我们已经掌握了通过RestTemplate负载均衡机制来调用注册中心中的服务,除了 RestTemplate外,SpringCloud还提供了另一个强大的组件Feign,这让客户端开发人员的编码更加简洁高效。

本章节的代码将以上一节的完整代码为基础进行feign改造实现

1. 服务调用者user-service

feign是工作在服务调用者一方的,因此只要改造user-service工程,将其中通过 RestTemplate调用message服务的部分改造成feign即可

a)添加Feign依赖

pom. xm1中添加

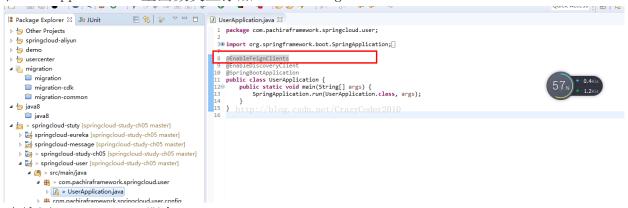
<dependency>

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

```
</dependency>
 # 4-springcloud-study
                                                               <name>4-springcioud-user</name>
  👺 4-springcloud-eureka
                                                         5
                                                               <description>用户微服务</description>
  🔛 4-springcloud-message
                                                         6
                                                               <artifactId>4-springcloud-user</artifactId>
  √ 👑 4-springcloud-user
                                                         7
                                                               <packaging>jar</packaging>
   > 乃 src/main/java
                                                         8
   > # src/test/iava
   > A JRE System Library [JavaSE-1.8]
                                                         9⊝
   > 🔌 Maven Dependencies
                                                                 <groupId>lizzy.springcloud</groupId>
                                                       <u>%</u>10
   > 📂 src
                                                        11
                                                                  <artifactId>4-springcloud-study</artifactId>
   > 🎘 target
    pom.xml
                                                                  <version>0.0.1-SNAPSHOT</version>
                                                       912
   m pom.xml
                                                        13
                                                               </parent>
> 🔛 5-springcloud-study
> 👺 6-springcloud-study
                                                        14
> 👺 7-springcloud-study
                                                        15⊜
                                                               <dependencies>
> 👺 8-springcloud-study
                                                       16⊝
                                                        17
                                                                     <groupId>org.springframework.cloud</groupId>
                                                        18
                                                                     <artifactId>spring-cloud-starter-feign</artifactId>
                                                        19
                                                                   </dependency>
                                                        20⊝
                                                                  <dependency>
                                                        21
                                                                    <groupId>org.springframework.cloud</groupId>
                                                        22
                                                                     <artifactId>spring-cloud-starter-eureka</artifactId>
                                                        23
                                                                  </dependency>
```

b)项目中启用Feign

在UserApplication主函数类上添加@EnableFeignClients



c)编写FeignClient服务

新增MessageService接口。使用@FeignClients注解进行标注,注解中接受一个参数,标示出客户端要调用的注册中心中服务的id,用于创建Ribbon负载均衡器

接口中的方法和服务提供者中的定义保持一致,注意,这里我们调用的是SmsController中的send2方法,因为这个方法的接收参数添加了@RequestBody参数。代码如下:

```
4-springcloud-study
                                       6 import org.springframework.http.ResponseEntity;
 👑 4-springcloud-eureka
                                       7 import org.springframework.web.bind.annotation.RequestMapping;
  4-springcloud-message
                                       8 import org.springframework.web.bind.annotation.RequestMethod;
   / 🌁 src/main/java
    10 import lizzy.springcloud.user.controller.UserController.SmsSendResponse;
                                       11 import lombok.Data;
      >  WebConfig.java
      ⊕ controller
     13 @FeignClient("message-service")
      ✓ MessageService.java

→ MessageService
                                       14 public interface MessageService {
                                           @RequestMapping(method=RequestMethod. POST, value="/message/sms/send2")
                                       15⊝
     >  UserApplication.java
  > 乃 src/test/java
                                      16
                                           public ResponseEntity<SmsSendResponse> send(SmsSendRequest request);
  > M JRE System Library [JavaSE-1.8]
  > 🚵 Maven Dependencies
                                      18
                                           public static class SmsSendRequest {
  > 🍃 src
  > 🇀 target
   pom.xml
                                       20
                                             * 短信模版ID
pom.xml
5-springcloud-study
6-springcloud-study
                                             private String templateld;
package lizzy. springcloud. user. service;
import java.util.Map;
import org. springframework. cloud. netflix. feign. FeignClient;
import org. springframework. http. ResponseEntity;
import org. springframework. web. bind. annotation. RequestMapping;
import org. springframework. web. bind. annotation. RequestMethod;
import lizzy. springcloud. user. controller. UserController. SmsSendResponse;
import lombok. Data;
@FeignClient("message-service")
public interface MessageService {
           @RequestMapping (method=RequestMethod. POST, value="/message/sms/send2")
           public ResponseEntity<SmsSendResponse> send(SmsSendRequest request);
           @Data
           public static class SmsSendRequest {
                      /**
                      * 短信模版ID
                      private String templateId;
                      /**
                      * 要发送的手机号
                      */
                      private String mobile;
                      /**
                      * 模版中携带的参数信息
                      */
                      private Map<String, Object> params;
```

