目录 | (六十四期·上)

每次不重样，教你收获最新测试技术！

微信扫一扫关注我们

投稿邮箱：editor@51testing.com

安全漏洞如何分级？

◆ 作者：刘沅斌

近年来，网络安全威胁层出不穷，网络攻击和数据泄露的事件频发。既有多个知名网站大量用户信息泄露的事件；也有委内瑞拉国家电网干线遭到攻击，造成全国大面积停电的针对关键基础设施实施的攻击；还有乌克兰武装部队的第聂伯罗军队自动化控制系统的服务器用户名和密码分别为“admin”、“123456 的弱密码事件。可见小到信息泄露，大到国家安全，网络安全关系着个人和国家的切身利益。

因此，安全测试也日益为企业所重视。安全测试的目的就是尽可能多的发现潜在漏洞。

漏洞是计算机信息系统在需求、设计、实现、配置、运行等过程中，有意或无意产生的缺陷。这些缺陷以不同的形式存在于计算机信息系统的各个层次和环节之中，一旦被恶意主体所利用，就会对计算机信息系统的安全造成损害，从而影响计算机信息系统的正常运行。

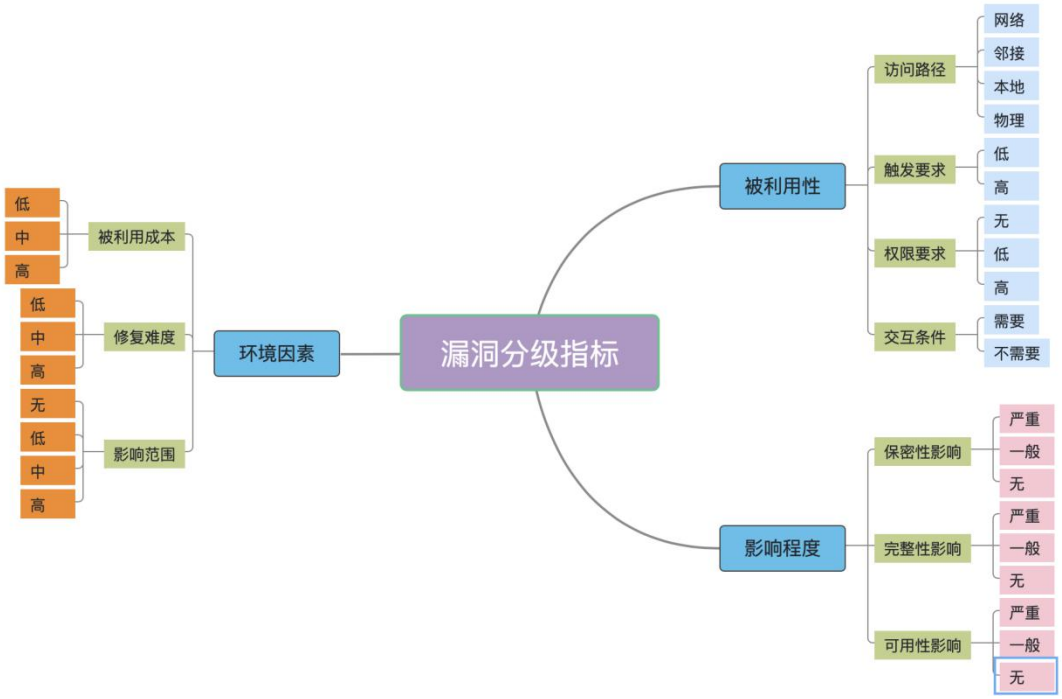
根据漏洞的定义，漏洞本质上也是缺陷，对于缺陷而言，进行缺陷的分级是一个比较重要的议题，因为需要评估修复缺陷的优先级。对于系统测试发现的缺陷，已有较为成熟的分级方案，大都是划分为致命、严重、一般、轻微四个级别。

漏洞也需明确其严重程度划分的标准。为判断修复不同漏洞的优先级提供依据。

本文先通过分析对比业界常用的三个漏洞等级划分模型：网络安全漏洞分类分级指南、CVSS、DREAD 来了解漏洞分级的通用思路。然后会选取两个典型的漏洞来运用这三种模型进行漏洞的定级实践。

【网络安全漏洞分类分级指南】

《GB/T 30279-2020 信息安全技术 网络安全漏洞分类分级指南》是国家标准，该标准替代了《GB/T 33561-2017 信息安全技术 安全漏洞分类》和《GB/T 30279-2013 信息安全技术 安全漏洞等级划分指南》，将漏洞的分级方式划分为技术分级和综合分级。将分级指标划分为被利用性指标类、影响程度指标类和环境因素指标类。



该指南在分级方法上包括漏洞指标类的分级方法、漏洞技术分级方法和漏洞综合分级方法。

具体的分级过程是：

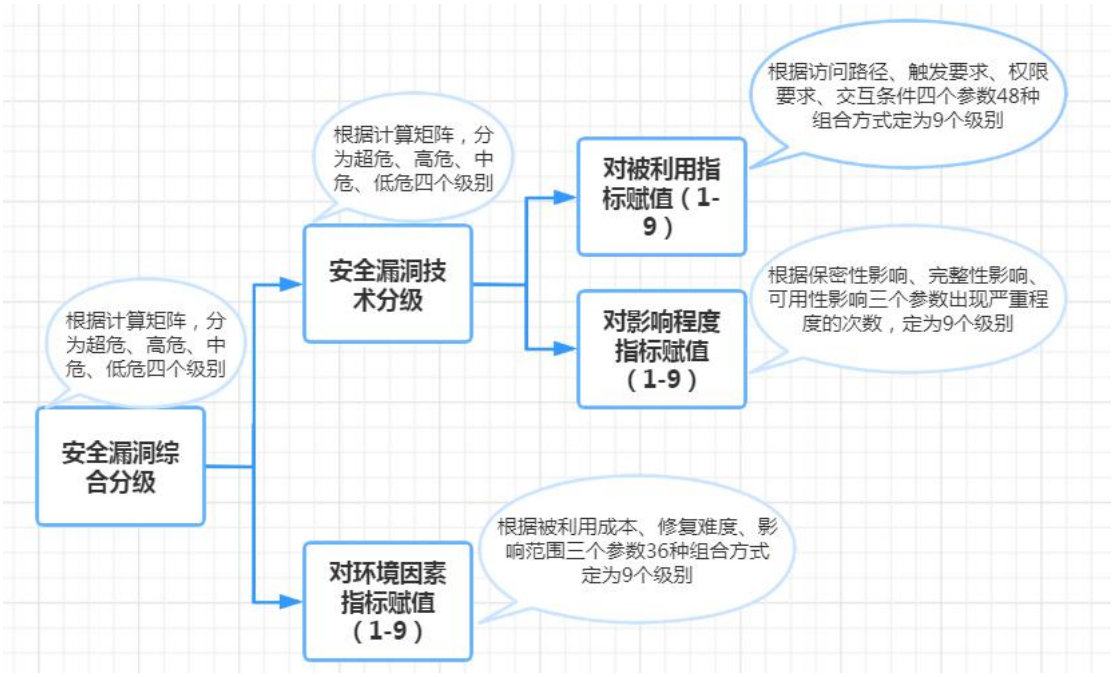
第一步：先对被利用性指标进行赋值，根据赋值结果，按照“被利用性分级表”得到漏洞的被利用性分级。

第二步：对影响程度指标进行赋值，根据赋值结果，按照“影响程度分级表”得到漏洞的影响程度分级。

第三步：根据漏洞的被利用分级和影响程度分级结果，按照“漏洞技术分级结果表”计算得到漏洞的技术分级结果。如果只是做漏洞的技术分级，那么到这一步就结束了，如果还需进行综合分级，则继续下一步。

第四步：对环境因素指标进行赋值，根据赋值结果，按照“环境因素分级表”得到漏洞的环境因素分级。

第五步：根据漏洞的技术分级结果和环境因素分级结果，按照“漏洞综合分级表”计算得到漏洞的综合分级结果。



【CVSS】

CVSS 全称为 Common Vulnerability Scoring System，即“通用漏洞评分系统，是由 NIAC 开发、FIRST 维护的一个开放并且能够被产品厂商免费采用的标准。历经 1.0、2.0、3.0、3.1 版本，目前最新的是 3.1 版本。在 FIRST 上可找到相应的说明文档和得分计算器。

CVSS 得分基于一系列维度上的测量结果，这些测量维度被称为量度（Metrics）。分值范围为 0-10，不同的得分区间对应不同的漏洞级别。

CVSS 由三个度量标准组组成：基本，时间和环境。每个标准组包含多个指标，根据指标的不同取值计算分数，然后根据分数所在区间对应漏洞严重级别。

CVSS 目前被 CVE、CND 等漏洞共享平台使用，在计算时一般是计算基本组分数。

基本组：漏洞的内在性质，随时间推移以及在用户环境中保持不变。基本指标产生的分数介于 0 到 10 之间，然后可以通过对时间和环境指标进行评分来进行修改。包含利

用代码维度和影响维度两个度量维度，代码维度包含五个指标：攻击向量、攻击复杂度、权限要求、用户交互、范围。影响维度包含三个指标：机密性影响、完整性影响、可用性影响。

时间组：随时间变化的漏洞的特征。包含三个指标：攻击代码成熟度、补丁水平、报告可信度。

环境组：用户环境特有的漏洞特征。除包括修正的基本度量的 8 个指标外还包含三个指标：机密性影响、完整性影响、可用性影响。

【DREAD】

DREAD 模型，是一个威胁评级模型，威胁评级是微软威胁建模流程中的一个阶段，DREAD 分别是威胁评级的 5 个指标的英文首字母：Damage Potential（潜在损失）、Reproducibility（重现性）、Exploitability（可利用性）、Affected users（受影响的用户）、Discoverability（可发现性）。

这 5 个指标每个指标的评级分为高中低三等，在下表中每个等级分别以 3、2、1 的分数代表权重值，最终威胁的危险评级由这 5 个指标的加权平均算出。

等级	高（3）	中（2）	低（1）
Damage Potential	获取完全验证权限，执行管理员操作，非法上传文件	泄露敏感信息	泄露其他信息
Reproducibility	攻击者可随意再次攻击	攻击者可重复攻击，但有时间限制	攻击者很难重复攻击过程
Exploitability	初学者短期能掌握攻击方法	熟练的攻击者才能完成这次攻击	漏洞利用条件非常苛刻
Affected users	所有用户，默认配置，核心用户	部分用户，非默认配置	极少数用户，匿名用户
Discoverability	漏洞很显眼，攻击条件很容易获得	在私有区域，部分人能看到，需要深入挖掘漏洞	发现漏洞极其困难

根据上表的标准，每个指标的取值范围为 1-3，则结果范围为 5-15，那就可以参考具体的得分和风险级别的如下：

总值	风险级别
5-7	低
8-11	中
12-15	高

【案例实践】

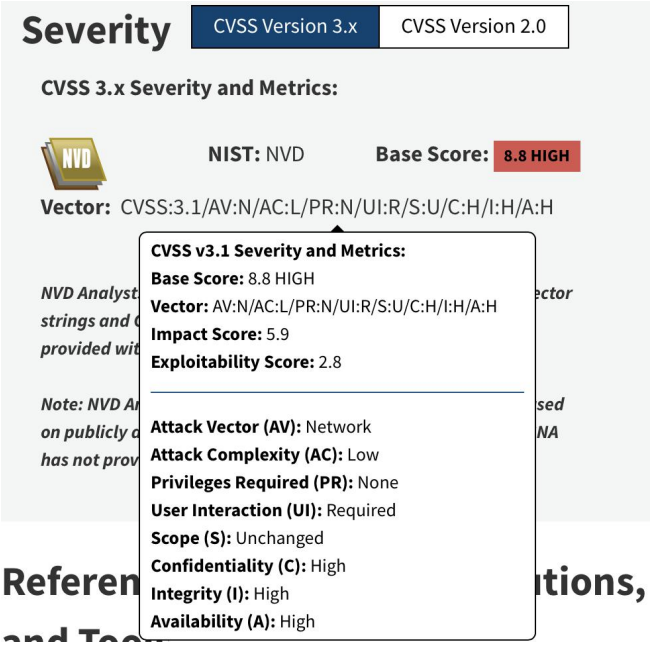
以下通过两个案例展示三种模型的漏洞分级计算过程：

案例一：CVE-2021-21206 谷歌 Google Chrome 资源管理错误漏洞

Google Chrome 存在资源管理错误漏洞，该漏洞源于 Google Chrome 浏览器中的 Blink 资源管理错误使得远程攻击者可以诱骗受害者访问特制的网页，并在系统上执行任意代码。

1.CVSS

我们在 NVD 官网上可以看到该漏洞的 CVSS3.1 评分是 8.8，对应的漏洞级别是高危。同时在该网站上可以看到评分的具体细节如下图。



具体的指标和取值见下表。

组别	指标	指标值	分值/级别
基本组	攻击向量	Network (N)	8.8/High
	攻击复杂度	Low (L)	
	权限要求	None (N)	
	用户交互	Required (R)	
	范围	Unchanged (U)	
	保密性影响	High (H)	
	完整性影响	High (H)	
	可用性影响	High (H)	

2.DREAD

指标	取值	分值/级别
Damage Potential	高 (3)	13/高
Reproducibility	高 (3)	
Exploitability	中 (2)	
Affected users	高 (3)	
Discoverability	中 (2)	

3.网络安全漏洞分类分级指南

此处主要展示漏洞技术分级计算。

类别	指标	取值	中间级别	漏洞技术分级级别
被利用性	访问路径	网络	8	高危
	触发要求	低		
	权限要求	无		
	交互条件	需要		
影响程度	保密性影响	高	9	
	完整性影响	高		
	可用性影响	高		

案例二：百度外卖平台系统遭非法侵入，账户被篡改 4900 余万元的事件

根据 2018 年的一篇报道，此次事件是北京小度科技有限公司在 2017 年 10 月主动报案，嫌疑人都是利用百度外卖系统漏洞，将提现金额改为负数，从而实现反向充值自己账户余额，并使用账户余额在“百度外卖”平台上进行消费，总共被篡改 4900 余万元。

这是一个典型的请求参数篡改的漏洞，对于用户输入的信息没有遵循用户输入不可信的原则，没有进行数据的类型长度范围的合法性检查，也没有进行必要的业务风险控制。

1.CVSS

组别	指标	指标值	分值/级别
基本组	攻击向量	Network (N)	7.6/High
	攻击复杂度	Low (L)	
	权限要求	Low(L)	
	用户交互	None (N)	
	范围	Unchanged (U)	
	保密性影响	Low(L)	
	完整性影响	High(H)	
	可用性影响	Low(L)	

2.DREAD

指标	取值	分值/级别
Damage Potential	中 (2)	14/高
Reproducibility	高 (3)	
Exploitability	高 (3)	
Affected users	高 (3)	
Discoverability	高 (3)	

3.网络安全漏洞分类分级指南

类别	指标	取值	中间级别	漏洞技术分级级别
被利用性	访问路径	网络	8	高危
	触发要求	低		
	权限要求	低		
	交互条件	不需要		
影响程度	保密性影响	一般	6	
	完整性影响	高		
	可用性影响	一般		

【总结】

通过分析上述三种比较典型的漏洞分级模型，可以看 CVSS 作为一个国际标准，指标较多，可通过提供的计算器自动计算。但也存在指标理解有一定难度且计算过程复杂的缺陷。DREAD 也是一个国际标准，指标维度较为精炼，但各个维度指标权重一样，需要在实际使用的时候进行适用性的调整。网络安全漏洞分类分级指南是国家标准，分级采用了参考分级表的形式，比较新颖，但不同指标赋值组合较多，在具体实施的时候如果能辅助自动化的手段会比较好。

虽然这三种模型有其不同的特点，但可以看出模型的思路均是先制定漏洞分级相关的指标，然后对每个指标赋值进行标准的定义和权重的分配，最后通过加权求和计算得分，不同得分区间对应相应漏洞级别，或者是通过不同指标赋值结果的组合匹配对应的漏洞级别。

Jenkins 简单操作

◆作者：测试安静

前言：

目前很多公司都已经实现了持续集成，说到了持续集成，当然离不开我们的 jenkins，一般我们日常测试的版本都是通过 jenkins 进行编译自动编译出来的，今天安静简单介绍下对于我们测试人员来说，jenkins 有那些需要掌握的。

Jenkins

Jenkins 基于 JAVA 开发环境的一种开源项目，是一款流行的开源持续集成（Continuous?Integration）工具，广泛用于项目开发，具有自动?化构建、测试和部署等功能。

