# Spring Cloud Bus

1. **Spring Cloud Bus是什么？**

Spring Cloud Bus隶属于Spring Cloud体系，使用轻量级的消息代理（例如RabbitMQ、Kafka等）连接分布式系统的节点，通过广播传播状态的更改（例如配置的更新）或者其他的管理指令。如果你熟悉Spring Boot Actuator，可以把Spring Cloud Bus想象成一个分布式的Spring Boot Actuator。

1. **Spring Cloud Bus意义**

Spring Cloud Bus使用轻量级的消息代理（例如RabbitMQ、Kafka等），因此具备了消息代理的特性。

1. **实时性高**

与REST服务的“请求-响应”模式不同，消息总线的实时性非常高。使用了消息总线，生产者一方只要把消息发送到队列后，就可以响应用户。无须等待处理结果，实现了异步处理；同时，对于消费者而言，消费者对于消息的到达感知也非常及时。消费者会对消息总线进行监听，只要有消息进入队列，就立即得到通知。

这种优势是REST服务所不能具备的。在REST服务中，要想及时获取到更新通知，就不得不进行轮询。这往往非常低效。

1. **生产者与消费者解耦**

在消息总线中，生产者负责将消息发送到队列中，而消费者把消息从队列中取出来。生产者无须等待消费者启动，消费者也无须关心生产者是否已经处于就绪状态。所以，这种模式能很好地实现生产者与消费者的解耦；对比REST服务，服务调用方必须等待服务的提供方准备就绪，否则调用失败。

1. **故障率低**

消息总线拥有对其他通信方式更高的成功率。一方面，生产者与消费者之间实现了解耦，所以，生产者与消费者之间不存在强关联关系，即便是生产者或消费者任意一方掉线，也不会影响消息的最终到达；另一方面，消息总线往往结合数据库来实现消息的持久化，并设置状态标识。只有消息消费成功，才会去修改状态标识。

消息总线同时还承担着缓冲区的作用。大量业务消息首先会进入消息队列进行缓存，消息的消费者可以根据自己的处理能力来进行消费，所以不管消息的数据量有多少，都不会对消费者造成冲击。

1. **程序解耦**

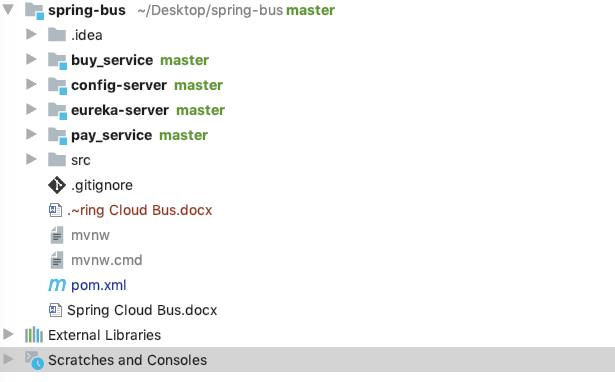
Spring Cloud Bus底层采用Spring Cloud Stream，Spring Cloud Stream针对不同的消息中间件封装了统一的Binder并暴露给应用核心，免去了接入不同消息中间件对应用程序的影响。

1. **Spring Cloud Bus实现消息总线**
2. **说明**

下面举例说明如何使用消息总线在程序不重启的情况下动态刷新配置。

本例使用RabbitMQ作为实现Spring Cloud Bus消息总线的中间件，更新各个微服务节点配置功能。

项目架构图如下



本例采用maven作为项目依赖管理工具，使用父pom的方式统一管理Spring Boot和Spring Cloud的版本，降低项目维护成本；同时，在未来，可以将父pom项目，抽象成一个maven的parent配置，发布在自己或公司的maven仓库中，统一版本管理的同时，简化各个分布式系统的配置。

项目有4个module：

1. eureka-server:服务的发现和注册中心；
2. config-server:分布式系统中的配置服务中心，用来读取远程配置中心的配置，本例中使用Git作为远程配置中心；
3. buy\_service:购买服务，模拟微服务系统中需要读取配置的服务节点。
4. pay\_service:支付服务，模拟微服务系统中需要读取配置的服务节点。
5. **实现环境**

