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

```
<?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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.soft863</groupId>
    <artifactId>springcloud-gateway</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springcloud-gateway</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
        <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-gateway</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-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </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>

```

yml配置

创建配置文件application-dev1.yml，添加内容

```
server:
  port: 2005

spring:
  application:
    name: gateway-service
  cloud:        # spring cloud gateway 路由配置方式
    gateway:
      routes:
      - id: USER-SERVICE            #网关路由到用户服务user-service,ID可以随意写
        uri: http://localhost:2003
        #路由断言，可配置映射路径
        predicates:
        - Path=/login/**
```

```yaml



eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    prefer-ip-address: true
```

创建application.yml配置文件，添加如下内容

```yaml
spring:
  profiles:
    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：

```java
package com.soft863.controller;

import com.soft863.entity.User;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/login")
public class LoginController {

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

    @GetMapping("/test2")
    public String log() {
        return "hello user";
    }

    @RequestMapping(value = "/getUser", method = RequestMethod.GET)
    public String getUser(@RequestParam String name) {

        return name + "恭喜您，登录成功， " + "我来自于端口:" + port;
    }

    @RequestMapping(value = "/getUserByID2", method = RequestMethod.GET)
    public User login(@RequestParam Integer id) {
        User user = new User();
        user.setId(1);
        user.setUsername("admin");
        user.setPassword("123456");
```

```
        return user;
    }

    @RequestMapping(value = "/getUserByID2/{id}", method = RequestMethod.GET)
    public User login1(@PathVariable int id) {
        User user = new User();
        user.setId(1);
        user.setUsername("admin");
        user.setPassword("123456");
        return user;
    }

    @GetMapping("/getUser2")
    public String getUser(){
        return "执行成功";
    }
}

```

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

```
server:
  port: 2005

spring:
  application:
    name: gateway-service
  cloud:        # spring cloud gateway 路由配置方式
    gateway:
      routes:
      - id: USER-SERVICE        #网关路由到用户服务user-service,ID可以随意写
        uri: lb://USER-SERVICE   # 代理的服务地址；lb表示从eureka中获取具体服务
        #路由断言，可配置映射路径
        predicates:
        - Path=/login/**


eureka:
  client:
    service-url:
      defaultZone: http://localhost:8762/eureka/
```

```
  instance:
    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：

```
server:
  port: 2005

spring:
  application:
    name: gateway-service
  cloud:        # spring cloud gateway 路由配置方式
    gateway:
      routes:
      - id: USER-SERVICE        #网关路由到用户服务user-service,ID可以随意写
        uri: lb://USER-SERVICE   # 代理的服务地址；lb表示从eureka中获取具体服务
        #路由断言，可配置映射路径
        predicates:
        - Path=/**
        filters:
        - PrefixPath=/login


eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    prefer-ip-address: true


```

修改application.yml配置，指向dev3

测试：

重启Gateway工程

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

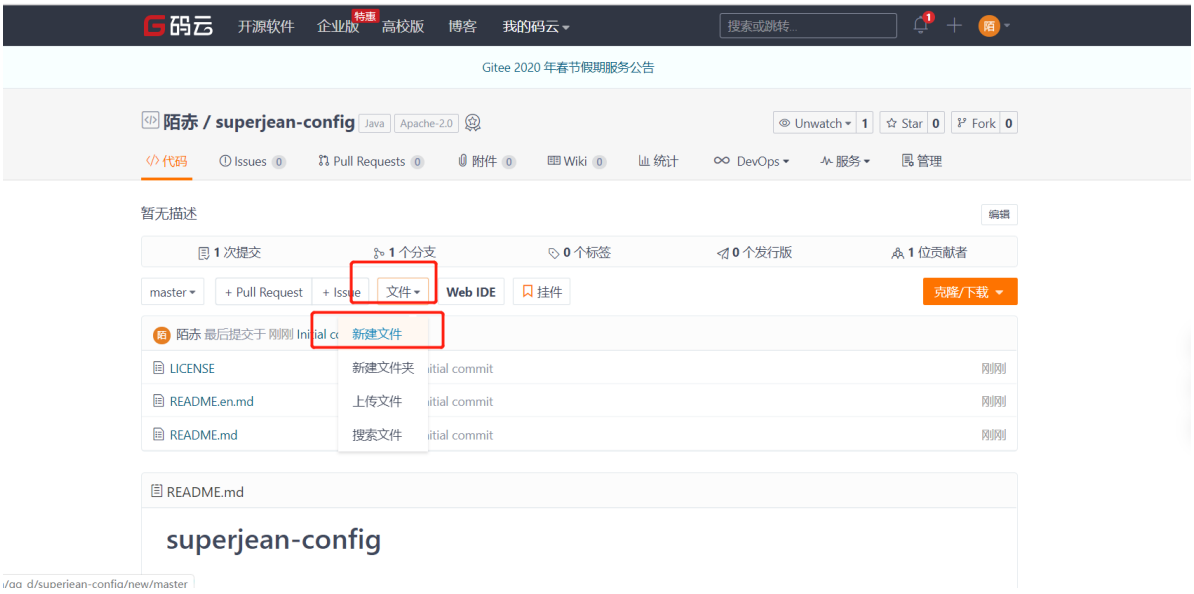
http://localhost:2005/test2 （可用）

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）。



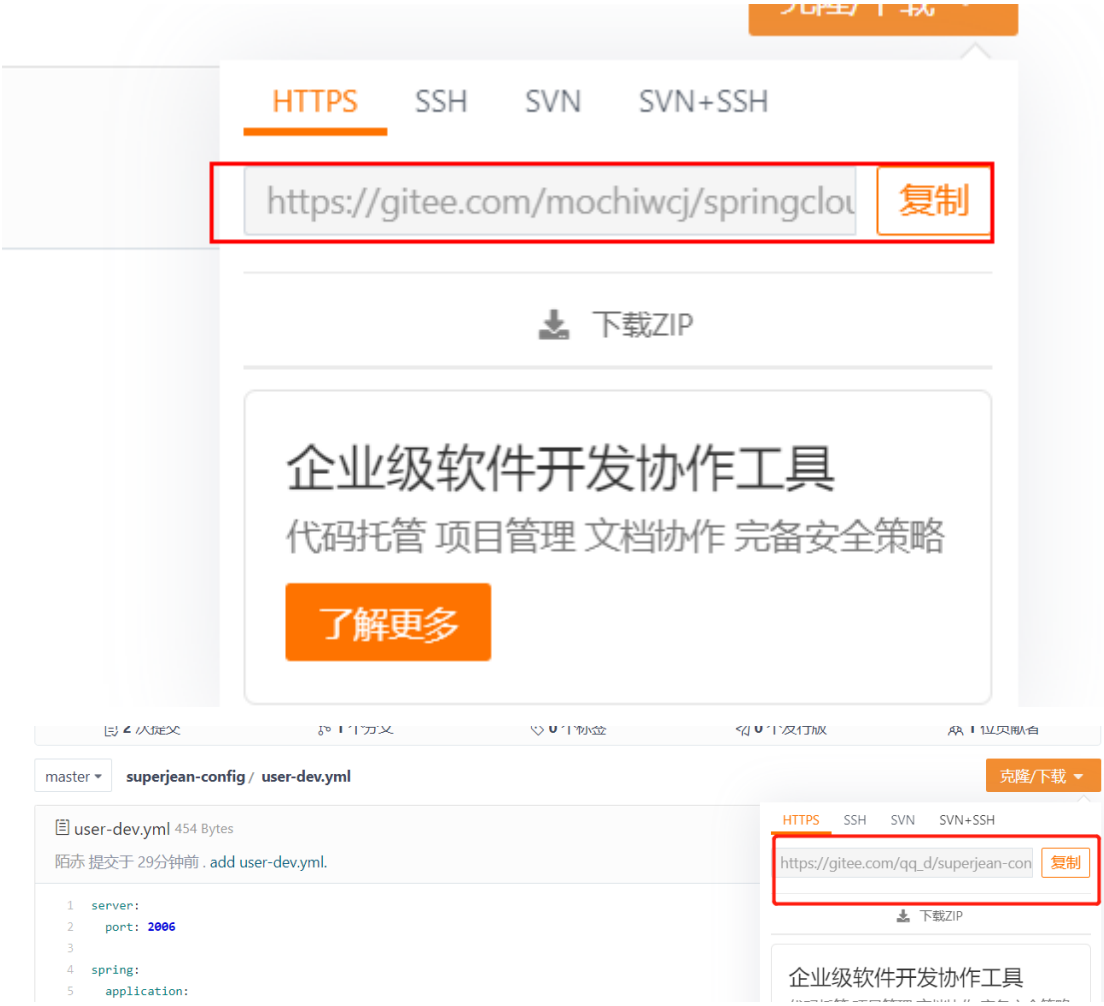
命名：user-dev.yml

添加如下代码：

```yaml
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/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
```

