

# 一、Gateway服务网关

Spring Cloud Gateway 是 Spring Cloud 新推出的网关框架，之前是 Netflix Zuul。网关通常在项目中为了简化前端的调用逻辑，同时也简化内部服务之间互相调用的复杂度；具体作用就是转发服务，接收并转发所有内外部的客户端调用；其他常见的功能还有权限认证，限流控制等等。

## 1、创建springcloud-gateway工程

### pom添加

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>2.2.4.RELEASE</version>
9         <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>com.soft863</groupId>
12    <artifactId>springcloud-gateway</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <name>springcloud-gateway</name>
15    <description>Demo project for Spring Boot</description>
16
17    <properties>
18        <java.version>1.8</java.version>
19        <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
20    </properties>
21
22    <dependencies>
23        <dependency>
24            <groupId>org.springframework.cloud</groupId>
25            <artifactId>spring-cloud-starter-gateway</artifactId>
26        </dependency>
27        <dependency>
28            <groupId>org.springframework.cloud</groupId>
29            <artifactId>spring-cloud-starter-netflix-eureka-
  client</artifactId>
30        </dependency>
31
32        <dependency>
33            <groupId>org.springframework.boot</groupId>
34            <artifactId>spring-boot-devtools</artifactId>
35            <scope>runtime</scope>
36            <optional>true</optional>
37        </dependency>
```

```

38         <dependency>
39             <groupId>org.springframework.boot</groupId>
40             <artifactId>spring-boot-starter-test</artifactId>
41             <scope>test</scope>
42             <exclusions>
43                 <exclusion>
44                     <groupId>org.junit.vintage</groupId>
45                     <artifactId>junit-vintage-engine</artifactId>
46                 </exclusion>
47             </exclusions>
48         </dependency>
49     </dependencies>
50
51     <dependencyManagement>
52         <dependencies>
53             <dependency>
54                 <groupId>org.springframework.cloud</groupId>
55                 <artifactId>spring-cloud-dependencies</artifactId>
56                 <version>${spring-cloud.version}</version>
57                 <type>pom</type>
58                 <scope>import</scope>
59             </dependency>
60         </dependencies>
61     </dependencyManagement>
62
63     <build>
64         <plugins>
65             <plugin>
66                 <groupId>org.springframework.boot</groupId>
67                 <artifactId>spring-boot-maven-plugin</artifactId>
68             </plugin>
69         </plugins>
70     </build>
71
72 </project>
73

```

## yaml配置

创建配置文件application-dev1.yml，添加内容

```

1  server:
2      port: 2005
3
4  spring:
5      application:
6          name: gateway-service
7      cloud:          # spring cloud gateway 路由配置方式
8          gateway:
9              routes:
10                 - id: USER-SERVICE          #网关路由到用户服务user-service, ID可以随意写
11                   uri: http://localhost:2003
12                   #路由断言，可配置映射路径
13                 predicates:
14                 - Path=/login/**

```

```

15
16
17
18 eureka:
19   client:
20     service-url:
21       defaultZone: http://localhost:8761/eureka/
22   instance:
23     prefer-ip-address: true

```

创建application.yml配置文件，添加如下内容

```

1 spring:
2   profiles:
3     active: dev1

```

## 启动类配置

添加@EnableDiscoveryClient注解

## 2、provider中添加测试服务

在springcloud-provider中添加LoginController类，代码如下：

```

1 package com.soft863.controller;
2
3 import com.soft863.entity.User;
4 import org.springframework.beans.factory.annotation.Value;
5 import org.springframework.web.bind.annotation.*;
6
7 @RestController
8 @RequestMapping("/login")
9 public class LoginController {
10
11     @Value("${server.port}")
12     String port;
13
14     @GetMapping("/test2")
15     public String log() {
16         return "hello user";
17     }
18
19     @RequestMapping(value = "/getUser", method = RequestMethod.GET)
20     public String getUser(@RequestParam String name) {
21
22         return name + "恭喜您，登录成功， " + "我来自于端口:" + port;
23     }
24
25     @RequestMapping(value = "/getUserByID2", method = RequestMethod.GET)
26     public User login(@RequestParam Integer id) {
27         User user = new User();
28         user.setId(1);
29         user.setUsername("admin");
30         user.setPassword("123456");

```

```

31         return user;
32     }
33
34     @RequestMapping(value = "/getUserByID2/{id}", method =
RequestMethod.GET)
35     public User login1(@PathVariable int id) {
36         User user = new User();
37         user.setId(1);
38         user.setUsername("admin");
39         user.setPassword("123456");
40         return user;
41     }
42
43     @GetMapping("/getUser2")
44     public String getUser(){
45         return "执行成功";
46     }
47 }
48

```

## 测试：

依次启动server、provider和gateway三个工程

浏览器中输入：<http://localhost:2005/login/test2>

相当于请求2003接口

输入：<http://localhost:2005/test> 请求不到地址，是因为路由断言为/login/\*\*，可去掉/login再次尝试

## 3、面向服务的路由

创建application-dev2.yml，修改application.yml配置，指向dev2

```

1  server:
2      port: 2005
3
4  spring:
5      application:
6          name: gateway-service
7      cloud:          # spring cloud gateway 路由配置方式
8          gateway:
9              routes:
10                 - id: USER-SERVICE          #网关路由到用户服务user-service,ID可以随意写
11                   uri: lb://USER-SERVICE    # 代理的服务地址: lb表示从eureka中获取具体服务
12                   #路由断言，可配置映射路径
13                   predicates:
14                       - Path=/login/**
15
16
17  eureka:
18      client:
19          service-url:
20              defaultZone: http://localhost:8762/eureka/

```

```
21 instance:
22   prefer-ip-address: true
```

## 测试

重启Gateway工程

浏览器中输入: <http://localhost:2005/login/test2>

## 4、路由前缀

在gateway中可以通过配置路由的过滤器PrefixPath, 实现映射路径中地址的添加;

通过 PrefixPath=/xxx 来指定了路由要添加的前缀。新建配置文件application-dev3.yml, 配置如下:

