接口测试工具 Postman 使用实践

主要内容包括：

- 接口的定义
- 接口的分类
- 为何要进行接口测试
- 接口文档示例
- Postman工具简介
- 借助Postman完成HTTP请求接口测试
- Postman + Newman + Jenkins 实现接口自动化测试

一、接口定义

> 软件不同部分之间的交互接口。通常就是所谓的API——应用程序编程接口，其表现的形式是源代码。——[百度百科]

我们常说的接口一般指两种：

1. API：应用程序编程接口。程序间的接口
2. GUI：图形用户界面。人与程序的接口

这里我们所说的接口特指API接口。API接口定义：对协议进行定义的引用类型。

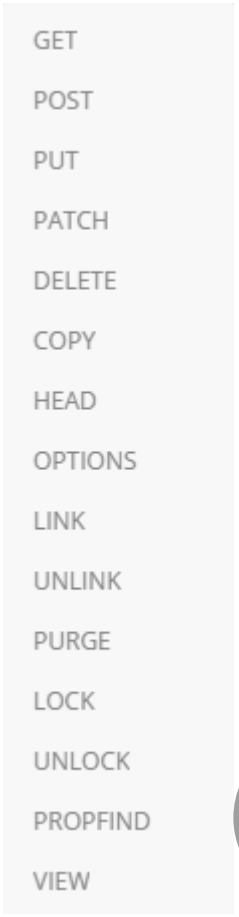
好多公司开发人员分前后端，他们之间如何配合工作的，就是其中一方定义接口，另一方来调用接口，以实现预期功能。

二、接口的分类

1. 接口分类

- HTTP接口
- Webservice接口
- RESTful接口

WebService接口是走soap协议，请求报文和返回报文都是xml格式，通过SoapUI工具进行测试；
HTTP API接口走HTTP协议，通过路径来区分调用的方法，请求报文入参有多种形式，返回报文一般为json串，最常见的是get和post方法。



三、为何要进行接口测试

1. 接口测试必要性

当今的系统复杂度不断上升，传统的测试方法成本急剧增加且测试效率大幅下降，所以就要做接口测试。同时，接口测试相对容易实现自动化持续集成，且相对UI自动化也比较稳定，可以减少人工回归测试人力成本与时间，缩短测试周期，支持后端快速发版需求。接口持续集成是为什么能低成本高收益的根源。现在很多系统前后端架构是分离的，从安全层面来说，只依赖前端进行限制已经完全不能满足系统的安全要求（绕过前面实在太容易）， 需要后端同样进行控制，在这种情况下就需要从接口层面进行验证。前后端传输、日志打印等信息是否加密传输也是需要验证的，特别是涉及到用户的隐私信息，如身份证，银行卡等。

2. 接口测试原理

模拟客户端向服务器发送请求报文，服务器接收请求报文后对相应的报文做处理并向客户端返回应答，客户端再接收应答的一个过程。

3. 接口测试范围

接口的功能、性能、安全性。重点关注数据的交换，传递和控制管理过程，还包括处理的次数。

接口测试对象是接口，但随着系统复杂度越来越高，接口越来越多，完全覆盖是一件很困难的事情。通常情况下主要测试最外层的两类接口：数据进入系统的接口（调用外部系统的参数为本系统使用）、数据流出系统接口（验证系统处理后的数据是否正常）

四、接口文档示例

1. 接口文档应该包括哪几部分？

- 接口说明
- 调用的url
- 请求方法（get、post）
- 请求参数，参数类型、请求参数说明
- 返回参数说明
- 返回示例

2. 示例

APP注册登录

简要描述：

- APP用户注册登录接口

请求URL：

- `http://[redacted]/s/login/appLogin`

请求方式：

- POST

参数：

参数名	必选	类型	说明
mobile	是	string	手机号
channel	是	string	渠道：APPIOS / APPANDROID
checkcode	是	string	验证码
deviceId	是	string	设备号

返回示例

```
{
  "data": {
    "code": 1,
    "addedBaby":false
  },
  "msg": "注册成功",
  "success": true,
  "total": 0
}
```

返回参数说明

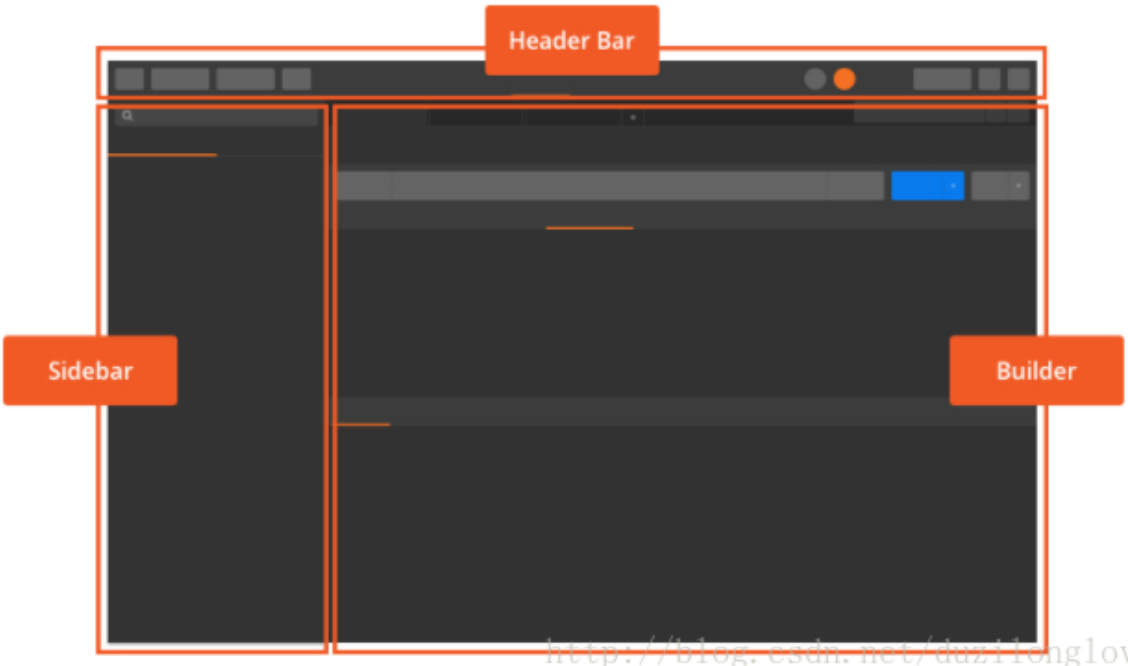
参数名	类型	说明
success	Boolean	返回状态，true：成功，false：失败
msg	String	返回信息描述
data	Objet	返回值，code：1-注册成功，2-登录成功, 当code为2时返回addedBaby：是否添加过宝宝（true-添加过，false-未添加）

备注

- 返回Cookie中存入deviceId：设备号，flagLogin：登录标志，channel：渠道

注：上图接口文档工具为ShowDoc

五、Postman工具简介



1. Sidebar侧边栏

Postman侧边栏允许你查找、管理请求和集合。侧边栏分为两个主要的选项卡，包括历史和集合选项卡。可以拖动右边的边来调整侧边栏的宽度。侧边栏也可以隐藏到小屏幕（标题栏 view—>toggle side bar）。

