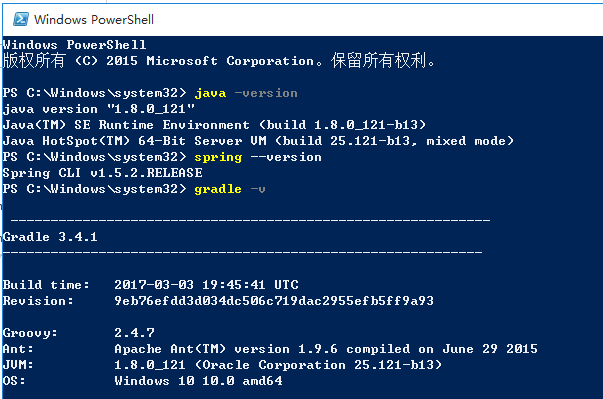
Spring Cloud学习

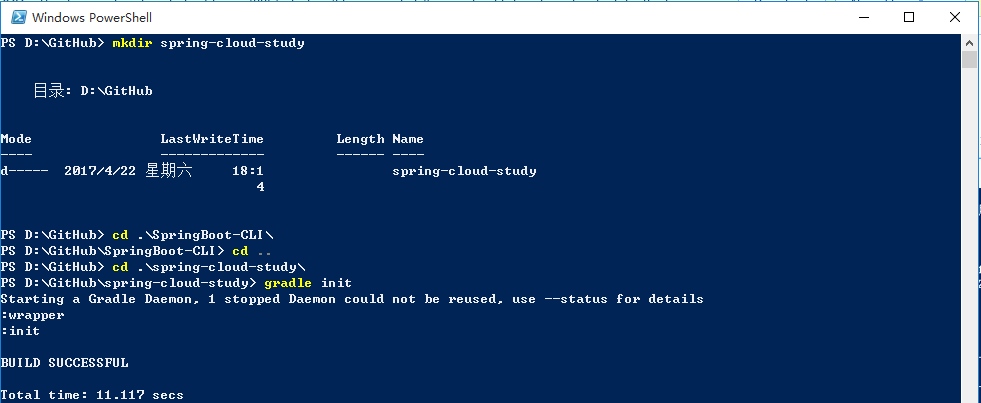
先看一下我的环境， JDK1.8+Spring CLI 1.5.2+Gradle 3.4.1



# Eureka快速入门

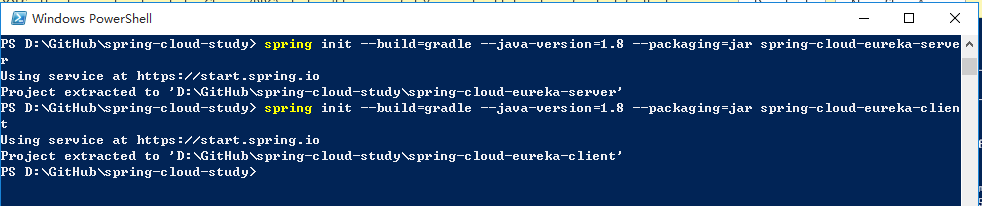
Eureka主要用于注册和发现服务的。

先使用gradle 创建主项目



修改build.gradle文件，使用idea插件。为了照顾有些同学还是很钟意eclipse ide，所以将eclipse插件也加上。然后用idea打开项目。

使用Spring CLI创建两个项目作为模块



同样我们更改一下build.gradle文件, 加入apply plugin: ‘idea’使其支持idea.

