# Spring Cloud中feign和**RestTemplate**的示例

# 使用feign 的方式实现微服务调用

## 生产者

### 核心依赖

|  |
| --- |
| <**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**> |

### 配置文件

|  |
| --- |
| **server**:  **port**: 8090 **spring**:  **application**:  **name**: provider **eureka**:  **client**:  **service-url**:  **defaultZone**: http://jackie:123456@alpha-eureka-01:8761/eureka/,http://jackie:123456@alpha-eureka-02:8762/eureka/,http://jackie:123456@alpha-eureka-03:8763/eureka/ |

### 核心代码

|  |
| --- |
| **package** com.littlefxc.examples.cloud;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.web.bind.annotation.\*; **import** org.slf4j.Logger; **import** org.slf4j.LoggerFactory;  **import** java.util.concurrent.ConcurrentHashMap;  @EnableDiscoveryClient @RestController @SpringBootApplication **public class** ProviderApplication {   **private static** Logger *log* = LoggerFactory.*getLogger*(ProviderApplication.**class**);   ConcurrentHashMap<String, Object> **hashMap** = **new** ConcurrentHashMap(16);   @GetMapping(value = **"/get"**)  **public** String get(@RequestParam String key) {  **return "GET SUCCESS => key: "**+ key +**", value: "** + **hashMap**.get(key);  }   @PostMapping(value = **"/post"**)  **public** String post(@RequestParam String key, @RequestParam String value) {  **hashMap**.put(key, value);  **return "POST SUCCESS => key: "**+ key +**", value: "** + **hashMap**.get(key);  }   @PutMapping(value = **"/put"**)  **public** String put(@RequestParam String key, @RequestParam String value) {  **hashMap**.put(key, value);  **return "PUT SUCCESS => key: "**+ key +**", value: "** + **hashMap**.get(key);  }   @DeleteMapping(value = **"/delete"**)  **public** String delete(@RequestParam String key) {  Object remove = **hashMap**.remove(key);  **return "DELETE SUCCESS => value: "** + remove;  }   **public static void** main(String[] args) {  SpringApplication.*run*(ProviderApplication.**class**, args);  } } |

## 开放feign接口

### 接口

|  |
| --- |
| **package** com.littlefxc.examples.cloud;  **import** com.littlefxc.examples.cloud.fallback.ProviderServiceFallback; **import** org.springframework.cloud.netflix.feign.FeignClient; **import** org.springframework.web.bind.annotation.\*;  */\*\*  \** ***@author*** *fengxuechao  \** ***@date*** *2019/2/25  \*\*/* @FeignClient(value = **"provider"**, fallback = ProviderServiceFallback.**class**) **public interface** ProviderService {   @GetMapping(value = **"/get"**)  **public** String get(@RequestParam(**"key"**) String key);   @PostMapping(value = **"/post"**)  **public** String post(@RequestParam(**"key"**) String key, @RequestParam(**"value"**) String value);   @PutMapping(value = **"/put"**)  **public** String put(@RequestParam(**"key"**) String key, @RequestParam(**"value"**) String value);   @DeleteMapping(value = **"/delete"**)  **public** String delete(@RequestParam(**"key"**) String key); } |

### 熔断

|  |
| --- |
| **package** com.littlefxc.examples.cloud.fallback;  **import** com.littlefxc.examples.cloud.ProviderService; **import** org.springframework.stereotype.Component;  */\*\*  \** ***@author*** *fengxuechao  \** ***@date*** *2019/2/25  \*\*/* @Component **public class** ProviderServiceFallback **implements** ProviderService {   @Override  **public** String get(String key) {  **return "get 请求熔断降级成功"**;  }   @Override  **public** String post(String key, String value) {  **return "post 请求熔断降级成功"**;  }   @Override  **public** String put(String key, String value) {  **return "put 请求熔断降级成功"**;  }   @Override  **public** String delete(String key) {  **return "delete 请求熔断降级成功"**;  } } |