Jenkins 安装

安装包形式（msi）

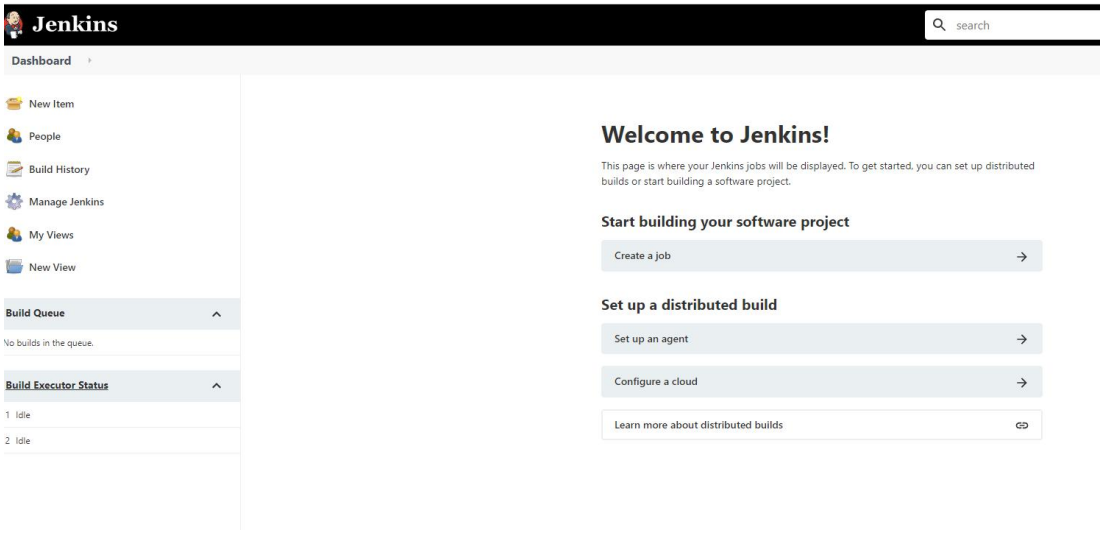
1、jenkins 是基于 java 环境的，所以肯定要安装 JDK，我们需要下载 JDK 的对应的版本，然后傻瓜式安装（疯狂点击下一步）然后配置对应的安装包即刻。

2、进入到 jenkins 官网中，找到对应下载的系统版本，然后进行下载安装。

3、再次疯狂点击下一步，就安装成功了。

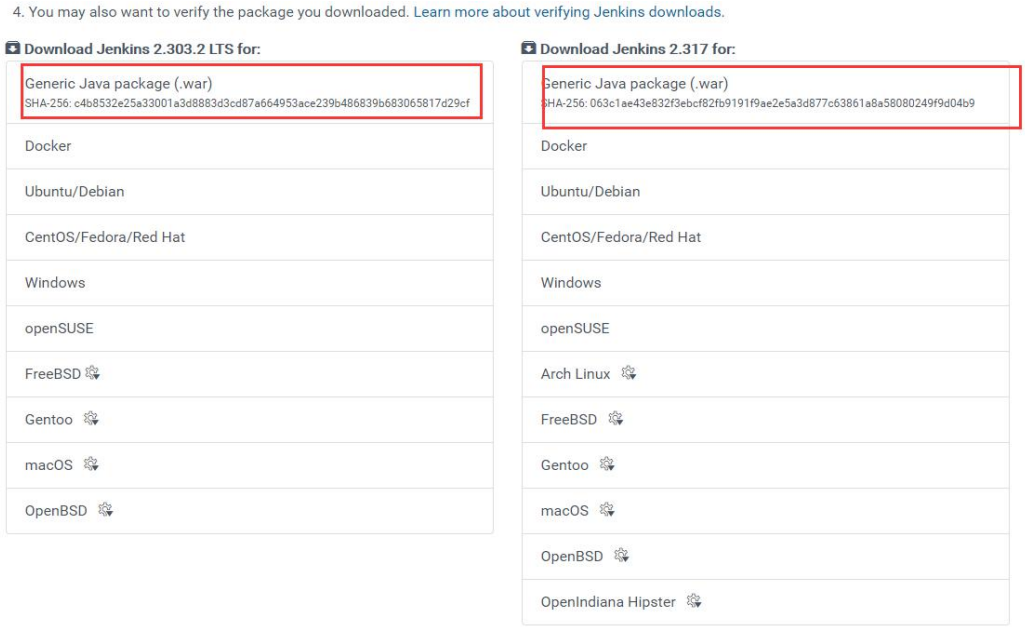
4、在浏览器中输入 http://localhost:8080/进行验证是否能进入到 jenkins 配置页面中。

5、根据上图内容找到对应的 jenkins 的账号密码，然后进行根据操作进行配置，配置完成后再次进入到 http://localhost:8080/ 完成登录操作后，就能进入到 jenkins 页面了。

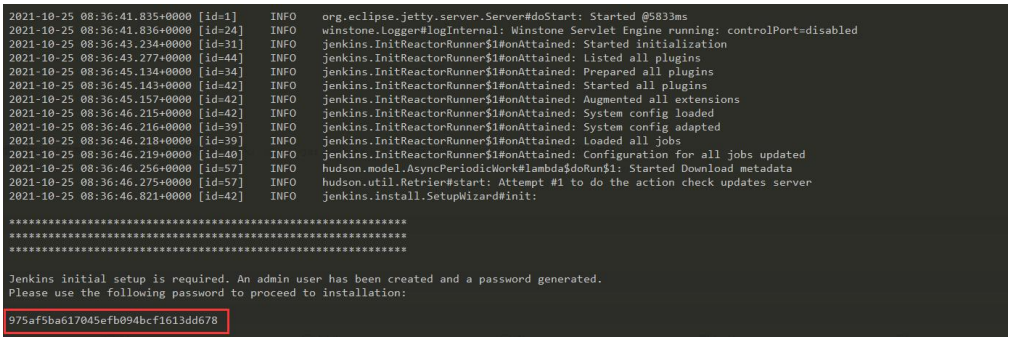


war 包形式安装

1、进入到 jenkins 的官网链接中，找到 war 包下载的方式进行下载到本地。



2、找到下载本地 war 路径，通过 cmd 进行打开，在 cmd 中输入 Java?-jar?jenkins.war 来进行安装，然后打开 http://localhost:8080/进行继续配置 jenkins，这里的 cmd 中也会显示我们需要完成配置的密码。



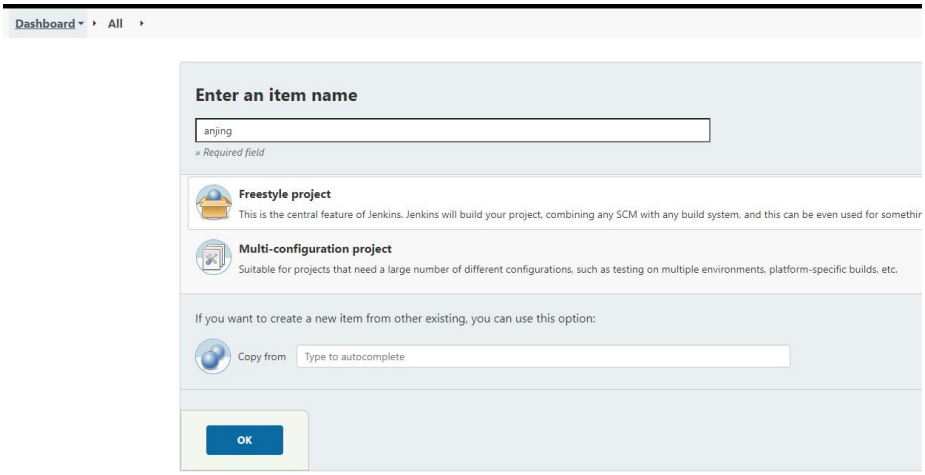
3、这样每次启动都需要进行输入对应命令，我们也可以将命令封装成 bat 的形式，方便我们每次都手动输入。

Jenkins 执行本地代码

Jenkins 已经安装成功了，接下来就是需要我们进行创建项目执行代码的一些操作了。

创建项目

1、登录到 jenkins 首页，点击新建项目，在新建项目中输入一个项目名称，点击确定。

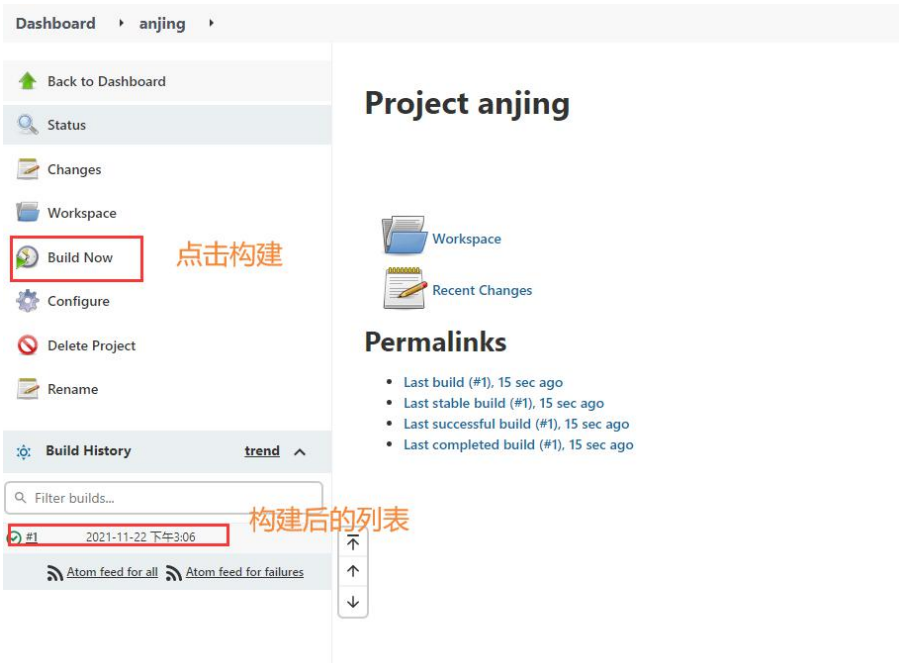


2、进入新创建的项目配置页面，找到 build 中，然后选择执行 Windows 的脚本（安静这里书 Windows 环境）。

3、在构建中输入对应执行运行执行测试代码。这路需要输入 CMD 的操作命令，然后点击保存，应用。

构建并查看结果

1、上步骤保存后，点击构建按钮进行构建。然后就能在下面看到我们的构建项目内容了。



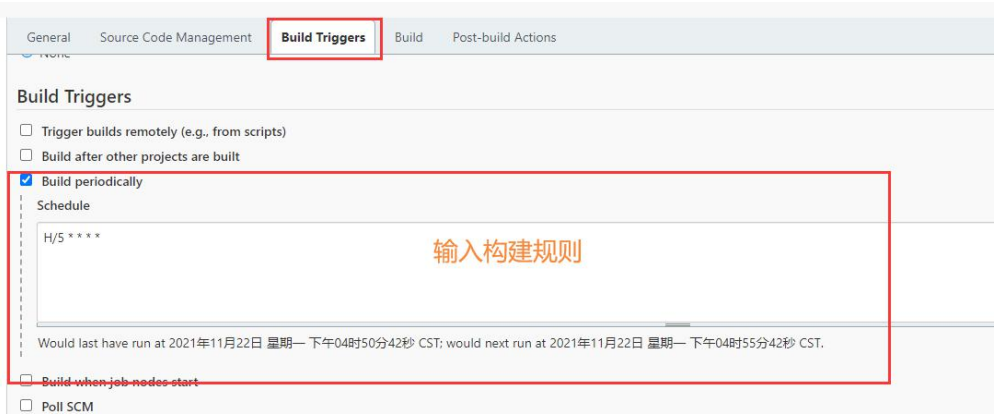
2、选择下面的构建后列表成功的点击进入，打开控制台输入信息（Console?Output）查看内容。

Jenkins 配置定时任务

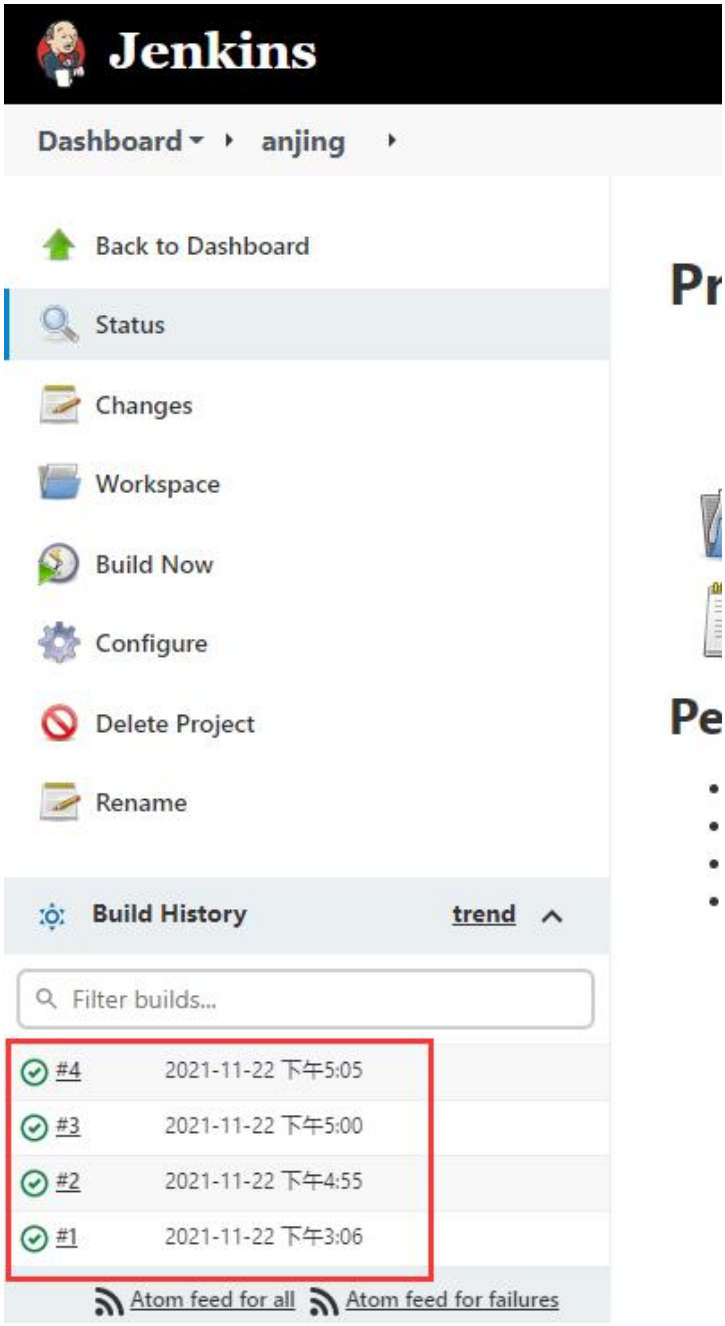
Jenkins 也有一个非常还用的功能就是定时任务，就是可以在通过 jenkisn 设置定时时间来执行我们的测试代码。

定时构建

1、找到刚刚创建的项目，然后进入到配置页面中，找到 Build?Triggers（构建触发器）选择 Build?periodically，在里面输入需要定时的规则，安静这里输入的是 H?5?*?*?*?*表示 5 分钟触发一次。输入完成后点击保存。



