# 一、使用RestTemplate

## 1、get请求

string类型

@Test

public void getCurrentVersion() {

RestTemplate restTemplate = new RestTemplate();

restTemplate.getMessageConverters().add(new StringHttpMessageConverter(Charset.forName("UTF-8")));

String version = restTemplate.getForObject("http://localhost/application/versions/current", String.class);

System.*out*.println(version);

}

json类型

@Test

public void getVersions() {

RestTemplate restTemplate = new RestTemplate();

restTemplate.getMessageConverters().add(new MappingJackson2HttpMessageConverter());

Gson gson = new Gson();

@SuppressWarnings("unchecked")

List<Object> list = restTemplate.getForObject("http://localhost/application/versions", List.class);

for (Object o : list) {

ApplicationDmo a = gson.fromJson(o.toString(), ApplicationDmo.class);

System.*out*.println(a.getVersion());

}

}

## 2、post请求

@Test

public void postLogin() throws JSONException {

RestTemplate restTemplate = new RestTemplate();

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.*APPLICATION\_JSON\_UTF8*);

headers.add("Accept", MediaType.*APPLICATION\_JSON\_VALUE*);

HttpEntity<String> formEntity = new HttpEntity<String>(headers);

ResultModel rm = restTemplate.postForObject("http://localhost/user/login?phone={phone}&password={password}",

formEntity, ResultModel.class, "17702525841", "000000");

System.*out*.println(rm.getData());

}

Controller层的参数支持字符串，对象，Map，一定要加@RequestBody

@RequestMapping("postMsg")

public Callable<ResultModel> post(@RequestBody String msg) {

return () -> {

return new ResultModel(true, msg, null);

};

}

@RequestMapping(value="postMap")

public Callable<ResultModel> post(@RequestBody Map<String,Object> map) {

return () -> {

return new ResultModel(true, null, map);

};

}

Client层

@Test

public void postMsg() {

RestTemplate restTemplate = new RestTemplate();

restTemplate.getMessageConverters().add(new MappingJackson2HttpMessageConverter());

restTemplate.getMessageConverters().add(new StringHttpMessageConverter(Charset.forName("UTF-8")));

ResultModel rm = restTemplate.postForObject("http://localhost/testAsync/postMsg", "hello", ResultModel.class);

System.*out*.println(rm.getMsg());

}

@Test

public void postMap() throws JSONException {

RestTemplate restTemplate = new RestTemplate(new SimpleClientHttpRequestFactory());

restTemplate.getMessageConverters().add(new MappingJackson2HttpMessageConverter());

restTemplate.getMessageConverters().add(new StringHttpMessageConverter());

Map<String,Object> map=new HashMap<>();

map.put("msg", "hello world");

ResultModel rm = restTemplate.postForObject("http://localhost/testAsync/postMap", map,

ResultModel.class);

System.*out*.println(rm.getData());

}

@Test

public void postMap2() throws JSONException {

RestTemplate restTemplate = new RestTemplate(new SimpleClientHttpRequestFactory());

HttpHeaders headers = new HttpHeaders();

MediaType type = MediaType.*parseMediaType*("application/json; charset=UTF-8");

headers.setContentType(type);

headers.add("Accept", MediaType.*APPLICATION\_JSON*.toString());

JSONObject json = new JSONObject();

json.put("phone", "17702525841");

json.put("password", "000000");

HttpEntity<String> formEntity = new HttpEntity<String>(json.toString(), headers);

ResultModel rm = restTemplate.postForObject("http://localhost/testAsync/postMap", formEntity,

ResultModel.class);

System.*out*.println(rm.getData());

}

# 二、使用okhttp

1. 添加依赖

<dependency>

<groupId>com.squareup.okhttp</groupId>

<artifactId>mockwebserver</artifactId>

<version>2.2.0</version>

</dependency>

1. 在类中导入

import okhttp3.\*;

1. 在方法中写

@RequestMapping(value = "getSmsCode", method = RequestMethod.*POST*)

public Object getSmsCode(String phone) {

final String code = this.smsCodeBo.getSmsCode();

final ResultModel result= new ResultModel();

OkHttpClient client = new OkHttpClient();

MediaType mediaType = MediaType.*parse*("application/json");

RequestBody body = RequestBody.*create*(mediaType,

"{\"mobile\":\"" + phone + "\",\"temp\_id\":25849,\"temp\_para\":{\"code\":\"" + code + "\"}}");

Request request = new Request.Builder().url("https://api.sms.jpush.cn/v1/messages").post(body)

.addHeader("content-type", "application/json").addHeader("authorization", keys.getJpushAuthorization())

.build();

// 异步请求

client.newCall(request).enqueue(new Callback() {

@Override

public void onResponse(Call call, Response response) throws IOException {

String msg\_id = response.body().string();

System.*out*.println("isSuccessful: " + response.isSuccessful());

System.*out*.println("response: " + msg\_id);

// 请求成功

if (msg\_id.contains("msg\_id")&&response.isSuccessful()) {

// 如果发送成功就把验证码和手机号存进数据库

SmsCodeDmo codeDmo = UserController.this.smsCodeDao.findByPhone(phone);

if (codeDmo == null) {

codeDmo = new SmsCodeDmo();

codeDmo.setPhone(phone);

}

codeDmo.setSmscode(code);

codeDmo.setExpiration(new Timestamp(System.*currentTimeMillis*() + 30 \* 60 \* 1000));

UserController.this.smsCodeDao.save(codeDmo);

} else {

// 失败，返回给客户端发送失败

result.setSuccess(false);

}

}

@Override

public void onFailure(Call call, IOException t) {

// 失败，就返回告诉客户端发送失败

result.setSuccess(false);

result.setMsg(t.getMessage());

*log*.error("发送验证码失败:" + t.getMessage());

}

});

return result;

}