spring-cloud-config 配置管理

spring-cloud-config client服务端

Maven导入

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

1.Application类上增加@EnableConfigServer

2.application.properties配置

```
server.port = 9090
spring.application.name = ws
### 拉取远程git中配置
spring.cloud.config.server.git.uri = file:///G:/project/git-test
### 关闭actuator验证
management.endpoint.beans.enabled = false
```

3.增加git文件

ws.properties 内容:my.name = default

ws-dev.properties 内容:my.name = dev

4.测试

http://localhost:9090/ws-default.properties会出现my.name: test123,表示成功!

spring-cloud-config client客户端

Maven导入

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-actuator</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-config</artifactId>
</dependency>
```

1.增加bootstrap.properties

```
# 配置服务器url
spring.cloud.config.uri = http://localhost:9090/
# 远程地址properties 前缀 [ws.properties]
spring.cloud.config.name = ws
# 远程地址properties -后缀 [ws-dev.properties]
spring.cloud.config.profile = default
# git仓局分支
spring.cloud.config.label = master
```

2.修改application.properties

```
spring.application.name = client
#springcloud-Finchley.SR1版本中,意思是:使用包含暴露的服务端口
management.endpoints.web.exposure.include = env,refresh
```

3.修改Application.java

```
package com.gupao.springcloudclientconfig;

import org.omg.CORBA.Environment;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class SpringCloudClientConfigApplication {

    @Value("${my.name}")
    public String name;
    public static void main(String[] args) {
        SpringApplication.run(SpringCloudClientConfigApplication.class, args);
    }

    @GetMapping("/config")
    public String getConfig(){
        return name;
    }
}
```

4.测试

http://localhost:8080/config,成功显示default.

原文链接：spring-cloud-config 配置管理

5.修改配置生效

http://localhost:8080/actuator/env -> search ws.properties -> 通过POST方法请求localhost:808/actuator/refresh 刷新配置文件，发现已变更则成功!

6.修改后刷新bean

在需要修改的类中增加@RefreshScope注解,就会开启刷新@Value的值