链滴

SpringBoot 如何优雅实现远程调用

作者：sirwsl

原文链接：https://ld246.com/article/1623834948867
来源网站：链滴

废话

实在不知道做什么，还是摸会鱼写个博客吧，接下来是纯干货，个人不是很喜欢理论知识，干脆直接手就干。

不行看这文章的文末有代码，自己下载去看去

故事

问：你有两个系统A和B，这个时候A系统需要B系统的数据或者B系统需要调用A系统数据，你应该怎办

答：常用的方式无非是远程调用和多数据源。

多数据源：大概意思就是一套系统，通常来说我们会创建一个数据库进行存储数据，但有时候我们可在系统中可以看见有多数据源的配置，多数据源可以在同一套系统中进行多数据库的访问，它可以在的系统中访问其他系统配置的数据库，虽然这样做也有优点也有缺点就不一一说了，毕竟今天重点不这。

远程调用：这个东西应该都熟悉，因为在微服务不断使用下，这玩意非常常见。大概说一下就是A系需要用到B系统的数据，这个时候B系统写一个接口给A系统提供数据，A系统调用B系统的接口进行获数据。大概就是这样，接下来上干货。

事先准备

做这玩意首先，先开始一个SpringBoot的项目，就构建一个空的springbootweb项目就行，按照个喜欢可以添加点自己需要的工具。接下来开始搞。

RestTemplate实现

RestTemplate是什么看看别人怎么说吧

RestTemplate 是从 Spring3.0 开始支持的一个 HTTP 请求工具，它提供了常见的REST请求方案的版，例如 GET 请求、POST 请求、PUT 请求、DELETE 请求以及一些通用的请求执行方法 exchange 以及 execute。RestTemplate 继承自 InterceptingHttpAccessor 并且实现了 RestOperations 接，其中 RestOperations 接口定义了基本的 RESTful 操作，这些操作在 RestTemplate 中都得到了实。

怎么用直接上代码：

1、先写一下RestTemplate的配置统一处理一下。新建RestTemplateConfig

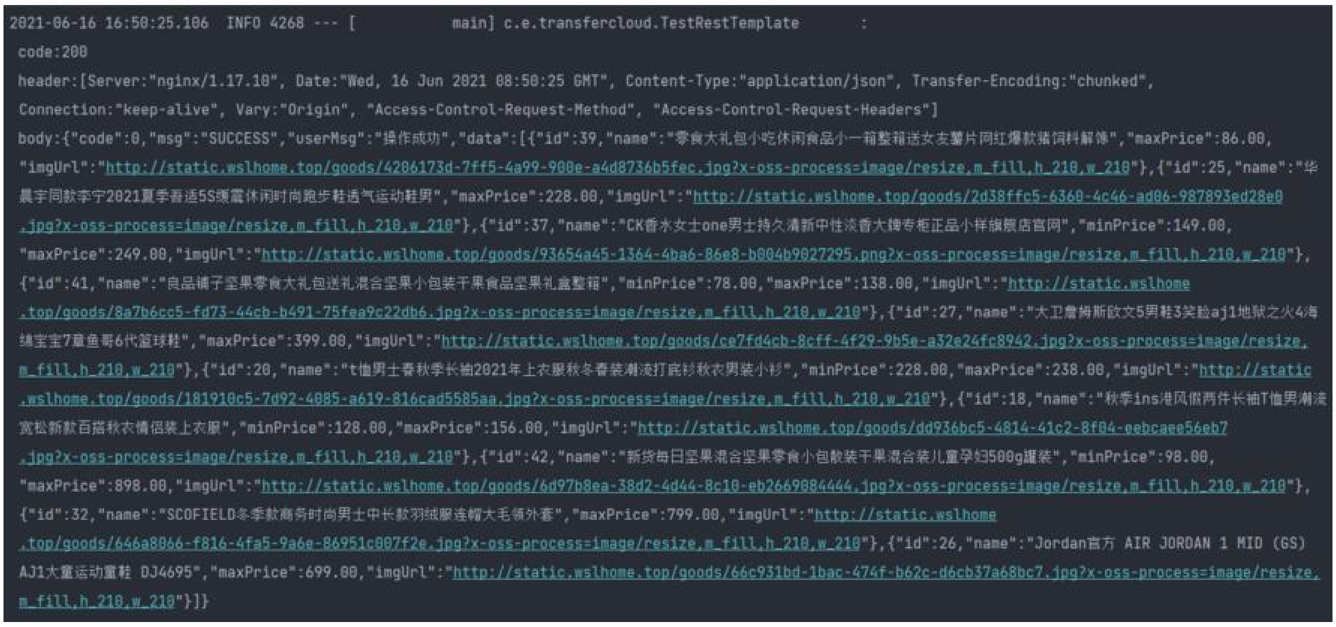
```
package com.example.transfercloud.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.client.ClientHttpRequestFactory;
import org.springframework.http.client.SimpleClientHttpRequestFactory;
import org.springframework.http.converter.StringHttpMessageConverter;
import org.springframework.web.client.RestTemplate;

import java.nio.charset.StandardCharsets;

/**
 * @author user
 */
@Configuration
public class RestTemplateConfig {
    @Bean
    public RestTemplate restTemplate(ClientHttpRequestFactory factory) {
        RestTemplate restTemplate = new RestTemplate(factory);
        // 支持中文编码
        restTemplate.getMessageConverters().set(1, new StringHttpMessageConverter(Standard
harsets.UTF_8));
        return restTemplate;
    }

    @Bean
    public ClientHttpRequestFactory simpleClientHttpRequestFactory() {
        SimpleClientHttpRequestFactory factory = new SimpleClientHttpRequestFactory();
        //单位为ms
        factory.setReadTimeout(5000);
        factory.setConnectTimeout(5000);
        return factory;
    }
}
```