```
1  server:
2    port: 2005
3
4  spring:
5    application:
6      name: gateway-service
7    cloud:      # spring cloud gateway 路由配置方式
8      gateway:
9        routes:
10       - id: USER-SERVICE      #网关路由到用户服务user-service,ID可以随意写
11         uri: lb://USER-SERVICE # 代理的服务地址; lb表示从eureka中获取具体服务
12         #路由断言, 可配置映射路径
13         predicates:
14           - Path=/**
15         filters:
16           - PrefixPath=/login
17
18
19  eureka:
20    client:
21      service-url:
22        defaultZone: http://localhost:8761/eureka/
23    instance:
24      prefer-ip-address: true
25
26
```

修改application.yml配置, 指向dev3

测试:

重启Gateway工程

<http://localhost:2005/login/test2> (不可用)

<http://localhost:2005/test2> (可用)

<http://localhost:2003/login/test2> (可用)

## 二、Config分布式配置

# 1、码云配置

在分布式系统中，由于服务数量非常多，配置文件分散在不同的微服务项目中，管理不方便。为了方便配置文件集中管理，需要分布式配置中心组件。在Spring Cloud中，提供了Spring Cloud Config，它支持配置文件放在配置服务的本地，也支持放在远程Git仓库（GitHub、码云）。



## 新建仓库

仓库名称 ✓

springcloud-config

归属

陌赤

路径

springcloud-config

仓库地址: <https://gitee.com/superjean/springcloud-config>

仓库介绍 非必填

用简短的语言来描述一下吧

是否开源

☐ 私有

☒ 公开

任何人都可以访问该仓库的代码和其他任何形式的资源

选择语言

Java

添加 .gitignore

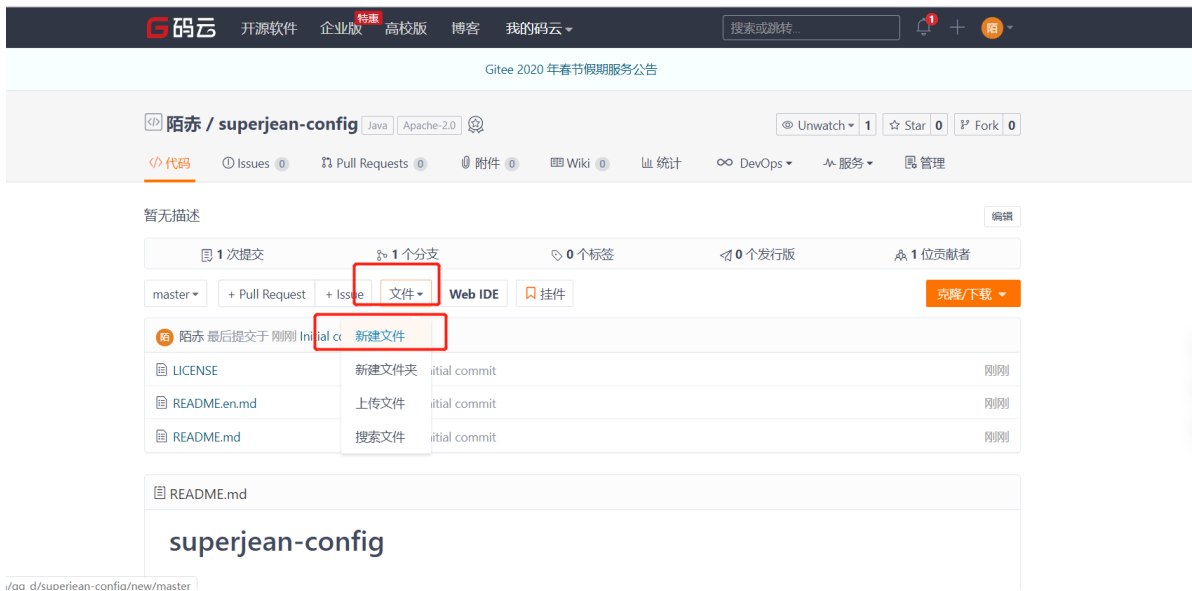
JetBrains

添加开源许可证

Apache-2.0

点此快速选择许可证

☒ 使用Readme文件初始化这个仓库



命名: user-dev.yml

添加如下代码:

```
1 eureka:
2   client:
3     service-url:
4       defaultZone: http://localhost:8761/eureka/
5   instance:
6     ip-address: 127.0.0.1
7     prefer-ip-address: true
8 server:
9   port: 2006
10 spring:
11   application:
12     name: user-server
13   datasource:
14     driver-class-name: com.mysql.jdbc.Driver
15     password: root
16     url: jdbc:mysql://localhost:3306/soft863db?
17     useUnicode=true&characterEncoding=utf-8&useSSL=true&serverTimezone=UTC
18     username: root
```

springcloud-config / user-dev.yml 提示: 输入 / 可以将文件创建到新文件夹下