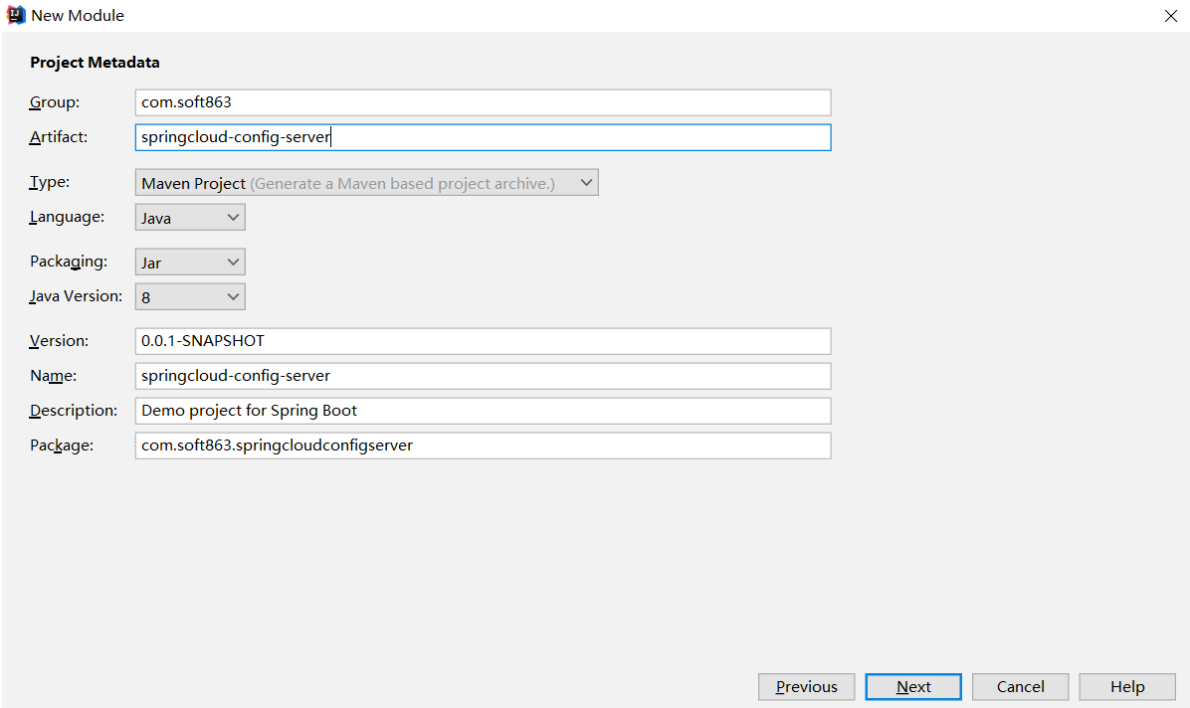
点击提交

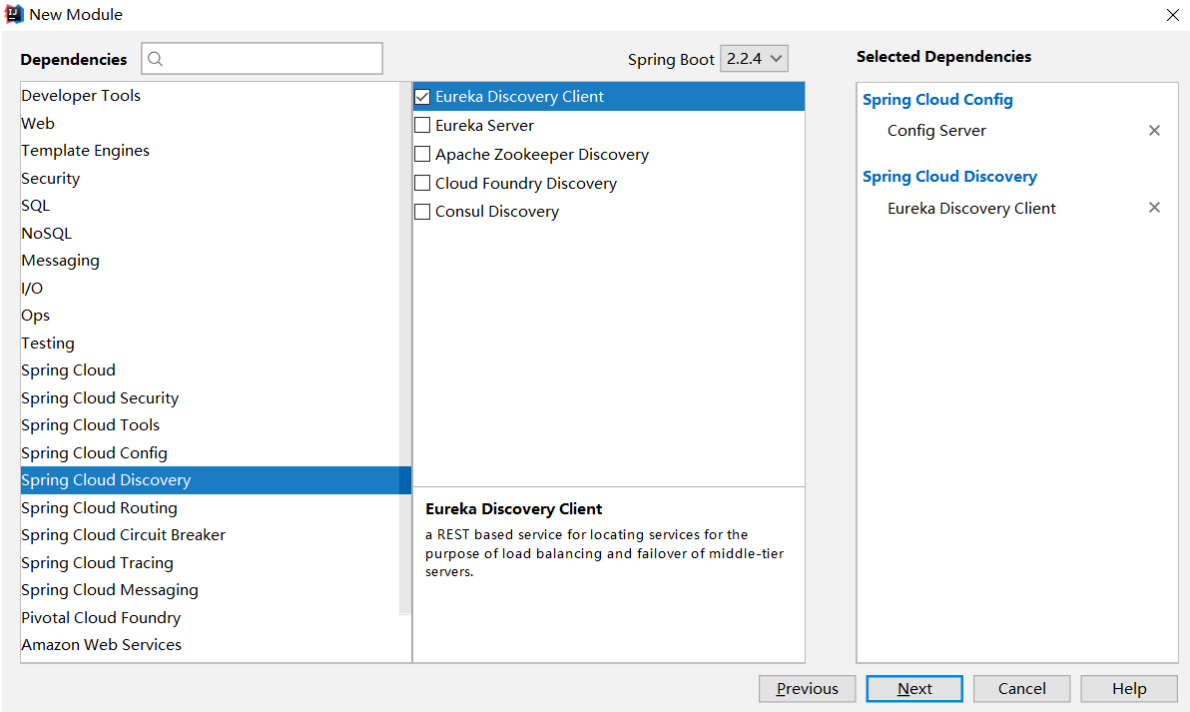
复制地址：



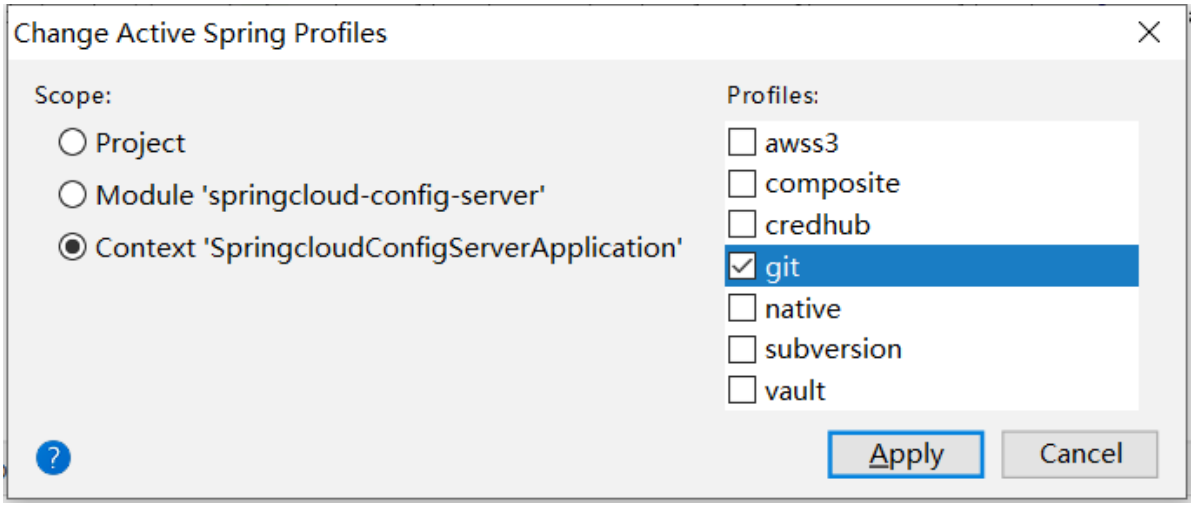
2、创建配置中心服务端

创建工程springcloud-config-server





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



编写配置文件

```yaml
server:
  port: 8763
spring:
  application:
    name: config-server
  cloud:
    config:
      server:
        git:
          uri: https://gitee.com/mochiwcj/springcloud-config.git
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 127.0.0.1
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

验证

启动Eureka注册中心和配置中心

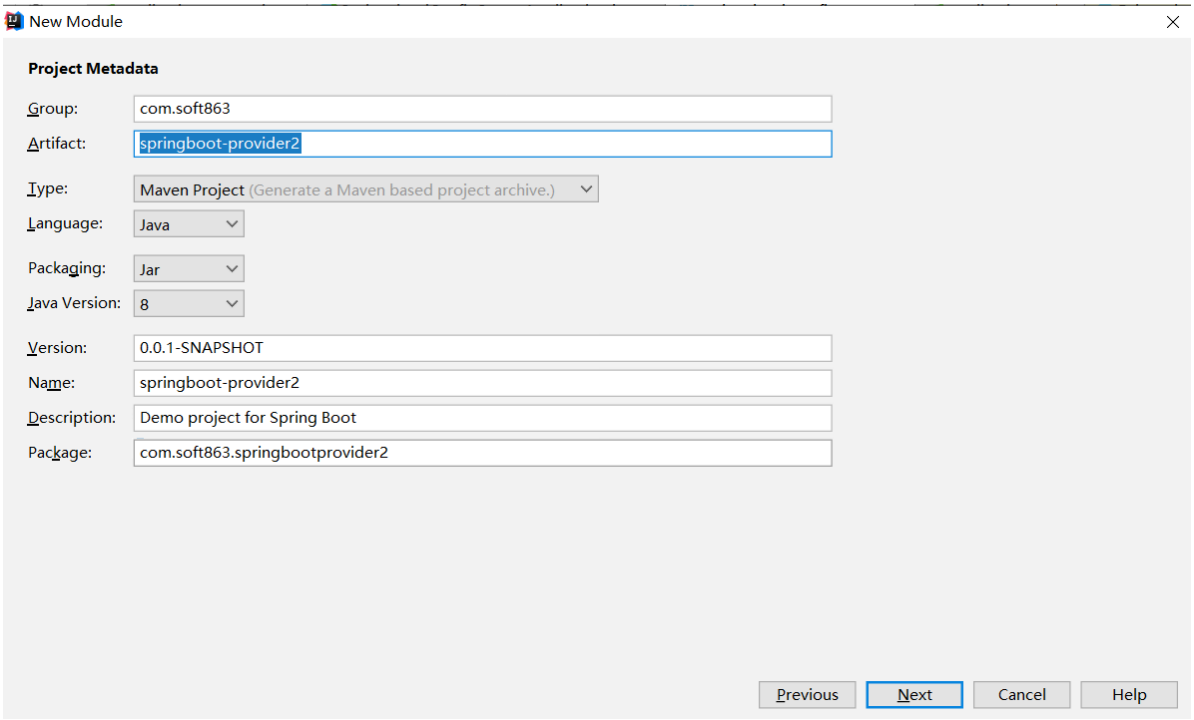
浏览器输入：http://localhost:8763/user-dev.yml

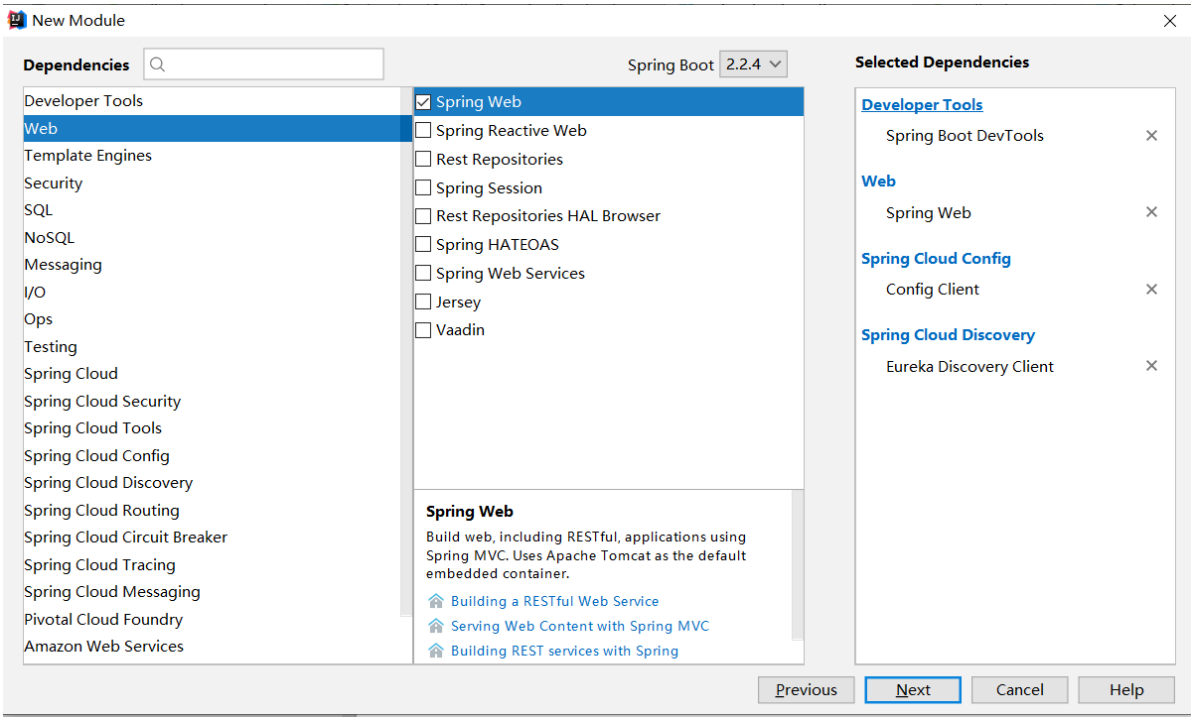
```
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8762/eureka/
  instance:
    ip-address: 127.0.0.1
    prefer-ip-address: true
server:
  port: 2006
spring:
  application:
    name: user-server
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    password: root
    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





pom配置：

```
<?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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.soft863</groupId>
    <artifactId>springcloud-provider2</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springcloud-provider2</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
        <spring-cloud.version>Hoxton.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.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bootstrap</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </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>

