# Postman介绍

1. 官网地址：<https://www.getpostman.com/>
2. 官方文档地址: <https://www.getpostman.com/docs/>
3. 问题反馈官方地址：<https://go.pstmn.io/app-issues>
4. 官方git地址：<https://github.com/postmanlabs>
5. 官方实例学习：<http://blog.getpostman.com/case-studies/>
6. 官方blog: <http://blog.getpostman.com/>
7. 官方下载地址：<https://www.getpostman.com/apps>

版本分别有：

chrome插件版本

mac版本

linux版本(32/64)

windows客户端版本(32/64)

newman: <https://www.npmjs.com/package/newman>,本说明不涉及newman。

1. 介绍：

很好用的一款API/接口功能测试工具，很方便的塞数据，查看响应，设置检查点/断言，能进行一定程度上的自动化测试，但是不要指望它能代替你进行性能测试，一次批量执行太多太久的话容易崩溃掉。

有免费版、pro版、企业版三个版本。个人学习及日常工作免费版完全够用。

本系列介绍中，如无特殊说明，均以postman chrome插件版为例。

**注:**

**POSTMAN里的各种语法是基于JS的，很容易上手.**

# Postman安装

Chrom版本、客户端版本、ios版本

1. Chrom版本

方法1.Chrom应用商店直接安装，需要翻墙。

方法2.搜索下载postman 的crx扩展文件，把crx后缀修改成rar或者zip，然后解压，修改解压目录下的”\_metadata”文件夹名为” metadata”,然后打开chrom扩展页面chrome://extensions/ ，打开开发者模式，加载已解压的扩展，把刚刚解压后的扩展加载进去。

点击详细信息，在桌面创建快捷方式。

此种方法安装的postman可正常使用，但不能配合Postman Interceptor抓包使用，应该是谷歌商店不认此种方法安装的原因。

1. 客户端版本

Postman官网，下载客户端版本，直接默认安装。安装完之后会在桌面创建一个快捷方式。

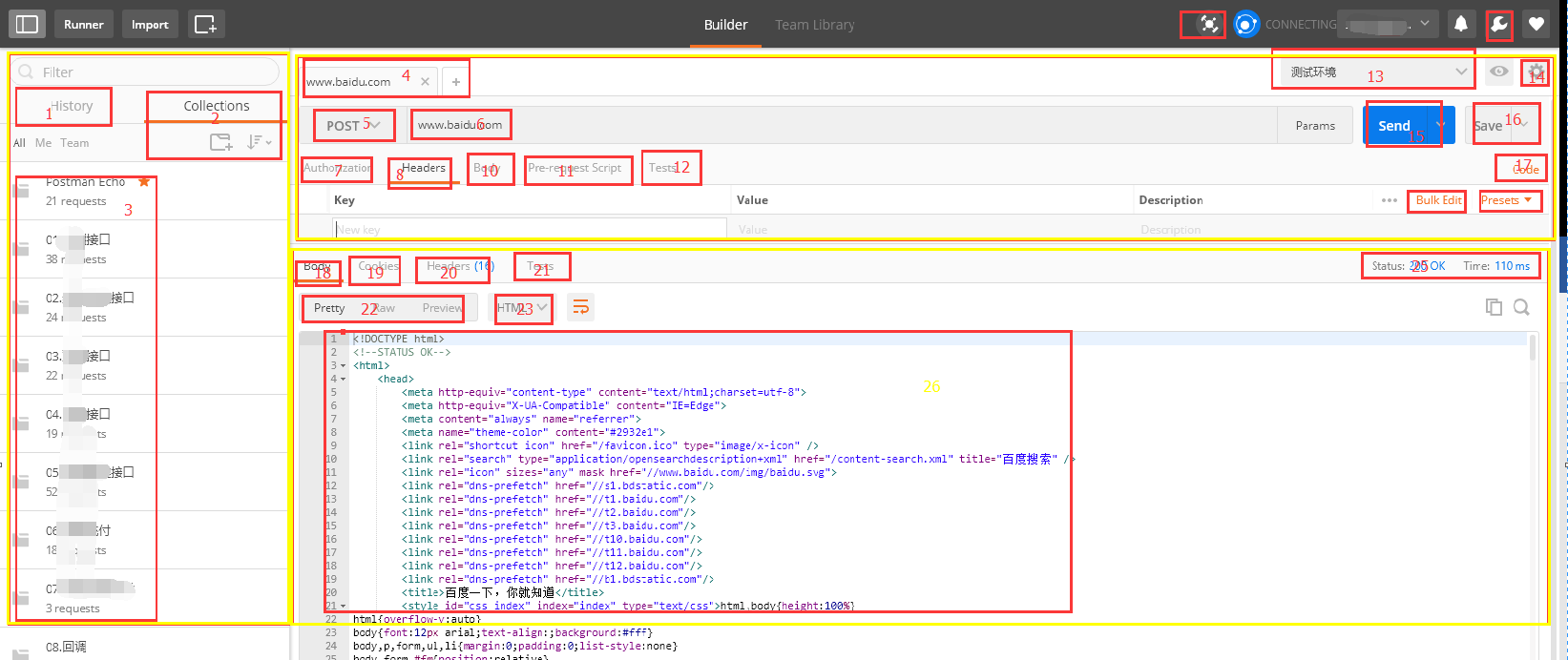
客户端版本与chrom版本不冲突，可以同时安装。客户端版本比chromapp版本多了编辑及调试功能，但是不支持请求URL中包含中文，需要encode一下才能使用。日常chrom版本够用。

