Spring Cloud Feign

1。该组件用于不同服务相互调用

假设调用端为user服务

被调用端为order服务

则除了配置信息之外应当注意：

GetMapping与RequestMapping等注解，

Spring4.3中引进了｛@GetMapping、@PostMapping、@PutMapping、@DeleteMapping、@PatchMapping｝，来帮助简化常用的HTTP方法的映射，并更好地表达被注解方法的语义。

以@GetMapping为例，Spring官方文档说：

@GetMapping是一个组合注解，是@RequestMapping(method = RequestMethod.GET)的缩写。该注解将HTTP Get 映射到特定的处理方法上。

2.usercontroller

```
@RestController
public class UserController {
    @Autowired
    private IFeignOrderService feignOrderService;
    @GetMapping("user-order")
    String userOrder(@RequestParam String name){
        return feignOrderService.feignOrder(name);
    }
}
```

IFeignOrderService 是调用order服务的interface（没有实现类）实际上它的实现是通过映射order服务来实现的

3.IFeignOrderService 的内容如下

```
@FeignClient("could-eruaka-feign-order")
public interface IFeignOrderService {

    @GetMapping("/feignOder")
    String feignOrder(@RequestParam("name")String name);

}
```

@FeignClient中的value值是被调用的服务名在该服务的配置文件中可以找到即

```
spring.application.name=could-eruaka-feign-order
server.port=1117
eureka.instance.hostname=localhost
eureka.client.serviceUrl.defaultZone=http://${eureka.instance.hostname}:1111/
```

@GETMapping中的value对应 该服务中被调用的方法的即orderController中的

```
import org.springframework.web.bind.annotation.RestController;

@RestController
public class OrderController {

    @GetMapping("/feignOder")
    String order(String name){
        return "ok"+name;
    }

}
```

如果有参数 参数必须加上@RequestParam或其他注解（RequestHeader等）声明该参数并且此时（“name”）不能省略否则会报异常

```
nested exception is java.lang.IllegalStateException: RequestParam.value() was empty on parameter 0
```

----------以上所述都是在user服务----------

4.被调用者order服务比较简单

```java
import org.springframework.boot.SpringApplication;
@EnableFeignClients
@EnableDiscoveryClient
@SpringBootApplication
public class CloudEruakaFeignOrderApplication {

    public static void main(String[] args) {
        SpringApplication.run(CloudEruakaFeignOrderApplication.class, args);
    }
}
```

```java
@RestController
public class OrderController {

    @GetMapping("/feignOder")
    String order(String name){
        return "ok"+name;
    }

}
```

被调用者就是一个普通的rest接口

另：如果需要传递的参数是对象：需要注意：

1.

User 对象的定义如下，这里省略了 getter 和 setter 函数，需要注意的是，这里必须要有 User 的默认构造函数。不然，Spring Cloud Feign 根据 JSON 字符串转换 User 对象时会抛出异常。

```java
public class User {

    private String name;
    private Integer age;

    public User() {
    }

    public User(String name, Integer age) {
        this.name = name;
        this.age = age;
    }

    // 省略 getter 和 setter

    @Override
    public String toString() {
        return "name=" + name + ", age=" +age;
    }

}
```

2. 各级接收参数都需要序列化即参数接收时用@requestBody注解 因为Feign在做数据传递时会将对象转成JSON格式，所以需要序列化

2. 1

```java
import org.springframework.beans.factory.annotation.Autowire
@RestController
public class UserController {
    @Autowired
    private IFeignOrderService feignOrderService;
    @GetMapping("user-order")
    String userOrder(@RequestParam String name){
        return feignOrderService.feignOrder(name);
    }
    @PostMapping("userBody")
    User getUser (@RequestBody User user){
        return feignOrderService.getUser(user);
    }
}
```

2. 2

```java
@FeignClient("could-eruaka-feign-order")
public interface IFeignOrderService {

    @GetMapping("/feignOder")
    String feignOrder(@RequestParam("name")String name
    @PostMapping("/userObj")
    User getUser(@RequestBody User user);
}
```

2.3

```java
@RestController
public class OrderController {

    @GetMapping("/feignOder")
    String order(@RequestParam("name")String name){
        return "ok"+name;
    }
    @PostMapping("/userObj")
    User getUser(@RequestBody User user){
        user.setAge(user.getAge()+1);
        return user;
    }

}
```

具体参数传递过程