2、这里安静设置的是 5 分钟自动构建一次，喝杯水，等待 5 分钟，然后我们进行查看构建项目列表中，已经有自动构建成功的内容了，并且时间都是 5 分钟。



构建语法规则

表格一共可以书写 5 个* * * * * 其中每个*之间需要通过空格或者 tab 键进行隔离开：

第一个*表示分钟，取值 0~59；

第二个*表示小时，取值 0~23；

第三个*表示一个月的第几天，取值 1~31；

第四个*表示第几月，取值 1~12；

第五个*表示一周中的第几天，取值 0~7，其中 0 和 7 代表的都是周几。

举例

H/15 * * * * ：表示每隔 15 分钟进行构建一次项目。

H H/3 * * *：表示每隔 3 小时进行构建一次项目。

H 12 * * * ：表示每天的 12 点进行构建一次项目。

H?H(9-18)/2 * * 1-5 ： 表示每个工作日的早上 9 点到 18 点每隔 2 个小时进行构建一次项目。

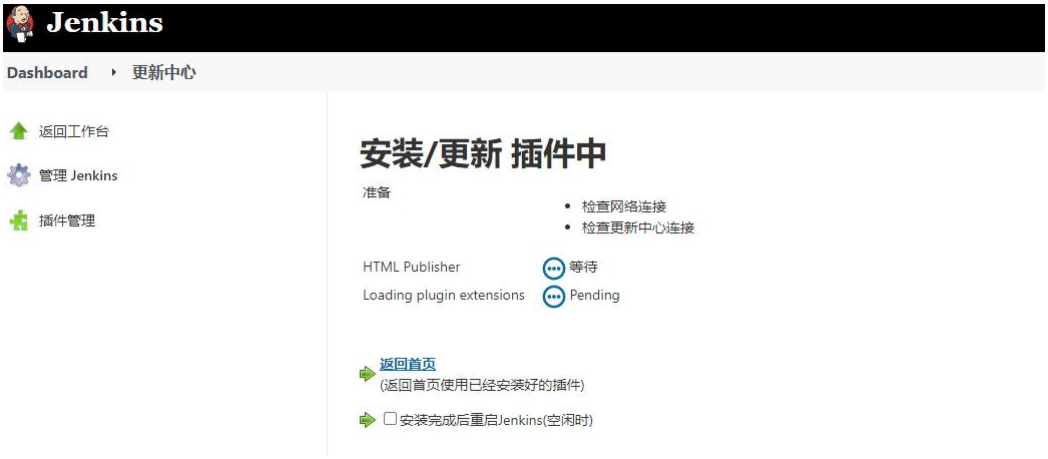
Jenkins 展示 HTML 报告

自动化测试结束后生成的测试报告内容，这里我们也可以将测试报告通过 jenkins 自动生成，这样就可以直接在 jenkins 上进行查看测试报告了。这里安静先介绍一种通过 html 展示报告内容。

插件下载

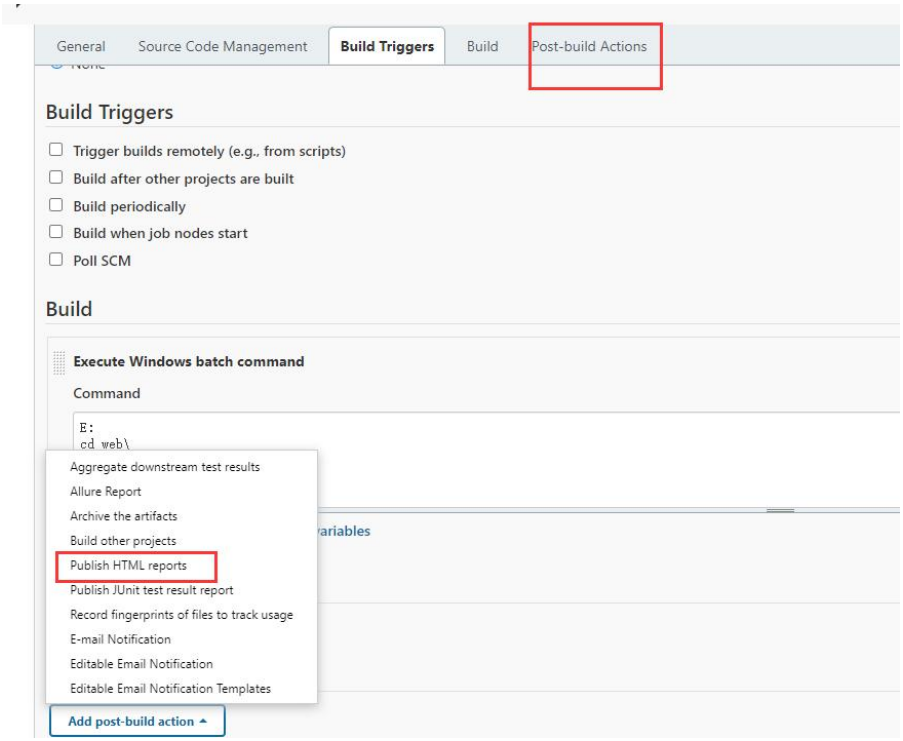
1、这里需要依赖于 jenkins 的插件 HTML?Publisher，直接在 jenkisn 的插件管理页面中进行搜索。

2、找到后进行下载安装，安装成功后进行输入 http://ip:8080/restart 进行重启 jenkins。



配置 HTML 参数

下载安装已经成功了，我们就需要进入到我们的配置环节了，进入到我们测试项目配置中找到构建后操作（Post-build?Actions）添加选择 HTML 报告。



添加完成后进行配置报告内容：

HTML directory to archive：表示报告路径，一定要和代码生成的报告路径一致。

Index page：表示报告名称，和代码生成的一致。

Report title：报告显示 jenkins 上的标题。

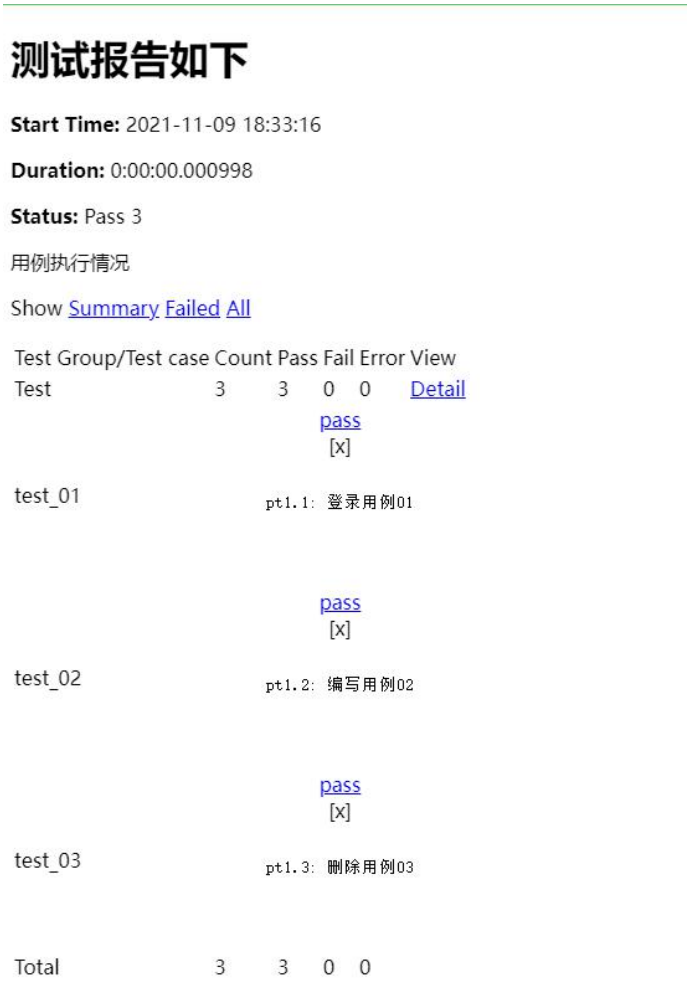
测试报告生成

我们需要在上述步骤中将我们生成测试报告的代码放到 build 执行的路径下进行执行，然后点击保存，进行构建项目。

```
# 测试代码
import unittest
import HTMLTestRunner_cn
class Test(unittest.TestCase):
    def test_01(self):
        '''测试用例 01'''
        print('---用例 01---')

    def test_02(self):
        print('---用例 02---')

    def test_03(self):
        '''测试用例 03'''
        print('---用例 03---')
if __name__ == '__main__':
    # 测试报告地址
    fp = open('report.html', "wb")
    # 报告详情
    runner = HTMLTestRunner_cn.HTMLTestRunner(stream=fp,
```

```
                                        title=u'自动化测试报告,测试结果如下：',
                                        description=u'用例执行情况：')
    # 实例化
    testunit = unittest.TestSuite()
    # 加载用例
    testunit.addTests(unittest.TestLoader().loadTestsFromTestCase(Test))
    # 执行用例
    runner.run(testunit)
    # 关闭报告
    fp.close()
```

构建成功后，直接打开测试报告查看测试报告内容，发现报告有问题，经过排查发现是这里的报告缺少了 css 文件。



测试报告如下

Start Time: 2021-11-09 18:33:16

Duration: 0:00:00.000998

Status: Pass 3

用例执行情况

Show Summary Failed All

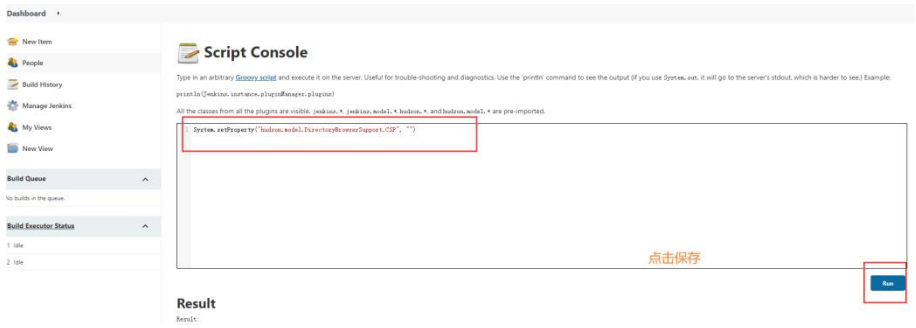
Test Group/Test case	Count	Pass	Fail	Error	View
Test	3	3	0	0	Detail
test_01		pass [x]			pt1.1: 登录用例01
test_02		pass [x]			pt1.2: 编写用例02
test_03		pass [x]			pt1.3: 删除用例03
Total	3	3	0	0	

问题解决

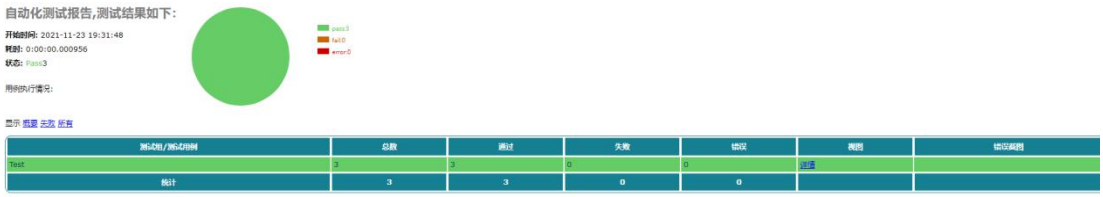
方法一

进入 jenkins 系统页面找到 Script?Console 进入后，在脚本命令处输入命令：

System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")



这个时候我们再去重新构建，打开我们的测试报告，就会发现报告已经成功的生成了（说明已经将我们的 css 文件成功的添加进去了）。



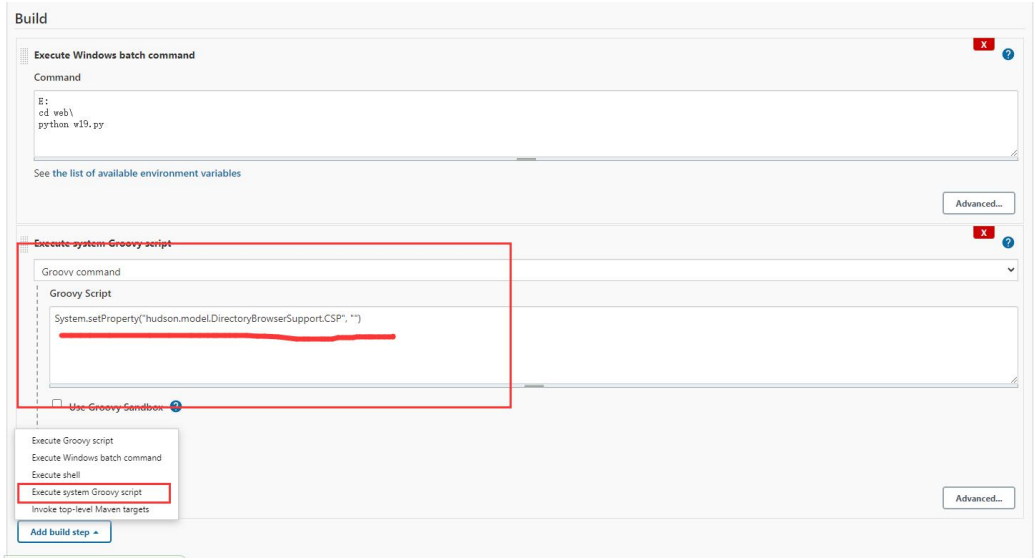
注意：这里需要注意，这里重启 jenkins 的话会失效

方法二：

安装 Grooy 插件，这里可以方便的帮助我们解决这个问题（安装过程不在介绍，和上述安装 HTML 插件一样）。

安装完成后，需要在项目构建中，找到 Execute?system?Groovy?script?中输入以下命令：

System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")



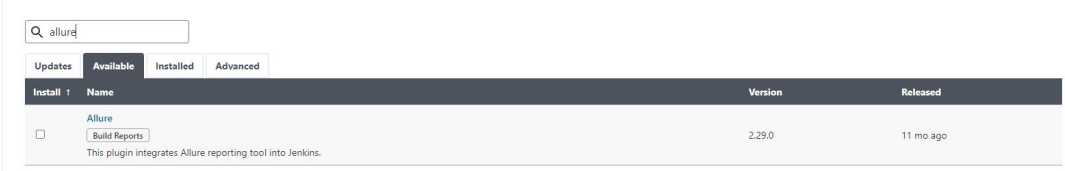
配置完成后进行保存，保存后再次点击构建项目内容，然后进行查看测试报告。发现已经是成功的了。

Jenkins 生成 Allure 报告

Jenkins 既然能生成 HTML 报告，那么肯定也能生成 Allure 报告，同样的也是需要去 Jenkins 上进行下载 Allure 的插件。

插件下载

1、在插件管理页面搜索 allure，进行下载安装，安装后需要重启 jenkins。



2、进入到 jenkins 管理页面找到 Global?Tool?Configuration 进入到全局工具设置页面中，在页面中找到 allure，进行安装 allure 的工具。

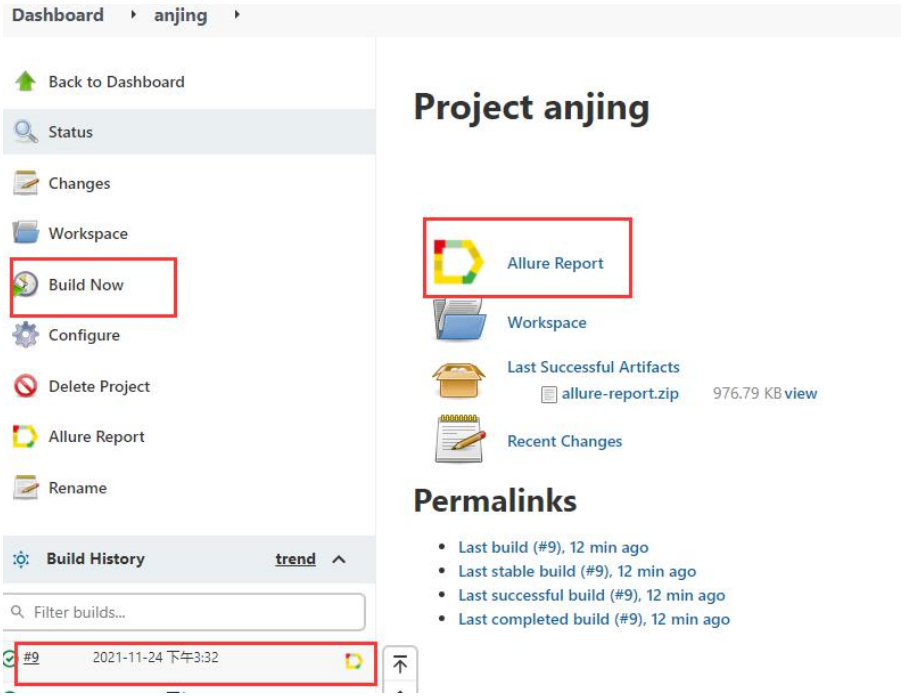
3、上面的配置完成后，写一个关于 pytest 的脚本。

```
# 测试脚本
import pytest

data = [{"user": 'anjing', "pwd": "123456"},
        {"user": "admin", "pwd": "admin"},
        {"user": "test", "pwd": "test"}]


@pytest.fixture(params=data)
def login(request):
    yield request.param


class Test01:

    # 通过 parametrize 的方法来实现参数化
    @pytest.mark.parametrize('aaa', data)
    def test_01(self, aaa):
        print('用户名：{}'.format(aaa['user']))
        print('密码：{}'.format(aaa['pwd']))
```

```
    # 通过 fixture 的方法来实现参数化
    def test_02(self, login):
        print('用户名：{}'.format(login['user']))
        print('密码：{}'.format(login['pwd']))


if __name__ == '__main__':
    pytest.main(['-vs' ,'--alluredir ./report/'])
```