2、写个Test文件测试一下，写入以下代码

```
package com.example.transfercloud;

import com.example.transfercloud.entity.TestPOJO;
import lombok.extern.slf4j.Slf4j;
import org.junit.jupiter.api.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.*;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.web.client.RestTemplate;

import java.util.HashMap;
import java.util.Map;

@SpringBootTest
@Slf4j
@RunWith(SpringJUnit4ClassRunner.class)
class TestRestTemplate {
    @Autowired
    private RestTemplate restTemplate;

    @Test
    void testRestTemplateOfGet() {
        ResponseEntity<String> responseEntity = restTemplate.getForEntity("https://test.wslho
e.top/api/getRecommendedGoods/v1?size=10", String.class);
        log.info("\n code:{}\n header:{}\n body:{}\n",responseEntity.getStatusCodeValue(),respon
eEntity.getHeaders(),responseEntity.getBody());

//        TestPOJO responseEntity0 = restTemplate.getForObject("https://test.wslhome.top/api/
etRecommendedGoods/v1?size=10", TestPOJO.class);
//        log.info("\n 实体{}",responseEntity0);
//需要创建对应的实体类，这里可以采用GesonFormat逆向生成


        ResponseEntity<String> responseEntity1 = restTemplate.getForEntity("https://test.wslh
me.top/api/getRecommendedGoods/v1?size={1}", String.class,2);
        log.info("\n code:{}\n header:{}\n body:{}\n",responseEntity1.getStatusCodeValue(),respo
seEntity1.getHeaders(),responseEntity1.getBody());

        Map<String, String> params = new HashMap<>(4);
        params.put("size","3");
        ResponseEntity<String> responseEntity2 = restTemplate.getForEntity("https://test.wslh
me.top/api/getRecommendedGoods/v1?size={size}", String.class,params);
        log.info("\n code:{}\n header:{}\n body:{}\n",responseEntity2.getStatusCodeValue(),respo
seEntity2.getHeaders(),responseEntity2.getBody());
    }

    @Test
    void testRestTemplateOfPost(){
        LinkedMultiValueMap<String, Object> paramMap = new LinkedMultiValueMap<>();
        paramMap.add("password","123");
        paramMap.add("name","234");
        paramMap.add("code","2131");
        HttpEntity<LinkedMultiValueMap<String, Object>> param = new HttpEntity<>(param
ap, null);
        ResponseEntity<String> responseEntity2 = restTemplate.postForEntity("https://test.wslh
me.top/login/user/v1",param,String.class);
```

```
        log.info("\n code:{}\n header:{}\n body:{}\n",responseEntity2.getStatusCodeValue(),respo
seEntity2.getHeaders(),responseEntity2.getBody());
    }

    @Test
    void testRestTemplateOfDelete(){
        Map<String,String> param = new HashMap<>(4);
        param.put("id","34");
        restTemplate.delete("https://test.wslhome.top/user/delCartByIds/v1", param);

        ResponseEntity<String> result = restTemplate.exchange("https://test.wslhome.top/user/
elCartByIds/v1" , HttpMethod.DELETE, null, String.class, param);
        log.info("result{}",result);

        //设置请求头
        HttpHeaders headers = new HttpHeaders();
        //如果发送的参数数据是json数据的话，需要添加如下特殊的请求头
        headers.setContentType(MediaType.APPLICATION_JSON);
        //或者headers.set("Content-Type", "application/json");
        headers.add("Authorization", "shoppingkilleyJhbGciOiJIUzI1NiJ9.eyJ2ZXJpZnk6dXNlcjpp
CI6MSwidmVyaWZ5OnVzZXI6ZmxhZyI6MTAsImV4cCI6MTYyMzc0Mjg4M30.joTLCSPy9gXlp0
BzbDlDr58hB1_wRToZaAMRVA4FqY");
        HttpEntity<Map<String, String>> httpEntity = new HttpEntity<>(param, headers);
        ResponseEntity<String> result1 = restTemplate.exchange("https://test.wslhome.top/user
delCartByIds/v1" , HttpMethod.DELETE, httpEntity, String.class);
        log.info("result{}",result1);
    }
}
```

