接口测试资料

深入浅出的解析session和cookie

·为什么需要session和cookie？

·http协议本身是一种无状态的协议，在web1.0时代，是完全适用。

进入web2.0时代，强调的是交互。需要记录操作的状态，才能保证流程的正常进行。

为了实现状态的记录，就引入了session和cookie。

·session和cookie的特征：

·session和cookie都是由服务器生成，都是键值对应的。

·session是存储在服务器上，而cookie是返回给客户端，由客户端保存的。

session的ID：服务器用来读取、标识特定的session的对象的标记。

sessionID一般会以cookie的形式返回给客户端。

·session和cookie都是有其生命周期的。

·cookie的生命周期由两个因素决定：

·服务器生成cookie的时候所指定的存活时间。

·客户端是否保留cookie文件。

PS：客户端在发送请求的时候，会自动将存活的cookie封装在请求头中向服务器发送。

请求头格式为：

Cookie：key1=value1,key2=value2

cookie的生命周期只对客户端有效，对封包工具（接口、性能测试工具）无效。

·Session的生命周期也是由两个因素决定：

·服务器对于session对象的存活周期的设定。

·客户端连接是否关闭。

sessionID虽然是以cookie形式返回，但是不会写入客户端的文件，而是客户端的进程绑定，这就意味着一旦客户端进程关闭，则客户端就会丢失所持有的sessionID。

PS：客户端连接关闭，只对客户端有效，对封包工具（接口、性能测试工具）无效。

·session和cookie都是有作用域的。

·为什么session比cookie要安全？

session的值是存储在服务器上，服务器如何对之进行校验，对于客户端来说是完全不可见的。

cookie的值是返回给客户端的，虽然说客户端也看不见服务器的校验逻辑。

但是这并不影响客户端（攻击者）根据cookie的键值对服务器校验逻辑做出推测，进而伪造cookie，达到攻击服务器的目的。

·接口测试核心：URL&HTTP协议详解

·URL：统一资源定位器，通常来说是由五个部分构成：

·protocol：协议

·domain:域名

·port：端口

·path：路径

·URL Parameters：URL地址参数。

示例：https://ke.qq.com/course/317690?tuin=15945f87

·协议：通常就是://之前的部分，协议是指通信双方对于通信的数据的组织格式、含义、规程等所做的约定。

一般来说，对于接口测试而言，最重要的就是协议。一般接口测试都是从应用层的协议进行模拟发送数据。

常见的应用层协议有：

http

https：http+ssl

ftp：

ssh：

smtp：

pop3：

oracle：

mysql：

sql server：

·域名：域名就是指要访问的服务器的地址，可能是域名、计算机名、IP地址。

比如说：

ke.qq.com

www.baidu.com

14.215.177.39

·端口：端口是跟在域名之后，是由服务器（服务器软件）来进行设置的。端口是服务器用来监听、接收客户端的请求的。

格式：“域名：端口”

如果端口是通信协议的默认通信端口，则用户在通过url访问服务器时，是可以省略不写的。

常见的协议及其默认通信端口：

http： 80

https：http+ssl 443 or8443

ftp： 21

ssh： 22

smtp： 25

pop3： 110

oracle： 1521

mysql： 3306

sql server： 1433

·路径：就是要访问的资源or服务在服务器容器路径下的地址。

一般来说，路径体现的就是业务的逻辑。

·URL地址参数：通常来说就是指以问号？作为连接符，连接在path之后的部分。

PS:

·URL地址参数的数据组织格式为：键值对应，即key1=value1&key2=value2&...&keyN=valueN

·在很多工具中，URL地址参数是直接算在path部分的。

·http协议：hypetex transfer protocol、超文本传输协议。

·http协议的特点：

·http协议是一种基于request（请求）和response（响应）的协议。

·http协议是一种简单、灵活的协议。

·http协议是一种快速、无连接的协议。

http1.0及以前默认是无连接（短连接）。

http1.1及之后默认是长连接的。

短连接：是指一个tcp连接上只能建立、完成一个http连接。

长连接：是指一个tcp连接上可以建立、完成多个http连接。

标志：通过信息头Connection：keep-alived

·http协议是一种无状态的协议。

后来引入了session和cookie机制来实现状态的记录。

·准备工作：准备一张抓包工具，看数据时建议直接看raw或者source的数据，这样的数据是原始数据，没有被抓包工具解析过的。

·http协议详解:

http协议是由两个部分构成：http request、http response。

了解和掌握http request是我们去实现接口测试的重要基础。

了解和掌握http response是保证我们的接口测试脚本准确性的基础。

·http request：http请求，通常来说是由三个部分构成：

·request line:请求行

·request headers：请求头

·request body：请求主体。

·request line:请求行、通常来说就是请求数据包中的第一行内容，包含以下信息：

request method、request path、 protocol/version

示例：

GET /phpwind/ HTTP/1.1

·request method：请求方法，一个http请求是必须有请求方法的，如果不指定，则默认为get方法。

常见的请求方法有：

get、post、put、delete、patch、options、trace、header等。

接口所使用的请求方法不是由测试决定，而是由应用、接口本身决定。

对于浏览器应用而言：

get方法所传递的参数一般都是在URL地址参数部分，反之亦然。

post方法所传递的参数一般都是在request body部分。

·request path：请求路径，即URL中的路径部分。

·protocol/version：协议/版本，请求和响应是必然一致的。

·request headers：请求头、是指从请求数据包的第二行开始，到第一个空行截止的部分。

一般来说，请求头是客户端用来和服务器进行通信、控制信息的交互的，和业务通常无关（状态信息头除外：cookie、author、token、csrf）。

请求头是键值对应，请求头的名称是由http协议规定的，不同的请求头代表不同的含义。

重要的请求头：

·User-Agent：客户端用来告知服务器客户端的配置信息，通常和cookie、session等状态信息的生成有关。

建议设置为一个统一的全局的信息头。

·Cookie：状态类的信息头。

·Content-Type: 是客户端用来告知服务器，所发送的请求主体（request body）的数据组织格式。

不同的请求，该信息头的值可能不一样。

要求：要保证Content-Type的值和实际发送的内容的一致性。

·request body：请求主体。是指请求数据包中从第一个空行开始到最后的所有内容。

请求主体就是由客户端（业务）向服务器发送的数据部分。

PS：只要请求主体有值，一定要记得添加信息头：Content-Type

·http response：http响应，通常来说也是由三个部分构成：

·response line：响应行

·response header：响应头

·response body：响应主体

·response line：响应行、通常来说就是指响应数据包中的第一行内容，通常包含以下信息：

protocol/version、response code、response message

示例：

HTTP/1.1 200 OK

·response code：响应代码，又叫状态码。是服务器用来告知客户端，服务器对于请求的逻辑处理情况。逻辑处理仅仅是指网络通信层面的处理，和业务无关。

响应代码通常是三位长度的数字，一般根据首位数字的不同，可以将状态码分为以下几类：

·1xx：表示通信建立过程中所传递的信息。

·2xx：典型的就是200，表示服务器处理成功。

·3xx：表示重定向。

PS:1xx、2xx、3xx表示客户端和服务器的连接通信是成功的。

·4xx：客户端错误

·5xx：服务器错误

从接口和性能测试的角度来说，出现4xx和5xx，则说明脚本一定是没有成功实现业务的。

一旦出现4xx和5xx错误，则建议根据成功的业务的抓包来进行对比分析。

·response message：响应信息，是用来描述响应代码的，没有任何实际作用。

·response header：响应头、是指从响应数据包中的第二行开始到第一个空行截止的部分。

类型、作用、格式等都和请求头类似，区别只是在于是服务器响应给客户端的。

·response body：响应主体、是指响应数据包中从第一个空行之后的所有内容。

通常来说就是服务器对于请求的处理结果。

**1、你们公司的接口测试流程是怎样的？（有没有感觉熟悉，貌似在哪里听过）**

接口测试我们是在XX项目做的，主要有XX接口，XX接口，XX接口等。  
1、首先是从开发那里拿到API接口文档，了解接口业务、包括接口地址、请求方式，入参、出参，token鉴权，返回格式等信息。  
2、然后使用Postman或Jmeter工具执行接口测试，一般使用Jmeter的步骤是这样的：  
     1、首先新建一个线程组。  
     2、然后就是新建一个HTTP请求默认值。（输入接口服务器IP和端口）  
     3、再新建很多HTTP请求，一个请求一个用例。（输入接口路径，访问方式，参数等。）  
     4、然后创建断言和查看结果树。  
3、最后调试并执行用例，最后编写接口测试报告  
4、其实我们做接口的时候也碰到了蛮多的问题，都是自己独立解决的，比如返回值乱码（修改jmeter的配置文件为UTF-8编码方式），比如需要登录后才能取得token鉴权码并且这个鉴权码在下面的请求中需要用到（使用正则表达式提取器提取token的值等。

