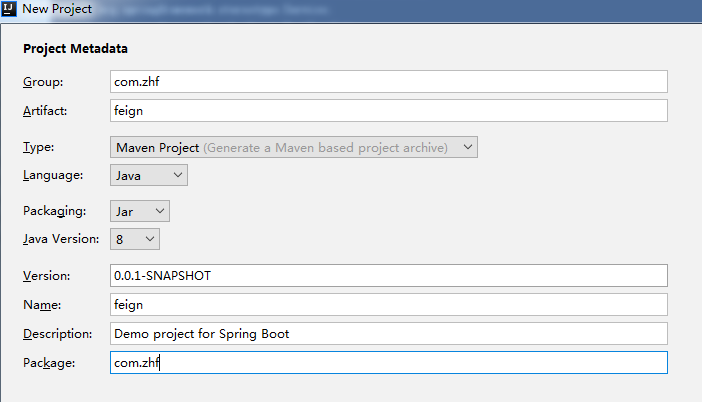
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>rb</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>rb</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**spring-cloud.version**>  </**properties**>   <**dependencies**>  <**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-eureka-client</**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**>  <**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**>  </**project**> |

## 四、消费服务之-伪Http客户端- Feign

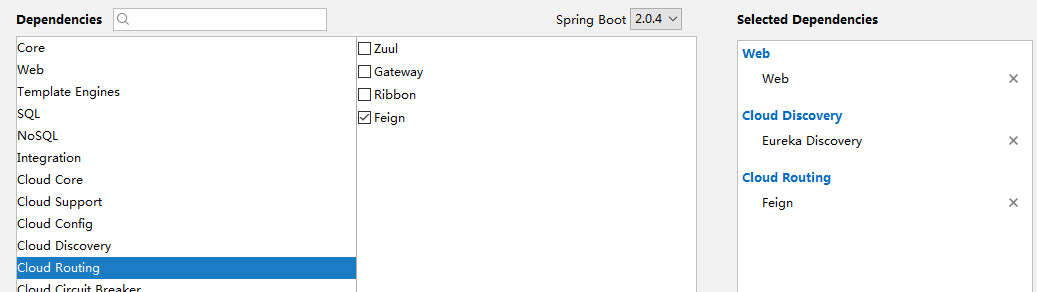
1.新建工程



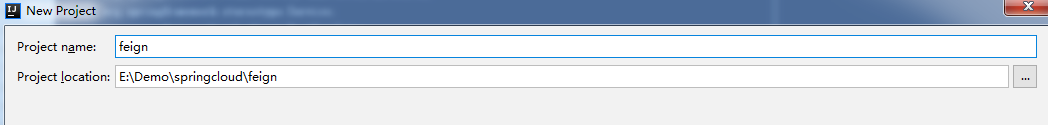
2.设置包名



3.选择依赖Web、Discovery、Feign



4.指定保存目录



5.配置yml

|  |
| --- |
| **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/ **server:  port:** 8200 **spring:  application:  name:** feign |

6.开启Feign功能@EnableFeignClients

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.openfeign.EnableFeignClients;  @SpringBootApplication @EnableFeignClients **public class** FeignApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(FeignApplication.**class**, args);  } } |

7.定义controller

|  |
| --- |
| **package** com.zhf.action;  **import** com.zhf.service.IHelloService; **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;  @RestController **public class** HelloController {  @Autowired  IHelloService **helloService**;   ***//对外暴露一个API接口，通过IHelloService来消费服务***@GetMapping(value = **"/get"**)  **public** String hi(@RequestParam String name) {  **return helloService**.getService();  } } |

8.定义API接口

|  |
| --- |
| **package** com.zhf.service;  **import** org.springframework.cloud.openfeign.FeignClient; **import** org.springframework.stereotype.Component; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RequestMethod; **import** org.springframework.web.bind.annotation.RequestParam;  ***//@ FeignClient指定调用服务名***@FeignClient(value = **"ec"**) @Component **public interface** IHelloService {  @RequestMapping(value = **"/index"**,method = RequestMethod.***GET***)  String getService(); } |

9.Ribbon与Feign比较

**Feign 采用的是基于接口的注解**

10. 在浏览器上多次访问http://localhost:8200/get?name=111，浏览器交替显示：

|  |
| --- |
| hello world i'm port:8001  hello world i'm port:8002 |

11.pom.xml如下

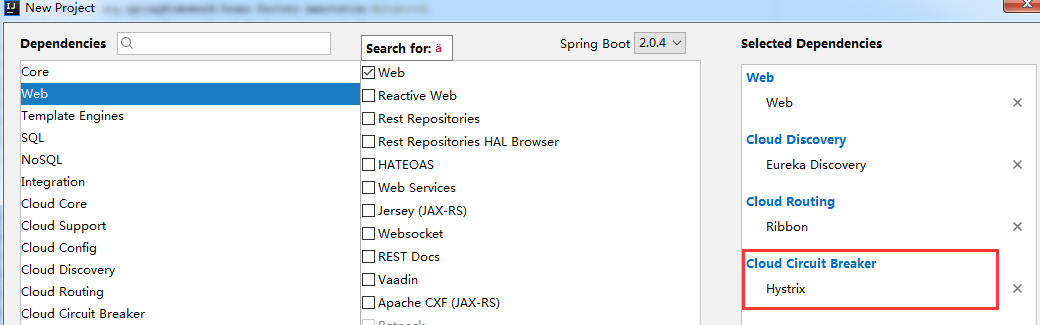
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>feign</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>feign</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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-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**>  </**project**> |

## 五、断路器- Hystrix

准备:

本节直接改造Ribbon与Feign工程,故要pom引入spring-cloud-starter-netflix-hystrix依赖

新创建工程需要选择Hysrix



### 改造Ribbon

1.加入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-hystrix</**artifactId**> </**dependency**> |

2. 开启Hystrix @EnableHystrix

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.client.loadbalancer.LoadBalanced; **import** org.springframework.cloud.netflix.hystrix.EnableHystrix; **import** org.springframework.context.annotation.Bean; **import** org.springframework.web.client.RestTemplate;  *//@EnableDiscoveryClient:向服务注册中心Eureka Sever注册* @SpringBootApplication @EnableDiscoveryClient **@EnableHystrix** **public class** RbApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(RbApplication.**class**, args);  }   @Bean  @LoadBalanced  RestTemplate restTemplate() {  **return new** RestTemplate();  } } |

3.改造service

|  |
| --- |
| **package** com.zhf.service;  **import** com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Service; **import** org.springframework.web.client.RestTemplate;  @Service **public class** HelloService {  @Autowired  RestTemplate **restTemplate**;   *//@HystrixCommand注解:创建了熔断器的功能，并指定了fallbackMethod熔断方法，熔断方法直接返回字符串* @HystrixCommand(fallbackMethod = **"getError"**)  **public** String getService(String name) {  *//ribbon中它会根据服务名来选择具体的服务实例* **return restTemplate**.getForObject(**"http://ec/index?name="** + name, String.**class**);  }   **public** String getError(String name) {  **return "sorry,"** + name + **",i cannot get it!"**;  } } |

4. 启动ribbon并停掉一个ec8002,在浏览器上多次访问http://localhost:8100/get?name=111浏览器交替显示：

|  |
| --- |
| hello world i'm port:8001  sorry,111,i cannot get it! |

### 改造Feign

说明:D版所说需要在yml文件中设置feign.hystrix.enabled=true, 文档中所有工程用的是Finchley版本故不需要再设置

|  |
| --- |
| **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/ **server:  port:** 8200 **spring:  application:  name:** feign  *##开启feign开关* **feign:  hystrix:  enabled:** true |

1.改造接口,加入回调类fallback指定

|  |
| --- |
| **package** com.zhf.service;  **import** org.springframework.cloud.openfeign.FeignClient; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RequestMethod;  *//@ FeignClient指定调用服务名,fallback指定断路时处理的类* @FeignClient(value = **"ec"**,fallback = HelloService.**class**) **public interface** IHelloService {  @RequestMapping(value = **"/index"**,method = RequestMethod.***GET***)  String getService(); } |

2.创建回调类,需要注入到IOC中

|  |
| --- |
| **package** com.zhf.service;  **import** org.springframework.stereotype.Component;  @Component **public class** HelloService **implements** IHelloService{  @Override  **public** String getService() {  **return "Sorry,i cannot get it"**;  } } |

3.controller没有变化

|  |
| --- |
| **package** com.zhf.action;  **import** com.zhf.service.IHelloService; **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;  @RestController **public class** HelloController {  @Autowired  IHelloService **iHelloService**;   *//对外暴露一个API接口，通过IHelloService来消费服务* @GetMapping(value = **"/get"**)  **public** String hi(@RequestParam String name) {  **return iHelloService**.getService();  } } |

4.加入EnableCircuitBreaker打开断路器

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.boot.web.servlet.ServletComponentScan; **import** org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient; **import** org.springframework.cloud.openfeign.EnableFeignClients;  @SpringBootApplication @EnableEurekaClient @EnableCircuitBreaker*//打开Hystrix断路器* @ServletComponentScan*//扫描缓存* @EnableFeignClients*//打开Feign注解* **public class** FeignApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(FeignApplication.**class**, args);  } } |

5.启动工程, 启动feign(停掉一个ec),在浏览器上多次访问http://localhost:8200/get?name=111显示：

|  |
| --- |
| Sorry,i cannot get it |

6.pom.xml如下

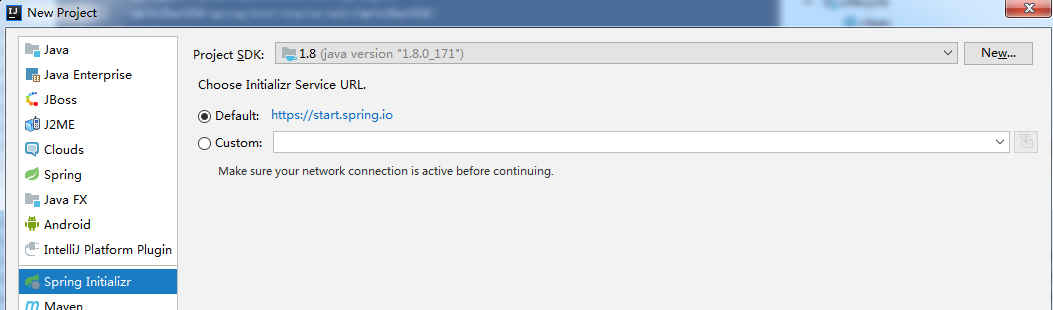
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>feign</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>feign</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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-openfeign</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-hystrix</**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**>   </**project**> |