配置构建操作后配置 allure 报告：

配置完成后点击保存，进入到配置项目中点击构建。



可以看到我们已经构建成功了，报告也生成了，点击查看 allure 报告。

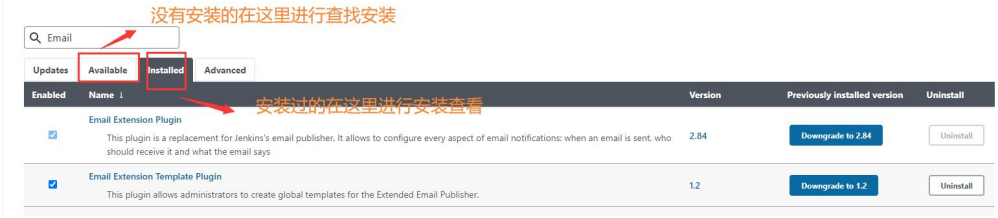
Jenkins 配置邮箱

自动化测试结束后都会输出测试报告，然后通过邮箱的形式进行发送给领导和组内成员，Jenkins 上也有邮箱发送配置，可以通过 jenkins 配置邮箱信息，然后每次构建完成后进行发送邮箱发送。

@#

插件下载

想要实现 jenkins 实现邮箱发送，需要进行下载 jenkins 的邮箱插件 Email Extension Plugin 和 Email Extension Template Plugin 两个插件，直接进入到 jenkins 的插件下载中进行下载，下载完成后同样需要重启 jenkins



配置邮箱

1、进入到 jenkins 设置页面中找到 Jenkins Location 进行配置邮箱地址和 jenkins 的地址。

2、继续往下找 Extended?E-mail?Notification 然后进行配置 smtp 服务，端口，发件人的默认邮箱，授权码，还需要打开 ssl 服务。

SMTP server：smtp 的服务；

SMTP Port：邮箱端口默认 465；

SMTP Username：邮箱默认发件人；

SMTP Password：邮箱授权码（需要到 QQ 邮箱设置中查看）；

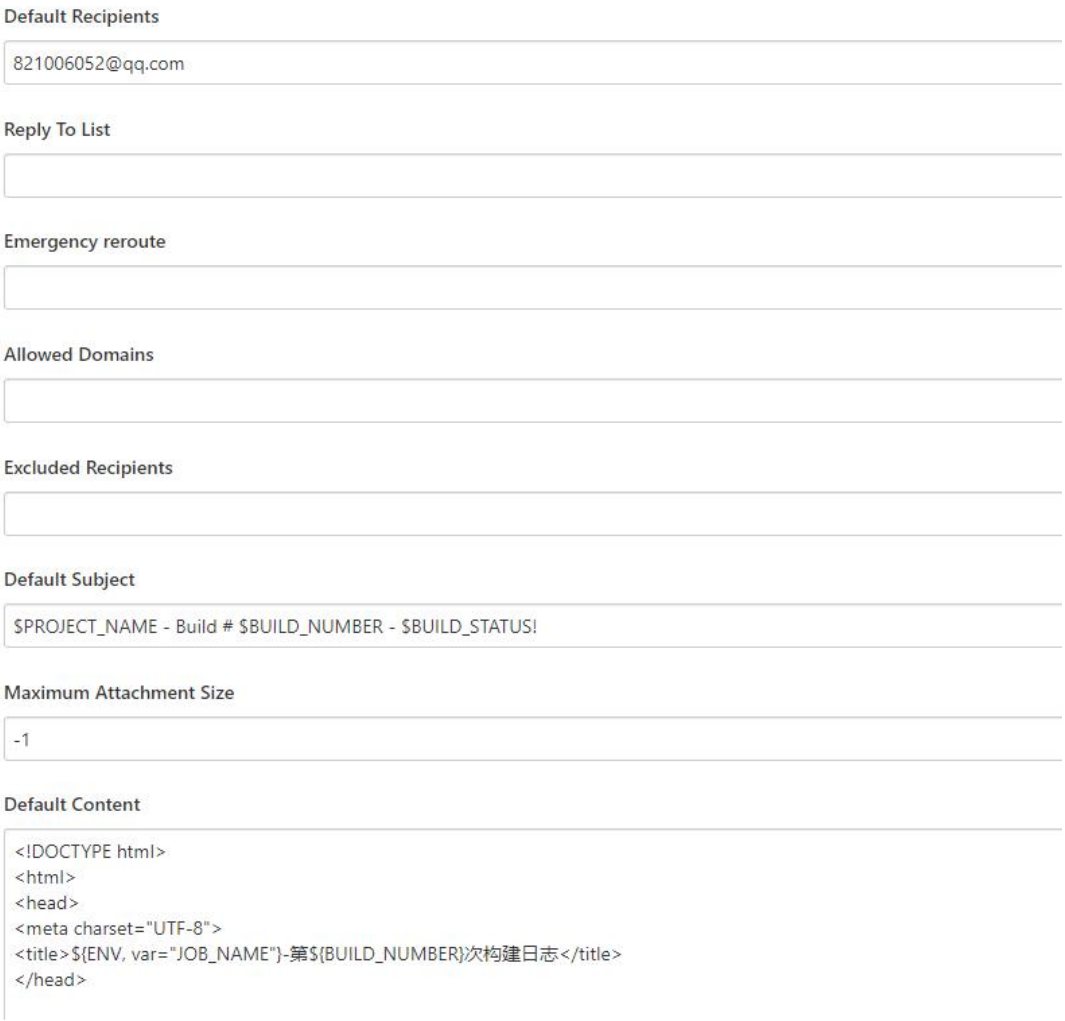
3、配置收件人已经邮箱格式内容

Default Recipients：表示发件人信息；

Reply To List：表示收件人信息；

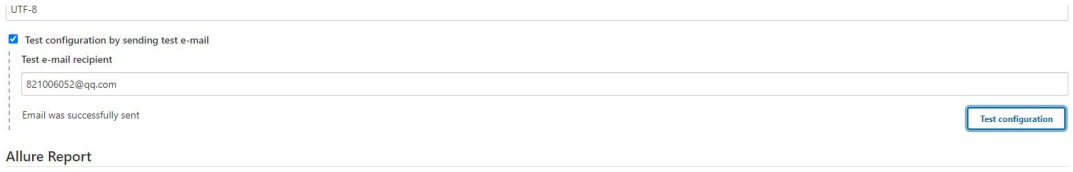
Default Subject：邮件标题，默认配置就行；

Default Content：设置固定正文；



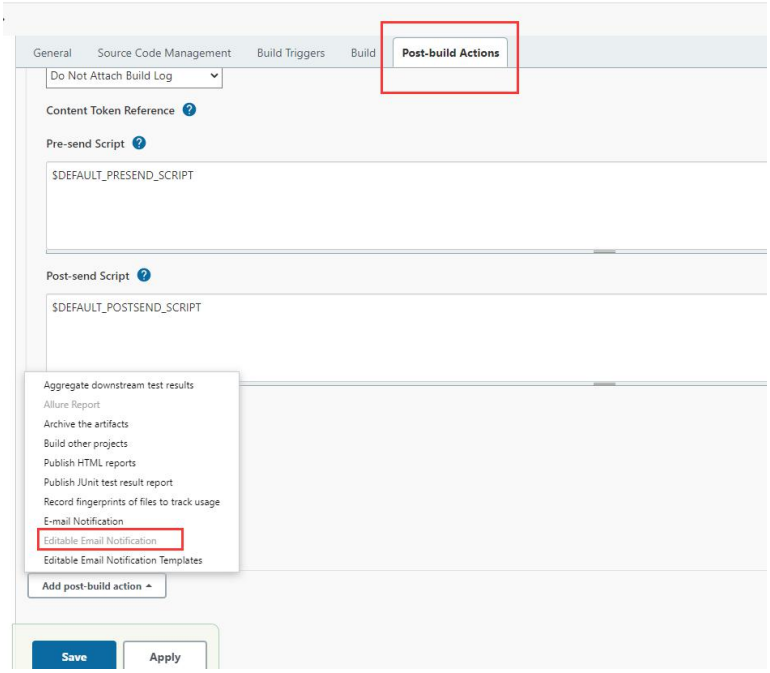
4、继续往下找，找到 E-mail?Notification 参数选项，需要配置 smtp 服务，邮箱后缀，账号，密码。

5、全部都能配置完成后，我们在测试邮箱这里，我们测试下，输入 QQ 后点击发送

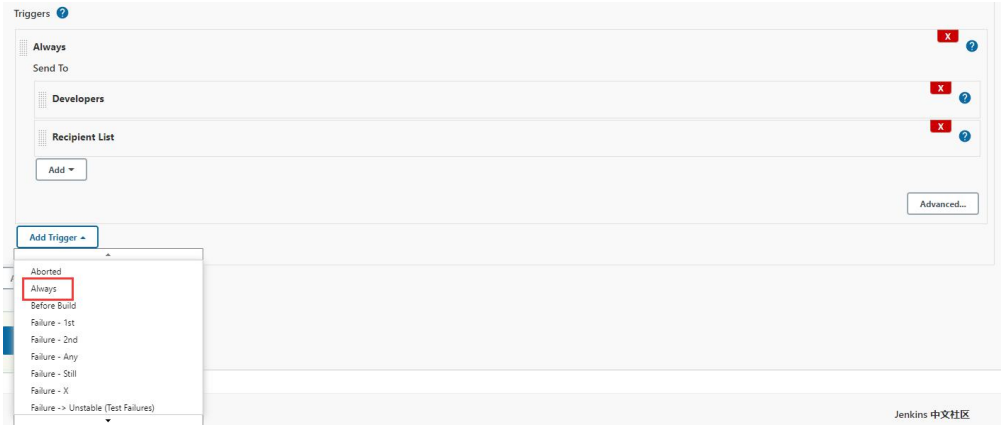


当我们收到下方的邮箱信息的时候就可以确定，邮箱已经配置完成了。

6、你以为现在就完成了吗？当然没有，邮箱配置完成了，我们还需要在项目配置下进行配置构建后的操作发送邮箱。



选择默认收件人和发送人，点击保存就可以，进行构建操作了。



查看邮箱

通过点击构建成功后，就会发现邮箱成功收到了，这里的邮件内容是安静配置的，大家也可以自行配置。


anjing - Build # 10 - Successful!

发件人：收到一份邮件！ <821006052@qq.com>
时 间：2021年11月24日（星期三）下午5：00
收件人：收到一份邮件！ <821006052@qq.com>

邮件可翻译为中文 立即翻译

本邮件由系统自动发出，无需回复！
各位同事，大家好，以下为anjing项目构建信息

构建结果 - Successful

构建信息

- 项目名称： anjing
- 构建编号： 第10次构建
- 触发原因： Started by user 安静
- 构建状态： Successful
- 构建日志： http://localhost:8080/job/anjing/10/console
- 构建 Url： http://localhost:8080/job/anjing/10/

总结

好了，到现在基本上把测试过程经常用到的 jenkins 简单操作介绍了一遍，当然 jenkins 的操作远不止这些，感兴趣的小伙伴们可以自行去学习。感谢您的阅读，希望对您有所帮助。

Postman 各类变量及其作用域详解

◆ 作者：罗狮小钉

Postman 是我们做接口测试的常用工具之一，然而对于刚接触 Postman 的小伙伴们来说，往往对 这款工具支持的各类变量感到迷茫，傻傻分不清这些不同级别的变量都有哪些区别，分别适用于哪些场景。

本次分享将对 Postman 各类变量的创建，执行请求时该变量的作用域，以及不同变量的适用场 景，进行详细讲解。

1. 关于变量那些事

Postman 中的变量并没有什么神奇之处，和任何编程语言一样，变量就是一个占位符，用来保存 执行过程中的初始值，中间值或结果值，这些值也可以通过表达式来生成。我们以简洁的 python 为例：

```
a = "hello"
b = input()
c = a +" "+ b
print(c)
```

以上代码运行后，首先通过 input() 函数，从控制台获取一个字符串，假设我们给到的字符串是“51testing”，那么这个时候 b 指代的内容就是“51testing”；通过 a +" "+ b 表达式进行字符串拼 接后，c 指代的内容就是“hello 51testing”，如此，最后通过 print(c) 运行后，打印输出的内容也就是“hello 51testing”。

我们可以看到以上代码中的 “a，b，c” 三个都是变量，在程序执行过程中指代着

不同的内容，变量 a 对应的是初始值；变量 b 对应的是运行中实时赋予的值；变量 c 对应的是一串表达式运算后的结 果值。这就是变量及其在程序执行中的普遍应用。

2. Postman 中的变量

Postman 中的变量常用于设置请求前的初始值，请求中用于替换固定参数值，请求后用于相关断 言的处理等。此外，不同级别的变量，有着各自所负责的作用域（即，访问范围的限定）。总而言之，这些变量都起到了承上启下，关联上下文接口业务的作用。

Postman 一共提供了 5 种不同类型的变量，对应 5 个作用域：

1. Global —— 全局变量

2. Collection —— 集合变量

3. Environment —— 环境变量

4. Data —— 数据变量

5. Local —— 局部变量，也有称本地变量

下面我们就这 5 个变量逐一介绍。

2.1 Global —— 全局变量

（1）Postman 全局变量及其作用域

全局变量，即通用变量，在 Postman 中所有请求（request）所有用例集合(Collections)等都能访问到。

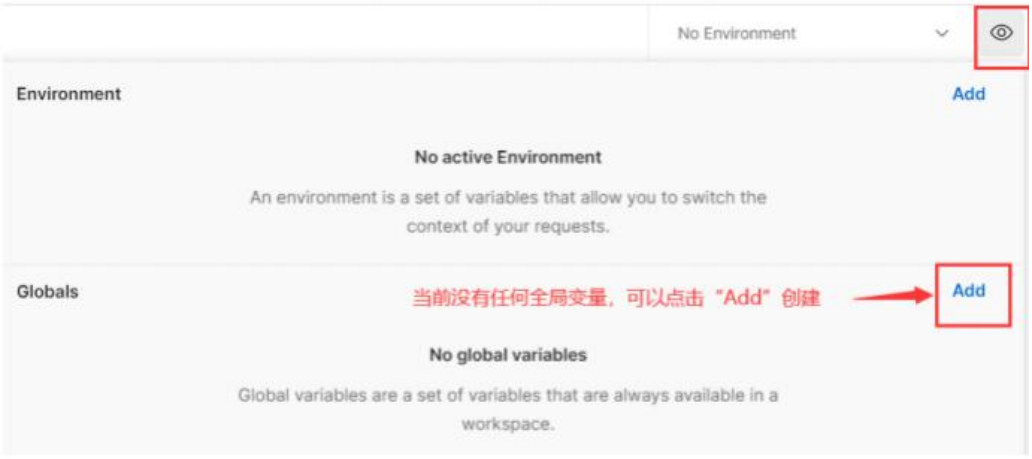
正因为如此，我们需要慎重使用全局变量，因为每一个请求（request），每一个断言脚本（Postman 中设置断言的地方：Tests），每一个前置处理（Postman 中前置处理：Pre-request Script），每一个集合（Postman 用于管理一组业务或上下文相关的接口：Collections），这些 地方都能随意访问且更改全局变量的值。一般而言，全局变量仅适用于快速创建原型设计的需求，非必要，不轻易使用。

（2）Postman 全局变量的创建

在 Postman 中可以通过界面和脚本两种方式来创建全局变量。

【通过界面创建全局变量】

创建全局变量



- 初始化全局变量

输入变量名称，并给一个初始值，这样一个新的全局变量就创建好了。

注意：创建完成后不要忘记 “Ctrl+S” 进行保存；

初始值是该变量的默认值。

【通过 Script 脚本创建全局变量】

创建全局变量并赋值：

```
pm.globals.set("g_value", "51testing 测试圈");
```

在 Postman 前置处理中创建全局变量。

注意：创建完成后不要忘记 “Ctrl+S” 进行保存。

（3）Postman 全局变量应用 Demo

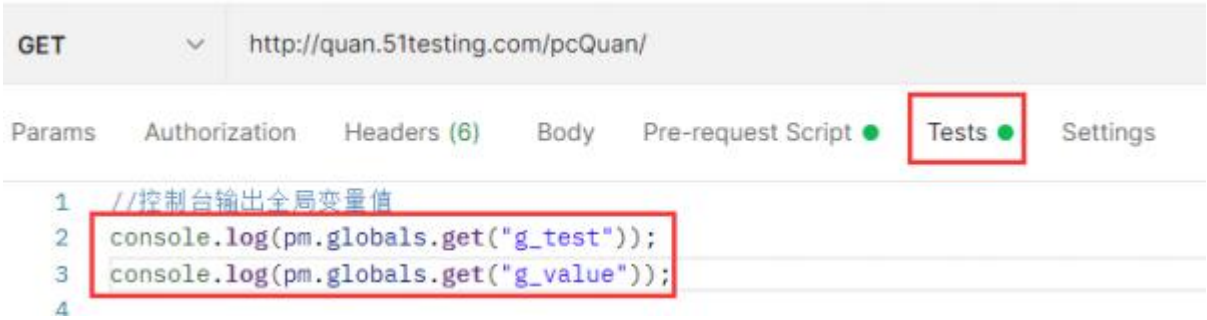
为了演示全局变量的实现效果，我们这里简单以 51testing 测试圈首页为例，发送一个请求，在该 请求的前置处理中，设置如上全局变量 g_value，且更改之前通过界面创建的全局变量 g_test 的 值，通过执行后，将日志打印输出到控制台，一起看一下效果。