```

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

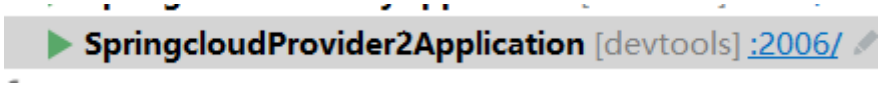
删除\src\main\resources\application.yml 文件（因为该文件从配置中心获取）,创建 \src\main\resources\bootstrap.yml 配置文件

bootstrap.yml 内容：

```yml
spring:
  cloud:
    config:
      # 与远程仓库中的配置文件的application保持一致
      name: user
      # 远程仓库中的配置文件的profile保持一致
      profile: dev
      # 远程仓库中的配置文件的profile保持一致
      label: master
      discovery:
        enabled: true
        service-id: config-server
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 127.0.0.1
  client:
    service-url:
      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

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

三、Spring Cloud Bus服务总线

如果想在不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。

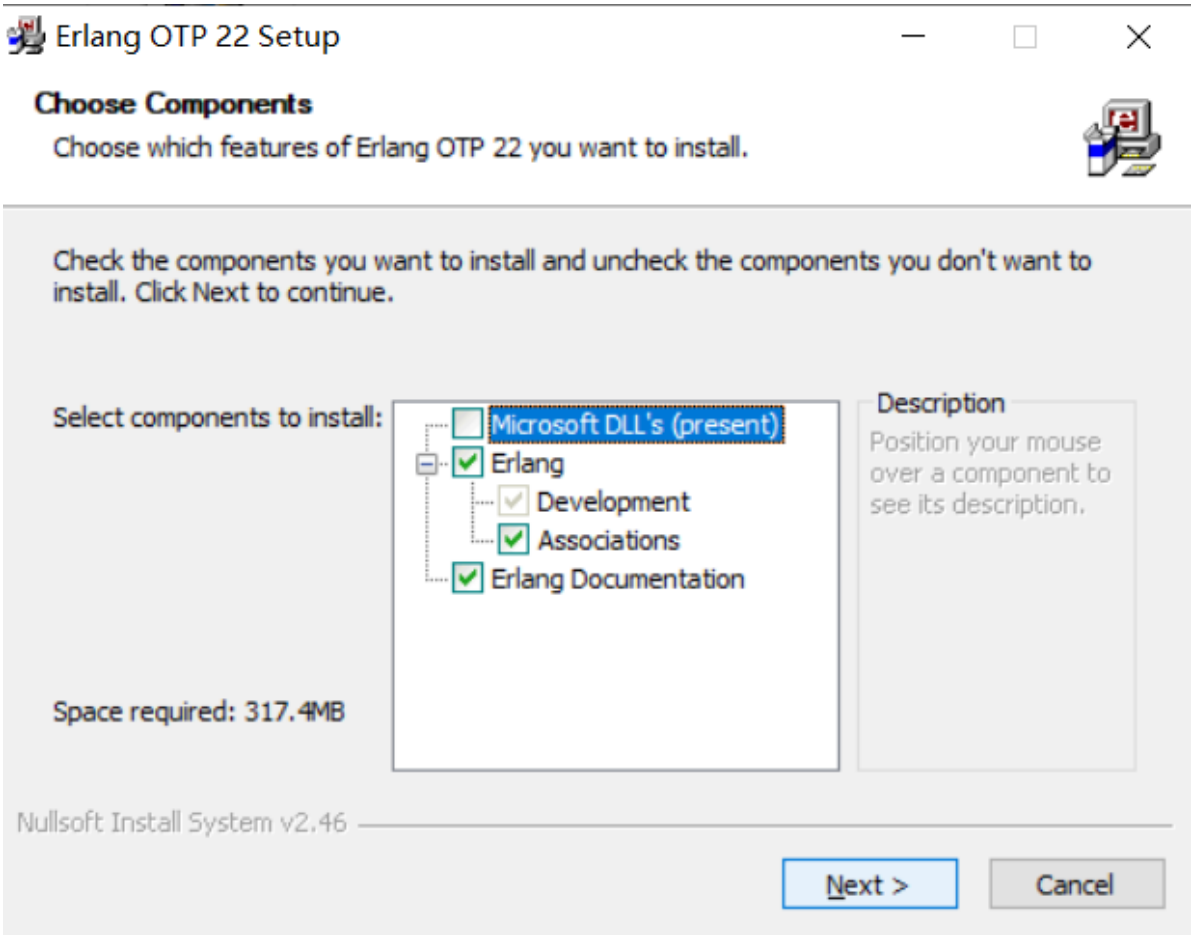
需要注意的是Spring Cloud Bus底层是基于RabbitMQ实现的，默认使用本地的消息队列服务，所以需要提前 启动本地RabbitMQ服务（安装RabbitMQ以后才有）

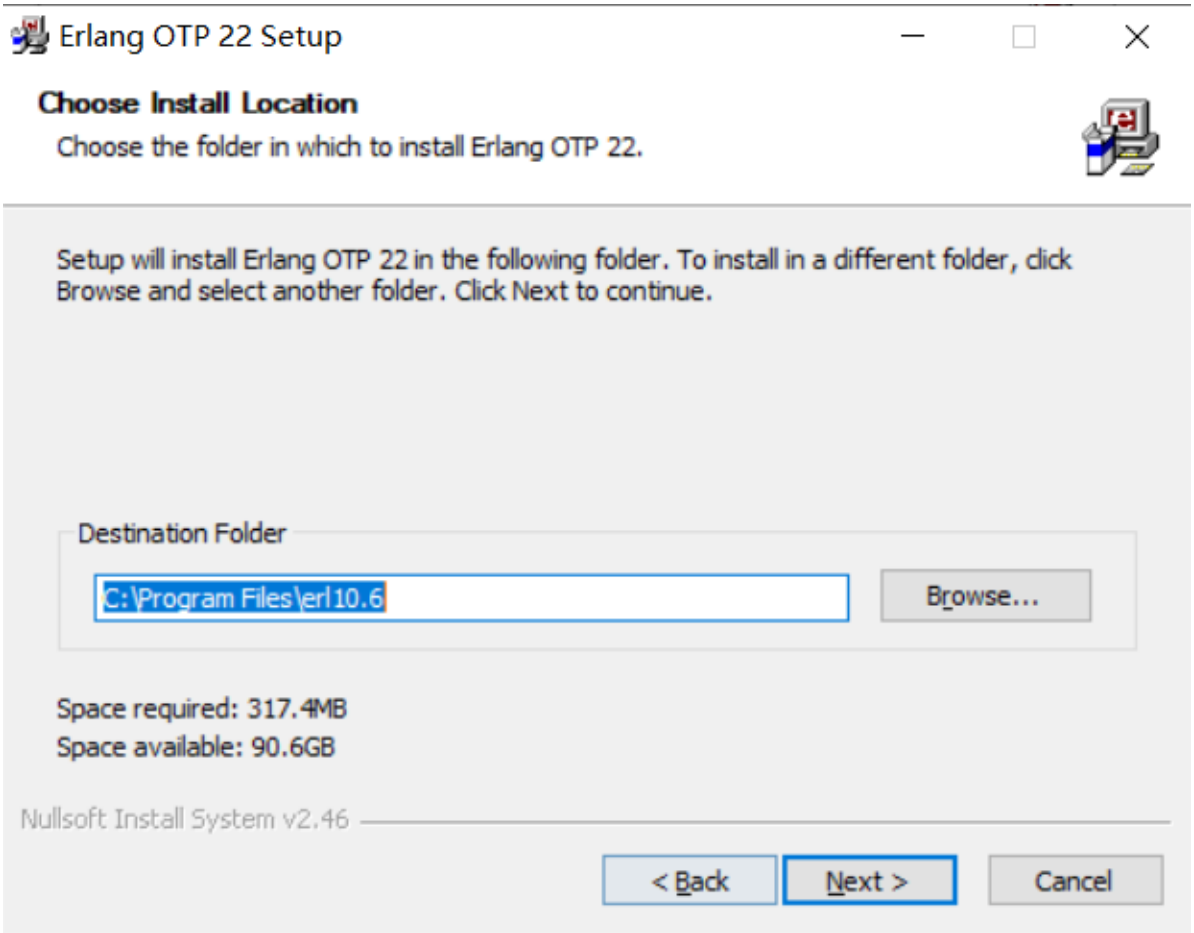
Spring Cloud Bus是用轻量的消息代理将分布式的节点连接起来，可以用于广播配置文件的更改或者服务的监控管 理。也就是消息总线可以为微服务做监控，也可以实现应用程序之间相互通信。 Spring Cloud Bus可选的消息代理有RabbitMQ和Kafka。 使用了Bus之后， SpringCloudBus可以不用重启provider2就可以将码云上最新配置及时更新到本地