## 六、路由网关- zuul

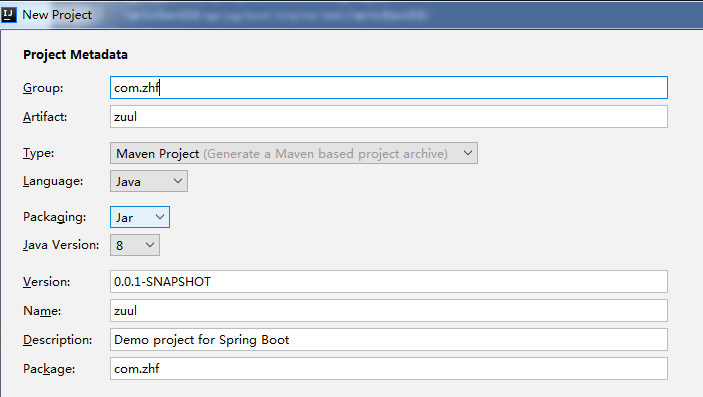
路由转发和过滤器

### 路由转发

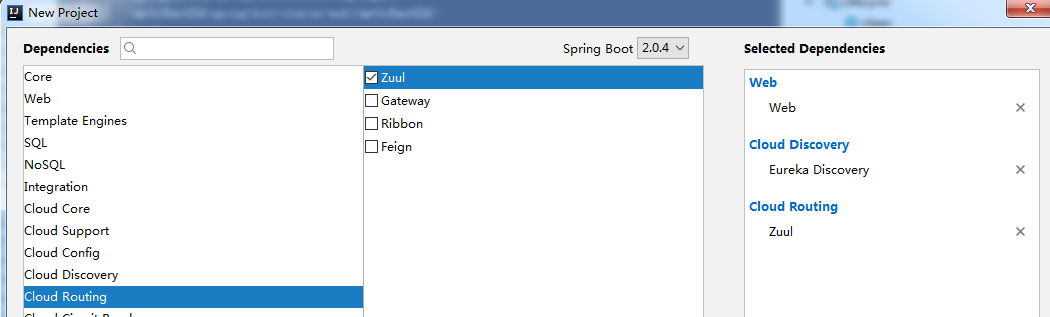
1.创建工程



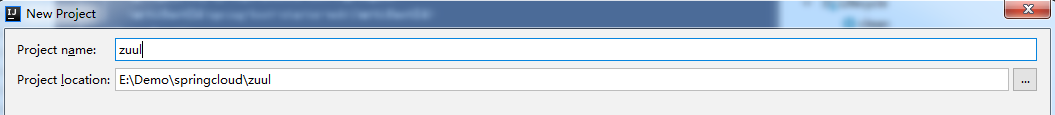
2.配置包名



3.选择依赖



4.指定工程目录



5.配置yml

|  |
| --- |
| **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/ **server:  port:** 8300 **spring:  application:  name:** zuul **zuul:  routes:  api-a:  path:** /api-a/\*\*  **serviceId:** rb  **api-b:  path:** /api-b/\*\*  **serviceId:** feign |

6.开启zuul功能

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient; **import** org.springframework.cloud.netflix.zuul.EnableZuulProxy;  @SpringBootApplication @EnableZuulProxy @EnableEurekaClient @EnableDiscoveryClient **public class** ZuulApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(ZuulApplication.**class**, args);  } } |

7.启动ec8001/ec8002/ribbon/feign和zuul工程, 浏览器访问:

http://localhost:8300/api-a/get?name=111和http://localhost:8300/api-b/get?name=111

交替显示:

hello world i'm port:8001

hello world i'm port:8002

### 过滤器

准备:基于上节内容

1.实现过滤接口

|  |
| --- |
| **package** com.zhf.filter;  **import** com.netflix.zuul.ZuulFilter; **import** com.netflix.zuul.context.RequestContext; **import** org.slf4j.Logger; **import** org.slf4j.LoggerFactory; **import** org.springframework.stereotype.Component;  **import** javax.servlet.http.HttpServletRequest; @Component **public class** ZuuFilter **extends** ZuulFilter {  **private static** Logger *log* = LoggerFactory.*getLogger*(ZuuFilter.**class**);   */\*\*  \* pre：路由之前  \* routing：路由之时  \* post： 路由之后  \* error：发送错误调用  \** ***@return*** *\*/* @Override  **public** String filterType() {  **return "pre"**;  }   */\*\*  \* 过滤的顺序  \** ***@return*** *\*/* @Override  **public int** filterOrder() {  **return** 0;  }   */\*\*  \* 逻辑判断，是否要过滤，本文true,永远过滤。  \** ***@return*** *\*/* @Override  **public boolean** shouldFilter() {  **return true**;  }   */\*\*  \* 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。  \** ***@return*** *\*/* @Override  **public** Object run() {  RequestContext ctx = RequestContext.*getCurrentContext*();  HttpServletRequest request = ctx.getRequest();  *log*.info(String.*format*(**"%s >>> %s"**, request.getMethod(), request.getRequestURL().toString()));  Object accessToken = request.getParameter(**"token"**);  **if** (accessToken == **null**) {  *log*.warn(**"token is empty"**);  ctx.setSendZuulResponse(**false**);  ctx.setResponseStatusCode(401);  **try** {  ctx.getResponse().getWriter().write(**"token is empty"**);  } **catch** (Exception e) {  }   **return null**;  }  *log*.info(**"ok"**);  **return null**;  } } |

2.启动zuul工程,浏览器访问:

http://localhost:8300/api-a/get?name=111和http://localhost:8300/api-b/get?name=111

显示:

token is empty

3.加入token再次访问:

http://localhost:8300/api-a/get?name=111&token=1和http://localhost:8300/api-b/get?name=111&token=1

交替显示

hello world i'm port:8001

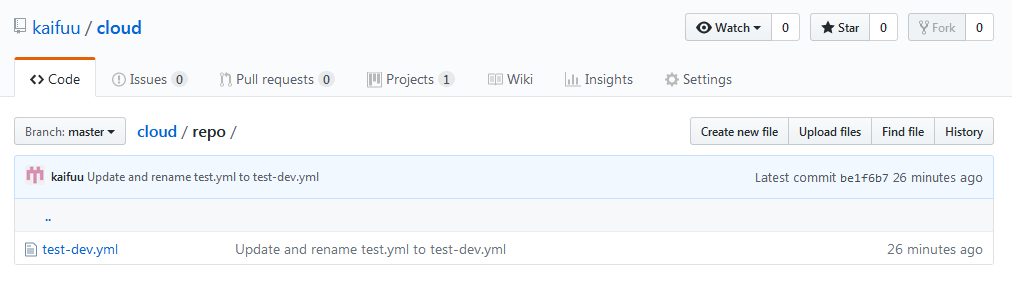
hello world i'm port:8002

## 七、配置中心

### Config Server

说明：配置中心从远程git读取配置文件，可以改造之前的Eureka Server工程，也可以新创建工程，本节选用新建工程

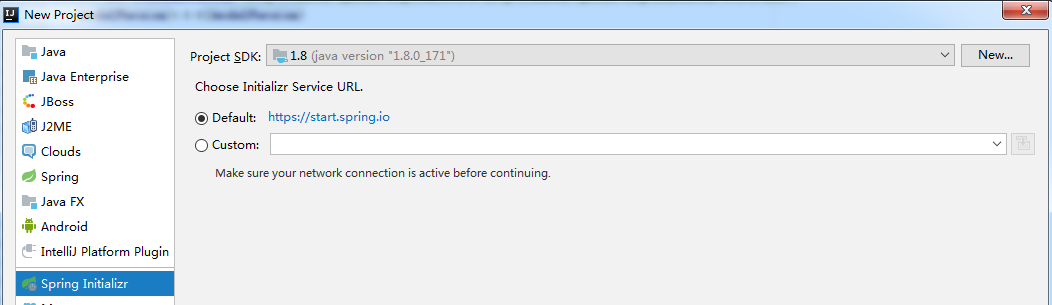
准备:需要在github上放上测试文件test-dev.yml(文件名与之后配置文件中保持一至)



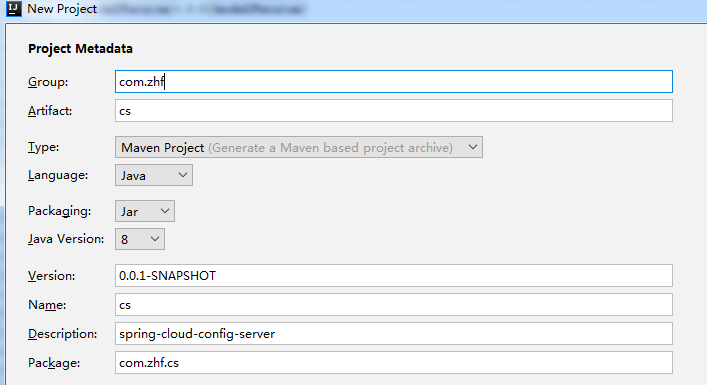
test-dev.yml文件内容如下 :

|  |
| --- |
| zhf:  test: 测试用的 |

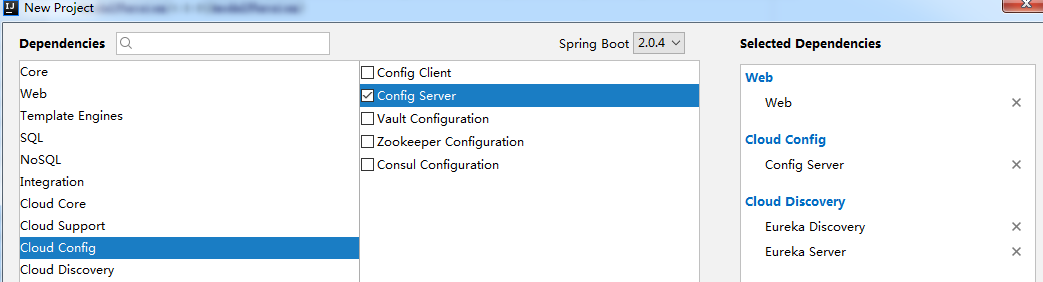
1.创建工程



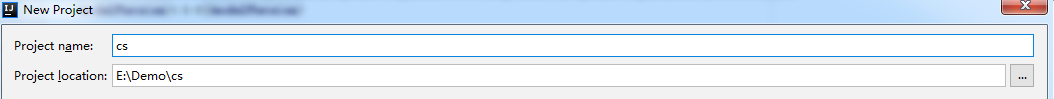
2.填写包名



3.选择依赖



4.选择工程路径



5.配置application.yml文件

|  |
| --- |
| **server:  port:** 8400 **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/ **spring:  application:  name:** cs  **cloud:  config:  server:  git:  uri:** https://github.com/kaifuu/cloud *# 配置git仓库的地址* **search-paths:** repo *# git仓库地址下的相对地址，可以配置多个，用,分割。* **username:** *# git仓库的账号* **password:** *# git仓库的密码* |