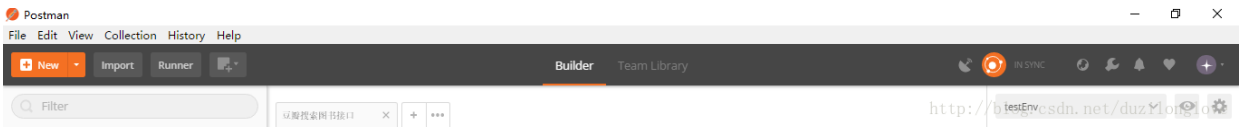
（**1**）历史选项卡

通过Postman应用程序发送的每个请求都保存在侧边栏的History选项卡中。

（**2**）集合选项卡

在侧栏中创建和管理集合选项卡的集合。

2. Header toolbar



Postman的顶部工具栏包含以下选项：

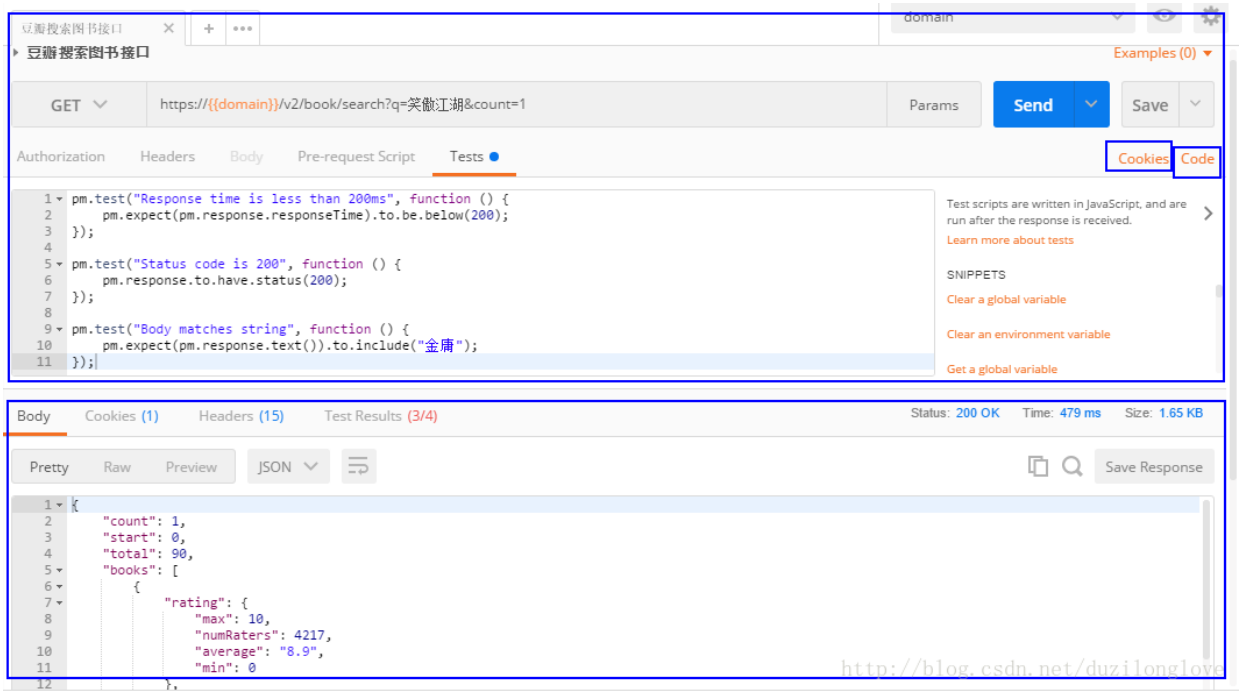
1. 新建按钮——可以新建请求，集合，环境等
2. 运行按钮-打开集合运行页面
3. 导入按钮——导入Postman文件、文件夹、form link等
4. 新窗口图标-打开一个新的tab页、新的窗口、新的runner等
5. 构建器和团队库选项卡——在请求生成器和Team Library视图之间切换
6. 抓取API请求图标——使用postman抓取API请求
7. 同步状态图标——同步API请求图标
8. 用户下拉——管理集合链接和你的个人资料或登录/登出，你的Postman帐户
9. 开放API集合（点击打开一个网址）
10. 通知图标-接收通知或广播
11. 设置图标——管理Postman应用程序设置，并找到其他支持资源
12. ♥——分享按钮

3. Builder

Postman通过选项卡布局，用于在构建器中发送和管理API请求。上半部分是请求构建器，下半部分是响应查看器。

1. Cookies——管理cookie模式是通过点击cookie链接访问的。该特性允许你管理与请求相关的cookie。

2. Code——生成的代码片段模式通过保存按钮下面的最右边的Code链接。该特性允许你生成与请求相关的代码片段，该请求支持20多种语言（http、java、go等语言）



4. Console

Postman有两个控制台，可以帮助我们了解系统后台到底发生了什么。

1. Postman Console——包含HTTP请求和响应的运行日志。来自脚本的日志消息(如在console. Log中)。这个功能只能在Postman的本地应用中使用。
2. DevTools Console——可以在开发期间记录诊断信息。

