

Postman实现接口自动化测试

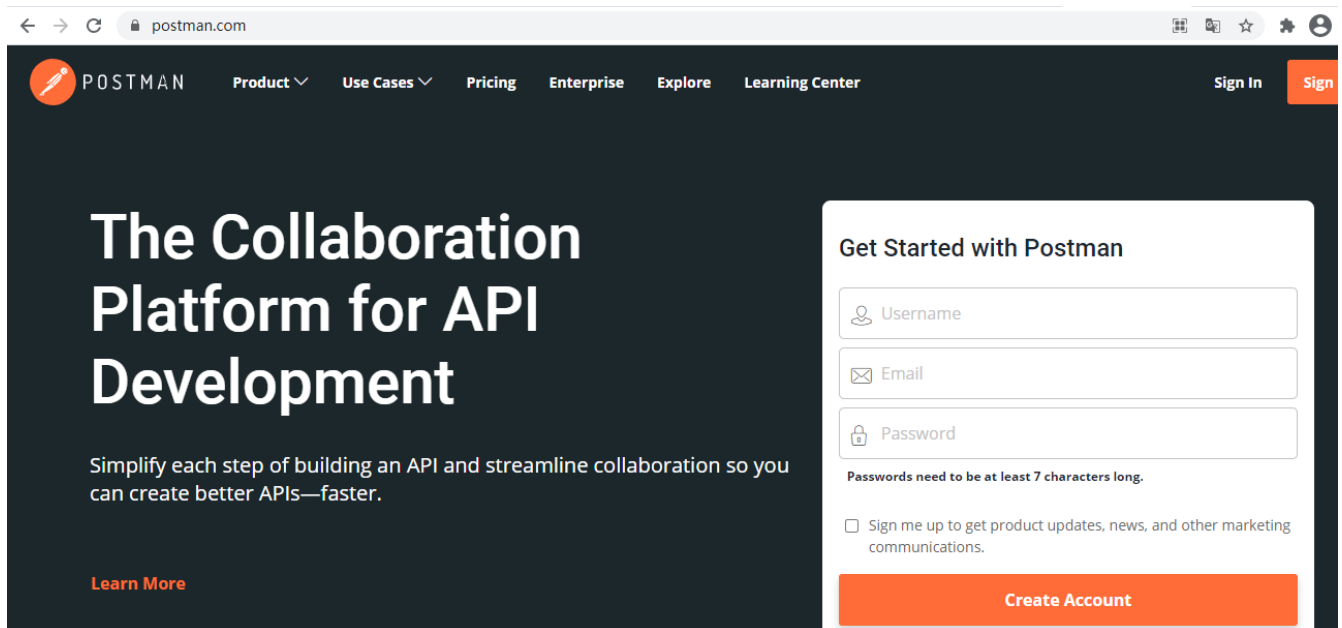
1 Postman介绍

Postman是一个用于调试HTTP请求的工具,它提供了友好的界面帮助分析、构造HTTP请求,并分析响应数据。

实际工作中,开发和测试基本上都有使用Postman来进行接口调试工作。

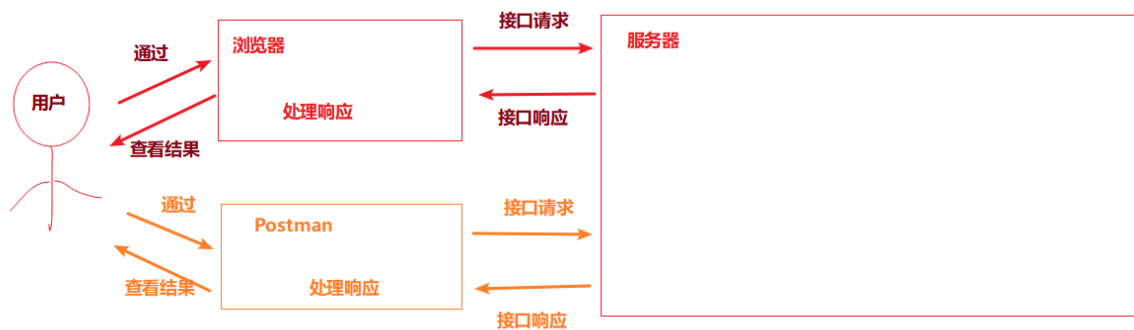
有一些其他流程的工具,也是模仿的Postman的风格进行接口测试工具设计。

官网: <https://www.postman.com/>



2 Postman工作原理

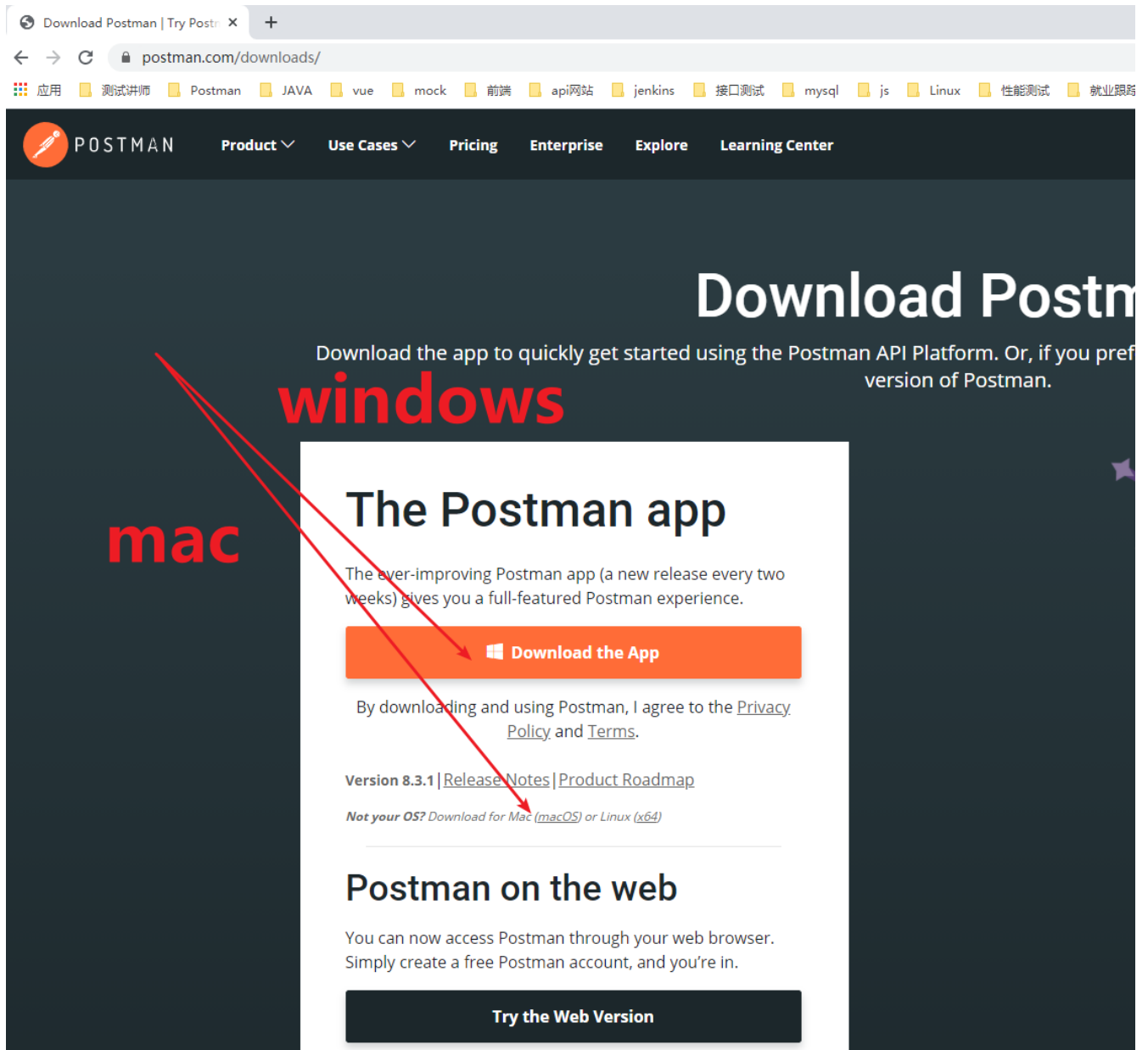
Postman近似于一个浏览器,它可以模拟浏览器、APP原生等客户端向服务器发送接口请求,并获取接口的响应数据。



3 Postman入门

3.1 Postman安装

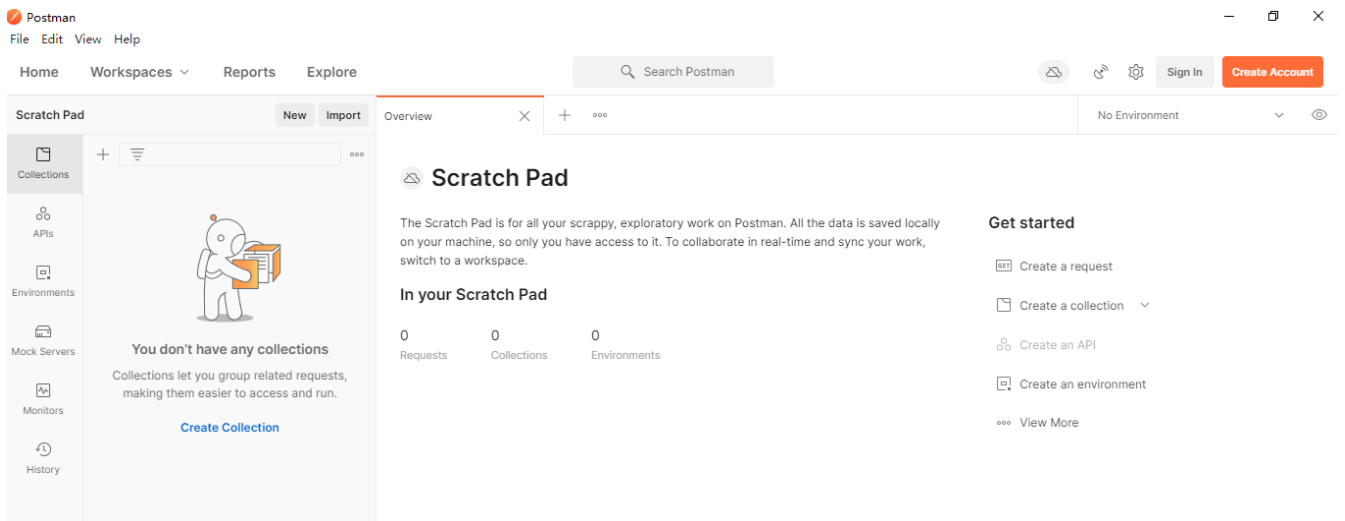
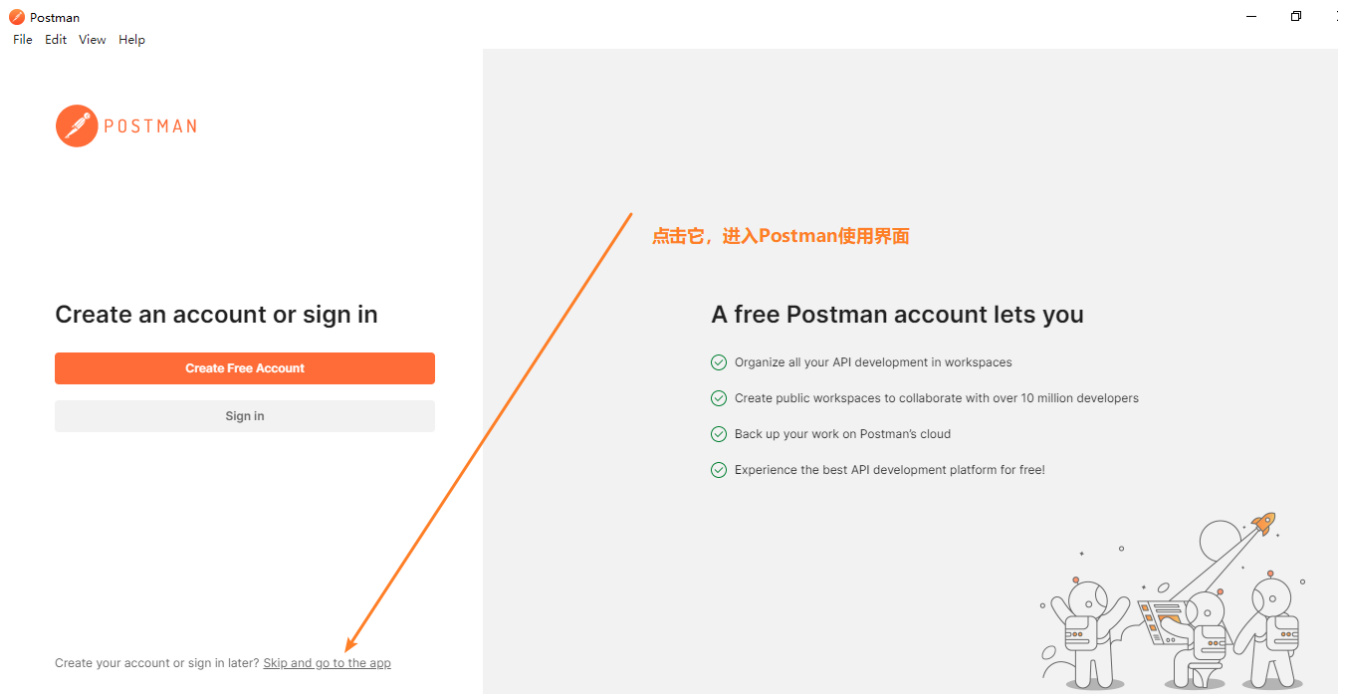
下载Postman:



安装Postman:

- windows:
双击下载下来的安装包，会全自动进行安装，安装时，建议连接外网。
- mac
双击下载下来的安装包，然后把postman拖动到应用(Application)中

安装好后的界面如图所示:



3.2 Postman基本用法

Postman能够按照接口文档的规定设置接口的请求方法、URL、请求头、请求体来完成请求数据的构造；然后查看响应数据，完成对响应数据的测试。

3.2.1 Postman请求百度搜索接口

百度搜索接口简要文档：

请求方法：GET

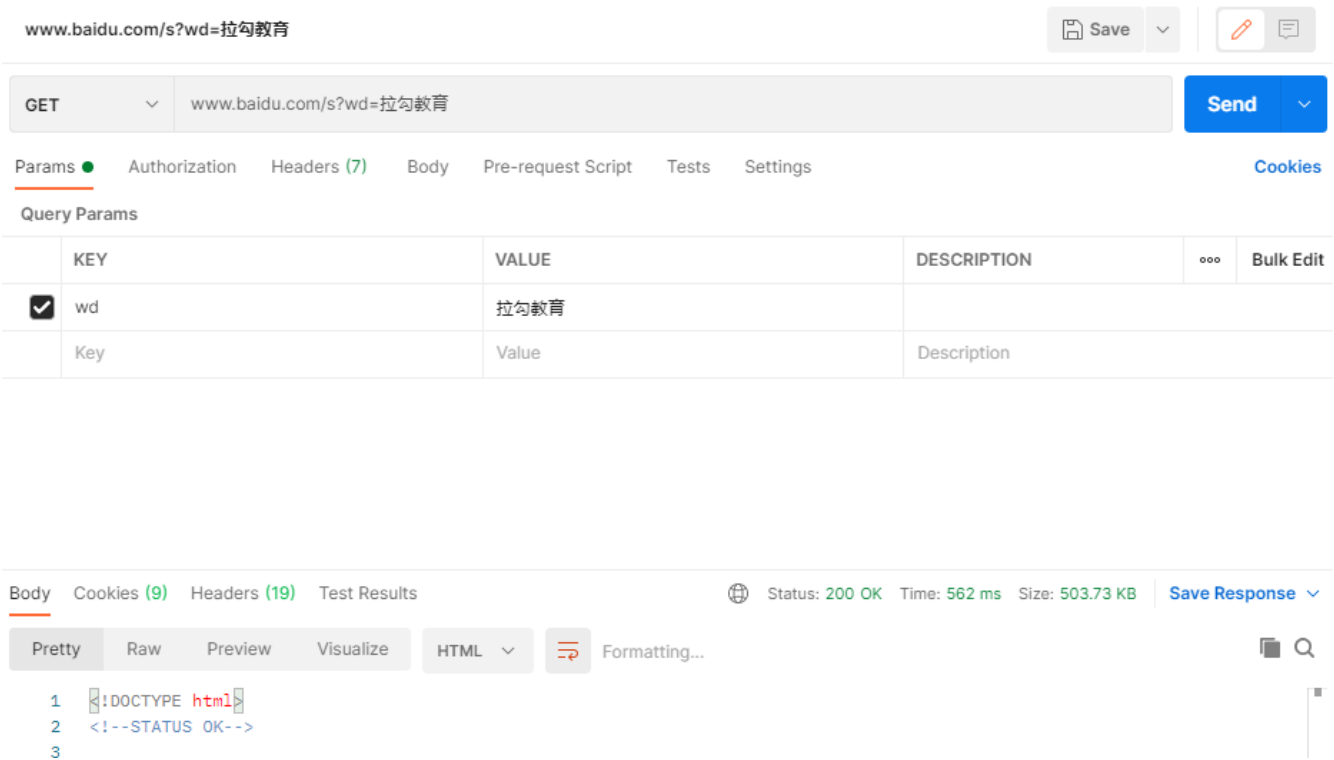
URL：<http://www.baidu.com/s?wd=拉勾教育>

请求头：无要求，默认即可

请求体：无

操作步骤：

- 第一步：打开Postman，添加一个请求
- 第二步：设置请求方法
- 第三步：设置URL
- 第四步：发送请求
- 第五步：查看结果



3.2.2 Postman请求拉勾教育搜索课程接口

拉勾教育搜索接口简要文档：

请求方法：POST

URL：/ssm_web/course/findAllCourse

请求头：

Headers	参数值
Content-Type	application/json
Cookie	{"JSESSIONID":"xxx"}

请求体：

```
{"courseName": "武林外传"}
```

操作步骤：

第一步：打开Postman，添加一个请求

第二步：设置请求方法

第三步：设置URL

第四步：设置请求头

第五步：设置请求体

第六步：查看结果

The screenshot shows the Postman interface for a POST request to `http://www.edu2.com:8080/ssm_web/course/findAllCourse`. The request headers include `Content-Type: application/json` and a `Cookie` with a session ID. The response body is a JSON object indicating success and returning course details.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> Cookie	{ "JSESSIONID": "FA36D8F677B83200C94D8B4..." }	
Key	Value	Description

```
1 {
2   "success": true,
3   "state": 200,
4   "message": "响应成功",
5   "content": [
6     {
7       "id": 31,
8       "courseName": "武林外传 第一部",

```

3.2.3 Postman请求添加课程接口

添加课程接口简要文档：

请求方法：POST

URL：/ssm_web/course/saveOrUpdateCourse

请求头：

Headers	参考值
Cookies	{"JSESSIONID":"xxx"}
Content-Type	application/json

请求体：

```
{
  "brief": "简介",
  "courseDescriptionMarkDown": "",
  "courseImgUrl": "",
  "courseListImg": "",
  "courseName": "测试添加课程111",
  "discounts": 1,
  "discountsTag": "泰坦",
  "id": 33,
  "previewFirstField": "这是一个测试用视频",
  "previewSecondField": "这是一个测试用视频",
  "price": 1,
  "sales": 1,
  "sortNum": 0,
  "status": null,
  "teacherName": "泰坦",
  "position": "讲师",
  "description": "讲师简介",
  "teacherDTO": Object{...},
  "activityCourse": false,
  "activityCourseDTO": {

  },
  "currentPage": 1,
  "pageSize": 10
}
```

操作步骤：

第一步：打开Postman，添加一个请求

第二步：设置请求方法

第三步：设置URL

第四步：设置请求头

第五步：设置请求体

第六步：查看结果3

3.2.4 Postman请求拉勾商城登陆接口

拉勾商城登陆接口简要文档：

请求方法：POST

URL：/index.php?m=Home&c=User&a=do_login

请求头：

Headers	参考值
Content-Type	application/x-www-form-urlencoded

请求体：username=xxx&password=xxx&verify_code=xxx

操作步骤：

第一步：打开Postman，添加一个请求

第二步：设置请求方法

第三步：设置URL

第四步：设置请求头

第五步：设置请求体

第六步：查看结果

3.2.5 Postman请求拉勾商城上传图片接口

拉勾商城上传图片简要文档：

请求方法：POST

URL：/index.php/Admin/Ueditor/imageUp/savepath/goods/pictitle/banner/dir/images

请求头：

Headers	参数值
Content-Type	multipart/form-data

请求体：

字段名	字段值
file	(二进制文件)
id	WU_FILE_0
name	xx图片
type	image/png
lastModifiedDate	Mon Feb 01 2021 11:46:44 GMT+0800 (中国标准时间)
size	(程序计算得出)

操作步骤：

第一步：打开Postman，添加一个请求

第二步：设置请求方法

第三步：设置URL

第四步：设置请求头

第五步：设置请求体

第六步：查看结果

3.3 Postman基本用法总结

根据接口文档，提取请求方法、URL、请求头、请求体后，依次输入请求信息，然后发送请求后，查看响应数据。

对比响应数据的参数与接口文档是否一致，参数值与需求规定的业务逻辑是否一致。

- 设置请求方法
- 设置URL
- 设置请求头
- 设置请求体
 - 设置urlencoded表单请求数据
 - URL查询参数数据
 - 请求体数据
 - 设置application/json请求体数据
 - 设置multipart/form-data上传文件
- 查看响应数据
- 需要采用的数据类型，由公司开发部门规定

