

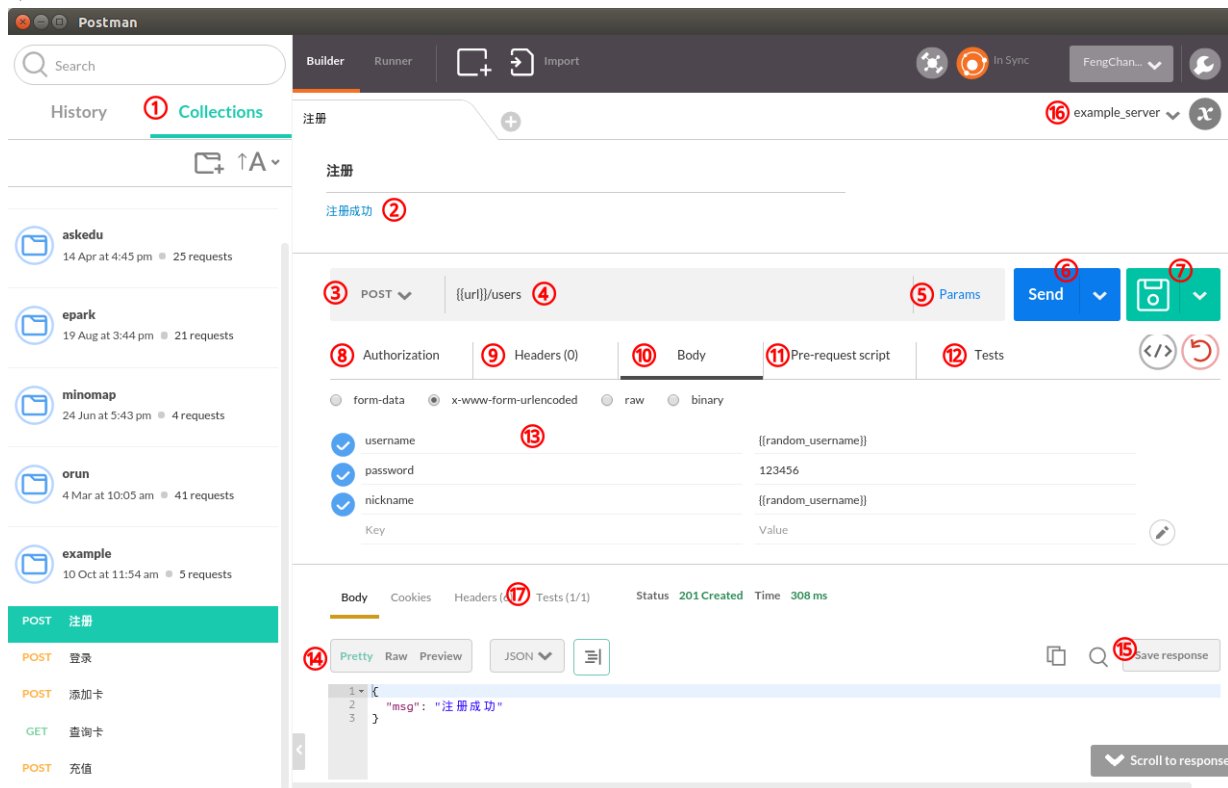
API自动化测试利器——Postman

作者: [fengchang](#) | 时间: September 26, 2015 | 分类: [工具](#)

自从开始做API开发之后，我就在寻找合适的API测试工具。一开始不是很想用Chrome扩展，用的[WizTools](#)的工具，后来试过一次[Postman](#)之后就停不下来了，还买了付费的Jetpacks。推出Team Sync Beta之后我又把这个工具推广给团队，作为API文档使用。看到中文网络上关于这个工具的文章并不多，于是决定写一篇小文介绍一下。