1. IOS版本appstore里搜索安装。
2. linux版本未使用过，暂不介绍，欢迎补充。

**注:postman官网可直接访问，不需要翻墙。**

# Postman Sendbox界面介绍

1. 安装后首次打开postman，会提示你是否需要登录，登录的话可以云端保存你的收藏及历史记录，**不登陆不影响使用**。
2. 进入后就是如下图所示的界面了。

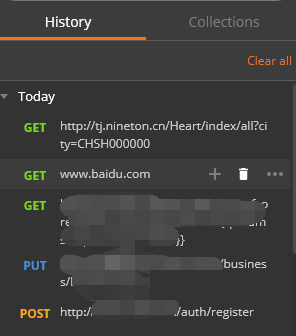


1. 我把postman主界面大概分为图示中的三个大黄框，代表不同的功能区域

## 左边:

### 1：请求历史栏

点击该处可看到所有使用postman发出的请求记录，在配合postmaninterceptor进行抓包时，抓取的请求记录也在该处显示。点击clear all可清除所有记录。

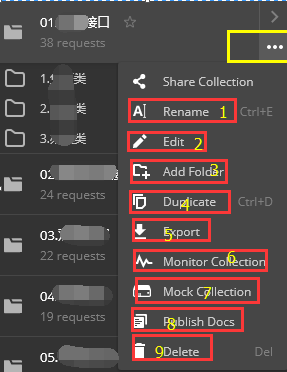


### 2.收藏夹

就是保存所有请求的地方。点击带+号那个图标 可创建新的文件夹，方便归类

### 3.用例文件夹

如图所示。需要注意的是大文件夹下还可以再创建一级子文件夹。如下图所示：



图中几个菜单的意思分别是:

1重命名

2编辑

3新建子文件夹

4复制一个当前文件夹

5导出当前文件夹下所有用例

6监控当前文件夹(免费版不可用)

7mock当前文件夹用例(免费版不可用)

8发布当前文件夹下接口用例文档

9删除

最上边那个 ShareCollection是分享当前文件夹下所有用例，Pro版才可用的，免费版的就老老实实的导入导出使用吧。本文最开始中的那个大图左上角 左数第三个按钮 Import，就是导入(这只是一种导入方法，还有一种在设置里全局导入，等下说)。



