**使用Postman自动化接口测试的步骤**

当前环境：

Window 7 - 64

Postman 版本（免费版）： Chrome App v5.5.3

在接口测试之前，要考虑一下几个问题：

如何判断接口是否请求成功

如何进行接口批量、定期测试

如何处理依赖接口问题（比如商品下单的接口必须要求先登录）

所以，接下来就主要分为 3 个部分进行介绍，以分别解决这 3 个问题。

接口结果判断

首先，既然是自动化测试，那么我们肯定需要工具 (Postman) 或者代码能帮我们直接判断结果是否符合预期。那么在接口测试上，大体就两个思路：

判断请求返回的 code 是否符合预期

判断请求返回的内容中是否包含预期的内容（关键字）

接下来我们看看如何利用 Postman 来解决上述的问题：

功能区

在 Postman 中相关的功能在非常显眼的地方，Tests 功能的使用需要我们有一定的编程语言基础，目前支持的脚本语言即为 JavaScript 。 但比较好的一点是，我们不需要再去考虑上下文问题以及运行环境的问题 ，也就是说我们只需要在这边完成结果逻辑判断的代码块即可。而 Postman 还为我们提供了一些常用的代码模板，在 Tests 面板右边的 SNIPPETS 功能区中，所以对 JavaScript 不大了解问题也不大。代码编写相关将在下文进行具体介绍。

脚本相关

先看上图的代码部分，我们可以发现 responseCode 、 responseBody 和 tests 三个变量（可直接使用） ：

responseCode ：包含请求的返回的状态信息(如：code)

responseBody： 为接口请求放回的数据内容（类型为字符串）

tests ： 为键值对形式，用于表示我们的测试结果是成功与否，最终展示在 Test Results 中。

key ：（如：code 200）我们可以用来当做结果的一个描述

value：其值为布尔型，ture 表示测试通过， false 表示测试失败。

所以上述代码应该不难理解了，而有了返回结果的数据以及表示结果成功与否的方式，那么我们“接口结果判断”的问题也就基本解决了。

另外还有几个比较常用的：

responseTime ：请求所耗时长

postman ：可以做的比较多，比如

获取返回数据的头部信息：postman.getResponseHeader("")

设置全局变量：postman.setGlobalVariable("variable\_key", "variable\_value");

更多功能可以查看官方文档(需梯子)

代码模板

Postman 在 SNIPPETS 功能区中为我们提供的代码模板已经能解决大部分情况了，以下先挑几个跟结果判断相关的进行讲解：

Status code : Code is 200

//根据返回的 Code 判断请求情况

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

​

Response body: Contains string

//判断返回的内容中是否存在“关键字”。（tests 的 key 可修改，将不再强调）

tests["Body matches string"] = responseBody.has("这里可以改为你要判断的关键字内容");

//如上文提到的：

// 判断结果中是否存在 access\_token 关键字

tests["has access\_token"] = responseBody.has("access\_token");

​

Response body: is equal to string

//判断返回内容是否跟预期完全相等。

tests["Body is correct"] = responseBody === "这里可以改为你的预期内容";

Response body: JSON value check

//上文提到，responseBody 为字符串类型，支持转为 Json 格式

var jsonData = JSON.parse(responseBody);

tests["Your test name"] = jsonData.value === 100;​

Response time is less than 200ms

//判断请求时长是否小于200ms ,具体时长按情况自定义

tests["Response time is less than 200ms"] = responseTime < 200;

以上介绍的这些基本已经足够完成对单一接口的测试了，但我们知道如果没有批量、定时任务, 那么这些都将毫无意义，继续…

集合(批量)测试

想要进行接口的批量测试、管理，那么我们需要将待测试的接口全部都保存到同一个集合（Collections）中，你可以认为就是保存到同一个文件夹中。先看看 Postman 中的操作步骤：