## 消费者

### 核心依赖

|  |
| --- |
| <**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**>com.littlefxc.examples</**groupId**>  <**artifactId**>provider-api</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**> </**dependency**> |

### 配置文件

|  |
| --- |
| **spring.application.name**=**consumer-feign server.port**=**8889 eureka.client.serviceUrl.defaultZone**=**http://jackie:123456@alpha-eureka-01:8761/eureka/,http://jackie:123456@alpha-eureka-02:8762/eureka/,http://jackie:123456@alpha-eureka-03:8763/eureka/ feign.hystrix.enabled**=**true** |

### 核心代码

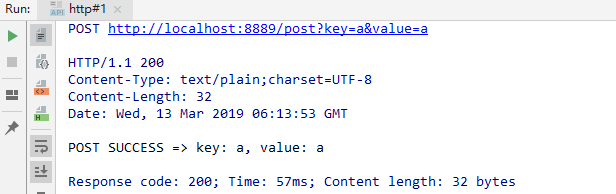
|  |
| --- |
| **package** com.littlefxc.examples.cloud;  **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication; **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient; **import** org.springframework.cloud.netflix.feign.EnableFeignClients; **import** org.springframework.web.bind.annotation.\*;  */\*\*  \** ***@author*** *fengxuechao  \*/* @EnableFeignClients @EnableDiscoveryClient @SpringBootApplication @RestController **public class** ConsumerFeignApplication {   **public static void** main(String[] args) {  SpringApplication.*run*(ConsumerFeignApplication.**class**, args);  }   @Autowired  ProviderService **service**;   @GetMapping(value = **"/get"**)  **public** String get(@RequestParam String key) {  **return service**.get(key);  }   @PostMapping(value = **"/post"**)  **public** String post(@RequestParam String key, @RequestParam String value) {  **return service**.post(key, value);  }   @PutMapping(value = **"/put"**)  **public** String put(@RequestParam String key, @RequestParam String value) {  **return service**.put(key, value);  }   @DeleteMapping(value = **"/delete"**)  **public** String delete(@RequestParam String key) {  **return service**.delete(key);  } } |

## 测试调用 get/post/put/delete

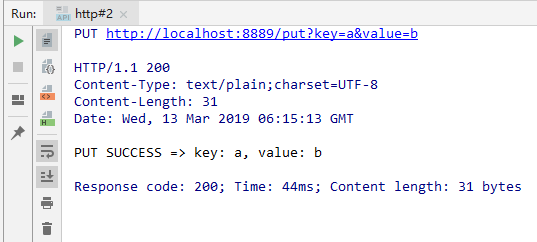
测试时为了方便起见，我采用IDEA 编辑器支持的 HTTP Request.。

|  |
| --- |
| *# For a quick start check out our HTTP Requests collection (Tools|HTTP Client|Open HTTP Requests Collection). # # Following HTTP Request Live Templates are available: # \* 'gtrp' and 'gtr' create a GET request with or without query parameters; # \* 'ptr' and 'ptrp' create a POST request with a simple or parameter-like body; # \* 'mptr' and 'fptr' create a POST request to submit a form with a text or file field (multipart/form-data);  # feign* **POST** http://localhost:8889/post?key=a&value=a  *###* **PUT** http://localhost:8889/put?key=a&value=b  *###* **GET** http://localhost:8889/get?key=a  *###* **DELETE** http://localhost:8889/delete?key=a |

