SpringCloud Config

一.简介

SpringCloud Config为微服务架构中的微服务提供集中化的外部配置支持，配置服务器为各个不同微服务应用的所有环境提供了一个中心化的外部配置

SpringCloud Config分为服务端和客户端。

服务端也称为分布式配置中心，它是一个独立的微服务应用，用来连接配置服务器并且为客户端提供获取配置信息，加密、解密信息等访问接口

客户端则是通过指定的配置中心来管理应用资源，以及与业务相关的配置内容。并且在启动的时候从配置中心获取和加载配置信息。配置服务器默认使用git来存储配置信息，这样有助于对环境配置进行版本管理，并且可以通过git客户端工具来方便的管理和访问配置内容。

二、 使用步骤

1.在github上新建Repository

名称为springcloud-config

https://github.com/mutourenldh/springcloud-config.git

2.在本地新建git仓库并且clone

新建文件夹E:\GitHub

在GitHub文件下执行git命令

```
git status
#在GitHub文件下初始化一个git仓库springcloud-config
git clone
https://github.com/mutourenldh/springcloud-
config.git

```

3.创建application.yml文件

在E:\GitHub\springcloud-config中新建一个application.yml文件

```
#保存为utf-8格式
spring:
  profiles:
    active:
    - dev

---
spring:
  profiles: dev   #开发环境
  application:
    name: springcloud-config-haoge-dev
```

```yaml
---
spring:
  profiles: test  #测试环境
  application:
    name: springcloud-config-haoge-test
```

4.将application.yml推送到github上

分别执行如下命令将我们新建的application.yml文件推送到github上

```
git add .
git commit -m "init file"
git push origin master

```

5.创建config服务端项目

引入依赖

```
<dependency>

<groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
```

新建maven项目springcloud-config-server-3344

5.1POM依赖

```
<dependencies>
        <!-- springCloud Config -->
        <dependency>

<groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-config-
server</artifactId>
        </dependency>
        <!-- 避免Config的Git插件报错：
org/eclipse/jgit/api/TransportConfigCallback -->
        <dependency>
            <groupId>org.eclipse.jgit</groupId>

<artifactId>org.eclipse.jgit</artifactId>
            <version>4.10.0.201712302008-
r</version>
        </dependency>
        <!-- 图形化监控 -->
        <dependency>

<groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-
actuator</artifactId>
        </dependency>
        <!-- 熔断 -->
        <dependency>

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

```
            <artifactId>spring-cloud-starter-hystrix</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-eureka</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-jetty</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
```

```
40
<groupId>org.springframework.boot</groupId>
41              <artifactId>spring-boot-starter-
test</artifactId>
42          </dependency>
43          <!-- 热部署插件 -->
44          <dependency>
45
<groupId>org.springframework</groupId>
46
<artifactId>springloaded</artifactId>
47          </dependency>
48          <dependency>
49
<groupId>org.springframework.boot</groupId>
50              <artifactId>spring-boot-
devtools</artifactId>
51          </dependency>
52      </dependencies>
```

5.2 application.yml

```yaml
server:
  port: 3344

spring:
  application:
    name:  springcloud-config
  cloud:
    config:
      server:
        git:
          uri:
https://github.com/mutourenldh/springcloud-
config.git #GitHub上面的git仓库名字

```

5.3 主启动类

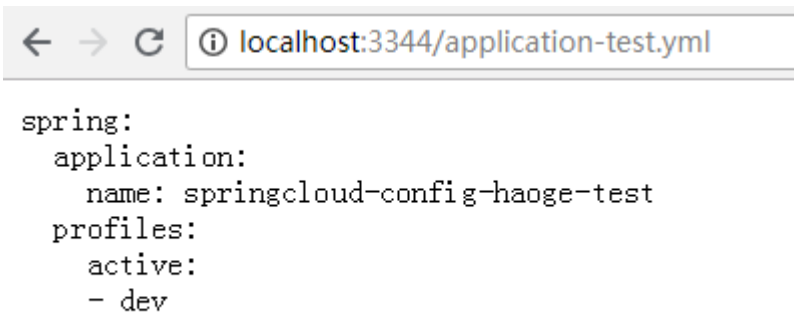
```java
@SpringBootApplication
@EnableConfigServer
public class Config_3344_StartSpringCloudApp
{
    public static void main(String[] args)
    {