6.启动Config server @EnableConfigServer

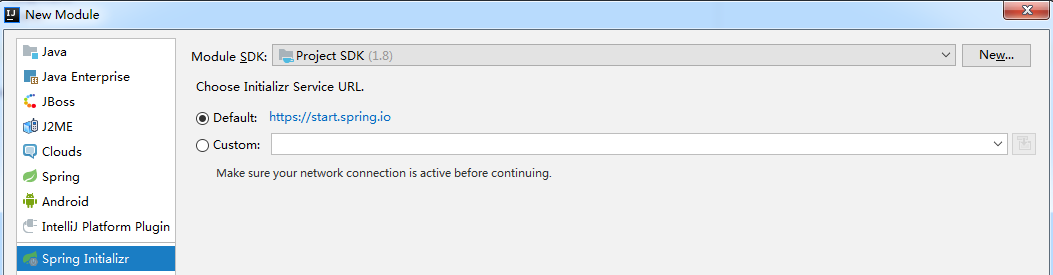
|  |
| --- |
| **package** com.zhf.cs;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.config.server.EnableConfigServer; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient; **import** org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  *// 服务注册中心注解@EnableEurekaServer* @SpringBootApplication @EnableConfigServer @EnableDiscoveryClient @EnableEurekaClient **public class** CsApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(CsApplication.**class**, args);  } } |

7.pom.xml内容如下

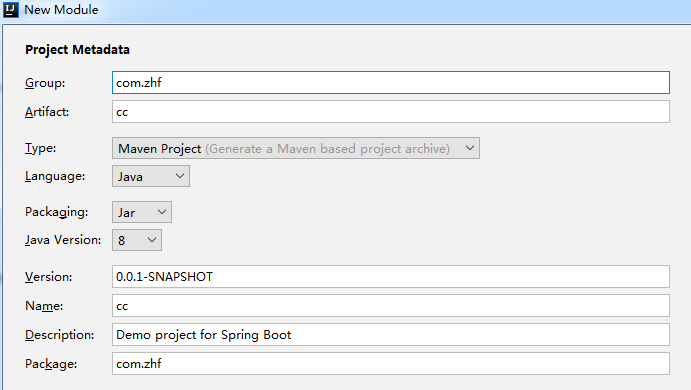
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>cs</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>cs</**name**>  <**description**>spring-cloud-config-server</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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.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**>   </**project**> |

### Config Client

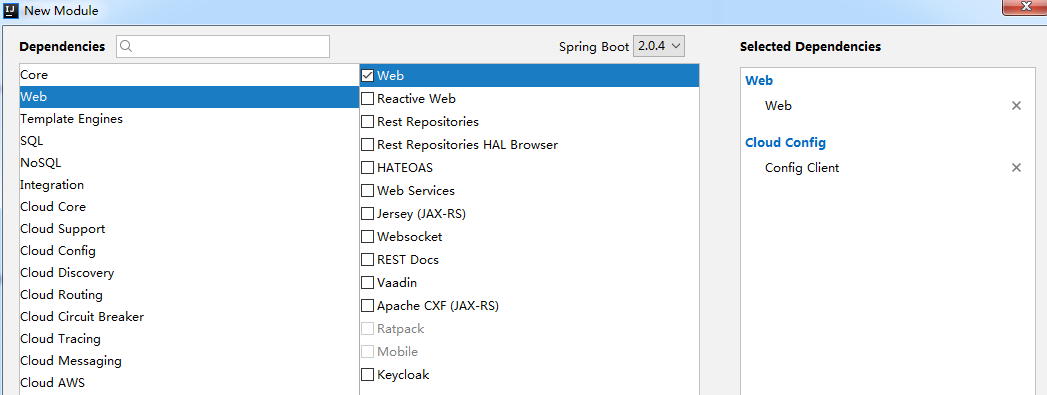
1.创建工程



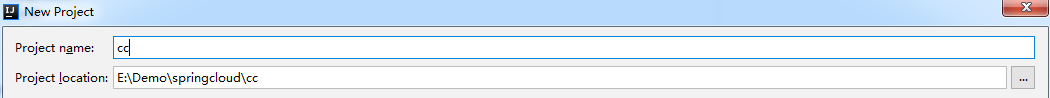
2.填写包名



3.选择依赖



4.设置工程目录



5.设置application.yml 本地服务配置文件

|  |
| --- |
| **spring:  application:  name:** cc **server:  port:** 8401 |

6.设置Config Client用的配置文件bootstrap.yml,注意test和dev对应github的test-dev文件

|  |
| --- |
| **spring:  cloud:  config:  name:** test  **profile:** dev  **uri:** http://localhost:8400/  **label:** master **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/ |

7.编写测试controller

|  |
| --- |
| **package** com.zhf.action;  **import** org.springframework.beans.factory.annotation.Value; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  @RestController **class** HelloController {   @Value(**"${zhf.test}"**)  **private** String **hello**;   @RequestMapping(**"/hello"**)  **public** String from() {  **return this**.**hello**;  } } |

8.编写启动类

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.beans.factory.annotation.Value; **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  @SpringBootApplication @EnableDiscoveryClient @EnableEurekaClient **public class** CcApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(CcApplication.**class**, args);  } } |

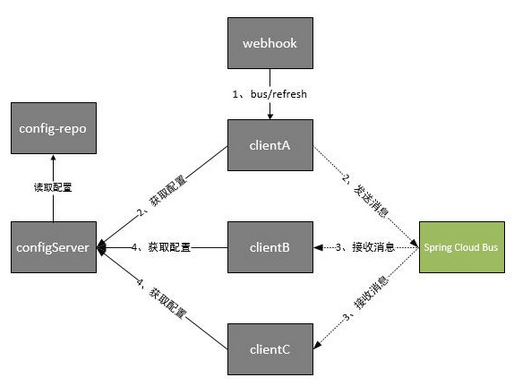
9.分别启动Eureka server/Config Server和Config Client,浏览器访问http://localhost:8401/hello,展示: "测试用的" ---test-dev.yml的内容

10.pom.xml内容如下:

|  |
| --- |
| *<?***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.zhf</**groupId**>  <**artifactId**>cc</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>cc</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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-config</**artifactId**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-test</**artifactId**>  <**scope**>test</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-eureka-client</**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**>  </**project**> |

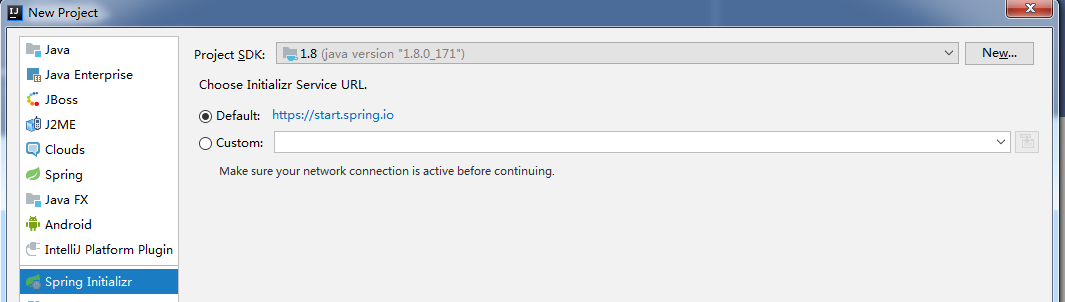
## 八、消息总线-Bus

说明：1）新建工程,也可以在Config Client基础上改造。2）搭载rabbitMQ---充当消息总线用(a.下载b.配置环境变量c.启动服务d.访问http://localhost:15672).3)注意pom文件的spring-cloud-starter-stream-rabbit依赖，否则报错。4)发送post请求，刷新页面没有用。

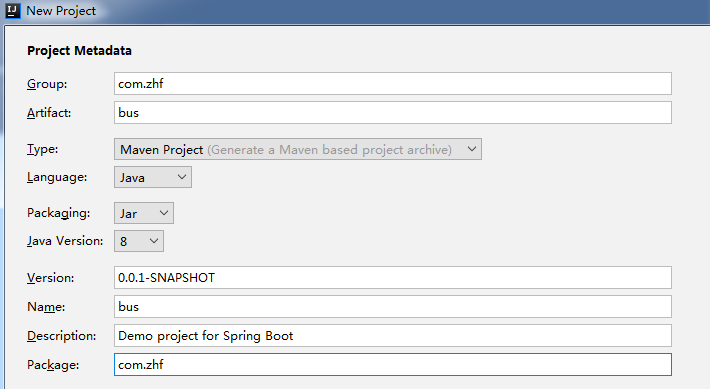


* 提交代码触发post请求给bus/refresh
* server端接收到请求并发送给Spring Cloud Bus
* Spring Cloud bus接到消息并通知给其它客户端
* 其它客户端接收到通知，请求Server端获取最新配置
* 全部客户端均获取到最新的配置

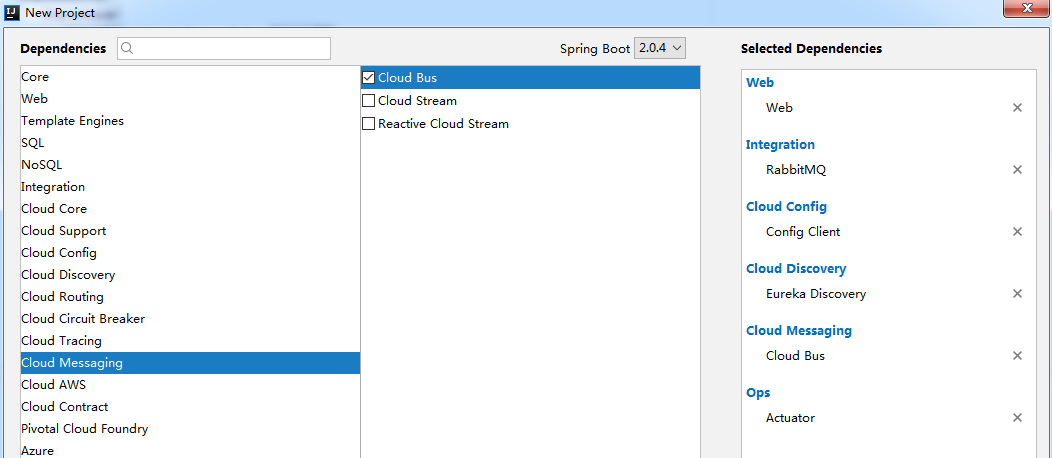
1.创建工程



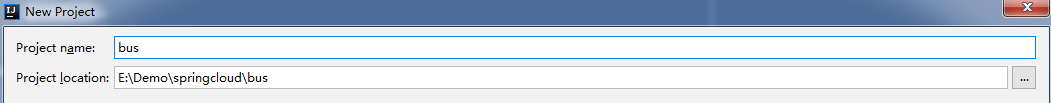
2.填写包路径



3.选择依赖



4.设置工程保存目录



5.新建application.yml

|  |
| --- |
| **spring:  application:  name:** bus  **rabbitmq:  host:** localhost  **port:** 5672  **username:** guest  **password:** guest **server:  port:** 8402 **management:  endpoints:  web:  exposure:  include:** bus-refresh |

