

**软件测试课程设计报告**

学 号： 20191001171

班级序号： 111192-12

学生姓名： 滕德淋

指导教师： 张剑波

**中国地质大学地理与信息工程学院软件工程系**

**2021年 8 月**

# 实习二 软件综合测试技术

## 1. UI测试

### 1.1 测试对象

**测试软件：哔哩哔哩平台**

哔哩哔哩，英文名称：bilibili，简称B站，现为中国年轻世代高度聚集的文化社区和视频平台，该网站于2009年6月26日创建，被粉丝们亲切地称为“B站” 。2018年3月28日，哔哩哔哩在美国纳斯达克上市 。2020年9月15日，B站定制的视频遥感卫星——“哔哩哔哩视频卫星”成功升空 。2021年3月29日，哔哩哔哩正式在香港二次上市 。

B站早期是一个ACG（动画、漫画、游戏）内容创作与分享的视频网站。 经过十年多的发展，围绕用户、创作者和内容，构建了一个源源不断产生优质内容的生态系统，B站已经涵盖7000多个兴趣圈层的多元文化社区，曾获得QuestMobile研究院评选的“Z世代偏爱APP”和“Z世代偏爱泛娱乐APP”两项榜单第一名并入选“BrandZ”报告2019最具价值中国品牌100强 。

2021年8月19日，哔哩哔哩公布了第二季度未经审计的财务报告。财报显示，B站二季度营收达44.9亿元人民币，同比增长72%。

**软件基本功能：**

1. **登录**

* 通过账号密码登录
* 通过手机号和验证码登录

1. **注册**

通过手机号注册，一个手机号只能注册一个账号

1. **关键字搜索视频**

通过在搜索栏中可以根据关键字搜索

1. **选择视频**

搜索出来的视频可以根据排序和筛选进行选择视频

1. **给视频点赞**

可以给视频点赞和投币

1. **评论视频**

也可以评论该视频

1. **收藏视频**

可以收藏该视频，方便下一次再看的时候容易查找

1. **关注UP主**

如果觉得该UP主的视频很有趣，可以关注该UP主，方便后面查找他的视频

1. **等等**

该平台还有其他很多功能，我主要测试前面所说的主要功能

### 1.2 测试环境

#### 1.2.1 基本测试环境

|  |  |
| --- | --- |
| 操作系统 | Linux/Ubuntu18.04 |
| UI测试自动化工具 | Selenium |
| 模拟浏览器 | Google Chrome 92.0.4515.159 |
| 脚本编写语言 | Python |

#### 1.2.2 Selenium学习

由于刚开始不会自动化UI测试，所以去学习了一下Selenium，学的是python版本。首先学习了配置Selenium环境，首先需要下载一个Chrome的驱动程序，来模拟浏览器，当然也可以不用，就用本机的浏览器。

* **安装客户端库：**

**pip install selenium**

* **安装浏览器驱动：**

浏览器驱动 是和 浏览器对应的。 不同的浏览器 需要选择不同的浏览器驱动。目前主流的浏览器中，谷歌 Chrome 浏览器对 Selenium自动化的支持更加成熟一些。强烈推荐大家使用 Chrome浏览器。

谷歌浏览器下载：<https://www.google.cn/chrome/>

浏览器驱动下载： <https://chromedriver.storage.googleapis.com/index.html>

注意浏览器驱动 必须要和浏览器版本匹配，下图红圈里面的版本号 就是和浏览器版本号对应的。



**注意：**驱动和浏览器的版本号越接近越好，但是略有差别，比如87和86 ，通常也没有什么问题。

打开目录，里面有3个zip包，分别对应Linux、Mac、Windows平台。

如果我们是Windows平台的电脑，就下载 chromedriver\_win32.zip，如果是其他，就选择对应的压缩包进行下载。

这是个zip包，下载下来之后，解压里面的程序文件 chromedriver.exe 到某个目录下面，注意这个目录的路径最好是没有中文名和空格的，比如，解压到 d:\webdrivers 目录下面。也就是保证我们的Chrome浏览器驱动路径为 d:\webdrivers\chromedriver.exe；

如果是Linux，建议解压在/usr/bin/目录下,保证路径为/usr/bin/chromedriver；

**到这里，Selenium的环境就安装成功了，之后就可以写python代码进行自动化UI测试了。**

* **Selenium基本使用方法：**

这里需要自己去学习如何使用Selenium进行自动化测试，个人写了一些笔记在CSDN博客里面：

Selenium入门：<https://blog.csdn.net/xiten/article/details/120047008>

Selenium基础： <https://blog.csdn.net/xiten/article/details/120047503>

由于学习时间比较短，所以只学了一点基础操作：

学习视频教程网址：<https://www.bilibili.com/video/BV1Z4411o7TA>

学习博客： <http://www.byhy.net/tut/auto/selenium/>

### 1.3 测试场景1-B站登录功能

#### 1.3.1 场景描述

准备好已经注册好的账号，然后模拟用户登录场景。

由于B站有图片文字定位验证码,所以调用了图鉴api进行破解。

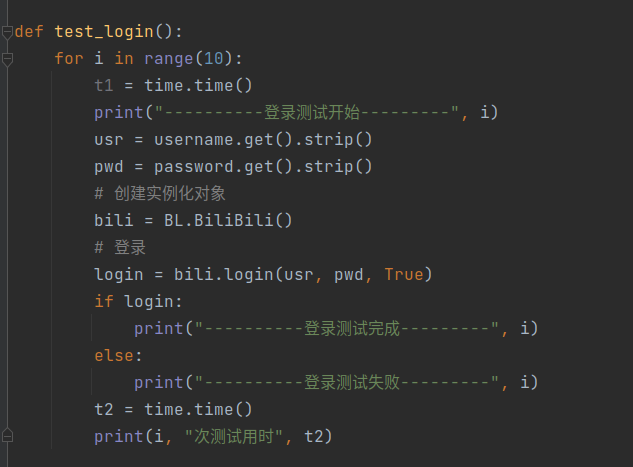
另外B站由于有防止爬虫措施，有两个版本，所以需要对应的写两个定位方法。

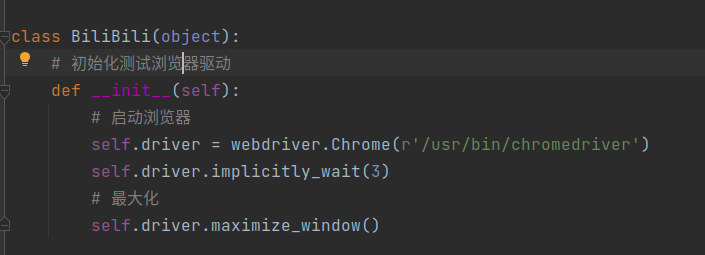
启动测试程序之后，输入账号和密码之后，点击登录10次，进行测试。

#### 1.3.2 测试脚本

由于我将整个测试对象封装为了一个类，所以这里给出部分核心代码:

* **核心代码：**







**验证码定位破解：**

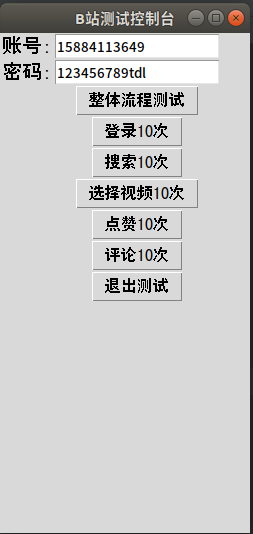
首先保存整个页面截图，然后将截图传入图鉴api，接口返回值是识别到验证文字的x和yd的坐标，然后经过字符串处理，调用Selenium的鼠标点击事件，在相应的位置进行点击，最后确认登录。（有的时候会识别失败，因为图鉴的接口模型可能不是很好，没有特别好的解决办法）





* **设计思路：**

首先写了一个简陋的测试控制台，便于操作，由于是用python的tkinter库写的，所以界面很丑：



然后提供了五个主要的功能测试和一个整体流程测试，根据需要测试的功能，点击对应的按钮，然后就开始测试。

登录功能思路：

首先使用Selenium打开B站主页，然后用CSS定位的方法，找到登录的页面，然后获取对应的username和password的输入框，将设定好的账号和密码输入进去。然后点击确定。输入验证码，最后看是否登录成功，如果成功则进行下一次的登录测试。

会测试10次登录功能，并且会输出每一次登录所用的时间。（一个账号不能频繁登录，注意多准备几个账号）

#### 1.3.3 测试过程

**表1-1 测试过程记录**

|  |  |
| --- | --- |
| 场景名称 | 用户登录 |
| 测试目的 | 模拟人工测试用户登录功能 |
| 测试详细过程 | 1. 打开测试控制台      1. 打开首页      1. 跳转到登录页      1. 自动输入账号和密码并点击登录按钮      1. 调用图鉴api破解验证码点击确定      1. 登录成功跳转到首页 |
| 10次平均测试时长 | 17.452392959594725 s |
| 测试成功率 | 100%(10/10) |
| 测试结果 | 用户登录功能正常 |

**表1-2 10次测试结果统计**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试时长（s） | 是否成功 | 备注（版本） |
| 1 | 16.71632480621338 | 是 | 1 |
| 2 | 15.690885066986084 | 是 | 1 |
| 3 | 16.536834239959717 | 是 | 1 |
| 4 | 15.688499450683594 | 是 | 1 |
| 5 | 19.344293117523193 | 是 | 2 |
| 6 | 18.44426155090332 | 是 | 2 |
| 7 | 18.832704782485962 | 是 | 2 |
| 8 | 15.869462251663208 | 是 | 1 |
| 9 | 18.678386926651 | 是 | 2 |
| 10 | 18.722277402877808 | 是 | 2 |

### 1.4 测试场景2-B站搜索功能

#### 1.4.1 场景描述

用户登录成功之后，可以进行关键字搜索视频。

编写脚本模拟搜索功能，进行测试。

#### 1.4.2 测试脚本

* 核心代码：





* 设计思路：

首先进行登录操作，登录成功之后回到主页面。

获取主页面搜索框的控件定位，并且输入“python”关键字，然后自动点击搜索。

搜索成功会跳转到详情页面。

#### 1.4.3 测试过程

**表1-3 测试过程记录**

|  |  |
| --- | --- |
| 场景名称 | 用户搜素 |
| 测试目的 | 模拟人工测试用户搜素功能 |
| 测试详细过程 | 1. 登录成功回到首页      1. 在输入框中输入“python”      1. 点击按钮跳转到详情页 |
| 10次平均测试时长 | 2.244815802574158 s |
| 测试成功率 | 100%(10/10) |
| 测试结果 | 用户搜索功能正常 |

**表1-4 10次测试结果统计**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试时长（s） | 是否成功 | 备注（版本） |
| 1 | 4.253305912017822 | 是 | 2 |
| 2 | 1.3204684257507324 | 是 | 1 |
| 3 | 4.234903573989868 | 是 | 2 |
| 4 | 1.4625868797302246 | 是 | 1 |
| 5 | 1.3490734100341797 | 是 | 1 |
| 6 | 1.3893685340881348 | 是 | 1 |
| 7 | 1.3454620838165283 | 是 | 1 |
| 8 | 1.3835432529449463 | 是 | 1 |
| 9 | 1.4194746017456055 | 是 | 1 |
| 10 | 4.289971351623535 | 是 | 2 |

### 1.5 测试场景3-B站选择视频功能

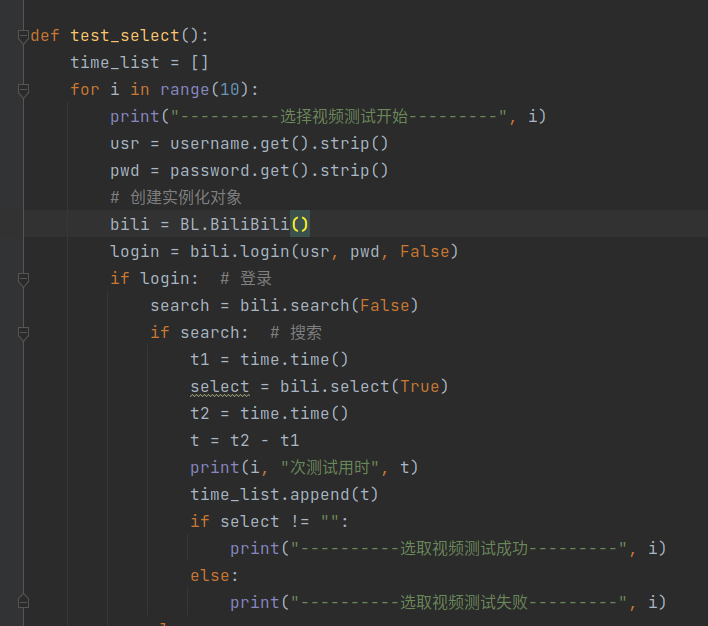
#### 1.5.1 场景描述

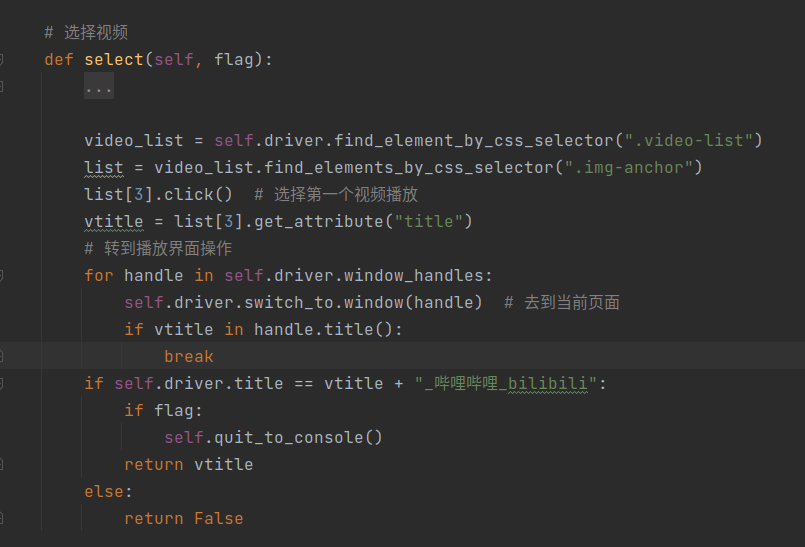
搜索成功之后，测试选择视频。

选择成功，跳转到播放视频页面。

#### 1.5.2 测试脚本

* 核心代码：





* 设计思路：

先经过登录、搜索之后，再进行选择视频。

由于搜索之后，会出现很多视频，所以可以获取所有视频的列表(这里可以进行爬虫），我们随机的选择一个视频进行点击，然后跳转进入播放视频页面。

#### 1.5.3 测试过程

**表1-5 测试过程记录**

|  |  |
| --- | --- |
| 场景名称 | 用户选择视频 |
| 测试目的 | 模拟人工测试用户选择视频功能 |
| 测试详细过程 | 1. 登录、搜索之后进入详情页      1. 获取所有视频播放url（不重要，可省略）      1. 随机选择一个视频进入播放页 |
| 10次平均测试时长 | 0.7288979291915894 s |
| 测试成功率 | 100%(10/10) |
| 测试结果 | 用户选择视频功能正常 |

**表1-6 10次测试结果统计**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试时长（s） | 是否成功 | 备注（版本） |
| 1 | 0.7243654727935791 | 是 | 2 |
| 2 | 0.7090039253234863 | 是 | 2 |
| 3 | 0.7057726383209229 | 是 | 2 |
| 4 | 0.6477124691009521 | 是 | 2 |
| 5 | 0.6422836780548096 | 是 | 2 |
| 6 | 0.6799042224884033 | 是 | 2 |
| 7 | 0.9315297603607178 | 是 | 1 |
| 8 | 0.7959885597229004 | 是 | 2 |
| 9 | 0.7328784465789795 | 是 | 1 |
| 10 | 0.7195401191711426 | 是 | 2 |

### 1.6 测试场景4-B站点赞功能

#### 1.6.1 场景描述

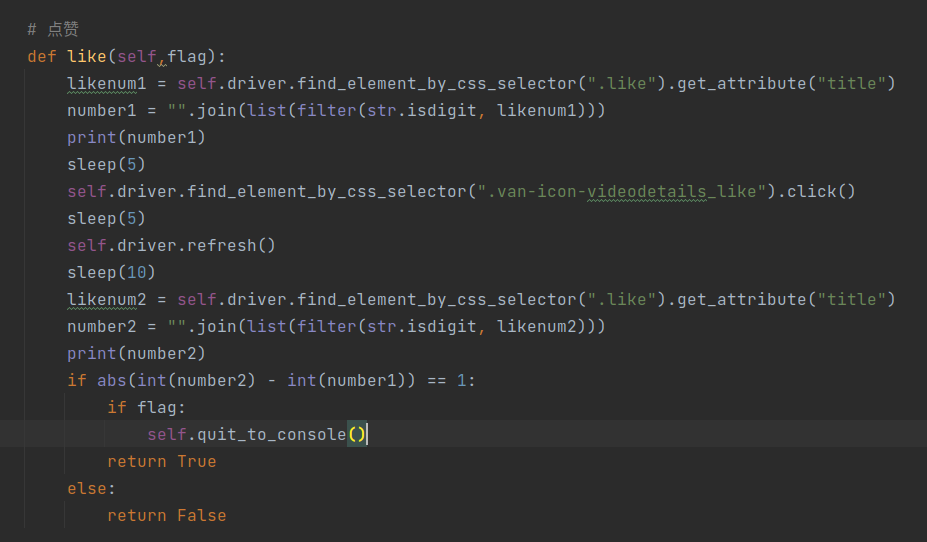
当用户觉得这个视频很好时，可以进行点赞。该脚本模拟点赞的行为，进行测试点赞功能。

由于一个视频只能点一次赞，所以判定点赞成功的标准是点赞前后数量相差1。

#### 1.6.2 测试脚本

* 核心代码：





* 设计思路：

在登录、搜索、选择视频之后，进入了播放视频的详情页面，我们可以看到有一个小标识，点击它就可以进行点赞。我们要测试的话，必须要判断如何才算点赞成功呢，所以我们必须得到点赞前后的点赞数进行对比才行。所以我将前后差1作为判断点赞成功的标准。

#### 1.6.3 测试过程

**表1-7 测试过程记录**

|  |  |
| --- | --- |
| 场景名称 | 用户点赞 |
| 测试目的 | 模拟人工测试用户点赞功能 |
| 测试详细过程 | 1. 登录、搜索、选择之后进入播放页面      1. 点击点赞按钮      1. 刷新页面   由于点赞这个操作执行之后，会进行数据库的修改，涉及到后端，所以，这里需要进行刷新页面才可以得到最新的数据。所以这里是非常耗时的。   1. 对比前后点赞数量 |
| 10次平均测试时长 | 20.33382946252823 s |
| 测试成功率 | 80%(8/10) |
| 测试结果 | 用户点赞功能正常 |

**表1-8 10次测试结果统计**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试时长（s） | 是否成功 | 备注（版本） |
| 1 | 20.326748609542847 | 是 | 1 |
| 2 | 20.326992988586426 | 是 | 1 |
| 3 |  | 否 | 2 |
| 4 | 20.307193756103516 | 是 | 1 |
| 5 |  | 否 | 2 |
| 6 | 20.399142742156982 | 是 | 2 |
| 7 | 20.47975206375122 | 是 | 2 |
| 8 | 20.28445529937744 | 是 | 2 |
| 9 | 20.279006004333496 | 是 | 1 |
| 10 | 20.2673442363739 | 是 | 2 |