其他的按钮，用到了再说。

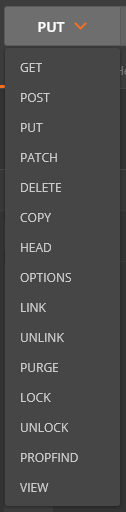
## 右上

### 4.标签栏

可以看待你当前执行的是那个接口用例

### 5.接口请求方法

目前支持: 15种请求方式，实际应用中常用到 GET、POST、PUT、DELETE这4中请求方法

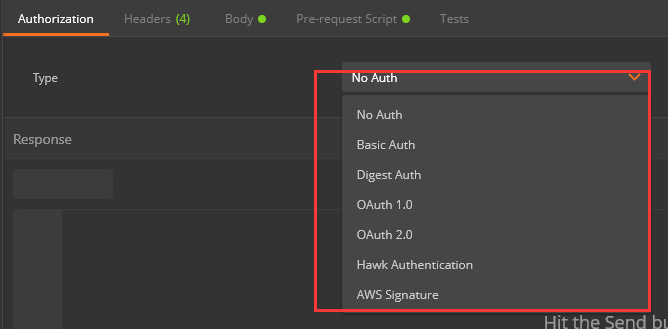


### 6.接口请求URL

请求url，需要注意的是GET请求的参数是直接跟在URL后边的。

### 7.认证

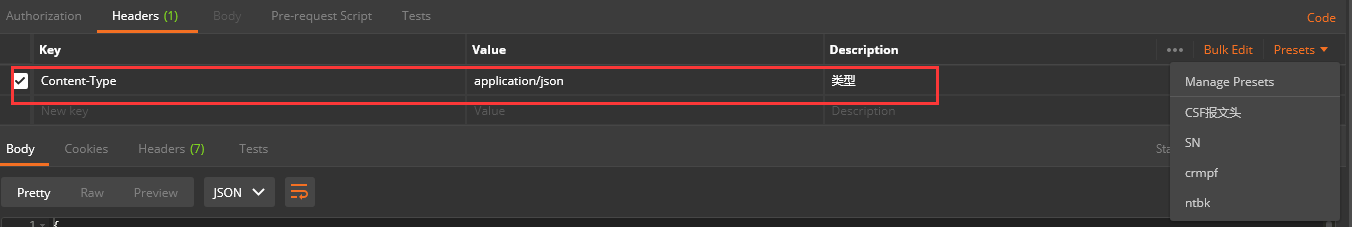
某些接口需要认证。



### 8.报文头

请求报文头。

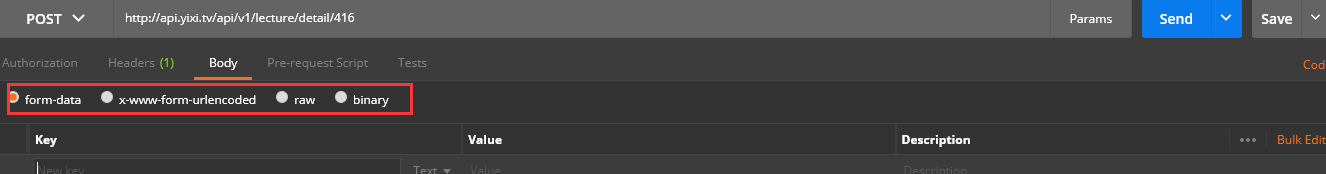
报文头有三列，Key是报文头参数列，Value是报文头参数值列，description是说明列



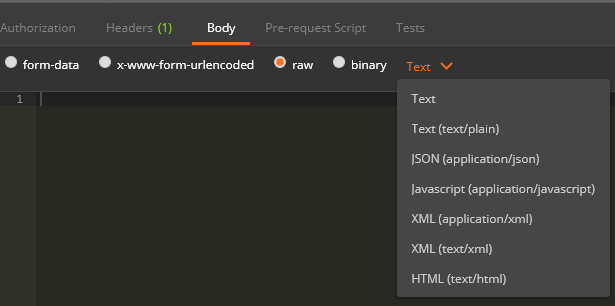
右侧的Presets是可以预先设置几组报文头，拼装报文的时候可以直接选中使用。Bulk Edit是文本编辑报文头。

### 10.报文体

请求报文体，GET请求下不可用。有四种报文体类型。



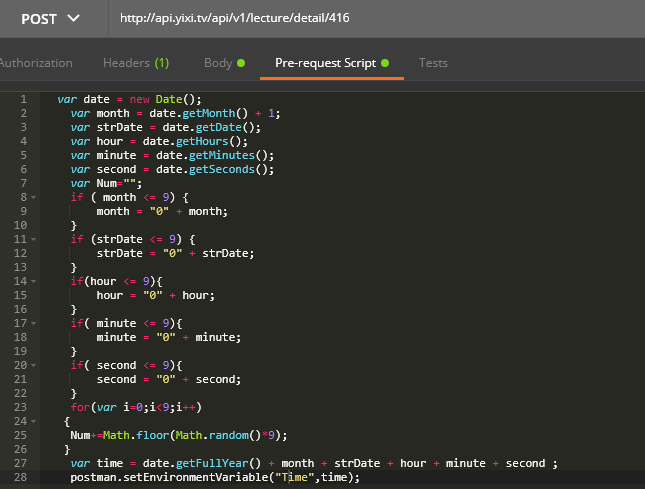
其中form-data和x-www-form-urlencoded两种类型是key-value格式的报文。Raw里可以自由选择报文格式，binary是自由选择文件作为报文体。