4 Postman高级用法

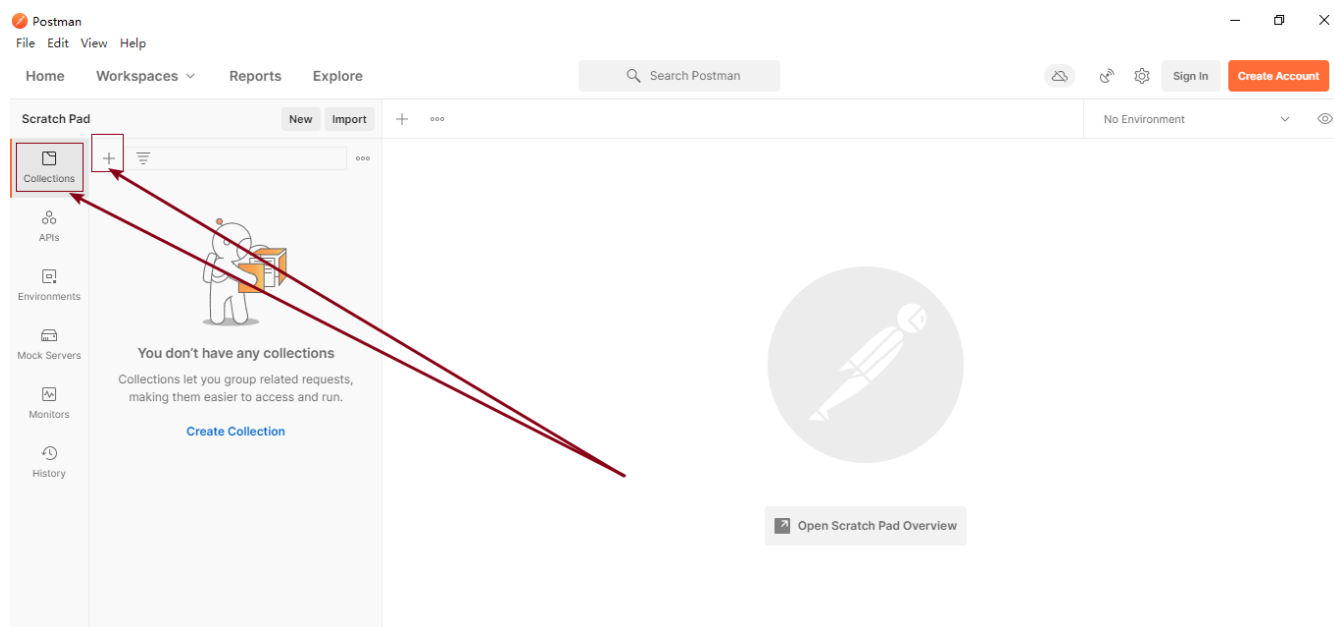
4.1 Postman管理测试用例

Postman可以使用自带的用例集管理测试用例（Collection）

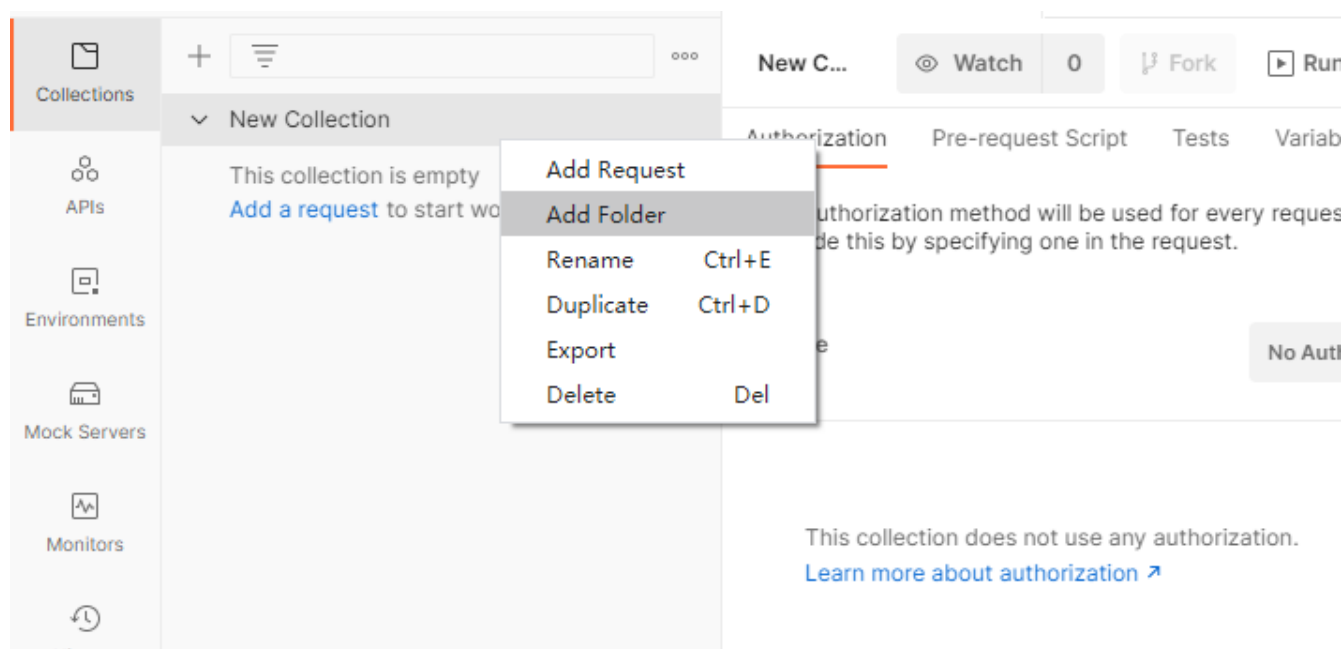
Collection：类似于一个文件系统，可以添加文件、子文件、请求等等

与操作系统的文件系统不同的是，它是Postman内部的模拟的文件系统，不是真的文件系统。

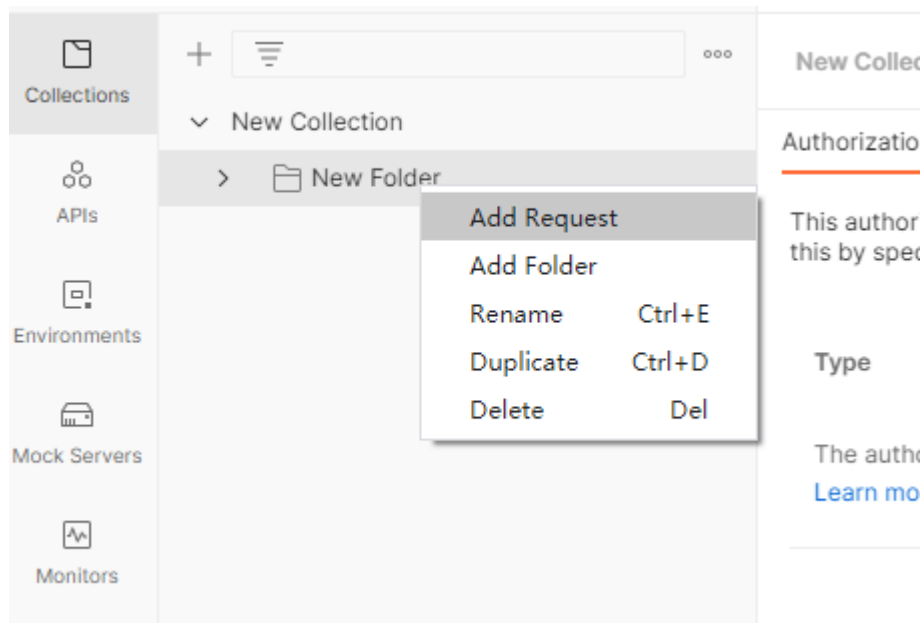
第一步：添加用例集



第二步：用例集下添加子文件夹



第三步：添加请求



工作中，如果有多个接口或者接口的测试用例，都可以通过Request来进行管理。

如果是业务场景测试或者是模块，可以用文件夹或者用例集来管理。

4.2 Postman断言

为了进行自动化测试，我们必须让计算机帮助我们判断实际结果与预期结果是否一致。

我们可以通过断言，来让计算机运行时，帮助我们判断结果

断言：计算机自动判断两组数据的关系是否为真时的过程，就叫做断言。

4.2.1 Postman的断言代码片段介绍

案例通过拉勾教育登陆接口，来完成学习

- 断言响应状态码

```
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
```

- 断言响应数据是否包含XX

```
pm.test("Body matches string", function () {
  pm.expect(pm.response.text()).to.include("string_you_want_to_search");
});
```

- 断言Json数据

```
pm.test("Your test name", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.value).to.eql(100);
});
```

- 断言整个响应体

```
pm.test("Body is correct", function () {
    pm.response.to.have.body("response_body_string");
});
```

- 断言响应头Content-Type是否存在

```
pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Content-Type");
});
```

- 断言响应时间不超过XX

```
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});
```

4.2.2 Json数据介绍

Json：是一种键值对形式的数据结构，可以跨平台跨语言使用。

Json路径：按照属性结构，从根节点寻找叶子节点所表示的一个表达式，就是Json路径

例如：

```
{
  "a": {
    "name": "泰坦巨人",
    "id": "001"
  }
}
```

如果要寻找获取到001，那么Json路径可以表达为：jsonData.a.id

如果要寻找获取到泰坦巨人，那么Json路径可以表达为：jsonData.a.name

Json数据格式的表达式形式：

```
j1 = {"a": "1", "b": "2"} # 提取a 那么是j1.a
j2 = {"a": [{"b": "1", "b": "2"}]} # 提取b 那么是 j2.a[0].b
j3 = [{"a": "1"}, {"b": "2"}] #提取b 那么是j3[1].b
```

4.3 Postman全局变量和环境变量

Postman提供了GUI界面的变量管理窗口，可以管理全局变量和环境变量

全局变量： 整个Postman都能使用的变量

环境变量： 选中环境后，才会全局生效的变量，叫做环境变量

- 环境变量作用： 可以通过变量进行参数化，方便集中管理测试数据；同时环境变量还可以起到快速切换环境的作用。

Postman 界面中对于变量的管理主要分成2大块，UI界面数据块和代码块

UI界面数据块：

- URL
- Params
- Authorization
- Headers
- Body

代码块：

- Pre-request-scripts
- Tests


UI界面数据块，引用变量方法：{{变量名}}

代码块引用变量方法：

- pm.globals.set("全局变量的变量名":"全局变量的变量值")
- pm.globals.get("全局变量的变量名")
- pm.environment.set("环境变量的变量名":"环境变量的变量值")
- pm.environment.get("环境变量名")
- 其中的环境变量，必须选择环境后才能引用

设置全局变量和环境变量的方法：

No Environment



Open Scratch Pad Overview

Environment

Add

No active Environment

An environment is a set of variables that allow you to switch the context of your requests.

[Learn more about environments](#)

Globals

Edit

No global variables

Global variables are a set of variables that are always available in a workspace.

[Learn more about globals](#)

① Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#)

×

添加环境 and 环境变量

添加全局变量

环境变量

New Environment

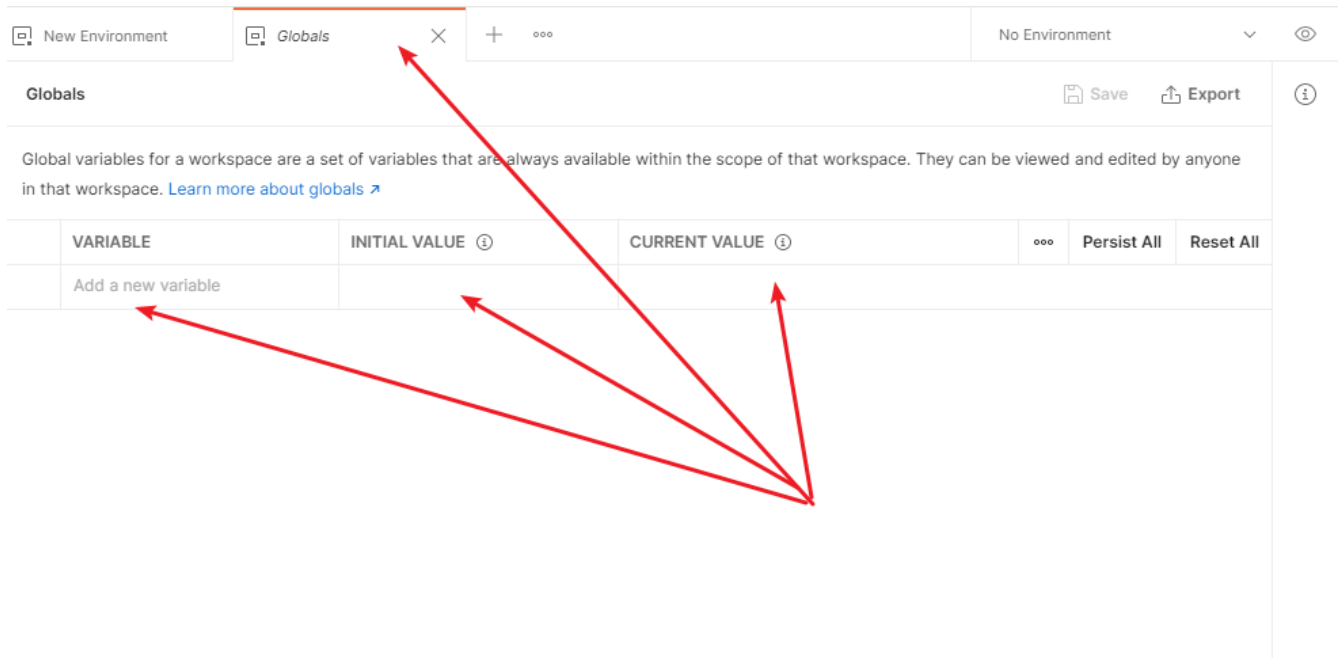
No Environment

New Environment

Fork Save Share

VARIABLE	INITIAL VALUE	CURRENT VALUE		Persist All	Reset All
Add a new variable					

全局变量



其中，当全局变量和环境变量的变量名完全一致时，优先使用环境变量。

实际工作中，对环境变量使用得更多一些。

案例一：使用全局变量替换拉勾商城后台登陆接口数据

- 先实现拉勾商城后台登陆接口（用Postman实现）
 - 按照业务逻辑，先获取拉勾商城后台登陆的验证码
抓包获取
 - 按照业务逻辑，调用拉勾商城后台登陆
抓包获取
- 设置全局变量
- 用全局变量中的变量替换拉勾商城后台登陆中请求体的数据
- 发送请求体，对比结果