运行结果大致如下：



```
2021-06-16 16:50:25.106  INFO 4268 --- [           main] c.e.transfercloud.TestRestTemplate       :
 code:200
 header:[Server:"nginx/1.17.10", Date:"Wed, 16 Jun 2021 08:50:25 GMT", Content-Type:"application/json", Transfer-Encoding:"chunked",
 Connection:"keep-alive", Vary:"Origin", "Access-Control-Request-Method", "Access-Control-Request-Headers"]
 body:{"code":0,"msg":"SUCCESS","userMsg":"操作成功","data":[{"id":39,"name":"零食大礼包小吃休闲食品小一箱整箱送女友薯片网红爆款猪饲料解馋","maxPrice":86.00,
 "imgUrl":"http://static.wslhome.top/goods/4206173d-7ff5-4a99-900e-a4d8736b5fec.jpg?x-oss-process=image/resize,m_fill,h_210,w_210"},{"id":25,"name":"华
 晨宇同款李宁2021夏季吾适5S缓震休闲时尚跑步鞋透气运动鞋男","maxPrice":228.00,"imgUrl":"http://static.wslhome.top/goods/2d38ffc5-6360-4c46-ad06-987893ed28e0
 .jpg?x-oss-process=image/resize,m_fill,h_210,w_210"},{"id":37,"name":"CK香水女士one男士持久清新中性淡香大牌专柜正品小样旗舰店官网","minPrice":149.00,
 "maxPrice":249.00,"imgUrl":"http://static.wslhome.top/goods/93654a45-1364-4ba6-86e8-b004b9027295.png?x-oss-process=image/resize,m_fill,h_210,w_210"},
 {"id":41,"name":"良品铺子坚果零食大礼包送礼混合坚果小包装干果食品坚果礼盒整箱","minPrice":78.00,"maxPrice":138.00,"imgUrl":"http://static.wslhome
 .top/goods/8a7b6cc5-fd73-44cb-b491-75fea9c22db6.jpg?x-oss-process=image/resize,m_fill,h_210,w_210"},{"id":27,"name":"大卫詹姆斯欧文5男鞋3笑脸aj1地狱之火4海
 绵宝宝7章鱼哥6代篮球鞋","maxPrice":399.00,"imgUrl":"http://static.wslhome.top/goods/ce7fd4cb-8cff-4f29-9b5e-a32e24fc8942.jpg?x-oss-process=image/resize,
 m_fill,h_210,w_210"},{"id":20,"name":"t恤男士春秋季长袖2021年上衣服秋冬春装潮流打底衫秋衣男装小衫","minPrice":228.00,"maxPrice":238.00,"imgUrl":"http://static
 .wslhome.top/goods/181910c5-7d92-4085-a619-816cad5585aa.jpg?x-oss-process=image/resize,m_fill,h_210,w_210"},{"id":18,"name":"秋季ins港风假两件长袖T恤男潮流
 宽松新款百搭秋衣情侣装上衣服","minPrice":128.00,"maxPrice":156.00,"imgUrl":"http://static.wslhome.top/goods/dd936bc5-4814-41c2-8f04-eebcaee56eb7
 .jpg?x-oss-process=image/resize,m_fill,h_210,w_210"},{"id":42,"name":"新货每日坚果混合坚果零食小包散装干果混合装儿童孕妇500g罐装","minPrice":98.00,
 "maxPrice":898.00,"imgUrl":"http://static.wslhome.top/goods/6d97b8ea-38d2-4d44-8c10-eb2669084444.jpg?x-oss-process=image/resize,m_fill,h_210,w_210"},
 {"id":32,"name":"SCOFIELD冬季款商务时尚男士中长款羽绒服连帽大毛领外套","maxPrice":799.00,"imgUrl":"http://static.wslhome
 .top/goods/646a8066-f816-4fa5-9a6e-86951c007f2e.jpg?x-oss-process=image/resize,m_fill,h_210,w_210"},{"id":26,"name":"Jordan官方 AIR JORDAN 1 MID (GS)
 AJ1大童运动童鞋 DJ4695","maxPrice":699.00,"imgUrl":"http://static.wslhome.top/goods/66c931bd-1bac-474f-b62c-d6cb37a68bc7.jpg?x-oss-process=image/resize,
 m_fill,h_210,w_210"}]}
```