• 构建 51testing 测试圈首页请求，在前置处理中创建全局变量

在请求后将全局变量值打印输出到日志。

Tests 区域专门用于处理请求后的一系列操作，例如断言等；这里我们仅在控制台输出全局变量的值。

```
//控制台输出全局变量值
console.log(pm.globals.get("g_test"));
console.log(pm.globals.get("g_value"));
```



执行 “Send” 后，在控制台查看日志输出。

（4）其他关于全局变量的常用脚本

这里再给大家汇总一下常用的全局变量脚本，感兴趣的都可以尝试一下：

```
//创建全局变量并赋初始值
pm.globals.set("全局变量名", "全局变量初始值");
//给全局变量赋值
pm.globals.get("全局变量名");
//通过脚本删除一个全局变量
pm.globals.unset("全局变量名");
//通过脚本清除当前 Postman 中的所有全局变量
pm.globals.clear();
```

2.2 Collection —— 集合变量

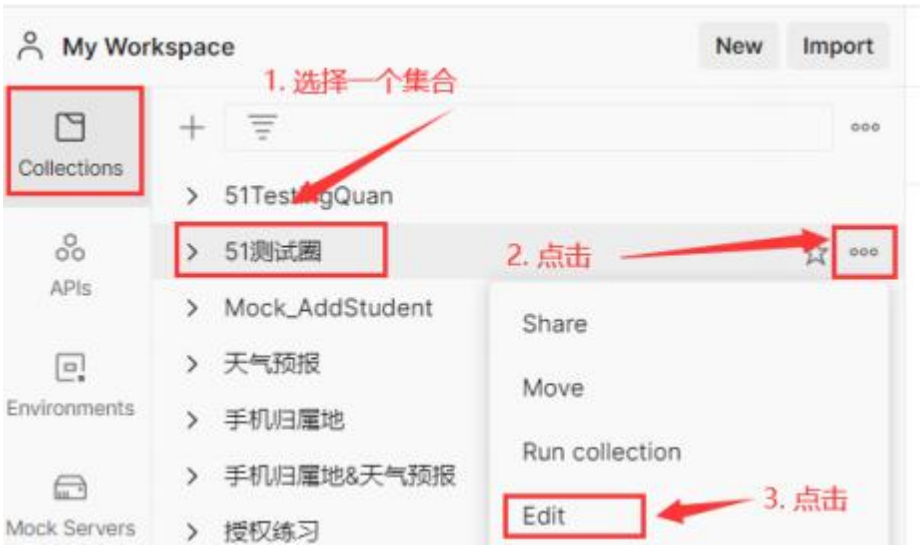
（1）Postman 集合变量及其作用域

Postman 中的 Collection 集合包含了一组 Postman 请求，所以该变量的作用域就是该集合范围。 集合变量只能由该集合内部的请求才能访问，不属于该集合的请求是访问不到的。

（2）Postman 集合变量的创建

Postman 中 Collection 集合变量只能通过界面创建。创建步骤如下：

选择你要创建集合变量的那个 Collection —— 点击该集合旁边的 “。。。” —— 点击 “Edit”。



在界面中选择 “Variables” ，创建一个 Collection 集合变量，变量名为 hot_id，可以不给初始值。

这样，我们就创建了一个名为 “hot_id” 的集合变量。

（3）Postman 集合变量应用 Demo

我们以 51testing 测试圈的搜索为例，展示集合变量的应用。经过接口分析得出如下信息：

请求方式： POST

请求地址： http://quan.51testing.com/searchAll

请求参数： type=XX；search=X；hotId=X；page=X

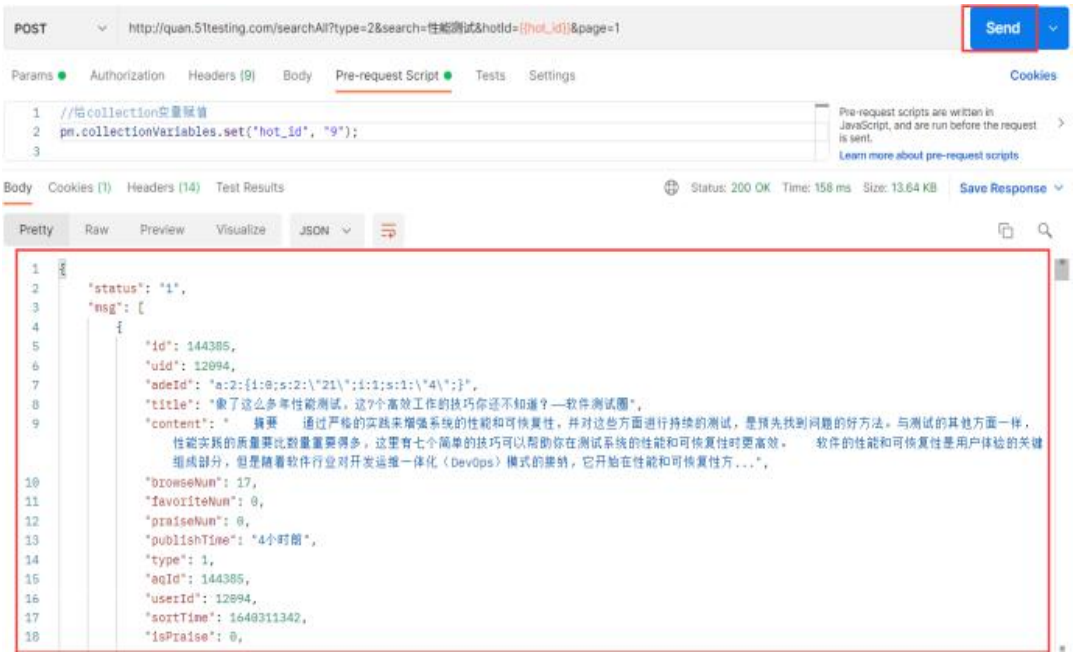
请求样例： http://quan.51testing.com/searchAll?type=2&search=性能测试&hotId=9&page=1 可以看到，虽然是 POST 请求，但这里的请求参数还是以“？”形式拼接在 url 后面，类似于 GET 请求方式。 其中 type 值固为是 2；page 代表页数；搜索关键字 search，对应的 hotId 如下：

Search	hotID
自动化测试工具	11
Selenium	13
Postman	14
性能测试	9
接口测试	8
Jmeter	12
自动化测试	10

我们以搜索 “性能测试” 测试为例，且当前接口中的参数 hotID 引用已经设置好的集合变量“hot_id”：

在 Pre-request Script 中设置集合变量的值为 “9”。

执行请求后，相应结果如下：



可以发现响应的结果，和我们直接通过网页搜索返回的页面信息一致。（以下为直接网页搜索”性能 测试“的结果）

2.3 Environment —— 环境变量

（1）Postman 环境变量及其作用域

环境变量是 Postman 中最为频繁使用的变量之一。环境变量的常见使用场景主要有两个：

1. 通常情况下，在软件研发过程中，对于被测对象往往有几套环境： 例如测试环境，研发环境，客户现场环境等。

而对于接口测试而言，环境变化所引起的仅仅是域名的不同，根据不同的环境会对应不同的域名。

2. 上下游关联业务的接口，即 —— A 接口请求的响应值中，有一部分作为请求参数，作为 B 接口请求参数 的一部分；

也就是说后一个接口的请求参数依赖于前一个接口的响应参数。

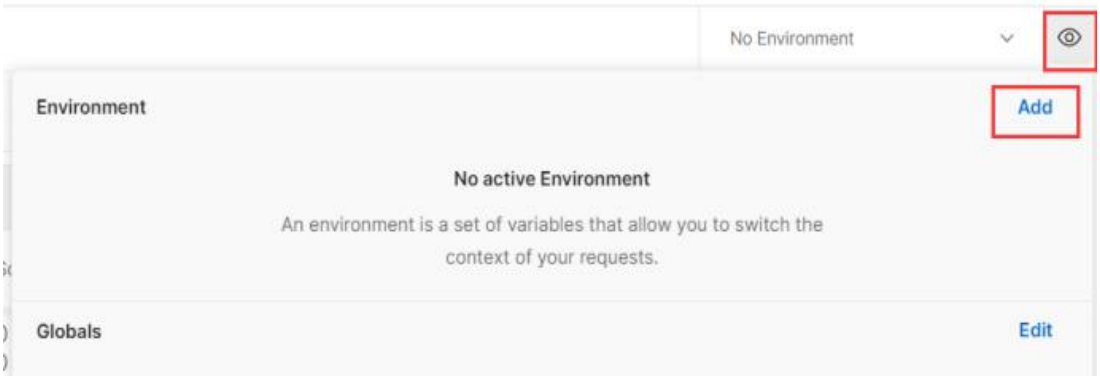
当需要将数据从一个请求传递到另一个请求时，环境变量是一个不错的选择；环境变量的作用域 小于全局变量。

（2）Postman 环境变量的创建

在 Postman 中可以通过界面和脚本两种方式来创建环境变量。

【通过界面创建环境变量】

通过界面创建环境变量的方式和之前创建全局变量类似。



分别创建两套环境变量，一套为 test 环境（用于测试环境），一套为 dev 环境（用于研发环境）。

【通过脚本创建环境变量】 在 Postman 的 Pre-request Script 和 Tests 中，可以通过如下语句创 建访问环境变量

```
//创建环境变量 num, 赋值为 10
pm.environment.set("num",10)
//获取环境变量 num
pm.environment.get("num")
```

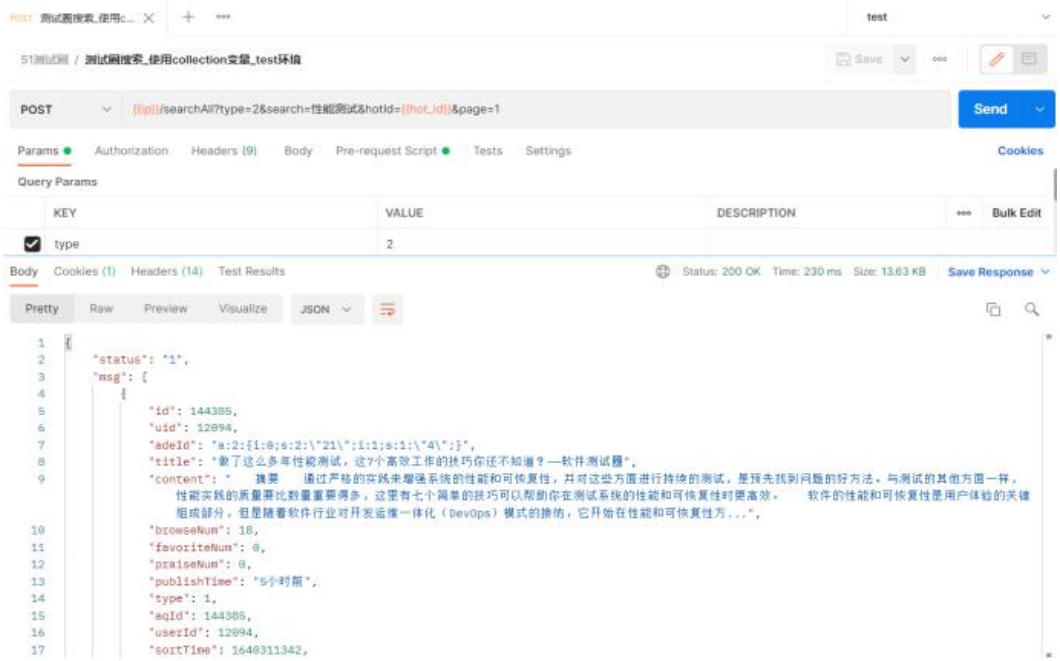
需要注意：这里使用的是 pm.environment 而不是 pm.variables，环境变量 “num” 的作用域仅 限于当前设置的环境，Postman 仅在选定的环境中对变量进行操作。

（3）Postman 环境变量应用 Demo

要将环境变量应用到接口测试用例中，非常方便。假设我们当前是在测试环境下，紧接着上面 51testing 测试圈搜索案例，选择对应的测试环境后，将域名替换成环境变量即可。

选择 test 环境并将域名替换成环境变量。

执行请求后，相应结果如下：



可以看到和我们在 2.2 中获取的结果是一致的，说明环境变量已经成功使用了。大家也可以将 test 环境替换成另一套 dev 环境，自行练习体会一下，加深对环境变量的理解。

2.4 Data —— 数据变量

（1）Postman 数据变量及其作用域

提到数据变量，不得不说数据驱动。对于一个被测接口而言，我们需要准备的测试数据往往不仅 限于一组。

例如最简单的登录，基于等价类边界值的测试设计，我们势必会设计出一系列被测登录账号，难 不成针对每一个登录账号，都构建一个接口请求用例？这明显是不和逻辑的冗余设计。这时就需 要用到数据驱动，即多个相同类型的测试数据，应用于同一个接口测试用例。

在 Postman 中，这些测试数据是通过测试用例集合（Collections）中的集合运行器添加到接口测 试用例中去。需要注意的是，Data 数据变量的来源是用户提供的 JSON，CSV 或 TXT 等 Postman 中 支持的数据文件格式。

（2）Postman 数据变量的创建

以之前 “Collection—— 集合变量”小节提到的搜索接口为例，假设当前有如下待测数据（共 7 组）：

请求方式： POST

请求地址： http://quan.51testing.com/searchAll

请求参数： type=XX；search=X；hotId=X；page=X

请求样例： http://quan.51testing.com/searchAll?type=2&search=性能测试&hotId=9&page=1

search	hotID
自动化测试工具	11
Selenium	13
Postman	14
性能测试	9
接口测试	8
Jmeter	12
自动化测试	10

通过 TXT 文件创建并保存测试数据：

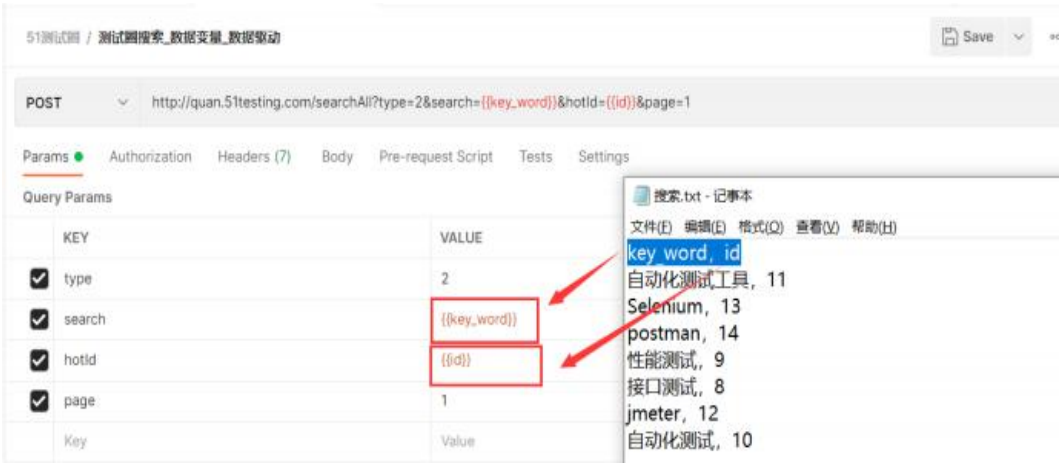
当然，这里的测试数据也可以是 CSV, JSON 等 Postman 支持的其他文件格式。