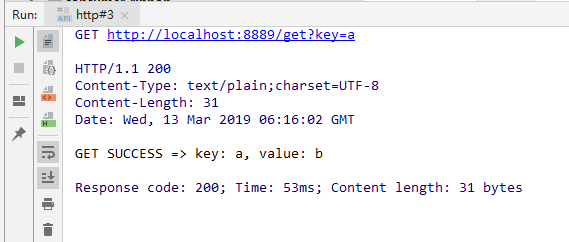
POST 请求对应结果：



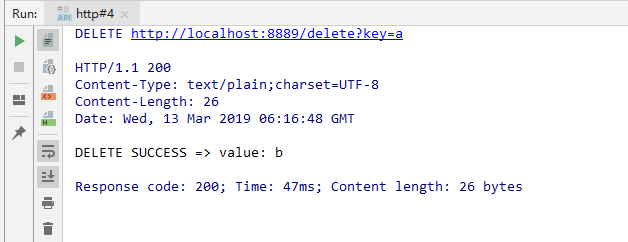
PUT 请求对应结果：



GET 请求对应结果：



DELETE 请求对应结果：



# 使用RestTemplate 的方式实现微服务调用

## 生产者

同上。

## 消费者

### 核心依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-eureka</**artifactId**></**dependency**> <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-ribbon</**artifactId**></**dependency**> <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-hystrix</**artifactId**></**dependency**> <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-web</**artifactId**> </**dependency**> |

### 配置文件

|  |
| --- |
| **spring.application.name**=**consumer-ribbon server.port**=**8888 eureka.client.serviceUrl.defaultZone**=**http://jackie:123456@alpha-eureka-01:8761/eureka/,http://jackie:123456@alpha-eureka-02:8762/eureka/,http://jackie:123456@alpha-eureka-03:8763/eureka/** |

### RestTemplate 定义

|  |
| --- |
| **package** com.littlefxc.examples.cloud;  **import** com.netflix.loadbalancer.IRule; **import** com.netflix.loadbalancer.RandomRule; **import** org.springframework.cloud.client.loadbalancer.LoadBalanced; **import** org.springframework.context.annotation.Bean; **import** org.springframework.context.annotation.Configuration; **import** org.springframework.web.client.RestTemplate;  */\*\*  \** ***@author*** *fengxuechao  \** ***@date*** *2019/2/26  \*\*/* @Configuration **public class** RestConfiguration {   */\*\*  \* Ribbon 是客户端负载均衡工具，所以在 getRestTemplate 方法上添加 @LoadBalanced 注解实现负载均衡  \** ***@return*** *\*/* @LoadBalanced  @Bean  **public** RestTemplate getRestTemplate() {  **return new** RestTemplate();  }   */\*\*  \* Ribbon 提供 IRule 接口，该接口定义了如何访问服务的策略,以下是该接口的实现类：  \* <ol>  \* <li>RoundRobinRule：轮询，默认使用的规则；</li>  \* <li>RandomRule：随机；</li>  \* <li>AvailabilityFilteringRule：先过滤由于多次访问故障而处于断路器跳闸状态以及并发连接数量超过阀值得服务，然后从剩余服务列表中按照轮询策略进行访问；</li>  \* <li>WeightedResponseTimeRule：根据平均响应时间计算所有的权重，响应时间越快服务权重越有可能被选中；</li>  \* <li>RetryRule：先按照 RoundRobinRule 策略获取服务，如果获取服务失败则在指定时间内进行重试，获取可用服务；</li>  \* <li>BestAvailableRule：先过滤由于多次访问故障而处于断路器跳闸状态的服务，然后选择并发量最小的服务；</li>  \* <li>ZoneAvoidanceRule：判断 server 所在区域的性能和 server 的可用性来选择服务器。</li>  \* </ol>  \*  \** ***@return*** *\*/* @Bean  **public** IRule iRule() {  **return new** RandomRule();  } } |

