# 1、服务端配置映射

## 1、新建工程 config-server 添加依赖

*<?*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.didispace</groupId>  
 <artifactId>config-server</artifactId>  
 <version>1.0.0</version>  
 <packaging>jar</packaging>  
  
 <name>config-server</name>  
 <description>Spring Cloud project</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.3.7.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-config-server</artifactId>  
 </dependency>  
  
 </dependencies>  
  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>Brixton.SR5</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>

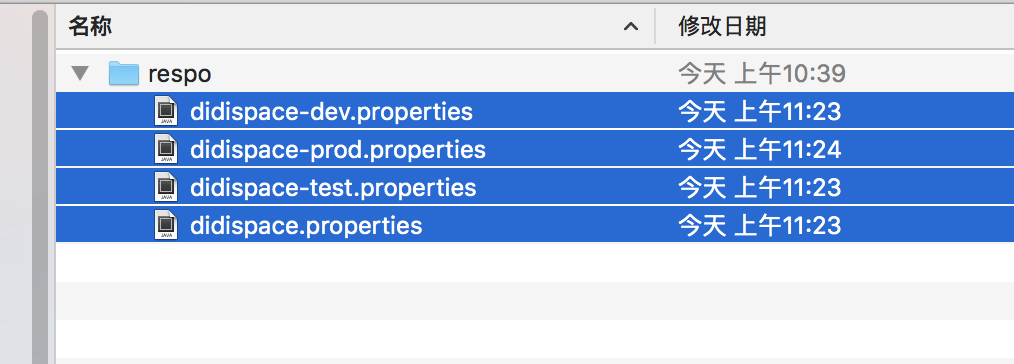
## 2、注解添加分布式配置服务端支持

@EnableConfigServer  
@SpringBootApplication  
public class Application {  
  
 public static void main(String[] args) {  
 new SpringApplicationBuilder(Application.class).web(true).run(args);  
 }  
  