6.新建bootstrap.yml

|  |
| --- |
| **spring:  cloud:  config:  name:** test  **profile:** dev  **uri:** http://localhost:8400/  **label:** master  **bus:  enabled: true  trace:  enabled: true** *#开启消息跟踪* **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/ |

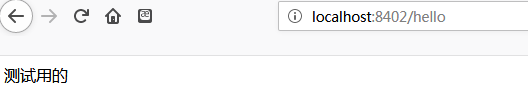
7.新建controller

|  |
| --- |
| **package** com.zhf.action;  **import** org.springframework.beans.factory.annotation.Value; **import** org.springframework.cloud.context.config.annotation.RefreshScope; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  @RestController @RefreshScope*// 使用该注解的类，会在接到SpringCloud配置中心配置刷新的时候，自动将新的配置更新到该类对应的字段中。* **class** HelloController {   @Value(**"${zhf.test}"**)  **private** String **hello**;   @RequestMapping(**"/hello"**)  **public** String from() {  **return this**.**hello**;  } } |

8.新建启动类

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.context.config.annotation.RefreshScope; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient;  @SpringBootApplication @EnableDiscoveryClient @EnableEurekaClient **public class** BusApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(BusApplication.**class**, args);  } } |

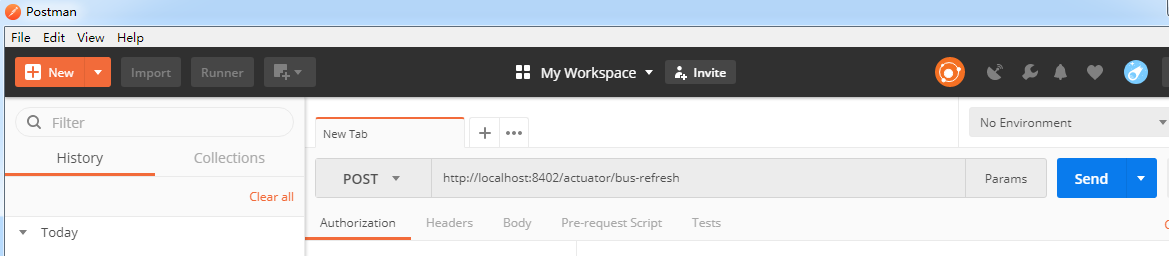
9.访问http://localhost:8402/hello显示：



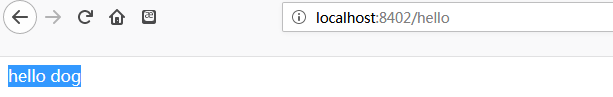
10.更改github上文件内容



11.发送post请求：http://localhost:8402/actuator/bus-refresh



12.浏览器显示新的内容



13.pom.xml内容如下

|  |
| --- |
| *<?***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.zhf</**groupId**>  <**artifactId**>bus</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>bus</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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-bus</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-config</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-eureka-client</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.amqp</**groupId**>  <**artifactId**>spring-rabbit</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-stream-rabbit</**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**>   </**project**> |

## 九、服务链路追踪- Sleuth

说明：

本文的案例主要有三个工程组成：

* server-zipkin：收集调用数据并展示
* zkclient,对外暴露hello接口
* zkclientnew,对外暴露miya接口

当相zkclient和zkclientnew互调用时server-zipkin会收集数据的，这就是为什么叫服务追踪了。

注意：

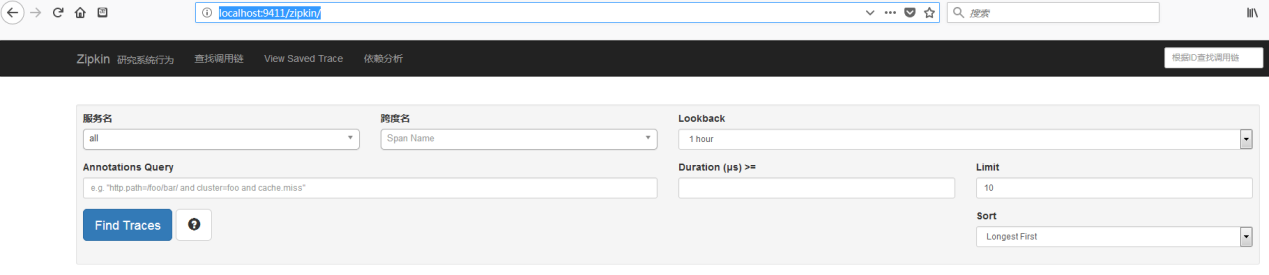
本例子踩坑，耗时近3个小时，就是配置问题没写对。配置中标黄的位置请注意。

### zipkin-server

1.下载https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/2.11.3/

2.运行zipkin-server: java -jar zipkin-server-2.11.0-exec.jar

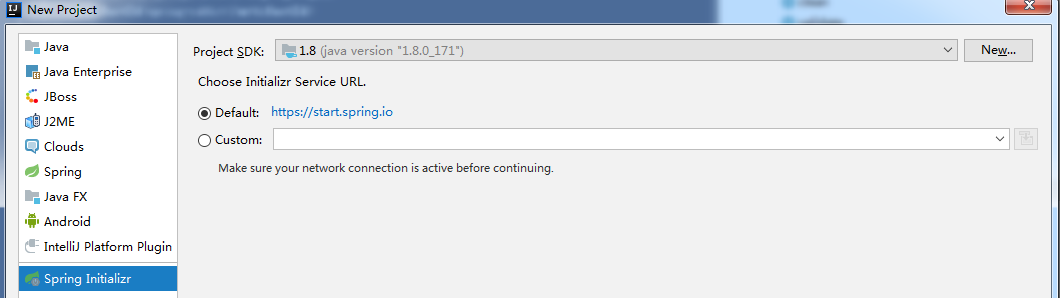
3.访问URL：http://localhost:9411/zipkin/



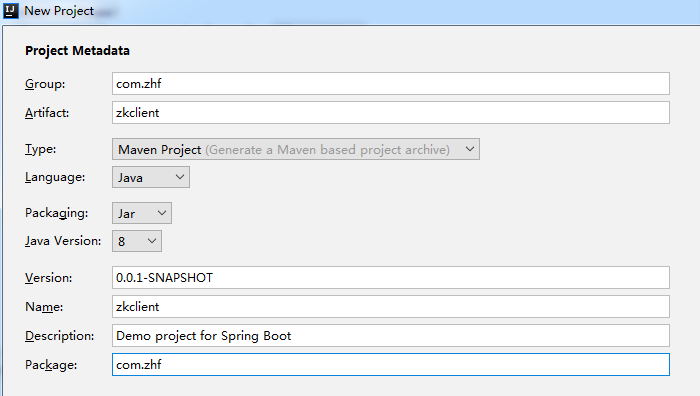
### zipkin-client

#### zkclient

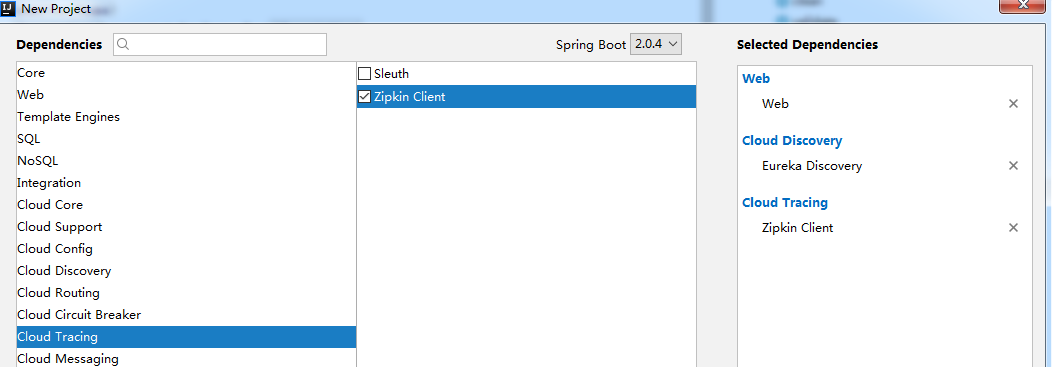
1.新建工程



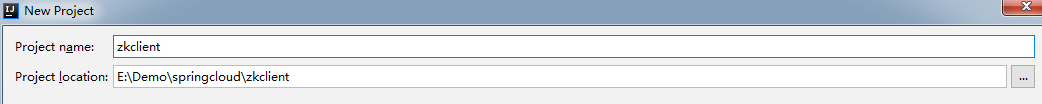
2.填写包名



3.选择依赖



4.设置工程保存目录



5.创建yml

|  |
| --- |
| **server:  port:** 8501 **spring:  application:  name:** zkclient  **zipkin:  base-url:** http://localhost:9411  **sleuth:  sampler:  probability:** 1.0 *#设置为1（默认为0.1），否则要调用10次才会上传成功1次* **trace-id128: true** |

6.创建启动类

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication **public class** ZkclientApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(ZkclientApplication.**class**, args);  } } |

7.创建controller

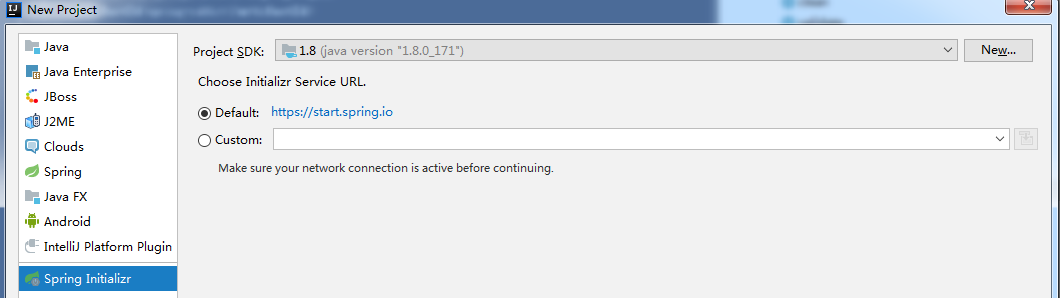
|  |
| --- |
| **package** com.zhf.action;   **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.context.annotation.Bean; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController; **import** org.springframework.web.client.RestTemplate;  @RestController **class** HelloController {   @Autowired  **private** RestTemplate **restTemplate**;   @Bean  **public** RestTemplate getRestTemplate(){  **return new** RestTemplate();  }   @RequestMapping(**"/hello"**)  **public** String hello(){  **return restTemplate**.getForObject(**"http://localhost:8502/info"**, String.**class**);  }   @RequestMapping(**"/info"**)  **public** String info(){  **return "this is 8501 info"**;  } } |