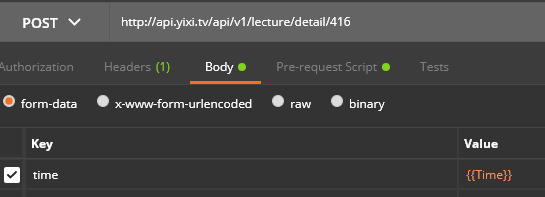
### 11.预处理脚本

预处理脚本，你可以理解为这里是在你接口请求之前执行代码的地方，支持JS语言。举例说明:

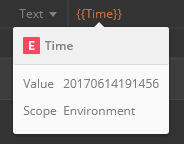
请求参数中有个参数time，参数值格式为YYYYMMDD24HHMMSS格式，每次请求时该time参数为当前时间，我们可以在预处理脚本这里直接生成一个当前时间，然后传给postman座位环境变量，在请求参数中引用这个变量，就可以了。如下图，上边是生成当前时间time，最后一句是把time存到环境变量里，这环境变量叫做Time。



接下来，我们在请求报文里(报问题、报文头都行)引入这个环境变量Time，如下图。如果环境变量中暂时没有Time这个变量，{{Time}}会是红色的。



如果环境变量中该变量已经有值，鼠标放上去，就会显示当前值。

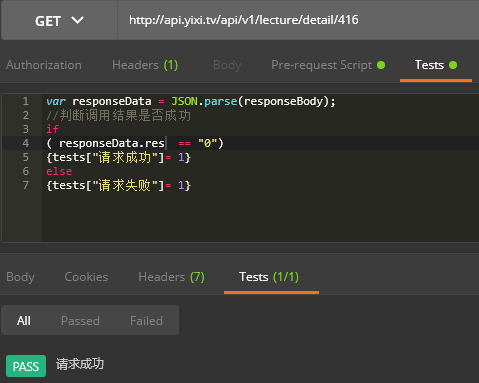


这里只是简单的说了一种使用方法，更多功能，还请自行摸索

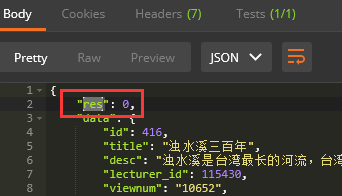
### 12.断言/检查点

有各种叫法，但是意思是一样的，可以理解为检查报文内容的地方, 支持JS语言。比如要判断响应报文中某个参数值等不等于预期值，就是在这里写代码。

Postman自带的接口用例中有示范如何使用，请自行摸索，多上网搜索，多练习，很快就会熟起来的。



下图是响应，上图中检验的就是这个res是不是等于0:



### 13.环境变量集

Postman中可以预先设置几组环境变量，使用的时候直接选择就行，比如测试、灰度、生产这三个环境的地址和请求里的参数值不一样，我们就可以把地址和参数值参数化，设置三个环境变量集，三个环境变量对应测试环境，生产环境，灰度环境。非常方便

### 14.设置环境变量

设置环境变量

### 15.发送请求

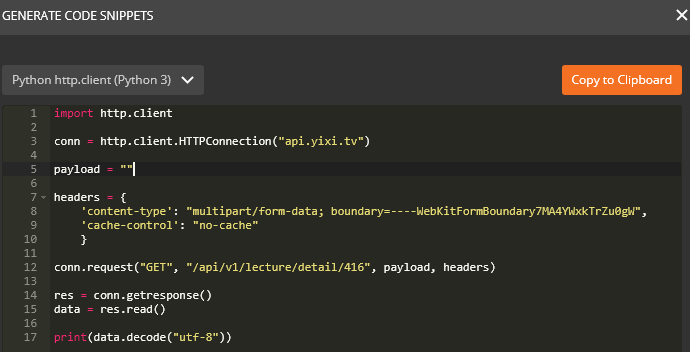
发送请求

### 16.保存

保存此时的请求报文信息

### 17.源代码

这是一个很有意思的地方，这里可以方便的把你的请求报文转换成各种程序语言，例如:



直接点击copy就能去代码中粘贴了。

## 右下

### 18.响应报文体

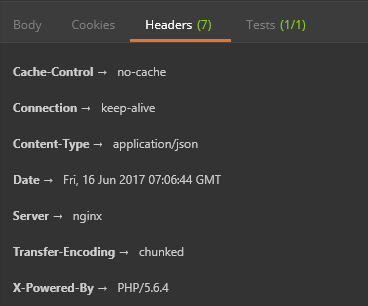
18、19、20、21这四个是响应内容选择标签，点击就能看到对应内容

### 19.返回cookies

如果有，就返回。

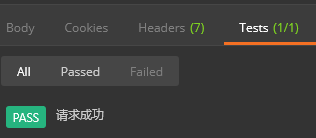
### 20.返回报文头

如下图所示:



### 21.断言/检查点执行结果

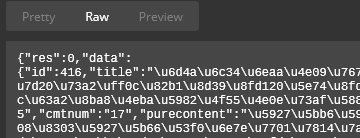
12中的断言、检查点执行结果展示区域，如下图所示:

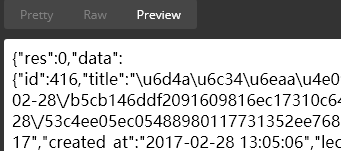


### 22.报文体展示方式

下图示例相同报文三种展示方式，一般使用pretty。

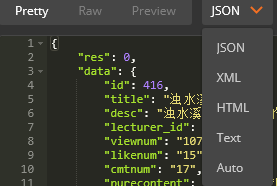






### 23.报文体展示类型

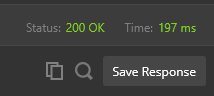
仅在Pretty模式下可用



该图标推测为自动换行图标~

### 24.报文体正文

### 25.httpstatus响应码及耗时



Httpstatus及请求耗时，客户端版本的postman这里还会多出一个响应报文大小。

为复制图标，点击就复制响应报文体至剪切板，

为搜索图标，可以在响应报文中搜索制定内容。

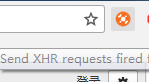
## 补充

### postman interceptor

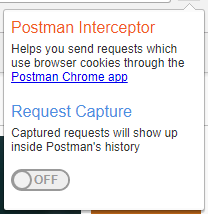
这是另外一个插件，作用是用来配合postman进行抓包，对，你没看错，就是抓包，但是只能抓请求数据，响应数据不能获取。该插件需要从chrome 应用商店安装才能正常使用，使用加载离线安装包方法安装的 该扩展不能正常抓包，即在postman中不能正常使用

下面说这个扩展的用法：

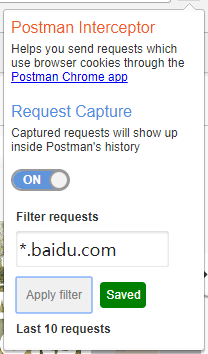
1. 安装好这个扩展后，首先在chrome浏览器扩展栏会出现这个图标



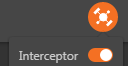
1. 点击该图标，在点击off切换到on状态



1. 在Filter requests 栏里输入自己想要抓包的域名，支持\*字符，如下图所示，代表所有请求url中包含baidu.com的请求都会被记录。最后记得要点 apply 应用保存。这个保存仅能应用本次抓包，如果关掉浏览器再次打开的话需要重新配置。