JDK : 1.8.0\_181;

Spring Boot : 2.0.3.RELEASE;

Spring Cloud :Finchley.RELEASE;

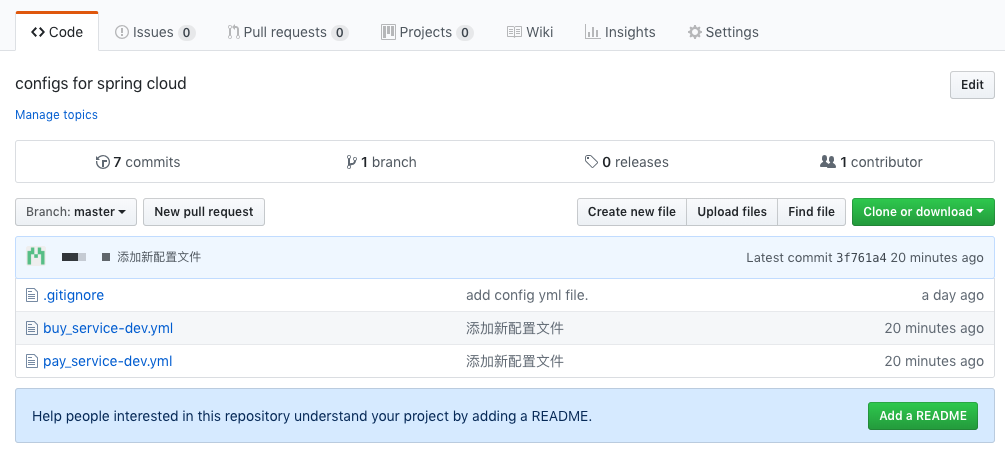
Rabbit MQ : 3.7.9

1. **实现步骤**

本例需要先安装Rabbit MQ，请到Rabbit MQ官网http://www.rabbitmq.com/下载安装。

1. 在Github上创建配置信息。

在Github上创建名称为configs的repository，并在configs目录下新建两个配置文件，如下图：



如上图所示，两个配置文件代码如下：

buy\_service-dev.yml中的代码如下：

spring:

redis:

port: 9091

pay\_service-dev.yml中的代码如下：

spring:

redis:

port: 9092

创建maven项目，pom文件如下：

<?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.example</groupId>

<artifactId>spring-bus</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-bus</name>

<description>spring-bus</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.3.RELEASE</version>

<relativePath/>

</parent>

<modules>

<module>eureka-server</module>

<module>config-server</module>

<module>buy\_service</module>

<module>pay\_service</module>

</modules>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

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

<spring-cloud.version>Finchley.RELEASE</spring-cloud.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-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>

右键项目，添加模块eureka-server，pom文件如下：

<?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.example</groupId>

<artifactId>eureka-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>eureka-server</name>

<description>eureka-server</description>

<parent>

<groupId>com.example</groupId>

<artifactId>spring-bus</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<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-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

在eureka-server项目启动类上添加@EnableEurekaServer注解。

在eureka-server配置文件application.yml中添加如下代码：

server:

port: 8889

eureka:

instance:

hostname: localhost

client:

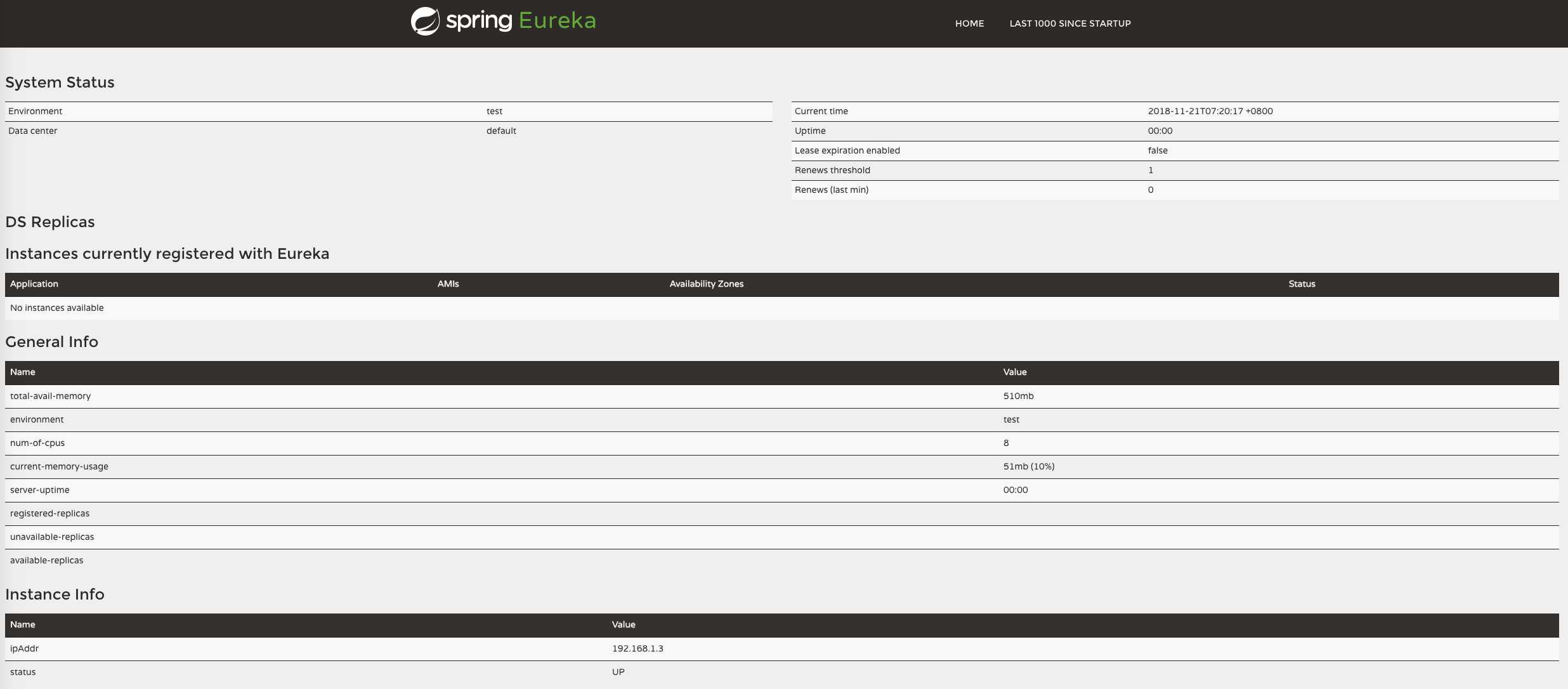
registerWithEureka: false

fetchRegistry: false

serviceUrl:

defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

启动项目，成功后访问：<http://127.0.0.1:8889/> 如下图



右键项目，添加模块config-server，pom文件如下：

<?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.example</groupId>

<artifactId>config-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>config-server</name>

<description>config-server</description>

<parent>

<groupId>com.example</groupId>

<artifactId>spring-bus</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<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-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-config-server</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