接下来，我们更改主项目下的setting.gradle文件，表示这个一个多模块的方案，其实很简单，只需加入include **'spring-cloud-eureka-server'**,**'spring-cloud-eureka-client'**即可, 对比两个模块的build.gradle，发现很多重复的配置，咋办？移到上一级吧，不做重复的轮子是我辈的终极目的。

移除多余后的spring-cloud-eureka-server端的build.gradle将简洁成这样：

version = **'0.0.1-SNAPSHOT'**dependencies {  
 compile(**'org.springframework.cloud:spring-cloud-starter-eureka-server'**)  
}  
  
dependencyManagement {  
 imports {  
 mavenBom **"org.springframework.cloud:spring-cloud-dependencies:Dalston.RELEASE"** }  
}

而spring-cloud-eureka-client的build.gradle是这样子的：

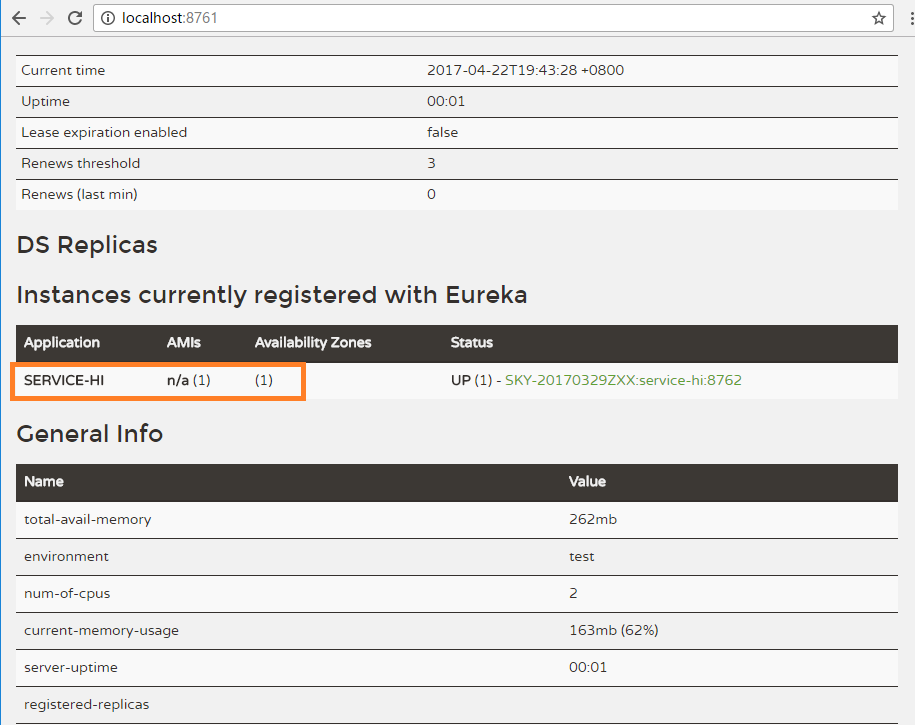
version = **'0.0.1-SNAPSHOT'**dependencies {  
 compile(**'org.springframework.cloud:spring-cloud-starter-eureka'**)  
}  
  
dependencyManagement {  
 imports {  
 mavenBom **"org.springframework.cloud:spring-cloud-dependencies:Dalston.RELEASE"** }  
}

也许你会说dependencyManagement也是重复的配置，一点不错，只是现今阶段只有eureka，后面还会更config等，他们可能并不需要，所以暂时就这样。

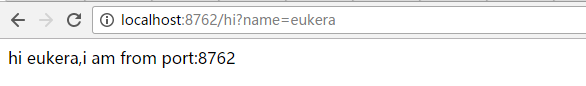
最后分别在spring-cloud-eureka-server和spring-cloud-eureka-client目录下运行

.\gradlew bootRun

然后在Chrome中输入<http://localhost:8761/>



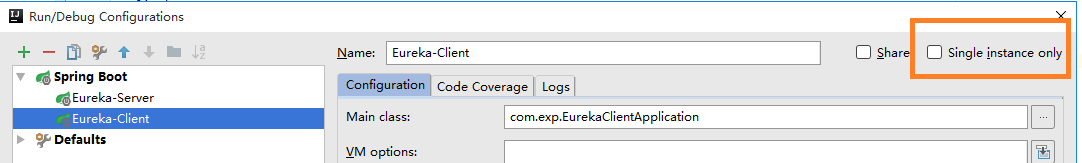
同样在另一个tab输入<http://localhost:8762/hi?name=eukera>