通过以上步骤，我们得到一个待测的接口集合，为了简化情况，我这边每个接口成功与否的条件都是用 code 是否为 200 来判断：

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

批量执行

以上准备就绪后，我们就可以开始批量运行接口进行测试了：

点击Run 后，会新打开一个页面：

Environment ：用于切换接口运行的环境，这里先不管，后面再讲

Iteration ：用于设置接口一共要运行的次数。

Delay : 设置每次运行接口之间的时间间隔，单位为毫秒。

Data File : 上传测试数据文件 （下文单独讲）

变化的参数数据

我们已经了解了，如何让多个接口循环运行多次，但是现在有个问题，按目前这个步骤，每次运行时接口的参数都是一样的，那么就算我们运行个100次、1000次意义也不大。

先看看我们写好的一个登录功能的接口：

使用变量

现在登录的账号和密码参数都是写死的，也就是不过我们执行多少次，都是拿这个账号去测试。 那么如果想要测试账号密码参数使用其它值有没有异常怎么办呢？（ 想要每次都手动改的可以跳过这部分 /手动滑稽）这里我们先简单讲一下在 Postman 中使用如何“变量”，如下图：

引用一个变量的语法：{{变量名}}， 图中可以看到，我们将账户和密码字段的参数值都设置为变量：{{username}} 、{{password}} 。修改完直接点击运行 （Send） 当然是不行的，因为目前这两个变量还未被赋值，不过我们可以在 Pre-request Script 面板中进行赋值操作：

Pre-request Script

Pre-request Script 与 Tests 类似，区别在于：Pre-request Script 中的脚本是在执行请求之前运行，而Tests 中的脚本则是在请求完成之后执行。所以，我们可以在 Pre-request Script 功能区中用脚本先个上面两个变量进行赋值，如:

//设置全局变量

postman.setGlobalVariable("username", "test1");

postman.setGlobalVariable("password", "123456");

但是用 Pre-request Script 进行赋值操作仍然不能解决我们的问题，因为按照这种写法，不论运行多少次其实都还是用固定（写死）的数据进行测试。当然既然是脚本语言，也会有更灵活的用法，这边先不将。

测试数据集

接下来我们讲讲 Data File , 在运行集合前的这个选项就是用来上传测试数据（文件）以赋值给相应变量的。我们先以 CSV 格式的测试数据为例：

username,password

test1,123456

test2,222222

test3,123456

test4,444444

数据格式类似表格，第一行表示对应的变量名，下面 4 行表示 4 组账号密码数据（其中两组为正确数据） ，我们保存一份内容为上述示例数据后缀名为.csv 的文件后，再次开始测试看看效果，我们选择运行次数为 4 （对应 4 组测试数据）、选择对应的 CSV 文件运行后，可以看到我们的结果确实如我们的预期。接口 Request 运行的结果为两次成功两次失败，也就是每一次运行都赋值了不同的账号密码的测试数据 （在最新的桌面客户端版本中可以看到每次具体的请求情况，这边就不再细说了）。

如果使用 Json 文件的话，那么格式如下：

[

{

"username": "test1",

"password": "123456"

},

{

"username": "test2",

"password": "222222"

},

{

"username": "test3",

"password": "123456"

},

{

"username": "test4",

"password": "444444"

}

]

定期任务

Postman 提供了一个 Monitors (监视器)功能，支持我们提交一个测试任务，按照设置的定时器进行运行，如每小时测试一次，具体操作如下：

请求依赖问题

讲完接口结果判断和集合批量测试后，我们再来看看比较复杂的情况，即依赖请求问题，比如我们的购物下订单接口要求必须先登录后才可访问。但大部分依赖问题其实本质上就是一个接口间数据传递的问题，比如调用登录接口后返回一个标识，假设为 token ，那么我们请求下订单接口时只要一起携带 token 参数进行请求即可。所以，问题变为：

保证接口调用顺序

将接口A返回的数据传递给后续的接口B、C、D

接口执行顺序

首先，说明一下，接下来说的接口都是默认属于同一个集合 (Collections) 中的。

还是以我们上文中创建好接口集合为例，如果你有注意我们执行批量测试的结果，就会发现接口的执行顺序其实就是按照这边目录中的顺序（从上到下），即： Request1 -> Request2 -> Request3。

这边接口名字可能有点误导性，所以再强调一下： 按目录中从上到下的顺序执行 （与字典排序无关）

所以有了这个默认的执行顺序后，那么我们便可以把需要优先执行的接口放前面即可，比如把“登录接口”放在第一个。

自定义执行顺序

当然，如果只有默认的一个执行顺序的话，通常没法满足我们复杂的业务需求，所以 Postman 为我们提供了一个函数：postman.setNextRequest("填写你要跳转的接口名") ，支持我们跳转到指定接口继续执行，举个例子：