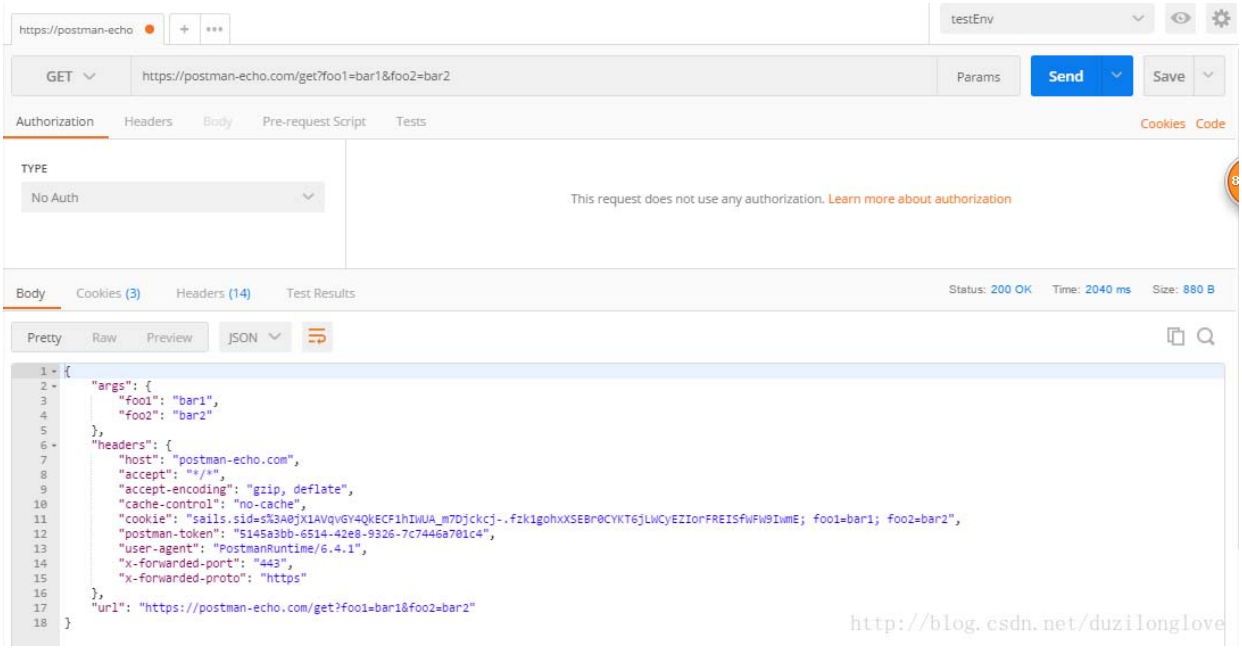
六、借助Postman完成HTTP请求接口测试

1. 借助Postman Echo 演示下各种请求的构建方法

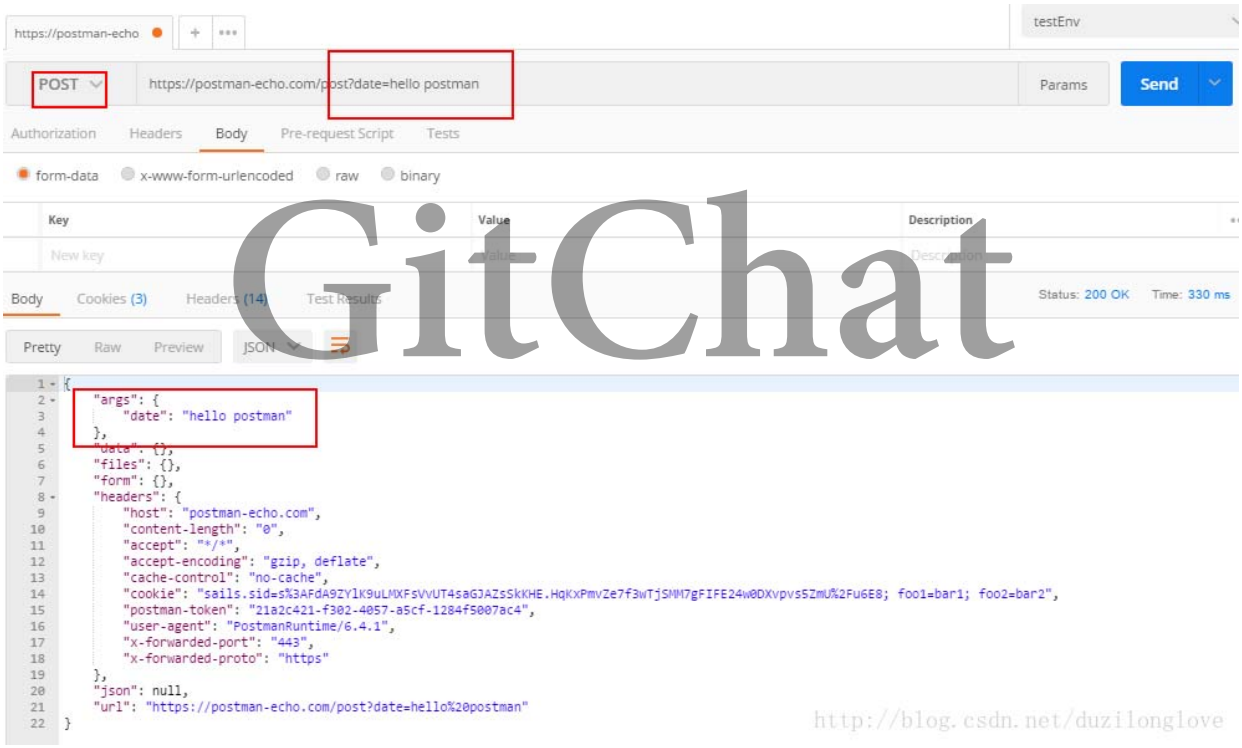
（1）Get 请求

https://postman-echo.com/get?foo1=bar1&foo2=bar2

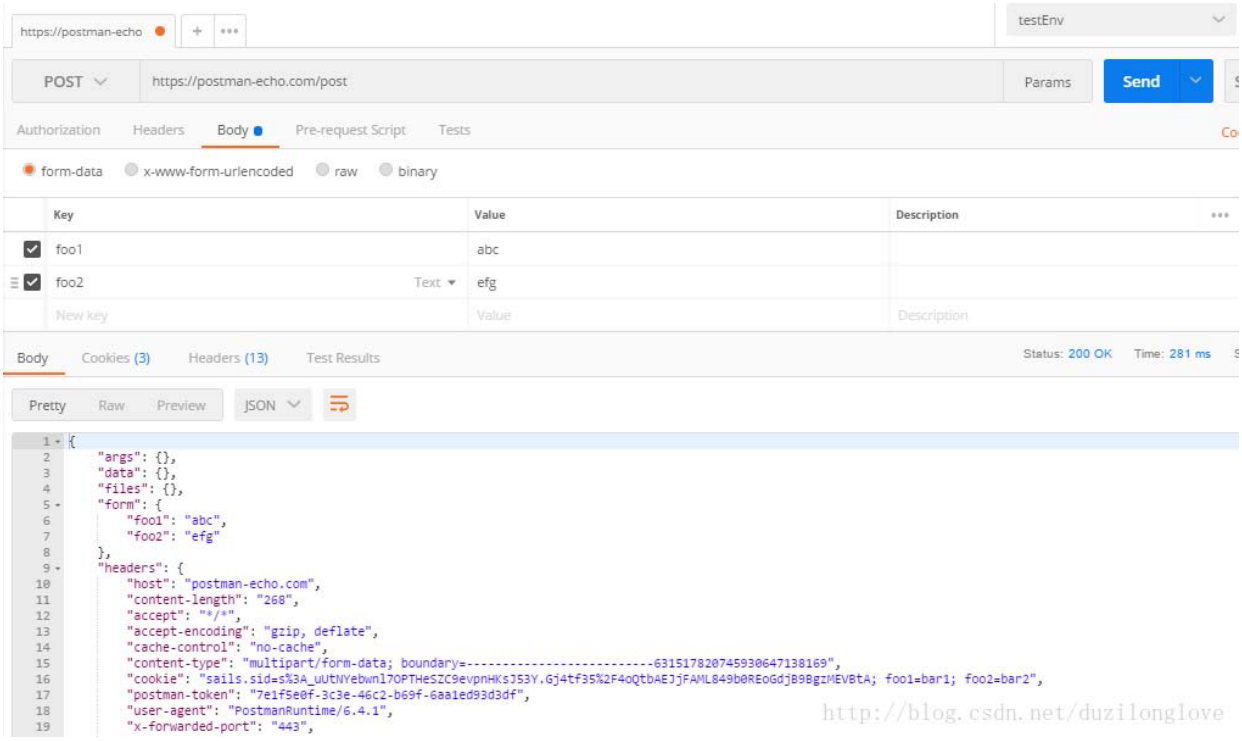
HTTP GET请求方法是从服务器检索数据。数据由惟一URI(统一资源标识符)标识。GET请求可以使用“查询字符串参数”将参数传递给服务器。例如，在下列请求中，http://example.com/hi/there?hand=wave，参数“hand”的值等于“wave”。