在config-server项目启动类上添加@EnableConfigServer和

@EnableEurekaClient注解

在config-server配置文件application.properties中添加如下代码：

spring.application.name=config-server

server.port=8888

spring.cloud.config.server.git.uri=https://github.com/juxincoder/configs.git

spring.cloud.config.server.git.searchPaths=configs

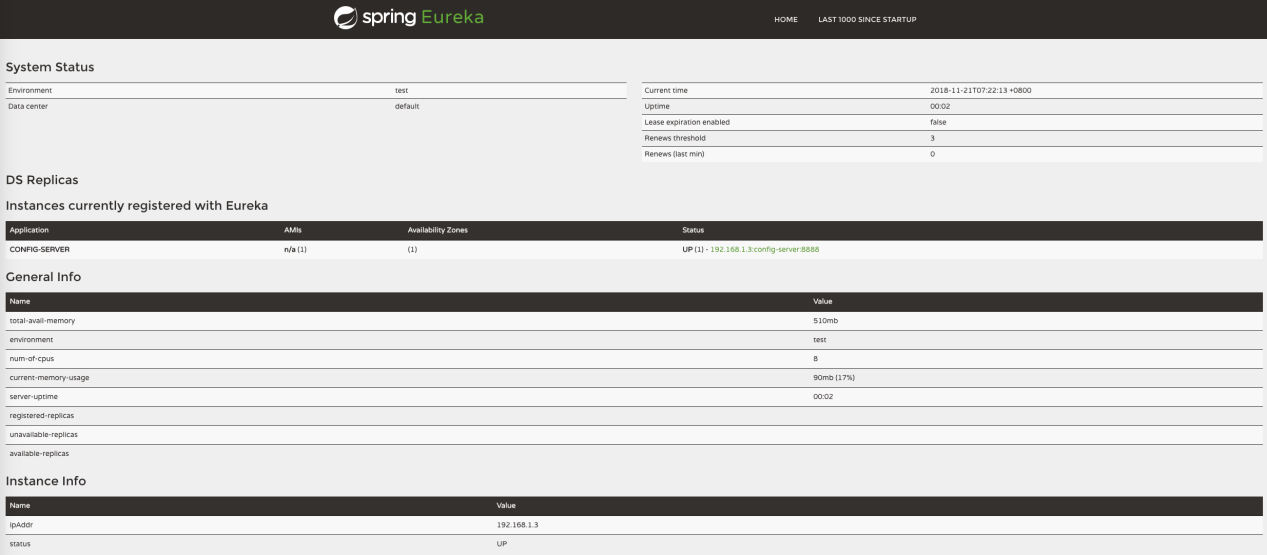
spring.cloud.config.label=master

spring.cloud.config.server.git.username=

spring.cloud.config.server.git.password=

eureka.client.serviceUrl.defaultZone=http://localhost:8889/eureka/

启动项目，成功后访问：<http://127.0.0.1:8889/> 如下图，在【Instance currently registered with Eureka】中可以看到【CONFIG-SERVER】服务已经注册到Eureka 注册中心。



右键项目，添加模块buy\_service，pom文件如下：

<?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.example</groupId>

<artifactId>buy\_service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>buy\_service</name>

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

<parent>

<groupId>com.example</groupId>

<artifactId>spring-bus</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</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-bus-amqp</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

在buy\_service项目启动类上添加@EnableEurekaClient注解

在buy\_service项目的resources文件夹下新建配置文件bootstrap.yml，添加如 下代码：

spring.application.name=buy\_service

spring.cloud.config.label=master

spring.cloud.config.profile=dev

eureka.client.serviceUrl.defaultZone=http://localhost:8889/eureka/

spring.cloud.config.discovery.enabled=true

spring.cloud.config.discovery.serviceId=config-server

server.port=8882

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

spring.cloud.bus.enabled=true

spring.cloud.bus.trace.enabled=true

为了方便验证【buy\_service】项目已经读取到【Git】上的配置文件，我们在项目 中添加一个rest接口，将读取到的配置文件作为接口的返回值。

新建【controller】包，在【controller】包下新建【TestController.java】文件，代 码如下：

package com.example.buy\_service.controller;

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;

@RefreshScope

@RestController

public class TestController {

@Value("${spring.redis.port}")

String redisPort;

@RequestMapping(value = "/redis/port")