案例二：使用环境变量替换拉勾商城后台登陆URL中的域名

4.4 Postman关联

4.4.1 关联介绍

关联：把多个接口联系起来的技术，就是关联。

本质上就是关联数据，例如可以把上一个接口返回的部分响应数据，当作下一个接口的入参数据，

关联的作用：实际工作中，每个接口都是拼图一样的碎片，多个接口组合起来时，才能形成真正的功能，这个时候，我们必须按照这些接口的数据依赖关系和顺序，关联多个接口，实现多接口的组合测试。也可以应用于业务场景测试。

操作步骤：

第一步：调用上一个接口

第二步：在Tests中编写提取上一个接口响应数据的代码

```
var jsonData = pm.response.json()
```

第三步：保存数据到全局变量

```
pm.globals.set("a", "a")
```

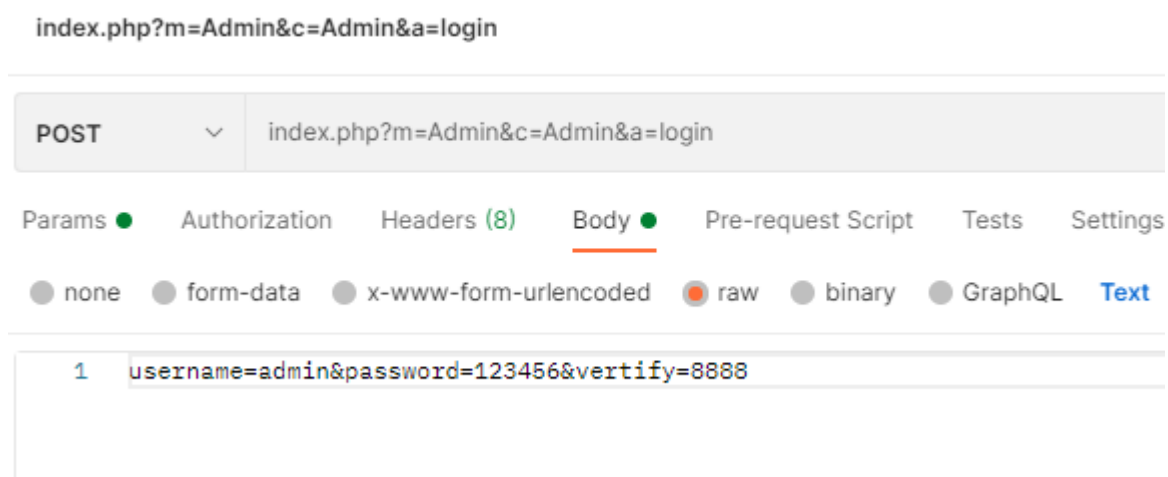
第四步：在下一个接口中引用保存的数据

```
{{保存的数据}}
```

4.4.2 拉勾商城上传图片 and 查询图片接口关联

实现关联技术时，都需要先确保接口能够正常使用。

第一步：使用Postman进行拉勾商城后台登陆请求



注意：由于这个有验证码，需要mock验证码为万能验证码才能进行


```
1 <?php
2 // +-----+
3 // | ThinkPHP [ WE CAN DO IT JUST THINK IT ] |
4 // +-----+
5 // | Copyright (c) 2006-2014 http://thinkphp.cn All rights reserved. |
6 // +-----+
7 // | Licensed ( http://www.apache.org/licenses/LICENSE-2.0 ) |
8 // +-----+
9 // | Author: 麦当苗儿 <zuojiayi@vip.qq.com> <http://www.zizai.cn> |
10 // +-----+
11
12 namespace think;
13
14 class Verify {
15     protected $config = array(
16         'seKey' => 'ThinkPHP.CN', // 验证码加密密钥
17         'codeSet' => '23456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ', // 验证码字符集合
18         'expire' => 1800, // 验证码过期时间 (s)
19         'useZh' => false, // 使用中文验证码
20         'zhSet' => '们以我到他作时要动国产的一是工就年阶义发成部民可出能方进在了不和有大这主中人上为来分生对于学下级地个用同行面说种过命度革而多子后自社',
21         'useImgBg' => false, // 使用背景图片
22         'fontSize' => 25, // 验证码字体大小(px)
23         'useCurve' => true, // 是否画混淆曲线
24         'useNoise' => true, // 是否添加杂点
25         'imageH' => 0, // 验证码图片高度
26         'imageW' => 0, // 验证码图片宽度
27         'length' => 5, // 验证码位数
28         'fontttf' => '', // 验证码字体, 不设置随机获取
29         'bg' => array(243, 251, 254), // 背景颜色
30         'reset' => true, // 验证码成功后是否重置
31     );
32 }
```

把这个改成8888

```
namespace think;

class Verify {
    protected $config = array(
        'seKey' => 'ThinkPHP.CN', // 验证码加密密钥
        'codeSet' => '8888', // 验证码字符集合
        'expire' => 1800, // 验证码过期时间 (s)
        'useZh' => false, // 使用中文验证码
        'zhSet' => '们以我到他作时要动国产的一是工就年阶义发成部民可出能方进在了不和有大这主中人上为来分生对于学下级地个用同行面说种过命度革而多子后自社',
        'useImgBg' => false, // 使用背景图片
        'fontSize' => 25, // 验证码字体大小(px)
    );
}
```

重启拉勾商城，再看验证码，就变成8888了



第二步：请求上传图片接口

localhost/index.php/Admin/Ueditor/imageUp/savepath/goods/pictitle/banner/dir/images

POST localhost/index.php/Admin/Ueditor/imageUp/savepath/goods/pictitle/banner/dir/images Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	file	素材_华为工程师职级2.png			
	Key	Value	Description		

Body Cookies (1) Headers (11) Test Results Status: 200 OK Time: 1655 ms Size: 474 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2    "url": "/public/upload/goods/2021/04-15/415c31f2db0fca099d1ec595048c75b6.png",
3    "title": "banner",
4    "original": "",
5    "status": "success"
6  }
```

第三步：提取上传图片接口返回的响应数据

localhost/index.php/Admin/Ueditor/imageUp/savepath/goods/pictitle/banner/dir/images

POST localhost/index.php/Admin/Ueditor/imageUp/savepath/goods/pictitle/banner/dir/images

Params Authorization Headers (9) Body Tests Settings