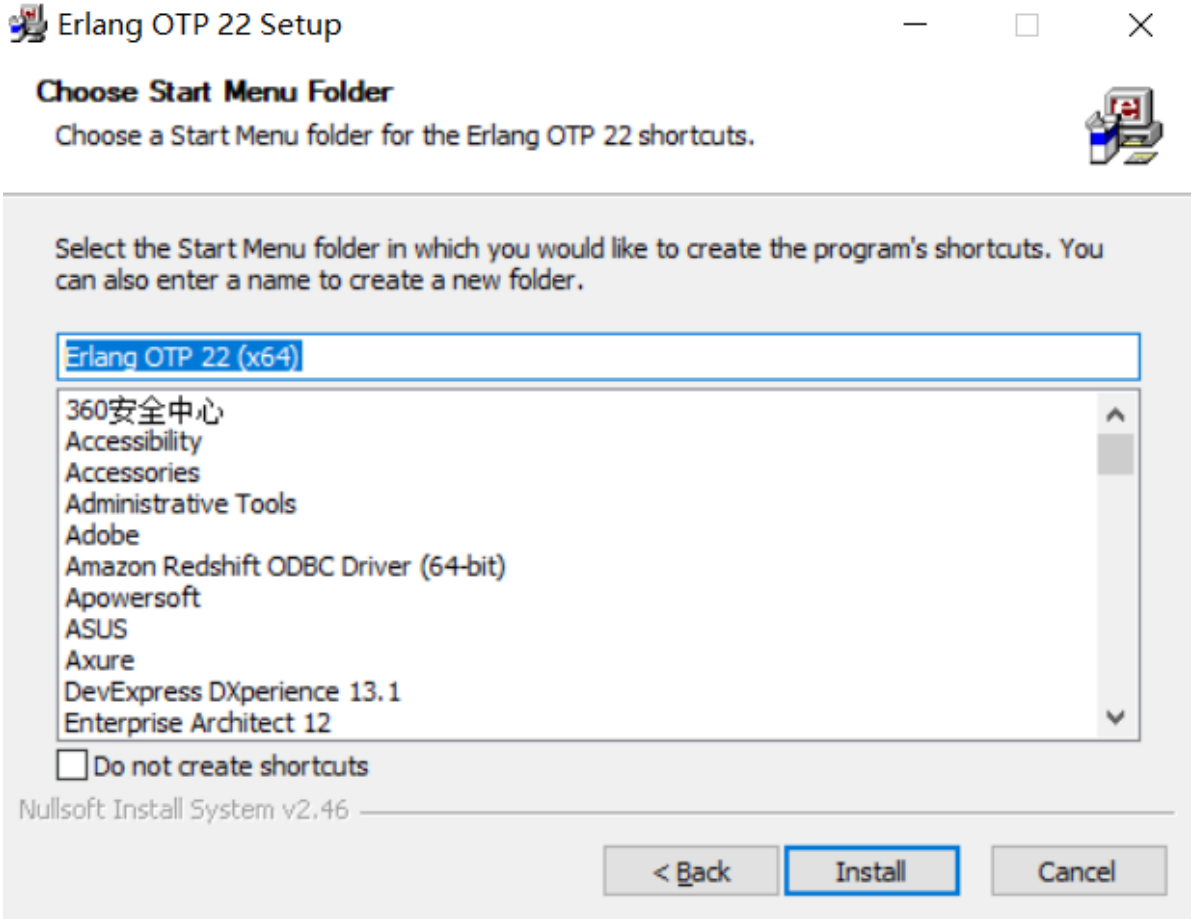
1、安装RabbtiMQ

1.1安装erlang环境

下载地址：https://www.erlang.org/downloads



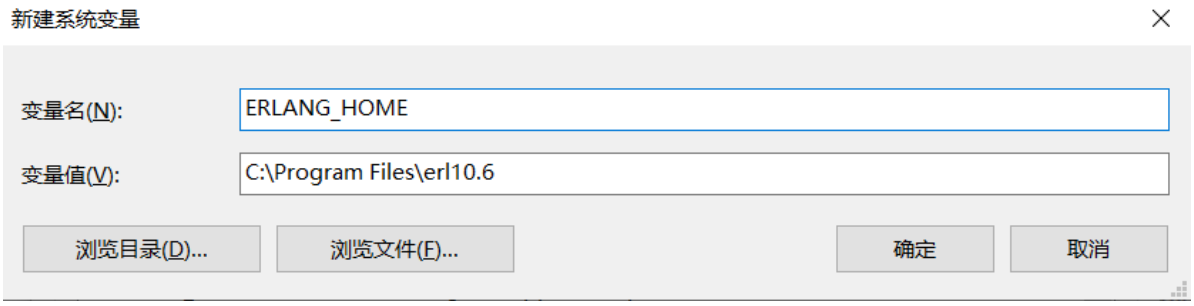




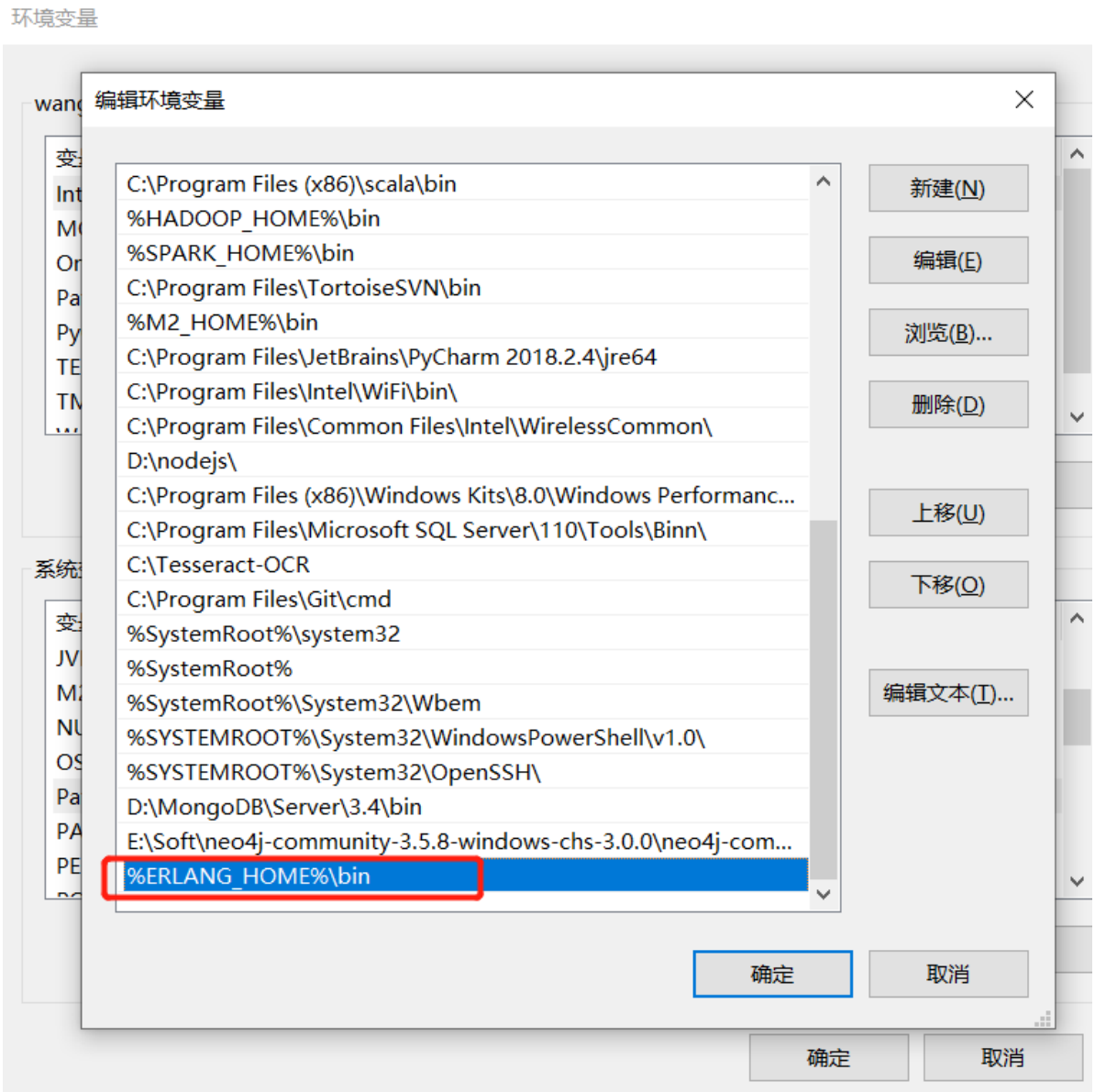
环境变量中添加:

ERLANG_HOME

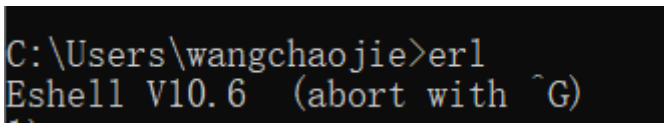
C:\Program Files\erl10.6



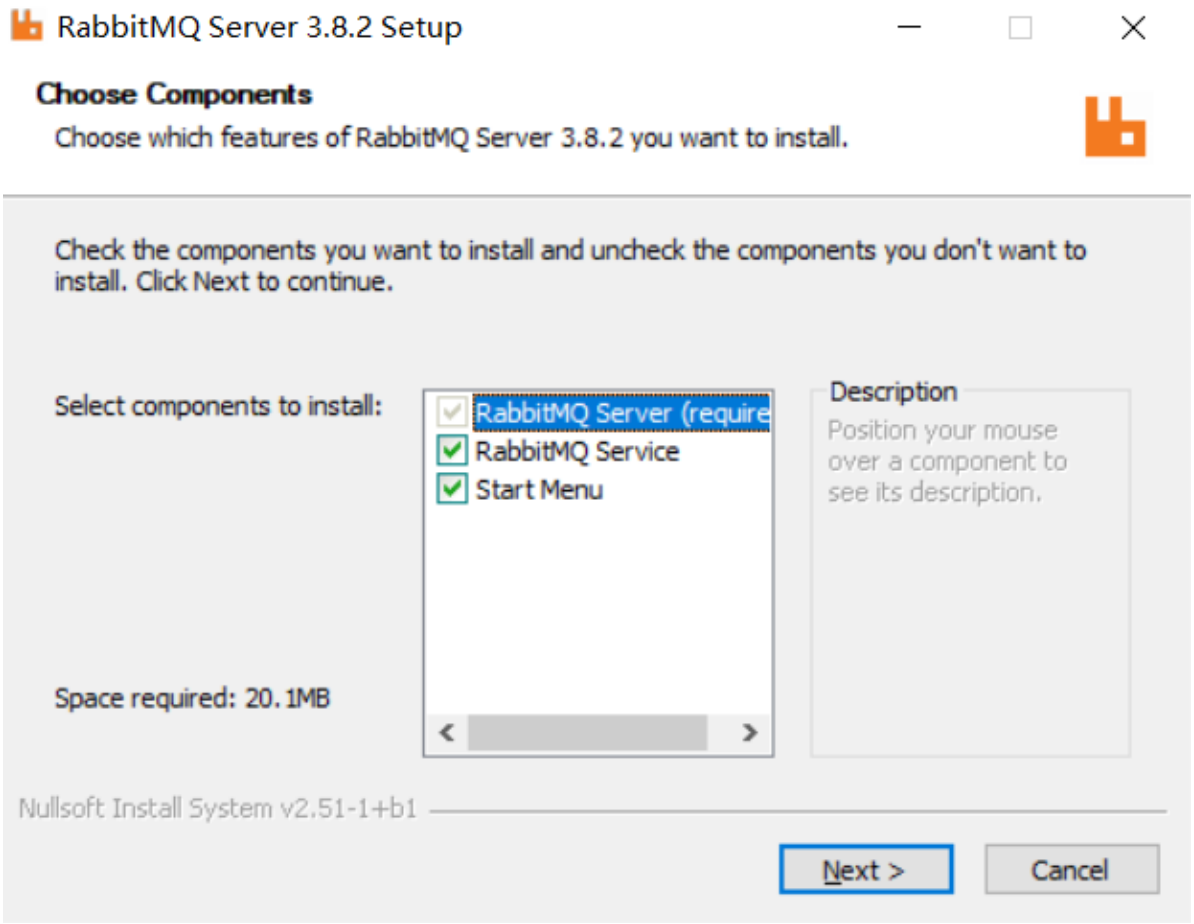
Path中添加：%ERLANG_HOME%\bin

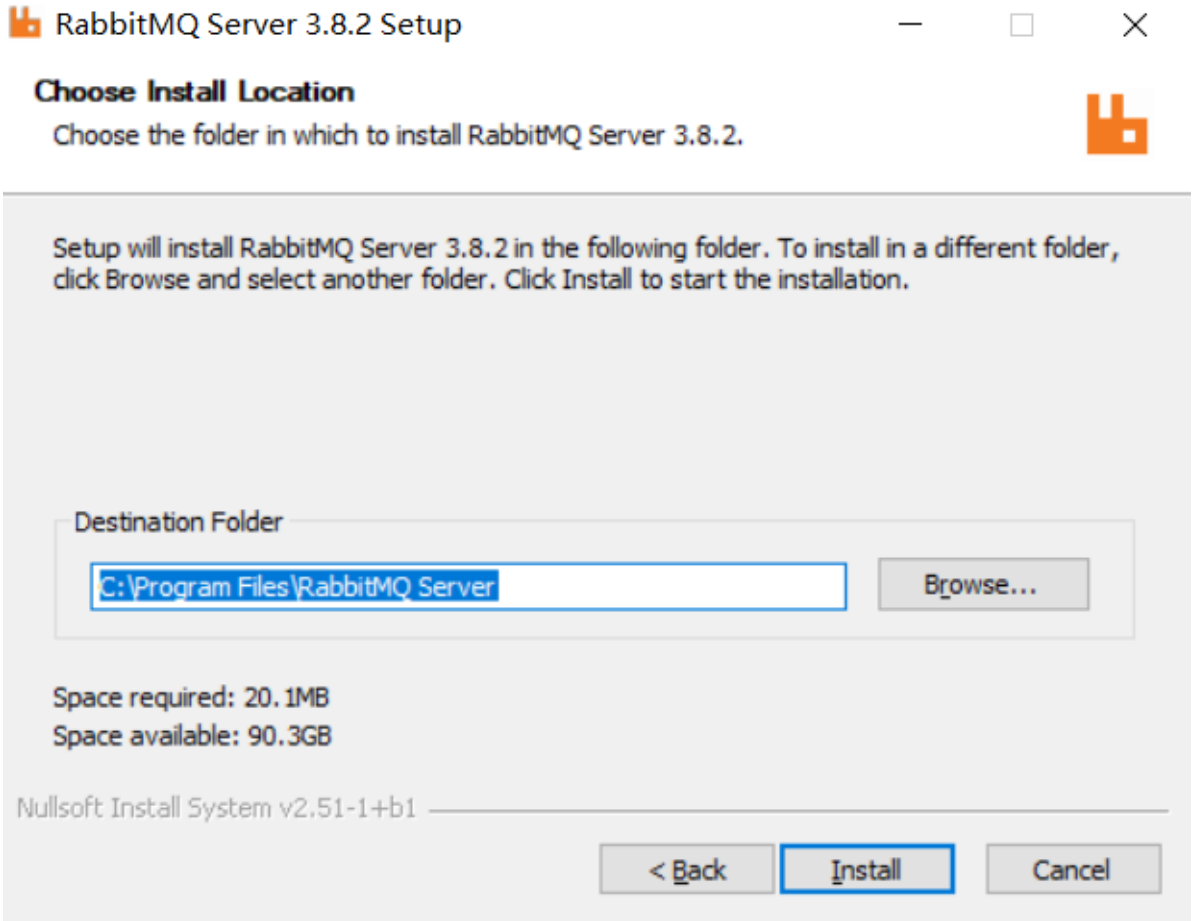


cmd中输入erl：



1.2、安装RabbitMQ





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