PS：上述代码中注释部分需要创建对应的实体类

```
 TestPOJO responseEntity0 = restTemplate.getForObject("https://test.wslhome.top/api/getRe
ommendedGoods/v1?size=10", TestPOJO.class);
```

HttpClient实现

httpClient是什么，这玩意你就理解成一个工具包，你可以使用它使得用代码能够模拟浏览器发出请求

- HttpClient库实现了所有可用的HTTP方法。
- HttpClient库提供API以使用安全套接字层协议保护请求。
- 使用HttpClient，可以使用代理建立连接。

因此httpClient也被用于接口之间的调用

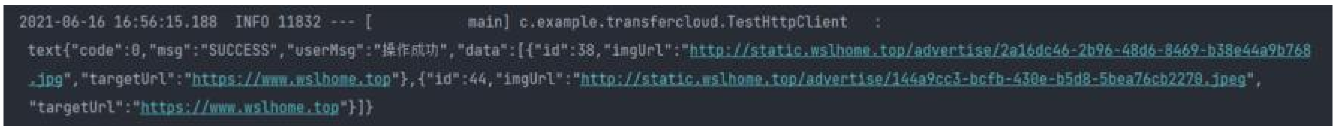
使用时候先引入依赖

```
<dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
</dependency>
```

然后直接上代码

```
package com.example.transfercloud;

import lombok.extern.slf4j.Slf4j;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.util.EntityUtils;
import org.junit.jupiter.api.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.Objects;

@Slf4j
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest
class TestHttpClient {

    /**
     * GET---无参测试
     *
     * @date 2018年7月13日 下午4:18:50
     */
    @Test
    void httpClientOfGet() {
       //创建客户端
        CloseableHttpClient httpClient = HttpClientBuilder.create().build();
        // 创建Get请求
        //HttpPost httpPost = new HttpPost("https://test.wslhome.top/api/getAdvertiseForView/
1");
        HttpGet httpGet = new HttpGet("https://test.wslhome.top/api/getAdvertiseForView/v1");
        // 响应模型
        CloseableHttpResponse response = null;
        try {
            // 执行Get请求
            response = httpClient.execute(httpGet);
            // 从响应模型中获取响应实体
            HttpEntity responseEntity = response.getEntity();
            //编码
            final String s = EntityUtils.toString(responseEntity, StandardCharsets.UTF_8);
            log.info("\n text{}",s);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                // 释放资源
                if (Objects.nonNull(httpClient)) {
                    httpClient.close();
                }
                if (Objects.nonNull(response)) {
                    response.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

实现效果如图：