**2、请简述一下cookie、session以及token的区别（有没有感觉整个是万年不变的面试题）**

（1）cookie数据存放在客户的浏览器上，session数据放在服务器上。而token是接口测试时的鉴权码，一般情况下登陆后才可以获取到token，然后在每次请求接口时需要带上token参数。

（2）cookie不是很安全，别人可以分析存放在本地的cookie并进行cookie欺骗，考虑到安全应当使用session，session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面应当使用cookie

（3）可以将登陆信息等重要信息存放为session；其他信息需要保存，可以放在cookie

**3、接口测试如何设计测试用例？（必问）**

接口测试一般考虑入参形式的变化和接口的业务逻辑，一般设计接口测试用例采用等价类、边界值、场景法居多！

接口测试设计测试用例的思路如下：  
1.接口业务逻辑测试?(正例)  
    接口逻辑测试是指根据业务逻辑、输入参数、输出值的描述，对正常输入情况下所得的输出值  
    是否正确的测试，也就是测试对外提供的接口服务是否正常工作。  
2.模块接口测试?(反例)  
    模块接口测试是为了保证数据的安全及程序在异常情况下的逻辑的正确性而进行的测试。?  
    模块接口测试的主要包括以下几个方面：?  
    1）鉴权码token异常(鉴权码为空<没有鉴权码>,错误的鉴权码,过期的鉴权码)。  
    2）其他参数异常。  
        1、必填项检查  
        2、参数的长度、类型、格式异常：  
                常规参数：(数字、字符串、日期)  
                参数长度：6-18位。或身份证、电话的长度。  
                参数类型：数字(精度)，字母，中文，带空格的参数，特殊字符。  
                日期格式：日期：年月日，年月日时分秒，日期格式(包括/,-,:等)。  
    3）错误码异常覆盖。  
    4）接口测试其他的关注点  
        接口有翻页时，页码与页数的异常值测试  
        数据库的增删改查，比如一个post接口操作完成后，通过列表页接口看下新的数据是否和刚才的post一致  
        接口返回的图片地址能否打开，图片尺寸是否符合需求  
        当输出参数有联动性时，需要校验返回两参数的实际结果是否都符合需求。  
        所有列表页接口必须考虑排序值  
        所有功能都要考虑兼容旧版本

**4、接口测试执行中比对数据库吗？**

肯定啊，因为接口返回值的数据来源于数据库，接口对数据的操作还要进行深层次的数据库检查！

**5、谈谈你对HTTP协议的了解？**

超文本传输协议，端口为80，特点（无记忆功能、快速）是由请求和响应两部分组成请求由请求头、请求行、请求正文组成；响应是由响应头、响应行、响应正文组成，之前我们公司的接口是采用https协议的。

https http+ssl协议 端口443 面向安全的超文本传输协议

**6、get和post请求有什么区别？**

get和post请求都是客户端向服务器提交的一种请求方式

get是明文传输参数、倾向于请求服务器资源。比如打开网站

post传输数据不可见，安全性高，倾向于向服务器提交数据，比如注册等

**7、响应状态码有哪些？**

1xx：指示信息--表示请求已接受，继续处理

2xx：成功--表示请求已被成功接收、理解、接受

3xx：重定向--要完成请求必须进行更进一步的操作

4xx：客户端错误--请求有语法错误或请求无法实现

5xx：服务器端错误--服务器未能实现合法的请求

**8、在接口测试过程中发现的bug多不多？能举几个栗子？**

这个问题其实回到起来很简单，只要做过接口测试的，总能发现几个BUG吧，把你平常发现的bug说2-3个就可以了。 面试官出这个题，主要是想知道你是不是真的做过接口测试，毕竟现在很多小伙伴简历都是写的假的（你要不写估计面试机会都没有，没办法，为了生存，能理解） 比如，提现输入框，在页面上输入负数，肯定是无法提交过去（前端页面会判断金额），如果我不走前端，直接用接口工具发请求，输入一个负数过去。 （假设服务端没做提现金额数据判断） 余额=当前余额（100）-提现金额（-100），那么提现-100，余额就变成200了，也就是越提现，余额越大了。

**9、接口测试中有哪些要注意的测试点？**

11.1)接口中返回了图片地址，要手工去进行图片的测试（大小、内容）

11.2)接口完成查询功能的时候，数据返回的排序显示

11.3)接口测试的时候，关注参数的默认值、必填项

**10、接口执行测试后返回结果做对比，一般比对哪部分内容？**

之前必须要对比的就是返回状态码，其次再去对比返回其它关键内容

**11、为什么开展接口测试？**

13.1)接口测试属于集成测试、测试介入越早、就越能在项目早期发现问题，其修复问题的成本越低

13.2)接口测试非常快速、UI自动化执行一个测试用例10S左右、接口测试用例执行的话，需要的时间是毫秒级的

**12、之前在接口测试过程中，使用的工具是什么？**

postman或jmeter(5.1)

**13、之前用过抓包工具没有？如何使用的？**

之前在项目中用过fiddler抓包工具进行HTTP协议请求的抓取

打开fiddler之后，默认浏览器配置了127.0.0.1  8888端口的代理，在fiddler设置好过滤策略后，打开需要进行抓包的网站进行操作，就可以进行抓包

**14、json数据是什么，你平时如何解析json数据？**

一种开发常用的数据报文格式，由键值对和数组两种格式构成。可以通过工具bejson网站等

**15、postman你在工作中使用流程是什么样的？**

1) 编写好用例

2) 在postman先建好url环境变量

3) 根据接口用例所属的模块新建集合管理

4) 在集合中不同模块下录入测试用例

5) 录入测试用例的时候根据预期结果在tests页签中增加断言

6) 导出通过Newman去运行

**16、postman中设置环境变量有什么用？**

在之前项目中，接口测试测试的环境有开发环境，测试环境等，为了测试的时候方便，就在postman设置环境变量，到时所有接口都引用该环境变量，这样就不用为了切换环境导致每次都去修改被测系统接口的主机地址；点击右上角环境变量管理按钮-新建环境变量，在脚本中使用{{变量名}}去调用

**17、在接口测试中关联是什么含义？如何用postman设置关联？**

关联就是把上一个接口返回值的部分截取出来，作为下一个接口的参数，能让接口串联运行

在postman中设置关联的步骤如下：

1) 先通过正则表达式提取的方式或json取值的方式把下一个接口需要的信息从上一个接口截取出来

2) 使用设置全局变量的代码把取出来的值保存到全局变量

3) 在下一个接口中，使用{{全局变量}}代替要替换的静态值

**18、postman支持什么类型的协议测试？**

http和https协议的

**19、postman参数化有哪几种方式？**

内建变量、pre-scripts编写js脚本、批量运行时导入csv或json格式的文件

**20、Newman如何执行postman脚本？**

Newman run 脚本名称 也可以添加参数生成html报表等

当然，以上功能也只是展示了postman的一部分功能，为了能更加全面的体现这款工具的特点，我将从以下三个维度来加以说明。它们分别是：

* 基础功能
  + 常见类型的接口请求
  + 接口响应数据的解析
  + 接口管理(Collection)
  + 批量执行接口请求
  + 日志调试
  + 断言
  + 变量
  + 请求前置脚本
  + 接口关联
  + 常见的返回值获取
* 便捷功能快
  + 速填写查询参数
  + 快速填写请求头信息
  + 快速实现添加一个请求
  + 如何继承集合认证
  + 批量断言
  + 快速查询和替换
* 高级功能
  + 读取文件进行参数化
  + 生成测试报告
  + 代码中发送请求
  + 编写接口文档
  + mock服务
  + 监控
  + 使用工作空间
  + 代码同步与分支管理
  + 连接[数据库](https://cloud.tencent.com/solution/database?from=10680" \t "_blank)
  + apis