一、基本功能

Postman的功能在[文档](#)中有介绍。不过文档略啰嗦，这里简单介绍一下主界面，入门功能就都提到了。



1. Collections: 在Postman中，Collection类似文件夹，可以把同一个项目的请求放在一个Collection里方便管理和分享，Collection里面也可以再建文件夹。如果做API文档的话，可以每个API对应一条请求，如果要把各种输入都测到的话，就需要每条测试一条请求了。这里我新建了一个example用于介绍整个流程，五个API对应五条请求。这个Collection可以通过<https://www.getpostman.com/collections/96b64a7c604072e1e4ee> 导入你自己的Postman中。
2. 上面的黑字注册是请求的名字，如果有Request description的话会显示在这下面。下面的蓝字是保存起来的请求结果，点击可以载入某次请求的参数和返回值。我会用这个功能给做客户端的同事展示不同情况下的各种返回值。保存请求的按钮在15。
3. 选择HTTP Method的地方，各种常见的不常见的非常全。
4. 请求URL，两层大括号表示这是一个环境变量，可以在16的位置选择当前的environment，环境变量就会被替换成该environment里variable的值。
5. 点击可以设置URL参数的key和value

6. 点击发送请求
7. 点击保存请求到Collection，如果要另存为的话，可以点击右边的下箭头
8. 设置鉴权参数，可以用OAuth之类的
9. 自定义HTTP Header，有些因为Chrome愿意不能自定义的需要另外装一个插件Interceptor，在16上面一行的卫星那里
10. 设置Request body，13那里显示的就是body的内容
11. 在发起请求之前执行的脚本，例如request body里的那两个random变量，就是每次请求之前临时生成的。
12. 在收到response之后执行的测试，测试的结果会显示在17的位置
13. 有四种形式可以选择，form-data主要用于上传文件。x-www-form-urlencoded是表单常用的格式。raw可以用来上传JSON数据
14. 返回数据的格式，Pretty可以看到格式化后的JSON，Raw就是未经处理的数据，Preview可以预览HTML页面
15. 点击这里把请求保存到2的位置
16. 设置environment variables和global variables，点击右边的x可以快速查看当前的变量。
17. 测试执行的结果，一共几个测试，通过几个。

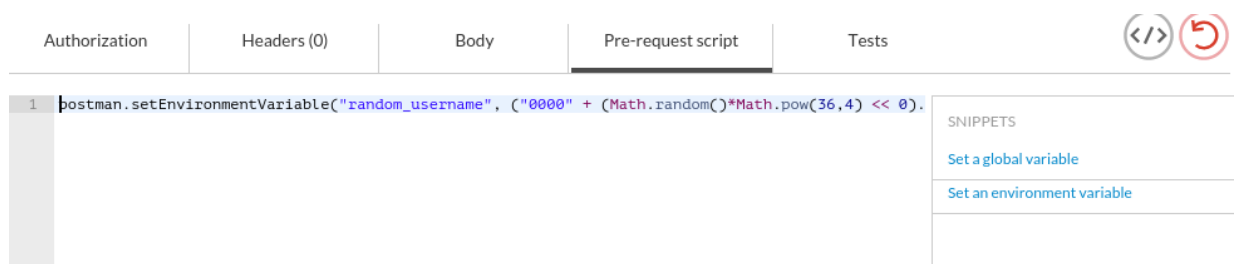
这个界面就是免费版的主要内容，和其他API测试工具相比，已经足够好用。如果要使用自动化测试，需要购买9.99美金的Jetpacks，暂时不想购买的话可以试一下[Team版Postman](#)。现在是免费试用的，不但拥有Jetpacks的功能，还能与其他账户同步Collection。

二、测试工具

测试工具主要包括三部分，在发起请求之前运行的Pre-request，在收到应答之后运行的Test，和一次运行所有请求的Collection Runner

1. Pre-request

Pre-request的编写界面如下：



Pre-request和Test用的语言都是JavaScript，Postman在一个沙盒里执行代码，提供给用户的库和函数可以在[这里](#)查看。而常用的功能都可以通过右边的**Code Snippets**实现，点击就可以插入到代码区域。

可以看到Pre-request里常用的功能就两种，设置环境变量和设置全局变量。这条请求的pre-request就是在注册之前生成一个字符串作为随机用户名。

```
postman.setEnvironmentVariable("random_username", ("0000" +  
(Math.random()*Math.pow(36,4) << 0).toString(36)).slice(-4));
```