### 1.7 测试场景5-B站评论功能

#### 1.7.1 场景描述

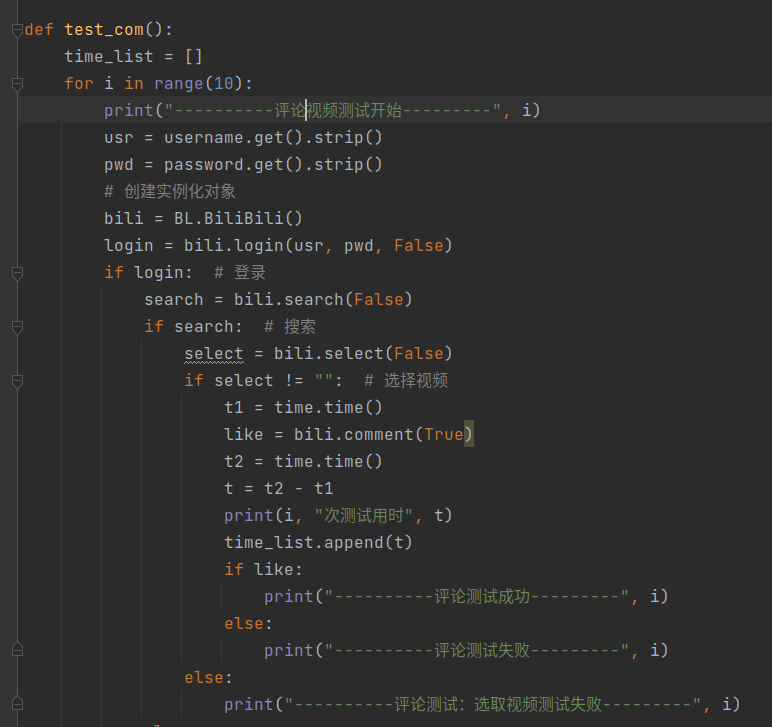
该脚本模拟的是用户对视频进行评论的功能。

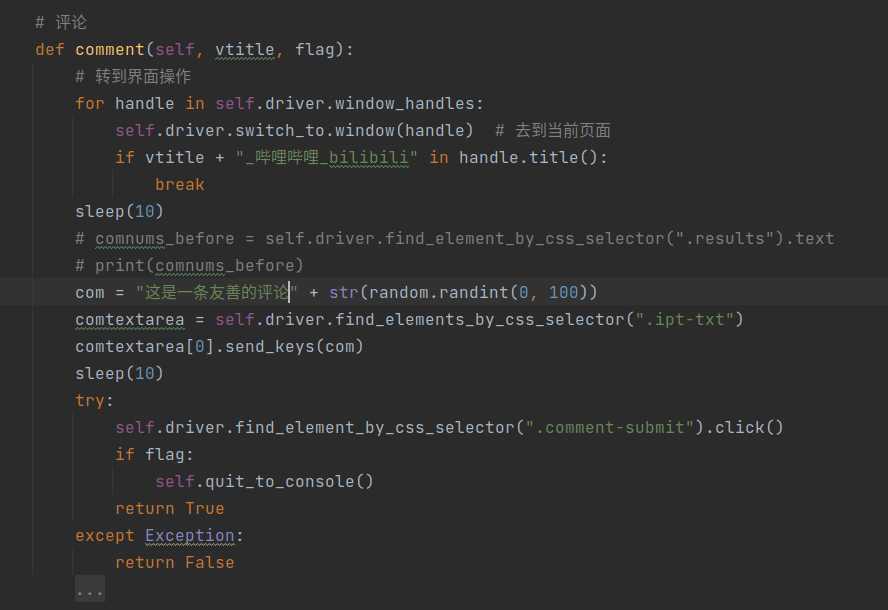
由于平台有限制，同一个用户不能重复评论相同的内容，而且评论的内容还必须进过审核。

所以判断是否评论成功就不能用数量来判断了，因为我尝试过了，数量是能更新的，但是更新的速度非常慢，可能这中间不仅仅是加入数据库这么简单，可能还有审核之类的操作。所以我判断是否成功的标准就是程序是否出错。

#### 1.7.2 测试脚本

* 核心代码：





* 设计思路：

其实这里之前的步骤跟点赞一样，只不过这里之前的步骤可以不用点赞。所以设计思路和点赞的思路一样。只不过这里的判断标准换了，另外不能重复评论也可以用随机数拼接解决。

#### 1.7.3 测试过程

**表1-9 测试过程记录**

|  |  |
| --- | --- |
| 场景名称 | 用户评论 |
| 测试目的 | 模拟人工测试用户评论功能 |
| 测试详细过程 | 1. 打开播放详情页面      1. 在评论区输入内容      1. 点击确定评论      1. 评论区显示评论 |
| 10次平均测试时长 | 20.402442741394044 s |
| 测试成功率 | 100%(10/10) |
| 测试结果 | 用户评论功能正常 |

**表1-10 10次测试结果统计**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试时长（s） | 是否成功 | 备注（版本） |
| 1 | 20.388011932373047 | 是 | 1 |
| 2 | 20.421759843826294 | 是 | 1 |
| 3 | 20.42155957221985 | 是 | 1 |
| 4 | 20.44981813430786 | 是 | 1 |
| 5 | 20.372491598129272 | 是 | 1 |
| 6 | 20.383891344070435 | 是 | 1 |
| 7 | 20.37850069999695 | 是 | 2 |
| 8 | 20.47348165512085 | 是 | 1 |
| 9 | 20.396491289138794 | 是 | 1 |
| 10 | 20.33842134475708 | 是 | 2 |

## 2. 接口测试

### 2.1 测试环境

|  |  |
| --- | --- |
| 操作系统 | Linux/Ubuntu18.04 |
| 接口测试工具 | Postman |
| 测试接口平台 | 聚合数据  <https://www.juhe.cn/docs> |
| 脚本编写语言 | Javascript |

#### 2.1.1 postman介绍

Postman一款非常流行的API调试工具。其实，开发人员用的更多。因为测试人员做接口测试会有更多选择，例如Jmeter、soapUI等。不过，对于开发过程中去调试接口，Postman确实足够的简单方便，而且功能强大。

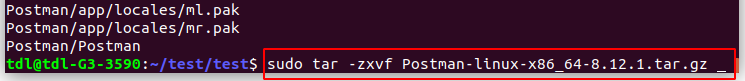
* Postman最早是作用chrome浏览器插件存在的，所以，可以到chrome商店搜索下载安装，因为重所周知的原因，所以，大家都会找别人共享的postman插件文件来安装。由于2018年初Chrome停止对Chrome应用程序的支持。
* Postman提供了独立的安装包，不再依赖于Chrome浏览器了。同时支持MAC、Windows和Linux，推荐你使用这种方式安装。

#### 2.1.2 postman安装

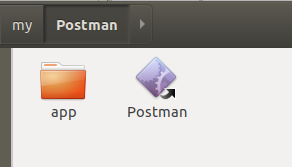
官网下载地址： <https://www.getpostman.com/downloads/>

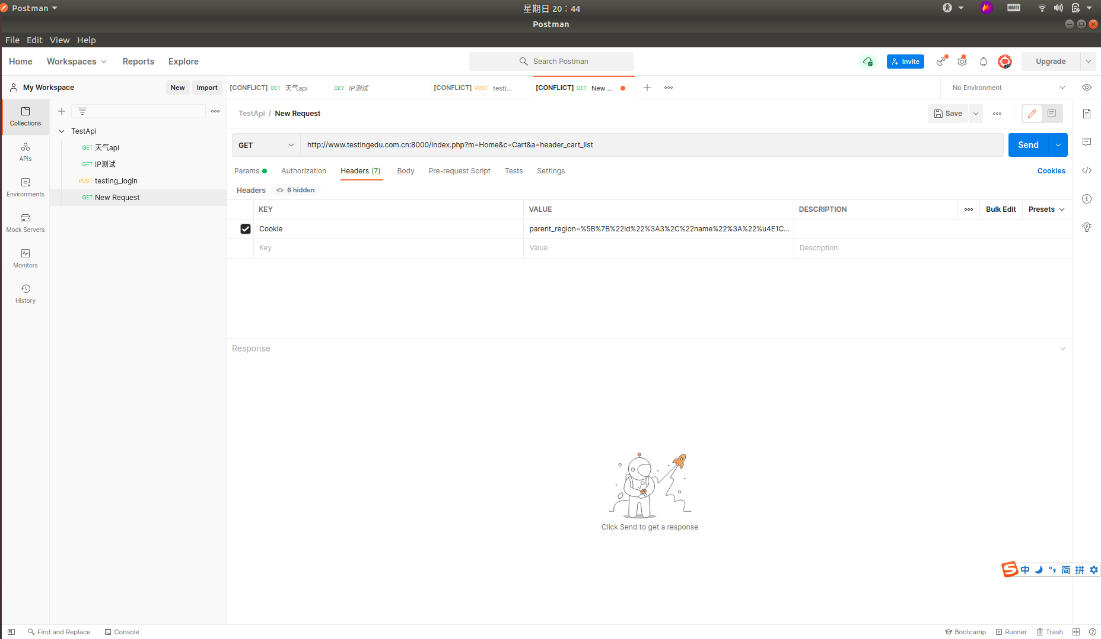
根据自己对应的操作系统进行下载压缩包

Linux解压： **sudo tar -zxvf '/home/tdl/下载/Postman-linux-x64-7.9.0.tar.gz'**



解压完成之后，会有一个文件夹，里面有一个postman的快捷方式，双击就可以运行软件





安装过程比较简单，如果是Windows直接双击exe程序就行。

### 2.2 测试对象1- 二维码生成

#### 2.2.1 接口描述

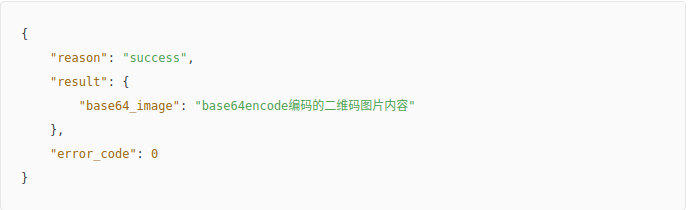
**表2-1 接口信息描述**

|  |  |
| --- | --- |
| 接口名称 | 二维码生成 |
| 接口类型 | http get/post |
| 接口地址 | <http://apis.juhe.cn/qrcode/api> |
| 接口功能 | 按照设定的参数、生成二维码 |

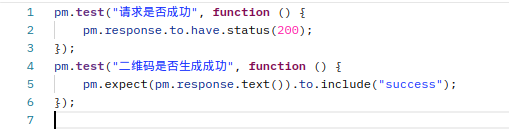
**表2-2 接口参数信息描述**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 必填 | 类型 | 说明 |
| 1 | key | 是 | string | 应用APPKEY，申请数据系统分配 |
| 2 | text | 否 | string | 二维码内容 |
| 3 | el | 否 | string | 纠错等级，el可用值：h\q\m\l，例如：h |
| 4 | bgcolor | 否 | string | 背景色代码，例如：ffffff |
| 5 | fgcolor | 否 | string | 前景色代码，例如：000000 |
| 6 | logo | 否 | string | logo图片URL地址或base64encode编码的图片内容，需要urlencode |
| 7 | w | 否 | int | 尺寸大小（像素），例如：300 |
| 8 | m | 否 | int | 边距大小（像素），例如：10 |
| 9 | lw | 否 | int | logo宽度（像素），例如：60 |
| 10 | type | 否 | int | 返回模式，1:二维码图片以base64encode编码返回 2:直接返回二维码图像，默认1 |

**返回示例：**



#### 2.2.2 测试断言



#### 2.2.3 测试过程

##### 2.2.3.1 创建请求

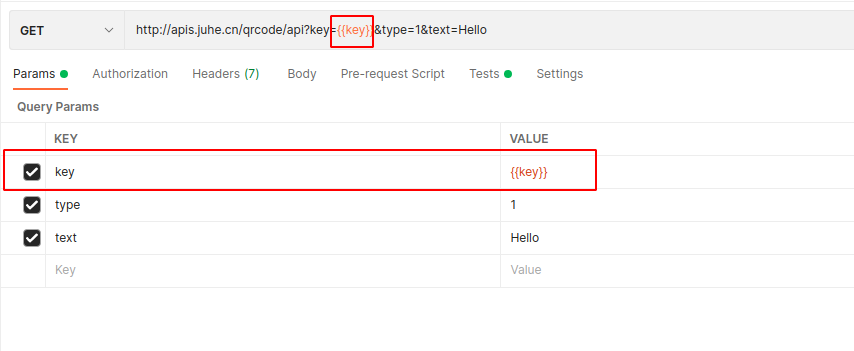
首先为整个接口测试创建一个数据集，然后分别对不同的测试接口创建请求

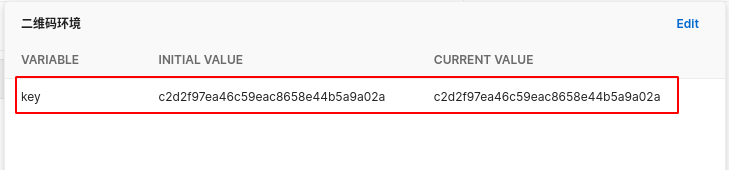


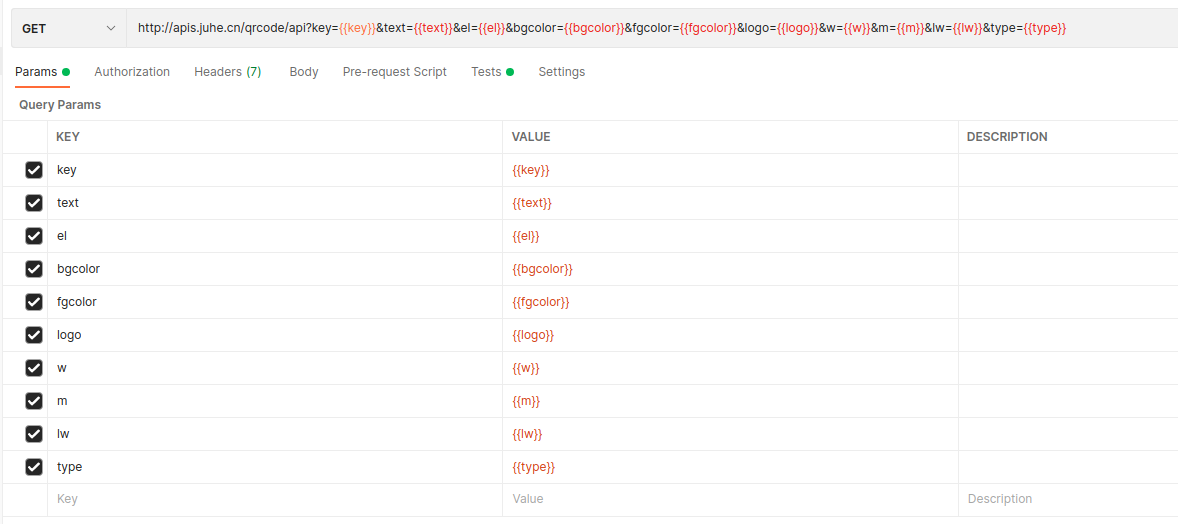
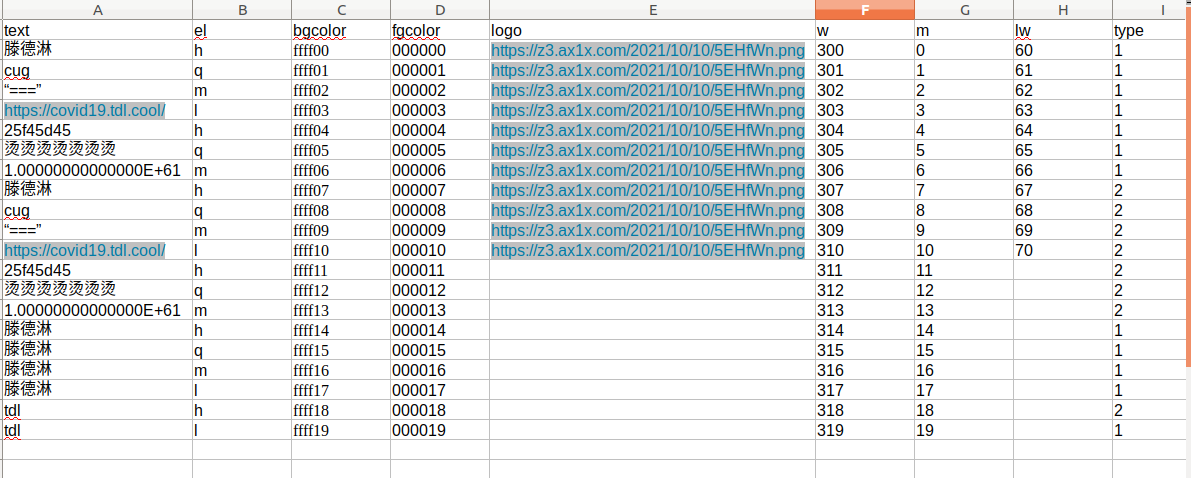
准备测试数据写在csv文件里面，然后以数据为驱动进行测试，可以一次进行多个测试用例的运行。

##### 2.2.3.2 编写测试用例与配置测试局部环境

编写测试用例需要按照接口提供的参数进行编写，一共10个参数，其中1个key参数是必须的，其他不是必须的。而且key值是唯一的，可以将key值设置为一个环境变量。



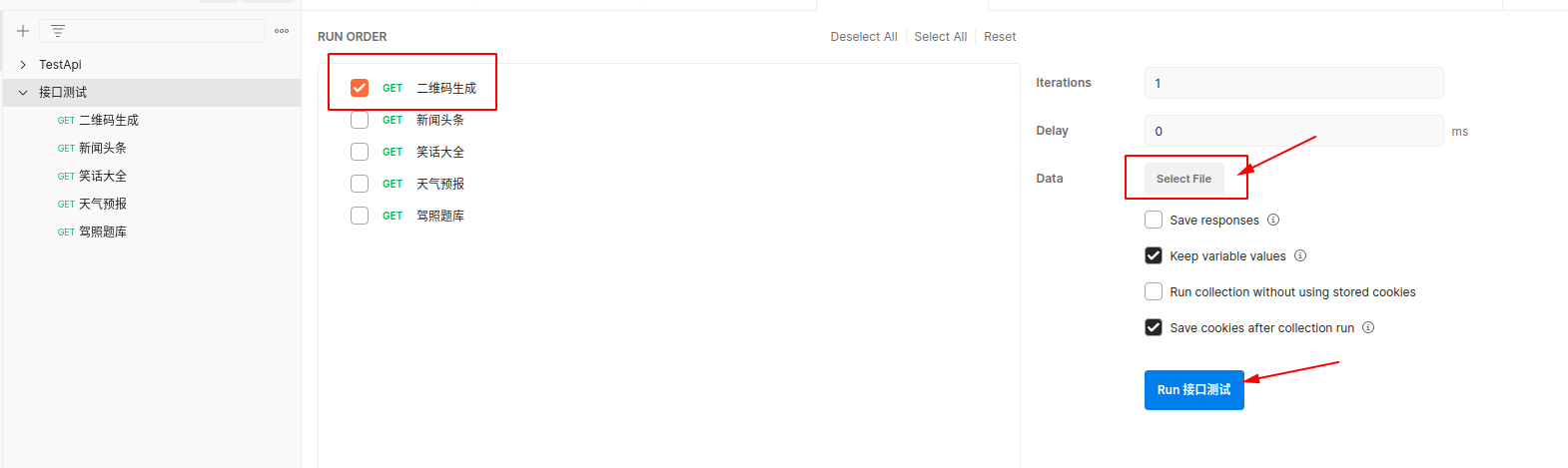
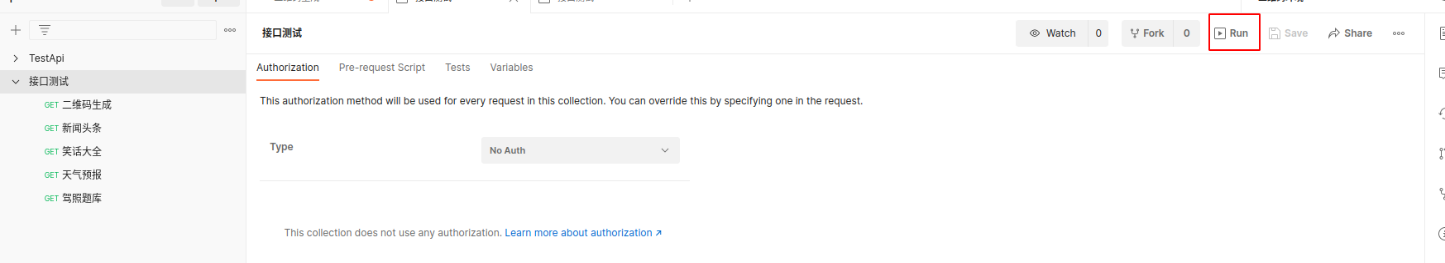
编写csv数据驱动文件类似于测试用例，是整个测试的核心，所以需要花多一点时间。