```
1 eureka:
2   client:
3     service-url:
4       defaultZone: http://localhost:8761/eureka/
5   instance:
6     ip-address: 127.0.0.1
7     prefer-ip-address: true
8 server:
9   port: 2006
10 spring:
11   application:
12     name: user-server
13   datasource:
14     driver-class-name: com.mysql.jdbc.Driver
15     password: root
16     url: jdbc:mysql://localhost:3306/soft863db?useUnicode=true&characterEncoding=utf-8&useSSL=true&serverTimezone=UTC
17     username: root
```

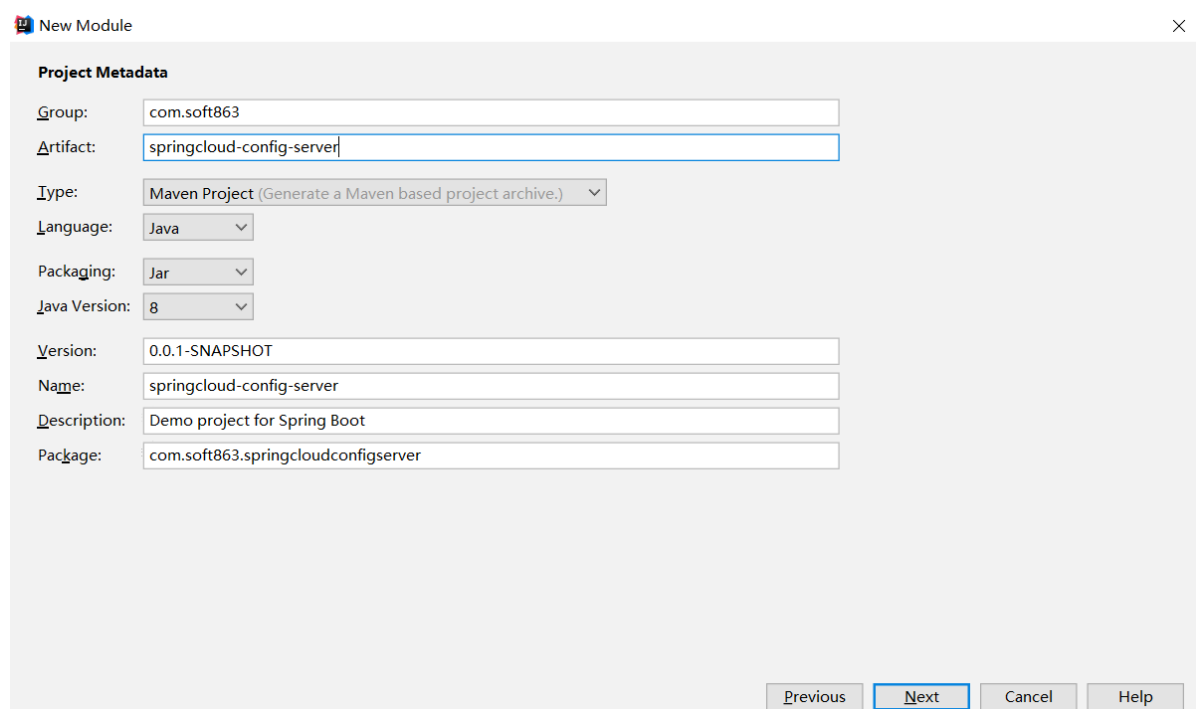
点击提交

复制地址:

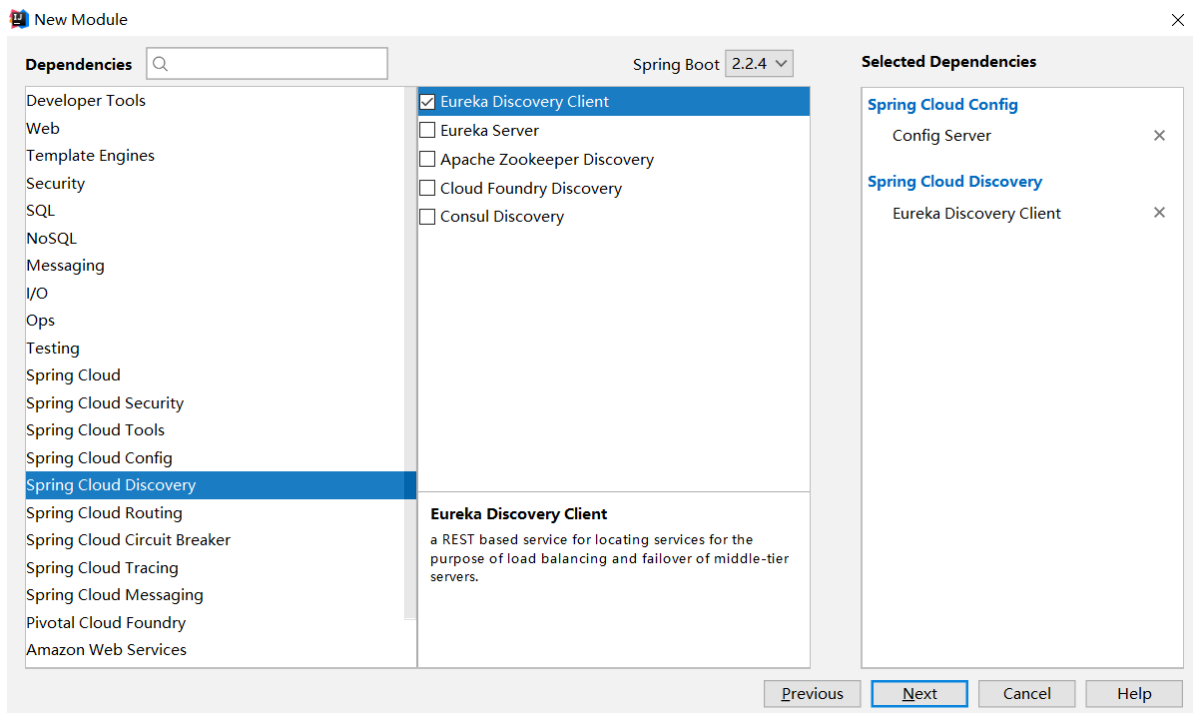


## 2、创建配置中心服务端

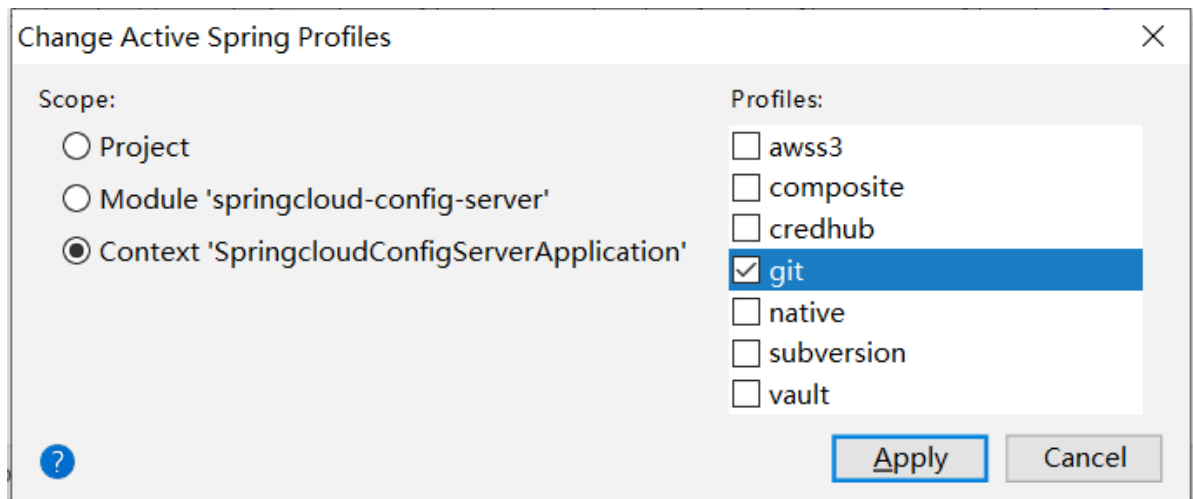
创建工程springcloud-config-server







启动类上添加注解: @EnableConfigServer、@EnableEurekaClient



编写配置文件

```

1  server:
2    port: 8763
3  spring:
4    application:
5      name: config-server
6    cloud:
7      config:
8        server:
9        git:
10         uri: https://gitee.com/mochiwcj/springcloud-config.git
11  eureka:
12    instance:
13      prefer-ip-address: true
14      ip-address: 127.0.0.1
15  client:
16    service-url:
17      defaultZone: http://localhost:8761/eureka/

```

验证

启动Eureka注册中心和配置中心

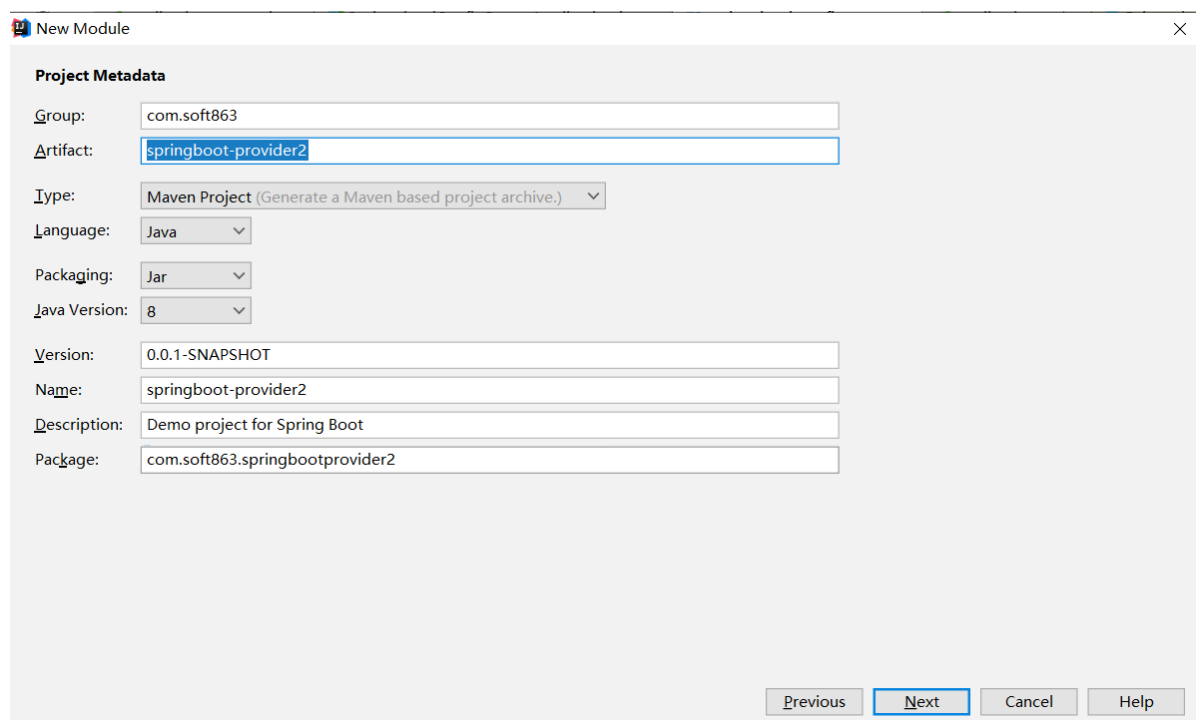
浏览器输入: <http://localhost:8763/user-dev.yml>

