**[[转]postman 官方文档解说](http://www.cnblogs.com/alamZ/p/6340601.html)**

**1. 安装**

两种安装方式，我热衷于以chrome插件形式安装   
[Chrome插件](https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=zh)   
[Mac App](https://www.getpostman.com/apps)

**2. 发送请求**

Postman最基础的功能就是发送http请求，支持GET/PUT/POST/DELETE，还有很多我不认识的http方法。

通过填写URL、header、body等就可以发送一个请求，这对于我们平时做一些简单的测试是够用的。

如果你的应用需要用到登录验证，可以通过填写Authorization以满足你的需求。  
另外也可以使用Chrome浏览器已经登录的cookie，同步浏览器的cookie需要安装另一个插件[Interceptor](https://chrome.google.com/webstore/detail/postman-interceptor/aicmkgpgakddgnaphhhpliifpcfhicfo?hl=zh" \t "_blank)（拦截机）。它可以在你发送请求时帮你将已经存在于浏览器的数据随header请求，另外它可以将浏览器的请求写到postman的历史中（需要开启“Request Capture”  
）。

**3. 集合**

每次配置完一个请求都可以保存到一个集合中，如此一来，下次测试可以直接从集合中找到你要执行的测试。

集合不单单只有分类和存储功能，Postman支持一键运行整个集合内的测试。

我们可以把一个请求当做一个Test Case, 那么集合就是一个Test Suite。

每个集合都对应一个URL，可以通过*Share*按钮获得你的集合URL，这个URL可以用于分享给你的队友，或者用于[Newman](https://www.npmjs.com/package/newman" \t "_blank)执行。

[Newman](https://www.npmjs.com/package/newman)是Postman的一个命令行工具，可以让API测试加入到你的持续集成任务上。

**4. 环境变量**

当做API测试时，你可能经常需要切换不同的设置。比如，开发环境的API设置、测试环境和产品环境的API设置，你可能需要在不同的测试环境下使用不同的配置。为此Postman提供了环境变量，这样你就可以通过修改环境变量，而不需修改请求了。

你可以通过右上角的下拉菜单选择环境，可以通过点击右侧的小眼睛来查看当前环境变量。

**5. API测试**

[Postman测试沙箱](https://www.getpostman.com/docs/sandbox)是一个JavaScript执行环境，可以通过JS脚本来编写pre-requist和测试脚本。pre-requist可以用来修改一些默认参数。

Postman沙箱集成了几个工具库，比如[lodash](https://lodash.com/" \t "_blank)、[SugarJs](http://sugarjs.com/" \t "_blank)、[tv4](https://github.com/geraintluff/tv4" \t "_blank)，还有一些内置函数如xml2JSON..

tv4用于验证JSON数据，通过编写JSON Schema来验证，JSON Schema的语法请[参照这里](http://json-schema.org/example1.html" \t "_blank)

测试语法：

// description 为该测试的描述

// value 只要Boolean(value)不等于false，这个测试就是PASS

tests[description] = value

// example

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

我们以[github status](https://status.github.com/" \t "_blank)的接口为例：  
url: <https://status.github.com/api/status.json>

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

// validate json schema

var schema = {

properties: {

status: {type: 'string'},

last\_updated: {type: 'string'}

}

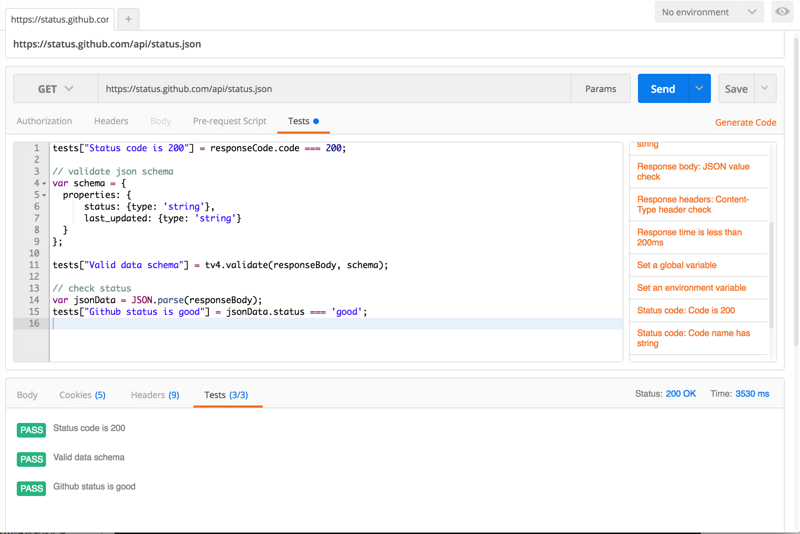
};

tests["Valid data schema"] = tv4.validate(responseBody, schema);