```
2021-06-16 16:56:15.188  INFO 11832 --- [           main] c.example.transfercloud.TestHttpClient   : 
 text{"code":0,"msg":"SUCCESS","userMsg":"操作成功","data":[{"id":38,"imgUrl":"http://static.wslhome.top/advertise/2a16dc46-2b96-48d6-8469-b38e44a9b768
 .jpg","targetUrl":"https://www.wslhome.top"},{"id":44,"imgUrl":"http://static.wslhome.top/advertise/144a9cc3-bcfb-430e-b5d8-5bea76cb2270.jpeg",
 "targetUrl":"https://www.wslhome.top"}]}
```

PS：因为在httpClient中get和post方法差别不大，这里为了方便就只用get来举例说明，需要POS的方法，照葫芦画瓢就行

consul+feign实现

在微服务中，也常用consul+feign+hystrix来实现服务之间的调用、熔断降级，然后我们来看看怎么用。

1、在创建一个springboot项目，然后写个controller为另一个服务提供接口

```
package com.example.cloudemo.controller;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/test")
```

```
public class TestController {

    @GetMapping("/getNum")
    public String getNum() throws InterruptedException {
        Thread.sleep(10000L);
        return "测试get请求远程调用";
    }

    @PostMapping("/postNum")
    public String PostNum(){
        return "测试POST请求远程调用";
    }

    @PostMapping("/postTest")
    public String PostNums(@RequestParam Integer num){
        if (num == 0){
            return "远程调用测试,参数0";
        }else if (num == 10){
            return "远程调用测试,参数 10";
        }else {
            return "远程调用参数只支持0、10";
        }
    }
}
```

2、在两个项目中都引入依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-consul-config</artifactId>
    <version>2.2.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-consul-discovery</artifactId>
    <version>2.2.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
    <version>2.2.1.RELEASE</version>
</dependency>
```

3、两个项目中新建bootstrap.yml文件，文件中写入下面内容

```
spring:
  application:
    name: feginTest

  cloud:
    consul:
      host: 127.0.0.1
      port: 8500
      #是否启用consul
      enabled: true
      discovery:
        # 启用服务发现
        enabled: true
        # 启用服务注册
        register: true
        # 服务停止时取消注册
        deregister: true
        # 表示注册时使用IP而不是hostname
        prefer-ip-address: true
        # 执行监控检查的频率
        health-check-interval: 30s
        # 设置健康检查失败多长时间后，取消注册
        health-check-critical-timeout: 30s
        # 开启心跳检测
        heartbeat:
          enabled: true
        # 健康检查的路径
        health-check-path: /test
        # 服务注册标识，格式为：应用名称+服务器IP+端口
        instance-id: ${spring.application.name}:${spring.cloud.client.ipaddress}:${server.port}
        config:
          # 启用consul的配置中心功能，默认是true
          enabled: true
          # 表示consul上面文件的格式 有四种 YAML PROPERTIES KEY-VALUE FILES，默认是KEY-VA
UE
          format: YAML
          #配置基本文件，默认值config
          prefix: ${spring.application.name}
          #表示开发环境：dev/test/prepped，生产环境独立部署consul服务器
          default-context: dev
          #表示consul上面的配置文件名，每个开发人员各自管理自己的配置文件
          data-key: infokey
          # watch选项为配置监视功能，主要监视配置的改变
          watch:
            enabled: true
            delay: 10000
            wait-time: 30


feign:
  hystrix:
    enabled: true

hystrix:
  shareSecurityContext: true
  command:
    default:
      circuitBreaker:
        requestVolumeThreshold: 1
        sleepWindowInMilliseconds: 15000
        forceOpen: false
        forceClosed: false
      execution:
        isolation:
          thread:
            timeoutInMilliseconds: 5000

ribbon:
  ConnectTimeout: 4000
  ReadTimeout: 4000
```