**一. postman安装说明**

postman 在 2018 年之后就不再支持浏览器版本，所以，想要使用它就必须先下载客户端再安装使用，下面就以 Windows 系统为例进行安装 。

**1. 下载与安装**

postman安装步骤：

1. 访问postman官方网站，下载最新版本

访问地址：https://www.getpostman.com/

2. 进入到下载页面，根据自己电脑下载对应的版本

图片

3. 双击下载的安装包，进入到安装界面，直到用户登录和注册界面。

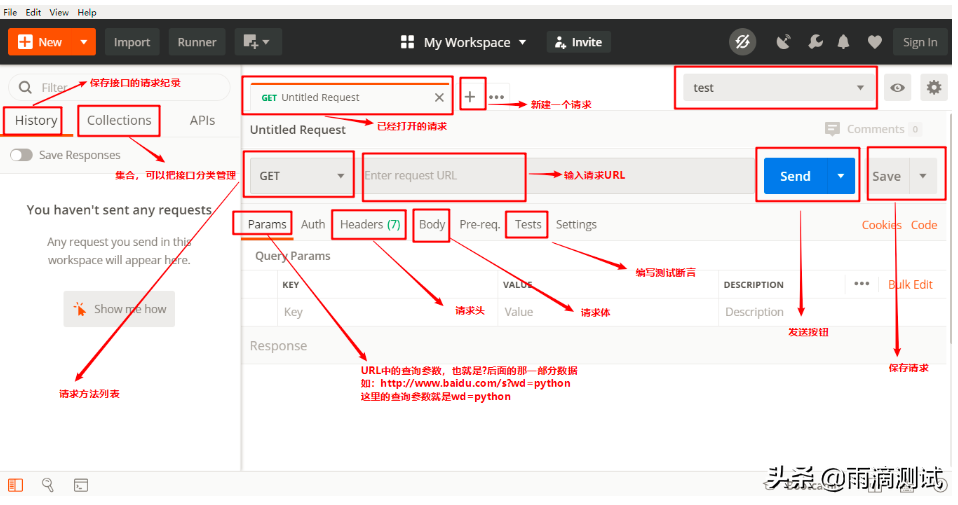
若个人使用，选择跳过即可，这时会进入到postman主界面，至此postman安装成功(下图)。

若团队使用，可以进行注册，注册后使用账号可以加入团队工作区

图片

**2.界面导航说明**

初次使用postman的朋友可能对界面上的一些元素不太熟悉，下面就通过一张图来说明这些元素的含义 。



**3.发送第一个请求**

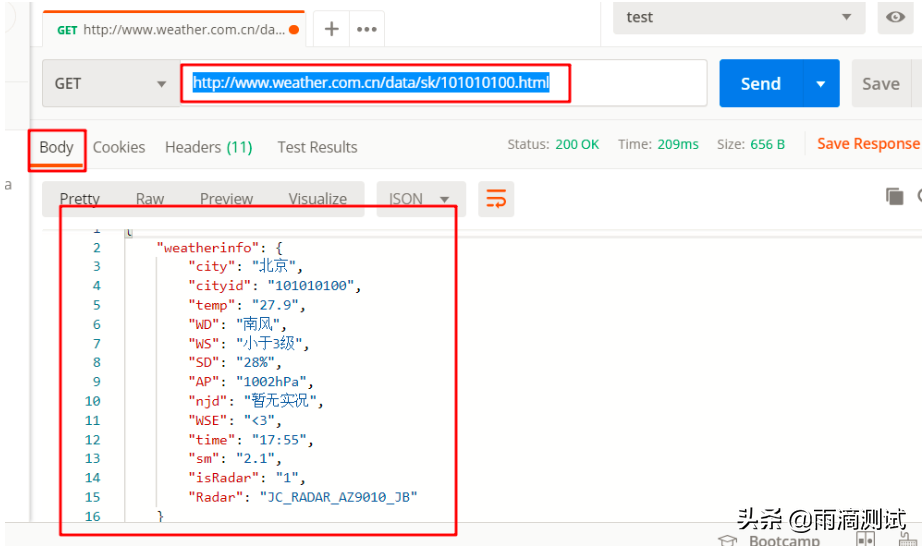
如果你是第一次使用postman发送请求，下面这个例子可以作为一个最基本的入门，可以帮我们建立一个初始印象 。

1. 打开postman，点击+加号打开一个新的请求页。

2. 在请求的URL中输入请求地址：

http://www.weather.com.cn/data/sk/101010100.html

3. 点击Send按钮，这时就可以在下部的窗格中看到来自服务器的json响应数据。



**二.postman基础功能**

**1.常见类型的接口请求**

常见的接口有如下四种类型，分别是含有查询参数的接口，表单类型的接口，json类型的接口以及含有上传文件的接口，以下就对这四种类型接口及如何在postman中请求进行说明 。

**1.1 .查询参数的接口请求**

**什么是查询参数?**

所谓的查询参数，其实就是URL地址中问号（?）后面的部分就叫查询参数，比如：http://cx.shouji.360.cn/phonearea.php?number=13012345678 。在这个接口中，查询参数就是:number=13012345678 。

而这一部分是由有键值对组成，格式为：key1=value1&key2=value2, 如果有多组键值对，要用&隔开 。

**postman如何请求**

在postman中实现对这类接口请求非常简单，一般就需要明确俩个参数即可，一个是请求方法，一个请求地址。

针对上面的那个接口，地址已经给出 ，而它的请求方法是get 。那么在postman中只需要把这俩个参数填写上即可请求 。

**具体实现步骤：**

1. 打开postman，新建一个请求。

2. 在请求方法中选择请求方法：GET,因为在postman中默认的请求方法就是GET,所以这一步可以忽略

3. 接口URL中输入地址，点击Send按钮就可以发送请求了 。



说明：查询参数的URL一般直接拷贝到输入的URL地址栏中就可以了，当然也可以把查询参数在Params中输入，俩者的效果是一样的

**1.2 表单类型的接口请求**

**什么是表单 ？**

我们都知道，在发送HTTP请求的时候，一个请求中一般包含三个部分，分别是请求行，请求头，请求体 。

不同的接口，请求体的数据类型是不一样的，比较常见的一种就是表单类型，那么什么是表单类型呢 ？简单理解就是在请求头中查看Content-Type，它的值如果是:application/x-www-form-urlencoded .那么就说明客户端提交的数据是以表单形式提交的 。见下图：



**postman中如何请求？**

如果在postman请求上图的接口，我们只需要填写四个参数，分别是(可以参考上图)：

* 请求方法：POST
* 请求URL：http://localhost/index.php?m=Home&c=User&a=do\_login&t=0.21942974229794432
* 请求头：Content-Type：application/x-www-form-urlencode
* 请求体：username=13088888888&password=123456&verify\_code=8888

实现步骤：

1. 打开postman，新建一个请求 。
2. 在请求中设置以上四个参数，点击Send按钮。在postman中设置请求体类型为，需要选择body-> x-www-form-urlencoded
3. 查看响应数据。



**1.3 上传文件的表单请求**

在做接口测试时，我们经常会遇到需要上传文件的接口，比如微信的更新头像。这就需要用到：multipart/form-data。它也属于一种表单，但它既支持表单请求，也支持文件上传。它的请求报文中数据往往是下面这样的。

POST http://localhost/index.php/home/Uploadify/imageUp/savepath/head\_pic/pictitle/banner/dir/images.html HTTP/1.1

Content-Type: multipart/form-data

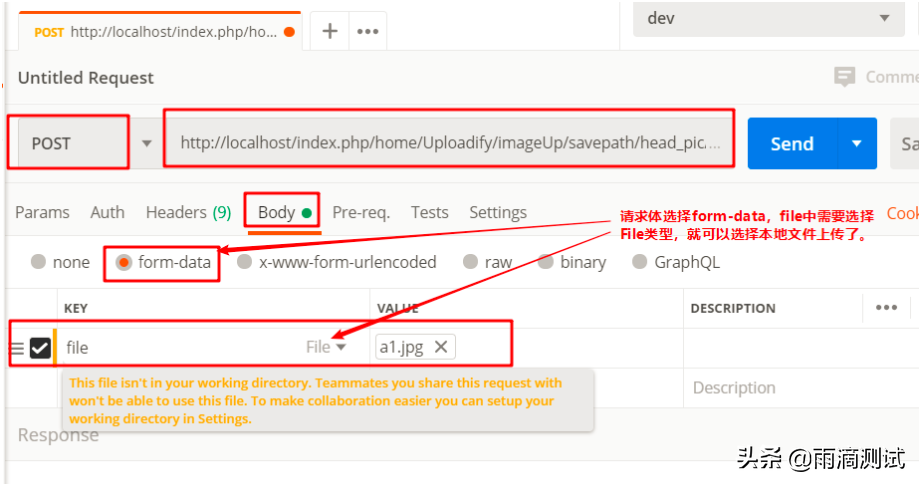
file=a1.jpg

这种类型的接口，在postman中该如何请求呢 ？我们先分析需要填写的参数 。

* 请求方法：POST
* 请求URL：http://localhost/index.php/home/Uploadify/imageUp/savepath/head\_pic/pictitle/banner/dir/images.html
* 请求类型：multipart/form-data
* 请求体：file=a1.jpg

实现步骤：

1. 打开postman，新建一个请求 。
2. 在请求中设置以上四个参数，点击Send按钮。注意：在postman中设置请求体类型，需要选择body-> form-data 。file中要选择File类型，然后上传本地的文件 。
3. 查看响应数据。



