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

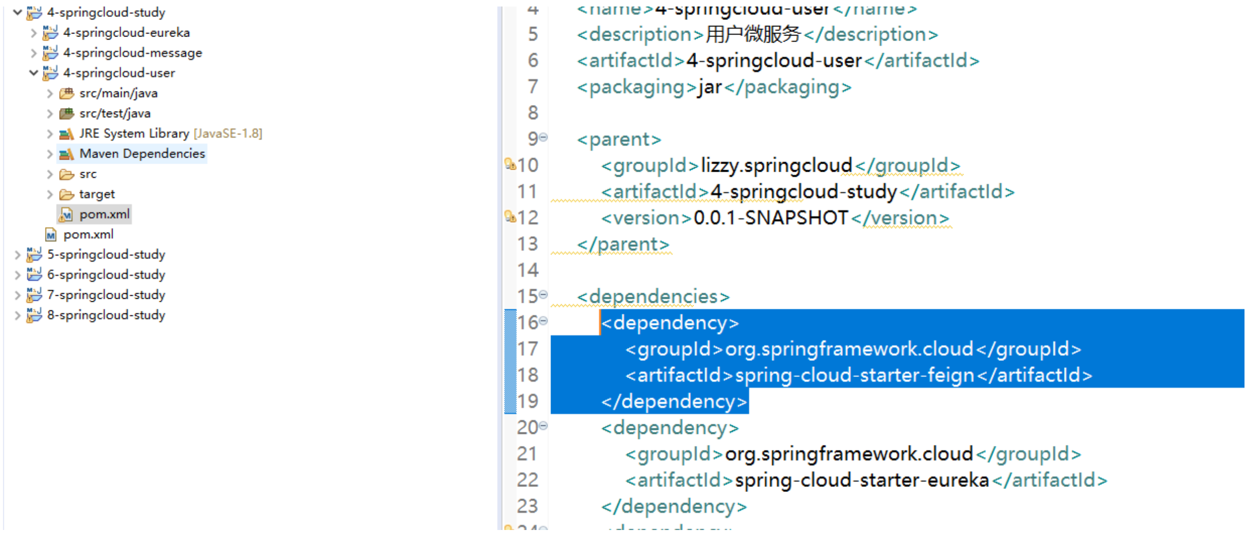
1. 服务调用者user-service

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.xml中添加

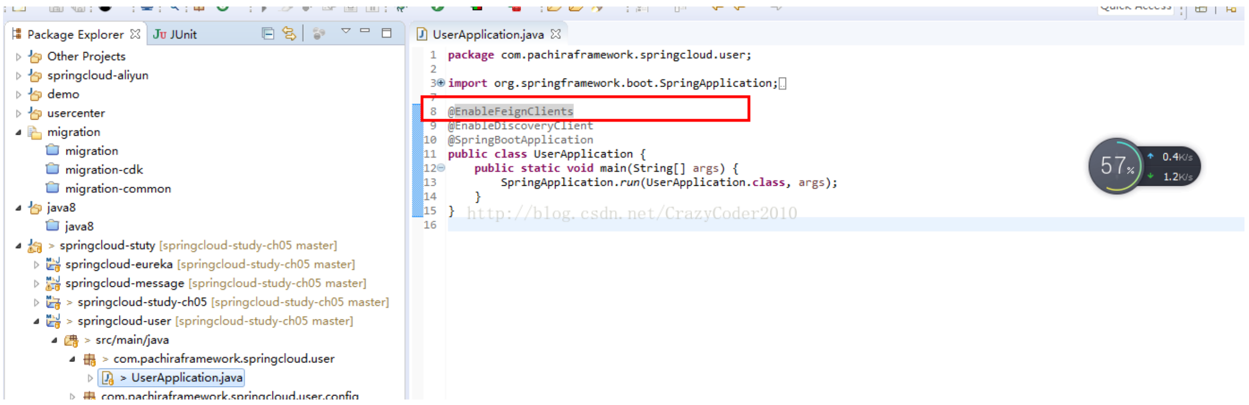
```
        <dependency>
                                <groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-starter-feign</artifactId>
                </dependency>
```



```
<name>4-springcloud-user</name>
<description>用户微服务</description>
<artifactId>4-springcloud-user</artifactId>
<packaging>jar</packaging>

<parent>
    <groupId>lizzy.springcloud</groupId>
    <artifactId>4-springcloud-study</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</parent>

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

b) 项目中启用Feign

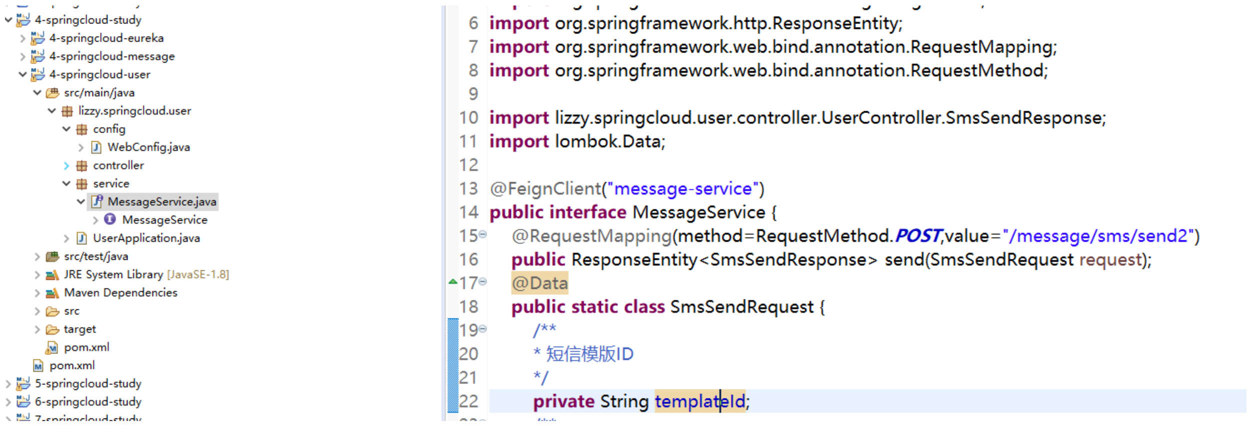
在UserApplication主函数类上添加@EnableFeignClients