SpringApplication.run(Config_3344_StartSpringClou
dApp.class, args);
    }
}
```

5.4 测试链接：

http://localhost:3344/application-test.yml

http://localhost:3344/application-dev.yml

测试结果：



支持的别的访问格式：

```
/{application}/{profile}[/{label}]
/{application}-{profile}.yml
/{label}/{application}-{profile}.yml
/{application}-{profile}.properties
/{label}/{application}-{profile}.properties
```

其中profile指的是版本，如dev；label指的是git分支名称，如master

例子：

http://localhost:3344/application/dev/master

http://localhost:3344/master/application-dev.yml

6.客户端测试

6.1新建springcloud-config-client.yml文件

在本地目录E:\GitHub\springcloud-config目录下新建文件springcloud-config-client.yml文件

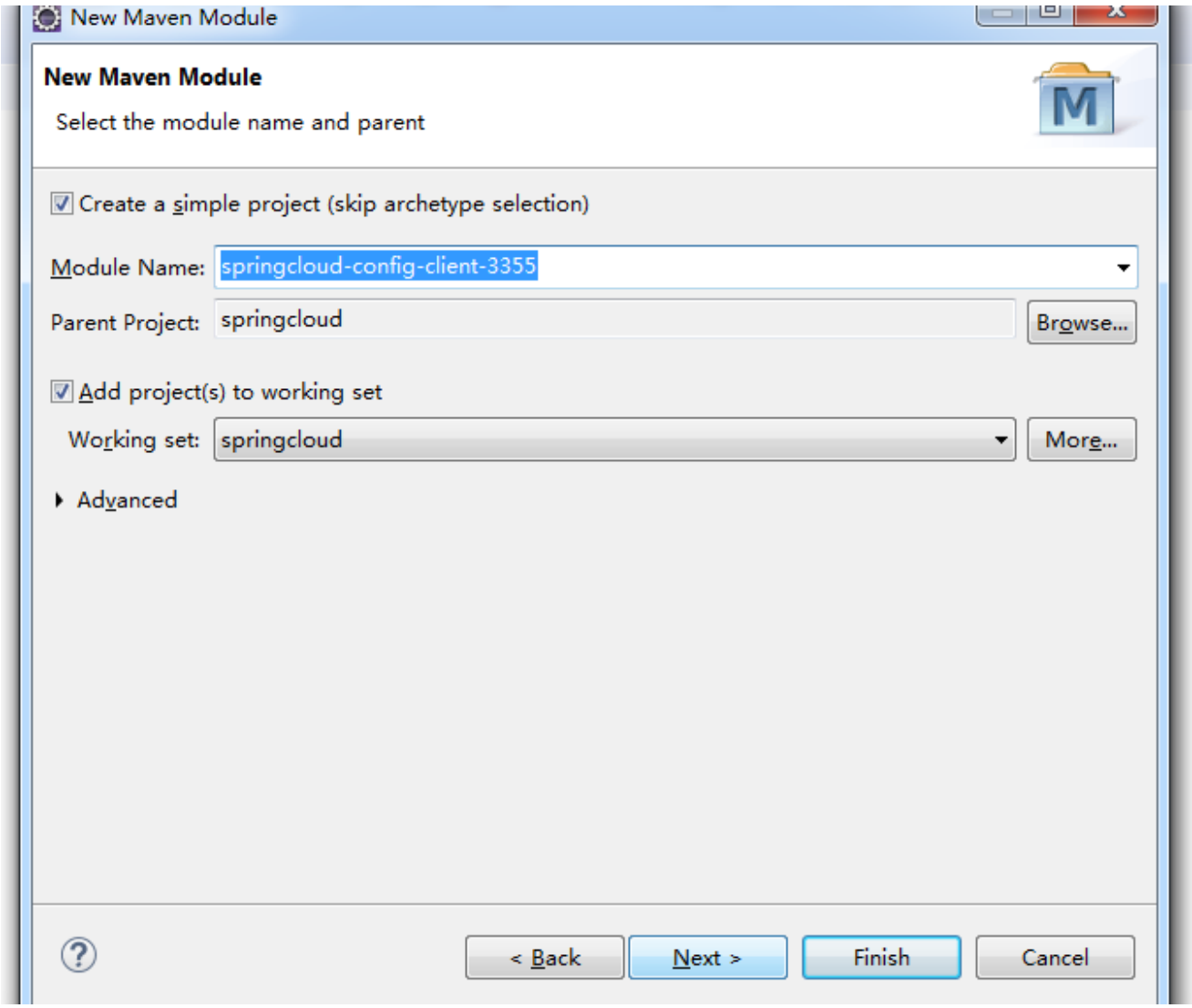
```yaml
#保存为utf-8格式
spring:
  profiles:
    active:
    - dev
---
server:
  port: 8201
spring:
```