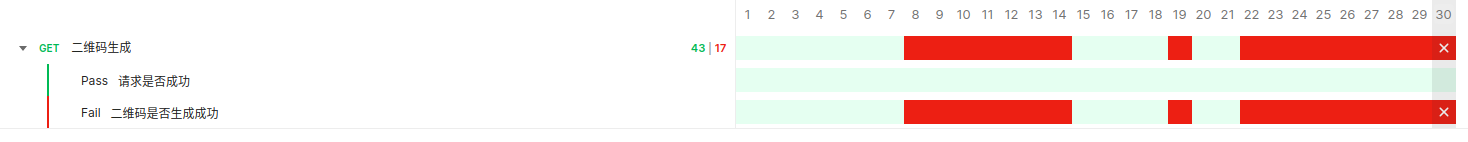
**表2-3 二维码生成接口测试用例表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 用例内容 | 前提条件 | 参数描述(按照参数顺序） | 预期结果 |
| 1 | 测试内容为中文；  并且返回类型为base64编码格式；  其余可不填信息进行依次递增； | 网络正常 | 滕德淋, h, ffff00, 0, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 300, 0, 60, 1 | 测试成功 |
| 2 | 测试内容为英文；  并且返回类型为base64编码格式；  其余可不填信息进行依次递增； | 网络正常 | cug, q, ffff01, 1, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 301, 1, 61, 1 | 测试成功 |
| 3 | 测试内容为符号；  并且返回类型为base64编码格式；  其余可不填信息进行依次递增； | 网络正常 | “===”, m, ffff02, 2, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 302, 2, 62, 1 | 测试成功 |
| 4 | 测试内容为网址；  并且返回类型为base64编码格式；  其余可不填信息进行依次递增； | 网络正常 | https://covid19.tdl.cool/, l, ffff03, 3, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 303, 3, 63, 1 | 测试成功 |
| 5 | 测试内容为数字；  并且返回类型为base64编码格式；  其余可不填信息进行依次递增； | 网络正常 | 25f45d45, h, ffff04, 4, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 304, 4, 64, 1 | 测试成功 |
| 6 | 测试内容为大数字；  并且返回类型为base64编码格式；  其余可不填信息进行依次递增； | 网络正常 | 1E+061, m, ffff06, 6, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 306, 6, 66, 1 | 测试成功 |
| 7 | 测试内容为中文；  并且返回类型为图片格式；  其余可不填信息进行依次递增； | 网络正常 | 滕德淋, h, ffff07, 7, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 307, 7, 67, 2 | 测试成功 |
| 8 | 测试内容为英文；  并且返回类型为图片格式；  其余可不填信息进行依次递增； | 网络正常 | cug, q, ffff08, 8, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 308, 8, 68, 2 | 测试成功 |
| 9 | 测试内容为符号；  并且返回类型为图片格式；  其余可不填信息进行依次递增； | 网络正常 | “===”, m, ffff09, 9, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 309, 9, 69, 2 | 测试成功 |
| 10 | 测试内容为网址；  并且返回类型为图片格式；  其余可不填信息进行依次递增； | 网络正常 | https://covid19.tdl.cool/, l, ffff10, 10, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 310, 10, 70, 2 | 测试成功 |
| 11 | 测试内容为数字；  并且返回类型为图片格式；  其余可不填信息进行依次递增； | 网络正常 | 25f45d45, h, ffff11, 11, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 311, 11,  , 2 | 测试成功 |
| 12 | 测试内容为大数字；  并且返回类型为图片格式；  其余可不填信息进行依次递增； | 网络正常 | 1E+061, m, ffff13, 13, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 313, 13,  , 2 | 测试成功 |
| 13 | 测试内容为敏感词汇；  并且返回类型为base64格式；  其余可不填信息进行依次递增； | 网络正常 | 烫烫烫烫烫烫烫, q, ffff05, 5, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 305, 5, 65, 1 | 测试成功 |
| 14 | 测试内容为敏感词汇；  并且返回类型为图片格式；  其余可不填信息进行依次递增； | 网络正常 | 烫烫烫烫烫烫烫, q, ffff12, 12, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 312, 12,  , 2 | 测试成功 |
| 15 | 测试内容为中文；  并且返回类型base64格式；  其余可不填信息进行依次递增；  Logo信息不填；  logo大小不填； | 网络正常 | 滕德淋, h, ffff14, 14,  , 314, 14,  , 1 | 测试成功 |
| 16 | 测试内容为中文；  并且返回类型base64格式；  其余可不填信息进行依次递增；  Logo信息不填；  logo大小不填； | 网络正常 | 滕德淋, q, ffff15, 15,  , 315, 15,  , 1 | 测试成功 |
| 17 | 测试内容为中文；  并且返回类型base64格式；  其余可不填信息进行依次递增；  Logo信息不填；  logo大小不填； | 网络正常 | 滕德淋, m, ffff16, 16,  , 316, 16,  , 1 | 测试成功 |
| 18 | 测试内容为中文；  并且返回类型base64格式；  其余可不填信息进行依次递增；  Logo信息不填；  logo大小不填； | 网络正常 | 滕德淋, l, ffff17, 17,  , 317, 17,  , 1 | 测试成功 |
| 19 | 测试内容为中文；  并且返回类型base64格式；  其余可不填信息进行依次递增；  Logo信息不填；  logo大小不填； | 网络正常 | 滕德淋, h, ffff18, 18,  , 318, 18,  , 1 | 测试成功 |
| 20 | 测试内容为中文；  并且返回类型base64格式；  其余可不填信息进行依次递增；  Logo信息不填；  logo大小不填； | 网络正常 | 滕德淋, h, ffff19, 19,  , 319, 19,  , 1 | 测试成功 |
| 21 | 测试内容为英文；  并且返回类型不属于规定格式；  其余可不填信息进行依次递增； | 网络正常 | tdl, h, ffff20, 20, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 320, 20,  , 4 | 测试失败 |
| 22 | 测试内容为英文；  并且返回类型不属于规定格式；  其余可不填信息进行依次递增； | 网络正常 | tdl, h, ffff21, 21, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 321, 21,  , 5 | 测试失败 |
| 23 | 测试内容为英文；  并且返回类型不属于规定格式；  其余可不填信息进行依次递增； | 网络正常 | tdl, h, ffff22, 22, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 322, 22,  , 6 | 测试失败 |
| 24 | 测试内容为英文；  并且返回类型不属于规定格式；  其余可不填信息进行依次递增； | 网络正常 | tdl, h, ffff23, 23, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 323, 23,  , 7 | 测试失败 |
| 25 | 测试内容为英文；  并且返回类型不属于规定格式；  其余可不填信息进行依次递增； | 网络正常 | tdl, h, ffff24, 24, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 324, 24,  , 8 | 测试失败 |
| 26 | 测试内容为英文；  并且返回类型为base64格式；  其余可不填信息进行依次递增；  纠错等级不属于规定等级； | 网络正常 | tdl, t, ffff25, 25, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 325, 25,  , 1 | 测试失败 |
| 27 | 测试内容为英文；  并且返回类型为base64格式；  其余可不填信息进行依次递增；  纠错等级不属于规定等级； | 网络正常 | tdl, t, ffff26, 26, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 326, 26,  , 1 | 测试失败 |
| 28 | 测试内容为英文；  并且返回类型为base64格式；  其余可不填信息进行依次递增；  纠错等级不属于规定等级； | 网络正常 | tdl, t, ffff27, 27, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 327, 27,  , 1 | 测试失败 |
| 29 | 测试内容为英文；  并且返回类型为base64格式；  其余可不填信息进行依次递增；  纠错等级不属于规定等级； | 网络正常 | tdl, t, ffff28, 28, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 328, 28,  , 1 | 测试失败 |
| 30 | 测试内容为英文；  并且返回类型为base64格式；  其余可不填信息进行依次递增；  纠错等级不属于规定等级； | 网络正常 | tdl, t, ffff29, 29, https://z3.ax1x.com/2021/10/10/5EHfWn.png, 329, 29,  , 1 | 测试失败 |

##### 2.2.3.3 数据驱动测试

在postman中点击run，然后选择csv文件作为数据驱动，然后配置好先前已经写好的局部环境变量，点击测试就可以进行一次测试全部的测试用例。



测试完成，可以看到有的测试用例通过了，有的测试用例没有通过，关注那些没有通过的测试用例，看是否符合预期结果。

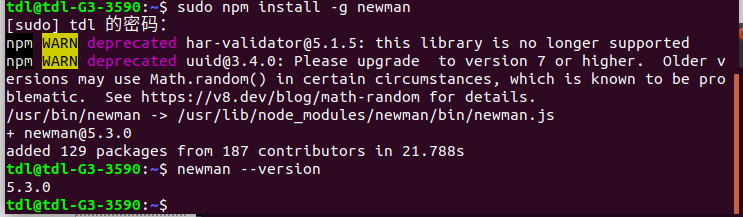
##### 2.2.3.4 Newman命令行测试与持续集成

* Newman 是 Postman 推出的一个 nodejs 库，直接来说就是 Postman 的json文件可以在命令行执行的插件。
* Newman 可以方便地运行和测试集合，并用之构造接口自动化测试和持续集成。

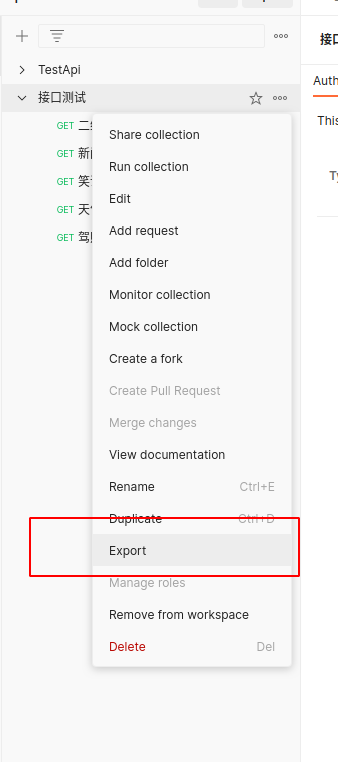
在安装Newman之前需要安装好nodejs环境同时还需要有npm环境；

在配置好之后，可以在终端中运行命令：sudo npm install -g newman 来安装Newman环境

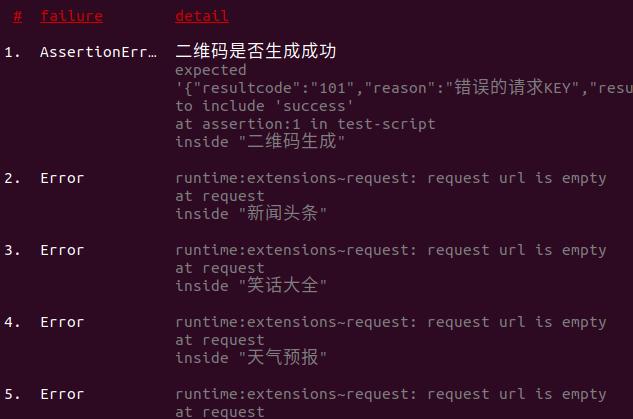


通过指令 newman –version 来检测是否安装好。

* **保存colltions**

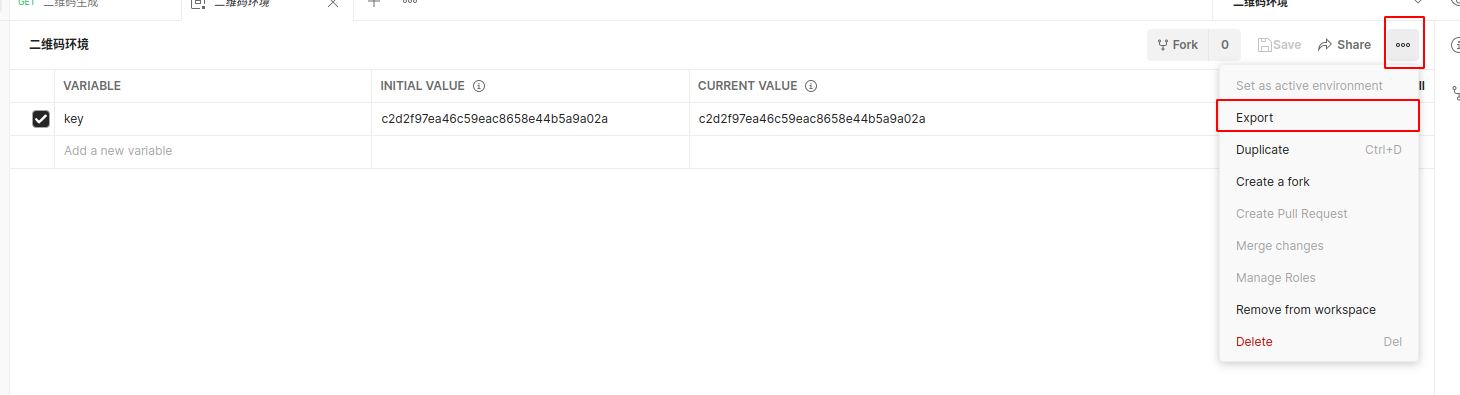


* **直接运行导出的json文件**

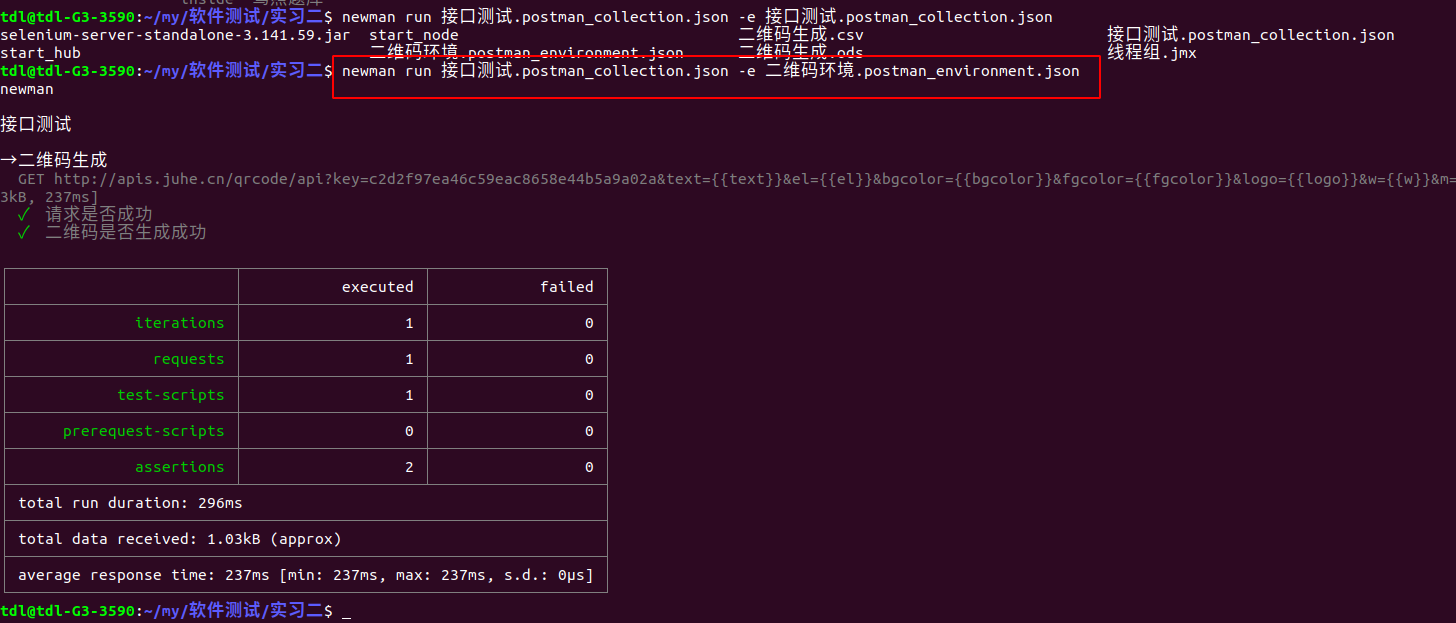
通过Newman这个工具库就可以脱离软件进行测试，就不再受到postman的束缚，可以直接在命令行里面进行测试，这是非常符合我们程序员的习惯的。

* **导出环境变量**

由于我们在终端运行测试用例的时候，我们也需要加载环境变量，所以我们可以先进行导出：

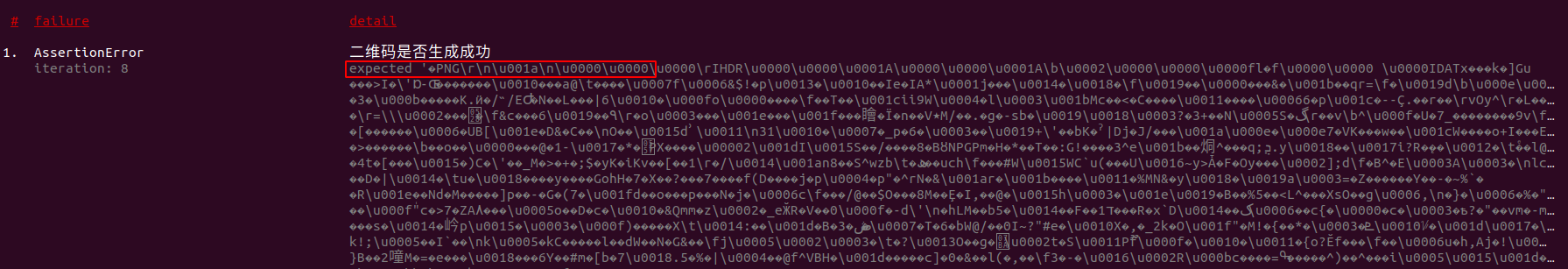


* **加载环境变量和数据驱动**

加载环境变量：-e 文件路径

加载数据驱动：-d 文件路径





### 2.3 测试对象2-新闻头条

#### 2.3.1 接口描述

由于第一个接口测试的时候已经将环境配置方面的东西写过了，所以接下来的几个接口就写的比较简略一些了，主要还是把时间花在编写测试用例上面。

**表2-4 接口信息描述**

|  |  |
| --- | --- |
| 接口名称 | 新闻头条 |
| 接口类型 | get/post |
| 接口地址 | <http://v.juhe.cn/toutiao/index> |
| 接口功能 | 最新新闻头条，各类国内、国际、体育、娱乐、科技等资讯，更新周期5-30分钟 |

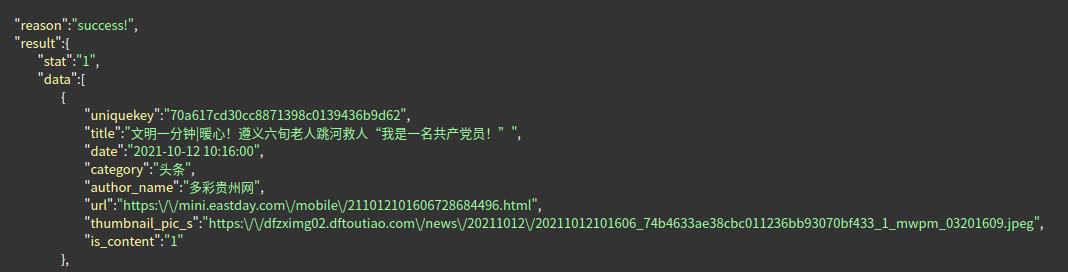
**表2-5 接口参数信息描述**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 必填 | 类型 | 说明 |
| 1 | key | 是 | string | 接口key, 在个人中心->我的数据,接口名称上方查看 |
| 2 | type | 否 | string | 支持类型 top(推荐,默认) guonei(国内) guoji(国际) yule(娱乐) tiyu(体育) junshi(军事) keji(科技) caijing(财经) shishang(时尚) youxi(游戏) qiche(汽车) jiankang(健康) |
| 3 | page | 否 | int | 当前页数, 默认1, 最大50 |
| 4 | page\_size | 否 | int | 每页返回条数, 默认30 , 最大30 |
| 5 | is\_filter | 否 | int | 是否只返回有内容详情的新闻, 1:是, 默认0 |

**表2-6 接口返回结果信息描述**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 说明 |
| 1 | error\_code | int | 返回码 |
| 2 | reason | string | 返回说明 |
| 3 | result | obj | 返回结果集 |
| 4 | data | array | 新闻列表, 无数据时为null |
| 5 | uniquekey | string | 新闻ID |
| 6 | title | string | 新闻标题 |
| 7 | date | string | 新闻时间 |
| 8 | category | string | 新闻分类 |
| 9 | author\_name | string | 新闻来源 |
| 10 | url | string | 新闻访问链接 |
| 11 | thumbnail\_pic\_s | string | 新闻图片链接 |
| 12 | is\_content | string | 是否有新闻内容,1表示有 可以通过查询新闻详细内容小接口获取新闻内容 |

**返回结果：**



#### 2.3.2 测试断言



#### 2.3.3 测试过程

**表2-7 新闻头条接口测试用例表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 用例内容 | 前提条件 | 参数描述 | 预期结果 |
| 1 | 国内新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | guonei, 1, 1, 0 | 测试成功 |
| 2 | 国际新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | guoji, 2, 2, 0 | 测试成功 |
| 3 | 娱乐新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | yule, 3, 3, 0 | 测试成功 |
| 4 | 体育新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | tiyu, 4, 4, 0 | 测试成功 |
| 5 | 军事新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | junshi, 5, 5, 0 | 测试成功 |
| 6 | 科技新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | keji, 6, 6, 0 | 测试成功 |
| 7 | 财经新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | caijing, 7, 7, 0 | 测试成功 |
| 8 | 时尚新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | shishang, 8, 8, 0 | 测试成功 |
| 9 | 游戏新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | youxi, 9, 9, 0 | 测试成功 |
| 10 | 汽车新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | qiche, 10, 10, 0 | 测试成功 |
| 11 | 健康新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | jiankang, 11, 11, 0 | 测试成功 |
| 12 | 推荐新闻；  页数、一页包含内容调试依次递增；  不过滤内容； | 网络正常  测试量未超标 | top, 12, 12, 0 | 测试成功 |
| 13 | 国内新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | guonei, 13, 13, 0 | 测试成功 |
| 14 | 国际新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | guoji, 14, 14, 0 | 测试成功 |
| 15 | 娱乐新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | yule, 15, 15, 0 | 测试成功 |
| 16 | 体育新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | tiyu, 16, 16, 0 | 测试成功 |
| 17 | 军事新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | junshi, 17, 17, 0 | 测试成功 |
| 18 | 科技新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | keji, 18, 18, 0 | 测试成功 |
| 19 | 财经新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | caijing, 19, 19, 0 | 测试成功 |
| 20 | 时尚新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | shishang, 20, 20, 0 | 测试成功 |
| 21 | 游戏新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | youxi, 21, 21, 1 | 测试成功 |
| 22 | 汽车新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | qiche, 22, 22, 1 | 测试成功 |
| 23 | 健康新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | jiankang, 23, 23, 1 | 测试成功 |
| 24 | 推荐新闻；  页数、一页包含内容调试依次递增；  过滤内容详情； | 网络正常  测试量未超标 | top, 24, 24, 1 | 测试成功 |
| 25 | 推荐新闻；  页数大于最大50；  过滤内容详情； | 网络正常  测试量未超标 | top, 100, 30, 1 | 测试成功 |
| 26 | 推荐新闻；  页数大于最大50；  过滤内容详情； | 网络正常  测试量未超标 | top, 101, 30, 1 | 测试成功 |
| 27 | 推荐新闻；  页数大于最大50；  过滤内容详情； | 网络正常  测试量未超标 | top, 102, 30, 1 | 测试成功 |
| 28 | 推荐新闻；  一页大于最大30；  过滤内容详情； | 网络正常  测试量未超标 | top, 50, 50, 1 | 测试成功 |
| 29 | 推荐新闻；  一页大于最大30；  过滤内容详情； | 网络正常  测试量未超标 | top, 50, 51, 1 | 测试成功 |
| 30 | 推荐新闻；  一页大于最大30；  过滤内容详情； | 网络正常  测试量未超标 | top, 50, 52, 1 | 测试成功 |