```
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8762/eureka/
    instance:
      ip-address: 127.0.0.1
      prefer-ip-address: true
  server:
    port: 2006
spring:
  application:
    name: user-server
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    password: root
    url: jdbc:mysql://localhost:3306/soft863db?useUnicode=true&characterEncoding=utf-8&useSSL=true&serverTimezone=UTC
    username: root
```

修改gitee上内容, 刷新上述地址查看是否变化

### 3、获取配置中心配置, 创建服务提供者

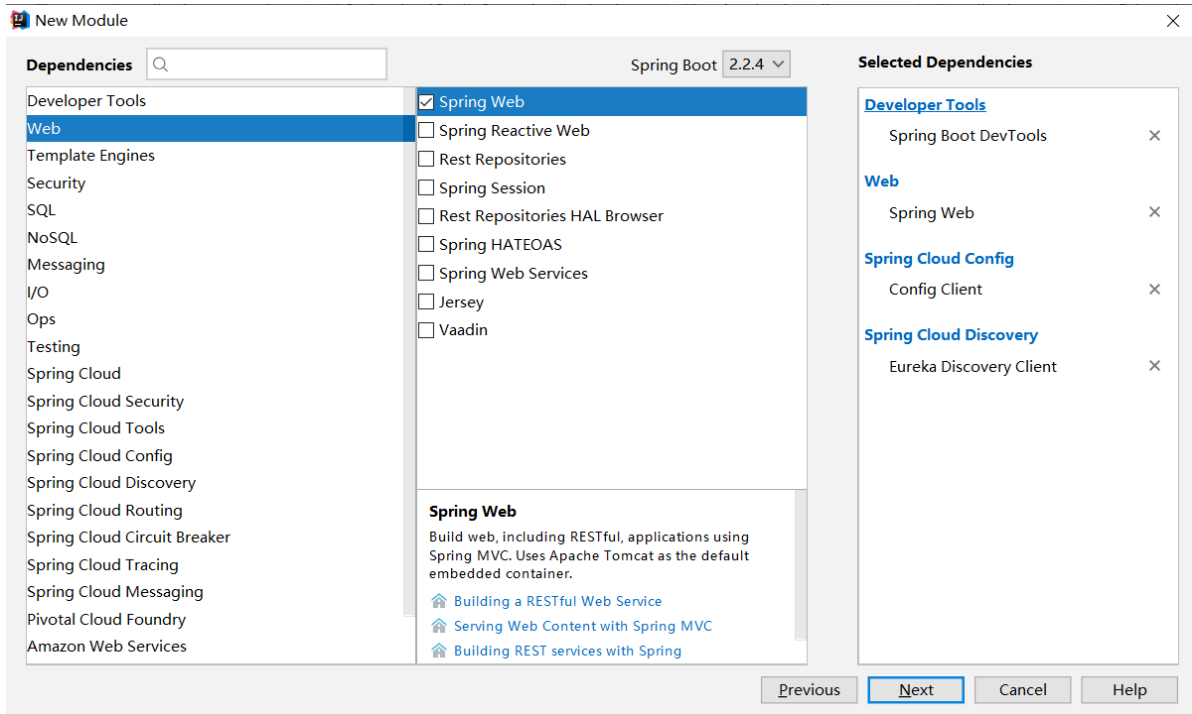
创建springboot-provider2工程



The screenshot shows the 'New Module' dialog box in an IDE. The 'Project Metadata' section is filled out with the following information:

- Group: com.soft863
- Artifact: springboot-provider2
- Type: Maven Project (Generate a Maven based project archive.)
- Language: Java
- Packaging: Jar
- Java Version: 8
- Version: 0.0.1-SNAPSHOT
- Name: springboot-provider2
- Description: Demo project for Spring Boot
- Package: com.soft863.springbootprovider2

At the bottom right, there are four buttons: 'Previous', 'Next' (which is highlighted with a blue border), 'Cancel', and 'Help'.



pom配置:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5      https://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <modelVersion>4.0.0</modelVersion>
7      <parent>
8          <groupId>org.springframework.boot</groupId>
9          <artifactId>spring-boot-starter-parent</artifactId>
10         <version>2.2.4.RELEASE</version>
11         <relativePath/> <!-- lookup parent from repository -->
12     </parent>
13     <groupId>com.soft863</groupId>
14     <artifactId>springcloud-provider2</artifactId>
15     <version>0.0.1-SNAPSHOT</version>
16     <name>springcloud-provider2</name>
17     <description>Demo project for Spring Boot</description>
18     <properties>
19         <java.version>1.8</java.version>
20         <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
21     </properties>
22     <dependencies>
23         <dependency>
24             <groupId>org.springframework.boot</groupId>
25             <artifactId>spring-boot-starter-web</artifactId>
26         </dependency>
27         <dependency>
28             <groupId>org.springframework.cloud</groupId>
29             <artifactId>spring-cloud-starter-config</artifactId>
30         </dependency>
31         <dependency>

```

```

32         <groupId>org.springframework.cloud</groupId>
33         <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
34     </dependency>
35     <dependency>
36         <groupId>org.springframework.cloud</groupId>
37         <artifactId>spring-cloud-starter-bootstrap</artifactId>
38     </dependency>
39     <dependency>
40         <groupId>org.springframework.boot</groupId>
41         <artifactId>spring-boot-devtools</artifactId>
42         <scope>runtime</scope>
43         <optional>true</optional>
44     </dependency>
45     <dependency>
46         <groupId>org.springframework.boot</groupId>
47         <artifactId>spring-boot-starter-test</artifactId>
48         <scope>test</scope>
49         <exclusions>
50             <exclusion>
51                 <groupId>org.junit.vintage</groupId>
52                 <artifactId>junit-vintage-engine</artifactId>
53             </exclusion>
54         </exclusions>
55     </dependency>
56 </dependencies>
57
58 <dependencyManagement>
59     <dependencies>
60         <dependency>
61             <groupId>org.springframework.cloud</groupId>
62             <artifactId>spring-cloud-dependencies</artifactId>
63             <version>${spring-cloud.version}</version>
64             <type>pom</type>
65             <scope>import</scope>
66         </dependency>
67     </dependencies>
68 </dependencyManagement>
69
70 <build>
71     <plugins>
72         <plugin>
73             <groupId>org.springframework.boot</groupId>
74             <artifactId>spring-boot-maven-plugin</artifactId>
75         </plugin>
76     </plugins>
77 </build>
78
79 </project>
80

```

bootstrap.yml文件也是Spring Boot的默认配置文件，而且其加载的时间相比于application.yml更早。application.yml和bootstrap.yml虽然都是Spring Boot的默认配置文件，但是定位却不相同。

bootstrap.yml 可以理解成系统级别的一些参数配置，这些参数一般是不会变动的。

application.yml 可以用来定义应用级别的参数。

如果搭配 spring cloud config 使用，application.yml 里面定义的文件可以实现动态替换。总结就是，bootstrap.yml文件相当于项目启动时的引导文件，内容相对固定。application.yml文件是微服务的一些常规配置参数

yml文件配置

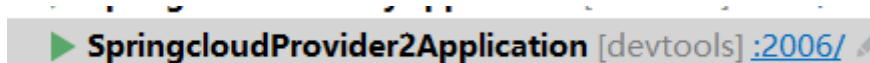
删除\src\main\resources\application.yml 文件（因为该文件从配置中心获取）,创建  
\src\main\resources\bootstrap.yml 配置文件

bootstrap.yml 内容：

```
1  spring:
2    cloud:
3      config:
4        # 与远程仓库中的配置文件的application保持一致
5        name: user
6        # 远程仓库中的配置文件的profile保持一致
7        profile: dev
8        # 远程仓库中的配置文件的profile保持一致
9        label: master
10     discovery:
11       enabled: true
12       service-id: config-server
13  eureka:
14    instance:
15      prefer-ip-address: true
16      ip-address: 127.0.0.1
17    client:
18      service-url:
19        defaultZone: http://localhost:8761/eureka/
```