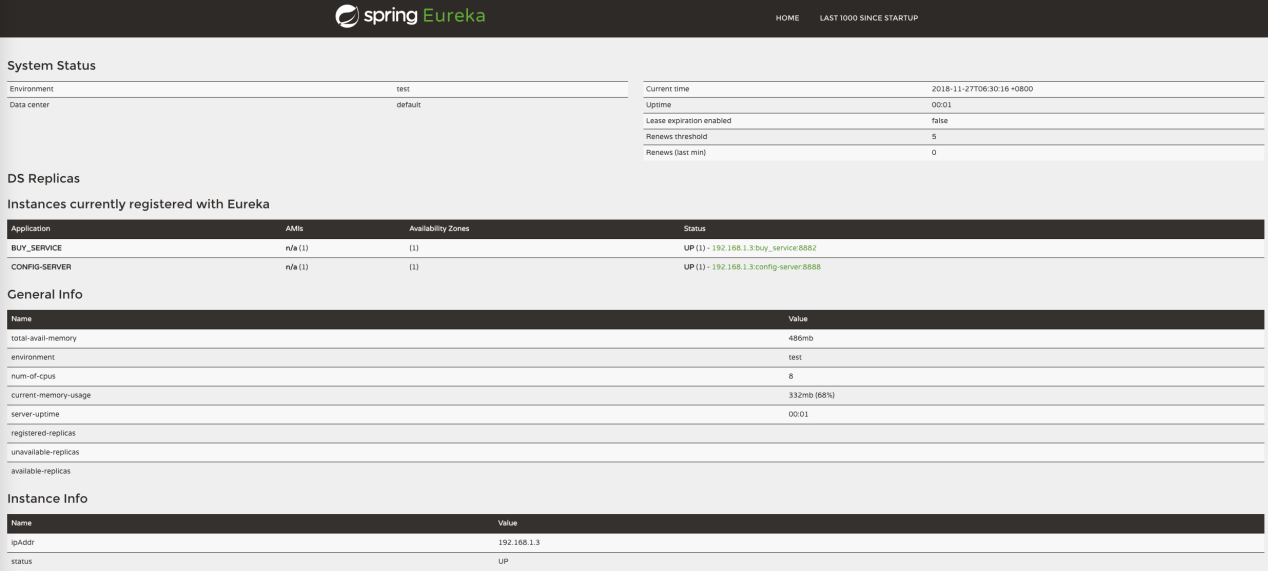
public String hi() {

return redisPort;

}

}

启动项目，成功后访问：<http://127.0.0.1:8889/> 如下图，在【Instance currently registered with Eureka】中可以看到【BUY\_SERVICE】服务已经注册到Eureka 注册中心；同时访问：<http://127.0.0.1:8882/redis/port> 可以在浏览器上看到9091， 说明读取【Git】配置文件成功。



右键项目，添加模块pay\_service，pom文件如下：

<?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.example</groupId>

<artifactId>pay\_service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>pay\_service</name>

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

<parent>

<groupId>com.example</groupId>

<artifactId>spring-bus</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</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-bus-amqp</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

在pay\_service项目启动类上添加@EnableEurekaClient注解

在pay\_service项目的resources文件夹下新建配置文件bootstrap.yml，添加如 下代码：

spring.application.name=buy\_service

spring.cloud.config.label=master

spring.cloud.config.profile=dev

eureka.client.serviceUrl.defaultZone=http://localhost:8889/eureka/

spring.cloud.config.discovery.enabled=true

spring.cloud.config.discovery.serviceId=config-server

server.port=8882

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

spring.cloud.bus.enabled=true

spring.cloud.bus.trace.enabled=true

为了方便验证【buy\_service】项目已经读取到【Git】上的配置文件，我们在项目 中添加一个rest接口，将读取到的配置文件作为接口的返回值。

新建【controller】包，在【controller】包下新建【TestController.java】文件，代 码如下：

package com.example.pay\_service.controller;

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

public class TestController {

@Value("${spring.redis.port}")

String redisPort;

@RequestMapping(value = "/redis/port")