**1.4 json类型的接口请求**

这应该是接口测试中最常见的一种情况了 ， 也就是请求体类型为json,我们来看下这个请求报文 。

POST http://xxx/api/sys/login HTTP/1.1

Content-Type: application/json;charset=UTF-8

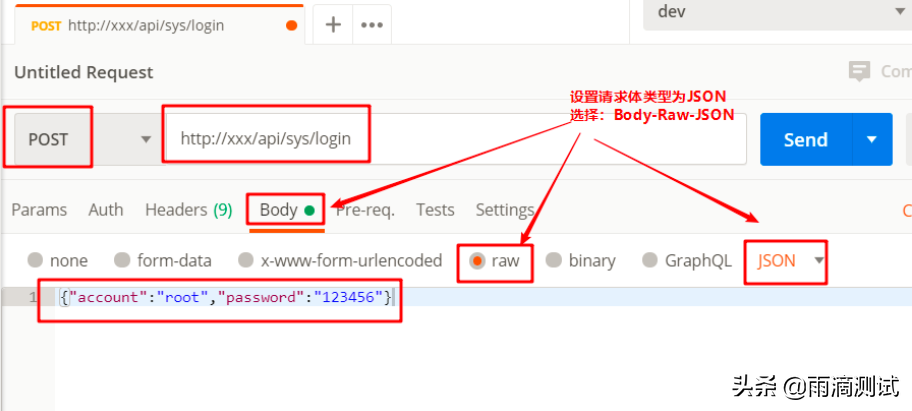
{"account":"root","password":"123456"}

根据以上报文，我们可以分析出，我们在postman只需要填写四个参数即可，具体如下：

* 请求方法：POST
* 请求地址：http://xxx/api/sys/login
* 请求体类型：json
* 请求体数据：{"account":"root","password":"123456"}

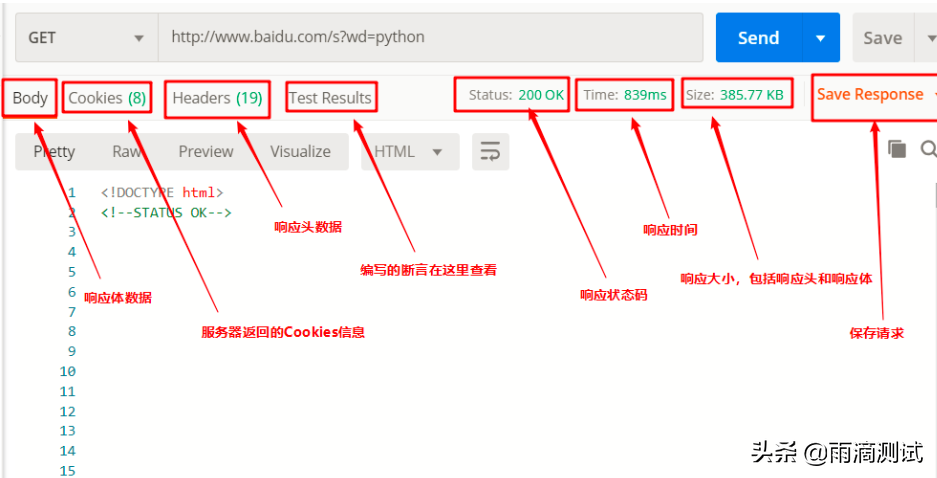
实现步骤：

1. 打开postman，新建一个请求 。
2. 在请求中设置以上四个参数，点击Send按钮。注意：在postman中设置请求体类型，需要选择body-> raw -JSON
3. 查看响应数据。



**2.接口响应数据解析**

响应数据是发送请求后经过服务器处理后返回的结果，响应由三部分组成，分别是状态行、响应头、响应体。我们来看下postman的响应数据展示。



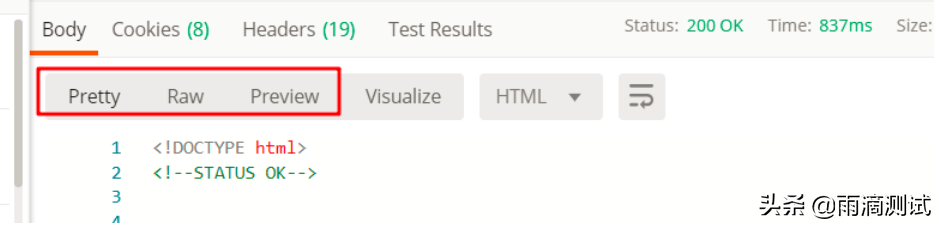
在postman中的响应数据展示：

* 状态行：Status：200 OK
* 响应头：Headers + Cookies，需要注意的是Cookies是包含在响应头中的，但是为了明显，工具会分开显示
* 响应体：Body

那么这些数据对我们做接口测试有什么作用呢 ？

* Body和Status是我们做接口测试的重点，一般来说我们都会验证响应体中的数据和响应状态码
* Test Results 是我们编写断言后，可以查看断言的执行结果 ，所以这个对我们也很有用 。
* Time 和Size 是我们做性能测试时，可以根据这两个参数来对所测接口的性能做一个简单的判断。

接下来我们再来关注下Body中的几个显示主题，分别是：Pretty，Raw，Preview .



Pretty:翻译成中文就是漂亮 ， 也就是说返回的Body数据在这个标签中查看 ，都是经过格式化的，格式化后的数据看起来更加直观，所以postman默认展示的也是这个选项。比如返回html页面，它会经过格式化成HTML格式后展示，比如返回json，那么也会格式化成json格式展示 。

Raw：翻译成中文未经过加工的，也就是原始数据 ，原始数据一般都是本文格式的，未经过格式化处理的，一般在抓包工具中都有这个选项 。

Preview：翻译成中文就是预览，这个选项一般对返回HTML的页面效果特别明显，如请求百度后返回结果，点击这个选项后就直接能查看到的页面 ，如下图 。同时这个选项和浏览器抓包中的Preview也是一样的 。



**3. 接口管理（Collection）**

当我们对一个或多个系统中的很多用例进行维护时，首先想到的就是对用例进行分类管理，同时还希望对这批用例做回归测试 。在postman也提供了这样一个功能，就是Collection 。通过这个Collection就可以满足我们的上面说的需求。

先对Collection功能的使用场景做个简单总结 。

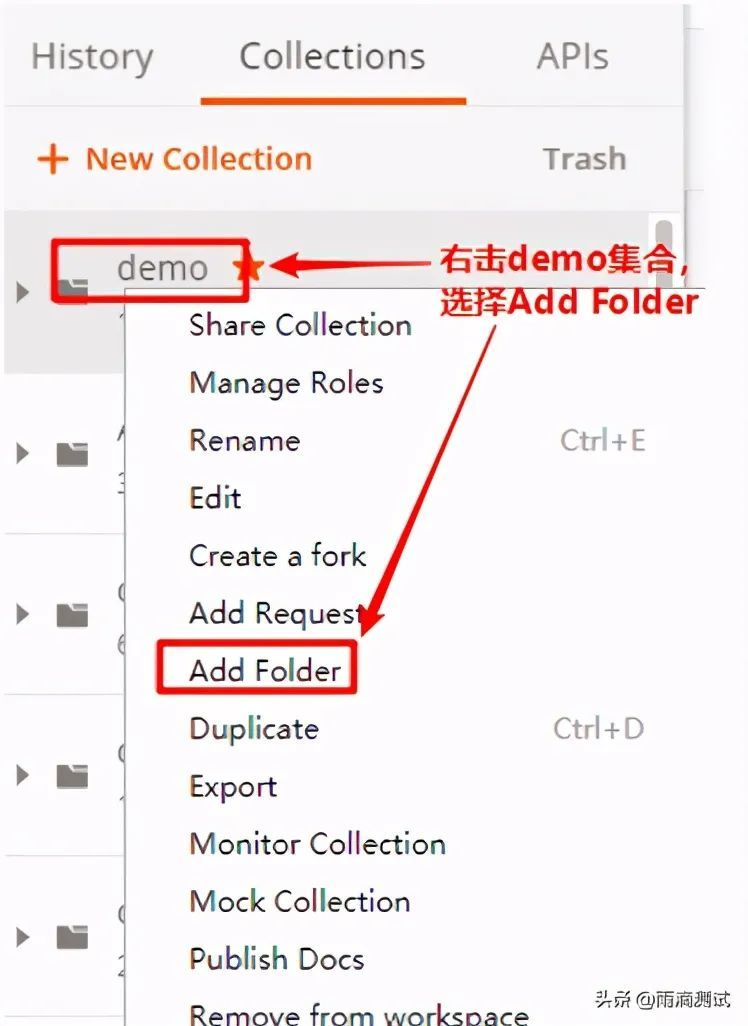
* 用例分类管理，方便后期维护
* 可以进行批量用例回归测试 。

那么Collection是如何去管理用例的呢 ？先想象我们要测试一个系统，系统下有多个模块，每个模块下有很多的被测接口用例 。那么基于这个场景，我们来通过Collection来进行实现：