```
1  var jsonData = pm.response.json();
2
3  // 提取图片的路径
4  var img_path = jsonData.url;
5  // 保存到全局变量
6  pm.globals.set("img_path", img_path);
```

第四步：查询提取的图片

http://localhost/{{img_path}}

Save

</>

GET

http://localhost/{{img_path}}

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 6 ms

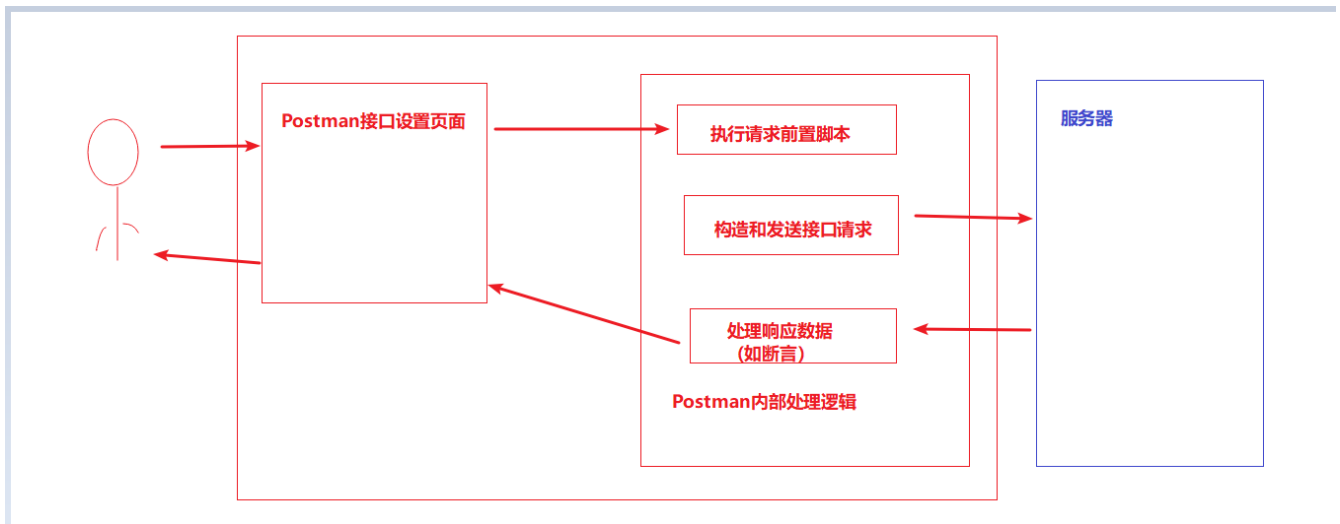
Size: 297.94 KB

Save Response

产品测试工程师									
技术职称	工程师三级			工程师二级			工程师一级		
学历	专科	本科	硕士	专科	本科	硕士	专科	本科	硕

4.5 Postman请求前置脚本

请求前置脚本：发送请求之前要执行的脚本。



请求前置脚本作用：和代码一样，Javascript代码能做的时候，请求前置脚本都能做。

Js (javascript) 能构造测试数据，设计接口请求，设置执行流程等等。

请求前置脚本常见作用：用来获取动态数据，签名接口数据，加密接口数据

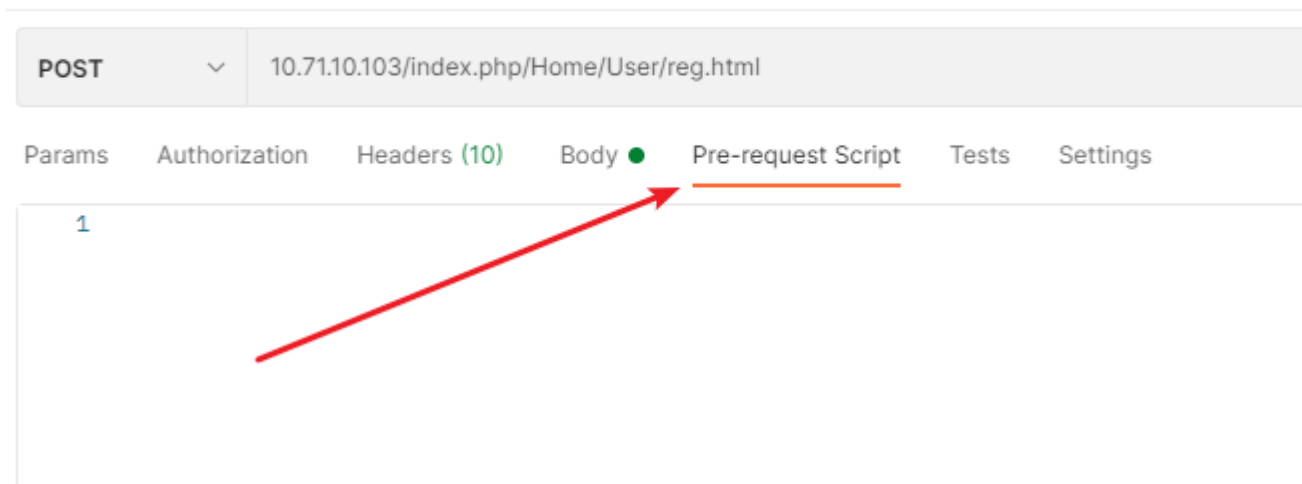
动态数据：时间戳、随机数等等

对整个接口数据签名：防止数据被篡改

加密数据：防止数据被偷窥

请求前置脚本设置位置：

10.71.10.103/index.php/Home/User/reg.html



案例：拉勾商城注册

拉勾商城注册接口

拉勾商城 3.5
www.lagou.com

欢迎注册

手机注册

邮箱注册

* 手机号码: 请输入手机号码

* 图像验证码: 图像验证码



* 设置密码: 6-16位大小写英文字母、数字或符号的组合

* 确认密码: 请再次输入密码

☒ 我已阅读并同意 《服务协议》

同意协议并注册

拉勾商城，注册用例时，可以发现，注册的接口请求数据密码参数值是被加密的

如果我们不按照要求生成加密数据，进行注册，那么就算注册成功也无法登陆

Structure Sequence Overview Contents Summary Chart Notes

http://10.71.10.103

- index.php
 - Home
 - Index
 - index.html
 - index.html
 - index.html
 - user
 - User
 - verify
 - type
 - user_reg.html (320x100)
 - template
 - public
 - index.php?m=Home&c=Index&a=qr_code&data=
 - index.php?m=Home&c=Index&a=qr_code&data=
 - index.php?m=Home&c=Cart&a=header_cart_1
 - index.php?m=Home&c=Index&a=qr_code&data=

Name	Value
auth_code	TPSHOP
scene	1
username	13800000003
verify_code	8888
password	519475228fe35ad067744465c42a19b2
password2	519475228fe35ad067744465c42a19b2

注册时，被加密了

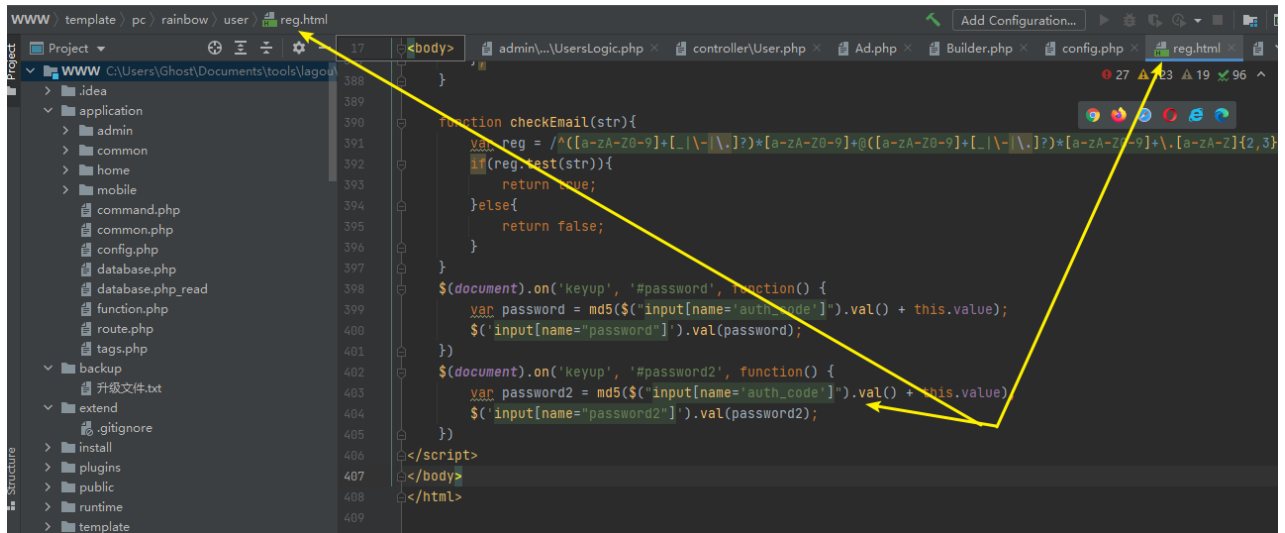
实现方法：

- 先熟悉加密算法

实际工作中，加密算法是开发提供的，为了安全他们不一定提供。

如果没有提供，那么就不能利用请求前置脚本生成密码了，只能写死。

找到reg.html中密码处理相关代码



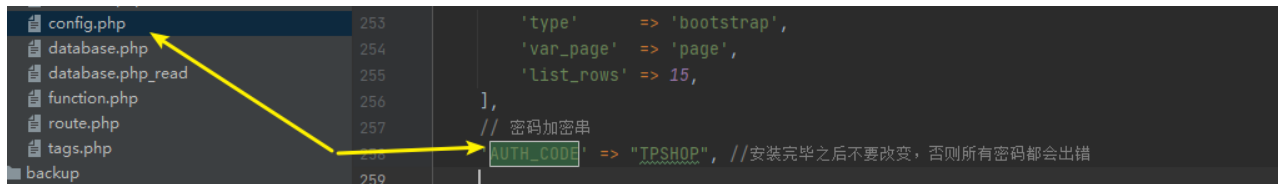
```
function checkEmail(str){
    var reg = /^[a-zA-Z0-9]+[_|\-|\.]?*[a-zA-Z0-9]+@[a-zA-Z0-9]+[_|\-|\.]?*[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$/;
    if(reg.test(str)){
        return true;
    }else{
        return false;
    }
}

$(document).on('keyup', '#password', function() {
    var password = md5($('input[name="auth_code"]').val() + this.value);
    $('input[name="password"]').val(password);
});

$(document).on('keyup', '#password2', function() {
    var password2 = md5($('input[name="auth_code"]').val() + this.value);
    $('input[name="password2"]').val(password2);
});
```

分析可以发现，代码对reg.html页面中，输入的密码进行了md5(auth_code + 输入的密码)的处理

而auth_code的值是TPSHOP



```
'type' => 'bootstrap',
'var_page' => 'page',
'list_rows' => 15,
],
// 密码加密串
AUTH_CODE => "TPSHOP", //安装完毕之后不要改变，否则所有密码都会出错
```

所以，如果我们注册时，输入的密码是123456，那么我们就应该输入md5("TPSHOP" + "123456")，就能生成对应的MD5之后的密码了。（MD5是进行签名运算，也可以当作密码使用）

- 先用Postman调试注册请求，并注册通过

http://10.71.10.103/index.php?m=Home&c=User&a=verify&type=user_reg

GET ⌵ http://10.71.10.103/index.php?m=Home&c=User&a=verify&type=user_reg

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	m	Home
<input checked="" type="checkbox"/>	c	User
<input checked="" type="checkbox"/>	a	verify
<input checked="" type="checkbox"/>	type	user_reg
	Key	Value

Body Cookies (2) Headers (12) Test Results 🌐 Status:



GET http://10.71.10.103/... ● POST 10.71.10.103/inde... ● + ...

10.71.10.103/index.php/Home/User/reg.html

POST ⌵ 10.71.10.103/index.php/Home/User/reg.html

Params Authorization Headers (10) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	auth_code	TPSHOP
<input checked="" type="checkbox"/>	scene	1
<input checked="" type="checkbox"/>	username	13800000004

- 先用Postman调试注册请求，并注册通过
利用请求前置脚本+全局变量，实现自动填充注册需要MD5的密码

第一步：在注册接口请求前置脚本，按照规则，输入MD5的代码

```
// MD5 加密
var auth_code = "TPSHOP"+"123456";
var signedMd5 = CryptoJS.MD5(auth_code).toString();
console.log(signedMd5);
// 保存生成的MD5123456后的密码
pm.globals.set("signed", signedMd5);
```

10.71.10.103/index.php/Home/User/reg.html

POST 10.71.10.103/index.php/Home/User/reg.html

Params Authorization Headers (10) Body Pre-request Script Tests Settings