d)调用方UserController改造

public class UserController {

新增一个regist/sms2方法,用于和先前的基于RestTemplate方法调用区分开

```
60
  4-springcloud-eureka

4-springcloud-message
                                         61⊜
  * Feign声明式调用
                                         62
                                         63
                                              * @param mobile

    H lizzy.springcloud.user

                                         64
                                              * @return
      65

⊕ controller

                                              @RequestMapping(value= {"regist/sms2"})
                                         66⊖
       ✓ ☑ UserController.java

> ☑ UserController
                                         67
                                              public ResponseEntity < SmsSendResponse > sms2(String mobile){
                                         68
                                                Random random = new Random();
      # service
       ✓ If MessageService.java
> ■ MessageService
                                                int next = random.nextInt(10000000):
                                         69
                                                String code = ""+(10000000-next);
                                         70
      >  UserApplication.java
   > # src/test/java
> M JRE System Library [JavaSE-1.8]
                                         71
72
                                                ResponseEntity<SmsSendResponse> response = doSendFeign(mobile, code);
                                                return response;
   > Maven Dependencies
                                         73
   > 🇀 src
> 🇀 target
                                         749
                                              public ResponseEntity<SmsSendResponse> doSendFeign(String mobile,String code) {
  pom.xml
pom.xml
5-springcloud-study
                                         75
                                                SmsSendRequest request = new SmsSendRequest();
                                         76
                                                request.setMobile(mobile):
                                         77
                                                request.setTemplateId("CHECK CODE");
 6-springcloud-study
7-springcloud-study
                                         78
                                                Map<String, Object> params = new HashMap<>();
 8-springcloud-study
                                               params.put("code", code);
                                         79
                                         80
                                                request.setParams(params);
                                         81
                                                return messageService.send(request);
                                         82
                                         83
具体代码内容如下
package lizzy. springcloud. user. controller;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import org. springframework. beans. factory. annotation. Autowired;
import org. springframework. http. HttpEntity;
import org. springframework. http. HttpHeaders;
import org. springframework. http. MediaType;
import org. springframework. http. ResponseEntity;
import org. springframework. util. LinkedMultiValueMap;
import org. springframework.util. MultiValueMap;
import org. springframework. web. bind. annotation. RequestMapping;
import org. springframework. web. bind. annotation. RestController;
import org. springframework. web. client. RestTemplate;
import lizzy. springcloud. user. service. MessageService;
import lizzy.springcloud.user.service.MessageService.SmsSendRequest;
import lombok. Data;
import lombok. extern. slf4j. Slf4j;
@S1f4j
@RestController
@RequestMapping("/user/")
```

```
@Autowired
        private RestTemplate restTemplate;
        @Autowired
        private MessageService messageService;
        @RequestMapping(value= {"regist/sms"})
        public ResponseEntity<SmsSendResponse> sms(String mobile) {
                Random random = new Random();
                int next = random.nextInt(10000000);
                String code = ""+(10000000-next);
                ResponseEntity (SmsSendResponse) response = doSend(mobile, code);
                return response;
        }
        /**
         * 直接的调用其他微服务
         * @param mobile
         * @param code
         * @return
         */
        public ResponseEntity<SmsSendResponse> doSend(String mobile, String
code) {
                final String sendUrl = "http://message-
service/message/sms/send";
                HttpHeaders headers = new HttpHeaders();
                headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
               MultiValueMap<String, String> map= new
LinkedMultiValueMap<String, String>();
                map. add("mobile", mobile);
                map.add("templateId", "CHECK CODE");
                map. add ("params ['code']", code);
                log. info("发送参数:{}", map);
                HttpEntity<MultiValueMap<String, String>> request = new
HttpEntity<MultiValueMap<String, String>>(map, headers);
                ResponseEntity<SmsSendResponse> response =
restTemplate.postForEntity(sendUrl, request, SmsSendResponse.class);
                return response;
        }
        /**
         * Feign声明式调用
```

```
* @param mobile
         * @return
         */
        @RequestMapping(value= {"regist/sms2"})
        public ResponseEntity<SmsSendResponse> sms2(String mobile) {
                Random random = new Random();
                int next = random.nextInt(10000000);
                String code = ""+(10000000-next);
                ResponseEntity (SmsSendResponse) response = doSendFeign (mobile,
code);
                return response;
        public ResponseEntity < SmsSendResponse > doSendFeign (String mobile, String
code) {
                SmsSendRequest request = new SmsSendRequest();
                request.setMobile(mobile);
                request.setTemplateId("CHECK CODE");
                Map<String, Object> params = new HashMap<>();
                params.put("code", code);
                request. setParams (params);
                return messageService.send(request);
        }
        @Data
        public static class SmsSendResponse {
                /**
                 * 返回消息
                 */
                private String message;
                /**
                 * 返回状态码
                 */
                private String code;
        }
2. 测试改造结果
```

运行UserApplication程序,访问

http://localhost:8082/user/regist/sms2?mobile=1234567895