8.pom.xml内容如下:

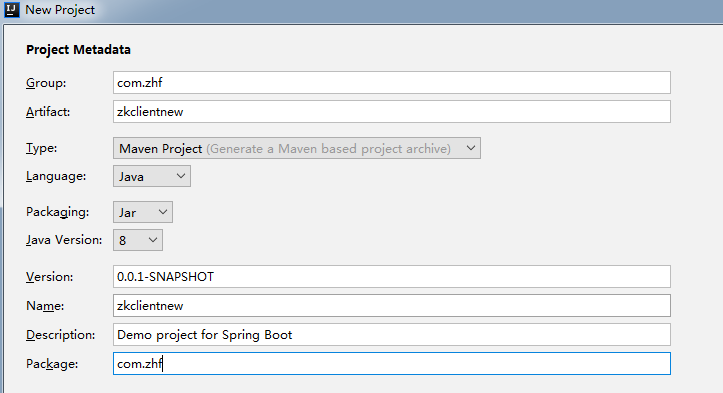
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>zkclient</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>zkclient</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**spring-cloud.version**>  <**sleuth.version**>2.0.0.RELEASE</**sleuth.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-zipkin</artifactId>-->  <!--</dependency>-->* <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-test</**artifactId**>  <**scope**>test</**scope**>  </**dependency**>  *<!-- zipkin-->  <!-- Sleuth automatically adds trace interceptors when in the classpath -->* <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-sleuth</**artifactId**>  </**dependency**>  *<!-- Sends trace data to zipkin over http (defaults to http://localhost:9411/api/v2/spans) -->* <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-sleuth-zipkin</**artifactId**>  </**dependency**>  *<!-- zipkin-->* </**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**>   </**project**> |

#### zkclientnew

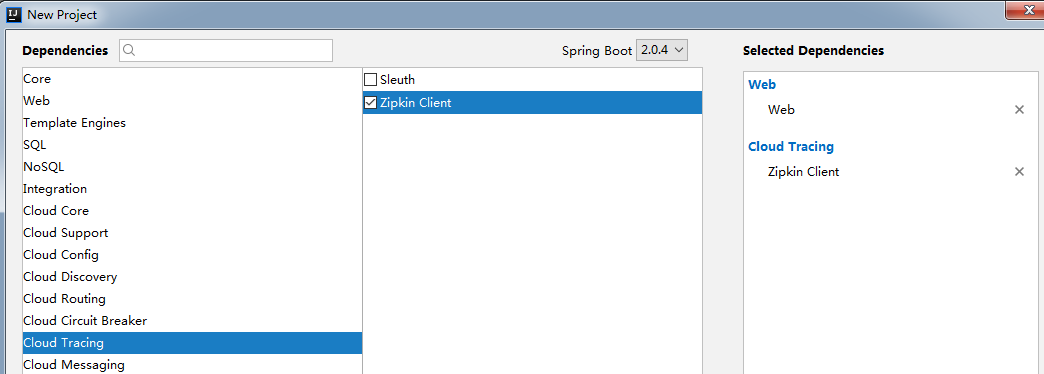
1.新建工程



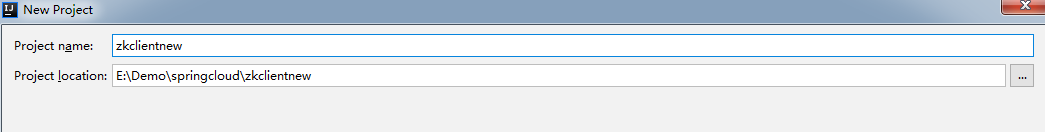
2.填写包名



3.选择依赖包



4.设置工程保存目录



5.创建yml

|  |
| --- |
| **server:  port:** 8502 **spring:  application:  name:** zkclientnew  **zipkin:  base-url:** http://localhost:9411  **sleuth:  sampler:  probability:** 1.0 *#设置为1（默认为0.1），否则要调用10次才会上传成功1次* |

6.创建启动类

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication **public class** ZkclientnewApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(ZkclientnewApplication.**class**, args);  } } |

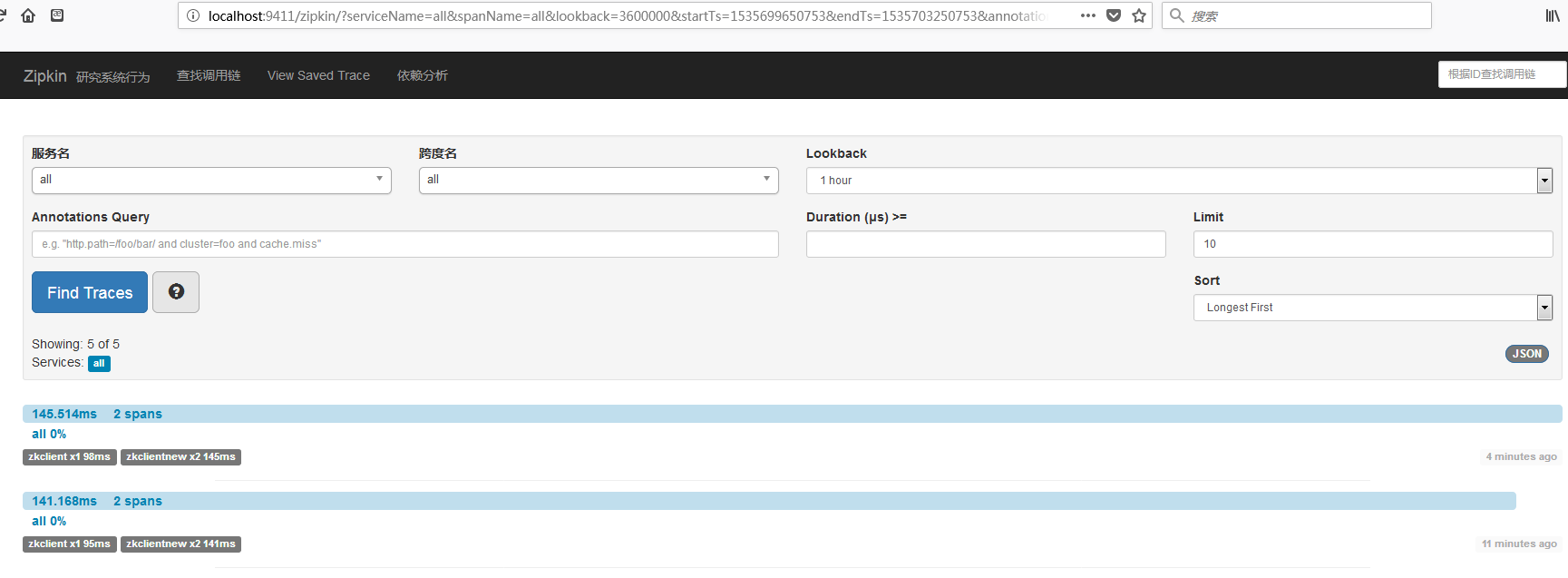
7.创建controller

|  |
| --- |
| **package** com.zhf.action;   **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.context.annotation.Bean; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController; **import** org.springframework.web.client.RestTemplate;  @RestController **class** HelloController {   @Autowired  **private** RestTemplate **restTemplate**;   @Bean  **public** RestTemplate getRestTemplate(){  **return new** RestTemplate();  }   @RequestMapping(**"/hello"**)  **public** String hello(){  **return restTemplate**.getForObject(**"http://localhost:8501/info"**, String.**class**);  }  @RequestMapping(**"/info"**)  **public** String info(){  **return "this is 8502 info"**;  } } |

8.pom.xml内容如下:

|  |
| --- |
| *<?***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.zhf</**groupId**>  <**artifactId**>zkclientnew</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>zkclientnew</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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-zipkin</artifactId>-->  <!--<version>2.0.1.RELEASE</version>-->  <!--</dependency>-->* <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-test</**artifactId**>  <**scope**>test</**scope**>  </**dependency**>  *<!-- zipkin-->  <!-- Sleuth automatically adds trace interceptors when in the classpath -->* <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-sleuth</**artifactId**>  </**dependency**>  *<!-- Sends trace data to zipkin over http (defaults to http://localhost:9411/api/v2/spans) -->* <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-sleuth-zipkin</**artifactId**>  </**dependency**>  *<!-- zipkin-->* </**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**>   </**project**> |

9.浏览器访问：http://localhost:8501/hello http://localhost:8502/hello，可以在http://localhost:9411看到依赖关系