```
1 // MD5 加密
2 var auth_code = "TPSHOP"+"123456";
3 var signedMd5 = CryptoJS.MD5(auth_code).toString();
4 console.log(signedMd5);
5 // 保存生成的MD5123456后的密码
6 pm.globals.set("signed", signedMd5);
```

第二步：在请求体中关联，请求前置脚本保存的全局变量signed

10.71.10.103/index.php/Home/User/reg.html

POST 10.71.10.103/index.php/Home/User/reg.html

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

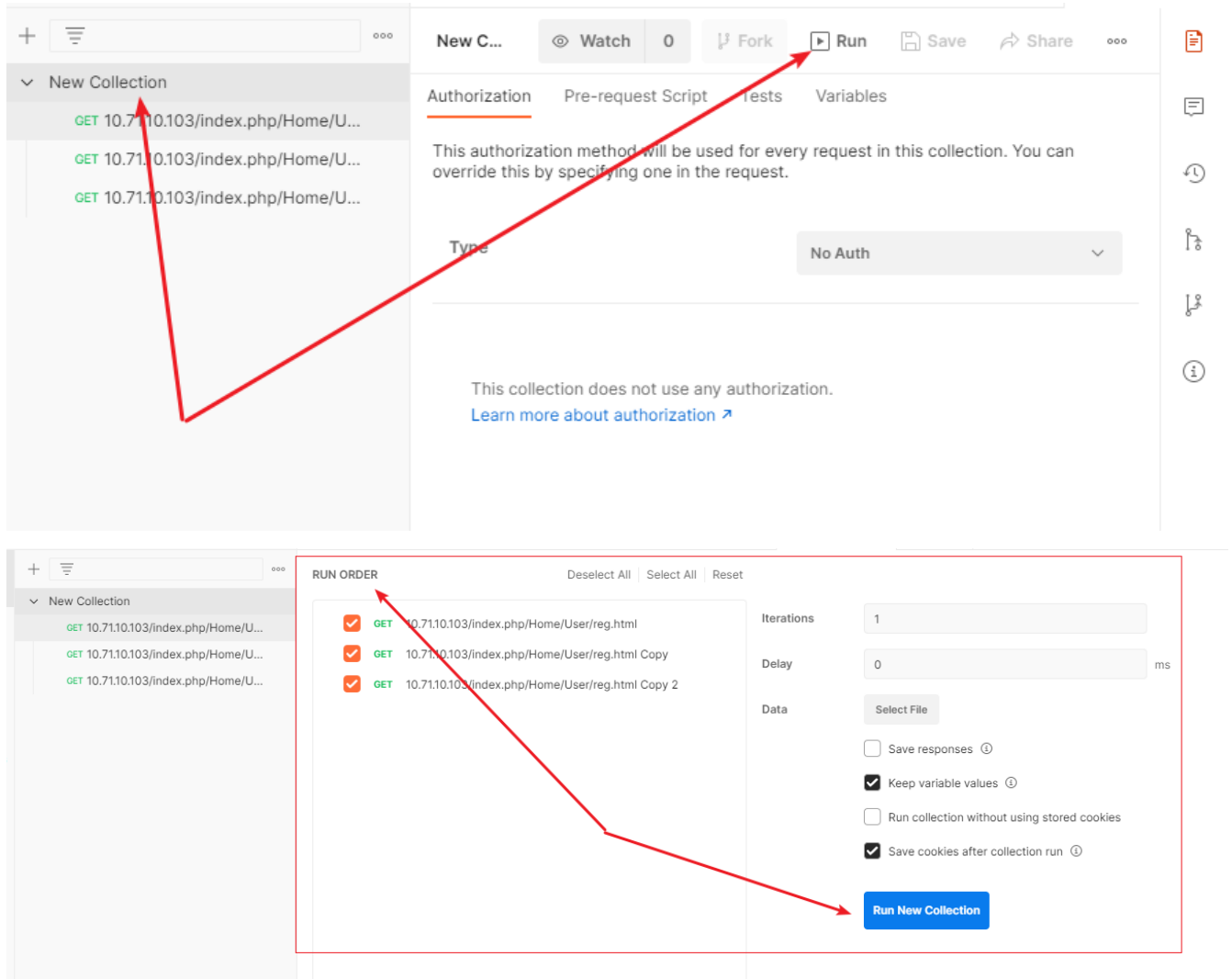
<input checked="" type="checkbox"/>	verify_code	8888
<input checked="" type="checkbox"/>	password	{{signed}}
<input checked="" type="checkbox"/>	password2	{{signed}}

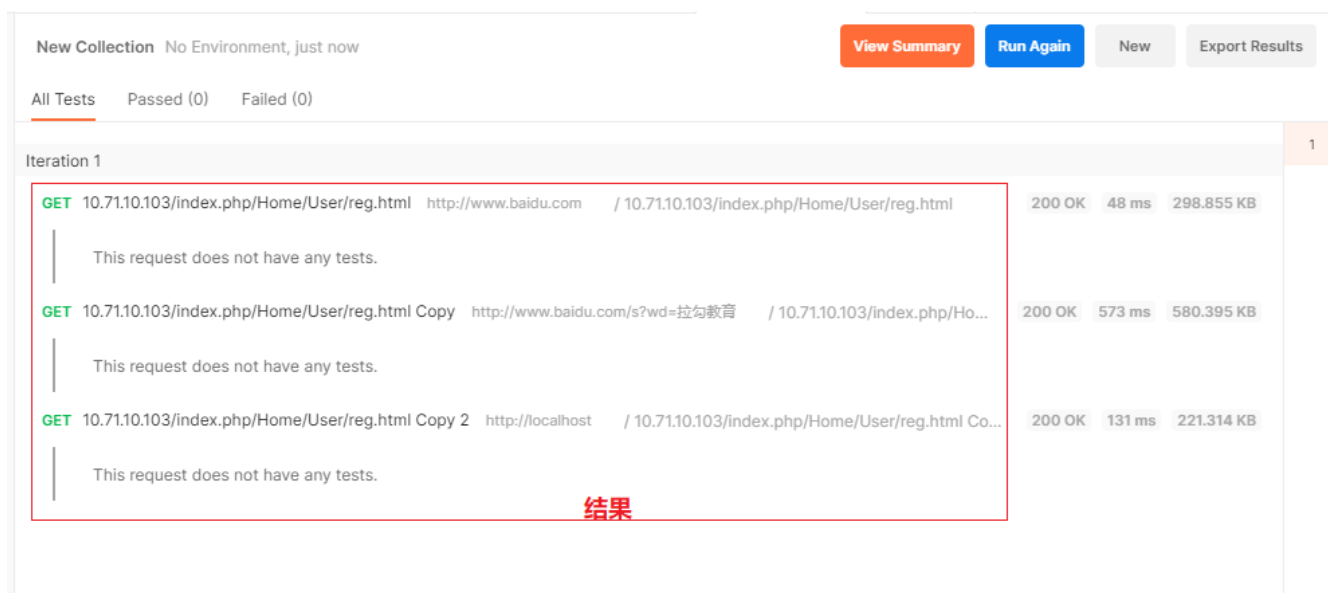
第三步：重新请求，查看结果

4.6 Postman批量运行测试用例

作用：自动执行编写好的测试用例，实现自动化接口测试。

Postman可以使用自带的Collection Runner（用例集运行器）来批量运行测试用例





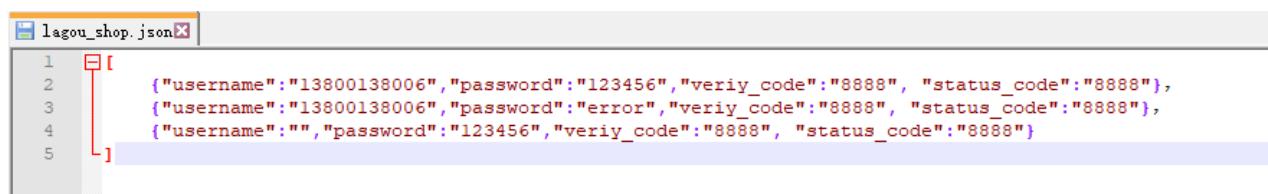
5 Postman参数化和数据驱动

参数化： 把数据用参数来代替，从而进行测试的过程。**参数化** 是实现数据驱动测试的前置技术

数据驱动： 把测试数据和测试脚本分离，用数据来驱动测试用例的执行。简单的说，就是一条数据对应一条测试用例。

5.1 Postman实现数据驱动--支持的文件

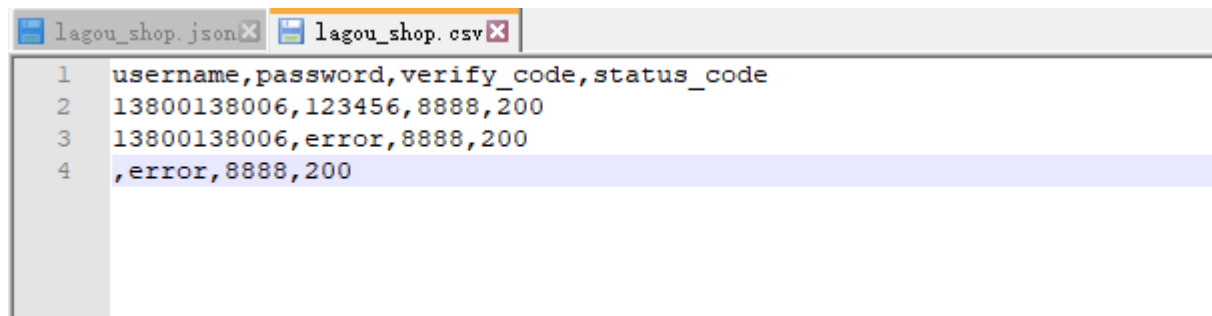
- Json



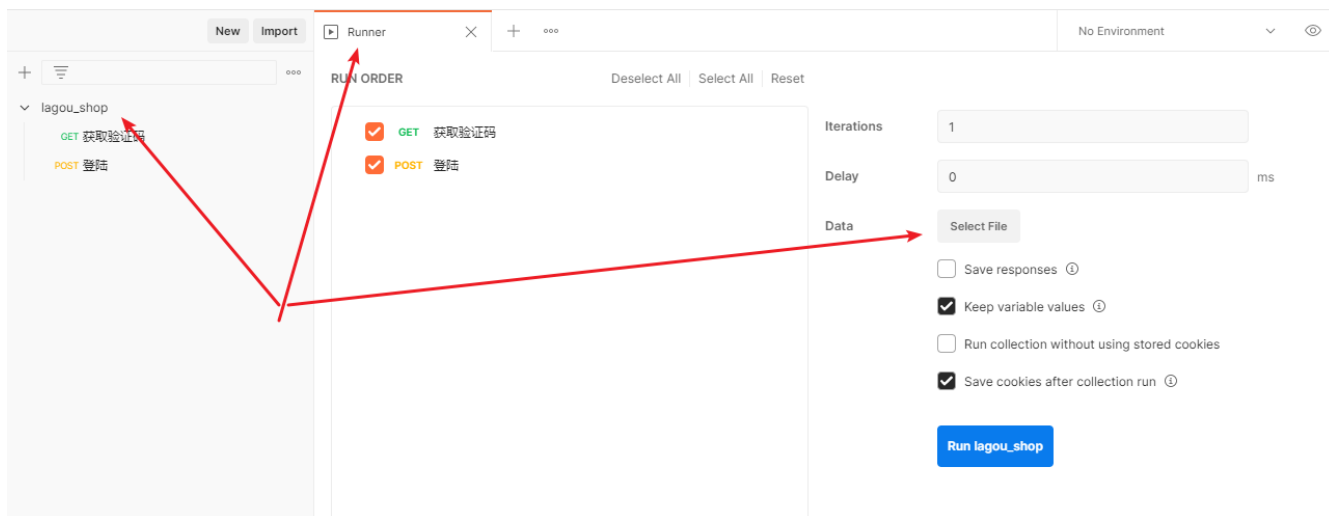
在postman支持的json数据文件中，**json数据文件的内容中，最外层必须是中括号**，中括号中的每一组数据必须是大括号，大括号中，就是我们要使用的数据，是键值对的形式

- CSV

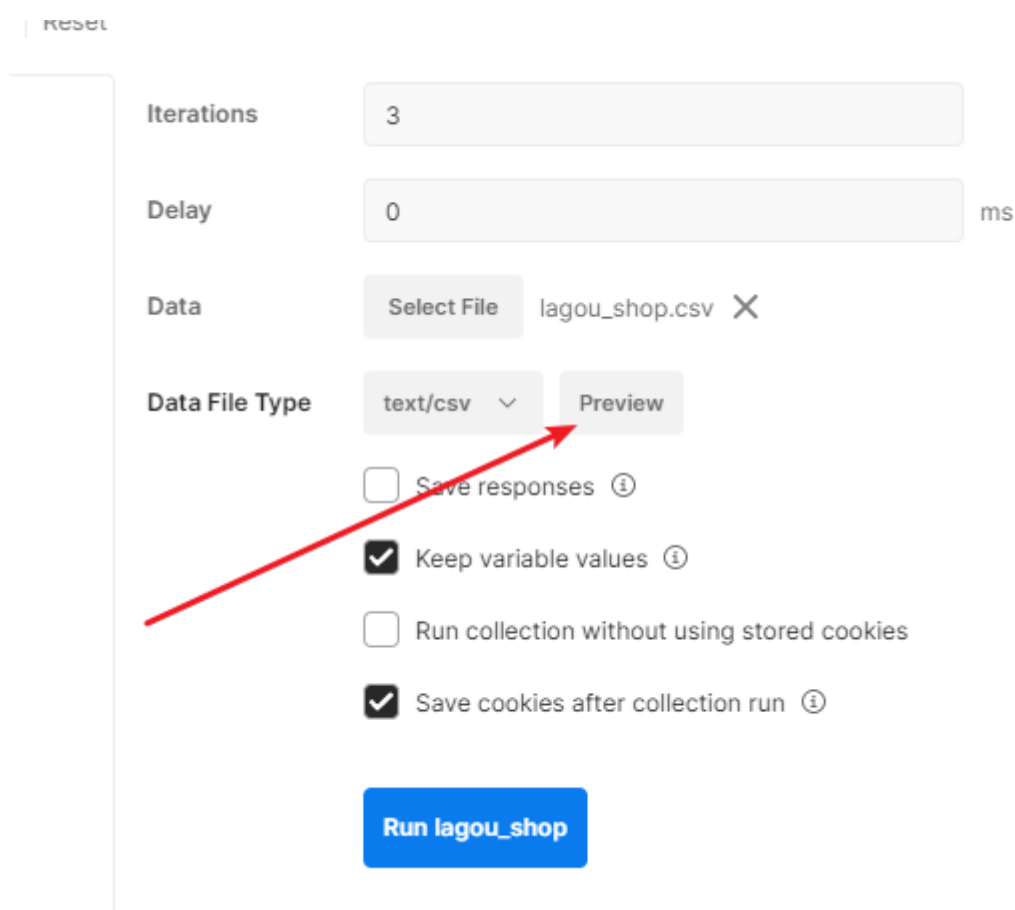
CSV适合大量数据的构造，以及测试



5.2 Postman实现数据驱动--导入外部文件



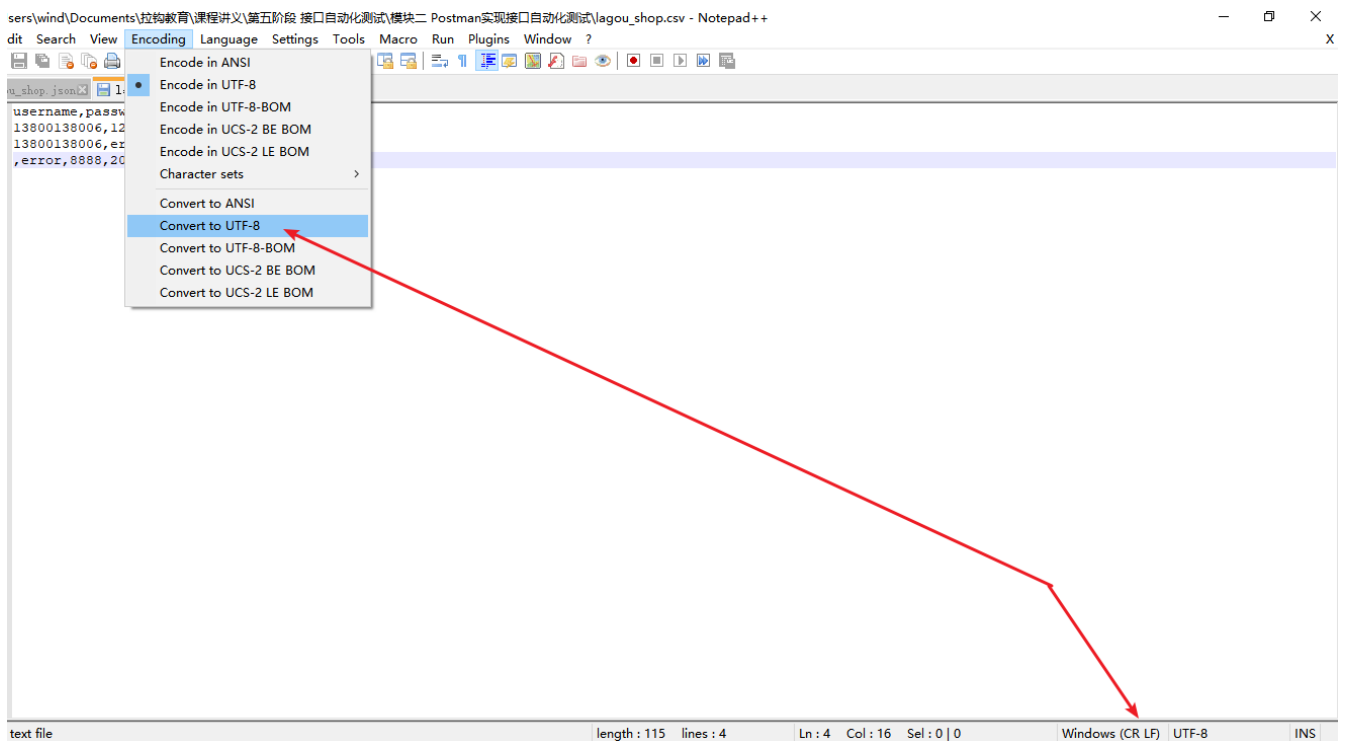
导入之后，点击Preview



可以查看到数据

Iteration	username	password	verify_code	status_code
1	13800138006	123456	8888	200
2	13800138006	"error"	8888	200
3	""	"error"	8888	200

如果有中文乱码，可以改变编码，重新查看



5.3 Postman实现数据驱动--关联数据文件中的参数

- 请求参数区域：用{{外部数据文件参数名}}关联
- 代码区域：请求前置脚本和Tests脚本都可以用data.外部数据文件变量关联

请求参数区域示例：

POST http://10.71.10.103/index.php?m=Home&c=User&a=do_login

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> username	{{username}}	
<input checked="" type="checkbox"/> password	123456	
<input checked="" type="checkbox"/> verify_code	8888	
Key	Value	Description

Response

agou_shop.json lagou_shop.csv