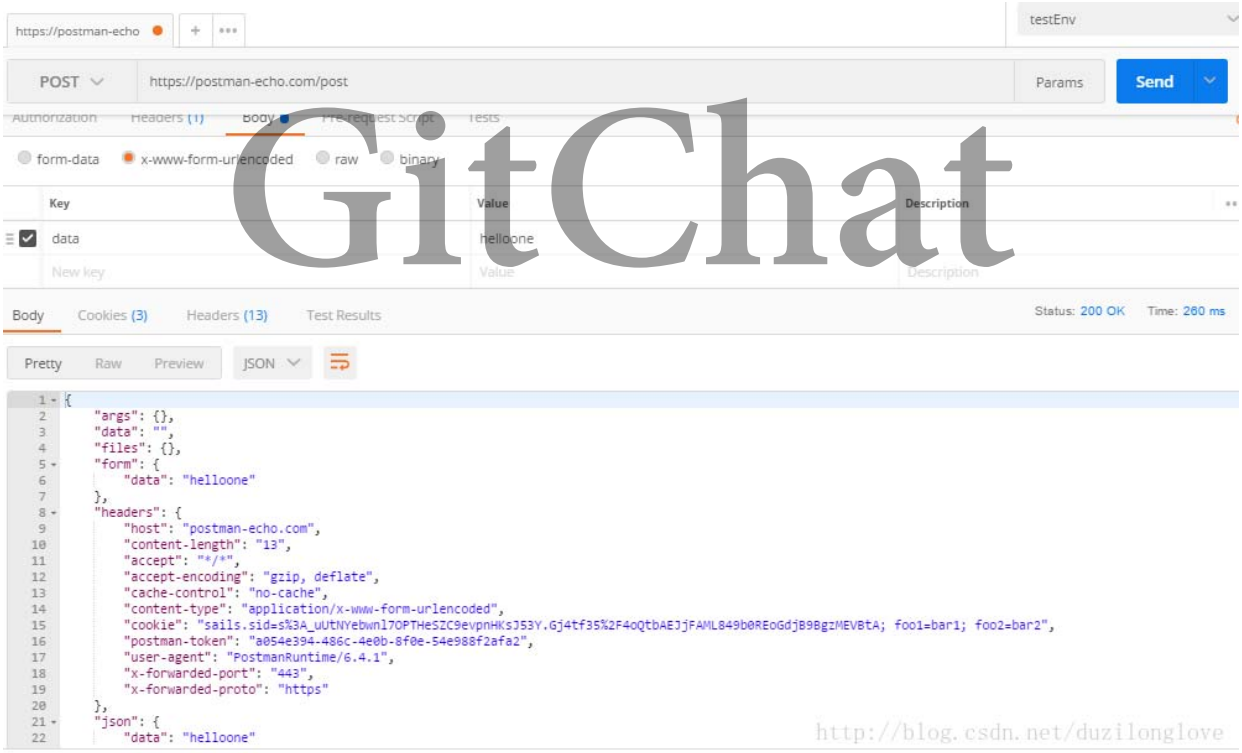
（2）POST：URI 传参



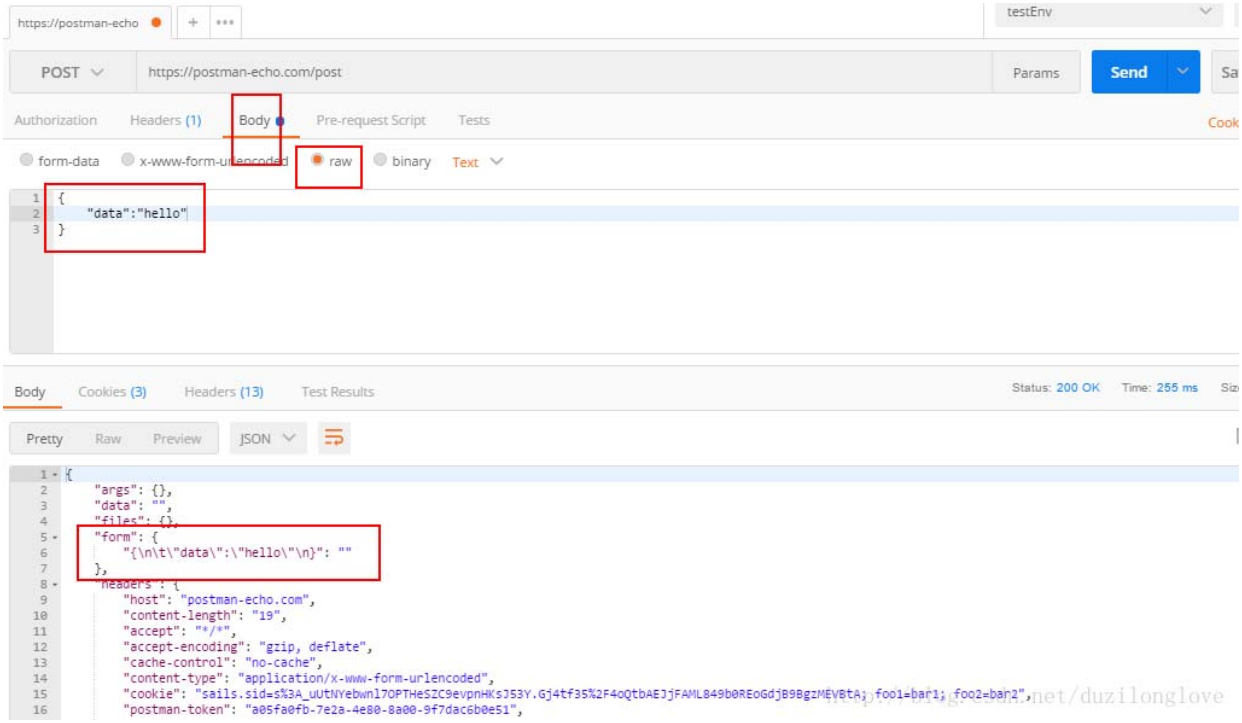
（3）POST：Form-data 传参



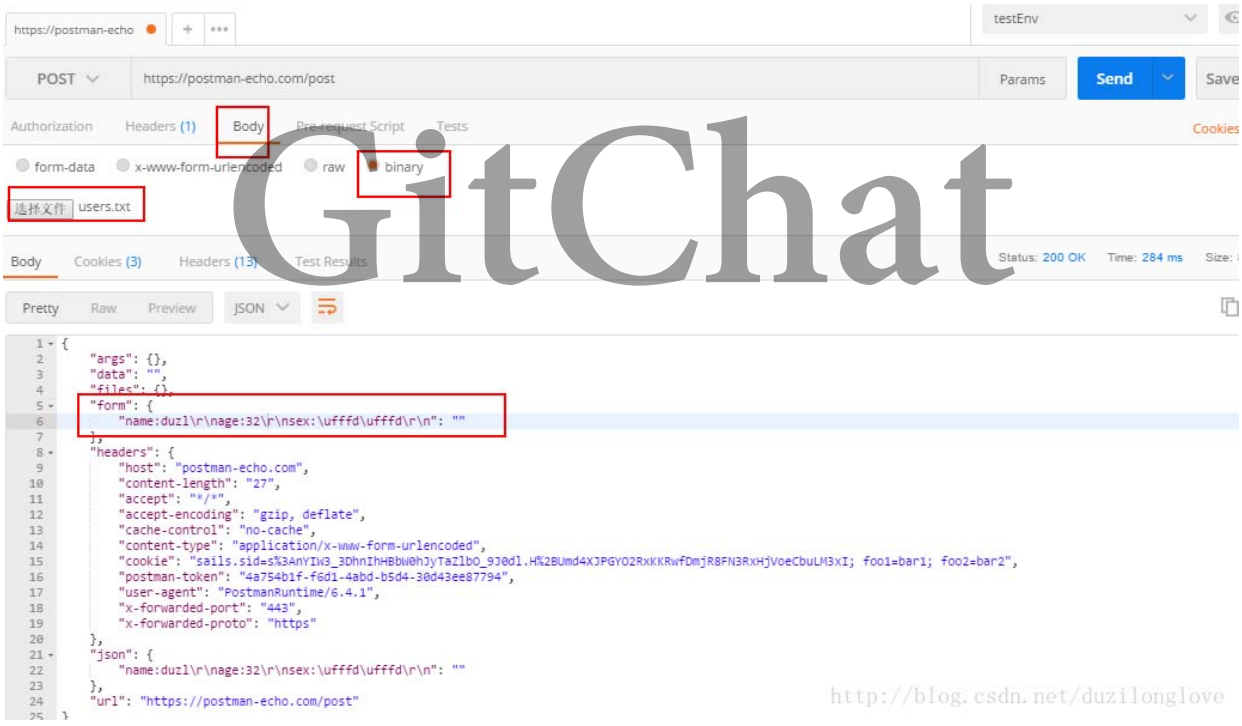
（4）POST： x-www-form-urlencoded传参



（5）POST： raw传参

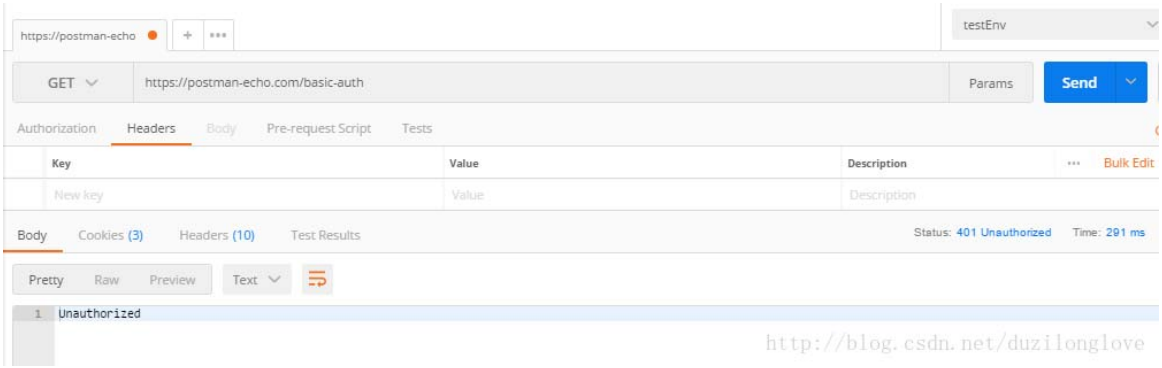


（6）POST： binary 传参

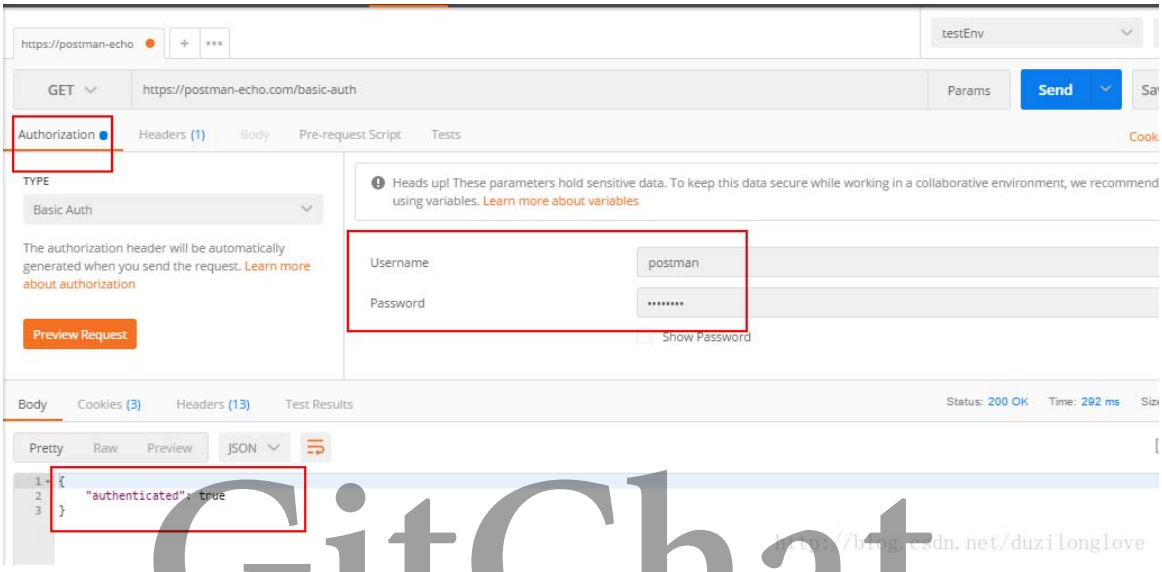


（7）Authentication Method——权限认证方法

- GET Basic Auth

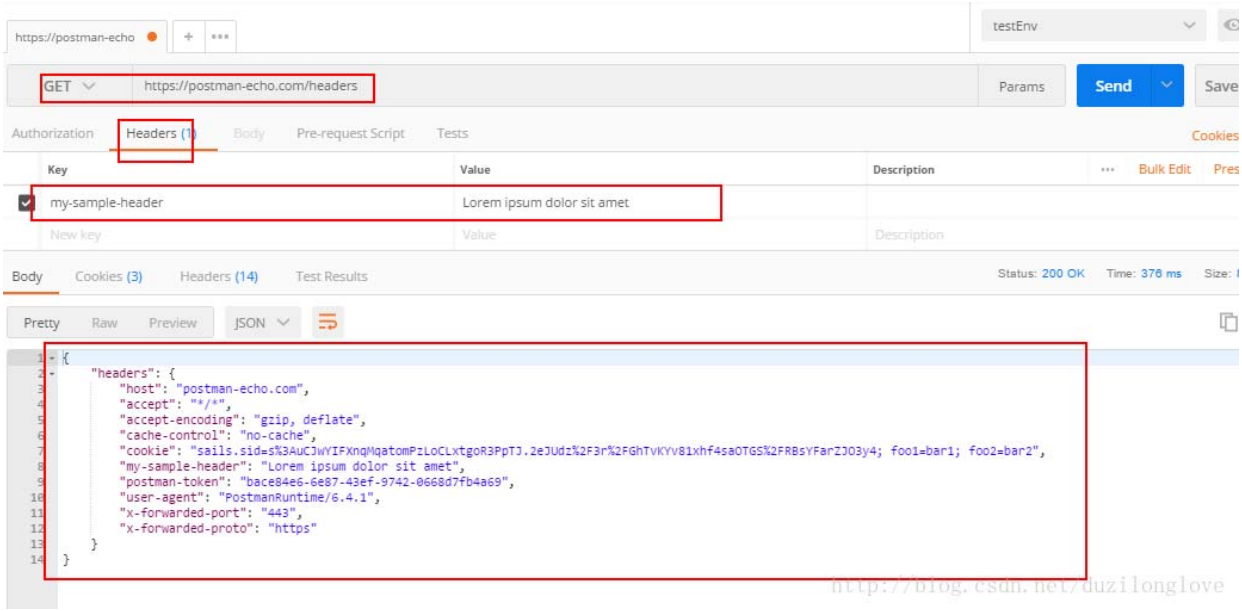


增加auth信息：



- DigestAuth
- Hawk Auth
- OAuth1.0(verify signature)