### 2.4 测试对象3-笑话大全

#### 2.4.1 接口描述

**表2-8 接口信息描述**

|  |  |
| --- | --- |
| 接口名称 | 笑话大全 |
| 接口类型 | http get |
| 接口地址 | <http://v.juhe.cn/joke/content/list.php> |
| 接口功能 | 搜集网络幽默、搞笑、内涵段子，不间断更新 |

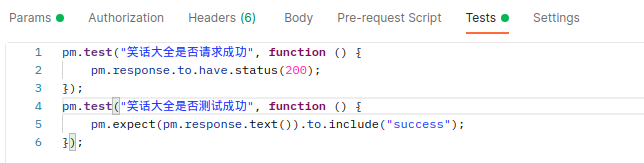
**表2-9 接口参数信息描述**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 必填 | 类型 | 说明 |
| 1 | sort | 是 | string | 类型，desc:指定时间之前发布的，asc:指定时间之后发布的 |
| 2 | page | 否 | int | 当前页数,默认1,最大20 |
| 3 | pagesize | 否 | int | 每次返回条数,默认1,最大20 |
| 4 | time | 是 | string | 时间戳（10位），如：1418816972 |
| 5 | key | 是 | string | 在个人中心->我的数据,接口名称上方查看 |

**表2-10 接口返回信息描述**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 说明 |
| 1 | error\_code | int | 返回码 |
| 2 | reason | string | 返回说明 |

#### 2.4.2 测试断言



#### 2.4.3 测试过程

**表2-11 笑话大全口测试用例表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 用例内容 | 前提条件 | 参数描述 | 预期结果 |
| 1 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 1, 1, 1418816972 | 测试成功 |
| 2 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 2, 2, 1418816973 | 测试成功 |
| 3 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 3, 3, 1418816974 | 测试成功 |
| 4 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 4, 4, 1418816975 | 测试成功 |
| 5 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 5, 5, 1418816976 | 测试成功 |
| 6 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 6, 6, 1418816977 | 测试成功 |
| 7 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 7, 7, 1418816978 | 测试成功 |
| 8 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 8, 8, 1418816979 | 测试成功 |
| 9 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 9, 9, 1418816980 | 测试成功 |
| 10 | 排序方式desc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | desc, 10, 10, 1418816981 | 测试成功 |
| 11 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 11, 11, 1418816982 | 测试成功 |
| 12 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 12, 12, 1418816983 | 测试成功 |
| 13 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 13, 13, 1418816984 | 测试成功 |
| 14 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 14, 14, 1418816985 | 测试成功 |
| 15 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 15, 15, 1418816986 | 测试成功 |
| 16 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 16, 16, 1418816987 | 测试成功 |
| 17 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 17, 17, 1418816988 | 测试成功 |
| 18 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 18, 18, 1418816989 | 测试成功 |
| 19 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常  测试量未超标 | asc, 19, 19, 1418816990 | 测试成功 |
| 20 | 排序方式asc；  页面、页面数量依次递增；  时间戳为10位； | 网络正常测试量未超标 | asc, 20, 20, 1418816991 | 测试成功 |
| 21 | 排序方式desc；  页面大于最大值20；  页面数量小于最大值；  时间戳为小于10位； | 网络正常  测试量未超标 | desc, 21, 20, 1 | 测试失败 |
| 22 | 排序方式desc；  页面大于最大值20；  页面数量小于最大值；  时间戳为小于10位； | 网络正常  测试量未超标 | desc, 22, 20, 11 | 测试失败 |
| 23 | 排序方式desc；  页面大于最大值20；  页面数量小于最大值；  时间戳为小于10位； | 网络正常  测试量未超标 | desc, 23, 20, 111 | 测试失败 |
| 24 | 排序方式desc；  页面大于最大值20；  页面数量小于最大值；  时间戳为小于10位； | 网络正常  测试量未超标 | desc, 24, 20, 1111 | 测试失败 |
| 25 | 排序方式desc；  页面大于最大值20；  页面数量小于最大值；  时间戳为小于10位； | 网络正常  测试量未超标 | desc, 25, 20, 11111 | 测试失败 |
| 26 | 排序方式desc；  页面小于最大  页面数量大于最大20；  时间戳为大于10位； | 网络正常  测试量未超标 | desc, 20, 50, 1111111111111 | 测试失败 |
| 27 | 排序方式desc；  页面小于最大  页面数量大于最大20；  时间戳为大于10位； | 网络正常  测试量未超标 | desc, 20, 51, 1111111111112 | 测试失败 |
| 28 | 排序方式desc；  页面小于最大  页面数量大于最大20；  时间戳为大于10位； | 网络正常  测试量未超标 | desc, 20, 52, 1111111111113 | 测试失败 |
| 29 | 排序方式desc；  页面小于最大  页面数量大于最大20；  时间戳为大于10位； | 网络正常  测试量未超标 | desc, 20, 53, 1111111111114 | 测试失败 |
| 30 | 排序方式desc；  页面小于最大  页面数量大于最大20；  时间戳为大于10位； | 网络正常  测试量未超标 | desc, 20, 54, 1111111111115 | 测试失败 |

### 2.5 测试对象4-天气预报

#### 2.5.1 接口描述

**表2-12 接口信息描述**

|  |  |
| --- | --- |
| 接口名称 | 天气预报 |
| 接口类型 | http get/post |
| 接口地址 | <http://apis.juhe.cn/simpleWeather/query> |
| 接口功能 | 查询天气情况：温度、湿度、AQI、天气、风向等 |

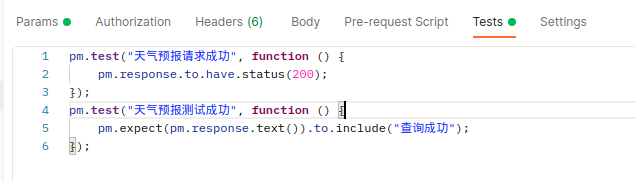
**表2-13 接口参数信息描述**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 必填 | 类型 | 说明 |
| 1 | city | 是 | string | 要查询的城市名称/id，城市名称如：温州、上海、北京，需要utf8 urlencode |
| 2 | key | 是 | string | 在个人中心->我的数据,接口名称上方查看 |

**表2-14 接口返回信息描述**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 说明 |
| 1 | error\_code | int | 返回码，0为查询成功 |
| 2 | reason | string | 返回说明 |
| 3 | result | string | 返回结果集 |
| 4 | realtime | - | 当前天气详情情况 |
| 5 | info | string | 天气情况，如：晴、多云 |
| 6 | wid | string | 天气标识id，可参考小接口2 |
| 7 | temperature | string | 温度，可能为空 |
| 8 | humidity | string | 湿度，可能为空 |
| 9 | direct | string | 风向，可能为空 |
| 10 | power | string | 风力，可能为空 |
| 11 | aqi | string | 空气质量指数，可能为空 |
| 12 | future | - | 近5天天气情况 |
| 13 | date | string | 日期 |
| 14 | temperature | string | 温度，最低温/最高温 |
| 15 | weather | string | 天气情况 |
| 16 | direct | string | 风向 |

#### 2.5.2 测试断言



#### 2.5.3 测试过程

**表2-15 天气预报接口测试用例表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 用例内容 | 前提条件 | 参数描述 | 预期结果 |
| 1 | 城市用城市id数字替代 | 网络正常测试量未超标 | 1 | 测试成功 |
| 2 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 2 | 测试成功 |
| 3 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 3 | 测试成功 |
| 4 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 4 | 测试成功 |
| 5 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 5 | 测试成功 |
| 6 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 6 | 测试成功 |
| 7 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 7 | 测试成功 |
| 8 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 8 | 测试成功 |
| 9 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 9 | 测试成功 |
| 10 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 10 | 测试成功 |
| 11 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 11 | 测试成功 |
| 12 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 12 | 测试成功 |
| 13 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 13 | 测试成功 |
| 14 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 14 | 测试成功 |
| 15 | 城市用城市id数字替代 | 网络正常  测试量未超标 | 15 | 测试成功 |
| 16 | 城市用城市名称替代 | 网络正常  测试量未超标 | 北京 | 测试成功 |
| 17 | 城市用城市名称替代 | 网络正常  测试量未超标 | 海淀 | 测试成功 |
| 18 | 城市用城市名称替代 | 网络正常  测试量未超标 | 朝阳 | 测试成功 |
| 19 | 城市用城市名称替代 | 网络正常  测试量未超标 | 顺义 | 测试成功 |
| 20 | 城市用城市名称替代 | 网络正常  测试量未超标 | 怀柔 | 测试成功 |
| 21 | 城市用城市名称替代 | 网络正常  测试量未超标 | 通州 | 测试成功 |
| 22 | 城市用城市名称替代 | 网络正常  测试量未超标 | 昌平 | 测试成功 |
| 23 | 城市用城市名称替代 | 网络正常  测试量未超标 | 延庆 | 测试成功 |
| 24 | 城市用城市名称替代 | 网络正常  测试量未超标 | 丰台 | 测试成功 |
| 25 | 城市用城市名称替代 | 网络正常  测试量未超标 | 石景山 | 测试成功 |
| 26 | 城市用城市名称替代 | 网络正常  测试量未超标 | 大兴 | 测试成功 |
| 27 | 城市用城市名称替代 | 网络正常  测试量未超标 | 房山 | 测试成功 |
| 28 | 城市用城市名称替代 | 网络正常  测试量未超标 | 密云 | 测试成功 |
| 29 | 城市用城市名称替代 | 网络正常  测试量未超标 | 门头沟 | 测试成功 |
| 30 | 城市用城市名称替代 | 网络正常  测试量未超标 | 武汉 | 测试成功 |

### 2.6 测试对象5-驾照题库

#### 2.6.1 接口描述

**表2-16 接口信息描述**

|  |  |
| --- | --- |
| 接口名称 | 驾照题库 |
| 接口类型 | get post |
| 接口地址 | <http://v.juhe.cn/jztk/query> |
| 接口功能 | 采用公安部最新驾校模拟考试题库；黑钻会员自动获取2021年最新题库； |

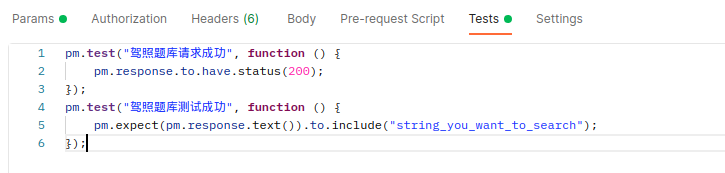
**表2-17 接口参数信息描述**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 必填 | 类型 | 说明 |
| 1 | key | 是 | string | 您申请的appKey |
| 2 | subject | 是 | int | 选择考试科目类型，1：科目1；4：科目4 |
| 3 | model | 是 | string | 驾照类型，可选择参数为：c1,c2,a1,a2,b1,b2；当subject=4时可省略 |
| 4 | testType | 否 | string | 测试类型，rand：随机测试（随机100个题目），order：顺序测试（所选科目全部题目） |

**表2-18 接口返回信息描述**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 说明 |
| 1 | error\_code | int | 返回状态码 |
| 2 | reason | string | 返回原因 |
| 3 | result | string | 题目内容 |

#### 2.6.2 测试断言



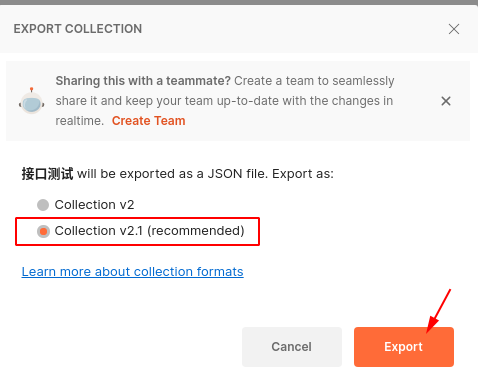
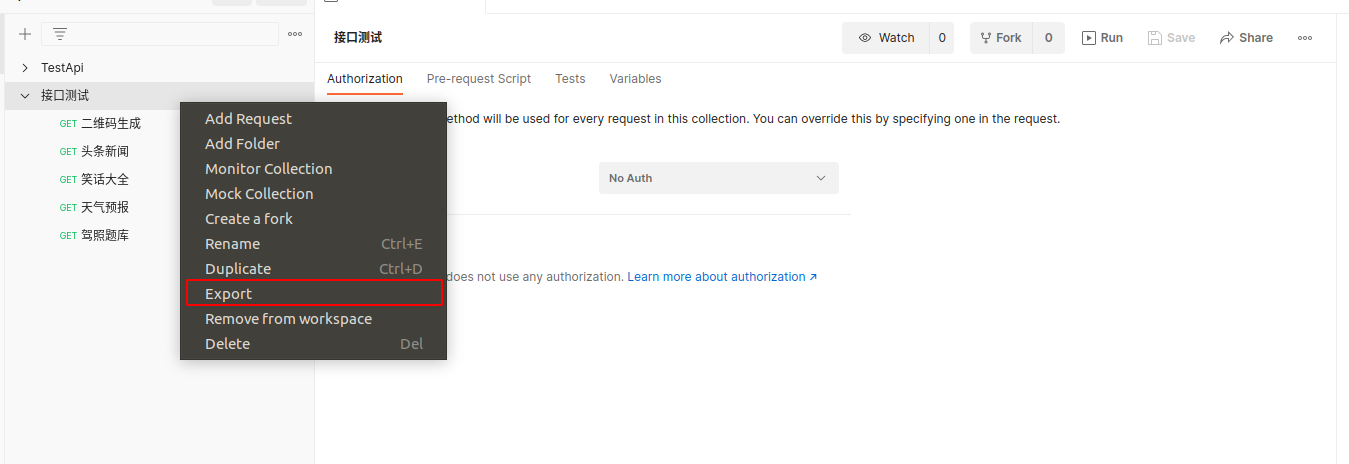
#### 2.6.3 测试过程

**表2-19 驾校题库接口测试用例表**

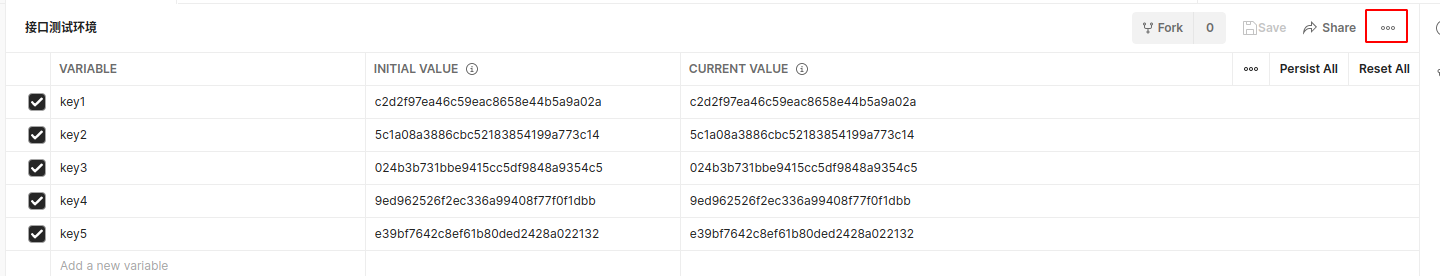
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 用例内容 | 前提条件 | 参数描述 | 预期结果 |
| 1 | 科目为科目一；  车型为c1;题目顺序随机； | 网络正常测试量未超标 | 1, c1, rand | 测试成功 |
| 2 | 科目为科目一；  车型为c2;题目顺序随机； | 网络正常  测试量未超标 | 1, c2, rand | 测试成功 |
| 3 | 科目为科目一；  车型为a1;题目顺序随机； | 网络正常  测试量未超标 | 1, a1, rand | 测试成功 |
| 4 | 科目为科目一；  车型为a2;题目顺序随机； | 网络正常  测试量未超标 | 1, a2, rand | 测试成功 |
| 5 | 科目为科目一；  车型为b1;题目顺序随机； | 网络正常  测试量未超标 | 1, b1, rand | 测试成功 |
| 6 | 科目为科目一；  车型为b2;题目顺序随机； | 网络正常  测试量未超标 | 1, b2, rand | 测试成功 |
| 7 | 科目为科目四；  车型为c1;题目顺序随机； | 网络正常  测试量未超标 | 4, c1, rand | 测试成功 |
| 8 | 科目为科目四；  车型为c2;题目顺序随机； | 网络正常  测试量未超标 | 4, c2, rand | 测试成功 |
| 9 | 科目为科目四；  车型为a1;题目顺序随机； | 网络正常  测试量未超标 | 4, a1, rand | 测试成功 |
| 10 | 科目为科目四；  车型为a2;题目顺序随机； | 网络正常  测试量未超标 | 4, a2, rand | 测试成功 |
| 11 | 科目为科目四；  车型为b1;题目顺序随机； | 网络正常  测试量未超标 | 4, b1, rand | 测试成功 |
| 12 | 科目为科目四；  车型为b2;题目顺序随机； | 网络正常  测试量未超标 | 4, b2, rand | 测试成功 |
| 13 | 科目为科目一；  车型为c1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, c1, order | 测试成功 |
| 14 | 科目为科目一；  车型为c2;题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, c2, order | 测试成功 |
| 15 | 科目为科目一；  车型为a1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, a1, order | 测试成功 |
| 16 | 科目为科目一；  车型为a2;题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, a2, order | 测试成功 |
| 17 | 科目为科目一；  车型为b1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, b1, order | 测试成功 |
| 18 | 科目为科目一；  车型为b2;题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, b2, order | 测试成功 |
| 19 | 科目为科目四；  车型为c1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 4, c1, order | 测试成功 |
| 20 | 科目为科目四；  车型为c2;题目顺序为按照顺序； | 网络正常  测试量未超标 | 4, c2, order | 测试成功 |
| 21 | 科目为科目四；  车型为a1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 4, a1, order | 测试成功 |
| 22 | 科目为科目四；  车型为a2;题目顺序为按照顺序； | 网络正常  测试量未超标 | 4, a2, order | 测试成功 |
| 23 | 科目为科目四；  车型为b1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 4, b1, order | 测试成功 |
| 24 | 科目为科目四；  车型为b2;题目顺序为按照顺序； | 网络正常  测试量未超标 | 4, b2, order | 测试成功 |
| 25 | 科目为不规范科目；  车型为c1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 2, c1, order | 测试失败 |
| 26 | 科目为不规范科目；  车型为c1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 3, c1, order | 测试失败 |
| 27 | 科目为不规范科目；  车型为c1;题目顺序为按照顺序； | 网络正常  测试量未超标 | 5, c1, order | 测试失败 |
| 28 | 科目为科目一；  车型不规范车型;  题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, c3, rand | 测试失败 |
| 29 | 科目为科目一；  车型不规范车型;  题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, a3, rand | 测试失败 |
| 30 | 科目为科目一；  车型不规范车型;  题目顺序为按照顺序； | 网络正常  测试量未超标 | 1, b3, rand | 测试失败 |