测试：

启动注册中心 eureka-server 、配置中心 config-server 、 provider2工程



浏览器输入：<http://localhost:8762/>

USER-SERVER	n/a (1)	(1)	UP (1) - wcj-pc:user-server:2006
-------------	---------	-----	----------------------------------

```
1  curl -X POST http://localhost:8089/actuator/refresh
```

## 三、Spring Cloud Bus服务总线

如果想在重启微服务的情况下更新配置该如何实现呢？可以使用Spring Cloud Bus来实现配置的自动更新。

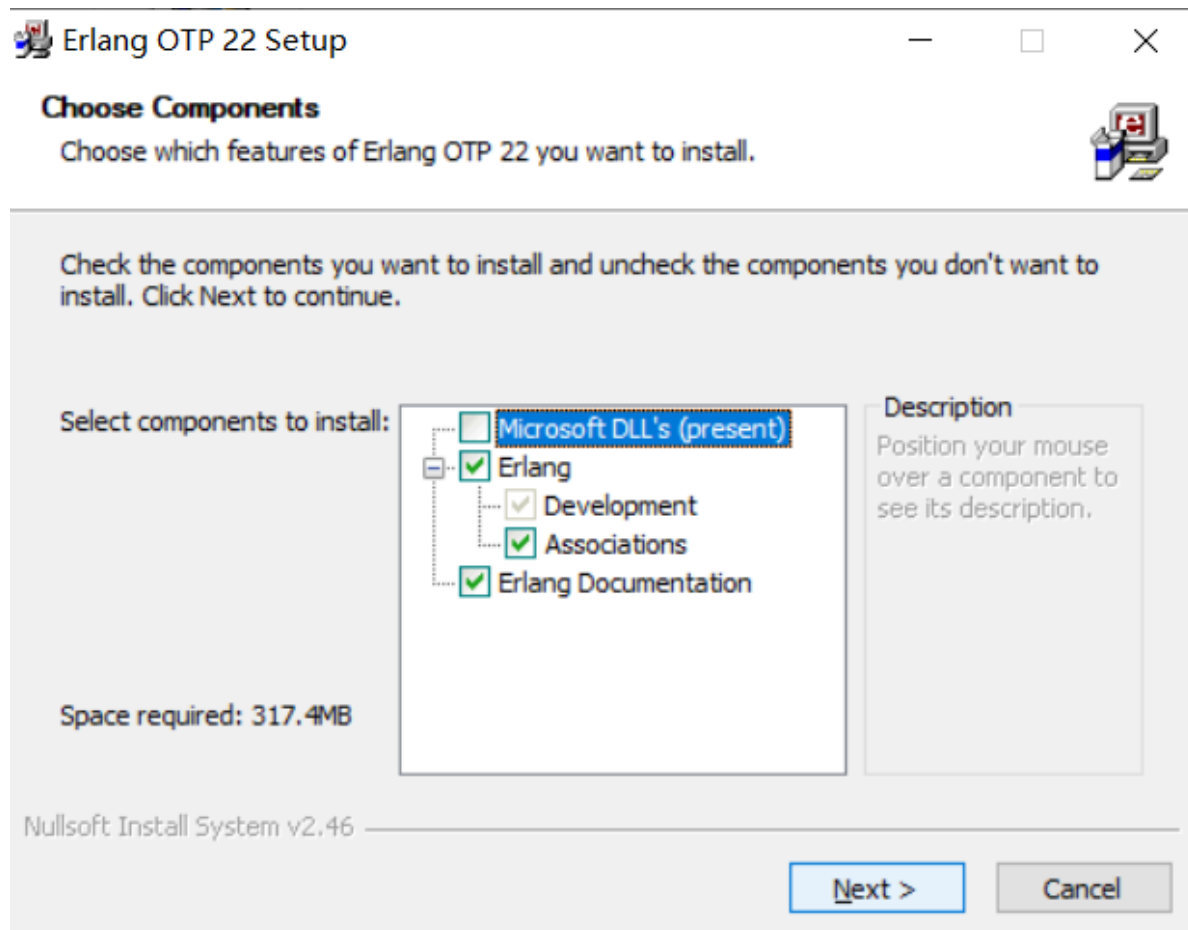
需要注意的是Spring Cloud Bus底层是基于RabbitMQ实现的，默认使用本地的消息队列服务，所以需要提前 启动本地RabbitMQ服务（安装RabbitMQ以后才有）