实现参数化

对于 51testing 测试圈的搜索请求，请求参数是拼接在请求地址中的，我们只需要将对应的请求参 数进行参数化即可。

参数化后如下，将数据文件中对应的名称列表替换请求中的固定数据：

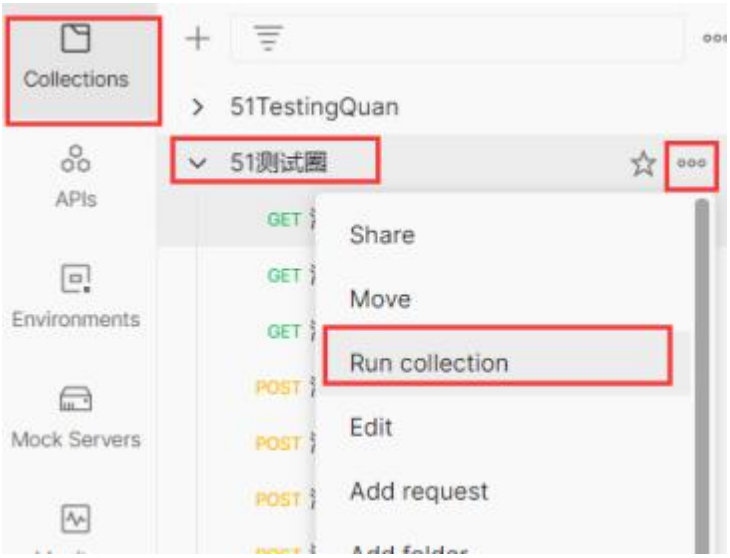
Postman 中的参数化表现形式为{{参数名}}



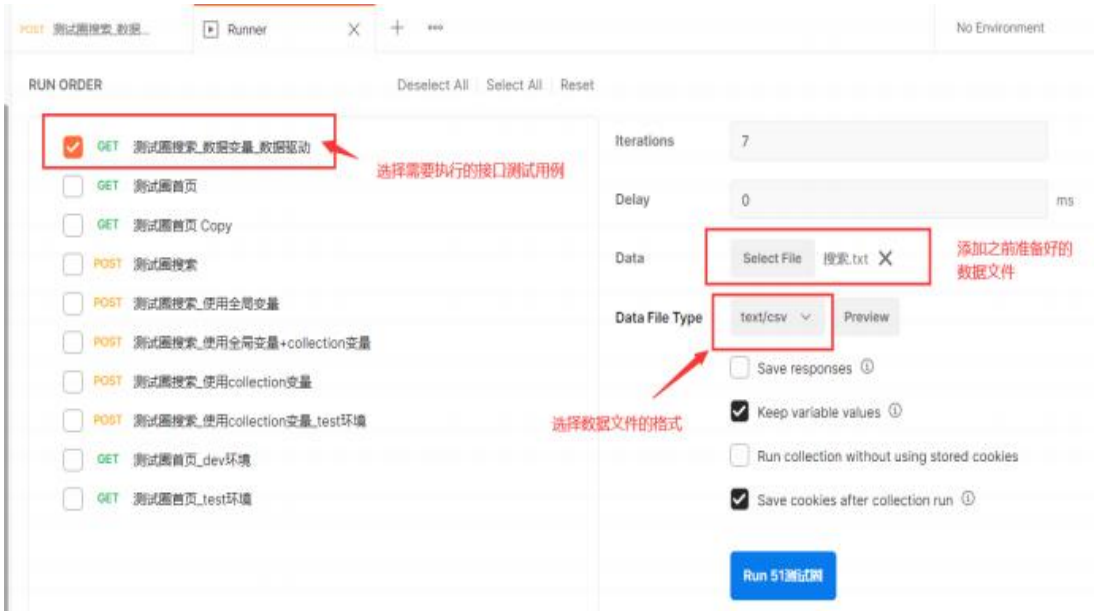
（3）Postman 数据变量应用 Demo

在完成了以上参数化，及数据文件的准备后（别忘记保存），我们就可以投入应用了。

首先，点击这个接口归属的 Collections 集合，选择 Run Collection。

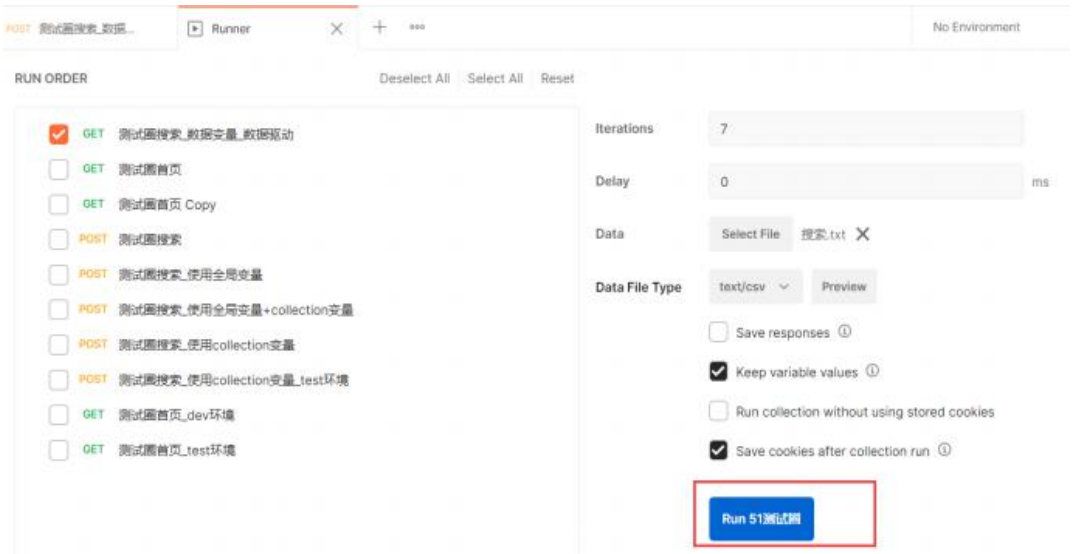


选择待测接口，并添加测试数据文件



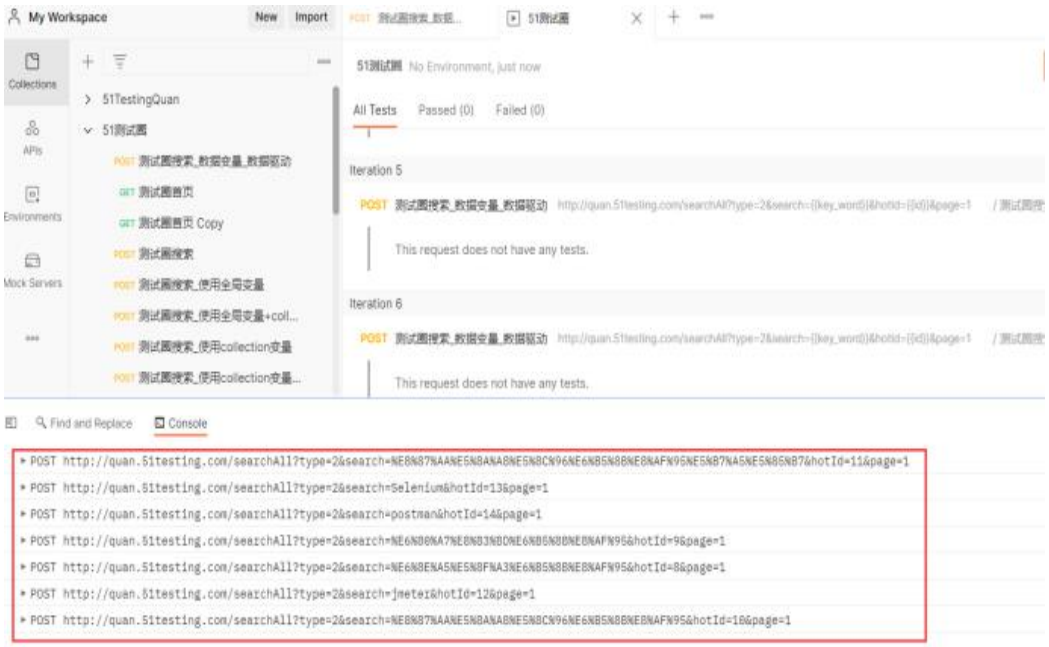
可以预览一下当前的数据文件

执行



执行结果

从 Console 控制台可以看到，一共执行了 7 次，每次都代入了不同的测试数据，由于在请求过程　中，中文字符进行了编码，所以看到的是编码后的效果，英文字符依旧保持原始数据效果不变。



2.5 Local —— 局部变量

局部变量，又称本地变量，仅在特定请求执行的上下文中运用。如果你熟悉其他编程语言 （Java，C，python 等），局部变量等同于编程语言中，函数级别的变量。

局部变量的优先级小于全局变量，小于集合变量，小于环境变量。如果全局/集合/环境变量名， 局部变量名相同的情况下，那么将默认使用局部变量。 和

局部变量创建，需要通过脚本编程的方式，例如：pm.variables.get / pm.variables.set

```
//jsonData 是一个局部变量，Postman 中的脚本是 JS，通过 var 来说声明一个变量
var jsonData = pm.response.json();
//token 是一个局部变量
var token = jsonData.msg.token;
//Postman 中创建局部变量 'testVar '，并赋值
pm.variables.set( 'testVar ',token);
//Postman 中获取局部变量 'testVar '的值
pm.variables.get( 'testVar ');
```

3. 总结

以上我们就 Postman 中不同类型变量的概念，以及各自使用方式，作用域，做了详细梳理，配合 Demo 演示进行应用介绍。

此外有这样几点需要大家注意：

1）Postman 不具备自动保存功能，所以在其中的每一步操作/编辑/更改后，必须记得保存，否则无法生 效；

2）建议在不同的范围内使用不同的变量名称， 以避免混淆；

3）对于环境变量，只有在选定的环境区域中才有效，即在解析变量时，Postman 只会查看选定的环境；

最后，希望通过本文的学习，能够有助于你对 Postman 中变量及其应用的深入理解，有效落实到 今后的接口测试项目中去。

Python 发送邮件

◆作者：测试安静

前言：
每当测试结束后，测试人员都会输出一份详细的测试报告给到领导或者组内人员，那么当我们自动化测试结束后的时候，也可以让其自动发送测试报告。这样领导和组内的成员就能看到自动化测试每次测试的内容了。安静先介绍下如何通过 python 发送邮件，再通过简单的小栗子在自动化

smtplib

smtplib 是属于 python 发送邮件的一个库。其简单的原理是通过 SMTP 的方式来实现发送报告的。SMTP（Simple?Mail?Transfer?Protocol）即简单邮件传输协议,它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式。smtplib 中对其 SMTP 的协议进行了一个封装。其中 email 是用来支持发送文本，图片，和携带附件等功能。

登录邮箱

发送邮件前肯定需要登录邮箱了，这里安静先通过 163 的邮箱进行介绍，登录是通过 smtplib 这个库进行来完成的。这里需要先确认发件人的邮箱是否开通了 SMTP 邮箱权限，可以登录到 163 邮箱中，选择 SMTP/IMAP 中选择开启 SMTP 服务。勾选后进行连接登录。

```
import smtplib
# 创建 smtplib 服务
smtp = smtplib.SMTP()
# 服务器地址
smtpserver = 'smtp.163.com'
# 发送账号
user = 'XXXXXXXX@163.com'
# 发送密码
password = 'xxxxxxx'
# 收件人
receivers = '821006052@qq.com'
# 连接服务器
smtp.connect(smtpserver)
# 登录邮箱账号
smtp.login(user, password)
```

邮件发送类型

邮件发送通过 python 中的 email 的库来实现的，其中 email 发送邮件可以支持多类型，比如纯文本，纯图片，文本加附件等方法，这里安静会一个个进行介绍。

文本发送

发送邮件肯定包含，发送人，收件人，邮件标题，邮件内容等内容，这里 email 中的 Mimetext 的方法可以帮助我们实现发送纯文本内容。

```
import smtplib
from email.mime.text import MIMEText
# 创建 smtplib 服务
smtp = smtplib.SMTP()
# 服务器地址
smtpserver = 'smtp.163.com'
# 发送账号
```

```
user = 'xxxxxxx@163.com'
# 发送密码
password = 'xxxxxxx'
# 收件人
receivers = '821006052@qq.com'
# 邮件标题
subject = '自动化测试报告'
# 发送内容 （文本内容，发送格式，编码格式）
message = MIMEText('这是测试文本内容，自动化测试通过', 'html', 'utf-8')
# 发送地址
message['From'] = user
# 接收地址
message['To'] = receivers
# 邮件标题
message['subject'] =subject
# 连接服务器
smtp.connect(smtpserver)
# 登录邮箱账号
smtp.login(user, password)
# 发送账号信息
smtp.sendmail(user,receivers,message.as_string())
# 关闭
smtp.quit()
```

通过执行后可以发现，QQ 邮箱已经成功的收到了邮件信息。

图片发送

正常发送邮件只需要将邮件全部都复制粘贴到邮件中就行了。但是这里我们通过 python 进行发送邮箱，需要用到 email 中的 MIMEimage 的模块了。这个模块可以帮助我们将我们需要的图片内容添加到邮件中，需要我们将本地的图片导入到 html 中，通过 html 中进行发送，如果你通过链接的形式发送会失败，邮件会识别成恶意链接，从而进行拦截。这里安静这接在上面的代码中进行加入 HTML 格式，将图片嵌套在 html 文本中发送。

```
import smtplib
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
# 创建 smtplib 服务
smtp = smtplib.SMTP()
# 服务器地址
smtpserver = 'smtp.163.com'
# 发送账号
user = 'xxxxxx@163.com'
# 发送密码
password = 'xxxxxx'
# 收件人
receivers = '821006052@qq.com'
# 邮件标题
subject = '自动化测试报告中加入图片'
# 发送内容 （文本内容，发送格式，编码格式）
text = '''
<html>
<head>自动化测试报告中带图片</head>
<body>
<p>
<p><img src="cid:anjing"></p>
<p>
```

```
</body>
</html>
'''
message = MIMEMultipart()
body = MIMEText(text, 'html', 'utf-8')
f = open('123.jpg','rb')
mag = MIMEImage(f.read())
f.close()
# 定义图片 ID 在 HTML 中展示
mag.add_header('Content-ID', 'anjing')
# 添加图片信息
message.attach(mag)
# 添加正文
message.attach(body)
# 发送地址
message['From'] = user
# 接收地址
message['To'] = receivers
# 邮件标题
message['subject'] =subject
# 连接服务器
smtp.connect(smtpserver)
# 登录邮箱账号
smtp.login(user, password)
# 发送账号信息
smtp.sendmail(user,receivers,message.as_string())
# 关闭
smtp.quit()
```

通过执行上面的代码可以看到 QQ 邮箱，已经接收到了邮件信息，打开邮箱清楚的看到，图片已经在文本中添加了。

附件发送

发送邮件需要带附件的情况下，我们可以使用 email 库中的 MIMEMUltipart 模块。

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
# 创建 smtplib 服务
smtp = smtplib.SMTP()
# 服务器地址
smtpserver = 'smtp.163.com'
# 发送账号
user = 'xxxxxx@163.com'
# 发送密码
password = 'xxxxxx'
# 收件人
```

```
receivers = '821006052@qq.com'
# 邮件标题
subject = '自动化测试报告中附件'
message = MIMEMultipart()
body = MIMEText('自动化测试报告携带附件内容', 'html', 'utf-8')
# 添加正文
message.attach(body)
att = MIMEText(open('123.jpg', 'rb').read(), 'base64', 'utf-8')
att["Content-Type"] = 'application/octet-stream'      # 死格式
# filename 表示附件的名称
att["Content-Disposition"] = 'attachment; filename="fujian.jpg"'
# 邮件中添加附件
message.attach(att)
# 发送地址
message['From'] = user
# 接收地址
message['To'] = receivers
# 邮件标题
message['subject'] =subject
# 连接服务器
smtp.connect(smtpserver)
# 登录邮箱账号
smtp.login(user, password)
# 发送账号信息
smtp.sendmail(user,receivers,message.as_string())
# 关闭
smtp.quit()
```

通过执行代码清楚的看到邮件中已经携带了附件内容，并且成功发送了。

zmail 邮件

一些测试同学看到上面这么多代码估计脑袋就大，安静在给大家介绍一种简单方便的发送邮件库 Zmail，这个库的目的就是为了方便发送邮件。但是要注意 zmail 这个库目前只支持 python3 不支持 python2，想必都 2021 年了，没人再用 python2 了吧。

安装：