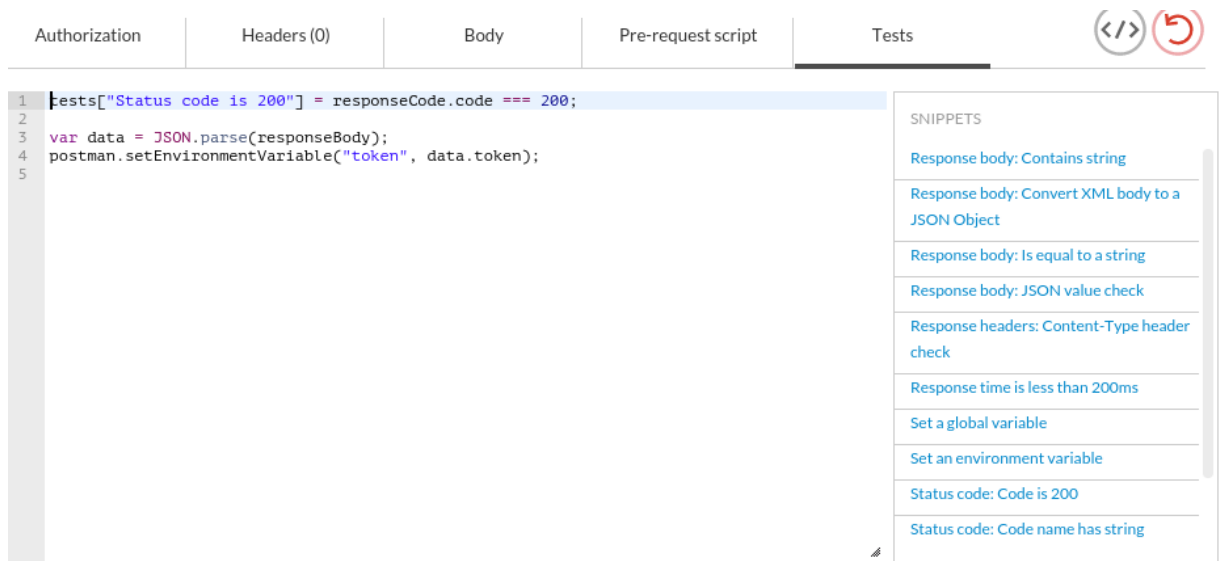
其他用法还包括在发起请求之前获取当前的时间戳放在参数里：

```
postman.setEnvironmentVariable("unixtime_now", Math.round(new  
Date().getTime()/1000));
```

当然也可以用来生成校验串。总之，在发请求之前需要手动修改的东西，都可以考虑用脚本自动实现。

2. Test

Test的编写界面如下：



和Pre-request相比，Test的Snippets就丰富多了，例如检查状态码、检查响应串、验证JSON、检查header、限制应答时间。

如果需要将服务器响应的数据保存下来，用在后面的请求里，也需要在这一步做。

在图中的Test里，我首先检查了状态码为200，然后解析返回的JSON，把环境变量里的token设为JSON里的token。

3. Collection Runner


当编写了很多测试之后，就可以使用Collection Runner来自动运行整个Collection了，入口就在主界面最上面一行的**Runner**。选好Collection、Environment，如果有需要还可以载入JSON和CSV作为数据源。点击**Start Test Run**，就可以看到结果了。

Collection runner

Test run

Stats

Integrate these tests with your build system! [Get Newman!](#)

**Newman**

Run your collections from the command line using Postman's companion utility!

[Get Newman!](#)

example

Environment example_server

10 Oct, 5:25PM

View stats

New run

注册

{{url}}/users

201 Created

131ms

PASS

Status code is 201 - 1 | 0

登录

{{url}}/authentication

200 OK

128ms

PASS

Status code is 200 - 1 | 0

添加卡

{{url}}/cards

200 OK

133ms

PASS

Status code is 200 - 1 | 0

查询卡

{{url}}/cards/{{random_cardno}}

200 OK

128ms

PASS

check cardname - 1 | 0

充值

{{url}}/deposit

403 Forbidden

114ms

PASS

Status code is 403 - 1 | 0

这里可以看到一共发起了5次请求，每个请求各有一个Test，全部Pass。（虽然最后一个请求的返回是403，但是这个请求的期望返回值就是403，所以也是Pass的）