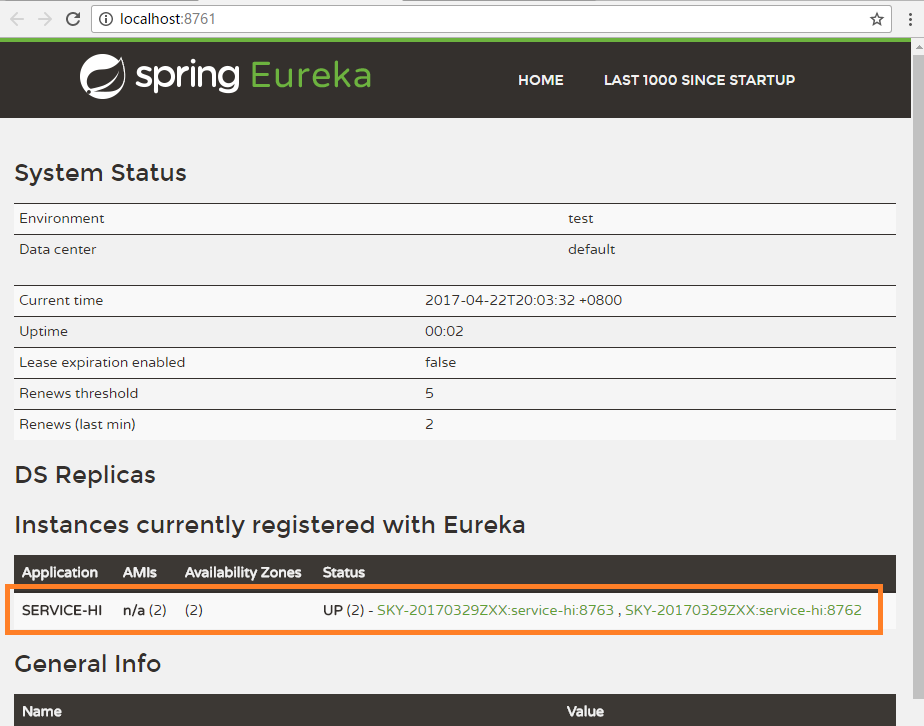
# Rest + Ribbon

ribbon是一个负载均衡客户端，可以很好的控制htt和tcp的一些行为。Feign也用到ribbon，当你使用@ FeignClient，ribbon自动被应用。

将spring-cloud-eureka-client的springboot配置改成可运行多个实例。



然后在application.yml中将端口改为8763,然后运行，就会启动两个服务，并且体现在eureka服务注册页面。



新建一个spring-boot项目

spring init --build=gradle --java-version=1.8 --packaging=jar spring-cloud-ribbon

修改build.gradle以及父项目的setting.gradle。

修改后的build.gradle

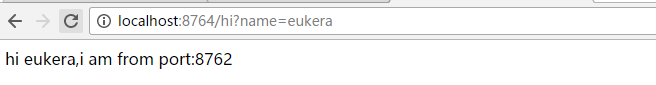
version = **'0.0.1-SNAPSHOT'**dependencies {  
 compile(**'org.springframework.cloud:spring-cloud-starter-eureka'**)  
 compile(**'org.springframework.cloud:spring-cloud-starter-ribbon'**)  
}  
  
dependencyManagement {  
 imports {  
 mavenBom **"org.springframework.cloud:spring-cloud-dependencies:Dalston.RELEASE"** }  
}

至于settings.gradle只是在后面加上’spring-cloud-ribbon’

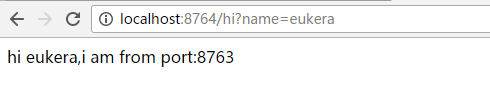
在application.yml中指定本服务的端口为8764，并建一个service和一个controller进行测试。

分别启动所有项目，然后在Chrome中输入<http://localhost:8764/hi?name=eukera>

会发现交替出现



和



这说明当我们通过调用restTemplate.getForObject(“http://SERVICE-HI/hi?name=“+name,String.class)，获取service-hi的方法时，已经做了负载均衡，访问了不同的端口的服务。