PS1:spring.application.name是服务名称，两个服务应该采用不通的名称。对了application.yml中端口号也应该改为不同的端口号。

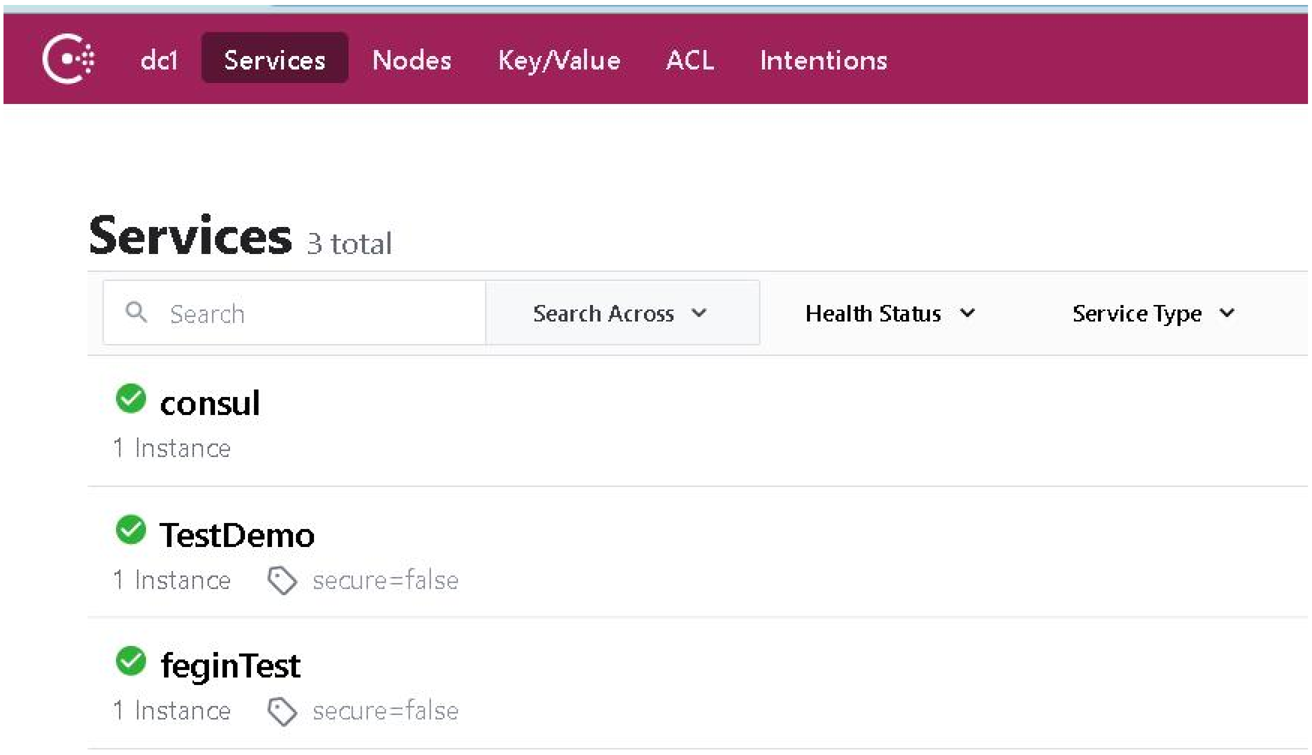
PS：我当时因为没有新建，直接在application.yml中写，然后访问注册中心一直都是localhost，论填什么地址都不会变，然后无奈了，也找不到解决办法，只有新建的bootstrap.yml然后就正常了。

4、修改启动类

在启动类Application.java的类上加入注解

```
@EnableDiscoveryClient
@SpringBootApplication
@EnableFeignClients
```