```
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.
```

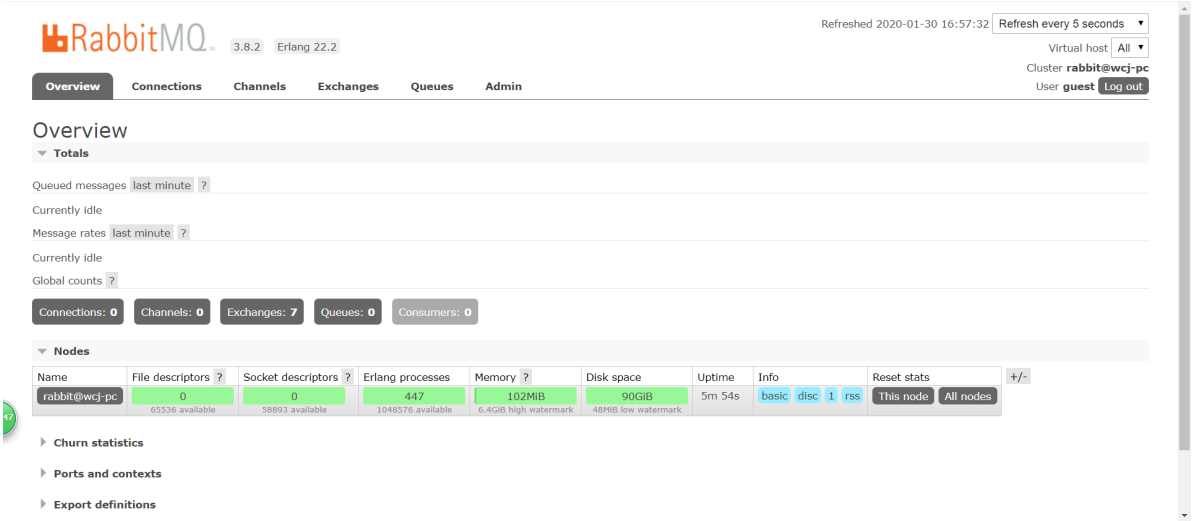
输入：rabbitmqctl status

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
```