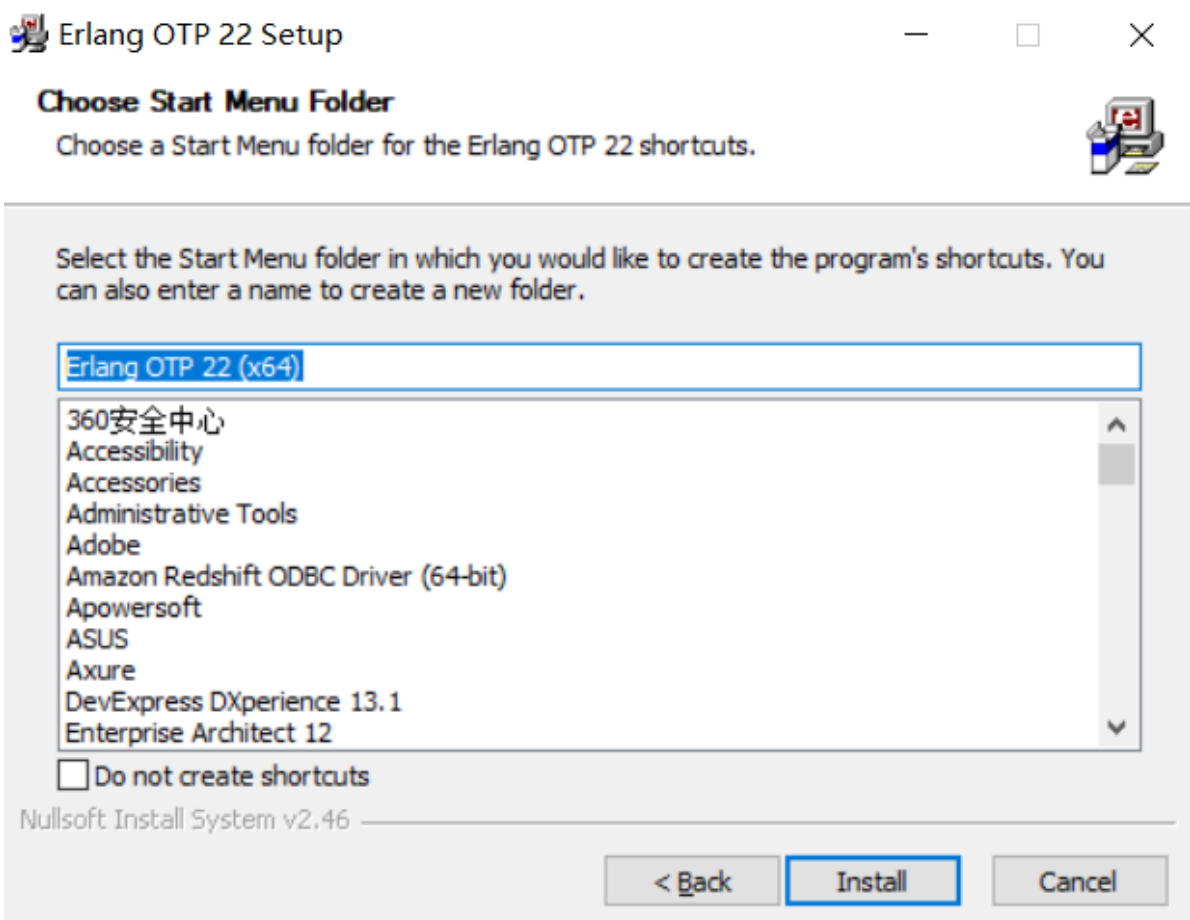
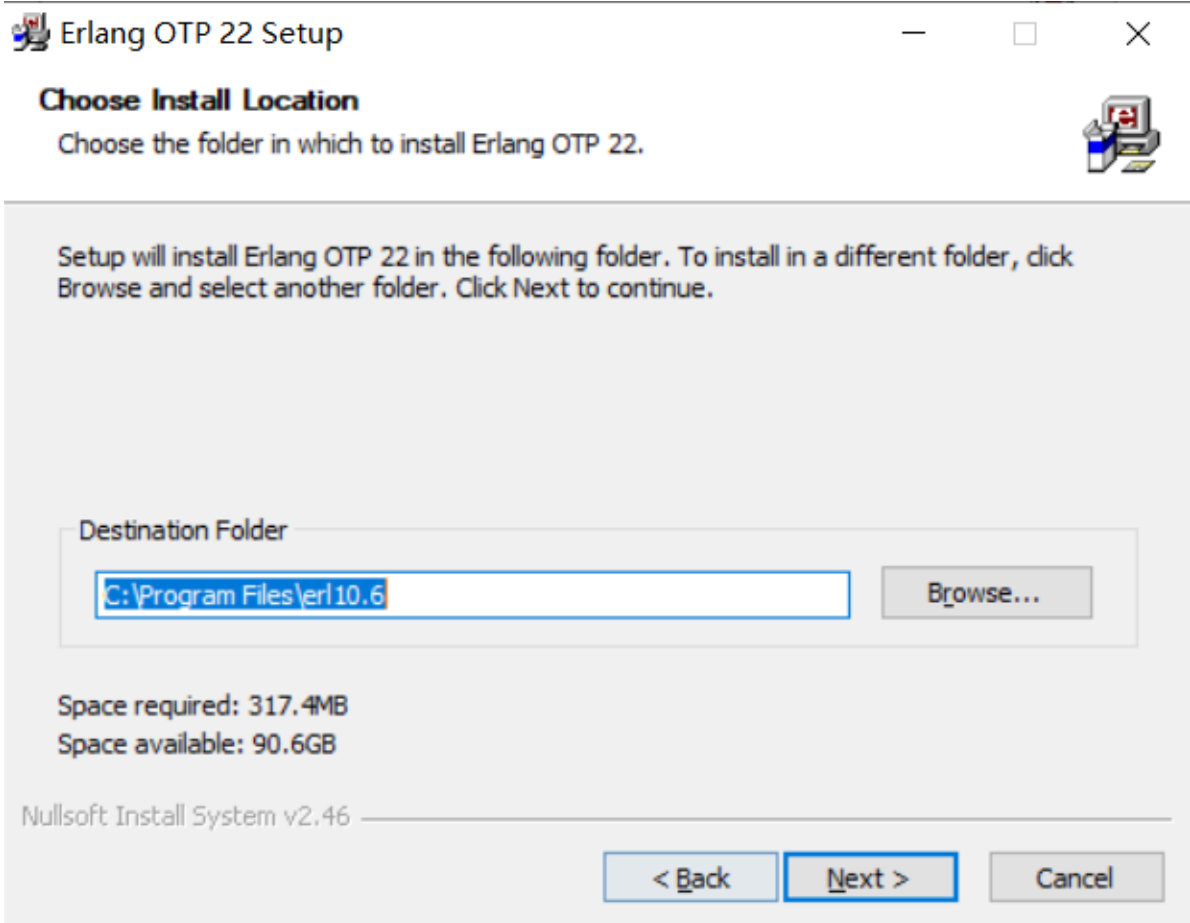
Spring Cloud Bus是用轻量的消息代理将分布式的节点连接起来，可以用于广播配置文件的更改或者服务的监控管理。也就是消息总线可以为微服务做监控，也可以实现应用程序之间相互通信。Spring Cloud Bus可选的消息代理有RabbitMQ和Kafka。使用了Bus之后，SpringCloudBus可以不用重启provider2就可以将码云上最新配置及时更新到本地

## 1、安装RabbitMQ

### 1.1安装erlang环境

下载地址：<https://www.erlang.org/downloads>





环境变量中添加:

ERLANG\_HOME

C:\Program Files\erl10.6

变量名(N):

变量值(V):

Path中添加: %ERLANG\_HOME%\bin

环境变量

编辑环境变量

新建(N) 编辑(E) 浏览(B)... 删除(D) 上移(U) 下移(Q) 编辑文本(I)...

确定 取消

确定 取消

cmd中输入erl:

```
C:\Users\wangchaojie>erl
Eshell V10.6 (abort with ^G)
```

## 1.2、安装RabbitMQ



### Choose Components

Choose which features of RabbitMQ Server 3.8.2 you want to install.



Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- ☒ RabbitMQ Server (required)
- ☒ RabbitMQ Service
- ☒ Start Menu

#### Description

Position your mouse over a component to see its description.

Space required: 20.1MB

Nullsoft Install System v2.51-1+b1

Next >

Cancel

### Choose Install Location

Choose the folder in which to install RabbitMQ Server 3.8.2.



Setup will install RabbitMQ Server 3.8.2 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Program Files\RabbitMQ Server

Browse...

Space required: 20.1MB

Space available: 90.3GB

Nullsoft Install System v2.51-1+b1

< Back

Install

Cancel

cd到安装路径的sbin下面

cd C:\Program Files\RabbitMQ Server\rabbitmq\_server-3.8.2\sbin

执行,启动可视化界面

```
1 | rabbitmq-plugins enable rabbitmq_management
```

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.18362.592]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\wangchaojie>cd C:\Program Files\RabbitMQ Server\rabbitmq_server-3.8.2\sbin
C:\Program Files\RabbitMQ Server\rabbitmq_server-3.8.2\sbin>rabbitmq-plugins enable rabbitmq_management
Enabling plugins on node rabbit@wcj-pc:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@wcj-pc...
The following plugins have been enabled:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
started 3 plugins.
```

输入: `rabbitmqctl status`

查看启动状态

```
C:\Program Files\RabbitMQ Server\rabbitmq_server-3.8.2\sbin>rabbitmqctl status
Status of node rabbit@wcj-pc ...
[1mRuntime[0m

OS PID: 12136
OS: Windows
Uptime (seconds): 175
RabbitMQ version: 3.8.2
Node name: rabbit@wcj-pc
Erlang configuration: Erlang/OTP 22 [erts-10.6] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:64]
Erlang processes: 448 used, 1048576 limit
Scheduler run queue: 0
Cluster heartbeat timeout (net_ticktime): 60