（8）Headers——添加header



2. 接下来，我们拿个开放API来演示下单一接口测试流程

示例API：https://developers.douban.com/wiki/?title=book_v2#get_book

搜索图书

GET https://api.douban.com/v2/book/search

参数	意义	备注
q	查询关键字	q和tag必传其一
tag	查询的tag	q和tag必传其一
start	取结果的offset	默认为0
count	取结果的条数	默认为20，最大为100

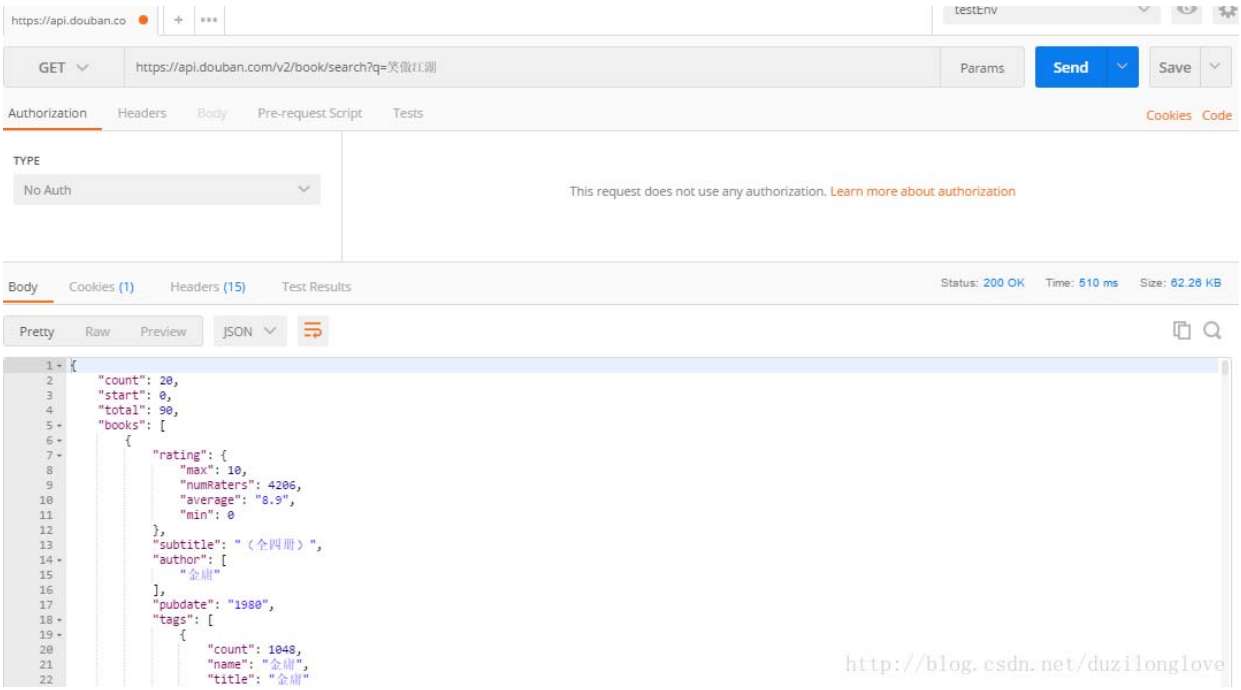
返回：返回status=200，

```
{
      "start": 0,
      "count": 10,
      "total": 30,
      "books" : [Book, ]
    }
```

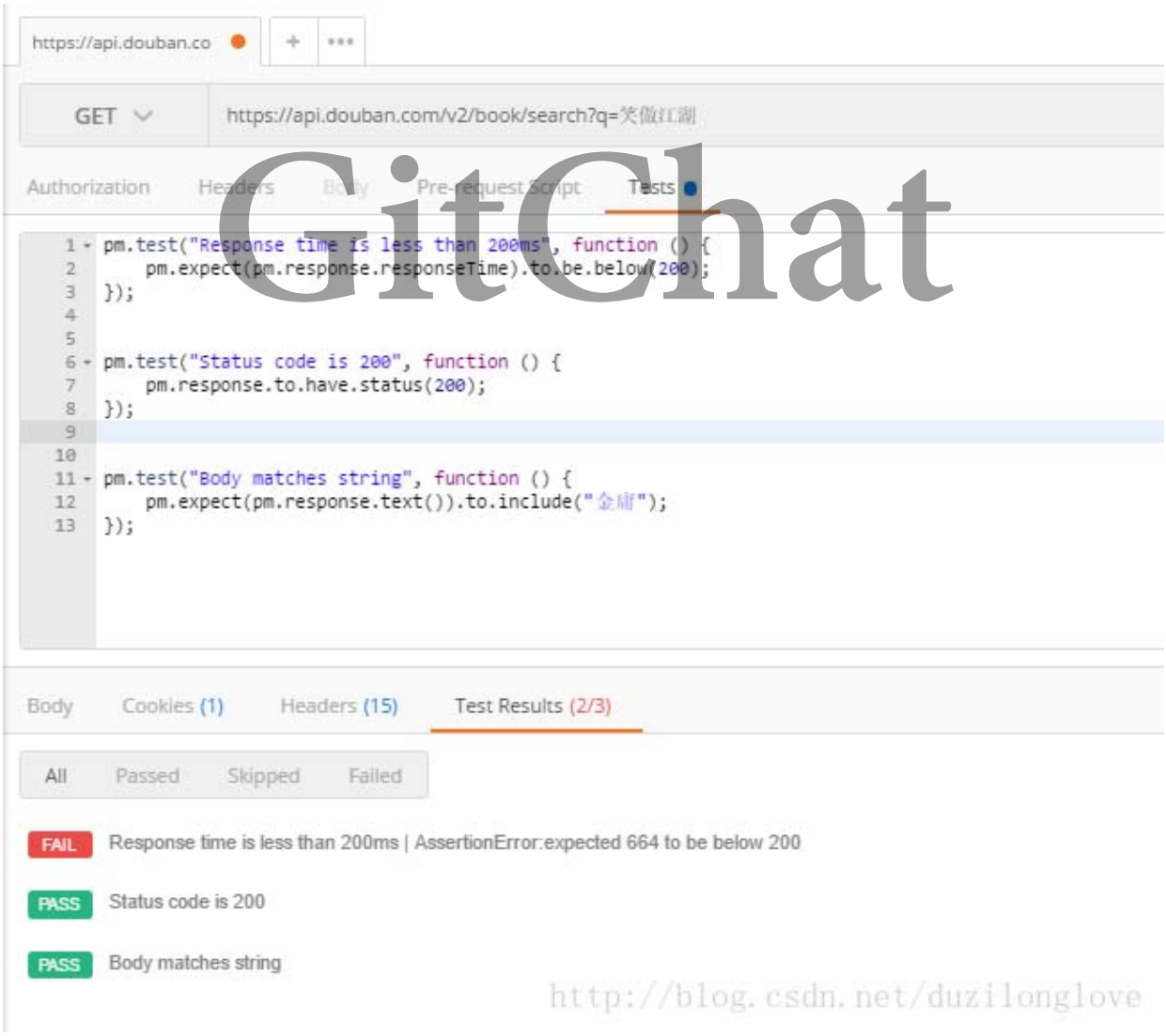
注：对于登录用户，若搜索结果图书在当前用户的图书收藏中，会在对应搜索结果信息中附加当前用户对此书的收藏信息，改部分的 Book 数据结构如下：

```
{
    ... (图书信息的其他部分)
    "current_user_collection": {
        "status":"read",
        "rating": {
            "max":5,
            "value":"5",
            "min":0
        },
        "updated":"2012-11-2012:08:04",
        "user_id":"33388491",
        "book_id":"6548683",
        "id":605519800
    }
}
```


步骤一：使用Postman工具发送该Get请求，如下图。



步骤二：添加测试。



上图针对该API添加了3个测试：

1. 要求响应时间小于200ms

2. 要求status code等于200
3. 要求Response body中包含字符串“金庸”

