<http://blog.csdn.net/ThinkingLink/article/details/45366777>

SpringMVC RestTemplate的几种请求调用

1.用统一的方法模板进行四种请求：**POST,PUT,DELETE,GET**

restTemplate.exchange(

String url,

HttpMethod method,

HttpEntity requestEntity,

Class responseType,

Object uriVariables[]

)

说明：

1）url: 请求地址；

2）method: 请求类型(如：POST,PUT,DELETE,GET)；

3）requestEntity: 请求实体，封装请求头，请求内容

4）responseType: 响应类型，根据服务接口的返回类型决定

5）uriVariables: url中参数变量值

例如：

(1)POST请求

String reqJsonStr = "{\"code\":\"testCode\", \"group\":\"testGroup\",\"content\":\"testContent\", \"order\":1}";

HttpHeaders headers = new HttpHeaders(); headers.setContentType(MediaType.APPLICATION\_JSON);

HttpEntity<String> entity = new HttpEntity<String>(reqJsonStr,headers);

ResponseEntity<Map> resp = restTemplate.exchange(DIC\_DATA\_URL, HttpMethod.POST, entity, Map.class);

(2)PUT请求

String reqJsonStr = "{\"id\":227,\"code\":\"updateCC\", \"group\":\"UPDATE\",\"content\":\"updateCT\", \"order\":9}";

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

HttpEntity<String> entity = new HttpEntity<String>(reqJsonStr,headers);

ResponseEntity<Map> resp = restTemplate.exchange(DIC\_DATA\_URL, HttpMethod.PUT, entity, Map.class);

(3)DELETE请求

ResponseEntity<Map> resp = restTemplate.exchange(DIC\_DATA\_URL + "?id={id}", HttpMethod.DELETE, null, Map.class, 227);

(4)GET请求

ResponseEntity<String> results = restTemplate.exchange(url,HttpMethod.GET, null, String.class, params);

2.用各种请求对应的专一接口

(1)POST请求

postForObject(String url, Object request, Class responseType, Object uriVariables[]):

返回数据对象Object，例如：

DicData data = new DicData();

data.setCode("cd123"); data.setGroup("TEST"); data.setContent("测试数据"); data.setOrder(5);

DicData obj = restTemplate.postForObject(DIC\_DATA\_URL, data, DicData.class);

> 或者

postForEntity:(String url, Object request, Class responseType, Object uriVariables[])

返回封装了数据对象的ResponseEntity对象，例如：

DicData data = new DicData();

data.setCode("cd123"); data.setGroup("TEST"); data.setContent("测试数据"); data.setOrder(5);

ResponseEntity<Map> respEntity = restTemplate.postForEntity(DIC\_DATA\_URL, data, Map.class);

> (2)PUT请求

put(String url, Object request, Object urlVariables[])

例如：

DicData data = new DicData();

data.setId(226L); data.setCode("updateCode"); data.setGroup("UPDATE");

data.setContent("测试数据"); data.setOrder(9);

restTemplate.put(DIC\_DATA\_URL, data);

> (3)DELETE请求

delete(String url, Object urlVariables[])

例如：

restTemplate.delete(DIC\_DATA\_URL + "?id={id}", 222);

> (4)GET请求

getForObject(String url, Class responseType, Object urlVariables[])：

返回请求的结果对象，例如

Order o = restTemplate.getForObject(Constants.SERVER\_URL+"/order?orderCode={orderCode}",

Order.class,order.getOrderCode());

getForEntity(String url, Class responseType, Object urlVariables[])：

返回封装了数据对象的ResponseEntity对象，例如：

ResponseEntity<EBTUser> ebtuserResponse = restTemplate.getForEntity(url,EBTUser.class);

EBTUser user = ebtuserResponse.getBody();

3.GET请求,要返回一些复合数据类型时的处理

> (1)返回List类型数据

DicData[] dicResult = restTemplate.getForObject( Constants.SERVER\_URL + "/dicDatas/dicData?"

+ "group={group}", DicData[].class, group);

List<DicData> list = Arrays.asList(dicResult);

或者

// pass generic information to resttemplate; ParameterizedTypeReference为spring3.2版本后引进的类

ParameterizedTypeReference<List<DicData>> responseType = new ParameterizedTypeReference<List<DicData>>();

ResponseEntity<List<DicData>> resp = restTemplate.exchange(Constants.SERVER\_URL + "/dicDatas/dicData?group={group}",

HttpMethod.GET, null, responseType);

List<DicData> list = resp.getBody();

> (2)返回属性中有范型数据的复合对象

比如，分页对象

ResponseEntity<String> results = restTemplate.exchange(url,HttpMethod.GET, null, String.class, params);

// 借助com.fasterxml.jackson.databind.ObjectMapper 对象来解析嵌套的json字符串

ObjectMapper mapper = new ObjectMapper(); mapper.configure(DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES, false);

PageInfo<Product> page = mapper.readValue(results.getBody(), new TypeReference<PageInfo<Product>>() { });