### 2.7 五个接口整体测试结果

#### 2.7.1 导出测试集

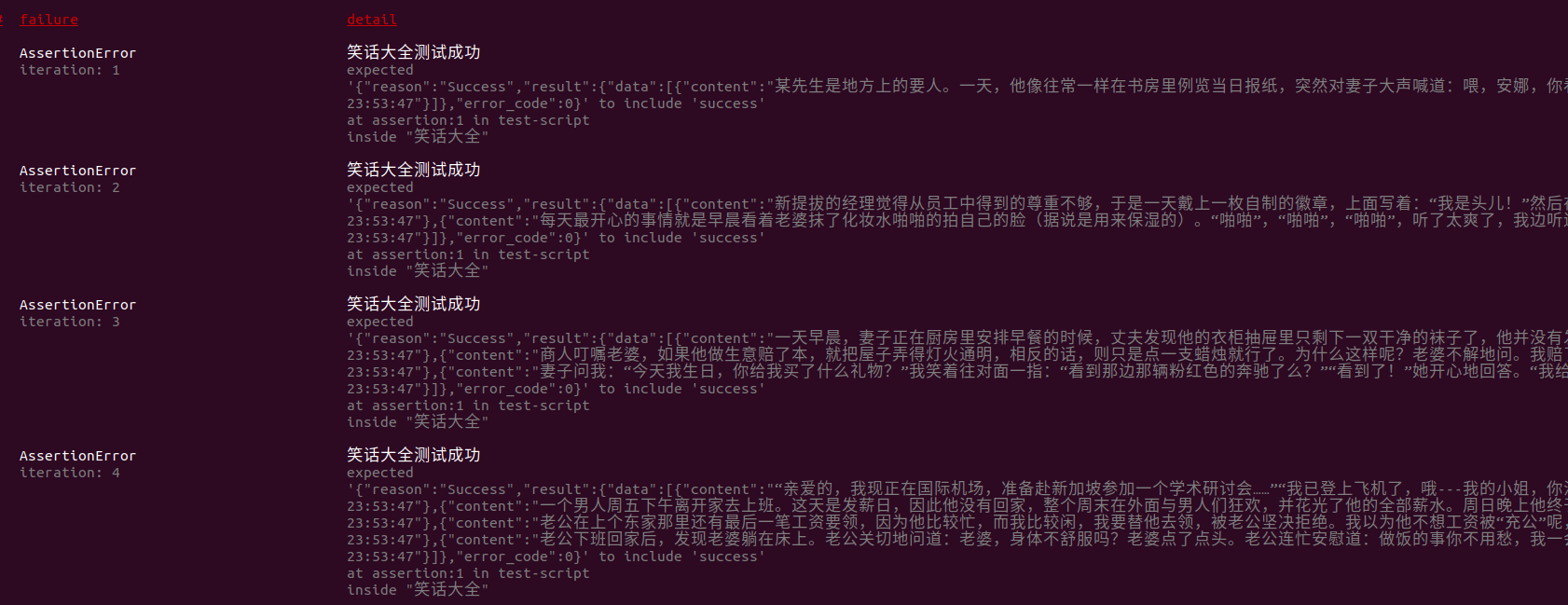


#### 2.7.2 导出环境变量



#### 2.7.3 Newman控制台运行

运行命令： **newman run 接口测试.postman\_collection.json -e 接口测试.postman\_collection.json -d 接口测试数据驱动.csv**



完成了5个接口的一次性测试，如果需要定时测试也有很多中测试方法，可以编写一个脚本，然后设置电脑的触发器事件，定时执行脚本。也可以自己搭建自动化测试框架，然后设置定时功能，具体如何实现就不再描述。好像第四次实现要自己搭建一个自动化测试框架。

## 3. 性能测试

### 3.0 环境配置

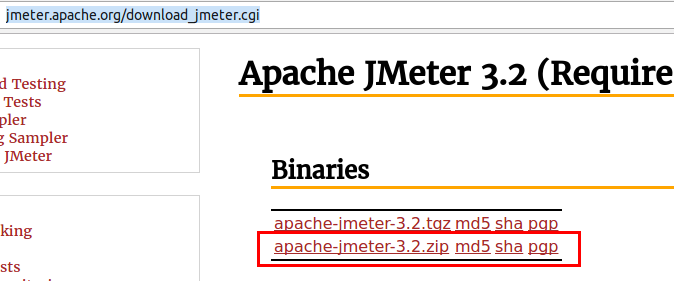
#### 3.0.1 Jmeter环境配置

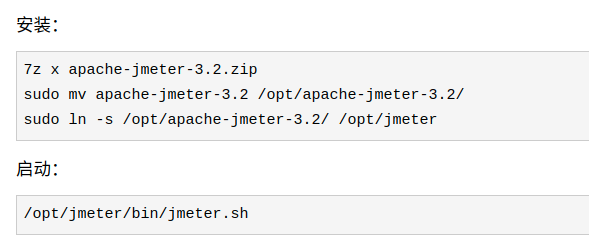
JMeter是Java的测试工具，由Apache开发。

同样，JMeter是跨平台的。

下载之前需要安装好Java的jdk；

下载： <http://jmeter.apache.org/download_jmeter.cgi>

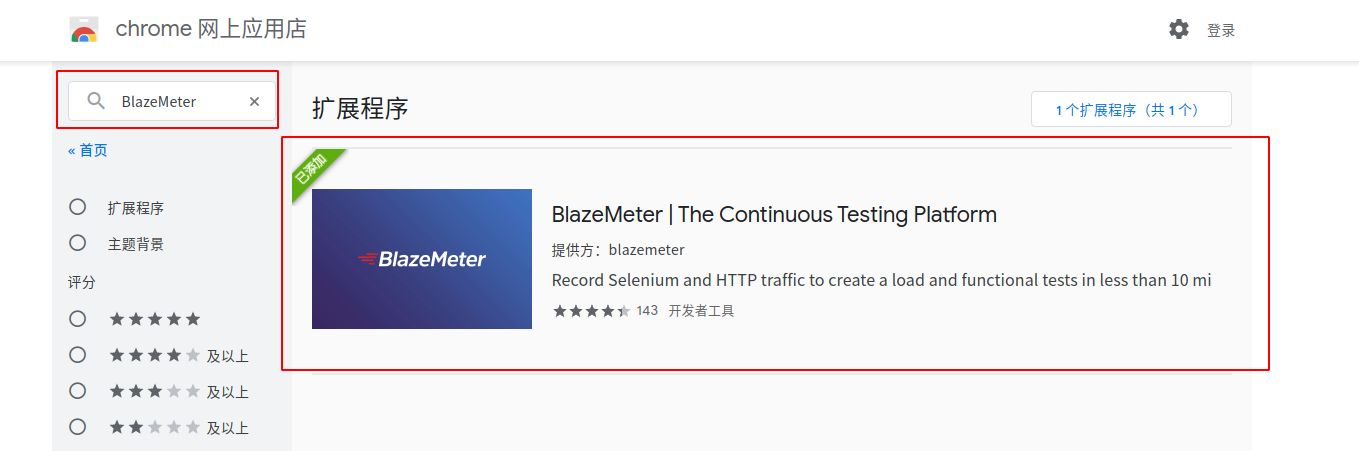


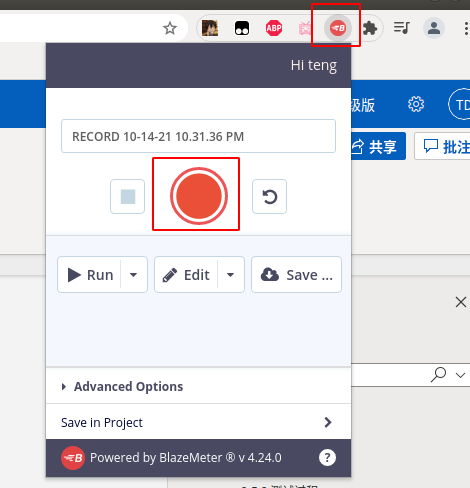


软件界面截图：

#### 3.0.2 脚本录制插件安装

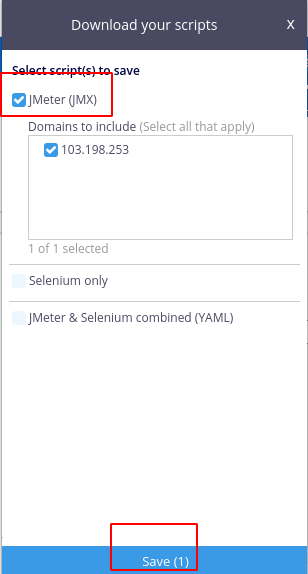
使用 Powered by BlazeMeter ® v 4.24.0插件进行录制脚本，这个脚本是适配在Chrome浏览器上面，如果要安装可能需要科学上网（翻墙）。

具体的使用方法就是，安装好之后，打开需要测试的页面，然后点击插件：

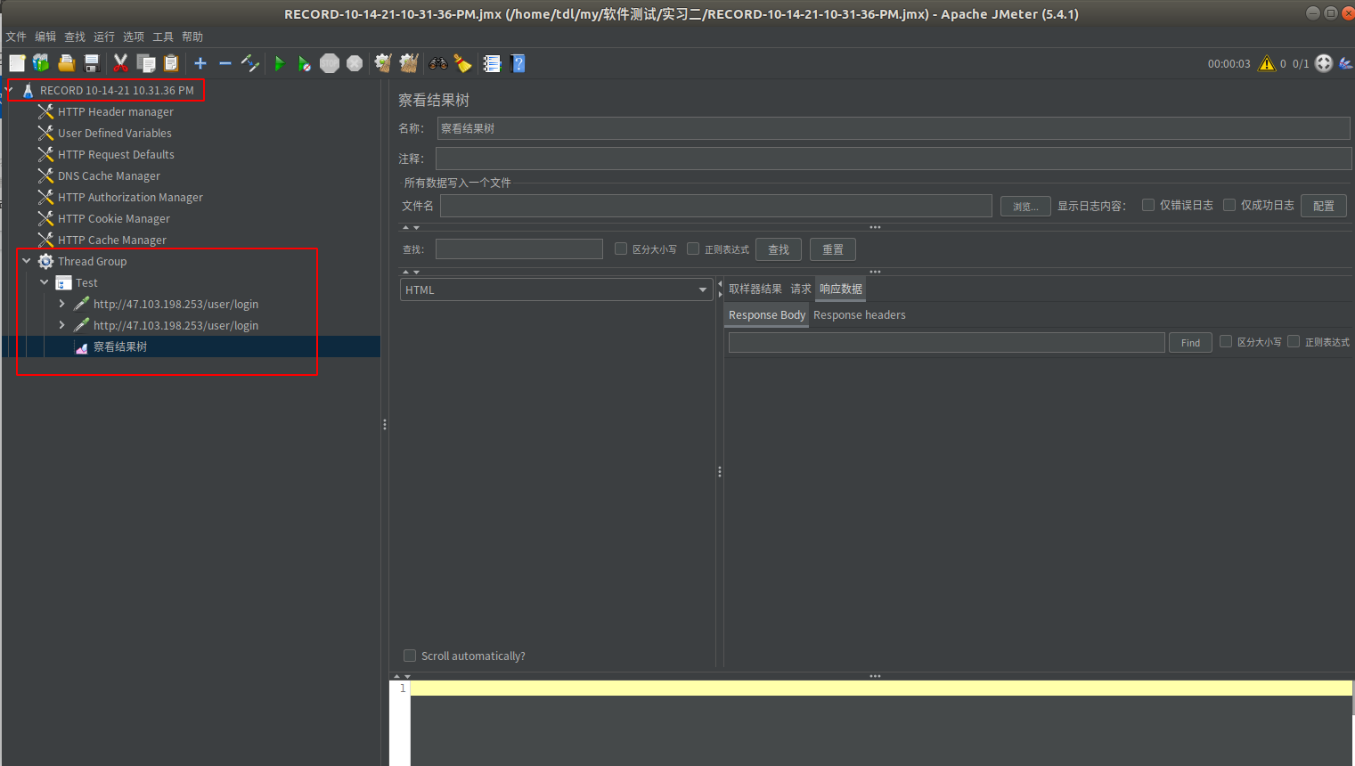


然后进行操作，该插件会自动录制脚本。完成之后结束录制。

然后进行保存和导出为jmx脚本文件：



然后用jmeter打开该脚本：



#### 3.0.3 自动化工具功能使用方法

##### 3.0.3.1 录制与回放

前面已经介绍过如何使用 BlazeMeter 录制脚本了，回放脚本就是用jmeter打开保存好的jmx脚本，然后可以看到里面有刚才录制好的脚本内容，可以经过自己的分析进行增加和删除。

##### 3.0.3.2 测试脚本语言的编写

Jmeter的Sampler用来模拟向被测试系统发起请求的，Jmeter会记录取样的结果，也就是TPS、响应时间等性能指标。

**大致流程：**

1、导入jar包：导入Jmeter相关jar包

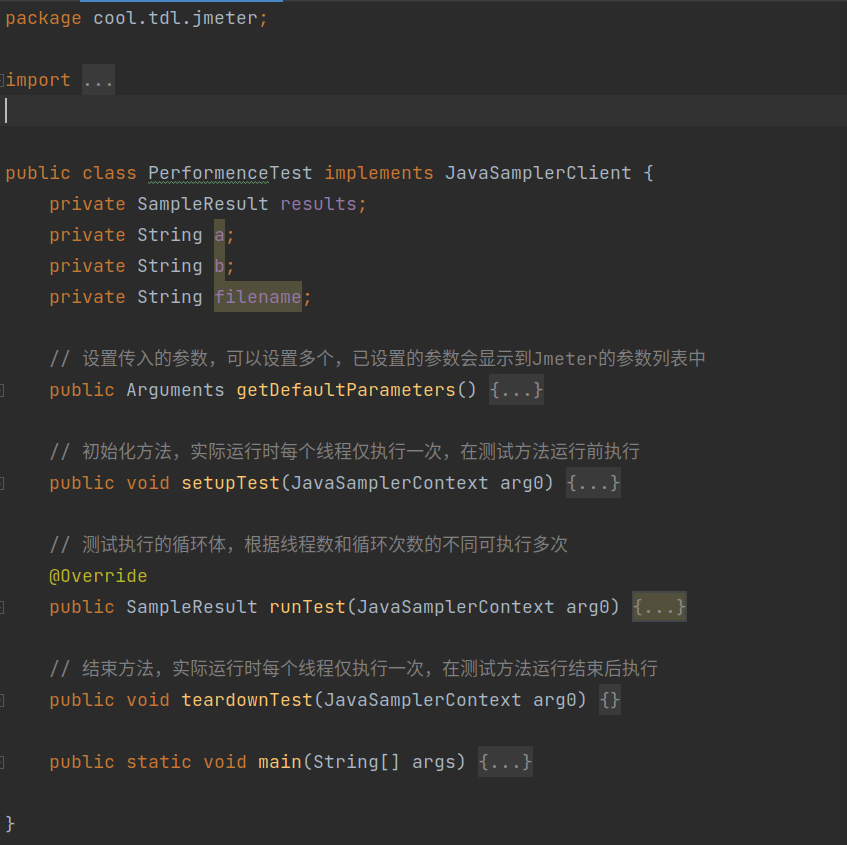
2、编写脚本：使用Eclipse或者IntelliJ IDEA，进行Java测试脚本的编写

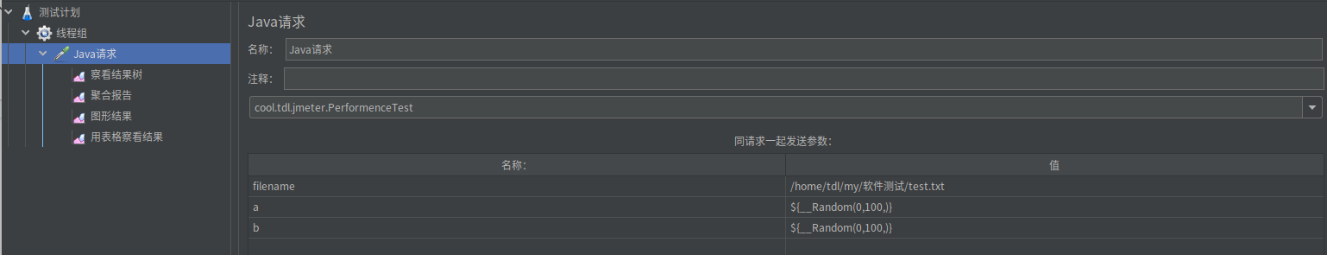
3、导出为jar包：使用Eclipse或者IntelliJ IDEA将测试脚本编译打包为jar包导出

4、配置Jmeter：使用自编译的jar包，以及一些相关配置进行性能测试

5、选择自编写jar包：自编译jar包

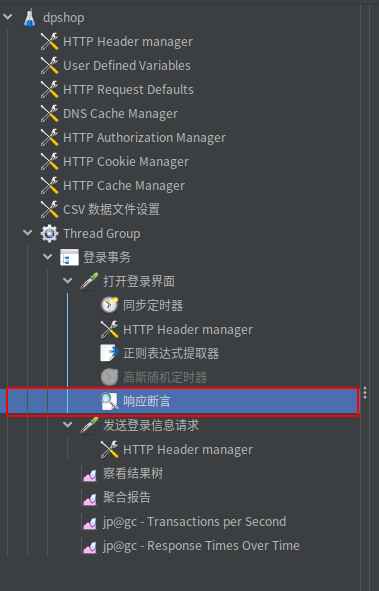
6、Jmeter运行：开始运行

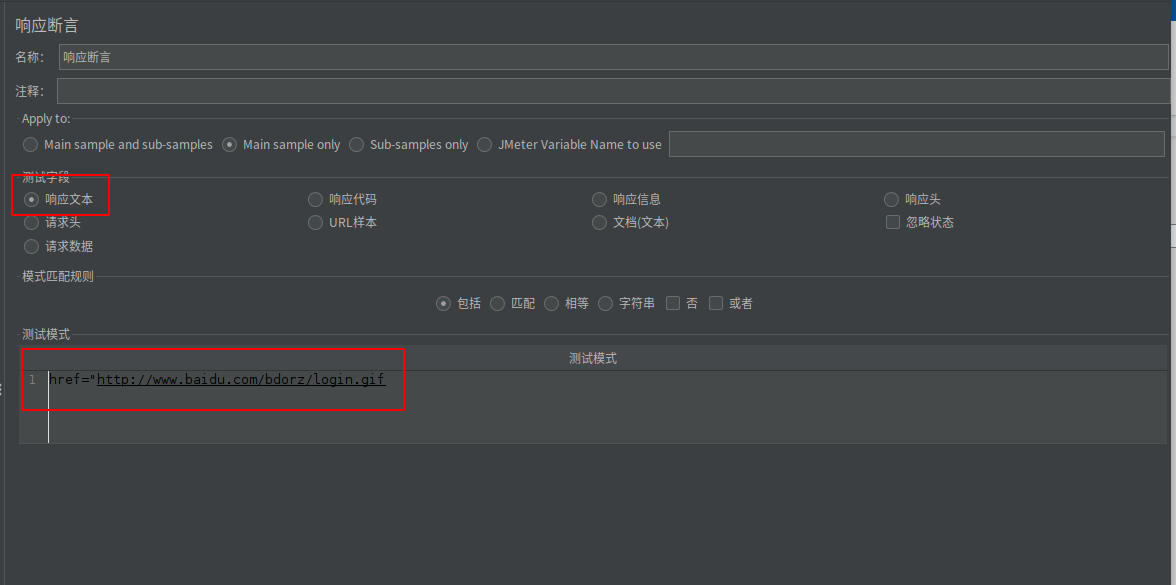




##### 3.0.3.3 创建检查点

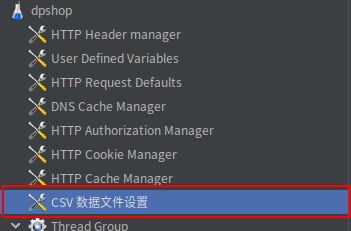
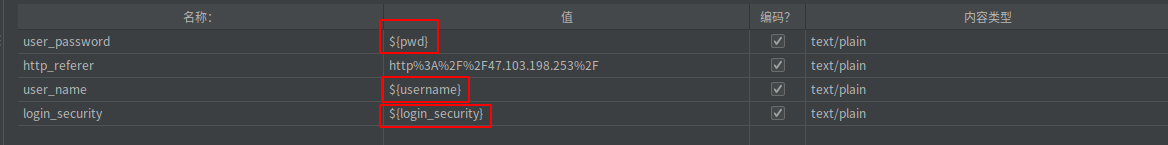
类似于断言功能，可以在jmeter工具中进行设置：