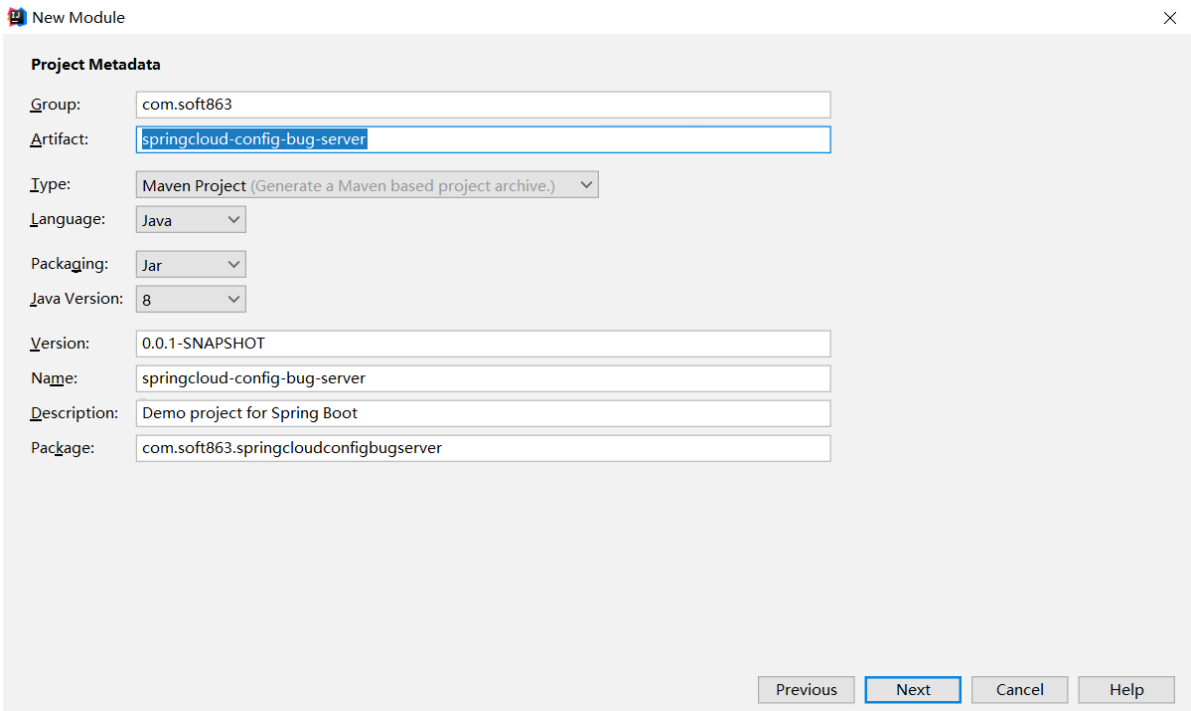
浏览器中访问：访问http://localhost:15672

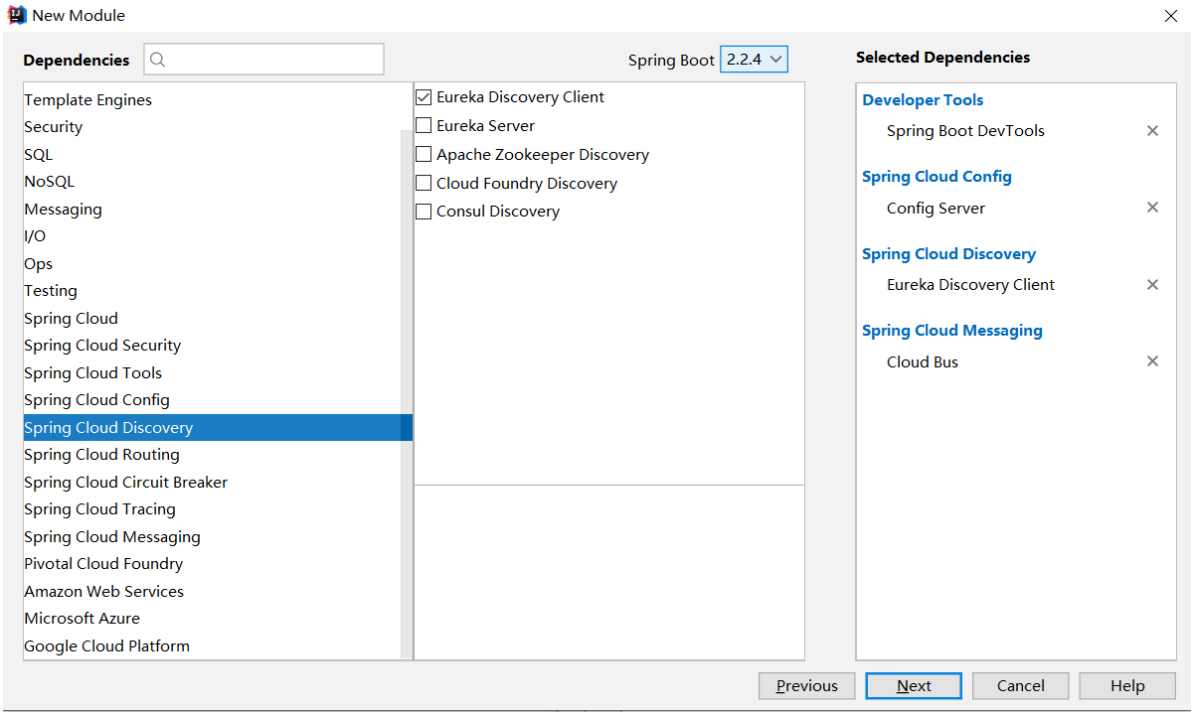
默认用户名和密码都是guest



2、创建Bus服务端

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





添加pom

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

整个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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.soft863</groupId>
    <artifactId>springcloud-config-bug-server</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springcloud-config-bug-server</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
        <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
```