```
username,password,verify_code,status_code
13800138006,123456,8888,200
13800138006,error,8888,200
,error,8888,200
```

Hit Send to get a response

代码区域示例:

POST http://10.71.10.103/index.php?m=Home&c=User&a=do_login

Params Authorization Headers (10) Body Pre-request Script Tests Settings

1 data.username

agou_shop.json lagou_shop.csv

```
username,password,verify_code,status_code
13800138006,123456,8888,200
13800138006,error,8888,200
,error,8888,200
```

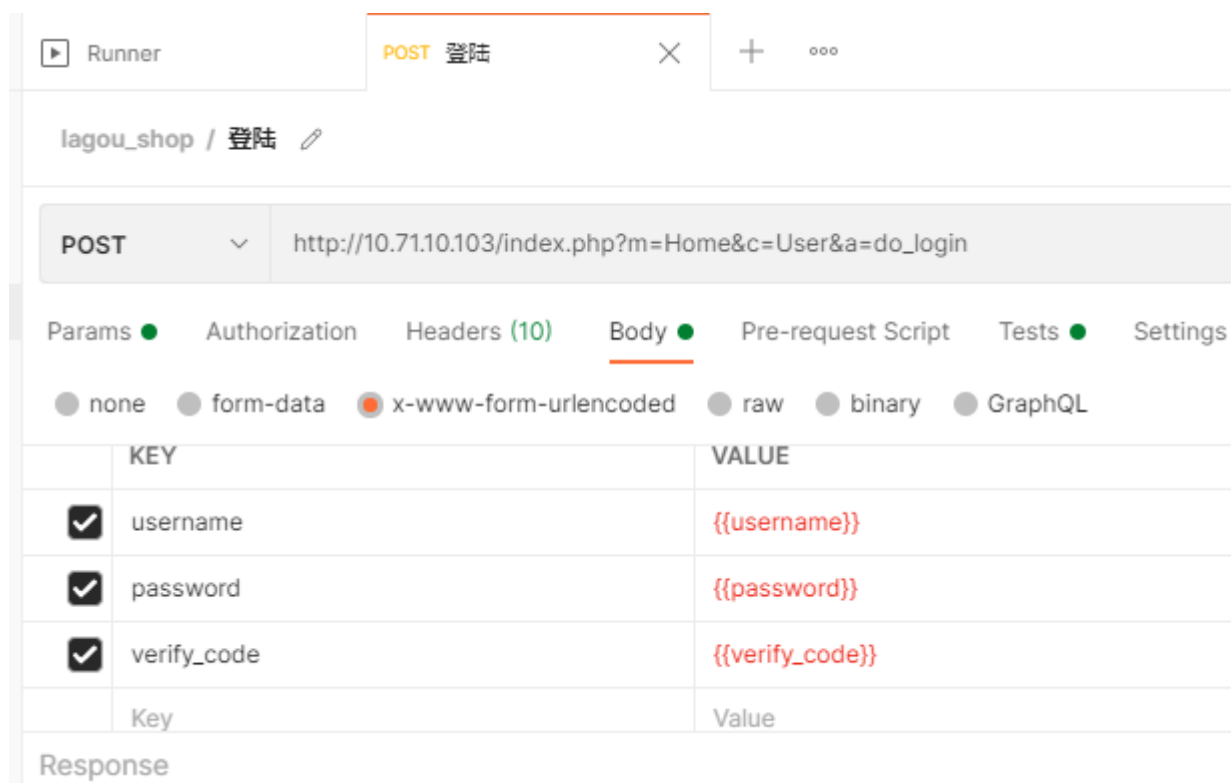
5.4 拉勾商城登陆案例数据驱动测试

5.4.1 设计JSON数据文件lagou_shop.csv