注：当然你还可以增加更多的测试点。

七、Postman + Newman + Jenkins 实现接口自动化测试

1. 准备工作（具体步骤参考附件文档-作者提供）

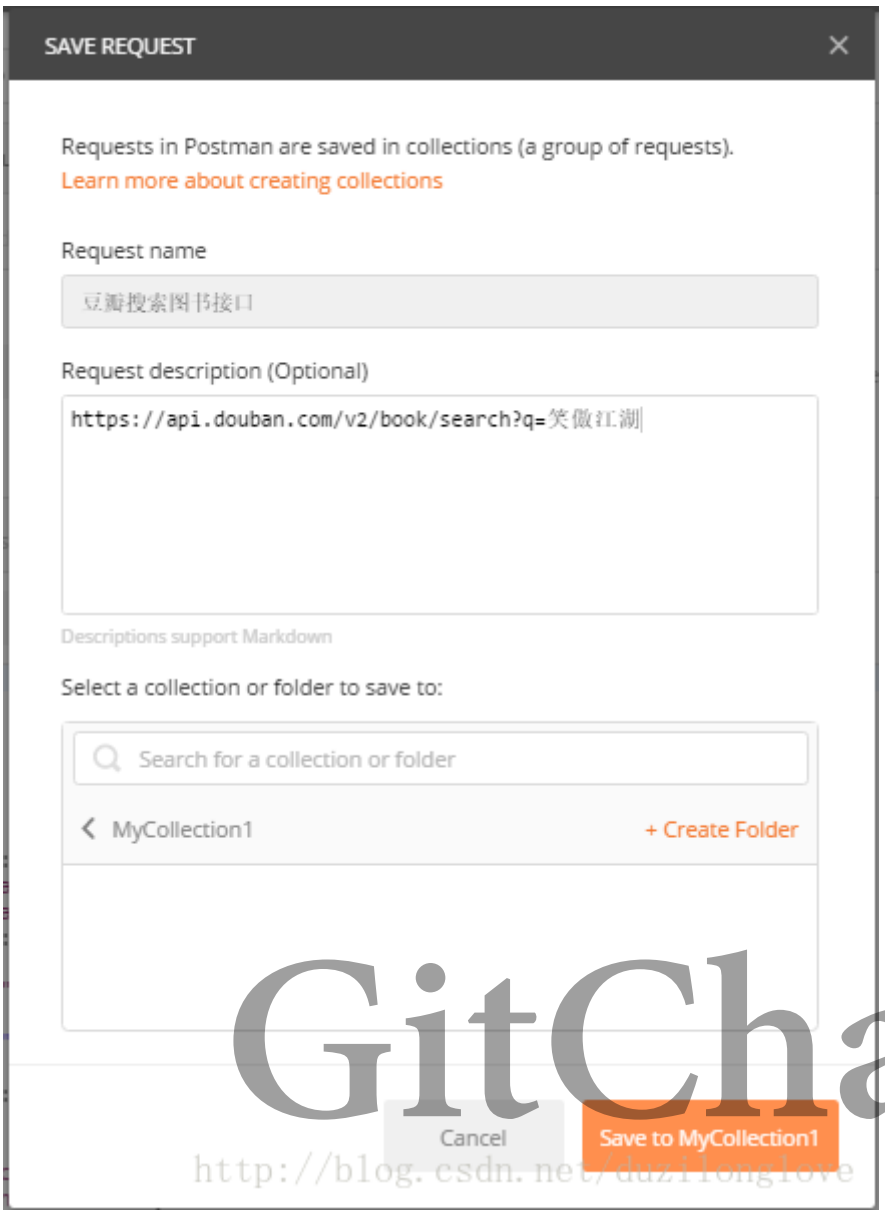
（**1**）安装**Newman** 工具

- 安装Node.js
- 安装Newman
- 查看Newman命令

（**2**）部署**Jenkins**

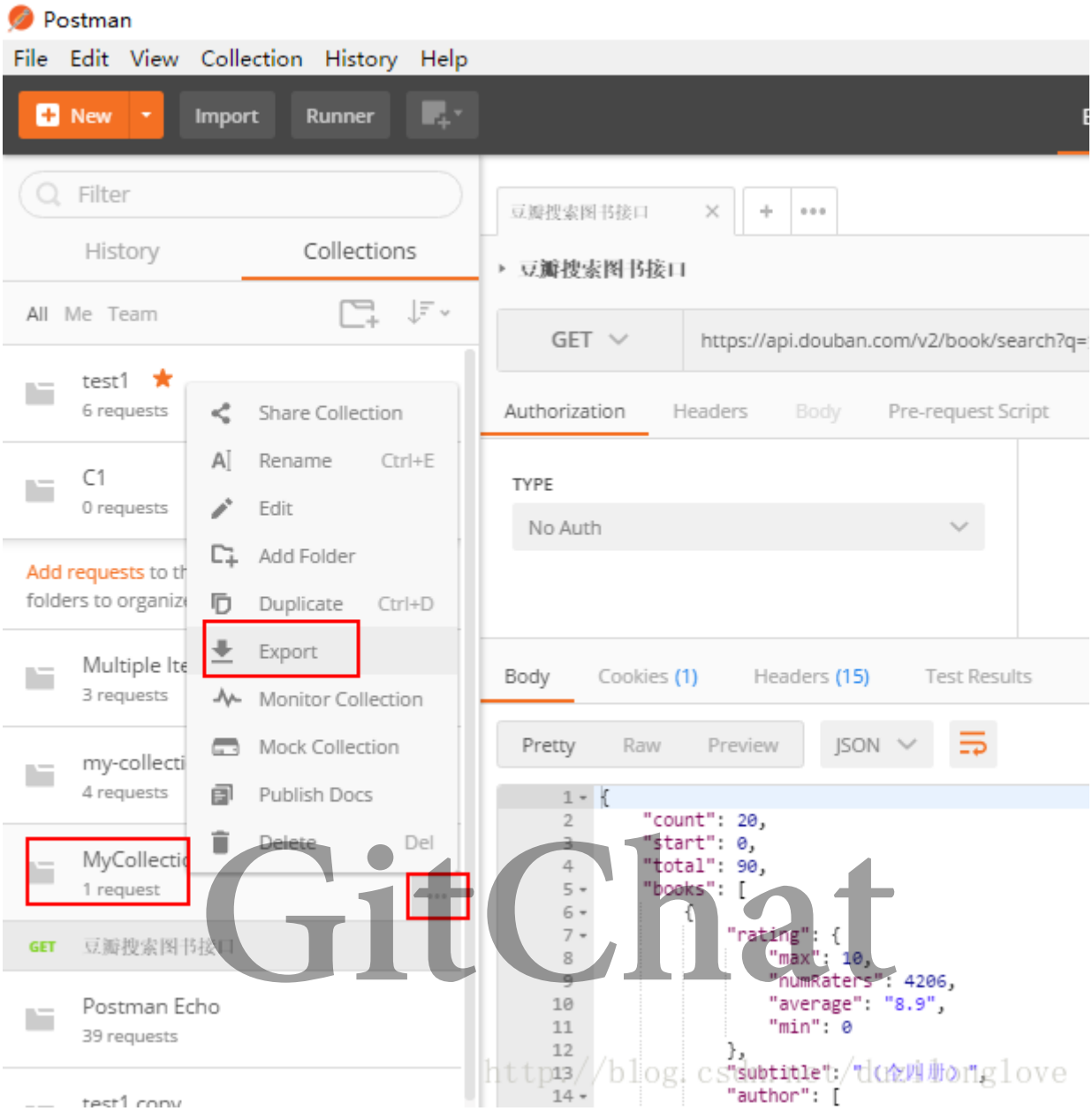
2. 将接口保存到集合

点击Save按钮，将接口保存到一个集合（可以保存到一个现有集合中或者新建一个集合），如下图：



3. 将集合保存到本地

将集合保存到本地，文件为.json格式，如下图：



4. 命令行通过Newman 运行集合

（1）打开命令行窗口，运行如下命令：

```
D:\git-local>newman run MyCollection1.postman_collection.json -g
globals.postman_globals1.json
```

（2）执行结果如下：

```
D:\git-local>newman run MyCollection1.postman_collection.json -g globals.postman_globals1.json
newman

MyCollection1

→ 豆瓣搜索图书接口
  GET https://api.douban.com/v2/book/search?q=笑傲江湖&count=1  [200 OK, 1.77KB, 270ms]
  1. Response time is less than 200ms
  ✓  Status code is 200
  ✓  Body matches string

┌─────────────────────────┬──────────┬──────────┐
│                         │ executed │   failed │
├─────────────────────────┼──────────┼──────────┤
│              iterations │        1 │        0 │
├─────────────────────────┼──────────┼──────────┤
│                requests │        1 │        0 │
├─────────────────────────┼──────────┼──────────┤
│            test-scripts │        1 │        0 │
├─────────────────────────┼──────────┼──────────┤
│      prerequest-scripts │        0 │        0 │
├─────────────────────────┼──────────┼──────────┤
│              assertions │        3 │        1 │
├─────────────────────────┴──────────┴──────────┤
│ total run duration: 440ms                     │
├───────────────────────────────────────────────┤
│ total data received: 1.22KB (approx)          │
├───────────────────────────────────────────────┤
│ average response time: 270ms                  │
└───────────────────────────────────────────────┘

  #  failure                  detail

1.  AssertionError            expected 270 to be below 200
                              at assertion:0 in test-script
                              inside "豆瓣搜索图书接口"
                                                  http://blog.csdn.net/duzilonglove
D:\git-local>
```