// check status

var jsonData = JSON.parse(responseBody);

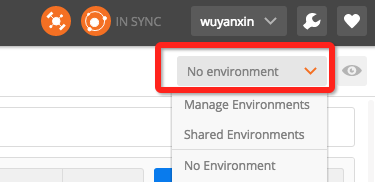
tests["Github status is good"] = jsonData.status === 'good';

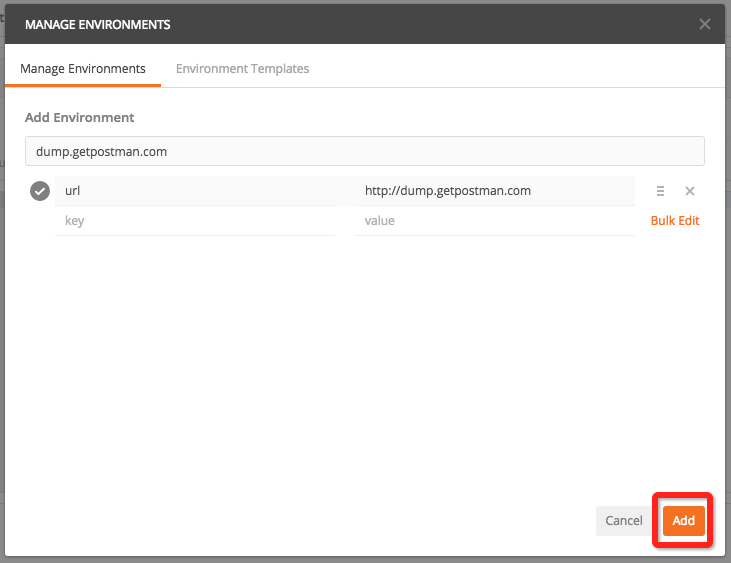
运行结果：  


**示例**

受 <http://httpbin.org/> 启发，Postman也提供了一套入门的API <http://dump.getpostman.com/> ，接下来我们将利用这套API做完整的测试。

**1. 创建一个环境变量**

 点击Manage Environments，然后点击Add

 添加一个URL变量，我们会在后续使用

**2. 请求一个新用户**

我们需要通过发送一个POST请求到{{url}}/blog/users/来创建一个用户，并需要附加下面的参数到请求body中：

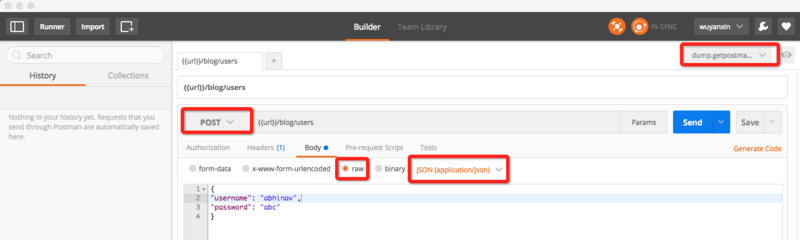
注：记得将环境变量切换到dump.getpostman.com，这样我们才能获取到{{url}}变量

{

"username": "abhinav",

"password": "abc"

