第五章 SpringCloud 使用Feign调用服务

本章节完整源代码地址：https://github.com/kwang2003/springcloud-study-ch05.git

1.项目概要介绍

在第三章和第四章的学习中，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。

本章节的代码将以第四章的完整代码为基础进行feign改造实现

https://github.com/kwang2003/springcloud-study-ch04.git

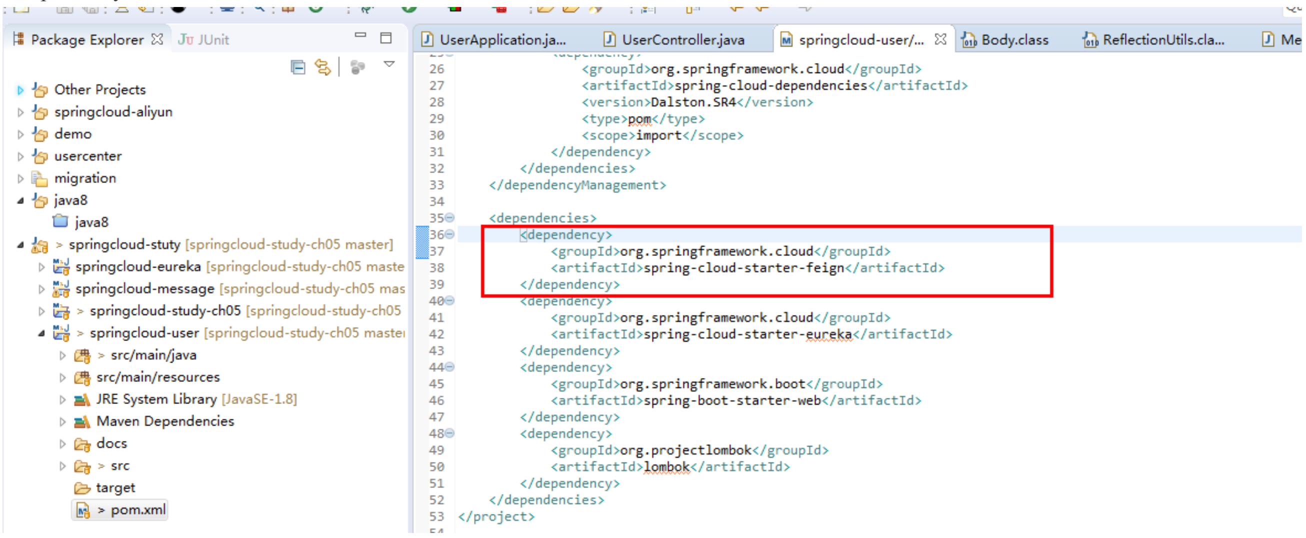
2.服务调用者user-service

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

springcloud-user工程

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

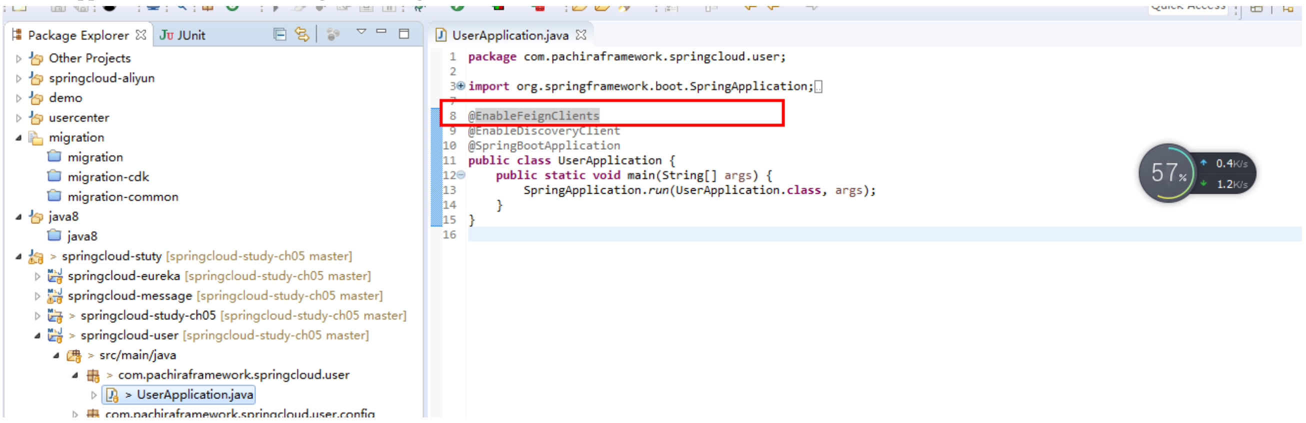


Package Explorer
JUnit
Other Projects
springcloud-aliyun
demo
usercenter
migration
java8
java8
> springcloud-stuty [springcloud-study-ch05 master]
springcloud-eureka [springcloud-study-ch05 master]
springcloud-message [springcloud-study-ch05 master]
> springcloud-study-ch05 [springcloud-study-ch05 master]
> springcloud-user [springcloud-study-ch05 master]
> src/main/java
src/main/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
docs
> src
target
> pom.xml
UserApplication.ja...
UserController.java
springcloud-user/...
Body.class
ReflectionUtils.cla...

```
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Dalston.SR4</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <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>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
    </dependencies>
</project>
```

b)项目中启用Feign

在UserApplication主函数类上添加@EnableFeignClients



Package Explorer
JUnit
Other Projects
springcloud-aliyun
demo
usercenter
migration
migration
migration-cdk
migration-common
java8
java8
> springcloud-stuty [springcloud-study-ch05 master]
springcloud-eureka [springcloud-study-ch05 master]
springcloud-message [springcloud-study-ch05 master]
> springcloud-study-ch05 [springcloud-study-ch05 master]
> springcloud-user [springcloud-study-ch05 master]