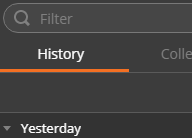


4.打开postman，点击右上角的interceptor图标，切换到on状态



当处于on状态时，图标是变橙色的。

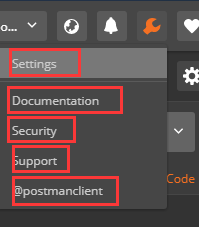
1. 打开请求记录，

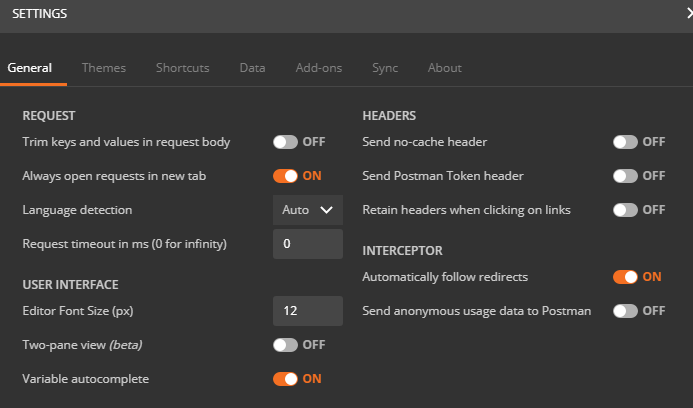


当访问想要抓包的网页时，请求记录会被记录在这里。

### 设置

右上角的小扳手就是设置按钮





要注意的一点是，chrome版本和client版本的设置里有些许不同，client版本的设置里会多一点点内容，但是大致是一样的，此处以chrome版本为例说明。

点击小扳手，选择settings(其他几个按钮分别是结果连接：文档、安全、支持、facebook‘反正你也上不去’)

新弹出的就是设置页面。从左至右几个标签依次为：通用设置、主题、快捷键、数据、扩展(NODE-js)、同步、关于。

1. 通用设置

可以看到通用设置标签 有行大写的字母，分别是 REQUEST(请求)、HEADERS(报文头)、USERINTERFACE(用户界面)、INTERCEPTOR(拦截器)

每个按钮是干啥用的我就不一一翻译了，你们google就行了......

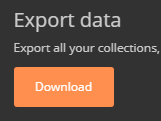
2.主题

白天、黑夜

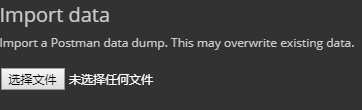
3.快捷键

4.数据

敲黑板了，划重点了，这里是postman数据导入和导出的地方，记住，是所有数据，包括你所有的的collection和environments。

export，导出，备份，随你咋叫都行。导出的是个json文件，如果有批量修改的用例信息，比如你所有接口的IP变更了，就可以在导出的json文件里ctrl+H批量修改。