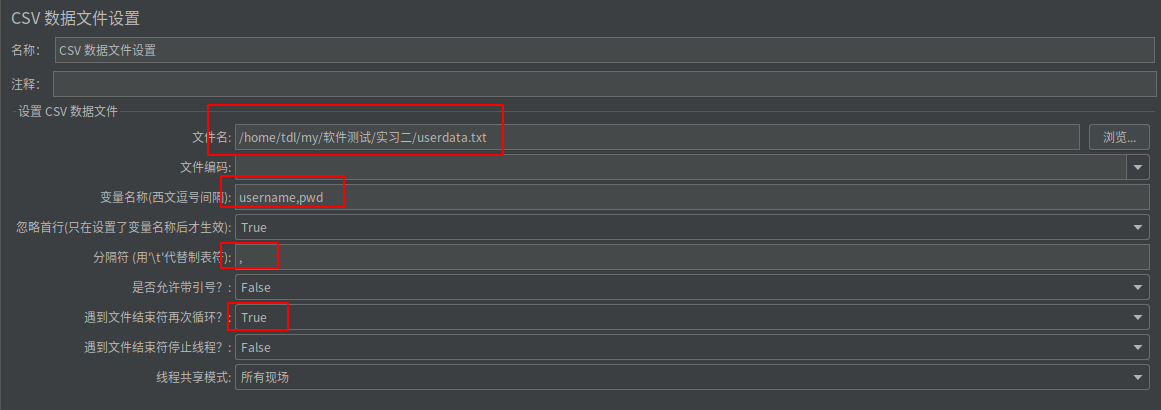


检测点就是用来判断响应结果是否和我们所预期的一样，有很多种判断的方式。

##### 3.0.3.4 参数化测试

类似与前面接口测试的数据驱动方式，就是将要测试的参数值写在一个单一文档中，然后在运行的时候进行导入。这样就可以一次性导入多个数据。

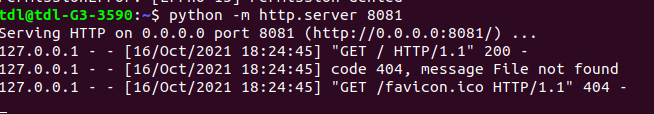


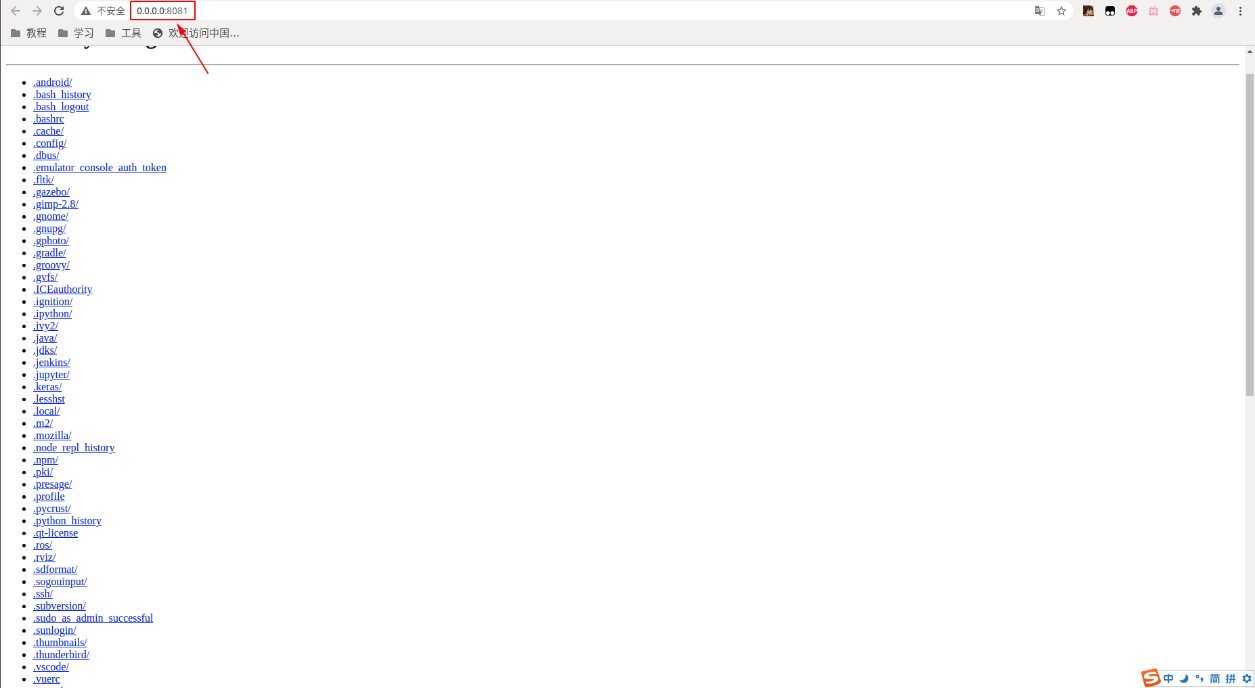


参数化很有必要，这不仅对多参数测试有非常大的作用，它还是正则表达式提取器和IP欺骗等等功能的必要功能。

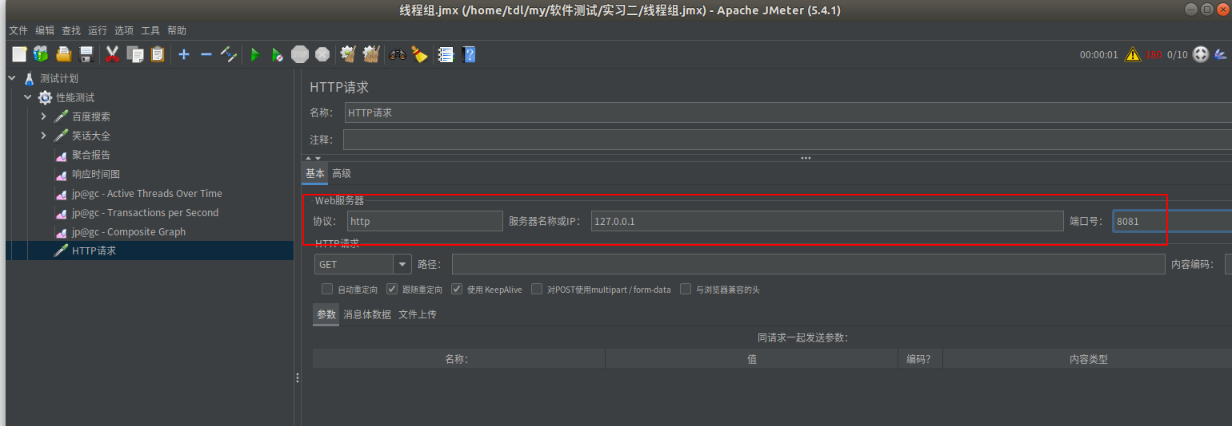
##### 3.0.3.5 虚拟对象

先通过python启动一个小的服务器来做测试

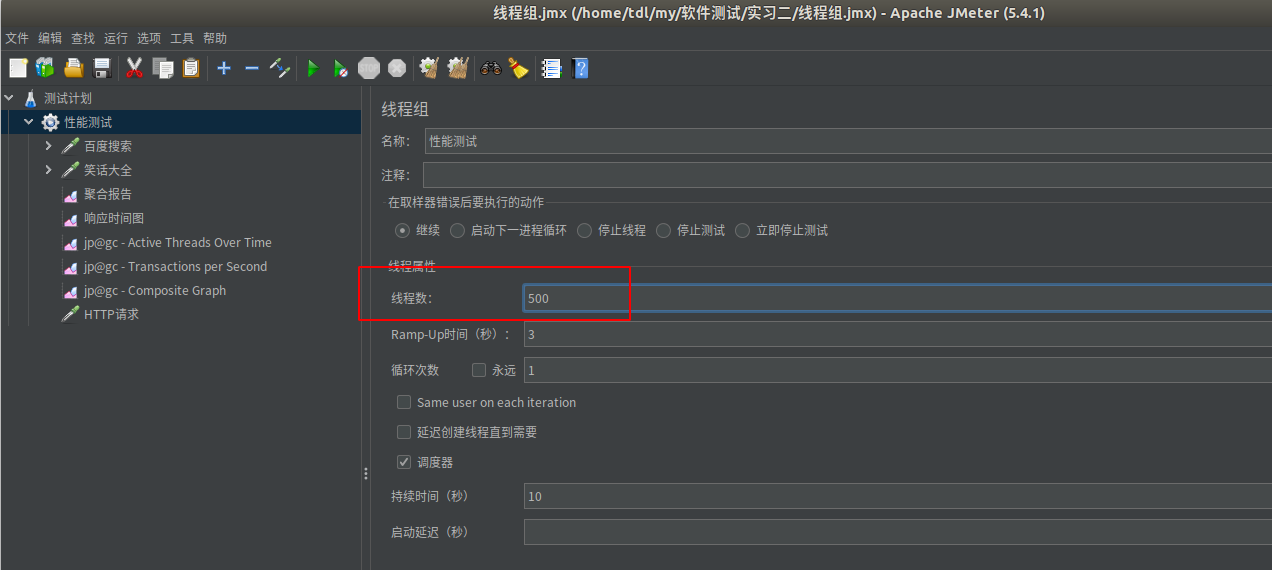
打开之后就是本机的目录：



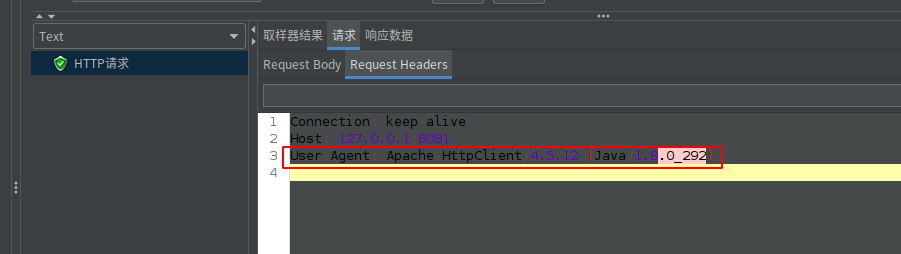
发送请求：



设置并发数：



模拟对象查看：

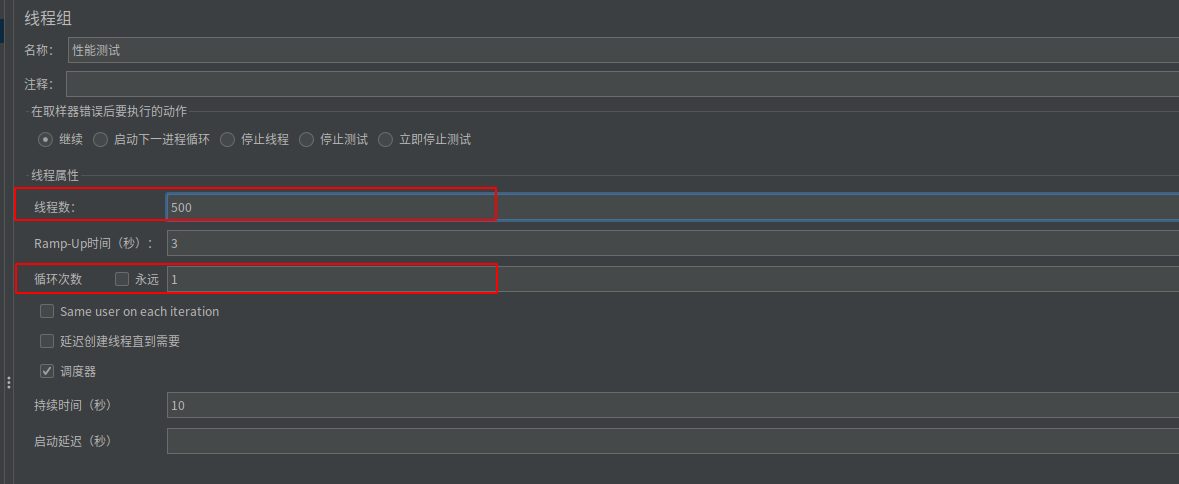


##### 3.0.3.6 对象侦探

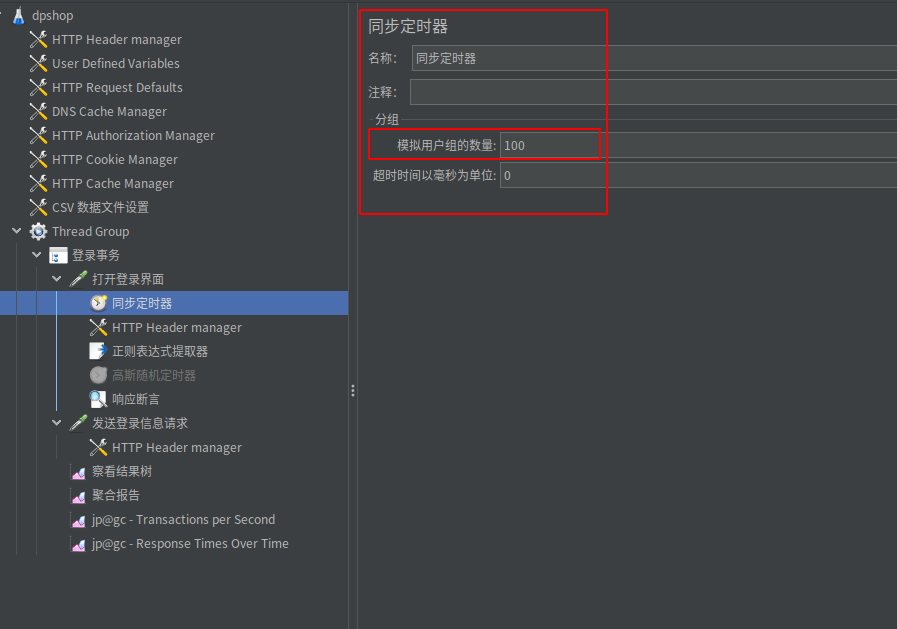
这个不知道是什么意思，大致理解为对测试对象信息的获取吧，可以从返回的response中的header中获得测试对象的信息。

##### 3.0.3.7 创建虚拟用户

创建虚拟用户和创建虚拟对象刚好相反，我们用多个线程来进行模拟多个用户的情况：



但其实单一的使用多线程还不能真实的模拟，需要对应的创建一个用户同步器来进行与之契合。



##### 3.0.3.8 容量测试:数据的上传/下载

要进行这个功能的使用的前提是需要测试接口的参数包含文件类型，这样在高并发的情况下，大量的获取服务器资源，能够测出服务器所能产生的最大网络带宽也就是容量。



下载的时候需要在beanshell中进行IO流的操作，进行保存文件。

##### 3.0.3.9 IP欺骗功能

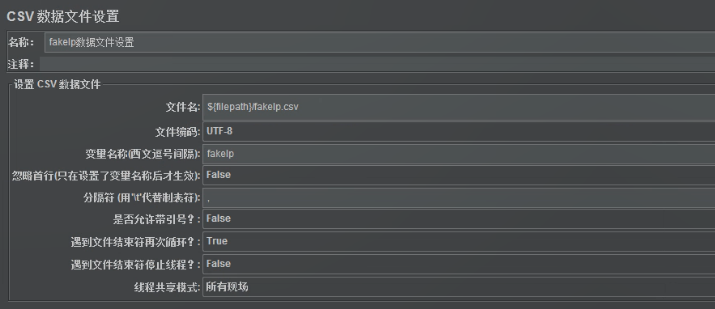
网上的教程全是基于Windows的，自己摸索Linux实现了，亲测可用。

首先给IP和需要伪造的IP处于同一网段的网卡起别名：

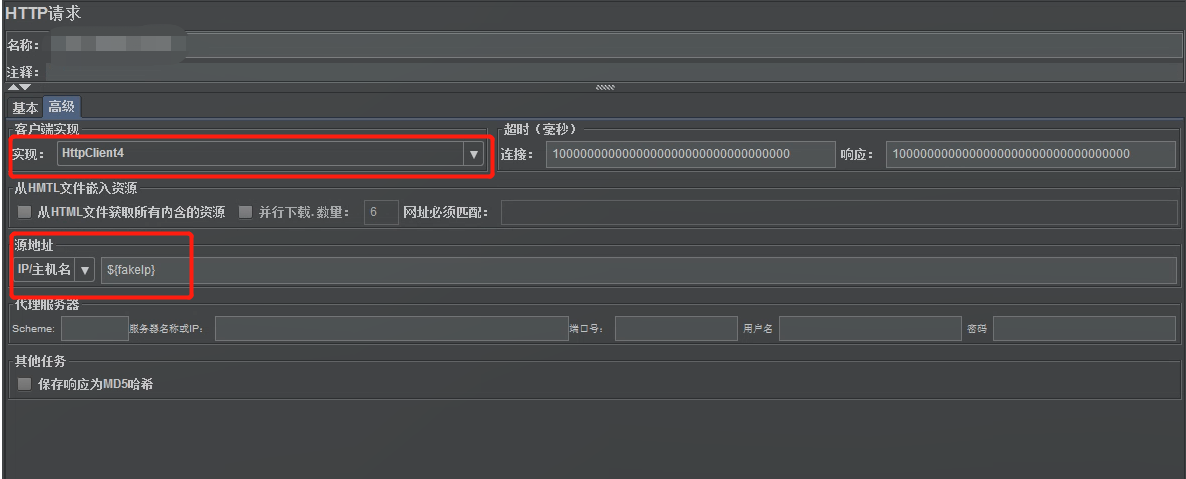
sudo ifconfig [net device]:0-255 [fake ip address] netmask [fake netmask] up

sudo route add -host [fake ip address] dev [net device]:0-255

然后修改JMeter脚本，配置一个csv读取器:

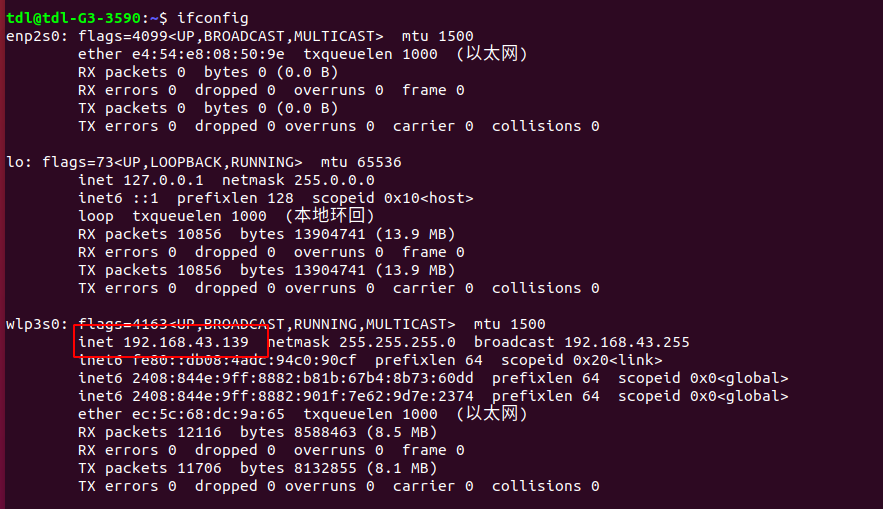


需要使用IP欺骗的接口按照图片配置，注意红框中内容:

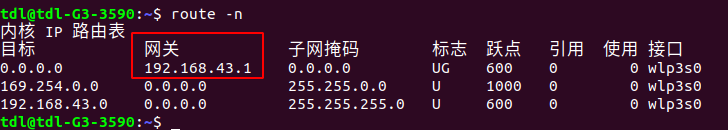


**演示：**

**本机原IP：**



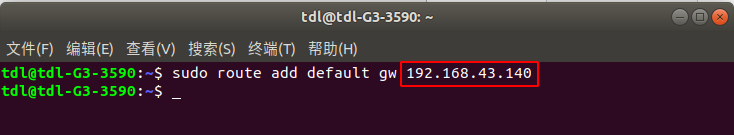
本机原网关：



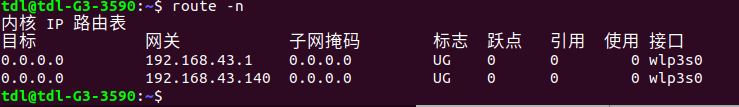
设置临时虚假IP并查看是否成功：

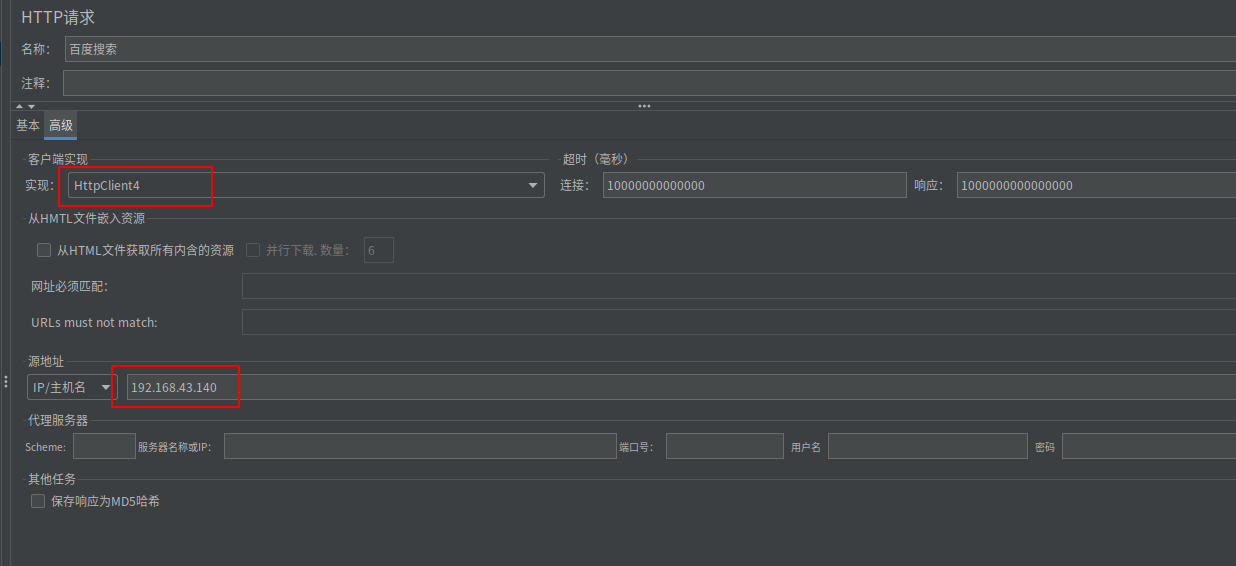


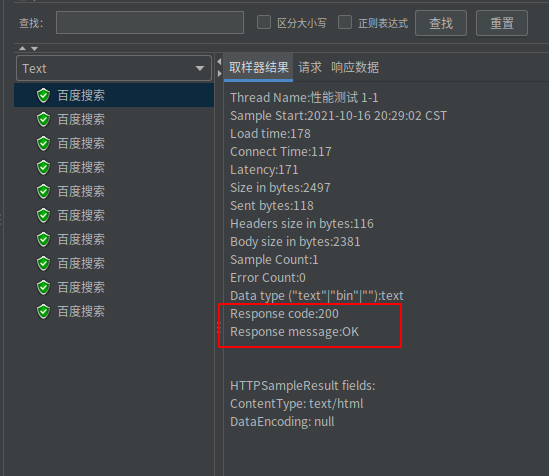
设置网关并查看：





Jmeter的虚假IP配置：

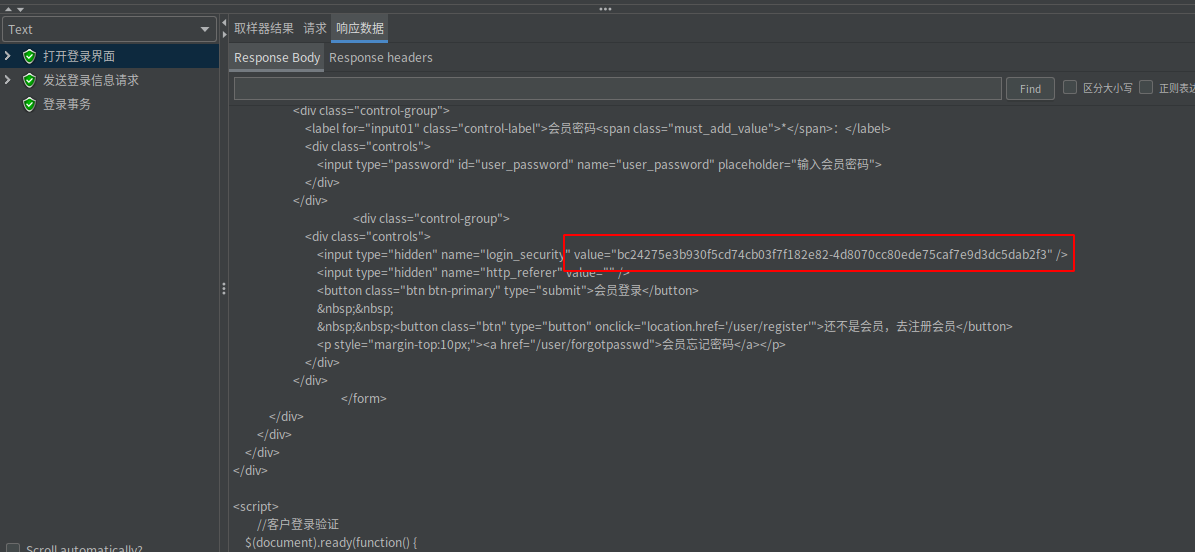
点击测试：

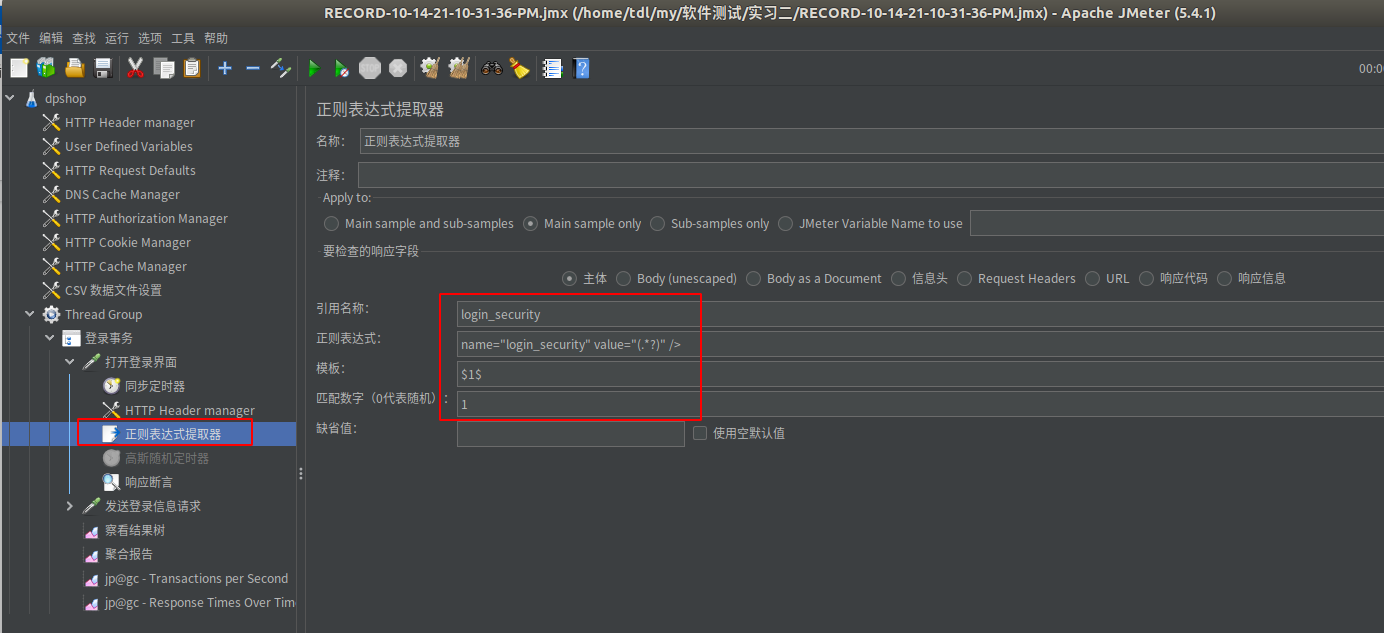


测试成功则说明IP欺骗完成；

##### 3.0.3.10 正则表达式提取器

有的脚本测试的时候，一个接口是需要上一个接口获取的值来进行实时更新的，比如token或者时间戳等信息，就需要使用正则提取，进行参数化，然后进行测试。





### 3.1 测试对象1-DBShop电子商务系统登录功能性能测试

#### 3.1.1 对象描述

**DBShop电子商务系统：**

这是一个开源的，面向学习者的测试环境。它系统提供了比较符合初学者的测试流程和接口，里面的功能比较全，是一个能够实际使用的电子商务平台。

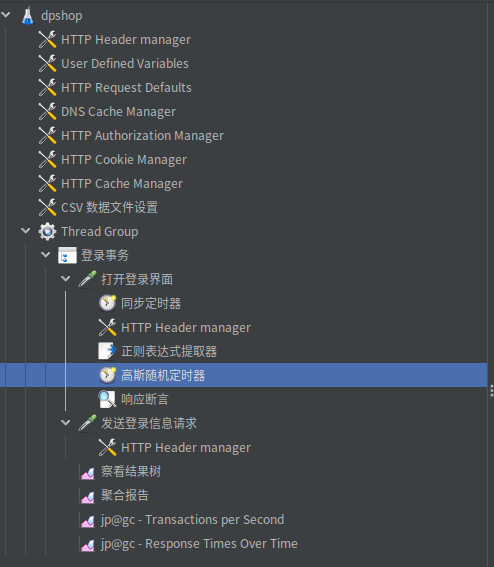
初学者能够比较清晰的了解软件测试流程和性能测试的实际体现意义。

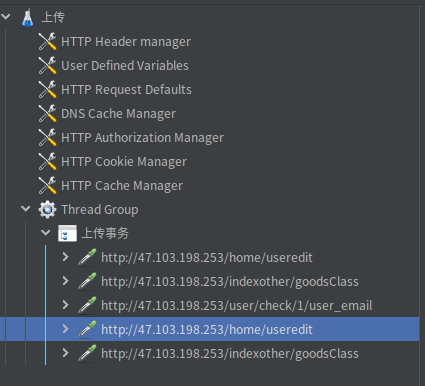
#### 3.1.2 测试描述

使用Jmeter进行测试；

用到了**脚本录制和回放、检测点创建、参数化测试、容量测试、IP欺骗、正则表达式提取器**等主要功能；

#### 3.1.3 测试脚本





这个过程主要是先录制脚本，其中包括登录、上传文件两个过程，最后再监听性能测试的一些重要指标。

其中登录的时候，需要注意的是，它有一个实时更新的token值，需要用正则提取，然后需要模拟多个用户登录，模拟并发，需要创建多个线程，数据参数化，同时需要判断是否成功需要进行检测点判断。还有就是需要进行文件上传。

#### 3.1.4 测试过程

测试过程与前面描述类似，就不再赘述，就是对工具的操作而已。

**表3-1 测试中需要记录的数据**

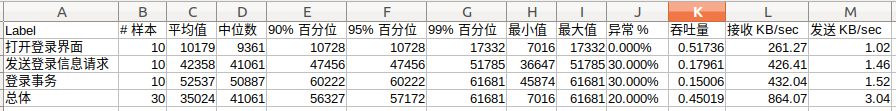
|  |  |
| --- | --- |
| 测试时间 | **1分06秒** |
| 平均响应时间 | **35s** |
| 成功次数 | **30** |
| 失败次数 | 6 |
| web服务器CPU利用率（平均、最大） |  |
| 数据库服务器CPU利用率（平均、最大） |  |

**表3-2 压力测试/负载测试**

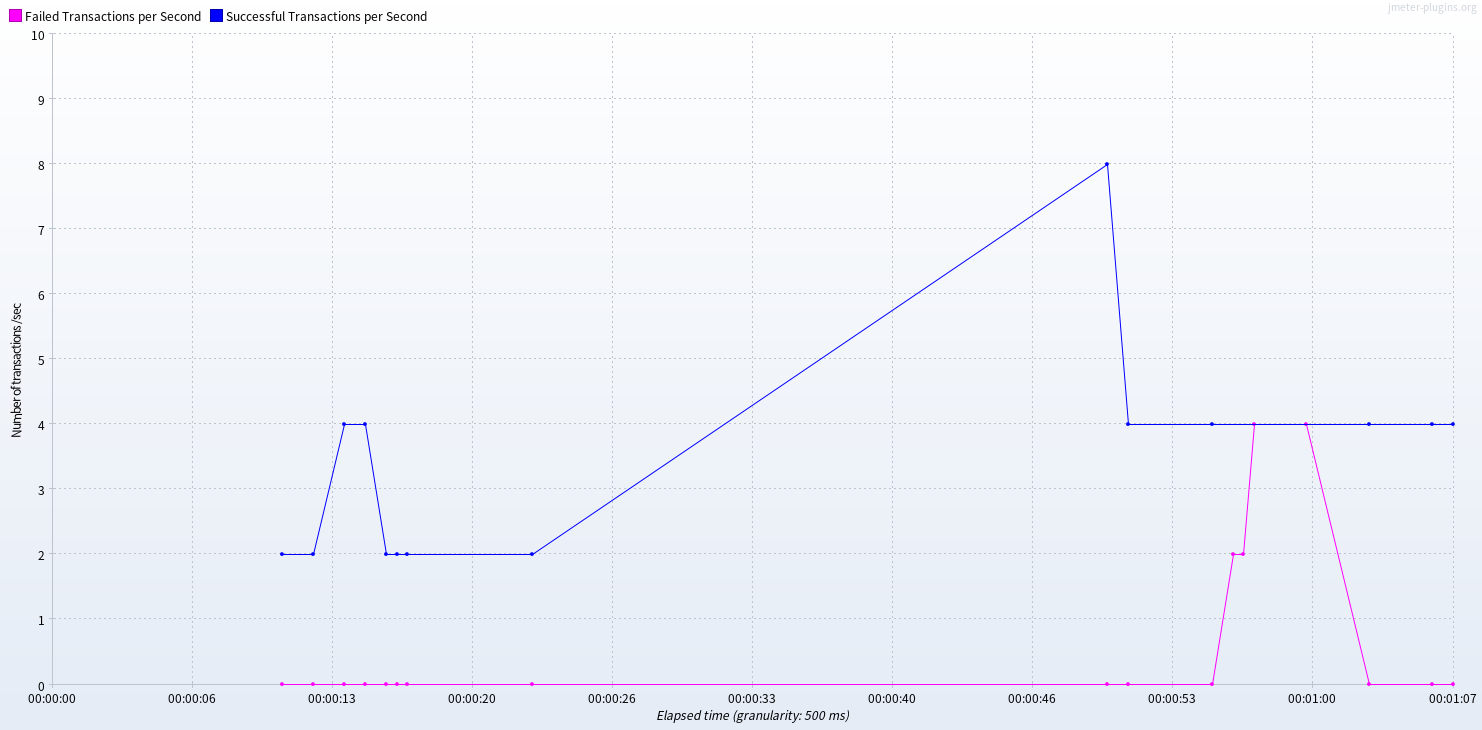
|  |  |  |
| --- | --- | --- |
| 输入/动作 | 输出/响应 | 能否正常工作 |
| 5个用户操作 | 正常返回 | 能 |
| 10个用户操作 | 正常返回 | 能 |
| 20个用户操作 | 正常返回，少量返回失败 | 能 |
| 30个用户操作 | 大部分返回失败 | 不能 |
| 50个用户操作 | 大部分返回失败 | 不能 |
| ……… |  |  |

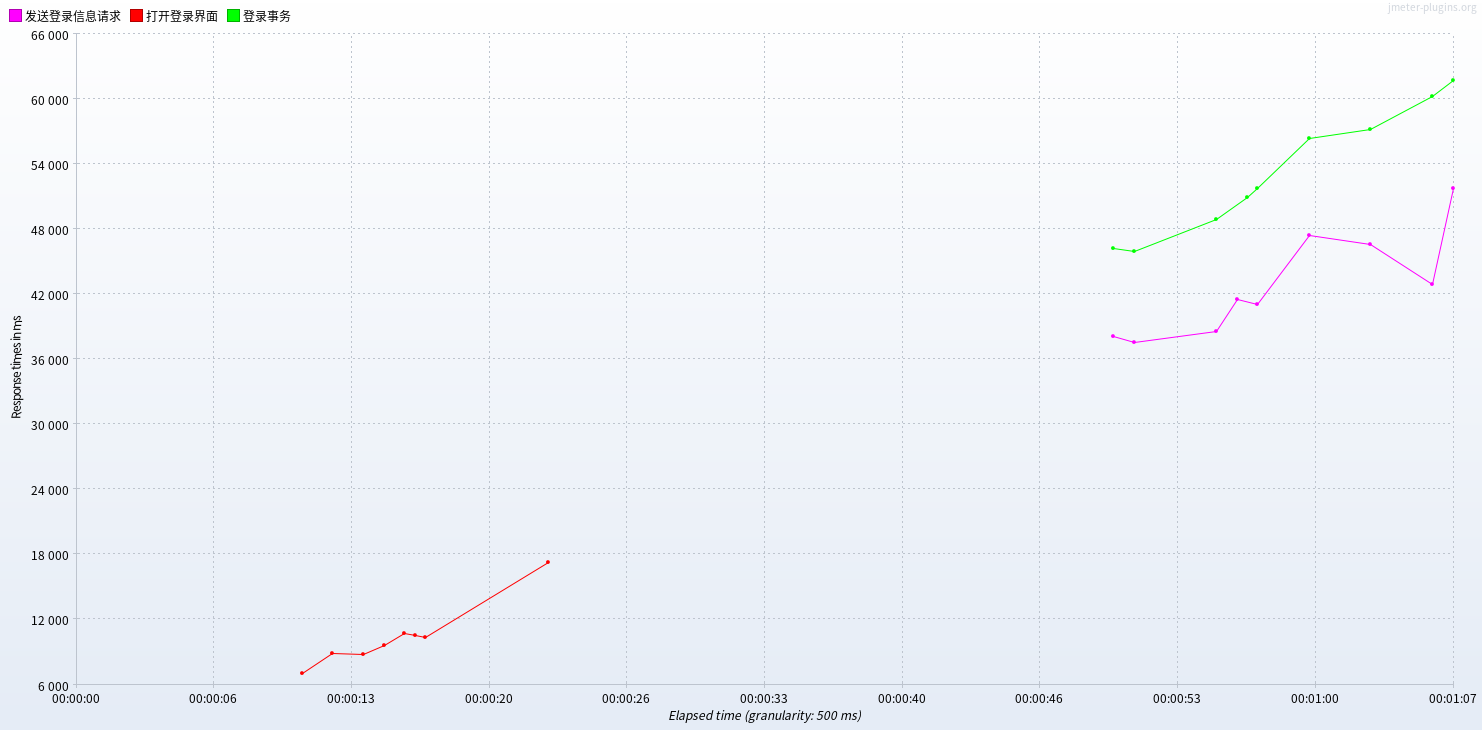
#### 3.1.5 测试结果与分析

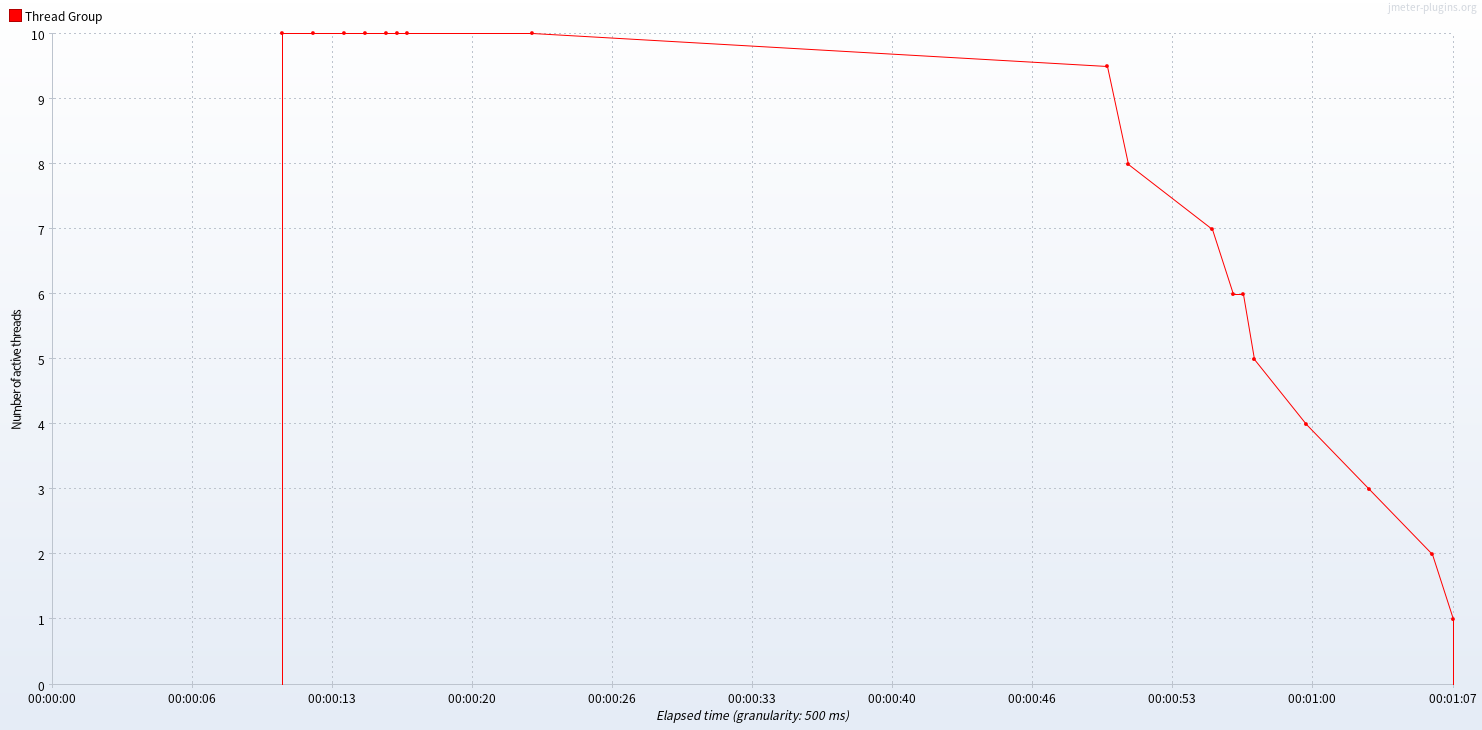
综合报告：

返回结果详情：

TPS：

返回响应时间：

单位时间内活动的线程数:

分析：

经过压力测试和容量测试，可以看出当用户量，也就是线程数多的时候，系统的处理能力是有很大的影响的，当在测试中期，用户量达到最高时，可以从汇报总表中看出，是有登录失败的现象的。