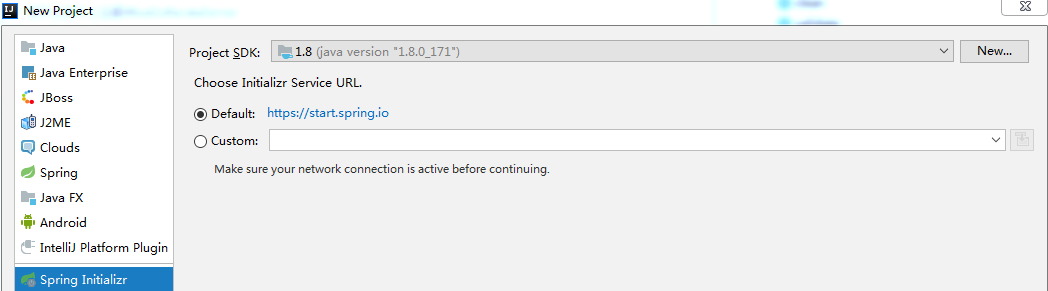
## 十、高可用的服务注册中心

可以改第一节的es和第二节的ec工程，本节内容是新建es1和es2

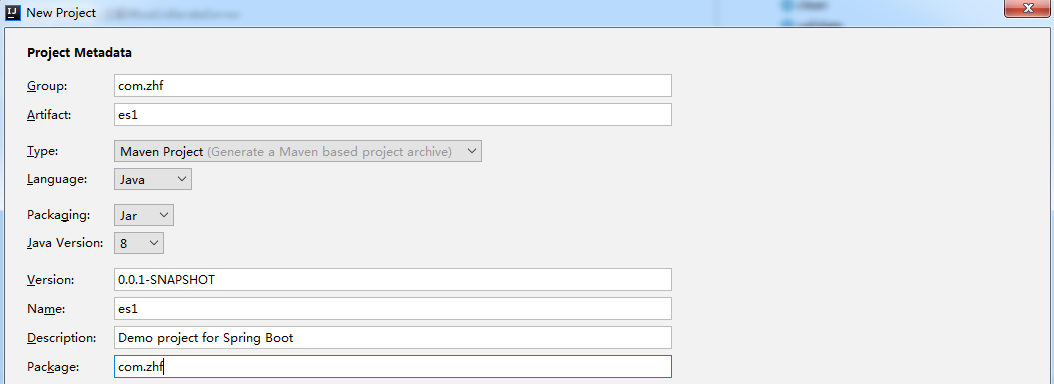
### ES

#### ES1

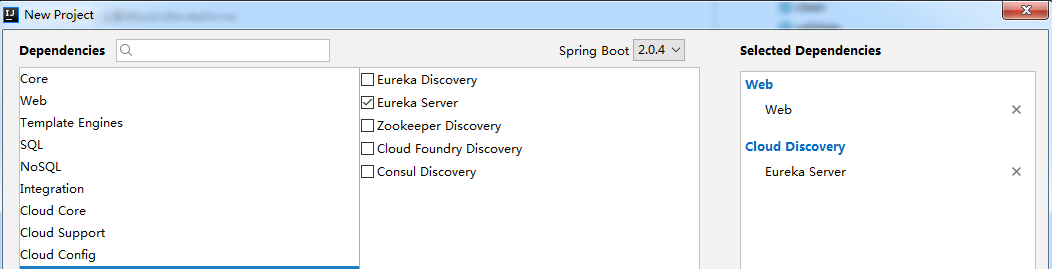
1.新建es1工程



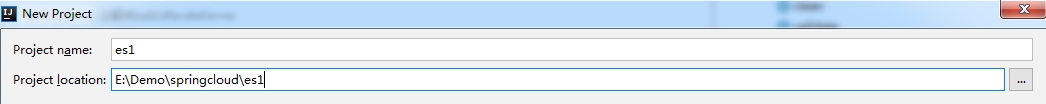
2.配置包名



3.选择依赖



4.选择保存目录



5.创建application.yml

|  |
| --- |
| **eureka:  instance:  hostname:** es1  **client:  serviceUrl:  defaultZone:** http://es2:8802/eureka/ **server:  port:** 8801 |

6.开启服务注册中心

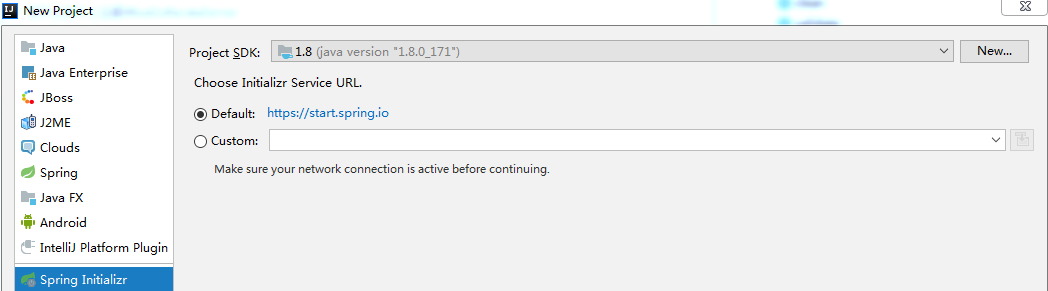
|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  @SpringBootApplication @EnableEurekaServer **public class** Es1Application {   **public static void** main(String[] args) {  SpringApplication.*run*(Es1Application.**class**, args);  } } |

7.pom.xml如下

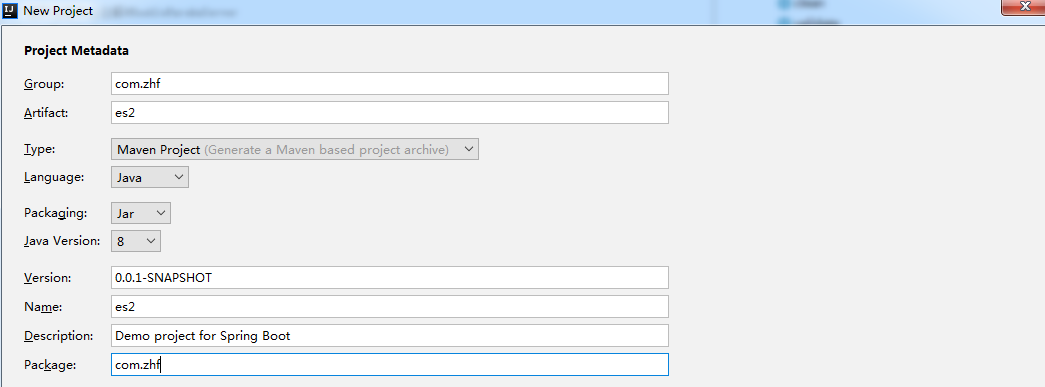
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>es1</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>es1</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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-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**>   </**project**> |

#### ES2

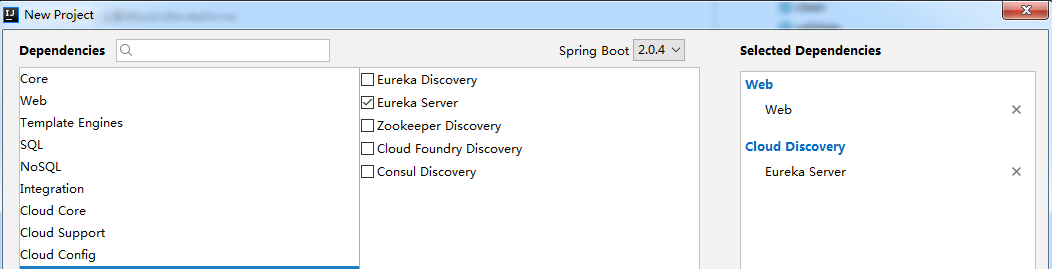
1.新建es2工程



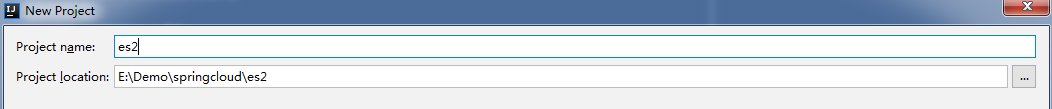
2.配置包名



3.选择依赖



4.选择保存目录



5.创建application.yml

|  |
| --- |
| **eureka:  instance:  hostname:** es2  **client:  serviceUrl:  defaultZone:** http://es1:8801/eureka/ **server:  port:** 8802 |

6.开启服务注册中心

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  @SpringBootApplication @EnableEurekaServer **public class** Es2Application {   **public static void** main(String[] args) {  SpringApplication.*run*(Es2Application.**class**, args);  } } |

7.pom.xml如下

|  |
| --- |
| *<?***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.zhf</**groupId**>  <**artifactId**>es2</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>es2</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**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-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**>   </**project**> |

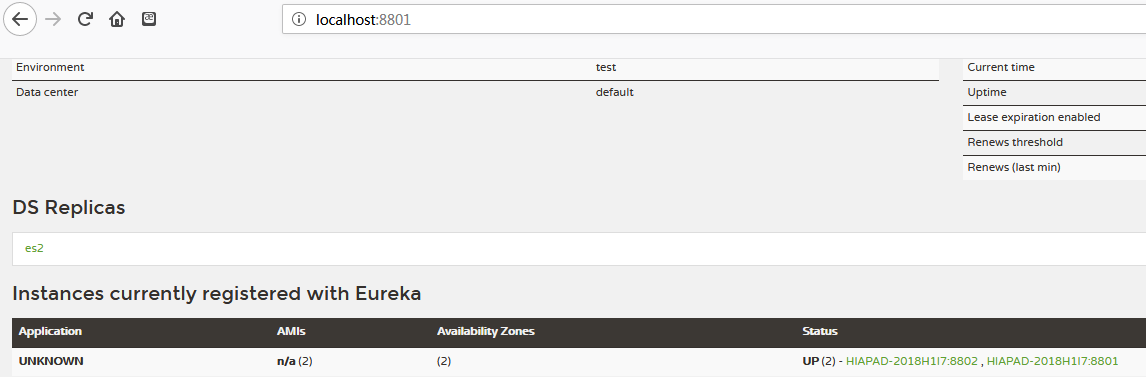
#### 启动

1.修改C:\Windows\System32\drivers\etc的hosts文件，加以下内容

|  |
| --- |
| 127.0.0.1 es1  127.0.0.1 es2 |

2.启动es1和es2

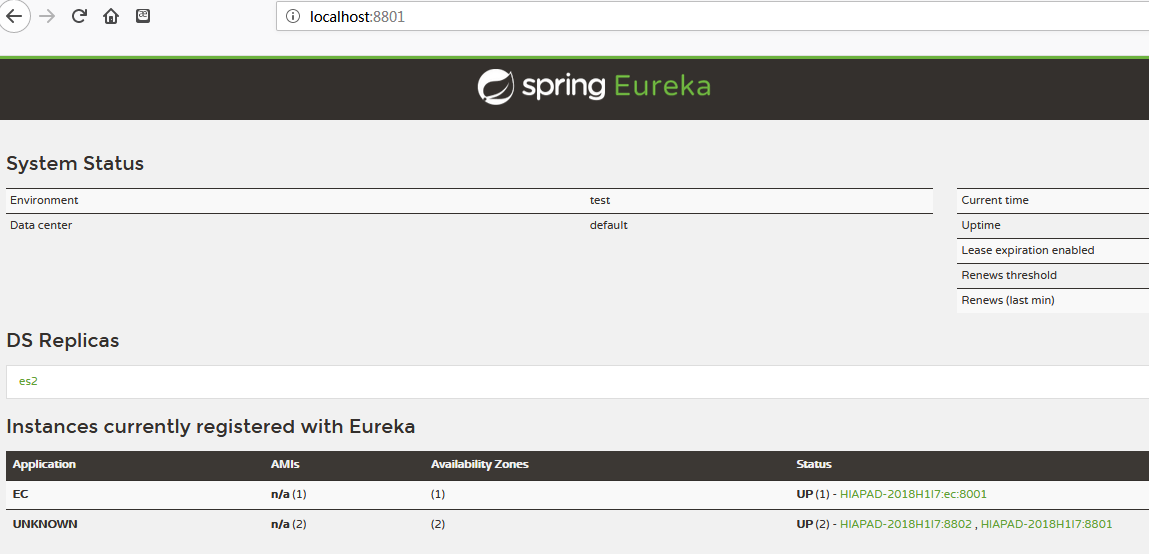
3.浏览器访问http://localhost:8801/ ，可以看到下图



4.修改ec工程的yml文件

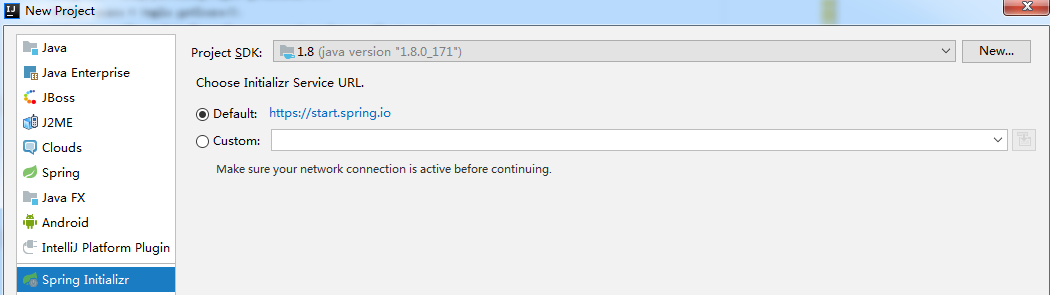
|  |
| --- |
| **server:  port:** 8001  **spring:  application:  name:** ec  **eureka:  client:  serviceUrl:** *# 注明自己的服务注册中心的地址 # defaultZone: http://localhost:8000/eureka/* **defaultZone:** http://localhost:8802/eureka/ |