}



这个接口现在好像不支持创建用户了，我们假设已经创建成功了，因为这不影响我们后续操作

**3. 获取用户的Token**

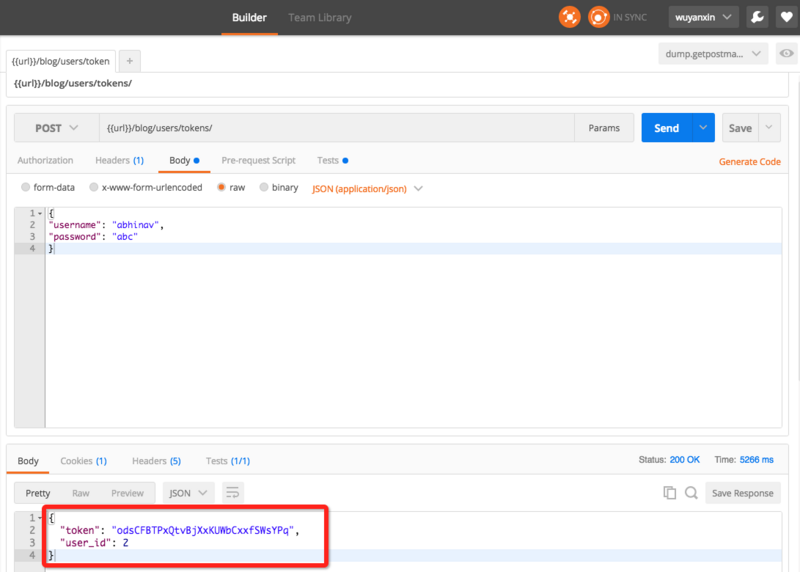
Token用于授予终端请求和访问权限的。我们可以通过POST用户名和密码请求 {{url}}/blog/users/tokens/ 来获取用户的Token，这个Token将用于其他请求中。

{

"username": "abhinav",

"password": "abc"

}



**4. 格式化JSON**

我们需要从上面的请求结果中获取到用户Token和用户ID，并将这两个值保存到环境变量中，以供后续使用。将下面这段代码添加到测试编辑器中：

var data = JSON.parse(responseBody);

if (data.token) {

tests["Body has token"] = true;

postman.setEnvironmentVariable("user\_id", data.user\_id);

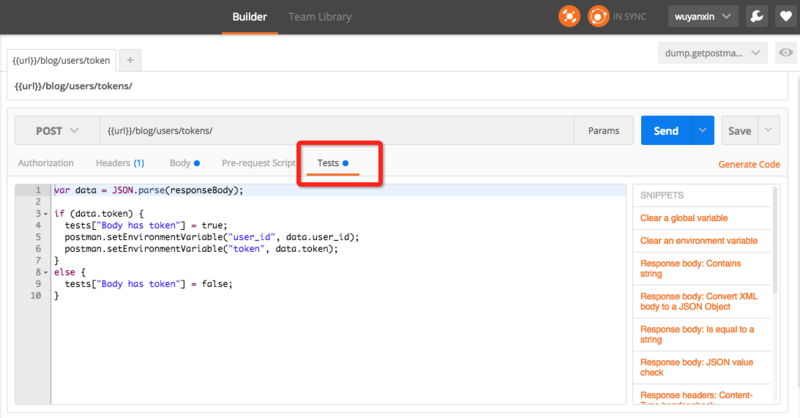
postman.setEnvironmentVariable("token", data.token);

}

else {

tests["Body has token"] = false;

}



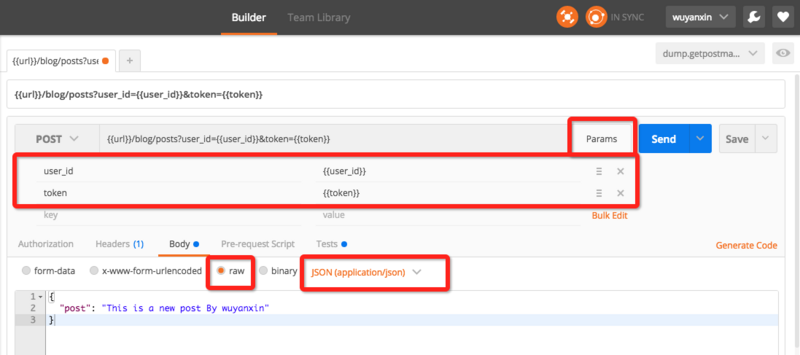
**5. 创建一篇文章**

如果上面的测试是在主窗口或者集合运行器中执行，那么 user\_id 和 token 会自动地被添加到环境变量中。  
为了创建一篇文章，我们需要发送一个POST请求到 {{url}}/blog/posts ，并将 user\_id 和 token 添加在URL参数中。POST的请求Body如下：