```
lagou_shop.json x lagou_shop.csv x
1 username,password,verify_code,status_code
2 13800138006,123456,8888,200
3 13800138006,error,8888,200
4 ,error,8888,200
```

5.4.2 关联postman脚本中的数据到lagou_shop.csv中的变量

请求参数区域关联

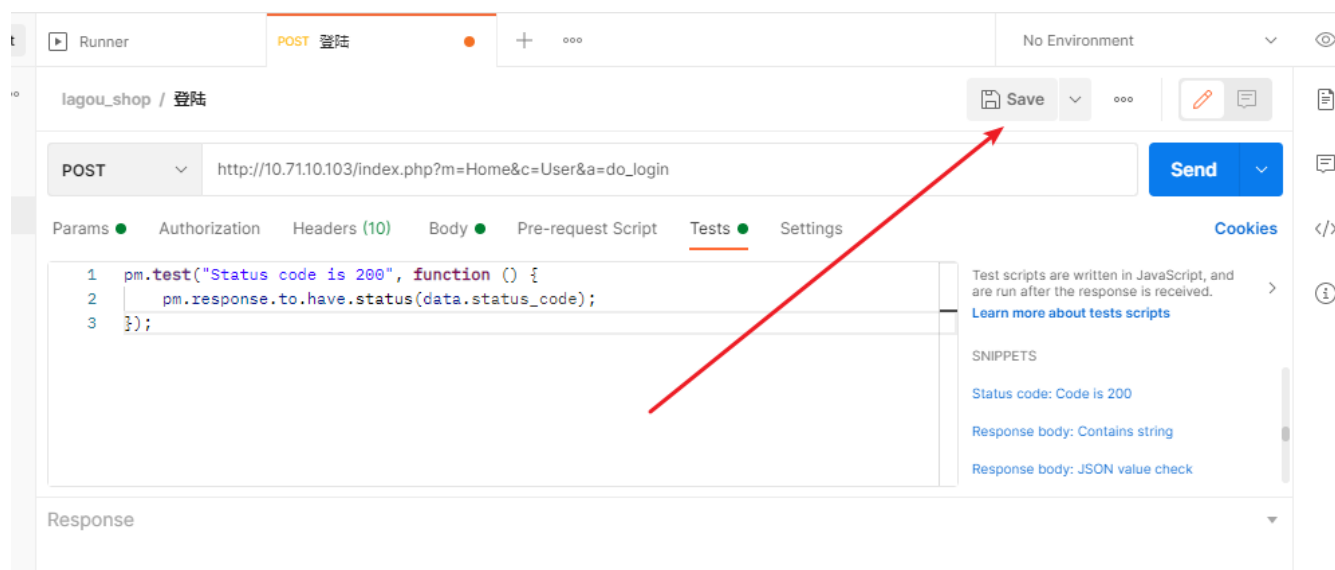


代码区域Tests编写断言，并关联数据文件中的status_code，用判断请求有没有通过

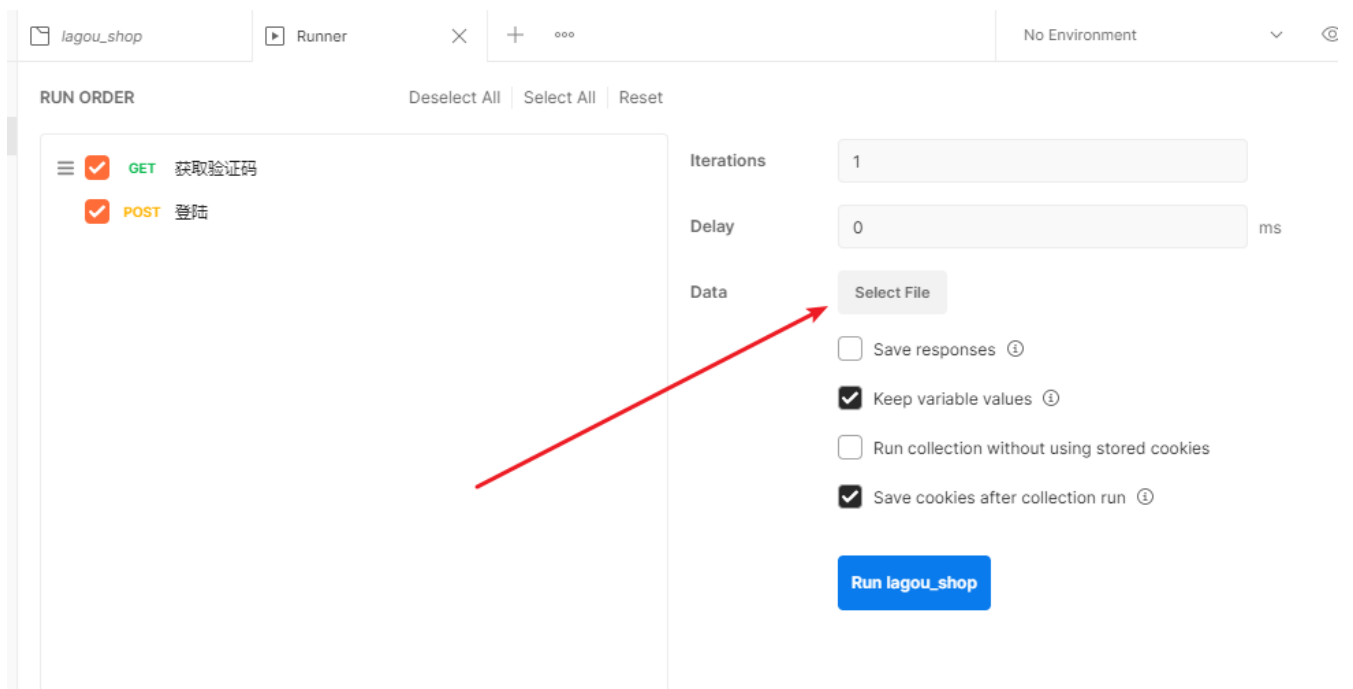
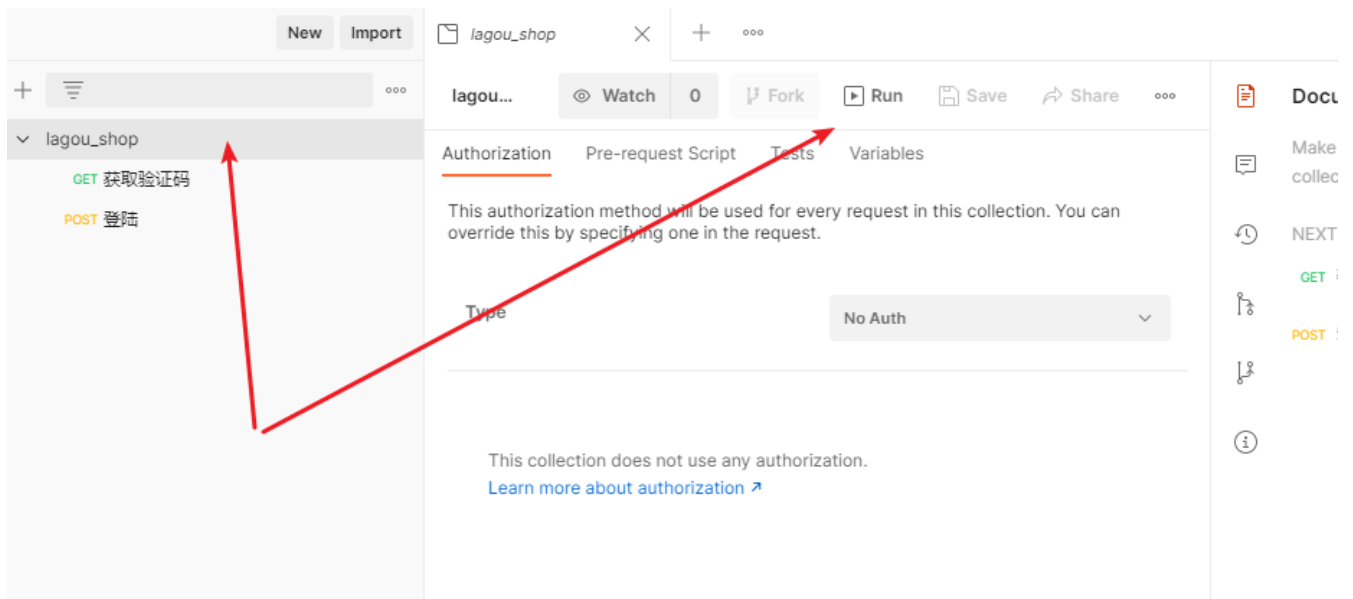


5.4.3 使用runner运行

保存



运行



名称	类型	大小
assets	文件夹	
lagou_shop.csv	XLS 工作表	1 KB
lagou_shop.json	JSON 文件	1 KB
Postman实现接口自动化测试.md	Markdown File	18 KB

Iterations

3

Delay

0

ms

Data

Select File

lagou_shop.csv

X

Data File Type

text/csv

Preview

☐

Save responses

i

☒

Keep variable values

i

☐

Run collection without using stored cookies

☒

Save cookies after collection run

i

Run lagou_shop

5.4.4 查看结果

点击蓝色的Run，查看结果

NewImport

lagou_shop

lagou_shop

No Environment

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

+

...

+

lagou_shop

+

<

####

6 Postman生成测试报告

Postman使用runner运行时，生成的报告只能在Postman内部查看，并不能很好的传递给其他人。

所以可以生成一个HTML报告，HTML容易传播，只要有浏览器都能打开查看

Postman需要生成HTML报告需要使用newman，借助newman工具生成

6.1 newman介绍和安装

newman是使用node.js开发，专门为postman做的生成测试报告的工具插件。

我们需要安装node.js、newman、newman插件：newman-reporter-html

6.1.1 安装node.js

下载node.js: <https://nodejs.org/en/>

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

#BlackLivesMatter

New security releases now available for 15.x, 14.x, 12.x and 10.x release lines

Download for Windows (x64)

14.16.1 LTS

Recommended For Most Users

15.14.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

安装node.js：双击安装默认方式进行安装即可

检查方法：打开cmd分别输npm -v能看到版本代表安装成功

```
C:\Users\wind>npm -v
6.11.3
```

6.1.2 安装newman

打开cmd，输入npm install -g newman

```
C:\Users\wind>npm install -g newman  
[.....] / rollbackFailedOptional: verb npm-session f497792d73fd184b
```

进度条

安装命令

安装完成后输入newman -v, 能看到版本代表安装成功

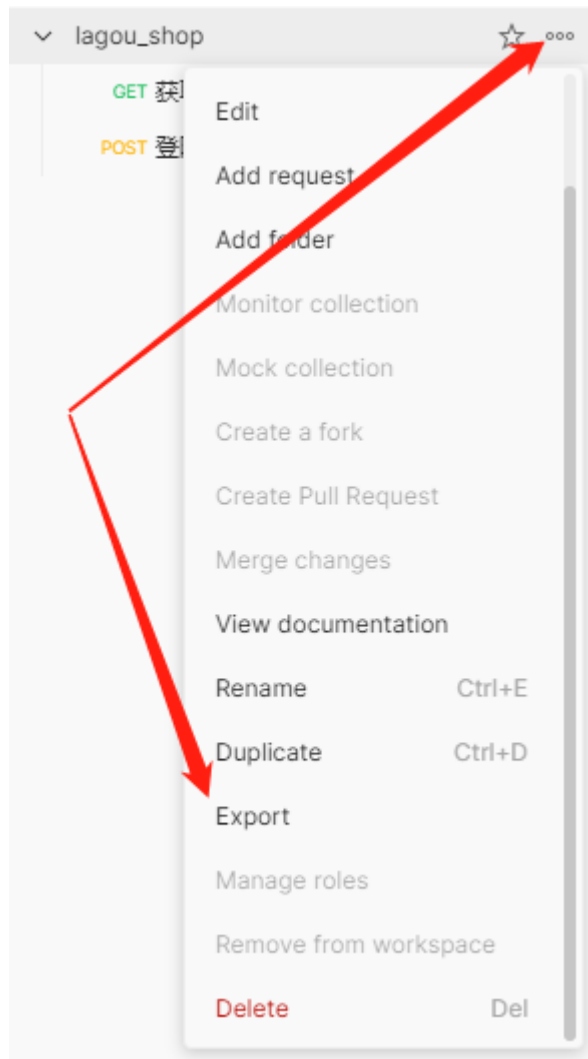
```
C:\Users\wind>newman -v  
4.5.6
```

6.1.3 安装newman-reporter-html

```
C:\Users\wind>npm install -g newman-reporter-html  
npm WARN newman-reporter-html@1.0.5 requires a peer of newman@4 but none is installed. You must install peer dependencies yourself.  
  
+ newman-reporter-html@1.0.5  
added 12 packages from 44 contributors in 12.523s
```

6.2 导出用例集和环境文件

导出用例集



导出环境文件



6.3 使用newman命令运行用例集


6.3.1 newman命令介绍

```
newman run 用例集.json -e 环境文件.json -d 数据文件.json -r html --reporter-html-export report.html
```

```
newman run 用例集.json 运行用例集的意思
-e 环境文件.json 指定运行的环境
-d 数据文件.json 指定运行的数据
-r html 生成html报告
--reporter-html-export report.html 指定html报告名称是report.html
```

6.3.2 案例

```
newman run lagou_shop.postman_collection.json
```

 lagou_shop.postman_collection.json

案例二：加上-r html


```
newman run lagou_shop.postman_collection.json -r html
```

案例三：再加上--reporter-html-export

```
newman run lagou_shop.postman_collection.json -r html --reporter-html-export report.html
```

案例四：使用环境文件

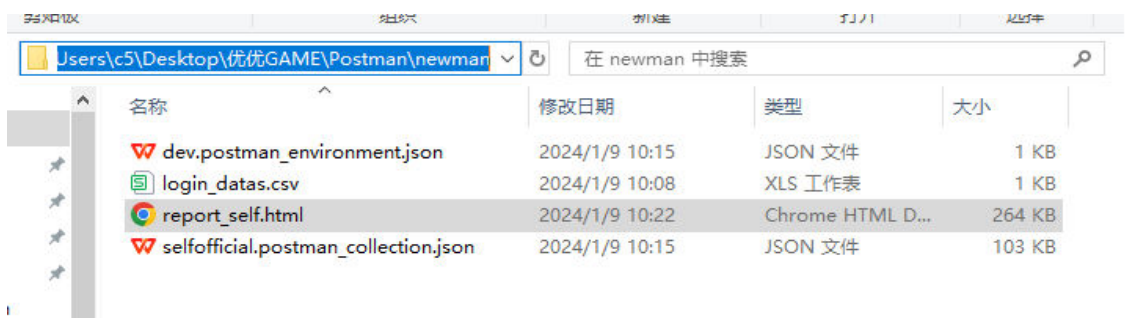
```
newman run lagou_shop.postman_collection.json -r html --reporter-html-export report.html -
e TEST.postman_environment.json
```

 TEST.postman_environment.json

案例五：使用数据文件

```
newman run lagou_shop.postman_collection.json -r html --reporter-html-export report.html -
e TEST.postman_environment.json -d lagou_shop.csv
```

```
newman run C:\Users\c5\Desktop\优优
GAME\Postman\newman\selfofficial.postman_collection.json -r html --reporter-html-export C:
\Users\c5\Desktop\优优GAME\Postman\newman\report_self.html -e C:\Users\c5\Desktop\优优
GAME\Postman\newman\dev.postman_environment.json -d C:\Users\c5\Desktop\优优
GAME\Postman\newman\login_datas.csv
```



名称	修改日期	类型	大小
dev.postman_environment.json	2024/1/9 10:15	JSON 文件	1 KB
login_datas.csv	2024/1/9 10:08	XLS 工作表	1 KB
report_self.html	2024/1/9 10:22	Chrome HTML D...	264 KB
selfofficial.postman_collection.json	2024/1/9 10:15	JSON 文件	103 KB