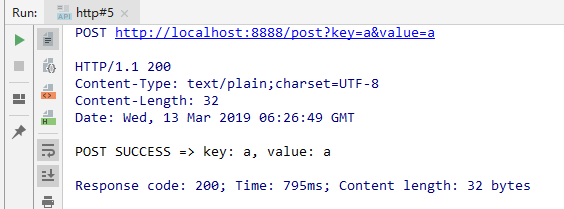
### 核心代码

|  |
| --- |
| **package** com.littlefxc.examples.cloud;  **import** com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.boot.SpringApplication; **import** org.springframework.cloud.client.SpringCloudApplication; **import** org.springframework.cloud.client.loadbalancer.LoadBalanced; **import** org.springframework.web.bind.annotation.\*; **import** org.springframework.web.client.RestTemplate;  */\*\*  \** ***@author*** *fengxuechao  \*/* @SpringCloudApplication @RestController **public class** ConsumerRibbonApplication {   @Autowired  RestTemplate **restTemplate**;   **public static void** main(String[] args) {  SpringApplication.*run*(ConsumerRibbonApplication.**class**, args);  }   @GetMapping(value = **"/get"**)  @HystrixCommand(fallbackMethod = **"getFallback"**)  **public** String get(@RequestParam String key) {  String url = **"http://provider/get?key="** + key;  **return restTemplate**.getForObject(url, String.**class**);  }   @PostMapping(value = **"/post"**)  @HystrixCommand(fallbackMethod = **"postFallback"**)  **public** String post(@RequestParam String key, @RequestParam String value) {  String url = **"http://provider/post?key={1}&value={2}"**;  **return restTemplate**.postForObject(url, **null**, String.**class**, key, value);  }   @PutMapping(value = **"/put"**)  @HystrixCommand(fallbackMethod = **"putFallback"**)  **public** String put(@RequestParam String key, @RequestParam String value) {  String url = **"http://provider/put?key="** + key + **"&value="** + value;  **restTemplate**.put(url, **null**);  **return "put 请求测试成功"**;  }   @RequestMapping(value = **"/delete"**, method = {RequestMethod.***DELETE***, RequestMethod.***GET***})  @HystrixCommand(fallbackMethod = **"deleteFallback"**)  **public** String delete(@RequestParam(**"key"**) String key) {  String url = **"http://provider/delete?key={1}"**;  **restTemplate**.delete(url, key);  **return "delete 请求测试成功"**;  }   */\*\*  \* ribbon 断路器  \*  \** ***@return*** *\*/* **public** String getFallback(String key) {  **return "get 请求熔断降级成功"**;  }   **public** String postFallback(String key, String value) {  **return "post 请求熔断降级成功"**;  }   **public** String putFallback(String key, String value) {  **return "put 请求熔断降级成功"**;  }   **public** String deleteFallback(String key) {  **return "delete 请求熔断降级成功"**;  }  } |

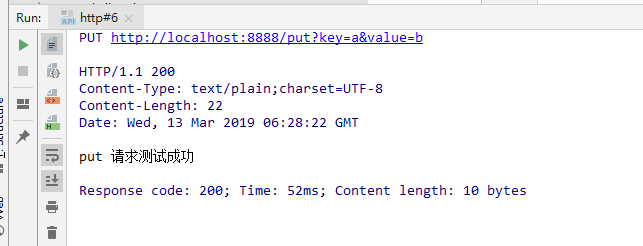
## 测试调用 get/post/put/delete

|  |
| --- |
| *# For a quick start check out our HTTP Requests collection (Tools|HTTP Client|Open HTTP Requests Collection). # # Following HTTP Request Live Templates are available: # \* 'gtrp' and 'gtr' create a GET request with or without query parameters; # \* 'ptr' and 'ptrp' create a POST request with a simple or parameter-like body; # \* 'mptr' and 'fptr' create a POST request to submit a form with a text or file field (multipart/form-data);  # ribbon* **POST** http://localhost:8888/post?key=a&value=a  *###* **PUT** http://localhost:8888/put?key=a&value=b  *###* **GET** http://localhost:8888/get?key=a  *###* **DELETE** http://localhost:8888/delete?key=a  *###* |

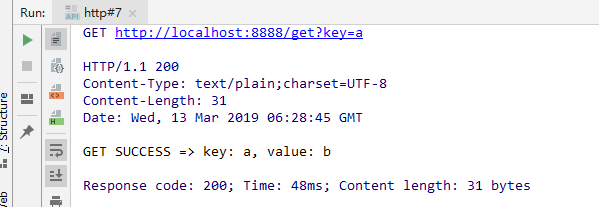
POST 请求结果：



PUT 请求结果：



GET 请求结果：



DELETE 请求结果：