```
            <artifactId>spring-cloud-bus</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-config-server</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </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>

```

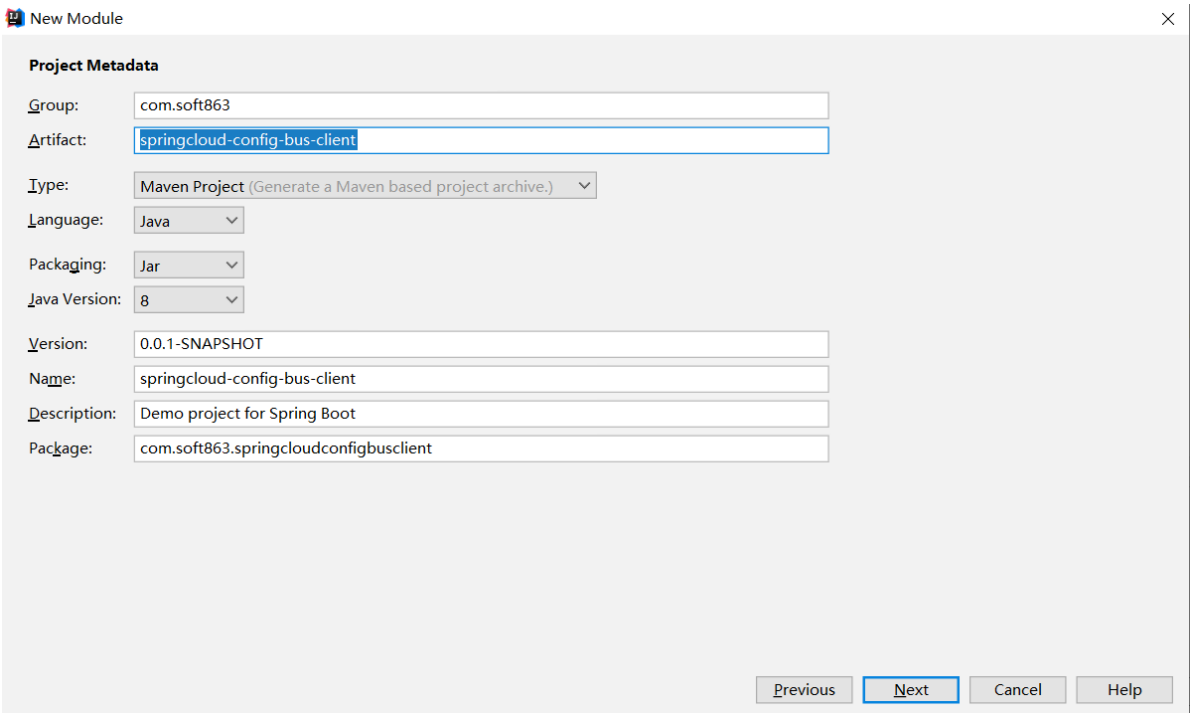
application.yml配置文件添加内容

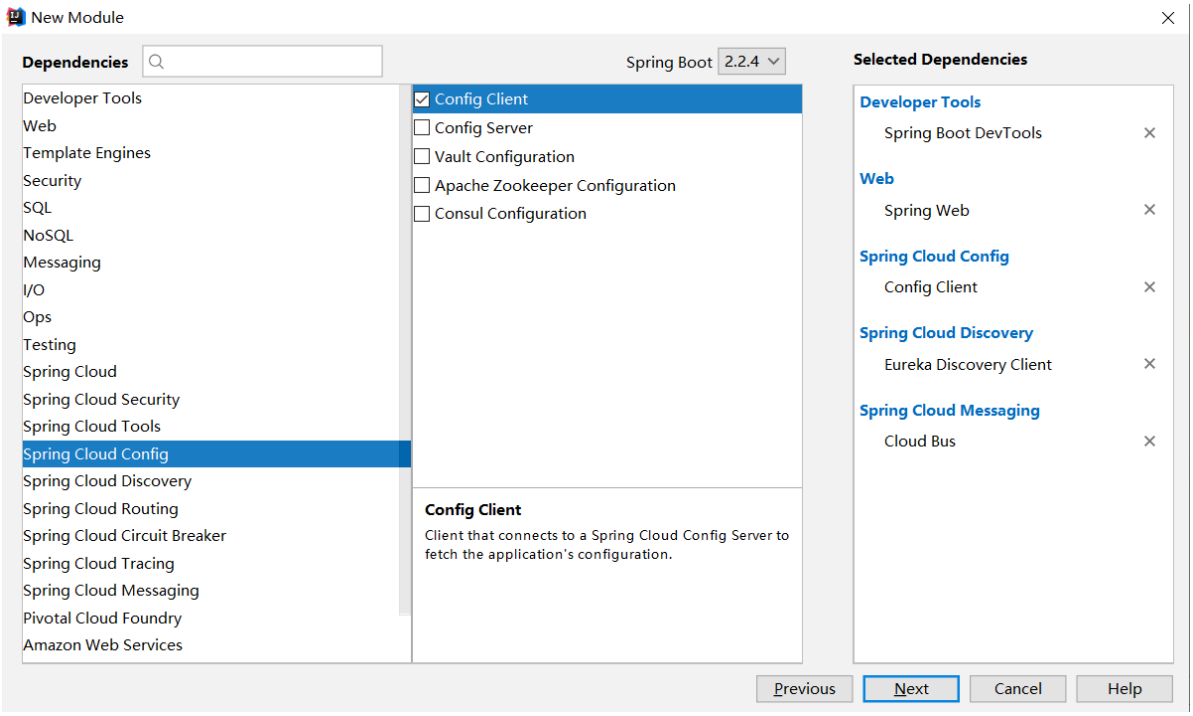
```
server:
  port: 8763
spring:
  application:
    name: configserver
  cloud:
    config:
      server:
        git:
          uri: https://gitee.com/mochiwcj/springcloud-config.git
  #默认，可以不配置
  rabbitmq:
    host: 127.0.0.1
    port: 5672
    username: guest
    password: guest
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 127.0.0.1
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
management:
  endpoints:
    web:
      exposure:
        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





pom文件中添加：

```
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>

```

完整pom

```
<?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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.soft863</groupId>
    <artifactId>springcloud-config-bus-client</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springcloud-config-bus-client</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
        <spring-cloud.version>Hoxton.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.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
```

```
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
    </dependencies>

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

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

</project>


```

删除项目的appliation配置文件，新增bootstrap.yml如下

```
spring:
  application:
    name: springcloud-config-bus-client
  cloud:
    config:
      # 与远程仓库中的配置文件的application保持一致
      name: user
      # 远程仓库中的配置文件的profile保持一致
      profile: dev
      # 远程仓库中的配置文件的profile保持一致
      label: master
      discovery:
        # 使用配置中心
        enabled: true
        service-id: configserver
```

```
    bus:
      trace:
        enabled: true
  #默认可以不写
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 127.0.0.1
  client:
    service-url:
      defaultZone: http://localhost:8762/eureka/

```

创建UserController

```
package com.soft863.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping("/user")
@RefreshScope
public class UserController {

    @Value("${soft863.test}")
    private String name;

    @RequestMapping("/test")
    @ResponseBody
    public String getResult() {
        return "配置文件测试内容：" + name;
    }
}


```

启动类上添加注解

```
@EnableEurekaClient
@ComponentScan("com.soft863")

```

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

```
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