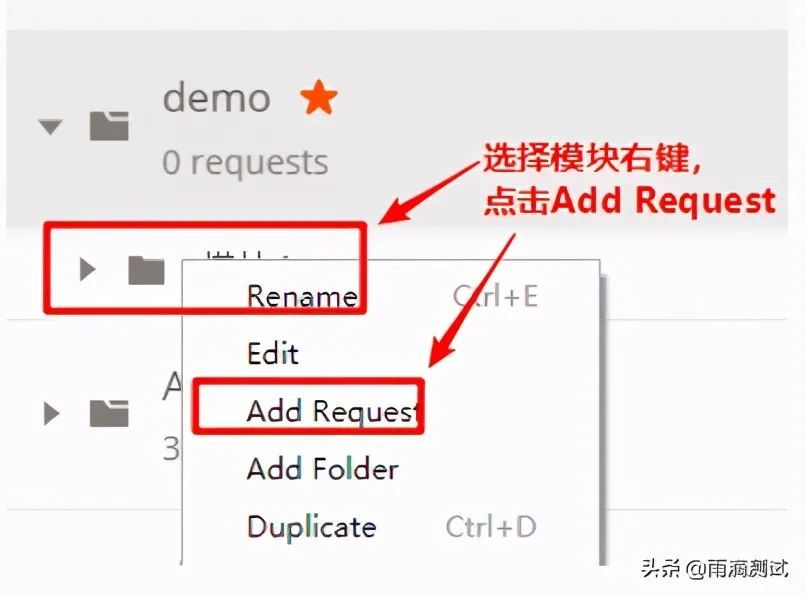
1. 点击Collection，点击+New Collection，在弹出的输入框中输入Collection名称（这个就可以理解为所测试的系统）



2. 选中新建的Collection右键，点击Add Folder ，在弹出对话框中输入文件夹名称（这个就可以理解为系统中的模块）



3. 选中新建的Folder，点击Add Request ，在弹出的对话框中输入请求名称，这个就是我们所测试的接口，也可以理解为测试用例 。



那么通过以上三个步骤，达到的效果就是如图所示：



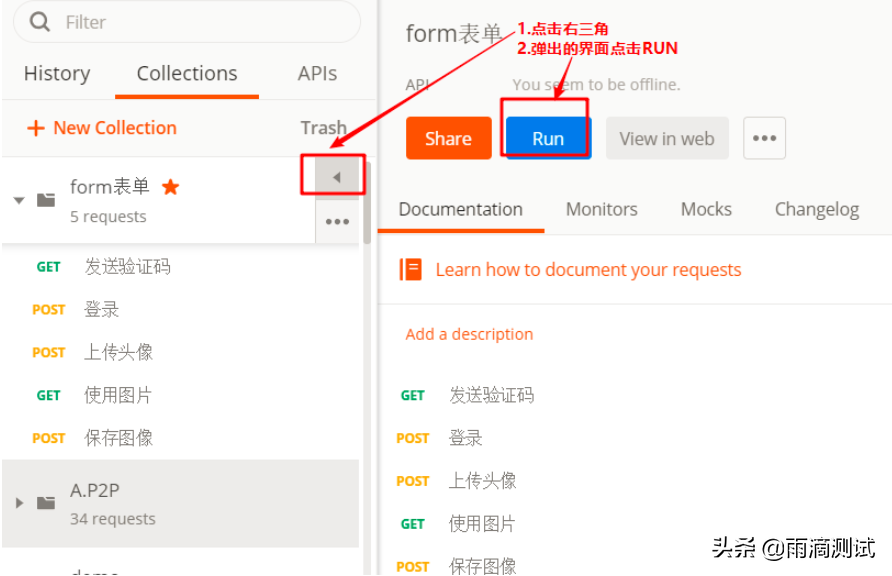
总结，通过上面的操作，我们实现了一个最简单的demo模型。但实际上，有了这个功能才是postman学习的开始，因为很多功能都是基础这个功能的基础上进行的，比如用例的批量执行，Mock ，接口文档等功能 。

**4. 批量执行接口请求**

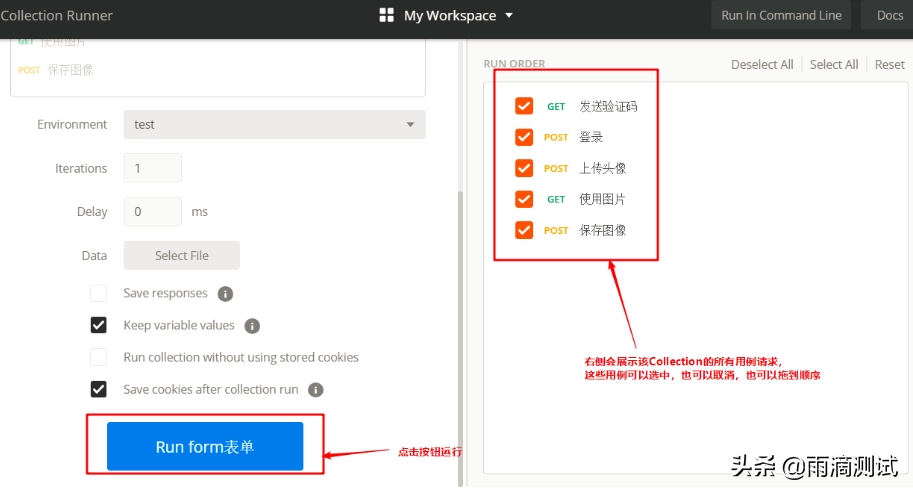
当我们在一个Collection中编写了很多的接口测试用例，想一起执行这批用例，在postman中是如何操作呢 ？

实现步骤：

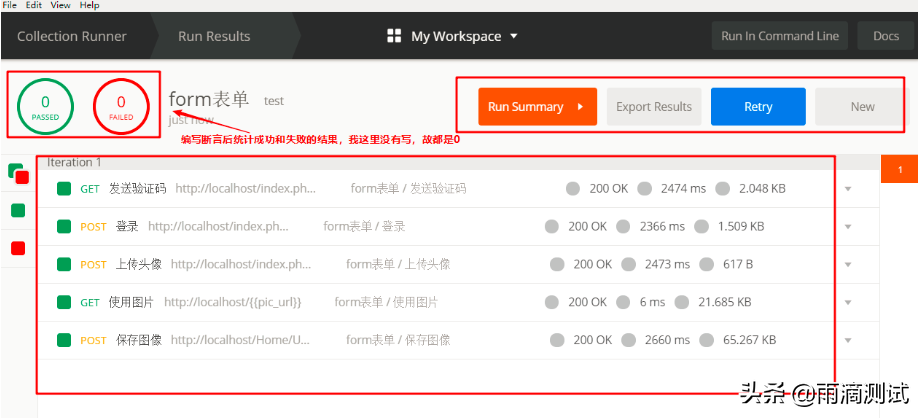
1. 选中一个Collection，点击右三角，在弹出的界面点击RUN



2. 这是会弹出一个叫Collection Runner的界面，默认会把Collection中的所有用例选中 。



3. 点击界面下方的RUN Collection，就会对Collection中选中的所有测试用例运行 。



对上面的几个红框内的功能进行简单说明：

* 断言统计：左上角的两个0是统计当前Collection中断言成功的执行数和失败的执行数，如果没有编写断言默认都为0 。
* Run Summary: 运行结果总览，点击它可以看到每个请求中具体的测试断言详细信息 。Export Result：导出运行结果，默认导出的结果json文件 。
* Retry: 重新运行，点击它会把该Collection重新运行一遍
* New：返回到Runner，可以重新选择用例的组合 。