5.再次刷新浏览器，可以看到ec工程已经注册到服务中

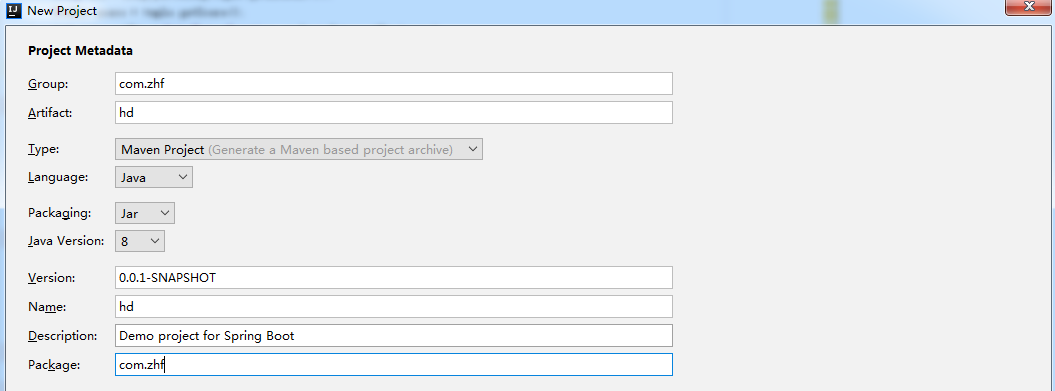


## 十一、断路器监控- Hystrix Dashboard

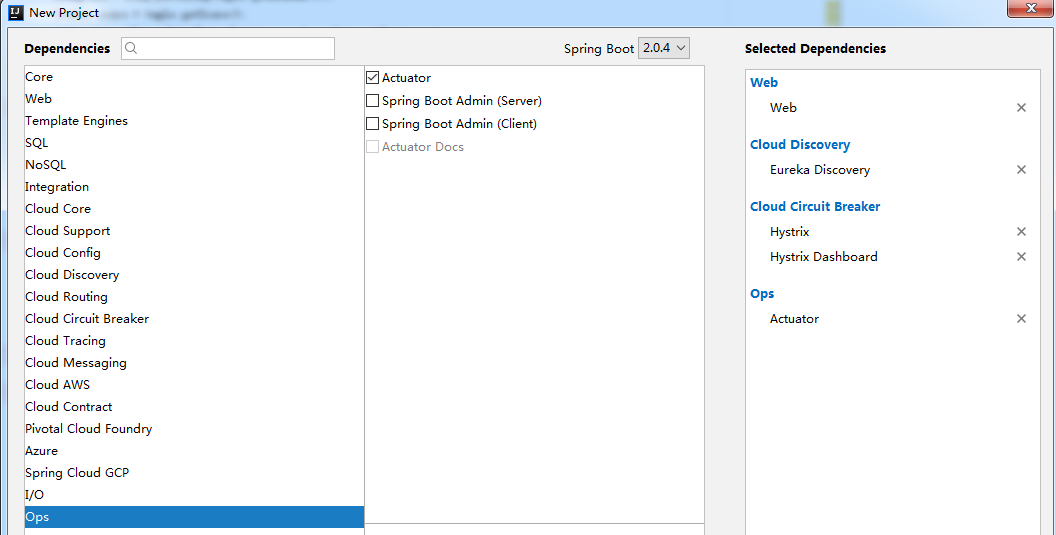
1.新建工程



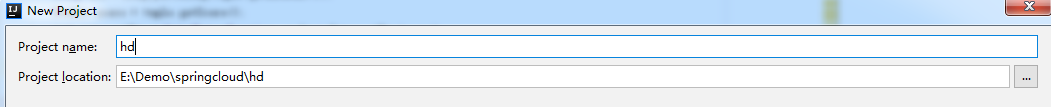
2.设置包名



3.选择依赖



4.指定工程目录



5.设置启动类

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient; **import** org.springframework.cloud.netflix.hystrix.EnableHystrix; **import** org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard; **import** org.springframework.web.bind.annotation.RestController;  @EnableEurekaClient @EnableDiscoveryClient @SpringBootApplication **public class** HdApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(HdApplication.**class**, args);  } } |

6.配置controller

|  |
| --- |
| **package** com.zhf.action;  **import** com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand; **import** org.springframework.beans.factory.annotation.Value; **import** org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker; **import** org.springframework.cloud.netflix.hystrix.EnableHystrix; **import** org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard; **import** org.springframework.stereotype.Component; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  @RestController @EnableHystrix*//开启断路器* @EnableHystrixDashboard @EnableCircuitBreaker @Component **public class** HelloController {  @Value(**"${server.port}"**)  String **port**;   @RequestMapping(**"/index"**)  @HystrixCommand(fallbackMethod = **"indexError"**)  **public** String home() {  **return "hello world i'm port:"** + **port**;  }   **public** String indexError() {  **return "The "** + **port** + **"is error!"**;  } } |

7.配置yml

|  |
| --- |
| **server:  port:** 8900  **spring:  application:  name:** hd  **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/  **management:  endpoints:  web:  exposure:  include:** [**"health"**,**"info"**,**"hystrix.stream"**]  **cors:  allowed-origins: "\*" allowed-methods: "\*"** |

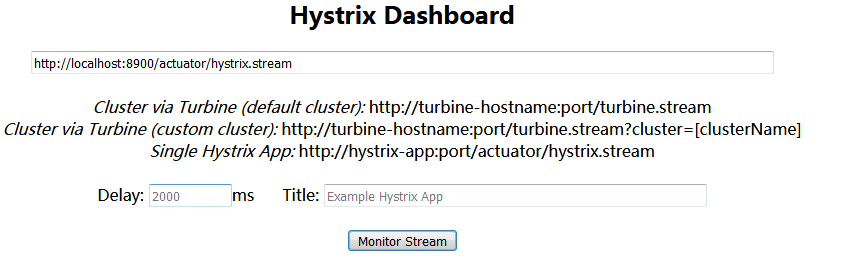
8.POM内容如下

|  |
| --- |
| *<?***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.zhf</**groupId**>  <**artifactId**>hd</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>hd</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**spring-cloud.version**>  </**properties**>   <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**>  </**dependency**>  <**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-hystrix</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-hystrix-dashboard</**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**>   </**project**> |

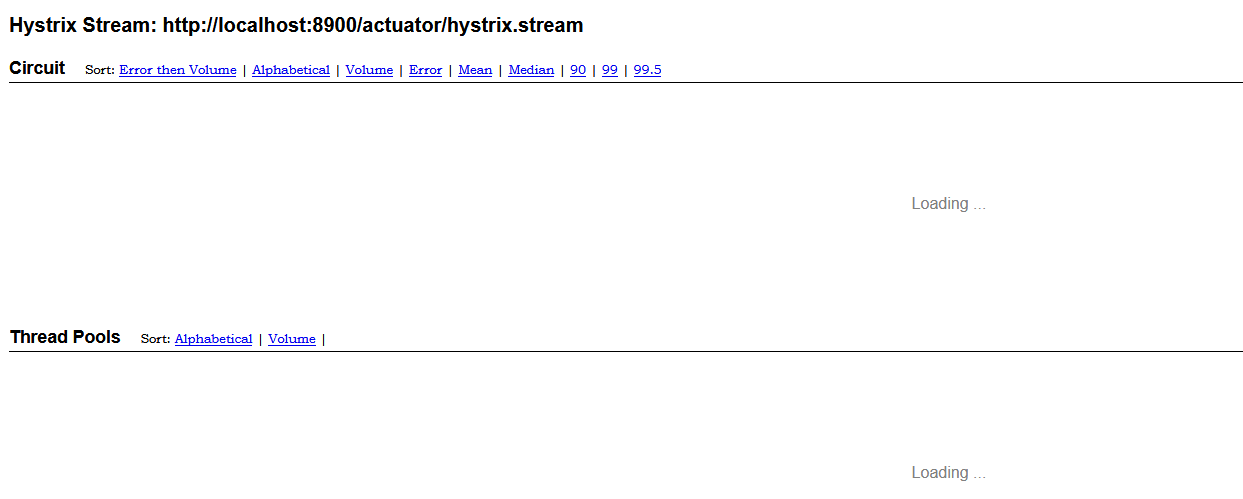
9.启动es

10.访问

输入：http://localhost:8900/actuator/hystrix.stream

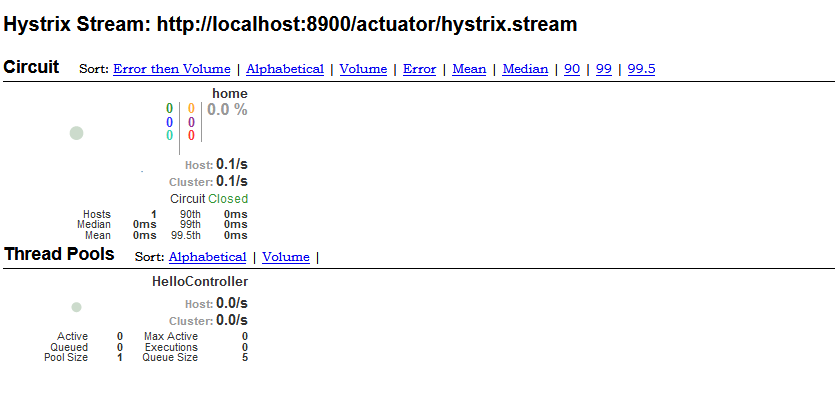


11.出现loading如下



12.访问：http://localhost:8900/index

13.再看hystrix可以显示以下内容



## 十二、断路器聚合监控- Hystrix Turbine

### 准备

参考第十章节内容hd工程创建hd1.