5、依次启动两个项目，然后去你的consul看看服务启动是否正常。你会看到个截图的样子就是正常的。



Services 3 total

consul
1 Instance

TestDemo
1 Instance secure=false

feginTest
1 Instance secure=false

6、开始写消费端的接口调用

（1）创建一个接口类用于远程调用，内容如下

```
package com.example.transfercloud.rpc;

import com.example.transfercloud.rpc.callBack.TestCloudCallBackFactory;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@FeignClient(value = "TestDemo",fallbackFactory = TestCloudCallBackFactory.class)
public interface TestCloud {

    @GetMapping("/test/getNum")
    String getTestNum();

    @PostMapping("/test/postNum")
    String getTestNumByPost();

    @PostMapping("/test/postTest")
    String getTestNumByPostParam(@RequestParam Integer num);

}
```

（2）创建callBack文件实现接口

```
package com.example.transfercloud.rpc.callBack;

import com.example.transfercloud.rpc.TestCloud;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;

@Slf4j
@Component
public class TestCloudCallBack implements TestCloud {
    @Override
    public String getTestNum() {
        log.error("调用失败，服务降级");
        return null;
    }

    @Override
    public String getTestNumByPost() {
        log.error("调用失败，服务降级");
        return null;
    }

    @Override
    public String getTestNumByPostParam(Integer num) {
        log.error("调用失败，服务降级");
        return null;
    }
}
```

（3）创建FallbackFactory文件处理服务不可用情况

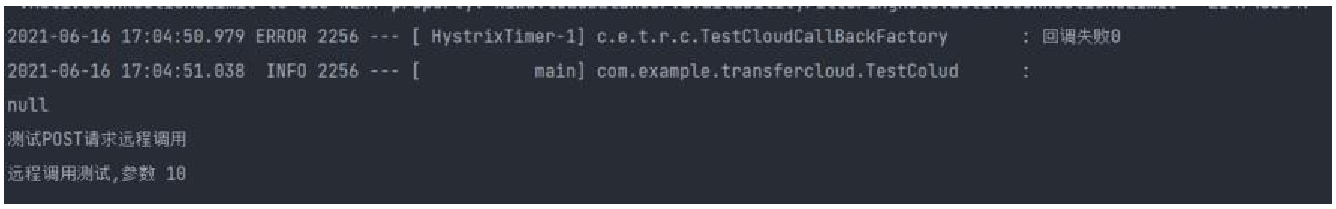
```
package com.example.transfercloud.rpc.callBack;

import com.example.transfercloud.rpc.TestCloud;
import feign.hystrix.FallbackFactory;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;

@Slf4j
@Component
public class TestCloudCallBackFactory implements FallbackFactory<TestCloud> {

    @Override
    public TestCloud create(Throwable throwable) {
        return new TestCloud() {
            @Override
            public String getTestNum() {
                log.error("回调失败0");
                return null;
            }

            @Override
            public String getTestNumByPost() {
                log.error("回调失败1");
                return null;
            }

            @Override
            public String getTestNumByPostParam(Integer num) {
                log.error("回调失败2");
                return null;
            }
        };
    }
}
```

6、编写测试类看看是否成功

```
package com.example.transfercloud;

import com.example.transfercloud.rpc.TestCloud;
import lombok.extern.slf4j.Slf4j;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import javax.annotation.Resource;

@Slf4j
@SpringBootTest
@RunWith(SpringJUnit4ClassRunner.class)
```

```java
public class TestColud {

    @Resource
    private TestCloud testCloud;
    @Test
    public void test(){
        final String testNum = testCloud.getTestNum();
        final String testNumByPost = testCloud.getTestNumByPost();
        final String testNumByPostParam = testCloud.getTestNumByPostParam(10);
        log.info("\n{}\n{}\n{}",testNum,testNumByPost,testNumByPostParam);
    }
}
```

7、结果



```
2021-06-16 17:04:50.979 ERROR 2256 --- [ HystrixTimer-1] c.e.t.r.c.TestCloudCallBackFactory       : 回调失败0
2021-06-16 17:04:51.038  INFO 2256 --- [           main] com.example.transfercloud.TestColud      :
null
测试POST请求远程调用
远程调用测试,参数 10
```

可以看到第一个接口被降级因为Thread.sleep(10000L);的原因，第二三个接口调用成功

文中代码

点击下载 demo.zip

点击下载 demo.zip

点击下载 demo.zip

彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋
蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋彩蛋
蛋彩蛋彩蛋彩蛋