可以看到，其中两条断言passed，一条断言failed，失败的原因是，我们期望接口响应时间小于200 ms，但是本次接口请求响应时间是270 ms。

5. 通过Jenkins 调用Newman，执行接口测试

执行一次构建，构建失败（上面的断言失败，我们并未修复），查看构建失败原因。

```
[test1] $ cmd /c call C:\Windows\TEMP\jenkins6981821130418322094.bat

C:\Jenkins\workspace\test1>C:\Users\admin\AppData\Roaming\npm\newman -c D:\git-local\MyCollection1.postman_collection.json -g D:\git-local\globals.postman_globals1.json
newman: the v2.x CLI options are deprecated. You should use newman run <path> [options] instead.
        refer https://github.com/postmanlabs/newman/blob/develop/MIGRATION.md for details.
newman

MyCollection1

→ 豆瓣搜索图书接口
  GET https://api.douban.com/v2/book/search?q=笑傲江湖&count=1 [200 OK, 1.77KB, 439ms]
  1. Response time is less than 200ms
  ✓  Status code is 200
  ✓  Body matches string

┌─────────────────────────┬────────────┬────────────┐
│                         │ executed   │ failed     │
├─────────────────────────┼────────────┼────────────┤
│              iterations │          1 │          0 │
├─────────────────────────┼────────────┼────────────┤
│                requests │          1 │          0 │
├─────────────────────────┼────────────┼────────────┤
│            test-scripts │          1 │          0 │
├─────────────────────────┼────────────┼────────────┤
│      prerequest-scripts │          0 │          0 │
├─────────────────────────┼────────────┼────────────┤
│              assertions │          3 │          1 │
├─────────────────────────┴────────────┴────────────┤
│ total run duration: 571ms                         │
├───────────────────────────────────────────────────┤
│ total data received: 1.22KB (approx)              │
├───────────────────────────────────────────────────┤
│ average response time: 439ms                      │
└───────────────────────────────────────────────────┘

#  failure         detail

1.  AssertionError  expected 439 to be below 200
                    at assertion:0 in test-script
                    inside "豆瓣搜索图书接口"
Build step 'Execute Windows batch command' marked build as failure
Finished: FAILURE
```

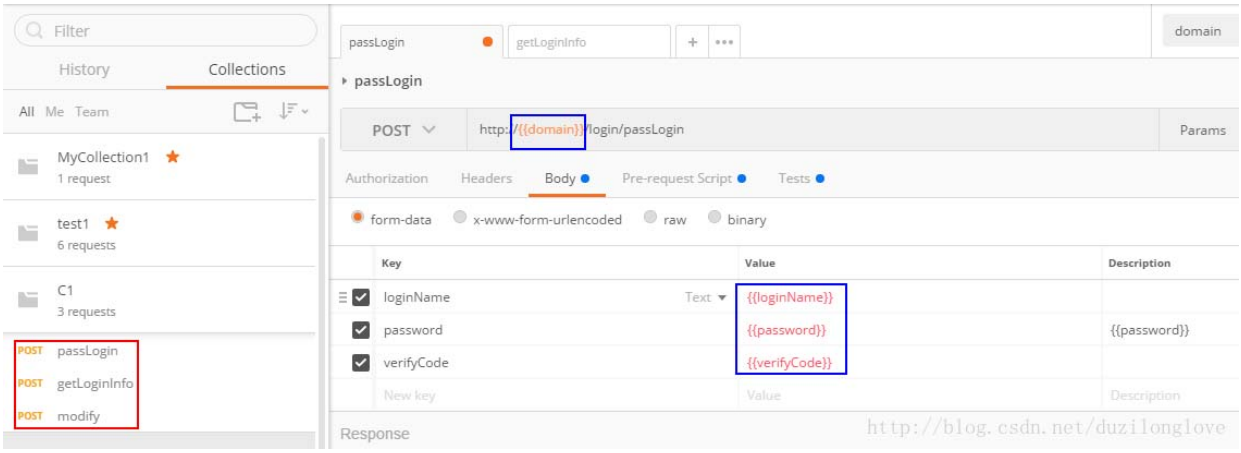

6. 假设开发修复了接口bug

接口响应时间减少了，我们需要回归测试。（我们将断言响应小于200 ms，修改成1000 ms，让断言passed）

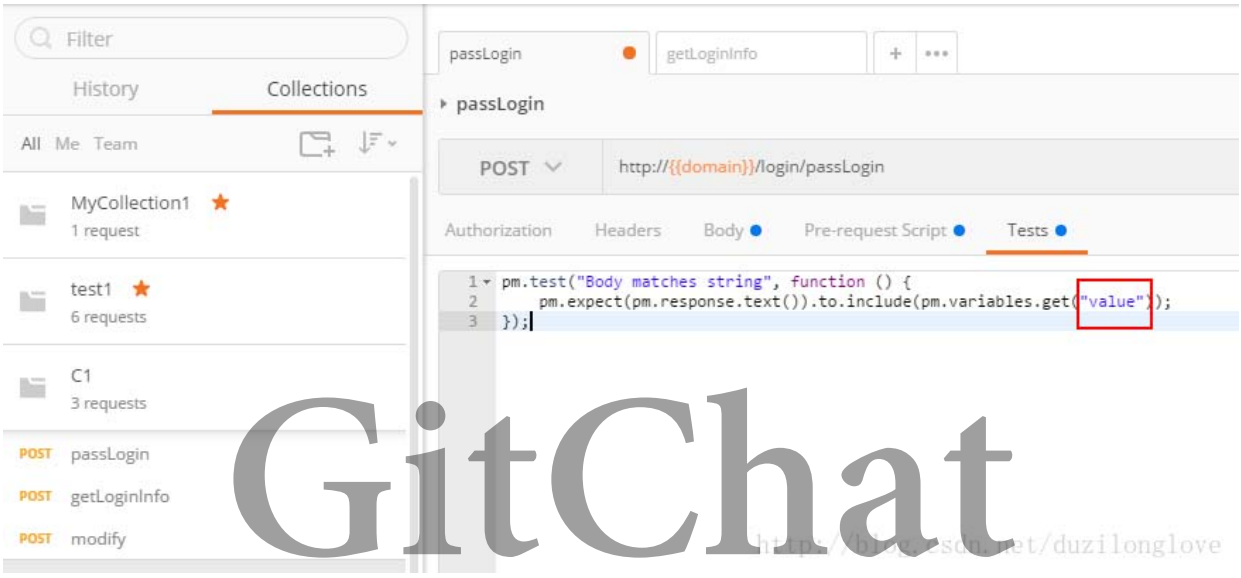

```
Started by user stormsprite
Building in workspace C:\Jenkins\workspace\test1
[test1] $ cmd /c call C:\Windows\TEMP\jenkins2530137079124905 98.bat

C:\Jenkins\workspace\test1>C:\Users\admin\AppData\Roaming\npm\newman -c D:\git-local\MyCollection3.postman_collection.json -g D:\git-local\globals.postman_globals1.json
newman: the v2.x CLI options are deprecated. You should use newman run <path> [options] instead.
        refer https://github.com/postmanlabs/newman/blob/develop/MIGRATION.md for details.
newman

MyCollection1

→ 豆瓣搜索图书接口
  GET https://api.douban.com/v2/book/search?q=笑傲江湖&count=1 [200 OK, 1.77KB, 103ms]
  ✓  Response time is less than 1000ms
  ✓  Status code is 200
  ✓  Body matches string

┌─────────────────────────┬────────────┬────────────┐
│                         │ executed   │ failed     │
├─────────────────────────┼────────────┼────────────┤
│              iterations │          1 │          0 │
├─────────────────────────┼────────────┼────────────┤
│                requests │          1 │          0 │
├─────────────────────────┼────────────┼────────────┤
│            test-scripts │          1 │          0 │
├─────────────────────────┼────────────┼────────────┤
│      prerequest-scripts │          0 │          0 │
├─────────────────────────┼────────────┼────────────┤
│              assertions │          3 │          0 │
├─────────────────────────┴────────────┴────────────┤
│ total run duration: 226ms                         │
├───────────────────────────────────────────────────┤
│ total data received: 1.22KB (approx)              │
├───────────────────────────────────────────────────┤
│ average response time: 103ms                      │
└───────────────────────────────────────────────────┘
Finished: SUCCESS
```