}

## 3、创建git仓库项目[SpringcloudConfig](https://github.com/HealerJean123/SpringcloudConfig)

### 1、其根目录下同时创建文件夹respo

### 2、该文件夹下创建配置文件 命名规则 ：自定义项目名称-profile.properties

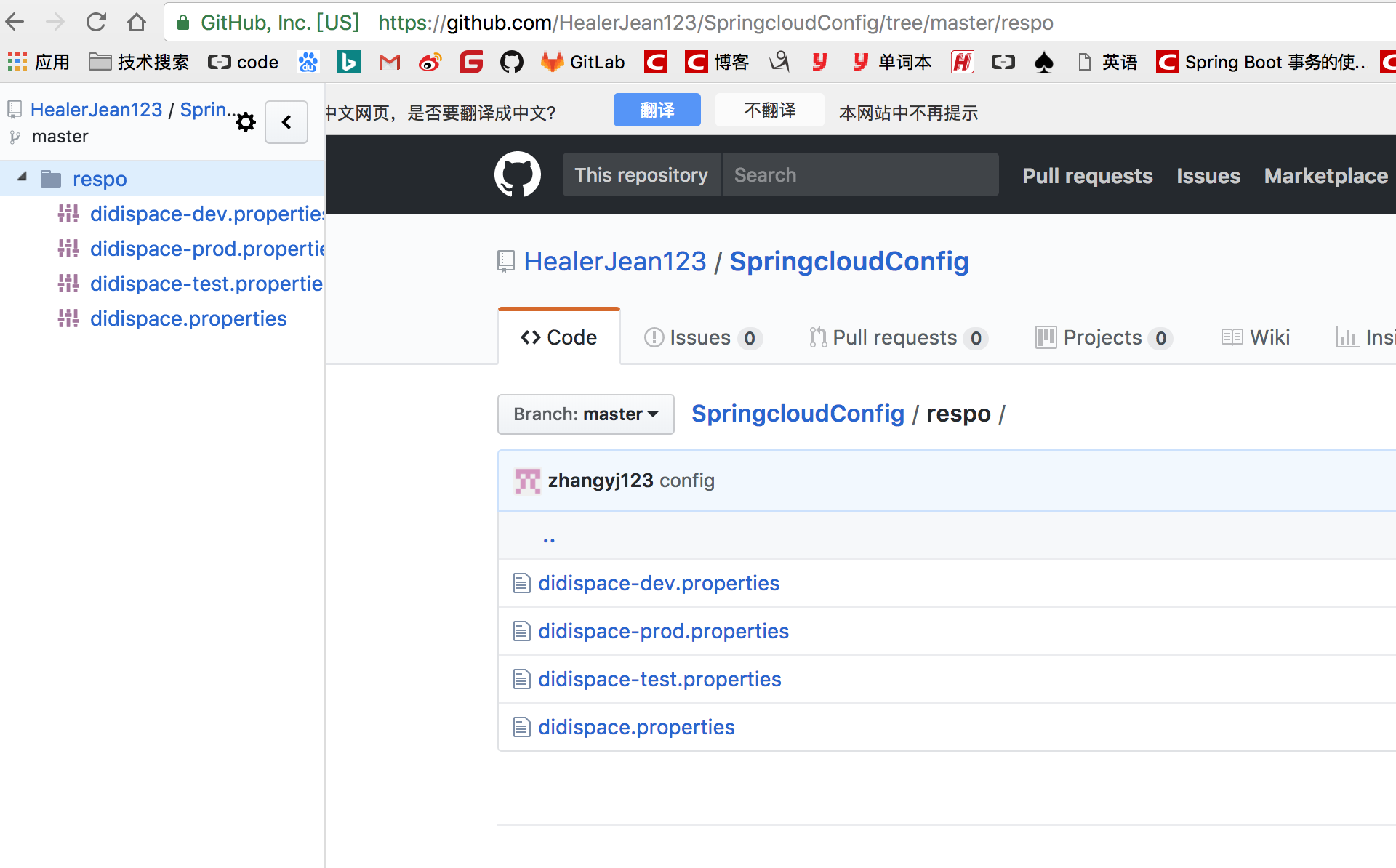


#### from=git-dev-1.0

#### from=git-prod-1.0

#### from=git-test-1.0

#### from=git-default-1.0



## 4、配置工程的配置文件，添加git仓库的属性，

spring.application.name=config-server  
server.port=7001  
  
# git仓库  
spring.cloud.config.server.git.uri=https://github.com/HealerJean123/SpringcloudConfig/  
spring.cloud.config.server.git.searchPaths=respo  
spring.cloud.config.label=master  
#公开仓库不需要写用户名和密码，如果是私有仓库则需要些用户名和密码  
spring.cloud.config.server.git.username=  
spring.cloud.config.server.git.password=

## 5、测试，启动这个项目7001端口 浏览器访问 http://localhost:7001/didispace/prod/master //+分支名称

### 解释：妈的，居然真的有显示，证明配置服务中心可以从远程程序获取配置信息。

#### name:应用名称 didispace

#### label 表示分支名称 master

#### version 表示git上对应的版本号

|  |
| --- |
| {  "name": "didispace",  "profiles": ["prod"],  "label": "master",  "version": "68a075d11aea67d4989989c70152a9ac53367e82",  "propertySources": [{  "name": "https://github.com/HealerJean123/SpringcloudConfig/respo/didispace-prod.properties",  "source": {  "from": "git-prod-1.0"  }  }, {  "name": "https://github.com/HealerJean123/SpringcloudConfig/respo/didispace.properties",  "source": {  "from": "git-default-1.0"  }  }]  } |

http请求地址和资源文件映射如下:

* /{application}/{profile}[/{label}] 上上面的链接http://localhost:7001/didispace/prod/master
* /{application}-{profile}.yml
* /{label}/{application}-{profile}.yml
* /{application}-{profile}.properties http://localhost:7001/didispace-dev.properties
* 
* /{label}/{application}-{profile}.properties

# 2、客户端映射

## 1、创建工程config-client，pom.xml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.didispace</groupId>  
 <artifactId>config-client</artifactId>  
 <version>1.0.0</version>  
 <packaging>jar</packaging>  
  
 <name>config-client</name>  
 <description>Spring Cloud project</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.3.7.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <java.version>1.8</java.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>  
 </dependencies>  
  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>Brixton.SR5</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>

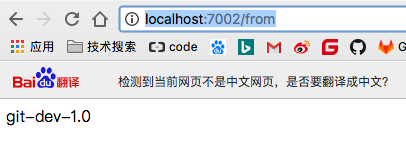
## 2、创建bootstrap.properties，来获取配置文件config-server的位置

server.port=7002  
  
spring.application.name=didispace  
spring.cloud.config.profile=dev  
spring.cloud.config.label=master  
spring.cloud.config.uri=http://localhost:7001/

## 3、创建controller开始测试，通过后台的from（只要在配置文件中，其实都是from）

@RefreshScope  
@RestController  
public class TestController {  
  
 @Value("${from}")  
 private String from;  
  
 @RequestMapping("/from")  
 public String from() {  
 return this.from;  
 }  
  
}

## 4、浏览器访问 http://localhost:7002/from



# 3、添加Eurake服务注册中心

## 1、使用之前的注册中心1111

## 2、配置config-server

### 1、修改config-server 中pom.xml 添加eurake依赖包

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka</artifactId>  
</dependency>

### 2、启用注册发现，和配置服务注册地址

@EnableDiscoveryClient  
@EnableConfigServer  
@SpringBootApplication  
public class Application {  
  
 public static void main(String[] args) {  
 new SpringApplicationBuilder(Application.class).web(true).run(args);  
 }  
  
}

#eurake  
#指定服务注册中心  
eureka.client.serviceUrl.defaultZone=http://localhost:1111/eureka/

## 2、配置config-client

### 1、添加eurake的依赖包

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka</artifactId>  
</dependency>

### 2、启用注册发现

@EnableDiscoveryClient  
@SpringBootApplication  
public class Application {  
  
 public static void main(String[] args) {  
 new SpringApplicationBuilder(Application.class).web(true).run(args);  
 }  
  
}

### 3、配置服务注册中心和服务端的config

server.port=7002  
spring.application.name=didispace  
  
  
spring.cloud.config.enabled=true  
spring.cloud.config.discovery.service-id=CONFIG-SERVER  
spring.cloud.config.profile=dev  
  
  
  
#eurake  
#指定服务注册中心  
eureka.client.serviceUrl.defaultZone=http://localhost:1111/eureka/

## 3、可以开始测试了，这里我不知道为什么没有成功，但是基本上就是这样的。