[1mPlugins[0m

Enabled plugin file: C:/Users/wangchaojie/AppData/Roaming/RabbitMQ/enabled_plugins
Enabled plugins:

* rabbitmq_management
* rabbitmq_web_dispatch
* rabbitmq_management_agent
```

浏览器中访问: 访问<http://localhost:15672>



Username:

\*

Password:

\*

Login

默认用户名和密码都是guest

RabbitMQ

3.8.2

Erlang 22.2

Refreshed 2020-01-30 16:57:32

Refresh every 5 seconds

Virtual host

All

Cluster **rabbit@wcj-pc**

User **guest** [Log out](#)

Overview

Connections

Channels

Exchanges

Queues

Admin

Overview

Totals

Queued messages

last minute

Currently idle

Message rates

last minute

Currently idle

Global counts

Connections: 0

Channels: 0

Exchanges: 7

Queues: 0

Consumers: 0

Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats
rabbit@wcj-pc	0 65536 available	0 38893 available	447 1048576 available	102MiB 6.4GiB high watermark	90GiB 48MiB low watermark	5m 54s	basic disc 1 rss	This node All nodes

Churn statistics

Ports and contexts

Export definitions

## 2、创建Bus服务端

创建新module,命名为: springcloud-config-bus-server

New Module

Project Metadata

Group:

com.soft863

Artifact:

springcloud-config-bug-server

Type:

Maven Project (Generate a Maven based project archive.)

Language:

Java

Packaging:

Jar

Java Version:

8

Version:

0.0.1-SNAPSHOT

Name:

springcloud-config-bug-server

Description:

Demo project for Spring Boot

Package:

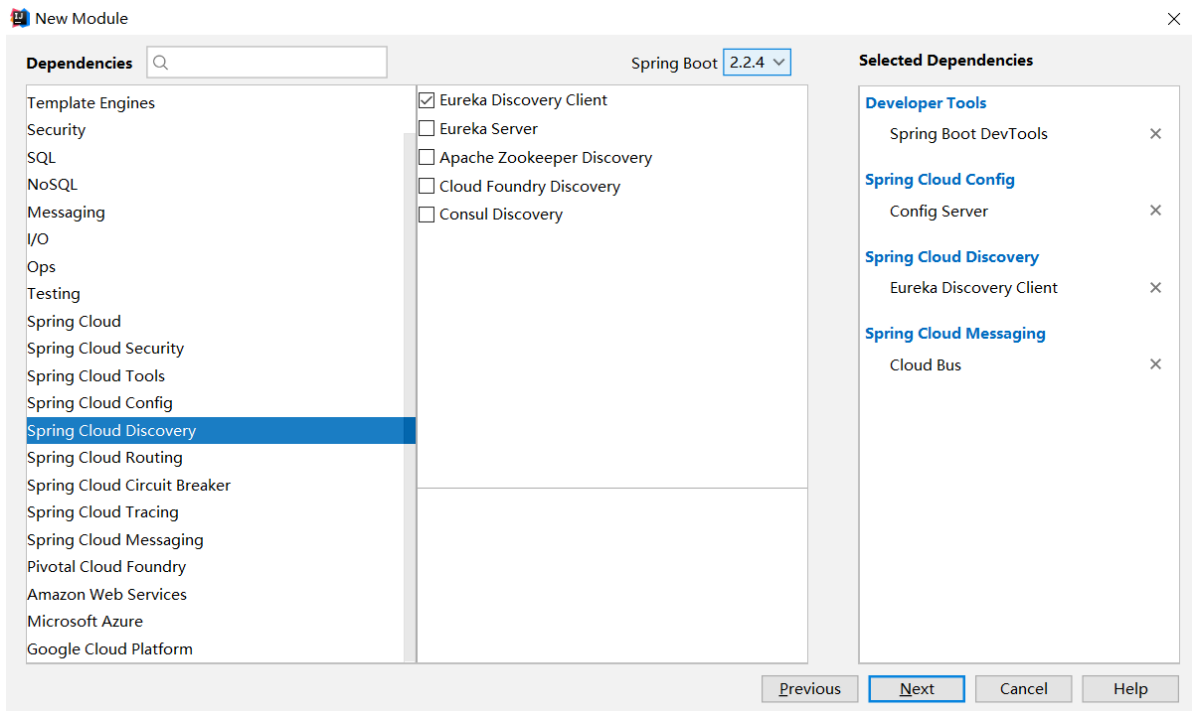
com.soft863.springcloudconfigbugserver

Previous

Next

Cancel

Help



添加pom

```
1 <dependency>
2     <groupId>org.springframework.cloud</groupId>
3     <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
4 </dependency>
```

整个pom文件如下:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5     https://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7   <parent>
8     <groupId>org.springframework.boot</groupId>
9     <artifactId>spring-boot-starter-parent</artifactId>
10    <version>2.2.4.RELEASE</version>
11    <relativePath/> <!-- lookup parent from repository -->
12  </parent>
13  <groupId>com.soft863</groupId>
14  <artifactId>springcloud-config-bug-server</artifactId>
15  <version>0.0.1-SNAPSHOT</version>
16  <name>springcloud-config-bug-server</name>
17  <description>Demo project for Spring Boot</description>
18
19  <properties>
20    <java.version>1.8</java.version>
21    <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
22  </properties>
23
24  <dependencies>
25    <dependency>
26      <groupId>org.springframework.cloud</groupId>
```

```

25         <artifactId>spring-cloud-bus</artifactId>
26     </dependency>
27     <dependency>
28         <groupId>org.springframework.cloud</groupId>
29         <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
30     </dependency>
31     <dependency>
32         <groupId>org.springframework.cloud</groupId>
33         <artifactId>spring-cloud-config-server</artifactId>
34     </dependency>
35     <dependency>
36         <groupId>org.springframework.cloud</groupId>
37         <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
38     </dependency>
39
40     <dependency>
41         <groupId>org.springframework.boot</groupId>
42         <artifactId>spring-boot-devtools</artifactId>
43         <scope>runtime</scope>
44         <optional>true</optional>
45     </dependency>
46     <dependency>
47         <groupId>org.springframework.boot</groupId>
48         <artifactId>spring-boot-starter-test</artifactId>
49         <scope>test</scope>
50         <exclusions>
51             <exclusion>
52                 <groupId>org.junit.vintage</groupId>
53                 <artifactId>junit-vintage-engine</artifactId>
54             </exclusion>
55         </exclusions>
56     </dependency>
57 </dependencies>
58
59 <dependencyManagement>
60     <dependencies>
61         <dependency>
62             <groupId>org.springframework.cloud</groupId>
63             <artifactId>spring-cloud-dependencies</artifactId>
64             <version>${spring-cloud.version}</version>
65             <type>pom</type>
66             <scope>import</scope>
67         </dependency>
68     </dependencies>
69 </dependencyManagement>
70
71 <build>
72     <plugins>
73         <plugin>
74             <groupId>org.springframework.boot</groupId>
75             <artifactId>spring-boot-maven-plugin</artifactId>
76         </plugin>
77     </plugins>
78 </build>

```

```
79
80 </project>
81
```

application.yml配置文件添加内容

```
1  server:
2    port: 8763
3  spring:
4    application:
5      name: configserver
6    cloud:
7      config:
8        server:
9          git:
10             uri: https://gitee.com/mochiwcj/springcloud-config.git
11      #默认, 可以不配置
12    rabbitmq:
13      host: 127.0.0.1
14      port: 5672
15      username: guest
16      password: guest
17    eureka:
18      instance:
19        prefer-ip-address: true
20        ip-address: 127.0.0.1
21      client:
22        service-url:
23          defaultZone: http://localhost:8761/eureka/
24    management:
25      endpoints:
26        web:
27          exposure:
28            include: "*"