总体来说，这个功能主要是用于对一个Collection中的所有用例或部分用例进行批量运行，已达到手工回归测试的目的。

**5.日志调试**

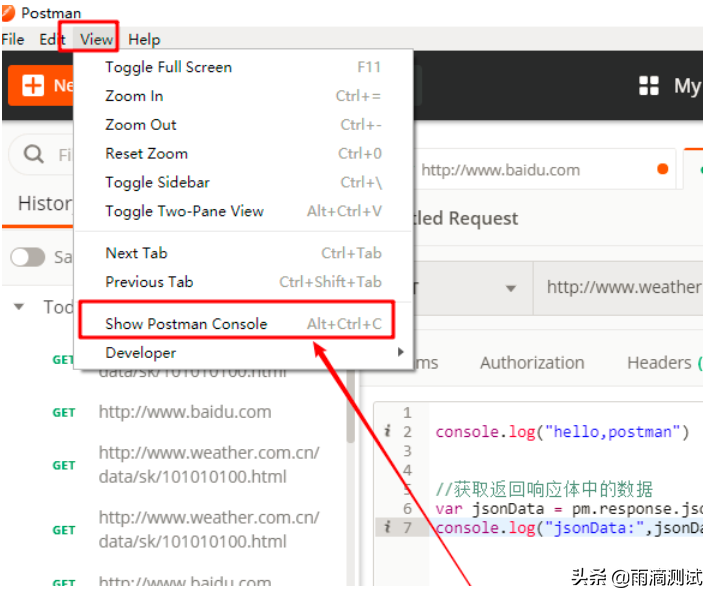
在做接口测试时，经常会因为代码写的有问题导致报错，这时通过查看日志就显得非常重要了，postman也提供了这样的功能，它允许我们在脚本中编写打印语句，查看打印的结果 ; 同时也可以查看每个请求的日志信息 。

在postman中编写日志打印语句使用的是JavaScript，编写的位置可以是Pre-request Script 或Tests标签中。编写打印语句如：console.log("我是一条日志")

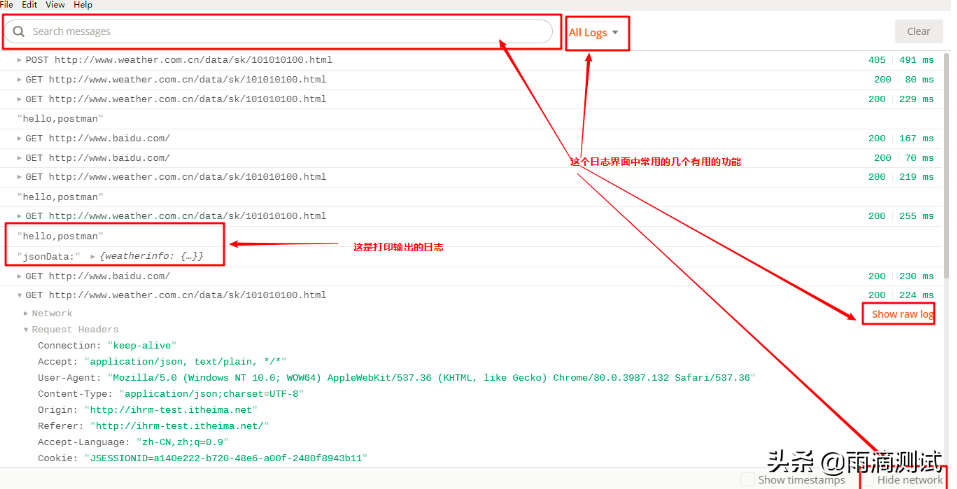


那么打印的日如何看呢 ？在postman中有俩个入口，第一个入口就是：view-show postman console 。

第二个入口就是左下角第三个图标 。



打开的日志界面：



这里面有几个比较实用的功能：

* 搜索日志：输入URL或者打印的日志就能直接搜索出我们想要的请求和日志，这对我们在众多日志中查找某一条日志是非常方便的 。
* 按级别搜索：可以查询log,info,warning,error级别的日志 ，有助于我们更快定位到错误 。
* 查看原始报文(Show raw log)：如果习惯看原始请求报文的话，这个功能可能更方便些 。
* 隐藏请求(Hide network)：把请求都隐藏掉，只查看输出日志 。

总之，通过这个功能，我们在请求接口报错时，通过打印响应的日志，就能很轻松的找到问题原因了 。

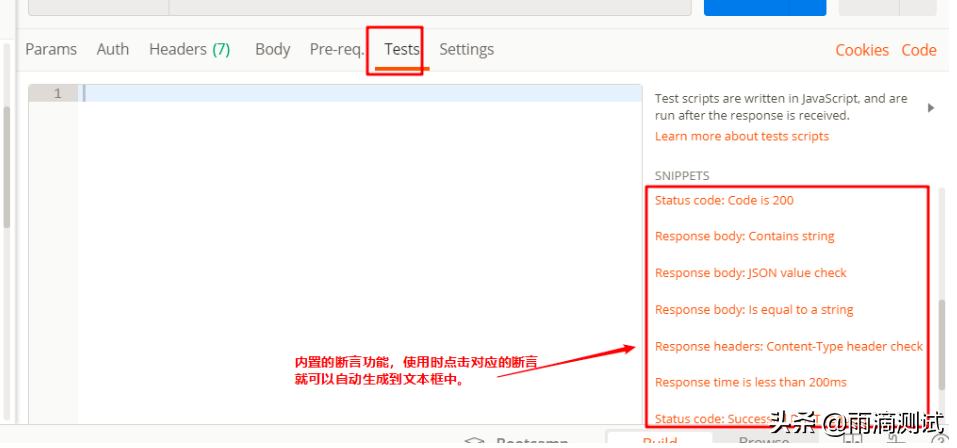
**6.断言**

如果没有断言，我们只能做接口的功能测试，但有了断言后，就为我们做自动化提供了条件，并且在postman中的断言是非常方便和强大的 。

我们先来了解下postman断言的一些特点 ，具体如下

* 断言编写位置：Tests标签
* 断言所用语言：JavaScript
* 断言执行顺序：在响应体数据返回后执行 。
* 断言执行结果查看：Test Results

在上面我们介绍到，编写的断言代码是JavaScript，那如果不会写怎么办 ？不用担心，因为postman已经给我们内置了一些常用的断言 。用的时候，只需从右侧点击其中一个断言，就会在文本框中自动生成对应断言代码块 。



接下来就让我们了解一些常用断言，还是按响应的组成来划分，分别是状态行，响应头，响应体。

状态行中又包括状态码，状态消息 。在postman也可以对这两个进行断言

状态行中的断言：

* 断言状态码：Status code: code is 200

pm.test("Status code is 200", function () {

pm.response.to.have.status(200); //这里填写的200是预期结果，实际结果是请求返回结果

});

* 断言状态消息：Status code：code name has string

pm.test("Status code name has string", function () {

pm.response.to.have.status("OK"); //断言响应状态消息包含OK

});

响应头中的断言

* 断言响应头中包含：Response headers:Content-Type header check

pm.test("Content-Type is present", function () {

pm.response.to.have.header("Content-Type"); //断言响应头存在"Content-Type"

});

断言响应体(重点)

断言响应体中包含XXX字符串：Response body:Contains string

pm.test("Body matches string", function () {

pm.expect(pm.response.text()).to.include("string\_you\_want\_to\_search");

});

//注解

pm.expect(pm.response.text()).to.include("string") 获取响应文本中包含string

断言响应体等于XXX字符串：Response body : is equal to a string

pm.test("Body is correct", function () {

pm.response.to.have.body("response\_body\_string");

});

//注解

pm.response.to.have.body("response\_body\_string"); 获取响应体等于response\_body\_string

断言响应体(json)中某个键名对应的值：Response body : JSON value check

pm.test("Your test name", function () {

var jsonData = pm.response.json();

pm.expect(jsonData.value).to.eql(100);

});

//注解

var jsonData = pm.response.json() 获取响应体，以json显示，赋值给jsonData .注意：该响应体必须返会是的json，否则会报错

pm.expect(jsonData.value).to.eql(100) 获取jsonData中键名为value的值，然后和100进行比较

* 响应时间(一般用于性能测试)
* 断言响应时间：Response time is less than 200ms

pm.test("Response time is less than 200ms", function () {

pm.expect(pm.response.responseTime).to.be.below(200); //断言响应时间<200ms

});

案例说明：

针对以下接口返回的数据进行断言：

{

"cityid": "101120101",

"city": "济南",

"update\_time": "2020-04-17 10:50",

"wea": "晴",

"wea\_img": "qing",

"tem": "16",

"tem\_day": "20",

"tem\_night": "9",

"win": "东北风",

"win\_speed": "3级",

"win\_meter": "小于12km/h",

"air": "113"

}

* 断言响应状态码为200
* 断言city等于济南
* 断言update\_time包含2020-04-17



总结，整体来说，如果用postman做接口测试，这个断言功能必不可少，其中我们常断言的响应体包含和JSON这俩个断言又是重重之重。

**7. 变量（全局/集合/环境）**

变量可以使我们在请求或脚本中存储和重复使用其值，通过将值保存在变量中，可以在集合，环境或请求中引用。

对我们做接口测试来说，又是一个非常重要的功能 。

在postman常用的三种变量分别是全局变量，环境变量，集合变量 。

* 全局变量：一旦申明了全局变量，全局有效，也就是说postman中的任何集合，任何请求中都可以使用这个变量。它的作用域是最大的 。
* 环境变量：要申明环境变量，首先的创建环境，然后在环境中才能创建变量 。如果要想使用环境变量，必须先选择(导入)这个环境，这样就可以使用这个环境下的变量了 。需要说明的是环境也可以创建多个 。每个环境下又可以有多个变量 。
* 集合变量：集合变量是针对集合的，也就是说申明的变量必须基于某个集合，它的使用范围也只是针对这个集合有效 。