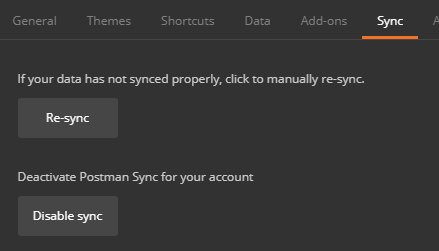
Import这里是导入数据的地方。



1. Adds-ons

请自动忽略这个

1. 同步数据到云端，前提是你有一个postman帐号。



Re-sync,立即同步；Disabel sync，禁止同步。

1. 关于

当前postman的版本信息及其他相关信息，在给postman提交bug时可以从这里复制。

# Runner 介绍

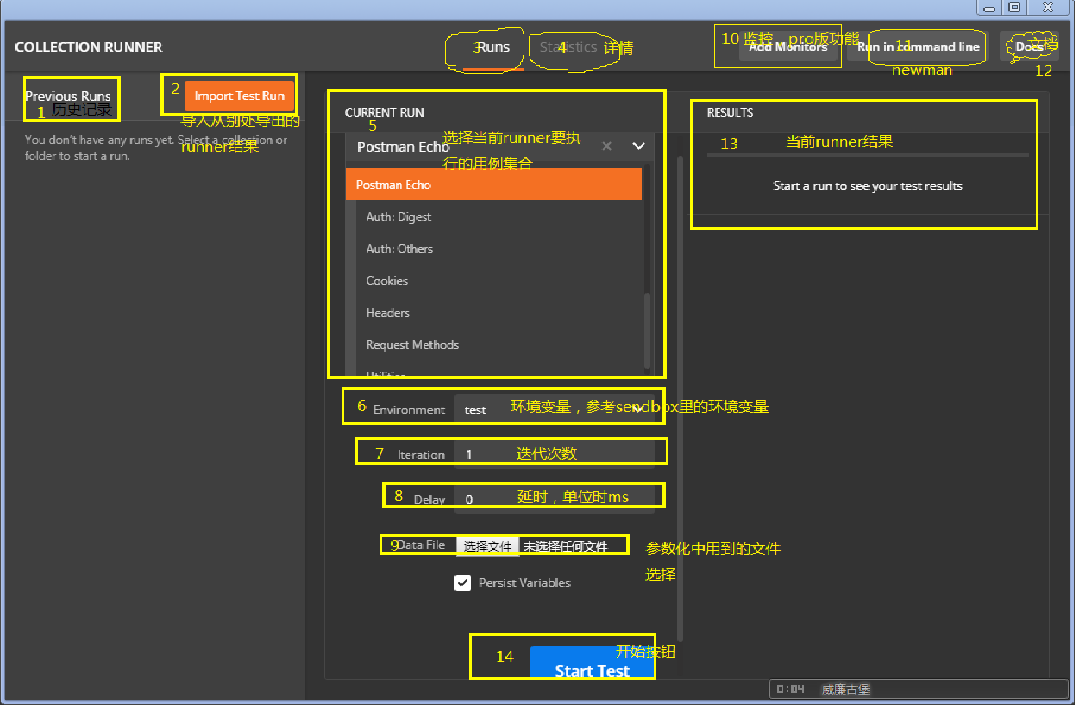
## 概述

Runner是用来批量调用collection里某个文件夹里的全部接口的。

(注意，是文件夹内所有接口,可以是一级文件夹，也可是二级文件夹)

## 示意图

在Sendbox界面里左上角打开runner，runner界面如图所示



## 说明

### 历史记录

历史执行记录

### 导入

导入别人或之前导出的runner结果以方便查看。



### 当前执行

### 执行详情

点击这个会跳到历史记录

### 5.选择用例集合

选择需要执行的用例集合

### 6.环境变量

选择执行时需要使用的环境变量

### 7.迭代次数

输入需要执行的次数

注意:在使用csv文件进行参数化时，最好有多少组数据，这里就填多少。

### 8.延时/间隔

每个请求之间的间隔

### 9.参数文件

在批量执行时，如果遇到想参数化的情况，在此处配置。举例:

一个登录请求为:http://127.0.0.1:80/login,请求方式为post，请求data为:

{

"account": "zhangsan",

"pwd": "123"

}

其中account和pwd分别是帐号、密码，在实际中，有一大批帐号需要登录，这时可以利用runner的参数化来解决。在本例中，需要把account和pwd的值参数化，首先在请求时先定义这两个值为分别变量{{user}}和{{密码}}如下:

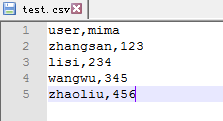
{

"account": "{{user}}",

"pwd": "{{mima}}"

}

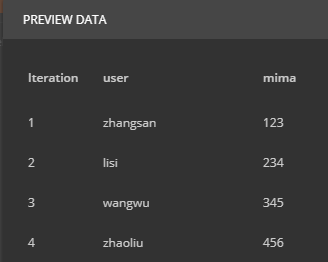
然后创建一个csv格式的文件，文件首行为两个变量名，其余各行值，如下图所示:



然后选中这个csv文件



点击preview，可以看到参数信息



，然后就可以开始了。

### ~~10.监控~~

~~Pro收费版本才有的功能，本文不作介绍~~

### ~~11.newman~~

~~本教程不做介绍~~

### 12.doc文档

Runner的相关官方英文文档。

### 13.当前结果

会显示tests的执行结果

### 14.开始按钮

# 常用操作说明

## 环境变量

环境变量就是我们存放一组公共数据的地方，比如我有100个接口，三套环境(开发环境、测试环境、生产环境)，我们没有必要把这100个接口根据三个环境分别写一套用例。只需要维护一套用例脚本，然后设置对应三个环境的变量就行了。

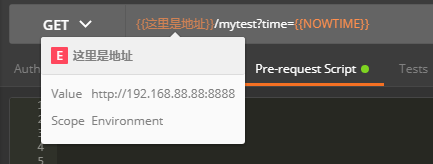
如下图所示：





这样对比应该能看出，我们在请求接口的时候，只需要把请求地址用参数 {{这里是地址}} 替换掉就行了。在测试的时候，测哪个环境就切换到哪个环境的环境变量组。

下图是我的请求：



另外，在postman的脚本里，还可以用postman.setEnvironmentVariable("key",value); 来直接设置环境变量，下面就会说到。