```
pip  install zmail
```

文本发送

继续从文本发送，先创建一个 zmail 服务，将其发件人邮箱账号以及邮箱授权码（163 设置中的 SMTP 打开）进行连接通过 zmail 服务连接。编辑文本进行发送

```
import zmail
# 发件人
username = 'xxxxxx@163.com'
# 授权码密码
password = 'xxxxxx'
# 创建 zmail 服务
server= zmail.server(username,password)
# 邮件主题
```

```
body = {
    'subject': "自动化测试报告",  # 邮件标题
    "content_text": '这是邮件的文本内容，自动化测试结果',   # 邮件文本
}
# 收件人
receivers = '821006052@qq.com'
# 发送邮件
server.send_mail(receivers,body)
```

通过代码就可以看出来很清楚的就讲邮件内容展现出来，执行代码。成功的收到邮件信息。

图片发送

同样文本发送完成后，继续来我们的图片发送。这里可以通过图片的 base64 的格式加入到 html 的代码中，然后放入到文本中进行发送。

```
import zmail
# 发件人
username = 'xxxxx@163.com'
# 授权码密码
password = 'xxxxxx'
```

```
# 创建 zmail 服务
server= zmail.server(username,password)
html = '''
<p> 这是邮件的文本内容，自动化测试结果 </p>
<img
src=' •••AAD/2Q==''/>
'''
# 邮件主题
body = {
    'subject': "自动化测试报告添加图片",  # 邮件标题
    "content_html": html,   # html 格式
}
# 收件人
receivers = '821006052@qq.com'
# 发送邮件
server.send_mail(receivers,body)
```

通过代码执行后可以发现，邮件成功的收到了并且图片和文本都存在邮件中。

附件发送

通过上面的两个例子这里应该很清楚的就能知道了，我们只需要将附件信息直接写在我们的 body 文本中就行了。

```
import zmail
# 发件人
username = 'xxxxx@163.com'
# 授权码密码
password = 'xxxxx'
# 创建 zmail 服务
server= zmail.server(username,password)
html = '''
<p> 这个邮件中携带附件，自动化测试结果 </p>
<img
src=' •••A
AD/2Q==''/>
'''
# 邮件主题
body = {
    'subject': '自动化测试报告添加附件', # 邮件标题
    "content_html": html, # html 格式
    "attachments": "123.jpg" # 附件
}
# 收件人
receivers = '821006052@qq.com'
# 发送邮件
server.send_mail(receivers,body)
```

通过制定代码后发现，邮件已经成功发送且携带了附件内容。

yagmail

yagmail 也是属于 python 发送邮件的一个库，这种库相比前面两种依旧做了很大的简介，使用更加方便，因为属于 python 的第三方库，我们安装

安装：

```
pip install yagmail
```

发送文本

这里一样先将 yagmail 创建一个服务对象，通过将发件人的账号，授权码进行连接登录。

```
import yagmail
# 发件人
username = 'xxxxxx@163.com'
# 授权码密码
password = 'xxxxxx'
# 创建 yagmail 服务,需要加上服务器地址
server = yagmail.SMTP(username,password, host='smtp.163.com')
# 收件人
receivers = '821006052@qq.com'
text = '这是测试报告内容'    # 报告内容
title = '自动化测试报告'    # 邮件标题
server.send(contents=text,to=receivers,subject=title)
```

通过发现 yagmail 的代码比 zmail 的代码更加简洁了一些,但是整体内容是差不多的。执行代码，发现我们已经将其报告发送成功了。

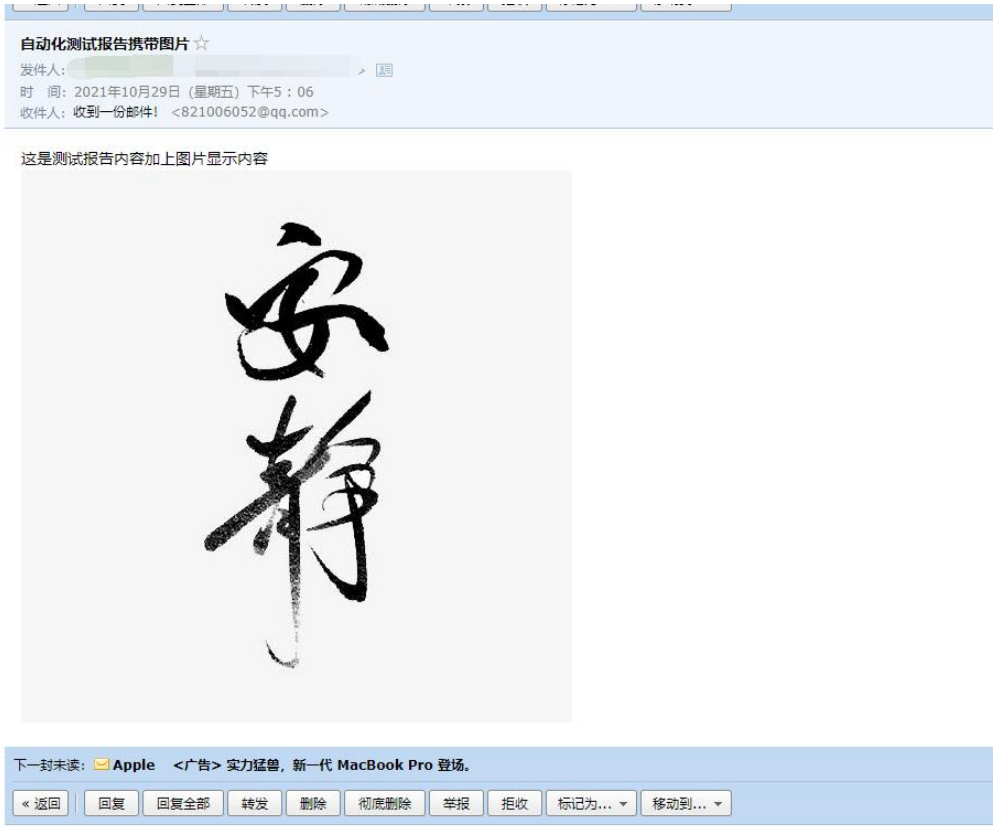
图片发送

yagmail 中携带了发送图片的方法，直接将图片路径方进入就可以了，其中这里需要使用 yagmai.inline 的方法将图片添加到正文中。

```
import yagmail
# 发件人
```

```
username = 'xxxx@163.com'
# 授权码密码
password = 'xxxxx'
# 创建 yagmail 服务,需要加上服务器地址
server = yagmail.SMTP(username,password, host='smtp.163.com')
# 收件人
receivers = '821006052@qq.com'
text = '这是测试报告内容加上图片显示内容'    # 报告内容
title = '自动化测试报告携带图片'    # 邮件标题
img = yagmail.inline('123.jpg')  # 图片地址
server.send(contents=text,to=receivers,subject=title,attachments=img)
```

执行代码发现，我们已经成功的将图片添加到了邮件中。



附件发送

附件形式 yagmail 中也提到了单独的方法，通过 attachments 的方法来添加附件文件。

```
import yagmail
# 发件人
username = 'xxxxxx@163.com'
# 授权码密码
password = 'xxxxx'
# 创建 yagmail 服务,需要加上服务器地址
server = yagmail.SMTP(username,password, host='smtp.163.com')
# 收件人
receivers = '821006052@qq.com'
text = '这是测试报告内容加上附件内容'    # 报告内容
title = '自动化测试报告携带附件'    # 邮件标题
fujian = '123.jpg'   # 附件
server.send(contents=text,to=receivers,subject=title, attachments=fujian)
```

通过执行代码发现，我们已经成功的将邮件携带附件发送成功了。

实战演示

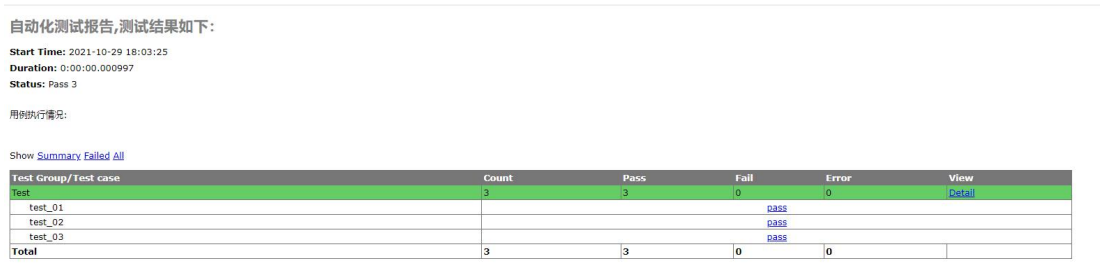
前面已经将 python 几种发送报告的库都做了一个简单的介绍，安静在这里在给大家通过 unittest 生成一份测试报告，然后在通过邮件的形式发送出去来进行演示，方便大家记忆。古话说，好记性不如烂笔头，脑子笨。咱多写几遍，就记住了。

unittest 生成报告并发送报告

安静随便写几个测试用例，然后通过 HTMLTestRunner 的方式生成测试报告。

```
import unittest
import HTMLTestRunner
class Test(unittest.TestCase):
    def test_01(self):
        print('测试用例 1')
    def test_02(self):
        print('测试用例 2')
    def test_03(self):
        print('测试用例 3')
if __name__ == '__main__':
    # 测试报告地址
    fp = open('result.html', "wb")
    # 报告详情
    runner = HTMLTestRunner.HTMLTestRunner(stream=fp,
                                           title=u'自动化测试报告,测试结果如下：',
                                           description=u'用例执行情况：')
    # 实例化
    testunit = unittest.TestSuite()
    # 加载用例
    testunit.addTests(unittest.TestLoader().loadTestsFromTestCase(Test))
    # 执行用例
    runner.run(testunit)
    # 关闭报告
    fp.close()
```

通过执行代码发现测试报告已经生成了。接下来我们需要通过将其测试报告内容添加到邮件中然后在进行发送。



自动化测试报告,测试结果如下:

Start Time: 2021-10-29 18:03:25
Duration: 0:00:00.000997
Status: Pass 3

用例执行情况:

Show Summary Failed All

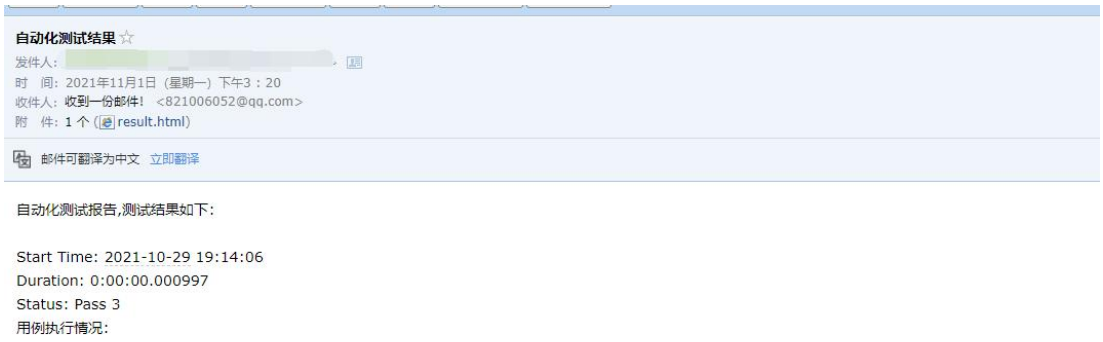
Test Group/Test case	Count	Pass	Fail	Error	View
Test	3	3	0	0	Detail
test_01			pass		
test_02			pass		
test_03			pass		
Total	3	3	0	0	

邮件加入测试报告结果

上面已经通过 unittest 单元测试框架生成了测试报告，接下来就是需要我们通过邮件库来进行发送了，安静这里选择了 yagmail。别问为什么，想用哪个就用那个了。

```
import yagmail
# 发件人
username = 'xxxxxx@163.com'
# 授权码密码
password = 'xxxxxx'
# 创建 yagmail 服务,需要加上服务器地址
server = yagmail.SMTP(username,password, host='smtp.163.com')
# 收件人
receivers = '821006052@qq.com'
with open('result.html', 'rb')as f:
    text = f.read()
title = '自动化测试结果'    # 邮件标题
fujian = r'E:\web\result.html'   # 附件
server.send(contents=text.decode('utf-8'), to=receivers, subject=title,attachments=fujian)
```

通过执行代码后发现测试报告内容已经成功发送了。(这里有个问题就是排版可能发生了一些改变）但是整体的报告内容以及附件全部都发送了。



总结

安静分别简单的介绍了 Python 发送邮件的方法，其中有简单的，也有复杂操作的，但是据图使用哪种就要看个人的喜好了。无论用哪一种，最终都是可以达到我们的最终需求，给领导发送我们的测试报告内容。好了，感谢大家的阅读，希望对您有所帮助。

Selenium 获取动态图片验证码

◆ 作者：米洛

关于图片验证码的文章，我想大家都应该看过类似的文章了。

在我们做 UI 自动化的时候，经常会遇到图片验证码的问题。

当开发不给咱们提供万能验证码，或者测试第三方网站比如知乎的时候，我们就需要自己去识别验证码。

OCR

OCR 是一种图像文字识别的技术，例如图中的验证码，我们用肉眼识别就是 c5s3，但机器可不比咱们肉眼。所以我们要利用 ocr 技术，让我们的 Python 脚本自动通过图片识别出对应的文字。

常见的识别类库

在 Python 中其实有许多识别类库，这里只介绍博主自己实践过的成功率还不错的：百度 ocr。

简单的说，就是百度提供了一个 SDK，让我们传入图片数据，从而拿到识别的结果。ocr 的细节我们不需要关心。

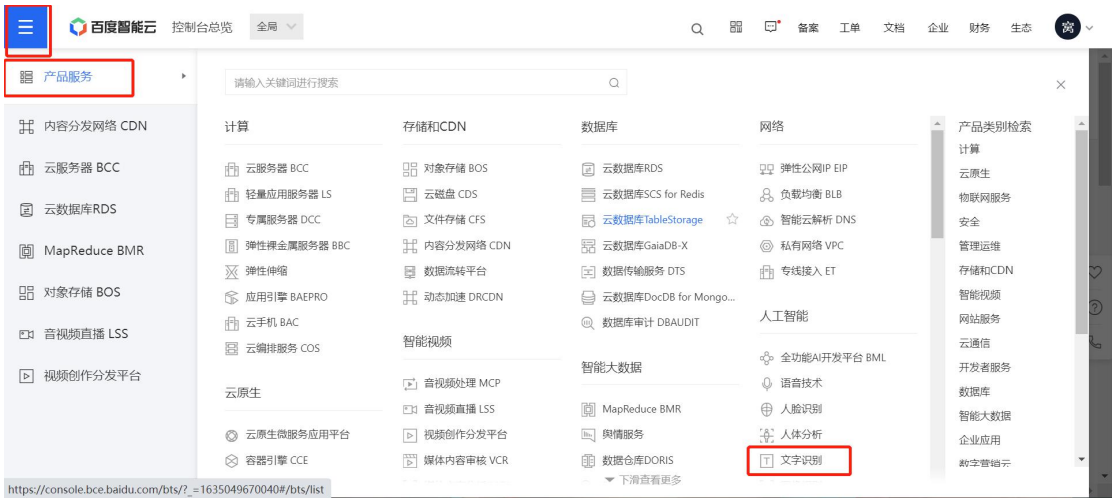
申请开通 OCR

首先我们得有一个百度账号，这个相信大家都有，没有的可以申请一个。

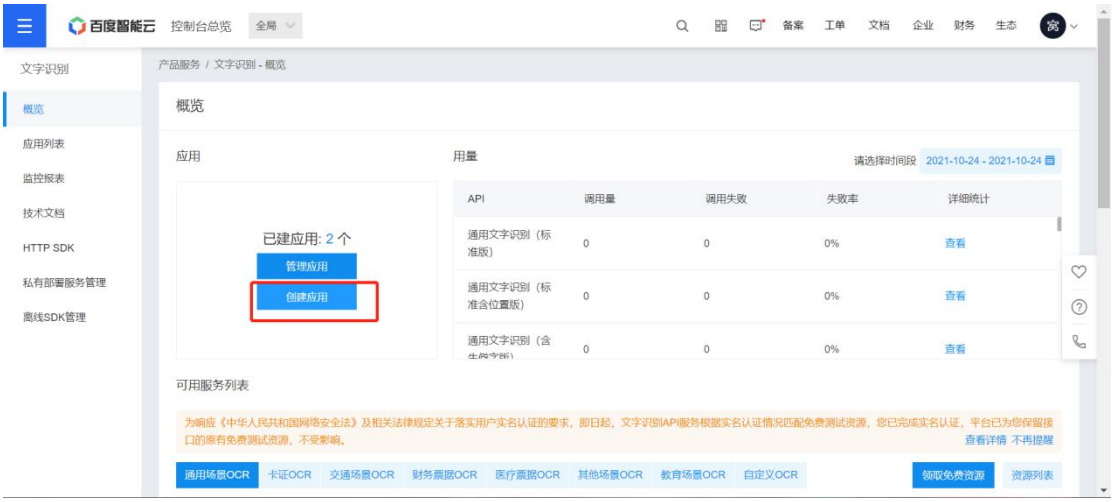
- 登录百度控制台

  进入 https://login.bce.baidu.com/并登录。

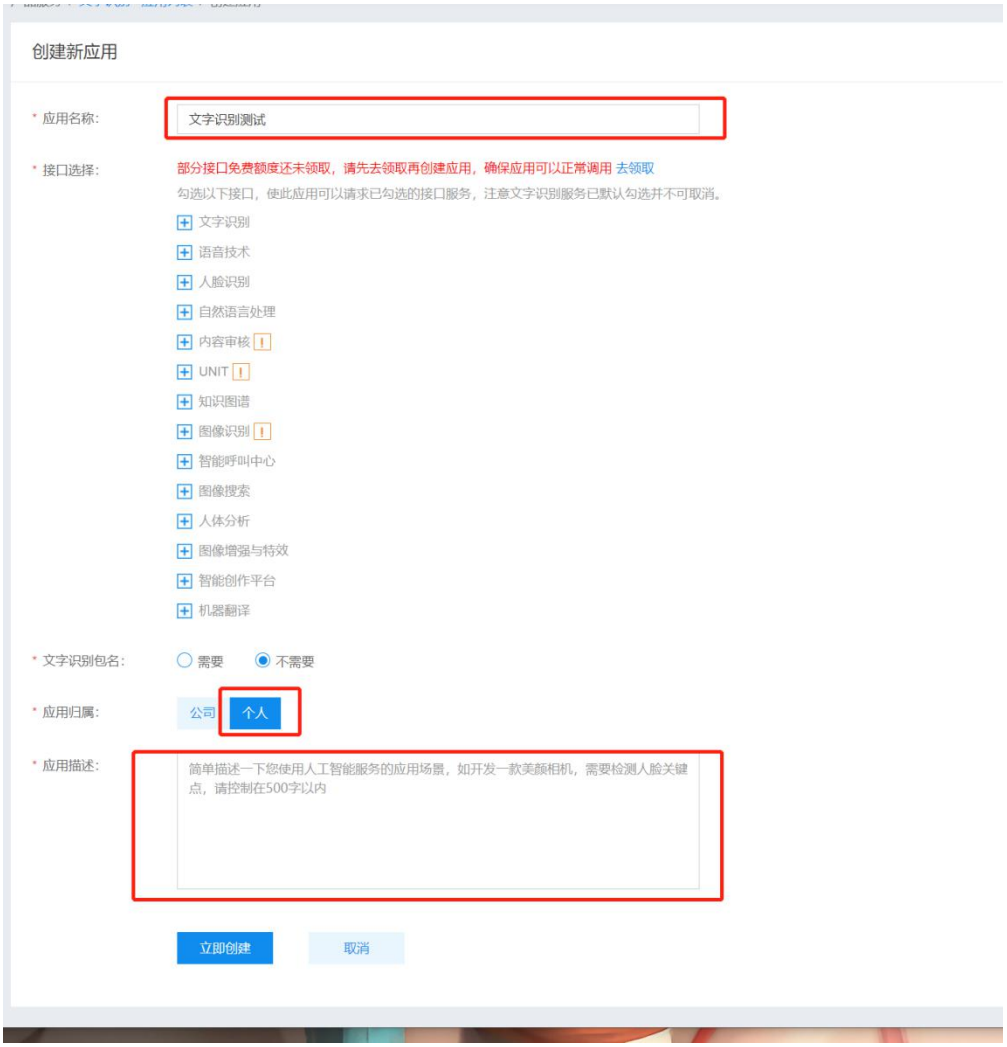
- 选择文字识别



• 创建应用



• 填写相关应用信息



创建好了之后可以看到具体的应用信息，记住这 3 个关键信息。待会会用到。

- appid
- apikey
- secret key

熟悉 OCR 文档

官方文档地址: https://cloud.baidu.com/doc/OCR/s/wkibizyjk

文档会写的比较清楚，简单的说就是通过你的 appid,api key 和 secret key 获取一个 client，接着你就可以调用 client 的 api 去获取图片中的文字了。官方的 SDK 还是比较贴心的。

- 安装 SDK