```
package com.pachiraframework.springcloud.user;

import org.springframework.boot.SpringApplication;

@EnableFeignClients
@EnableDiscoveryClient
@SpringBootApplication
public class UserApplication {
    public static void main(String[] args) {
        SpringApplication.run(UserApplication.class, args);
    }
}
```

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：

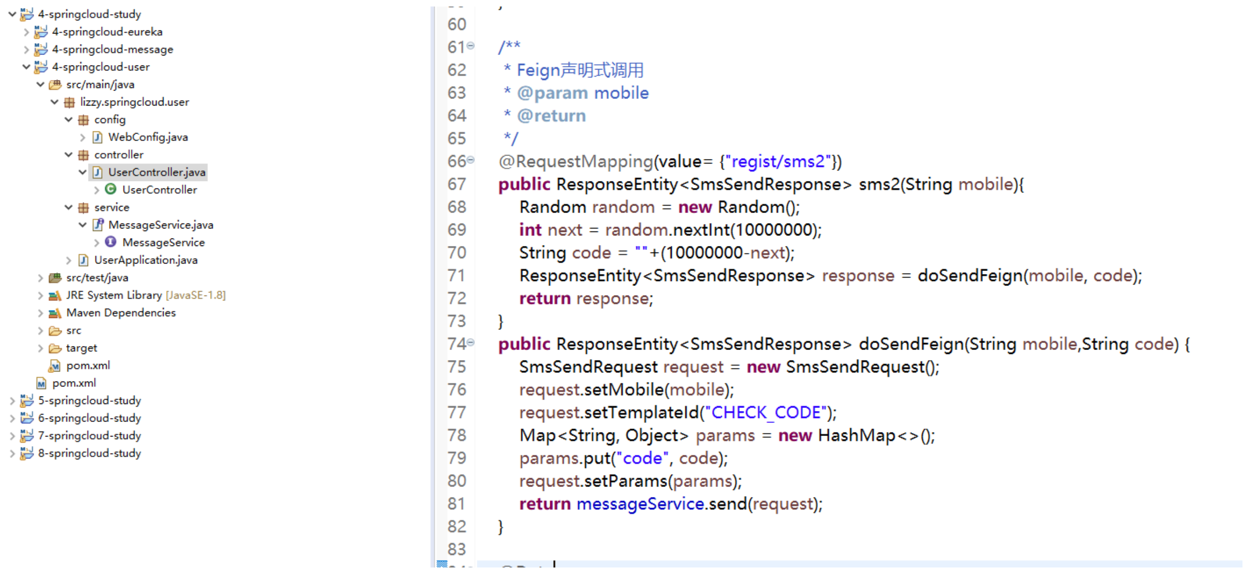


```java
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

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;
```

```java
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

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



```java

/**
 * 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);
}

```

具体代码内容如下

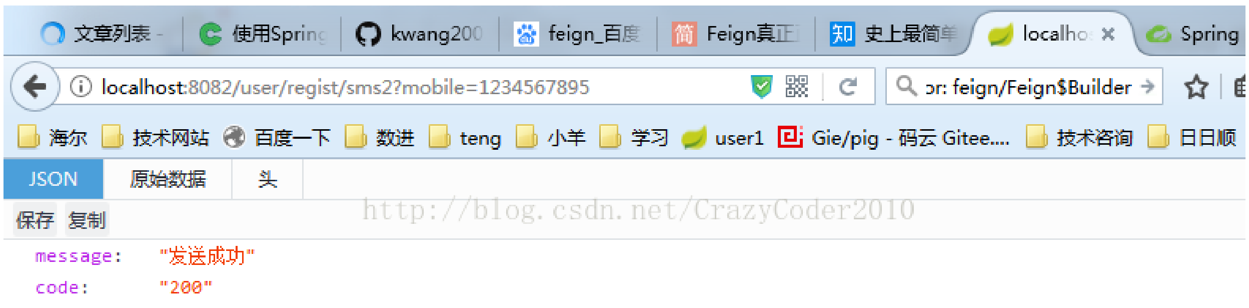
```java
package lizzy.springcloud.user.controller;

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;

@Slf4j
@RestController
@RequestMapping("/user/")
public class UserController {
	@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



```
message:   "发送成功"
code:      "200"
```