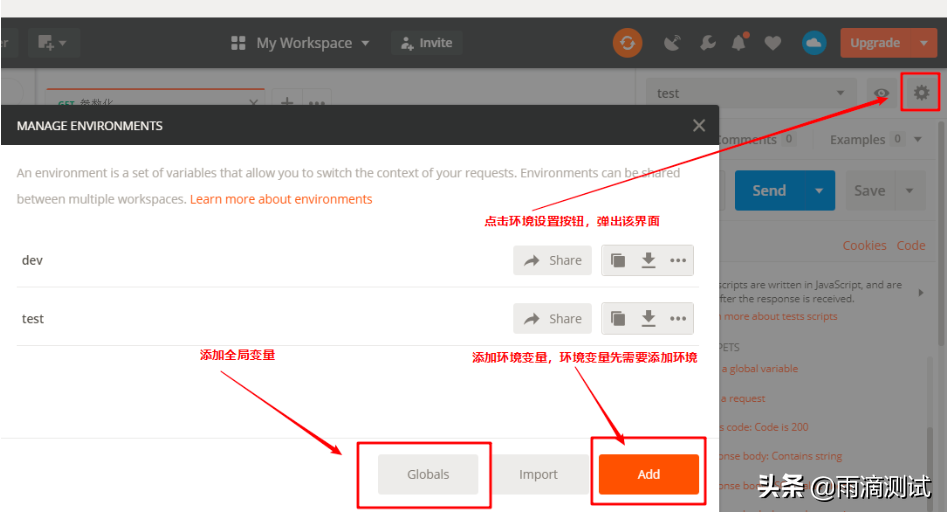
其中，他们的作用域范围依次从大到小：全局变量>集合变量>环境变量 。当在几个不同的范围内都申明了相同的变量时，则会优先使用范围最小的变量使。

想要使用变量中的值只需俩个步骤，分别是定义变量和获取变量 。

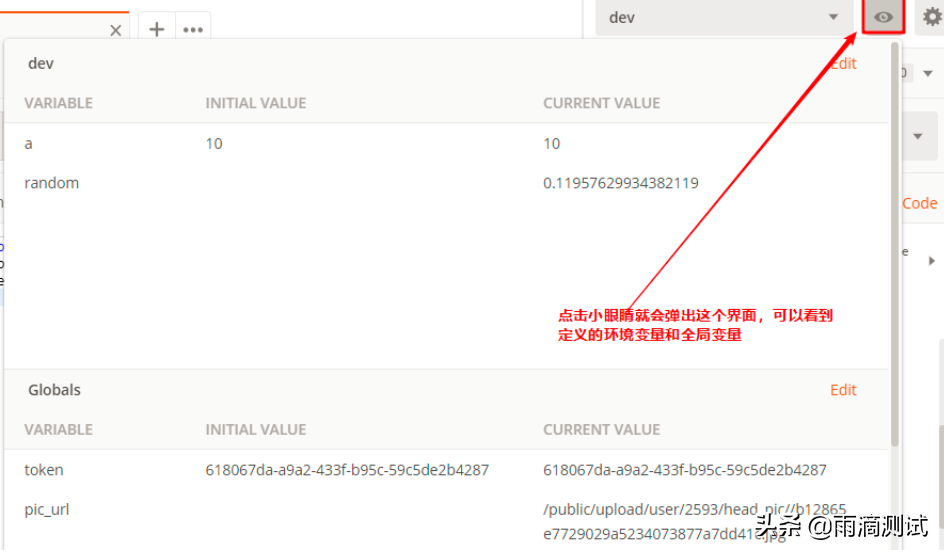
1. 定义变量（设置变量）
2. 获取变量（访问变量）

**定义变量**

定义全局变量和环境变量，点击右上角的小齿轮，弹出如下界面，就可以根据需求定义全局变量或者环境变量了。

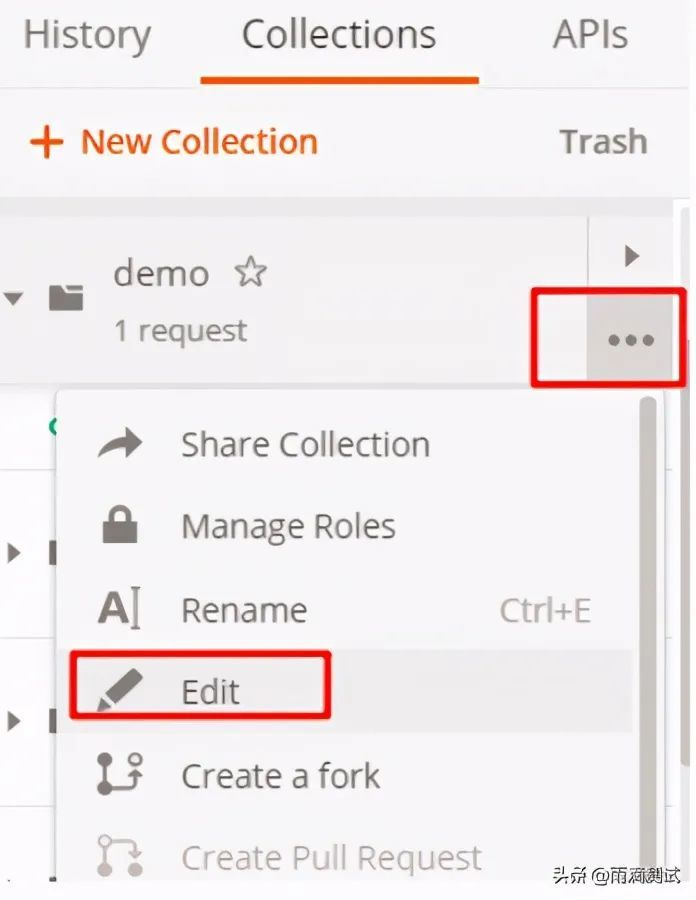


已经定义的全局变量和环境变量，可以进行快速查看：



**定义集合变量**

选择一个集合，打开查看更多动作(...)菜单，然后点击编辑 。选择“变量”选项卡以编辑或添加到集合变量。

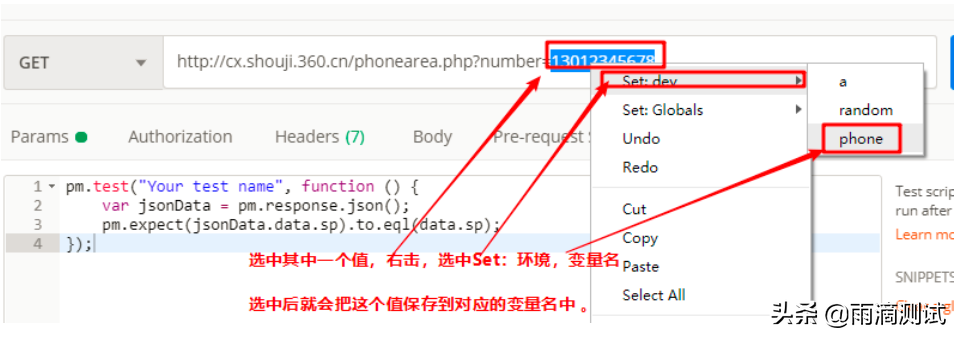


定义变量除了以上方式，还有另外一种方式 。但是这种方式在不同的位置定义，编写不一样。

在URL，Params , Authorization , Headers , Body中定义：

1. 手工方式创建一个空的变量名

2. 在以上的位置把想要的值选中右击，选中Set：环境|全局 ，选中一个变量名，点击后就会保存到这个变量中



在Tests，Pre-requests Script：

* 定义全局变量：pm.collectionVariables.set("变量名",变量值)
* 定义环境变量：pm.environment.set("变量名"，变量值)
* 定义集合变量：pm.variables.set("变量名",变量值)