7. 演示一个如何调用data file 参数化用例

我这里有一个集合，3个接口，第一个接口为登录接口，第二个接口为获取登录用户信息接口，第三个接口为修改密码接口。登录接口如下：



测试脚本如下：

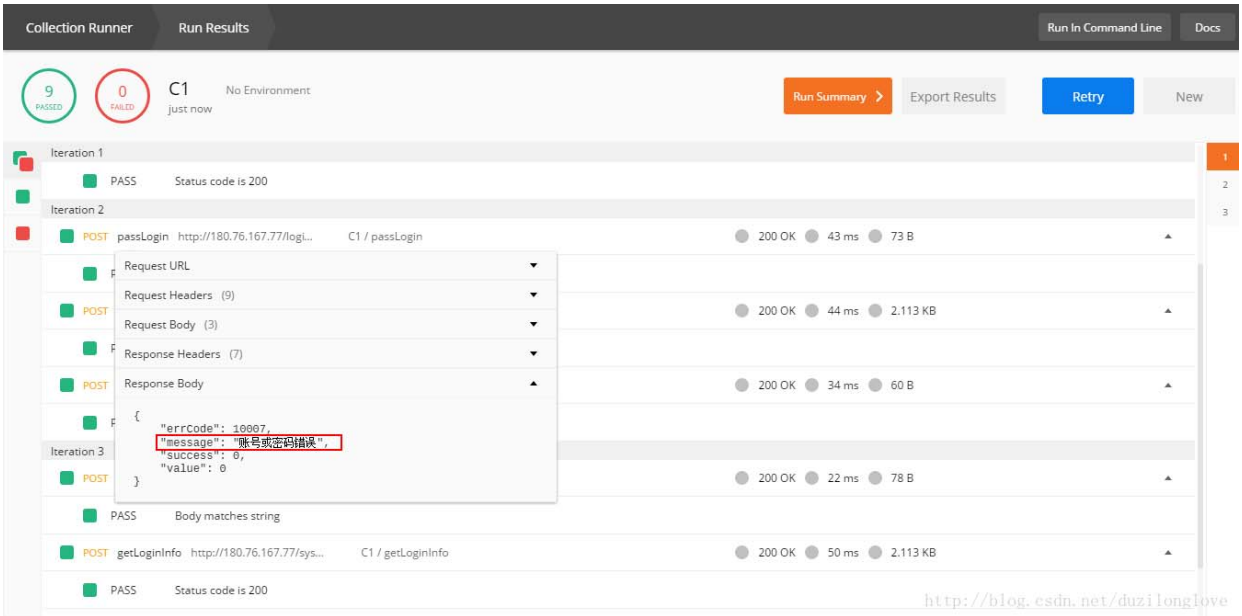


参数化json文件内容如下：

```
[{
  "loginName": "duzl",
  "password": "admin123",
  "verifyCode": "adf",
  "value": "/index"
}, {
  "loginName": "duzl",
  "password": "admin",
  "verifyCode": "adf",
  "value": "账号或密码错误"
}, {
  "loginName": "duzl",
  "password": "",
  "verifyCode": "adf",
  "value": "参数password不能为空"
}]
```

```
[{
    "loginName": "duzl",
    "password": "admin123",
    "verifyCode": "adf",
    "value": "/index"
}, {
    "loginName": "duzl",
    "password": "admin",
    "verifyCode": "adf",
    "value": "账号或密码错误"
}, {
    "loginName": "duzl",
    "password": "",
    "verifyCode": "adf",
    "value": "参数password不能为空"
}]
```

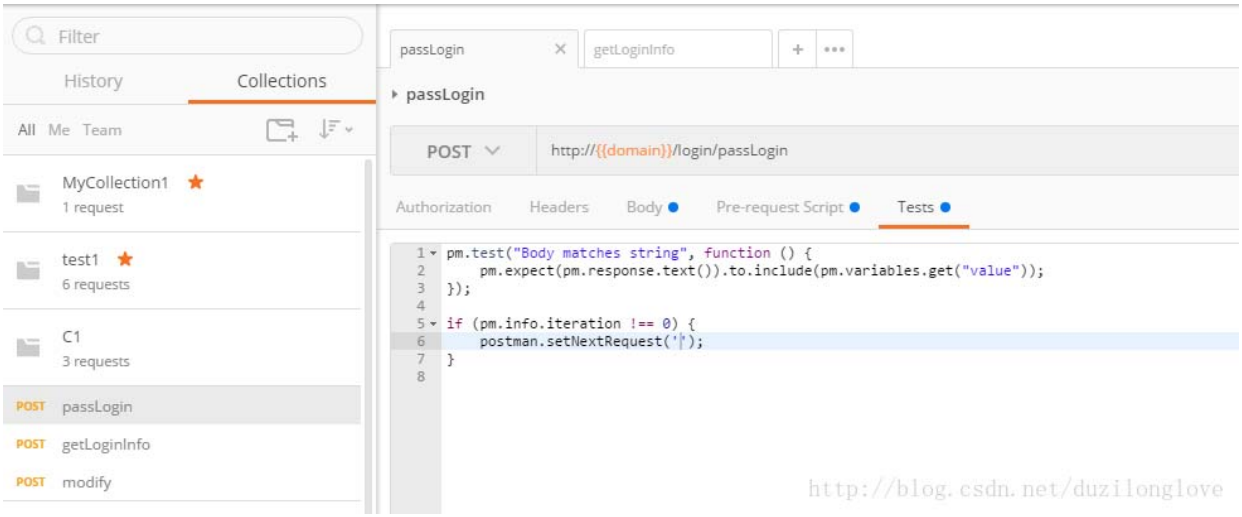

（1）好我们调用json文件，执行下集合，结果如下：



结果还不错，执行了3次，参数都是取自用例文件（json文件），断言也取自用例文件。

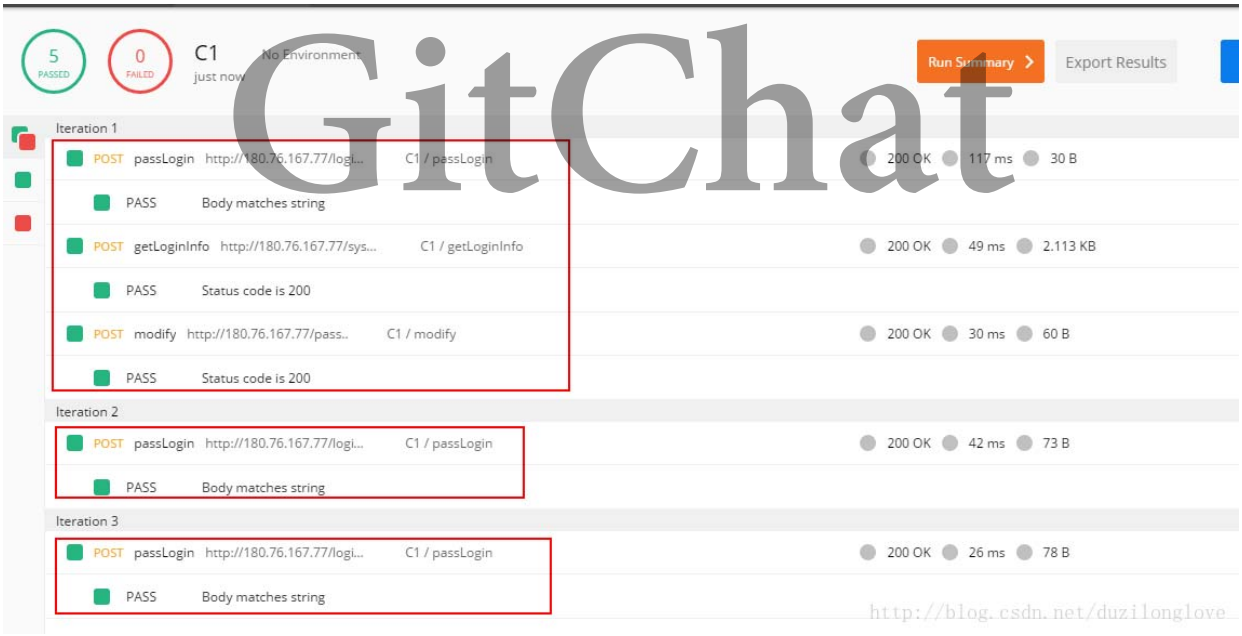
美中不足的是，第二个和第三个接口也跟着迭代了3次（这并不是我们期望的结果），这是因为集合运行器中的迭代次数是针对所有接口的设置。

（2）那如果，我们想第一个接口运行3遍，第二、三个接口只运行一遍，该如何做呢？Postman 给我们提供了一个内置方法，设置接口运行顺序 `postman.setNextRequest('');`。



注意：迭代次数从0开始。

当迭代次数!==0时，就停止本次迭代（意思就是，第一次迭代全运行，第二次迭代开始就不执行第二、三个接口了），好，再次运行集合，看看结果：



很好，第一次迭代，执行了3个接口；第二、三次迭代只执行了第一个接口。