10 - Spring Cloud Config-Get updated config file by manually refresh

10.1 Manually fresh using actuator

In order to solve the problem that the latest configuration cannot be obtained without restarting the Config client, next, we will modify micro-service-cloud-config-client-3355, and the modification steps are as follows.

(1). Add the following dependencies to the pom.xml of micro-service-cloud-config-client-3355, and introduce the Spring Boot actuator monitoring module.

```
1  <dependency>
2      <groupId>org.springframework.boot</groupId>
3      <artifactId>spring-boot-starter-actuator</artifactId>
4  </dependency>
```

(2). Add the following configuration in the configuration file bootstrap.yml to expose the monitoring nodes of Spring Boot actuator to the outside world.

```
1  # Spring Boot 2.50 screens most of the nodes for actuator monitoring, and only exposes the
   health node. The configuration (*) in this section is to enable all nodes
2  management:
3    endpoints:
4      web:
5        exposure:
6          include: "*"   # * is a keyword in the yaml file, so quotation marks are required
```

(3). Use the @RefreshScope annotation on the ConfigClientController class to enable configuration refresh. The code is as follows.

```
1  package com.luxbp.controller;
2
3  import org.springframework.beans.factory.annotation.Value;
4  import org.springframework.cloud.context.config.annotation.RefreshScope;
5  import org.springframework.web.bind.annotation.GetMapping;
6  import org.springframework.web.bind.annotation.RestController;
7
8  //Read the content of the specified configuration file in the configuration center and
   display it on the page
9  @RefreshScope //In order to get the latest git configuration dynamically (manually), add the
   actuator to monitor and load RefreshScope,
10 @RestController
```

```
11 public class ConfigClientController {
12     @Value("${server.port}")
13     private String serverPort;
14     @Value("${config.info}")
15     private String configInfo;
16     @Value("${config.version}")
17     private String configVersion;
18     @GetMapping(value = "/getConfig")
19     public String getConfig() {
20         return "info: " + configInfo + "<br/>version: " + configVersion + "<br/>port: " +
    serverPort;
21     }
22 }
```

(4). Restart micro-service-cloud-config-client-3355, and then modify the config.version in the configuration file config-dev.yml to 3.0, the configuration is as follows.

```
1 config:
2   info: com.luxbp engin
3   version: 0.0.3
```

(5). Use the browser to access "http://localhost:3355/getConfig" again, and the result is as shown in the figure below.

> info: com.luxbp engin
> version: 0.0.2
> port: 3355

As can be seen from the above figure, even if we modify the Spring Cloud Config client, we still cannot directly obtain the latest configuration.

The reason is that we have not notified the actuator monitoring module to refresh the configuration. The code in this section is no different from the code in the previous section. In terms of architecture, the main difference is that we have added an actuator in this part. The so-called manual refresh means that we send a message to the actuator integrated in the client, and let the actuator notify the client to pull information. It should be noted that if there are multiple clients, you need to send messages to the actuators of each client separately.

(6). Open the command line window, use the following command to send a POST request to refresh the Spring Cloud Config 3355 client, and notify the client that the configuration file has been modified and the configuration needs to be pulled again.

```
1 curl -X POST "http://localhost:3355/actuator/refresh"
```

7. Use the browser to visit "http://localhost:3355/getConfig" again, and the result is as shown in the figure below.

info：com.luxbp engin
version：0.0.3
port：3355

10.1 Problems in manually refresh

......