{

"post": "This is a new post"

}



**6. 检查返回数据**

如果上述的请求成功的话将返回一个带有post\_id的JSON。我们将在这里验证是否创建文章成功，并且将文章ID保存到环境变量。将下面这段代码添加到测试编辑器中：

var data = JSON.parse(responseBody);

if (data.post\_id) {

tests["post\_id found"] = true;

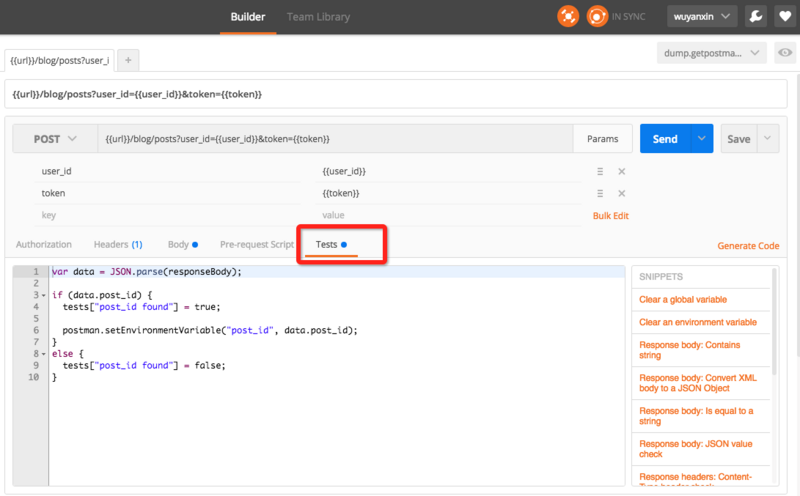
postman.setEnvironmentVariable("post\_id", data.post\_id);

}

else {

tests["post\_id found"] = false;

}



**7. 获取一篇文章并验证JSON**

我们将通过上面返回的文章ID来获取我们创建的文章。这里我们将用到Postman内置的 [tv4](https://github.com/geraintluff/tv4) JSON 验证器来检查服务器响应的JSON。  
创建一个GET请求到 {{url}}/blog/posts/{{post\_id}}，并将下面这段代码添加到测试编辑器中：

var schema = {

"type": "object",

"properties": {

"content": "string",

"created\_at": "integer",

"id": "integer"

},

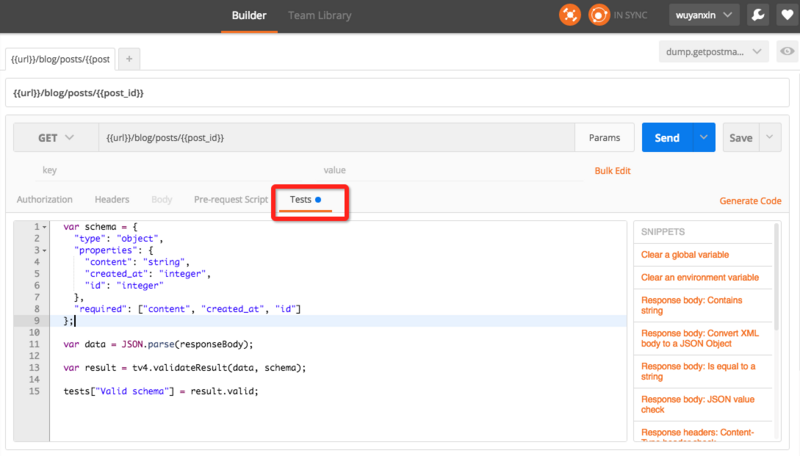
"required": ["content", "created\_at", "id"]

};

var data = JSON.parse(responseBody);

var result = tv4.validateResult(data, schema);

tests["Valid schema"] = result.valid;



**8. 一键运行与分享集合**

我们将上述每一个测试保存到PostmanTest的集合中，这样我们就可以在任何时候打开和运行你想要的测试，并且可以一键运行所有，或者将集合分享给你的小伙伴，也可以获取嵌入式代码（如下面的按钮）。  