> src/main/java
> com.pachiraframework.springcloud.user
> UserApplication.java
com.pachiraframework.springcloud.user.config
UserApplication.java
57%
0.4K/s
1.2K/s

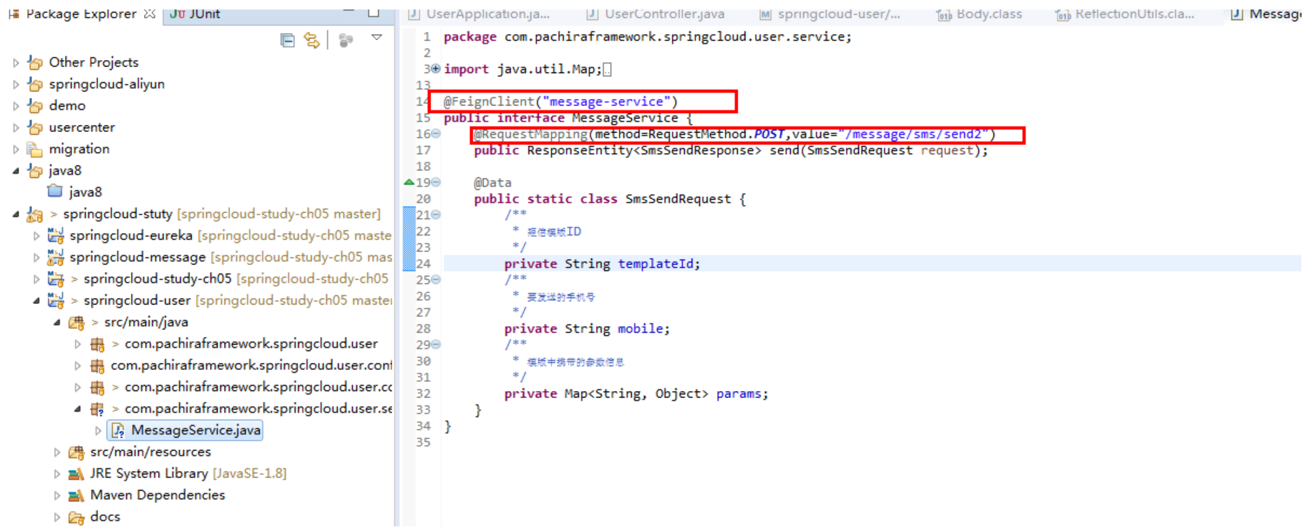
```
package com.pachiraframework.springcloud.user;

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

新增一个接口**com.pachiraframework.springcloud.user.service.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



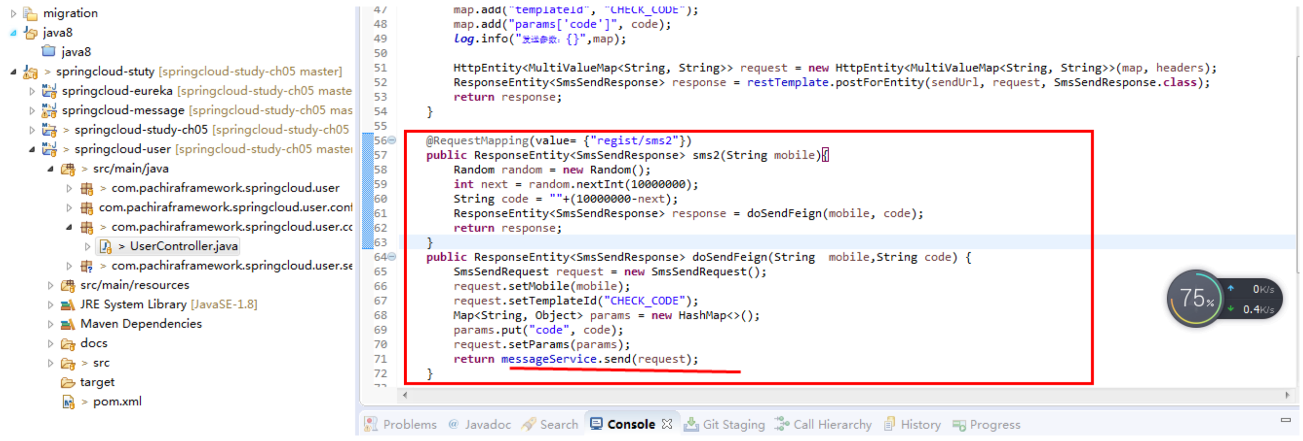
```
Other Projects
springcloud-aliyun
demo
usercenter
migration
java8
    java8
> springcloud-stuty [springcloud-study-ch05 master]
    springcloud-eureka [springcloud-study-ch05 maste
    springcloud-message [springcloud-study-ch05 mas
    > springcloud-study-ch05 [springcloud-study-ch05
    > springcloud-user [springcloud-study-ch05 maste
        > src/main/java
            > com.pachiraframework.springcloud.user
            com.pachiraframework.springcloud.user.cont
            > com.pachiraframework.springcloud.user.cc
            > com.pachiraframework.springcloud.user.se
                MessageService.java
        src/main/resources
        JRE System Library [JavaSE-1.8]
        Maven Dependencies
        docs
```

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

import java.util.Map;

@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 {
        /**
         * 短信模板ID
         */
        private String templateId;
        /**
         * 要发送的手机号
         */
        private String mobile;
        /**
         * 模板中携带的参数信息
         */
        private Map<String, Object> params;
    }
}
```

```
package com.pachiraframework.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 com.pachiraframework.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