```

启动类上添加注解@EnableConfigServer, @EnableEurekaClient

### 3、添加bus客户端

创建Module,命名为: springcloud-config-bus-client

New Module

### Project Metadata

Group:

Artifact:

Type:

Language:

Packaging:

Java Version:

Version:

Name:

Description:

Package:

Previous Next Cancel Help

New Module

### Dependencies

Spring Boot 2.2.4

Developer Tools

Web

Template Engines

Security

SQL

NoSQL

Messaging

I/O

Ops

Testing

Spring Cloud

Spring Cloud Security

Spring Cloud Tools

Spring Cloud Config

Spring Cloud Discovery

Spring Cloud Routing

Spring Cloud Circuit Breaker

Spring Cloud Tracing

Spring Cloud Messaging

Pivotal Cloud Foundry

Amazon Web Services

☒ Config Client

☐ Config Server

☐ Vault Configuration

☐ Apache Zookeeper Configuration

☐ Consul Configuration

**Config Client**

Client that connects to a Spring Cloud Config Server to fetch the application's configuration.

### Selected Dependencies

**Developer Tools**

Spring Boot DevTools

**Web**

Spring Web

**Spring Cloud Config**

Config Client

**Spring Cloud Discovery**

Eureka Discovery Client

**Spring Cloud Messaging**

Cloud Bus

Previous Next Cancel Help

pom文件中添加:

```

1      <dependency>
2          <groupId>org.springframework.cloud</groupId>
3          <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
4      </dependency>
5      <dependency>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-actuator</artifactId>
8      </dependency>
9  
```

完整pom

```

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

```

```
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>2.2.4.RELEASE</version>
9         <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>com.soft863</groupId>
12    <artifactId>springcloud-config-bus-client</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <name>springcloud-config-bus-client</name>
15    <description>Demo project for Spring Boot</description>
16
17    <properties>
18        <java.version>1.8</java.version>
19        <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
20    </properties>
21
22    <dependencies>
23        <dependency>
24            <groupId>org.springframework.boot</groupId>
25            <artifactId>spring-boot-starter-web</artifactId>
26        </dependency>
27        <dependency>
28            <groupId>org.springframework.cloud</groupId>
29            <artifactId>spring-cloud-bus</artifactId>
30        </dependency>
31        <dependency>
32            <groupId>org.springframework.cloud</groupId>
33            <artifactId>spring-cloud-starter-config</artifactId>
34        </dependency>
35        <dependency>
36            <groupId>org.springframework.cloud</groupId>
37            <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
38        </dependency>
39
40        <dependency>
41            <groupId>org.springframework.boot</groupId>
42            <artifactId>spring-boot-devtools</artifactId>
43            <scope>runtime</scope>
44            <optional>true</optional>
45        </dependency>
46        <dependency>
47            <groupId>org.springframework.boot</groupId>
48            <artifactId>spring-boot-starter-test</artifactId>
49            <scope>test</scope>
50            <exclusions>
51                <exclusion>
52                    <groupId>org.junit.vintage</groupId>
53                    <artifactId>junit-vintage-engine</artifactId>
```



```

54         </exclusion>
55     </exclusions>
56 </dependency>
57 <dependency>
58     <groupId>org.springframework.cloud</groupId>
59     <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
60 </dependency>
61 <dependency>
62     <groupId>org.springframework.boot</groupId>
63     <artifactId>spring-boot-starter-actuator</artifactId>
64 </dependency>
65 </dependencies>
66
67 <dependencyManagement>
68     <dependencies>
69         <dependency>
70             <groupId>org.springframework.cloud</groupId>
71             <artifactId>spring-cloud-dependencies</artifactId>
72             <version>${spring-cloud.version}</version>
73             <type>pom</type>
74             <scope>import</scope>
75         </dependency>
76     </dependencies>
77 </dependencyManagement>
78
79 <build>
80     <plugins>
81         <plugin>
82             <groupId>org.springframework.boot</groupId>
83             <artifactId>spring-boot-maven-plugin</artifactId>
84         </plugin>
85     </plugins>
86 </build>
87
88 </project>
89
90

```

删除项目的application配置文件，新增bootstrap.yml如下

```

1  spring:
2    application:
3      name: springcloud-config-bus-client
4    cloud:
5      config:
6        # 与远程仓库中的配置文件的application保持一致
7        name: user
8        # 远程仓库中的配置文件的profile保持一致
9        profile: dev
10       # 远程仓库中的配置文件的profile保持一致
11       label: master
12     discovery:
13       # 使用配置中心
14       enabled: true
15       service-id: configserver

```

```

16     bus:
17         trace:
18             enabled: true
19     #默认可以不写
20     rabbitmq:
21         host: localhost
22         port: 5672
23         username: guest
24         password: guest
25     eureka:
26         instance:
27             prefer-ip-address: true
28             ip-address: 127.0.0.1
29         client:
30             service-url:
31                 defaultZone: http://localhost:8762/eureka/
32

```

### 创建UserController

```

1  package com.soft863.controller;
2
3  import org.springframework.beans.factory.annotation.Value;
4  import org.springframework.cloud.context.config.annotation.RefreshScope;
5  import org.springframework.stereotype.Controller;
6  import org.springframework.web.bind.annotation.RequestMapping;
7  import org.springframework.web.bind.annotation.ResponseBody;
8
9  @Controller
10 @RequestMapping("/user")
11 @RefreshScope
12 public class UserController {
13
14     @Value("${soft863.test}")
15     private String name;
16
17     @RequestMapping("/test")
18     @ResponseBody
19     public String getResult() {
20         return "配置文件测试内容: " + name;
21     }
22 }
23
24

```

### 启动类上添加注解

```

1  @EnableEurekaClient
2  @ComponentScan("com.soft863")
3

```

## 测试

启动eureka-server、config-bus-server、config-bus-client

访问地址<http://127.0.0.1:2006/user/test>

## 配置文件测试内容: wcj

修改码云上的配置文件: <https://gitee.com/superjean/springcloud-config/blob/master/user-dev.yml>

将soft863.test内容修改为wangchaojie,

通过postman的post执行<http://127.0.0.1:8764/actuator/bus-refresh>

或在cmd中执行

```
1 | curl -X POST http://127.0.0.1:8763/actuator/bus-refresh
```

```
C:\Users\wangchaojie>curl -X POST http://127.0.0.1:8763/actuator/bus-refresh
```

再次访问地址: <http://127.0.0.1:2006/user/test>

## 配置文件测试内容: wangchaojie

说明: 1、Postman或者RESTClient是一个可以模拟浏览器发送各种请求 (POST、GET、PUT、DELETE 等) 的工具 2、请求地址<http://127.0.0.1:8764/actuator/bus-refresh>中 /actuator是固定的, /bus-refresh对应的是配置 中心config-server中的application.yml文件的配置项include的内容 3、请求<http://127.0.0.1:8764/actuator/bus-refresh>地址的作用是访问配置中心的消息总线服务, 消息总线 服务接收到请求后会向消息队列中发送消息, 各个微服务会监听消息队列。当微服务接收到队列中的消息后, 会重新从配置中心获取最新的配置信息