**获取变量**

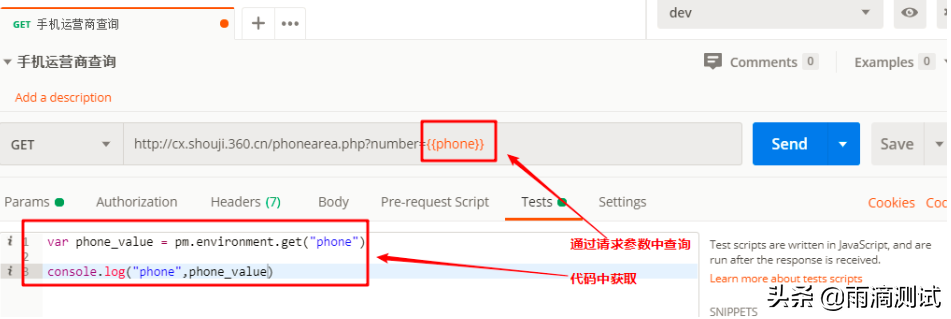
定义好变量，接下来就可以使用变量了 。需要注意的是，在不同的位置获取变量，编写的规则也是不一样的 。

如果在请求参数中获取变量，无论是获取全局变量，还是环境变量，还是集合变量，获取的方式都是一样的编写规则：{{变量名}} 。

* 请求参数指的是：URL，Params , Authorization , Headers , Body

如果是在编写代码的位置(Tests,Pre-requests Script)获取变量，获取不同类型的变量，编写的代码都不相同，具体如下：

* 获取环境变量：pm.environment.get(‘变量名’)
* 获取全局变量：pm.globals.get('变量名')
* 获取集合变量：pm.pm.collectionVariables.get.get('变量名')



变量的使用场景非常广泛，比如我们后面要提到的接口关联，请求前置脚本都会使用到变量 。

**8. 请求前置脚本**

前置脚本其实就是在Pre-requests Script中编写的JavaScript脚本，想要了解这个功能，需要先了解它的执行顺序。那么下面就来看下它的执行顺序 。

可以看出，一个请求在发送之前，会先去执行Pre Request Script（前置脚本）中的代码 。那么这个功能在实际工作中有什么作用呢 ？

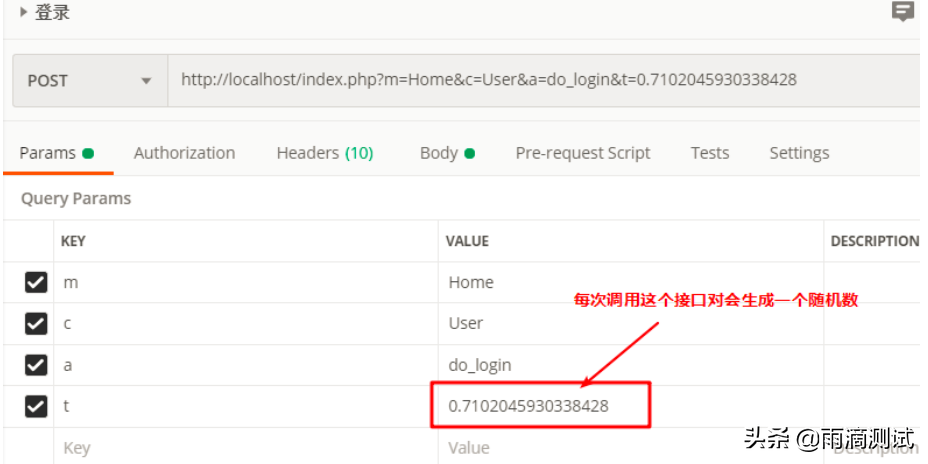
主要场景：一般情况下，在发送请求前需要对接口的数据做进一步处理，就都可以使用这个功能，比如说，登录接口的密码，在发送前需要做加密处理，那么就可以在前置脚本中做加密处理，再比如说，有的接口的输入参数有一些随机数，每请求一次接口参数值都会发送变化，就可以在前置脚本中编写生成随机数的代码 。

总体来说，就是在请求接口之前对我们的请求数据进行进一步加工处理的都可以使用前置脚本这个功能。

接下来通过一个案例来看下该功能是如何使用 ？

案例：

* 请求的登录接口URL，参数t的值要求的规则是每次请求都必须是一个随机数。
* 接口地址：http://localhost/index.php？m=Home&c=User&a=do\_login&t=0.7102045930338428



实现步骤：

1. 在前置脚本中编写生成随机数
2. 将这个值保存成环境变量
3. 将参数t的值替换成环境变量的值 。



总之，这个前置脚本对我们做接口测试也非常有用，对一些复杂的场景，都可以使用前置脚本进行处理后再请求接口 。

**9. 接口关联**

在我们测试的接口中，经常出现这种情况 。上一个接口的返回数据是下一个接口的输入参数 ，那么这俩个接口就产生了关联。这种关联在做接口测试时非常常见，那么在postman中，如何实现这种关联关系呢 ？

实现思路：

1. 提取上一个接口的返回数据值，
2. 将这个数据值保存到环境变量或全局变量中
3. 在下一个接口获取环境变量或全局变量

案例：

* 用户上传头像功能，需要用户先上传一张图片，然后会自动预览 。那么在这个过程中，会调用到俩个接口 ，第一个上传头像接口，第二个预览图像接口 。
* 其中调用上传头像接口成功后会返回如下信息：

{

"url": "/public/upload/user//head\_pic//ba51d1c2f7f7b98dfb5cad90846e2d79.jpg",

"title": "banner",

"original": "",

"state": "SUCCESS",

"path": "images"

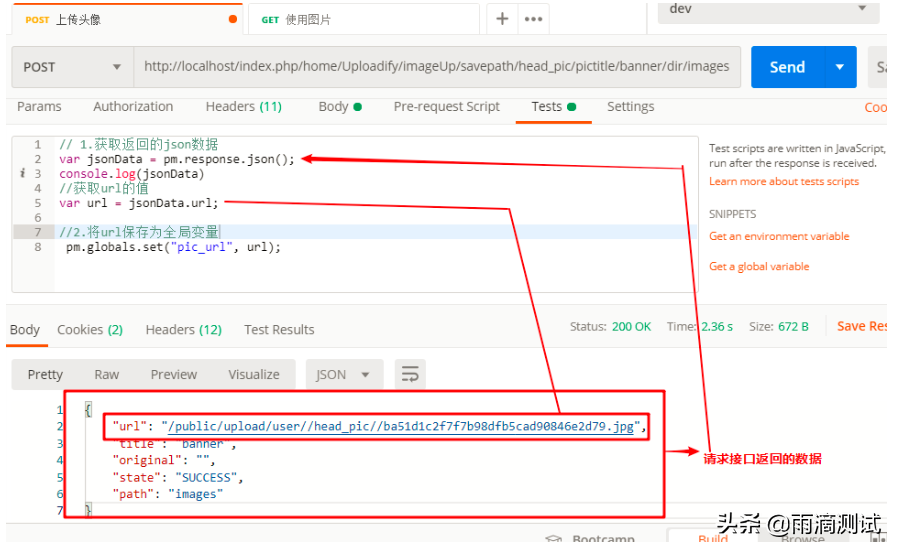
}

而图像预览接口URL为：

http://localhost/public/upload/user//head\_pic//ba51d1c2f7f7b98dfb5cad90846e2d79.jpg 。可以看出这个接口的URL后半部分其实是上一个接口返回的url的值 。那么这俩个接口就产生了关联。那么在postman 可以通过以下三步完成这俩个接口的关联实现 。

实现步骤：

1. 获取上传头像接口返回url的值
2. 将这个值保存成全局变量(环境变量也可以)
3. 在图像预览中使用全局变量



可以看出，接口的关联的解决方案都是用的是变量中的知识，也就是说只要你明确了要提取的值，后面就是保存该值，然后在其他接口使用该值就可以了。

**10.常见返回值获取**

在做接口测试时，请求接口返回的数据都是很复杂的json数据，有着多层嵌套，这样的数据层级在postman怎么获取呢 ？

案例1：多层json嵌套, 获取user\_id的值

{

"code": 0,

"message": "请求成功！",

"data": {

"user\_id": "1252163151781167104"

}

}

//获取json体数据

var jsonData = pm.response.json()

// 获取user\_id的值,通过.获取

var user\_id = jsonData.data.user\_id

案例2：json中存在列表，获取points中的第二个元素

{

"code": 0,

"message": "请求成功！",

"data": {

"roles": {

"api": [

"API-USER-DELETE"

],

"points": [

"point-user-delete",

"POINT-USER-UPDATE",

"POINT-USER-ADD"

]

},

"authCache": null

}

}

//获取json体数据

var jsonData = pm.response.json()

// 获取user\_id的值,通过下标获取列表中某个元素

var user\_id = jsonData.data.roles.points[1]

案例3：列表中取最后一个元素

{

"code": 0,

"message": "请求成功！",

"data": {

"total": 24,

"rows": [

{

"id": "1066370498633486336",

"mobile": "15812340003",

"username": "zbz"

},

{

"id": "1071632760222810112",

"mobile": "16612094236",

"username": "llx"

},

...

{

"id": "1075383133106425856",

"mobile": "13523679872",

"username": "test001",

},