```
migration
java8
  java8
springcloud-stuty [springcloud-study-ch05 master]
  springcloud-eureka [springcloud-study-ch05 maste
  springcloud-message [springcloud-study-ch05 mas
  > springcloud-study-ch05 [springcloud-study-ch05
  > springcloud-user [springcloud-study-ch05 maste
    > src/main/java
      > com.pachiraframework.springcloud.user
      com.pachiraframework.springcloud.user.con
      > com.pachiraframework.springcloud.user.cc
        > UserController.java
      > com.pachiraframework.springcloud.user.se
    src/main/resources
    JRE System Library [JavaSE-1.8]
    Maven Dependencies
    docs
    > src
    target
    > pom.xml
```

```java
        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;
    }

    @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);
    }
```

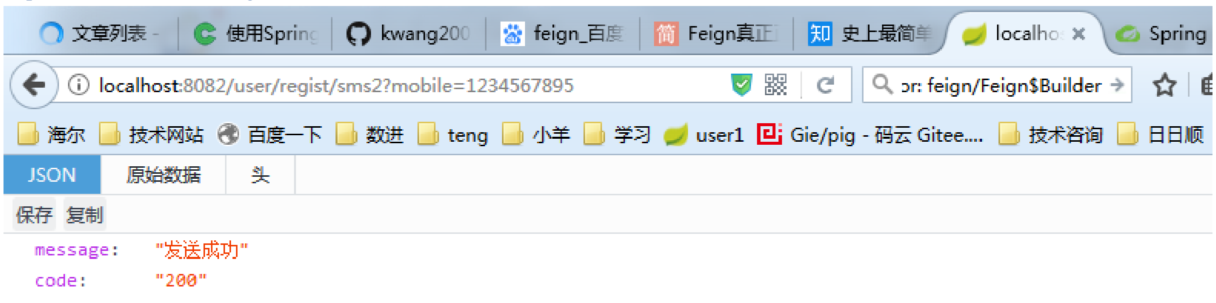
具体代码内容如下

```java
package com.pachiraframework.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 com.pachiraframework.springcloud.user.service.MessageService;
import com.pachiraframework.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;
 }

 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;
 }

 @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;
 }

}
```

3.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"
```