从上传文件也可以看出，在当用户量多时，上传文件的速度会变慢，至于容量我不知道怎么看。

### 3.2 测试对象2-百度搜索功能性能测试

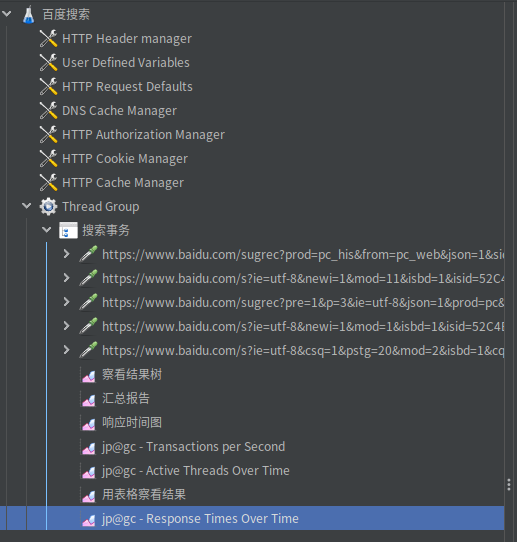
#### 3.2.1 对象描述

测试百度搜索功能的性能

#### 3.2.2 测试环境

与上述一致

#### 3.2.3 测试脚本



进行脚本录制，模拟搜索过程，然后将脚本回放。

注意这里为了保险起见需要进行IP欺骗，虽然可能CUG已经是动态IP地址了。

#### 3.2.4 测试过程

**表3-1 测试中需要记录的数据**

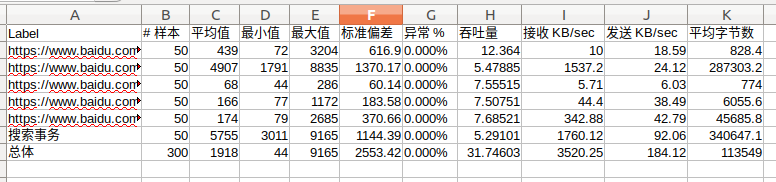
|  |  |
| --- | --- |
| 测试时间 | **9s** |
| 平均响应时间 | **1.9s** |
| 成功次数 | **100%** |
| 失败次数 | 0 |
| web服务器CPU利用率（平均、最大） |  |
| 数据库服务器CPU利用率（平均、最大） |  |

**表3-2 压力测试/负载测试**

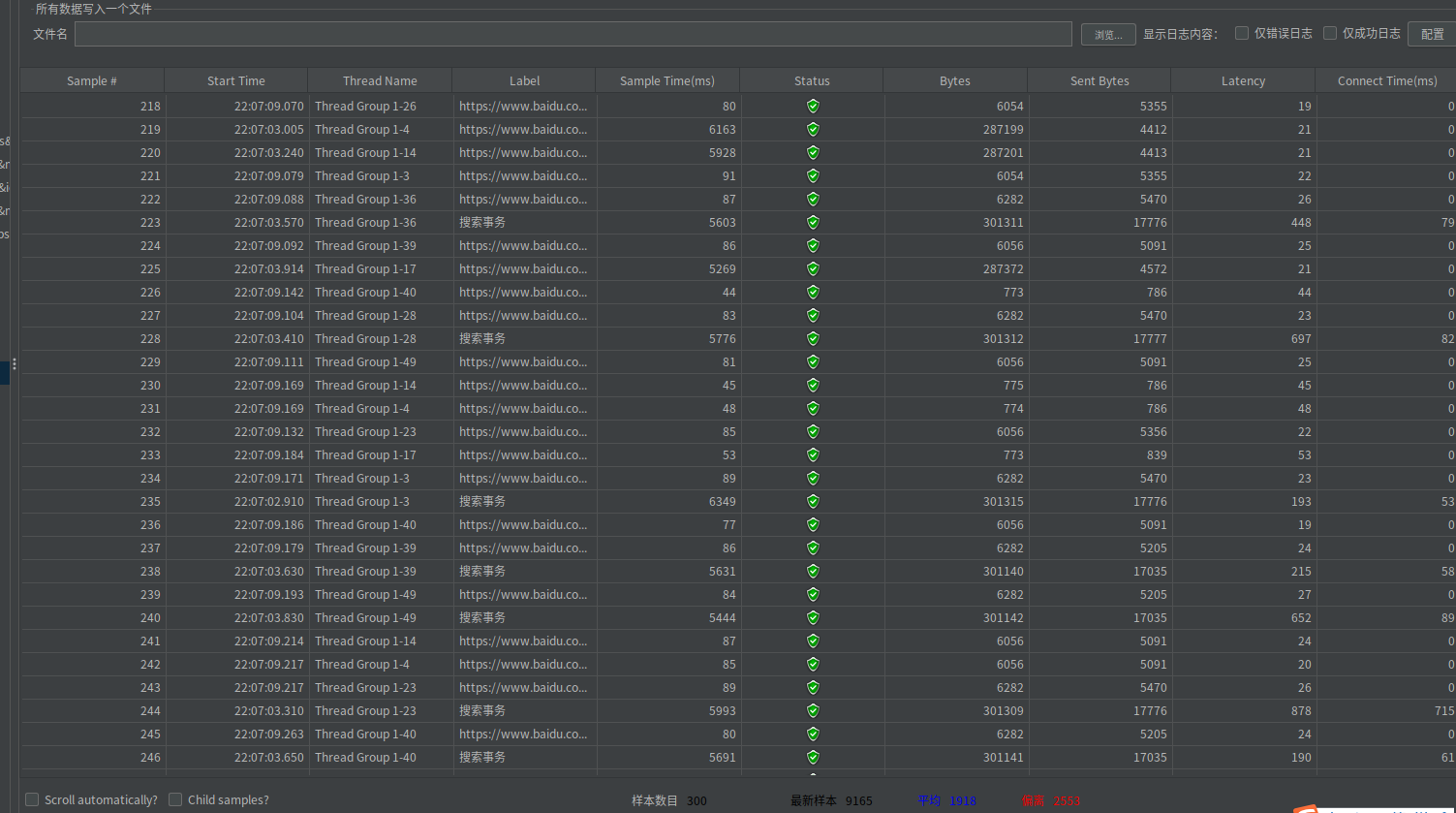
|  |  |  |
| --- | --- | --- |
| 输入/动作 | 输出/响应 | 能否正常工作 |
| 5个用户操作 | 正常 | 能 |
| 10个用户操作 | 正常 | 能 |
| 20个用户操作 | 正常 | 能 |
| 30个用户操作 | 正常 | 能 |
| 50个用户操作 | 正常 | 能 |
| 100个用户操作 | 有少量失败 | 能 |
| 150个用户操作 | 有30%的失败 | 能 |
| 160个用户操作 | 有50-60%左右的失败 | 不能 |
| 170个用户操作 | 大部分失败 | 不能 |
| ……… |  |  |

#### 3.1.5 测试结果与分析

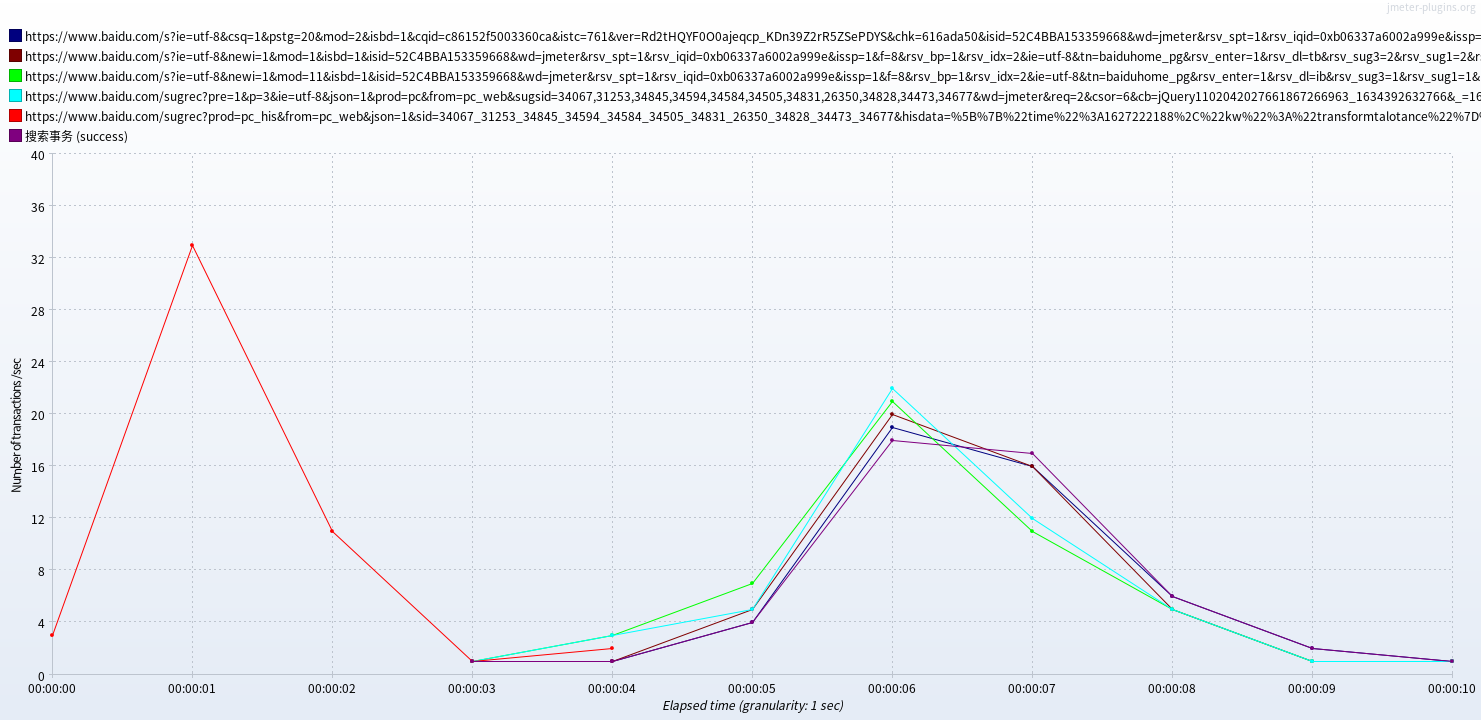
综合报告：



返回结果详情：



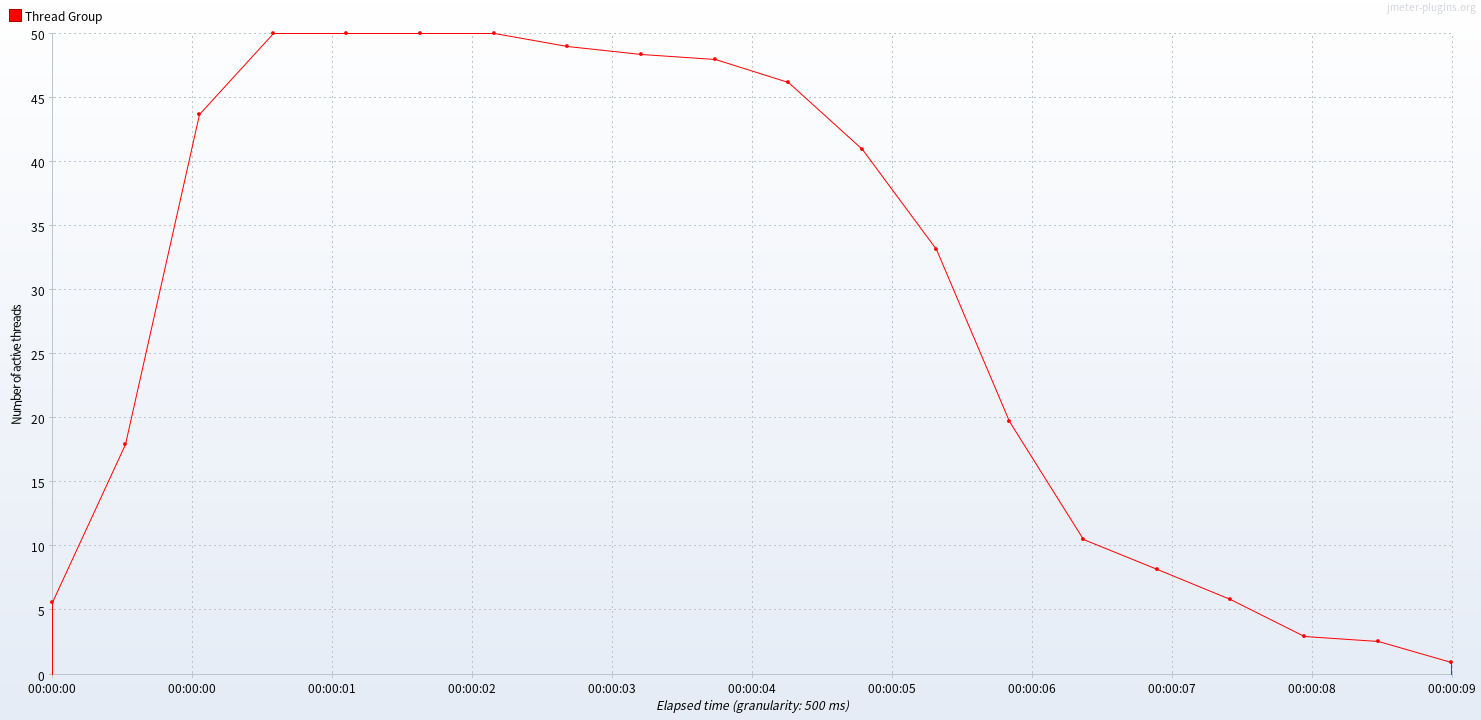
TPS：



返回响应时间：



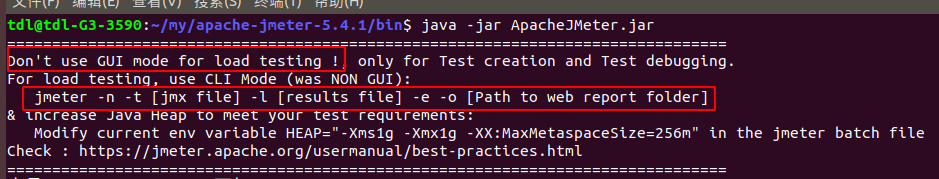
单位时间内活动的线程数:

分析：

主要是测试了一下百度搜索的并发量，很明显要比前面测试的小平台性能好，支持150并发量的良好性能。在170并发左右，展现出明显的下降趋势，其实有可能是我自身电脑的性能的原因，导致一致卡着不动，一直测试不完。

### 3.3 Jmeter命令行和自动化框架搭建简述

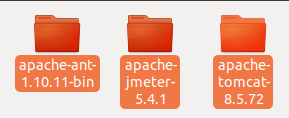
由于我们测试的时候，非常占用自身电脑的资源，所以我们要尽量不用GUI界面进行操作，而是用我们熟悉的命令行操作：

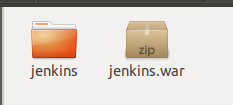


Jmeter在一开始就提醒了我们，所以我们如果要进行测试可以将写好的脚本导出，然后在命令行里面进行运行，这样GUI的占用资源率就会很小。将资源留给测试线程。

当然，这样其实我们的电脑如果配置不够高的话，还是不能很好的进行测试，所以我们可以考虑搭建一个自动化测试服务器：

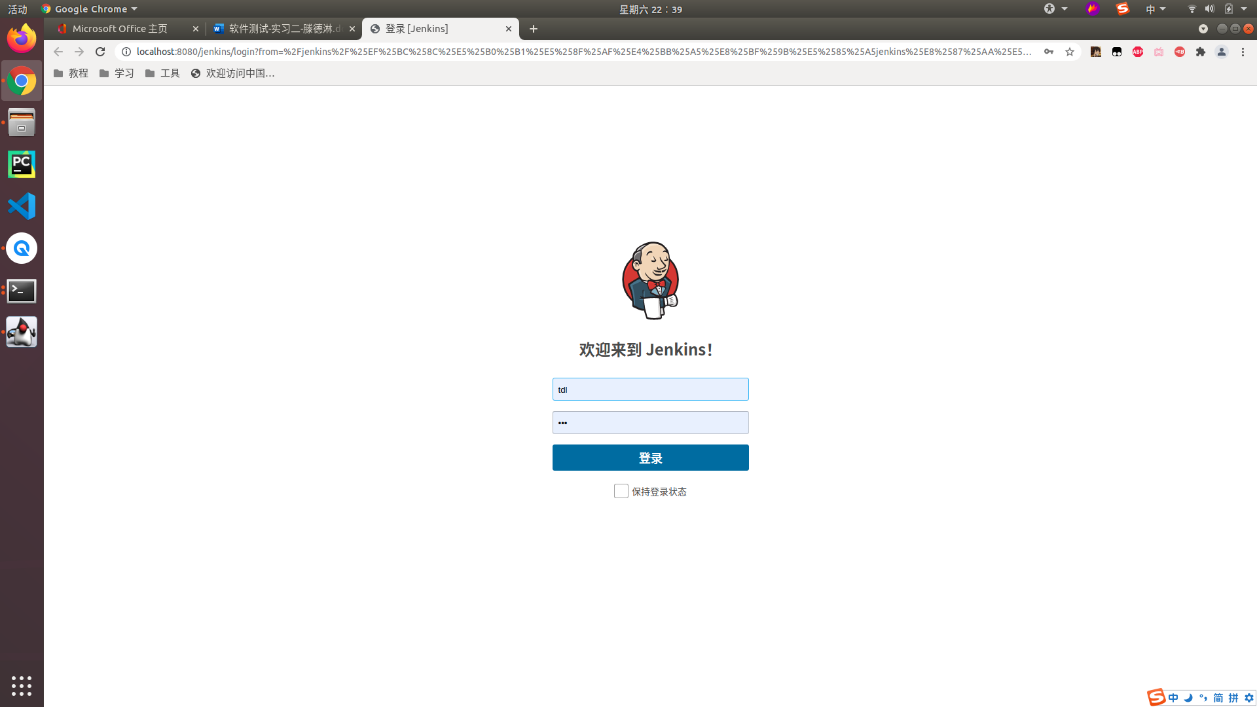
这里我就简单的描述一些怎么搭建：



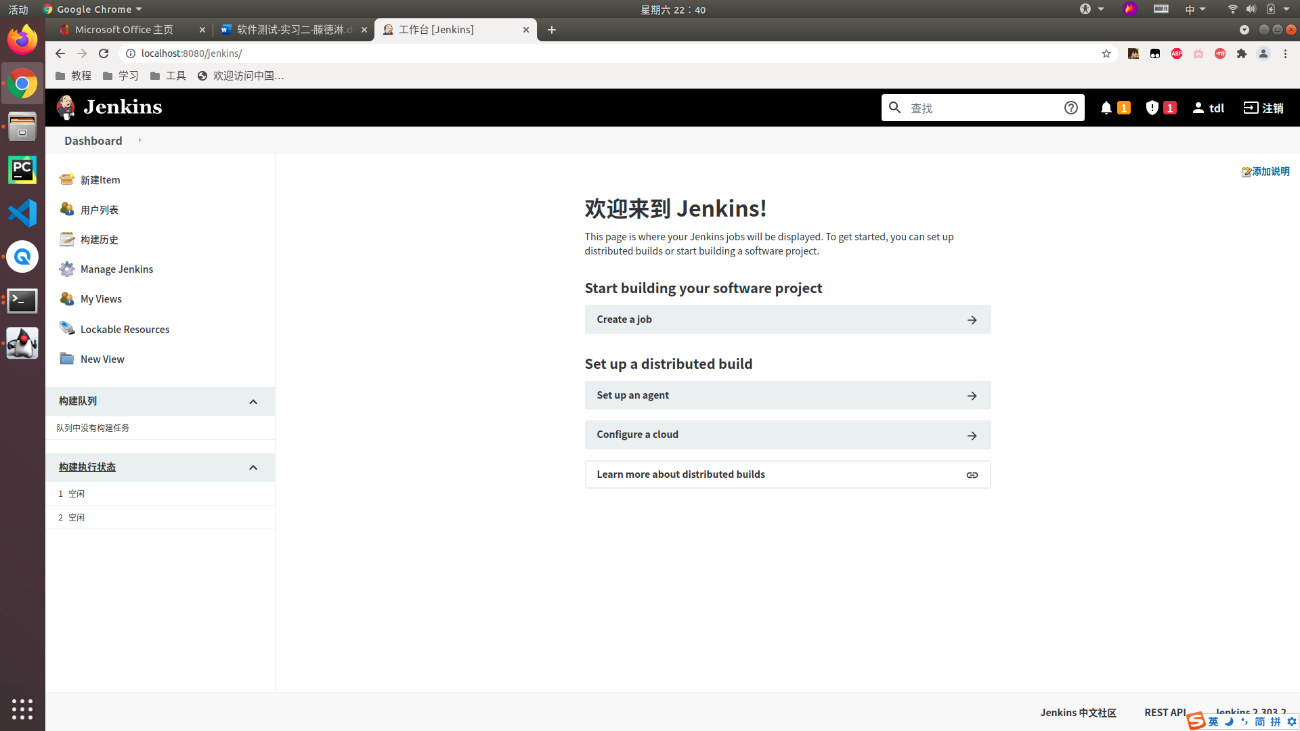


需要拥有这几个环境，而且需要配置好环境变量，然后启动Tomcat，访问 http://localhost:8080/jenkins/，就可以进入jenkins自动化测试平台。

首次登录需要填写一些注册的信息，安装引导就可以完成，之后登录就会出现登录页面：



输入前面自己注册的账号进行登录：

之后我们的测试就都可以在这个平台进行，至于如何使用这个平台还需要再进行学习。