三、示例

最后完整的看一下我用的例程。这个例子是一个非常简单的小系统，用户可以注册并登录，然后在系统里新建充值卡，并给这张卡充值。整个流程如下：

1. 注册

生成一个随机字符串作为用户名和昵称

```
postman.setEnvironmentVariable("random_username", ("0000" +  
(Math.random()*Math.pow(36,4) << 0).toString(36)).slice(-4));
```

发起请求

```
POST /index.php/users HTTP/1.1
Host: postmanexample.sinaapp.com
Cache-Control: no-cache
Postman-Token: 76791813-aac2-71fb-cad4-3e737f37c4d0
Content-Type: application/x-www-form-urlencoded

username=2mjk&password=123456&nickname=2mjk
```

运行测试、检查结果

```
tests["Status code is 201"] = responseCode.code === 201;
```

2. 登录

直接用刚才生成的环境变量发起请求

```
POST /index.php/authentication HTTP/1.1
Host: postmanexample.sinaapp.com
Cache-Control: no-cache
Postman-Token: aac7d0ac-e0e3-ecf2-39da-b8dca672e3d7
Content-Type: application/x-www-form-urlencoded

username=2mjk&password=123456
```

运行测试、检查结果，并将返回的token记录下来

```
tests["Status code is 200"] = responseCode.code === 200;

var data = JSON.parse(responseBody);
postman.setEnvironmentVariable("token", data.token);
```

3. 添加一张卡

先生成一个卡号和卡名

```
postman.setEnvironmentVariable("random_cardno", Math.round(Math.random()*9999999));

postman.setEnvironmentVariable("random_cardname", ("0000" +
(Math.random()*Math.pow(36,4) << 0).toString(36)).slice(-4));
```

然后发起请求，这里调用了刚才获取到的Token，放在header的自定义字段里作为鉴权（SAE不能用Authorization这个字段，不清楚原因）

```
POST /index.php/cards HTTP/1.1
Host: postmanexample.sinaapp.com
X-Authorization: d4c4a0b7b36c73e7a13b7e24a596093b
Cache-Control: no-cache
Postman-Token: d44d573f-f17a-366c-2cd7-1d5b8b709233
Content-Type: application/x-www-form-urlencoded

cardno=1385526&desc=2mo8
```

运行测试

```
tests["Status code is 200"] = responseCode.code === 200;
```

4. 查询刚才生成的卡

发起请求，调用了刚才生成的卡号

```
GET /index.php/cards/1385526 HTTP/1.1
Host: postmanexample.sinaapp.com
Cache-Control: no-cache
Postman-Token: 1e5aca57-c3bb-7404-2791-c639cd60b5c8
```

运行验证，和刚才生成的卡名对比，并记录新卡的ID

```
var data = JSON.parse(responseBody);
tests["check cardname"] = data.desc === environment.random_cardname;

postman.setEnvironmentVariable("new_card_id", data.id);
```

5. 充值

发起请求，使用了刚才获得的新卡ID

```
POST /index.php/deposit HTTP/1.1
Host: postmanexample.sinaapp.com
X-Authorization: d4c4a0b7b36c73e7a13b7e24a596093b
Cache-Control: no-cache
Postman-Token: 388c95e0-b5ce-9bbf-5816-084db7523384
Content-Type: application/x-www-form-urlencoded

cardid=1&amount=10
```

运行验证（由于是新建的用户，没有余额，无法给卡片充值，故返回403 Forbidden）

```
tests["Status code is 403"] = responseCode.code === 403;
```

P.S. `postmanexample.sinaapp.com` 这个网站是真实存在的，可以Import我上传的Collection(<https://www.getpostman.com/collections/96b64a7c604072e1e4ee>)到你自己的Postman中，并设置环境变量 `url` 为 `http://postmanexample.sinaapp.com/index.php`，就能运行这个Collection看效果了。

标签: [http](#), [api](#), [restful](#)