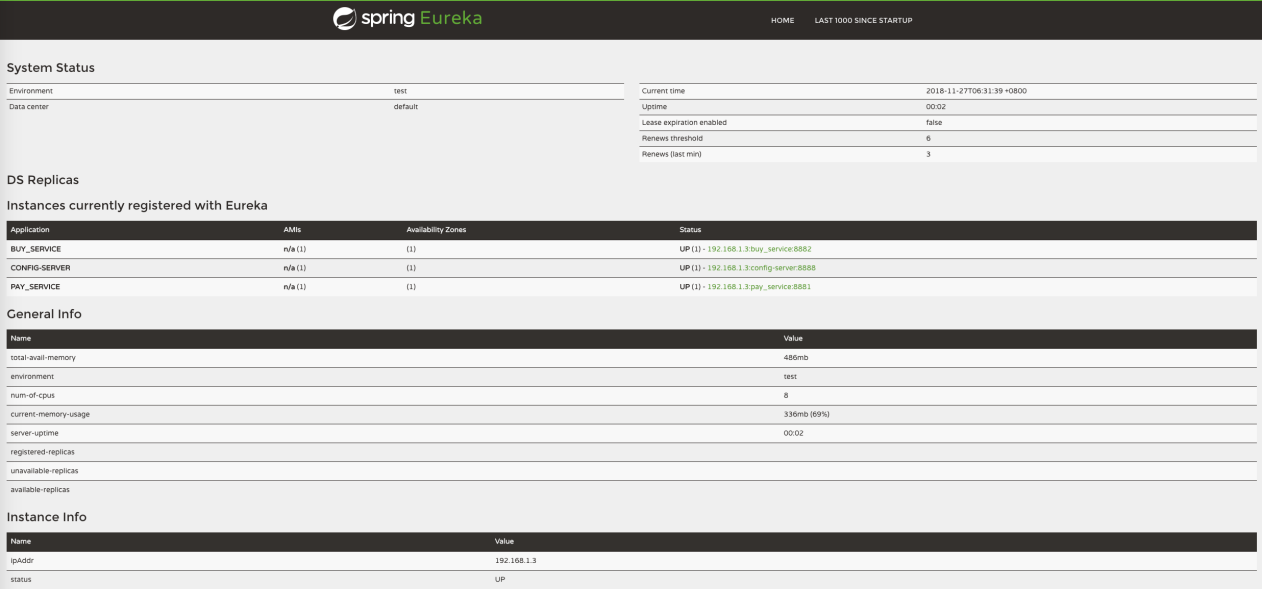
public String hi() {

return redisPort;

}

}

启动项目，成功后访问：<http://127.0.0.1:8889/> 如下图，在【Instance currently registered with Eureka】中可以看到【PAY\_SERVICE】服务已经注册到Eureka 注册中心；同时访问：[http://127.0.0.1:8881/redis/port](http://127.0.0.1:8882/redis/port) 可以在浏览器上看到9092， 说明读取【Git】配置文件成功。



至此，我们实现了两个微服务【buy\_service】和【pay\_service】各自从配置中心 【config-server】中读取远程【Git】目录中的配置文件。

但是，这种方式是在项目加载的时候读取的配置文件，如果远程【Git】目录中的 配置文件发生变化，微服务中无法读取到变化的配置。

下面说明如何在微服务中利用spring bus在不重启项目的情况下读取变化的配置 文件。

在【buy\_service】和【pay\_service】的【bootstrap.yml】文件中添加如下配置：

management.endpoints.web.exposure.include=bus-refresh

说明：此配置的含义是在当前服务中暴露/actuator/bus-refresh的http端点，通

过向此端点发送POST请求，可以刷新所有服务的配置。

在配置文件中添加好上面配置后，重新启动两个服务，访问

<http://127.0.0.1:8881/redis/port> 会看到浏览器上显示9092，

<http://127.0.0.1:8882/redis/port> 会看到浏览器上显示9091。

此时修改远程【Git】的配置文件，将【buy\_service.yml】中的端口号改成9092，

【pay\_service.yml】中的端口号改成9091。

修改好后，通过如下命令

curl -X POST <http://127.0.0.1:8082/actuator/bus-refresh>

发送成功后，访问

<http://127.0.0.1:8881/redis/port> 会看到浏览器上显示9091，

<http://127.0.0.1:8882/redis/port> 会看到浏览器上显示9092。

至此，通过spring bus实现了微服务配置的不重启动态刷新。