此时的架构是这样子的



* 一个服务注册中心，eureka server,端口为8761
* service-hi工程跑了两个副本，端口分别为8762,8763，分别向服务注册中心注册
* sercvice-ribbon端口为8764,向服务注册中心注册
* 当sercvice-ribbon通过restTemplate调用service-hi的hi接口时，因为用ribbon进行了负载均衡，会轮流的调用service-hi：8762和8763 两个端口的hi接口；

# Feign

Feign是一个声明式的web服务客户端，它使得写web服务变得更简单。使用Feign,只需要创建一个接口并注解。它具有可插拔的注解特性，包括Feign 注解和JAX-RS注解。Feign同时支持可插拔的编码器和解码器。spring cloud对Spring mvc添加了支持，同时在spring web中次用相同的HttpMessageConverter。当我们使用feign的时候，spring cloud 整和了Ribbon和eureka去提供负载均衡。

简而言之：

* feign采用的是接口加注解
* feign 整合了ribbon

创建一个新的spring-boot服务

spring init --build=gradle --java-version=1.8 --packaging=jar spring-cloud-feign

同ribbon一样的操作之后，我们在Chrome中输入<http://localhost:8765/hi?name=eukera>

同样发现8762和8763两个端口的数据交替出现。

# 熔断器Hystrix

在微服务架构中，我们将业务拆分成一个个的服务，服务与服务之间可以相互调用（RPC）。为了保证其高可用，单个服务又必须集群部署。由于网络原因或者自身的原因，服务并不能保证服务的100%可用，如果单个服务出现问题，调用这个服务就会出现网络延迟，此时若有大量的网络涌入，会形成任务累计，导致服务瘫痪，甚至导致服务“雪崩”。

为了解决这个问题，就出现熔断器模型。Netflix已经创建了一个名为Hystrix的库来实现熔断器模式。

改造前面的ribbon，使其支持熔断器。

首先引入引用compile(**'org.springframework.cloud:spring-cloud-starter-hystrix'**)

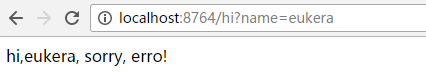
在启动RobbinApplicaiton中声明加上@EnableHystrix

并且在HelloService里加上@HystrixCommand(fallbackMethod = **"hiError"**)

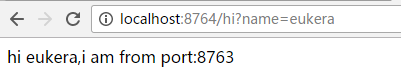
**public** String hiError(String name) {  
 **return** String.*format*(**"hi,%s,sorry,error!"**,name);  
}

启动之后，正常情况下，交替出现8762和8763

停掉一个8762后，就会出现



和



说明熔断器起作用了。

由于feign自带断路模式，而且默认是打开的(也就是feign.hystrix.enabled=false,这个理解起来有点怪异)，所以如果我们停掉一个spring-cloud-eureka-client，就会发现8762始终不会出现了。当然如果不想使用默认的，也想和我们上面的ribbon一样的行为的话，首先要在application.yml中停止使用默认的熔断器。

**feign:  
 hystrix:  
 enabled:** true

不过，经实验发现，这个并不会如ribbon一样交替出现错误和正常显示。而是有时还是会持续出现8763，偶尔出现错误显示而已。

查看Dashboard. 加入引用compile(**'org.springframework.cloud:spring-cloud-starter-hystrix-dashboard'**)

并在启动程序中加上：@EnableHystrixDashboard

然后在Chrome中输入<http://localhost:8764/hystrix>

或者<http://localhost:8765/hystrix>

# 路由网关zuul

zuul的主要功能是路由和过滤器。路由功能是微服务的一部分，比如／api/user映射到user服务，/api/shop映射到shop服务。zuul实现了负载均衡

创建一个新的spring-boot服务

spring init --build=gradle --java-version=1.8 --packaging=jar spring-cloud-zuul

更改build.gradle和settings.gradle

# 附录：

本文主要参考<http://blog.csdn.net/forezp/article/details/70148833>