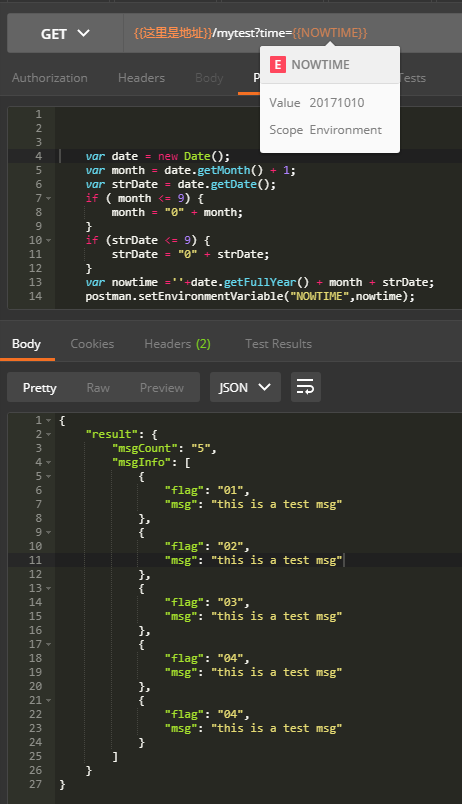
## Pre-request Script

顾名思义，这是一个在请求前执行脚本的地方。

举例说明：

还是刚刚那个接口，请求时需要带上YYYYMMDD格式的当天日期作为参数值。那么我们就可以在这里用JS语法生成一个当前时间nowtime，然后把这个nowtime用上边说道的postman.setEnvironmentVariable(key,value)设置为环境变量NOWTIME，在请求的时候，在引用这个变量NOWTIME就行了。

如下图所示是我的测试接口



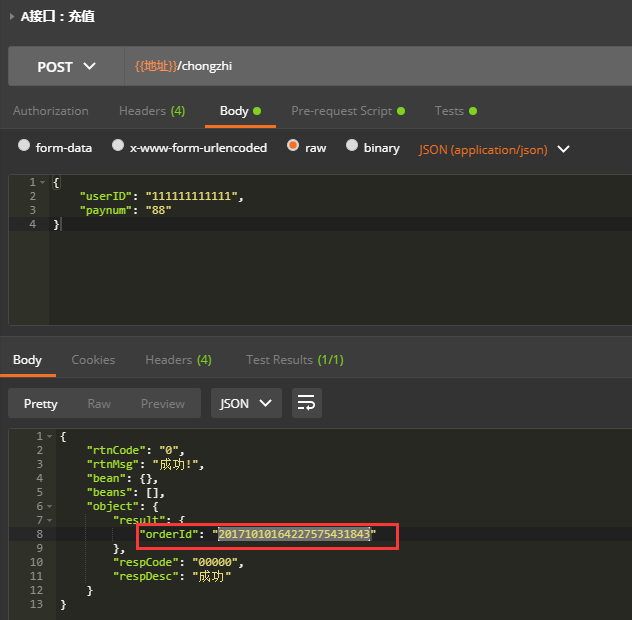
## 关联

这是我被问到最多的一个问题......

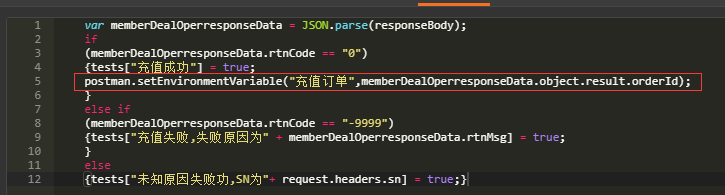
举例说明：

现有A、B两个接口，A接口是充值接口，充值成功会返回一个充值订单；B接口是充值订单查询接口，需要用A接口返回的充值订单作为请求参数去查询充值结果。

如下图所示



现在B接口需要用A接口返回的 "orderId": "20171010164227575431843" 作为入参去查询，那么我们可以在A接口的tests里这么写：



这样，在充值成功时，就会自动把返回的orderId写进环境变量{{充值订单}}，我们在B接口请求的时候，直接引用这个变量就行了。