我们在运行完 Request1 接口成功后，不需要再运行 Request2 而是直接跳至 Request3 ，那么我可以在 Request1 接口的 Tests 功能区中执行跳转代码，如：

这里需要注意几点：

postman.setNextRequest() 只在运行集合测试的时候生效，也就是说我们单独运行 (Send) 接口Request1 时，函数是不起作用的。

当我们运行集合测试成功从 Request1 -> Request3 后，如果 Request3 后面还有接口，那么后面的接口仍然继续按默认顺序执行，即图中的接口 Request4 仍会被执行。

指定的跳转接口必须属于同一个集合中。

setNextRequest() 函数不管在 Tests 脚本中何处被调用，它都只在当前脚本最后才被真正执行。比如我们将图中的第二行与第一行互调后，那么在运行跳转函数后第二行代码仍会被执行。

所以，利用 setNextRequest() 函数，我们便可以按照条件跳过不必要的接口，或者建立我们自己的一个逻辑测试。

数据传递

在讲数据传递前，先聊聊 Postman 中全局变量、环境切换的使用。

全局变量

全局变量的概念其实我们在上文中讲 Pre-request Script 时有简单提到，也就是说我们可以通过脚本代码来设置全局变量，我们可以看看运行上文的脚本后的效果：

我们可以看到运行后，username 和 password 两个变量已经被成功保存下来，那么我们在任意接口中便都可以通过变量引用的语法如：{{username}} 来使用它们。

另外，Postman 不仅支持代码设置全局变量的方式，它还支持可视化操作：

进入对应界面后，便可直接进行管理：

多环境区分与切换

通常情况下，我们的接口都会分为测试版本和线上版本（或者更多），而他们的区别可能仅是 ULR 不同，那么全局变量便不大合适解决这个问题。

参数的创建

可能你已经注意到，上图中我已经建有几个不同环境的参数“集合”了，再看一下：

我在每个环境中都创建了一个 host 参数，如：

当然，我们的环境参数也可以通过脚本的方式来进行设置，函数为：

//注意，该参数只添加到你当前选择的环境的“参数集”中

postman.setEnvironmentVariable("variable\_key", "variable\_value");

使用与切换

环境“参数集” 中的参数使用方式和全局变量一致，如图中 {{host}} ，不同环境的切换见下图：

解决依赖问题

掌握以上的预备知识后，我们开始看看如何用 Postman 解决存在依赖关系的接口测试。

假设场景

我们的接口 Request1 为登录接口，登录成功将会返回一个 access\_token 字段作为标识（已实现）。那么假设接口 Request3 为一个下订单的接口，需要携带登录返回的 access\_token 才能正常访问。

思路

保证 Request1 在 Request3 之前被运行

将 Request1 返回的 access\_token 的值添加到环境变量"参数集"中。

Request3 在请求时引用 access\_token 的值

将返回值存在 “全局变量” 或者 “环境变量” 中，视具体业务情况而定，该例中 access\_token 的值是与环境有关的，所以这里选择使用环境变量集存储。

Postman 中的操作

我们目录中已保证 Request1 接口优先执行

Request1 中 Tests 的代码情况：

if(responseCode.code === 200 && responseBody.has("access\_token")){

//如果 code 为 200， 并且返回的数据中存在 access\_token 关键字，则认为登录成功

tests["login"] = true;

//将返回的内容转为 json 格式，并且取到 access\_token 内容，添加到环境变量中

var jsonData = JSON.parse(responseBody);

//access\_token的取值方式视具体的 json 数据结构而定

postman.setEnvironmentVariable("token",jsonData.result.access\_token);

//跳转到 Request3 接口

postman.setNextRequest("Request3")

}else{

tests["login"] = false;

//登录失败，可以选择跳转到对应失败后的处理接口进行测试

//postman.setNextRequest("Other Request")

}

在接口 Request3 中使用变量 token :

我这边是将 token 放在头部信息中， 具体使用方式时接口参数规则而定。

运行并查看结果

运行集合测试，可以看到我们结果符合我们的预期，Request1 和 Request3 通过测试，Request2 被跳过，Request4 仍被执行。

————————————————

版权声明：本文为CSDN博主「\_wiky\_」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/cai\_iac/article/details/81030619