```
  profiles: dev
  application:
    name: springcloud-config-client
eureka:
  client:
    service-url:
      defaultZone: http://eureka-dev.com:7001/eureka/

---
server:
  port: 8202
spring:
  profiles: test
  application:
    name: springcloud-config-client
eureka:
  client:
    service-url:
      defaultZone: http://eureka-test.com:7001/eureka/
```

6.2 把这个文件推送到远程github上

```
git add .
git commit -m "init file"
git push origin master
```

6.3新建maven子项目（模块）



6.4 pom文件

```
<dependencies>
        <!-- SpringCloud Config客户端 -->
        <dependency>

<groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
config</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-hystrix</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-eureka</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-jetty</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>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>springloaded</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
        </dependency>
    </dependencies>
```

6.5 application.yaml文件

```
spring:
  application:
    name: springcloud-config-client
```

6.6 bootstrap.yml文件

```
spring:
  cloud:
    config:
      name: springcloud-config-client #需要从github上读取的资源名称，注意没有yml后缀名
      profile: test   #本次访问的配置项
      label: master
      uri: http://localhost:3344  #本微服务启动后先去找3344号服务，通过SpringCloudConfig获取GitHub的服务地址

```

application.yml是用户级别的资源配置项

bootstrap.yml是系统级别的资源配置项，优先级更高

6.6 测试文件

```
package com.haoge.cloud.config.rest;
@RestController

```

```
3  public class ConfigClientRest
4  {
5      //获取配置文件spring.application.name的值
6      @Value("${spring.application.name}")
7      private String applicationName;
8      //获取配置文件eureka.client.service-
   url.defaultZone的值
9      @Value("${eureka.client.service-
   url.defaultZone}")
10     private String eurekaServers;
11     //获取配置文件server.port的值
12     @Value("${server.port}")
13     private String port;
14
15     @RequestMapping("/config")
16     public String getConfig()
17     {
18         String str = "applicationName: " +
   applicationName + "\t eurekaServers:" +
   eurekaServers + "\t port: " + port;
19         System.out.println("******str: " + str);
20         return "applicationName: " +
   applicationName + "\t eurekaServers:" +
   eurekaServers + "\t port: " + port;
21     }
22 }
```

6.8 主启动类

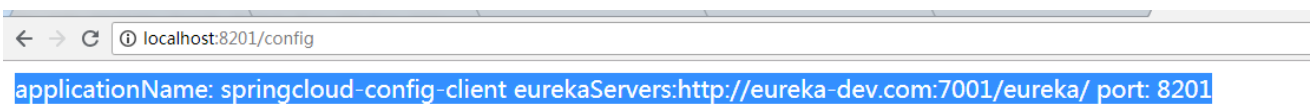
```
package com.haoge.cloud.config;
@SpringBootApplication
public class
ConfigClient_3355_StartSpringCloudApp
{
    public static void main(String[] args)
    {

SpringApplication.run(ConfigClient_3355_StartSpri
ngCloudApp.class, args);
    }
}
```

6.9 测试

测试：首先启动3344的服务端项目，再启动3355的客户端的项目

测试链接：http://localhost:8202/config

将bootstrap.yml中的profile改为dev,则用链接http://localhost:8201/config测试结果：



三、原理

1.3355客户端项目启动之后，首先通过 bootstrap.yml文件中的配置，到3344服务上通过SpringCloudConfig获取GitHub的服务地址，在该地址下读取master分支上springcloud-config-client.yml配置文件中profile为test中的配置。端口号为8202，所以这个时候我们访问http://localhost:8202/config才可以返回结果。即根据bootstrap.yml配置文件中配置信息到gitHub上读取相应的本项目的配置信息。