相关内容如下:

0-1创建Action

|  |
| --- |
| **package** com.zhf.action;  **import** com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand; **import** org.springframework.beans.factory.annotation.Value; **import** org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker; **import** org.springframework.cloud.netflix.hystrix.EnableHystrix; **import** org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard; **import** org.springframework.stereotype.Component; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  @RestController @EnableHystrix*//开启断路器* @EnableHystrixDashboard @EnableCircuitBreaker @Component **public class** HelloController {  @Value(**"${server.port}"**)  String **port**;   @RequestMapping(**"/hd1"**)  @HystrixCommand(fallbackMethod = **"indexError"**)  **public** String home() {  **return "hello world i'm port:"** + **port**;  }   **public** String indexError() {  **return "The "** + **port** + **"is error!"**;  } } |

0-2创建启动类

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient;  @EnableEurekaClient @EnableDiscoveryClient @SpringBootApplication **public class** Hd1Application {   **public static void** main(String[] args) {  SpringApplication.*run*(Hd1Application.**class**, args);  } } |

0-3配置yml

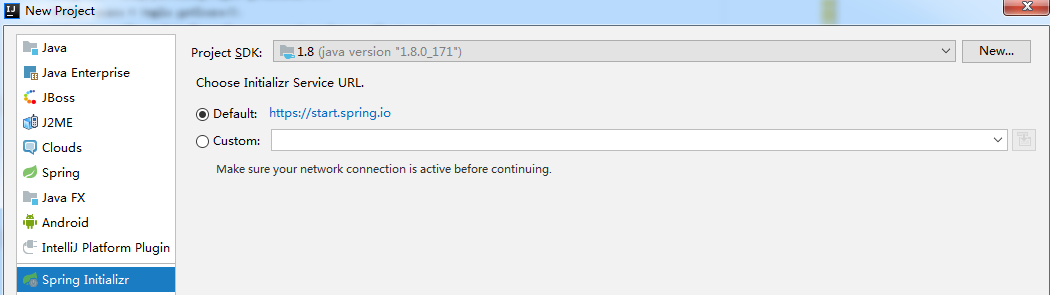
|  |
| --- |
| **server:  port:** 8901  **spring:  application:  name:** hd1  **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/  **management:  endpoints:  web:  exposure:  include:** [**"health"**,**"info"**,**"hystrix.stream"**]  **cors:  allowed-origins: "\*" allowed-methods: "\*"** |

0-4配置pom.xml

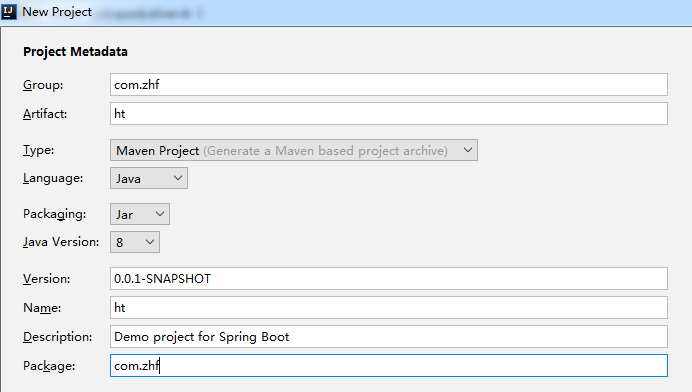
|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.zhf</**groupId**>  <**artifactId**>hd1</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>hd1</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**spring-cloud.version**>  </**properties**>   <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**>  </**dependency**>  <**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-hystrix</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-hystrix-dashboard</**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**>   </**project**> |

### Ht

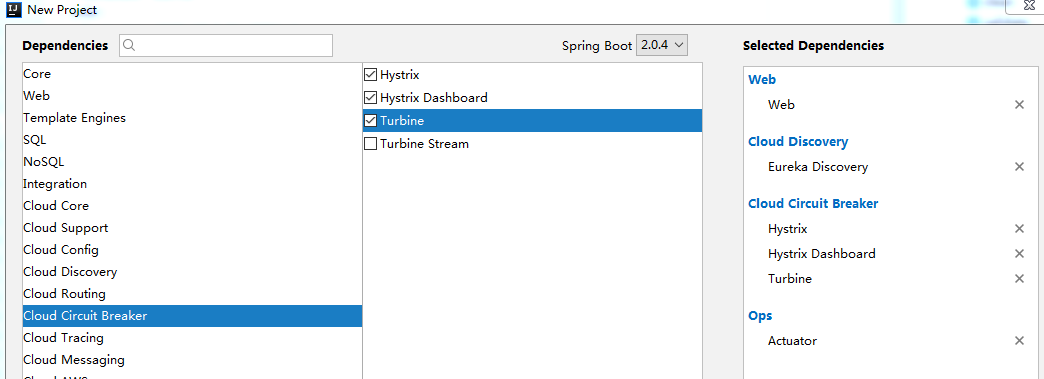
1.新建工程



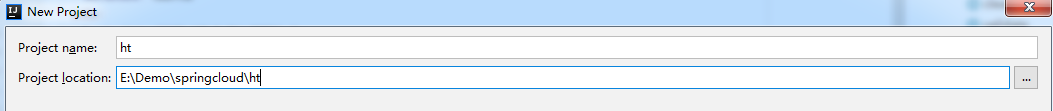
2.配置包路径



3.导入依赖



4.配置工程目录



5.配置yml

|  |
| --- |
| **server:  port:** 9000  **spring:  application:  name:** ht  **eureka:  client:  serviceUrl:  defaultZone:** http://localhost:8000/eureka/ **management:  endpoints:  web:  exposure:  include: "\*"  cors:  allowed-origins: "\*"  allowed-methods: "\*"  turbine:  app-config:** hd,hd1  **aggregator:  clusterConfig:** default  **clusterNameExpression:** new String("default")  **combine-host:** true  **instanceUrlSuffix:  default:** actuator/hystrix.stream |

6.配置启动类

|  |
| --- |
| **package** com.zhf;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.netflix.eureka.EnableEurekaClient; **import** org.springframework.cloud.netflix.hystrix.EnableHystrix; **import** org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard; **import** org.springframework.cloud.netflix.turbine.EnableTurbine; **import** org.springframework.web.bind.annotation.RestController;  @SpringBootApplication @EnableEurekaClient @EnableDiscoveryClient @RestController @EnableHystrix @EnableHystrixDashboard @EnableCircuitBreaker @EnableTurbine **public class** HtApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(HtApplication.**class**, args);  } } |

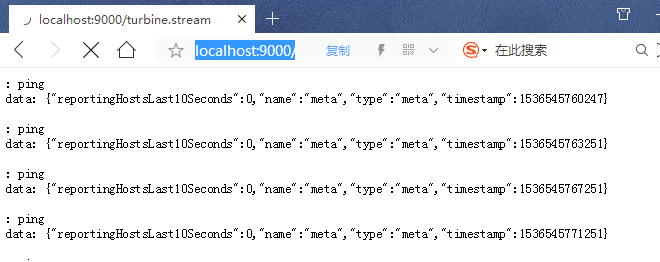
7.pom.xml如下

|  |
| --- |
| *<?***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.zhf</**groupId**>  <**artifactId**>ht</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  <**packaging**>jar</**packaging**>   <**name**>ht</**name**>  <**description**>Demo project for Spring Boot</**description**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.0.4.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.SR1</**spring-cloud.version**>  </**properties**>   <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**>  </**dependency**>  <**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-hystrix</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-hystrix-dashboard</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-turbine</**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**>   </**project**> |

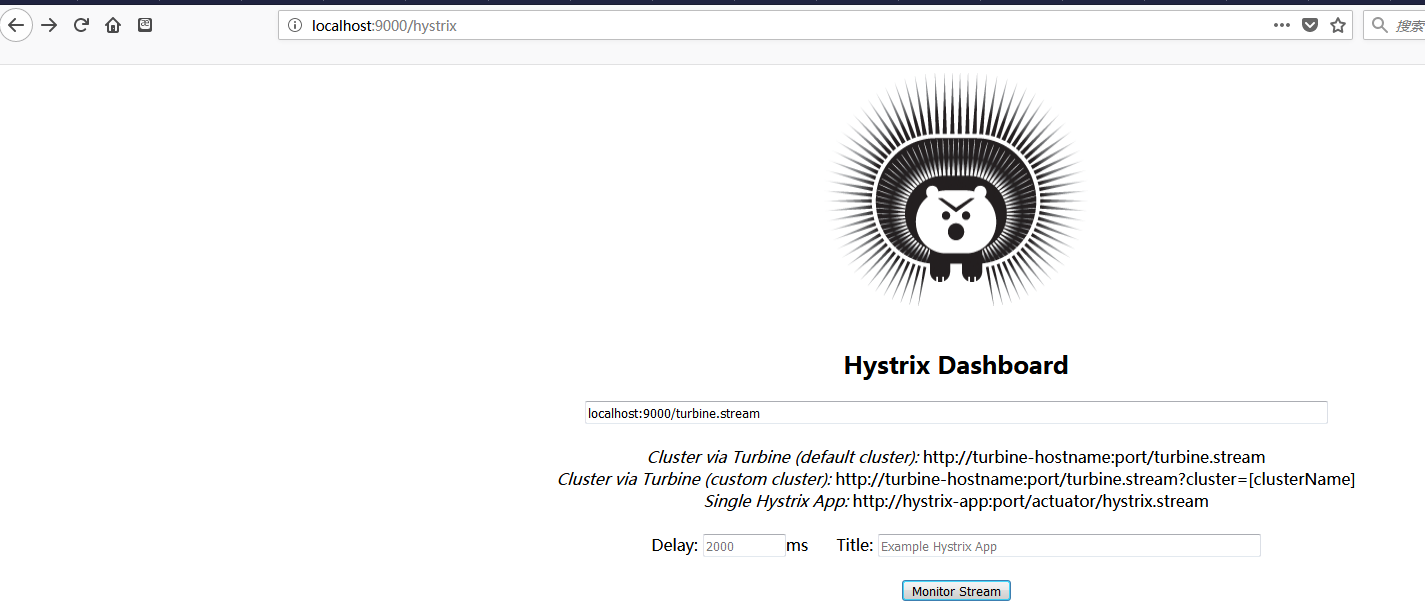
8.依次启动es,hd,hd1及ht工程

9.浏览器访问

http://localhost:9000/turbine.stream



10.浏览器访问: http://localhost:9000/hystrix ,输入 : localhost:9000/turbine.stream



再依次访问: http://localhost:8900/index 和 http://localhost:8901/hd1 可以看到变化