```
pip install baidu-aip
```

讲完了文字怎么识别，接着就来说说标题中的动态图片验证码。

动态图片验证码

这个概念是我自己命名的，一般来说，我们的一张图片都是对应唯一一个 url 的，比如:

https://yuque.com?image=dshqadiau

(这个地址是我编的)

一般来说 image 字段的值不同，图片也就不同，都是一串随机的或者规律的不重复数据，确保图片不会重复。

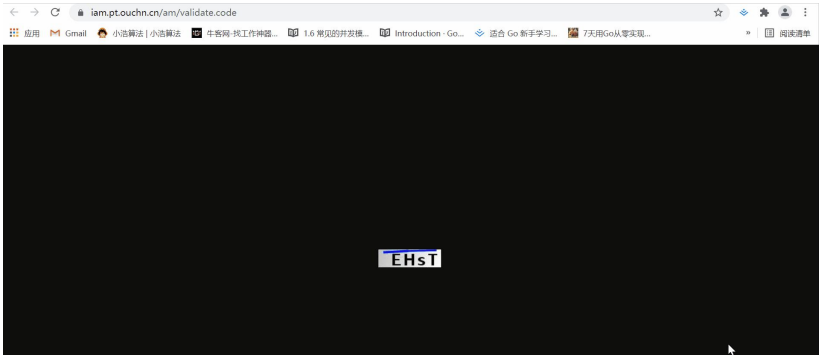
但是博主最近遇到了这样一种情况:

输入一个 url，每次输入，拿到的图片都不一样。

这样就会带来一个很严重的问题，页面上你虽然读取了图片的信息。我们把图片的 url 传递给百度 sdk 的时候，url 由于再次调用，导致图片发生了变化。

比如网站上显示的是: c5s3，调用百度 sdk 的时候，百度会通过 url 读取图片，但再次读取，图片可能变成了 lfew。

不信大家可以看看这个图片地址:



怎么解决呢？

好在百度 sdk，他不仅仅支持 url，还支持图片文件和 base64 的图片数据。我们看看官方文档:

参数名称	是否必选	类型	可选值范围	默认值	说明
image	是	string			图像数据，base64编码，要求base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式
url	是	string			图片完整URL，URL长度不超过1024字节，URL对应的图片base64编码后大小不超过4M，最短边至少15px，最长边最大4096px,支持jpg/png/bmp格式，当image字段存在时url字段失效

再回到 Selenium 里面，我们怎么才能获取到验证码那张图片呢？

思考一下:

1.读取 img 标签的 src，然后下载图片，保存图片文件再转为 base64，很显然这个方法行不通，为什么呢？

因为 img 的 src 属性就是刚才这个 url，你去获取一遍 url，它同样会变化。

2.截图，裁剪出验证码部分，扔给百度去识别可行是可行，但是会不会太复杂了？

如果我只对验证码的 img 元素进行截图，生成 base64 的数据是不是更方便？

其实呢，selenium 作为一款老牌的自动化测试工具，很多方法供大于求了。所以它是有这样的功能的！

Selenium 对指定区域截图

我们都知道，selenium 有一些截图方法。

```
driver.get_screenshot_as_file(filename)
```

但其实，针对元素，也是有截图方法的。

伪代码如下:

```
# 通过 id 获取到图片
img = driver.find_element_by_id("image")
# 调用 WebElement 的 screenshot_as_png 属性方法，获取到 png 的数据，因为百度需要 png
data = img.screenshot_as_png
```

接着我们就可以用这个获取到的图片数据去找百度要答案了！

完整版代码:

```
from aip import AipOcr
from selenium import webdriver
client = AipOcr("你的 appid", "你的 app_key", "你的 secret_key")
```

```
driver = webdriver.Chrome()

driver.get("https://iam.pt.ouchn.cn/am/UI/Login")

img = driver.find_element_by_id("kaptchaImage")

data = img.screenshot_as_png

res = client.basicGeneral(data, {})

print(res)
```

可以看到，只识别到了 CFX，而且图片没有继续变化了。

毕竟文字识别是从图片里面找文字，而且文字会有一些横线这样的干扰，所以如果一次不行，可以多试几次。

思路就是写一个 while 循环，不断尝试去识别验证码并登录，接着判断是否登录成功，没成功则重复上一个步骤。

以我个人的经验，一般 1-10 次就可以成功。

好了，以上是博主简单替大家尝试一下 UI 自动化过程中对于验证码的识别。主要重点在于验证码的识别和对部分区域截图。

从国企文员到大厂测试，说说我的六年

◆作者：咖啡猫

前言：
今年是笔者本科毕业的第 6 年，在接近三十而立的年龄里，回首自己从毕业到现在的职业生涯，可以说是一波三折。趁着自己现在有时间，就做一个复盘和总结，分享给曾经和我一样迷茫的朋友，希望能够带给你一些启发。

一、考研失利，“捡到”一个国企的 offer

笔者是某末流 211 大学电子信息工程专业科班出身的，大四那年有些不知道“天高地厚”地拒绝了本校保研，准备跨考复旦大学的金融专业研究生，结果当然是华丽丽地当了一回“分母”。话说那时候考研还不是很卷，如果我坚持考自己专业，应该也是可以上岸一所不错的学校的，但是造化弄人.....于是乎，完美地错过了当时的秋招，只能急匆匆地追赶春招的步伐。地球人都知道春招无论是从岗位数量还是岗位质量上都远不及秋招。何况笔者本身专业水平并不强，大学期间也没有什么拿得出手的项目竞赛经历，再加上考研失利确实让我沮丧了很久，所以找工作的进程并不顺利。记得是当年的六月初了，周围的人都在准备毕业答辩、拍毕业照、各种散伙饭，唯独我还在为工作的事到处面试奔波。

突然有一天，辅导员私信我说：“有一家国企要招一个技术支持，因为前面拿 offer 的同学临时决定出国留学，所以多了一个 HC 出来，面试估计不会很难。你要不要去试试？”我赶紧应了下来，第二天就参加这家公司的面试，所幸面试题目也真的不难，而且当时很多同学都已经签约了，所以竞争并不大，再加上 211 学校牌子的加成吧，我很顺利地拿到了 Offer 并签了三方协议。对于这个“捡来”的 offer，我真的心存感激，如果没有它，我真的可能会毕业就失业了。

【经验教训】考研一定不要眼高手低，盲目跟风跨专业。

二、初出茅庐，开始了“打酱油”的日子

其实直到签约我都不太明白技术支持到底是干什么的，只是想着好歹是个国企，有个工作总比毕业就失业强吧。接下来就水到渠成地写论文、答辩、毕业。入职国企是七月中旬，我们一行 50 多个应届生参加了为期一个月的岗前培训，培训结束后才会分配具体的岗位和工作地点。这里不得不说，国企的培训制度还是很完善的。岗前培训的内容丰富多彩：军事训练、企业文化、职场礼仪、专业理论知识、专业实训.....也是在这一个月的培训里我才明白了技术支持这个岗位主要的工作内容是：客户项目方案的规划设计、公司软硬件系统的安装实施、系统调试、故障排查、售后答疑、客户培训等等。很有特色的一点就是这个岗位要经常出差，一年基本有 250 天都在客户现场。当时初出茅庐的我，似乎并不排斥需要出差的工作，甚至很喜欢。因为出差可以到处玩。

但是培训结束后岗位分配的结果却打碎了我的“美梦”---50 多个应届生，好像只有我被分配到了北京的服务运营岗，其他人都是全国各地的技术支持岗。服务运营岗的主要工作内容就是：制定客户服务的政策和流程；审批客户服务合同的内容是否合规；项目招投标文案的撰写；服务文化的宣传等等。总而言之，是一个类似“文员”的岗位，除了写招投标文案的时候有些忙，其他时间几乎都很闲，线上审一下合同、打几个电话、回复几封邮件一天的时间就晃过去了。也许这种工作对于有家庭有孩子需要照顾并且事业心不强的职场妈妈来说，是个不错的选择，因为工作稳定轻松、待遇一般、福利齐全.....但是对于工科生的我来说，真的很难熬：明明很闲，却还要在工位上呆上一整天；一年到头，感觉什么都没学到，没有任何成就感；工资在帝都也不过勉强果腹的感觉.....于是，这份工作我干满一年后，年轻气盛的我果断“裸辞”去寻找诗和远方了。

【经验教训】工作不易，千万千万不要裸辞，尤其是自己能力还很弱的时候。

三、误入外包干测试，在人生低谷积攒能量

裸辞后，我先是和几个大学同学去了一趟云南旅游，然后又开始到处投简历。果然裸辞都是没有好下场的，当时的我虽然还很年轻，但是由于只有一年打酱油的文员工作经历，所以工作找的很困难，连续一个月连面试机会都没有。后来我就把工作地点改成了全国，终于有一家杭州的外包公司给我打了电话，简单地问了一下我过往的工作经历后，他们就给我发了 OFFER 并且将我派到 HW 项目上做测试。

HW 作为一家业内知名的大公司，有着非常完善的测试流程和规范，让我这个测试小白也能在这样的大框架下很快地上手。刚开始的时候，师傅让我照着产品手册搭建测试环境，顺便熟悉一下产品的业务和功能。并且每次开需求评审会、项目沟通会之类的都会叫我参加。在熟悉了产品功能后，师傅开始让我参与其中一个小模块的测试工作，需要自己写用例、执行用例、管理 BUG 等。那时候的我像一个上紧发条的机器，白天在公司上班完成领导的任务，晚上自己学习软件测试的基本知识：比如软件开发的流程是什么、什么是软件测试、什么是 BUG 闭环、如何设计测试用例、如何撰写测试报告、什么是需求评审、如何同开发沟通等等。除了测试知识外，我也自学一些必备的技术知识：比如如何安装数据库，数据库的 SQL 语句怎么写，LINUX 的文件结构是什么，主要的 shell 指令有哪些等等。在当时，我的学习渠道主要是自己购买书籍以及关注一些测试方面的网站和公众号，也就是从那个时候开始我关注了 51test 公众号，公众号每天都能推送几篇硬核技术文，让测试小白的我大开眼界。工作中，遇到不会的我也会主动请教组内的同事，这样做了半年后，我基本上是一个合格的功能测试人员了，给我一个功能模块，我能很快地学习它的业务，搭建测试环境，编写用例，执行测试，管理 BUG，输出测试报告。但是那个时候，提交 BUG 只局限于把复现步骤写清楚并且把报错截图发给开发，其他的都等着开发去解决就好了。

直到有一次我和甲方（也就是 HW）的一个正式测试工程师出差去客户现场做验收测试，我看到甲方工程师做测试的时候不仅是简单的点击鼠标，而是会运用各种调试抓包工具抓取 Log，分析定位问题，找到问题大概的原因后，再反馈给相应的开发。这时，我知道我离大厂的差距还很远。于是乎，我开始主动请缨要求从事一些比较复杂的工作，并且也慢慢熟悉那些调试工具的使用，了解产品背后的技术原理、系统内部的结构、各模块之间的通信协议等等。此后在测试中再发现问题，我就不是简单地截图丢给开发了，而是学着和开发一起抓 log，复现 bug 最终定位问题所在，然后辅助解决问题。期间，我还自学了性能测试工具 loadrunner 的使用，以及持续集成工具 Jenkins。就这样在外包公司又干了半年。

但是新的瓶颈又来了，毕业两年还是一个外包公司的“点工”，接触不到公司的核心技术，做的产品都是边角料的内部系统。记得有一次大学同学聚餐，有个在大厂做 JAVA 开发的男生说自己在自学机器学习神经网络的东西，我连忙问了一下“什么是神经网络啊？”我才意识到大厂正式工和外包工的差距不仅是工牌，而是眼界和格局。那时候我

就暗下决心：最多再过一年，我一定要去大厂。

【经验教训】持续学习才能成长，阳光总在风雨后。即使在外包公司或者小私企，工作待遇工作内容工作环境都不太尽如人意的情况下，也要持续学习丰富自己。毕竟机会总是给有准备的人的。

四、念念不忘必有回响，终于等到了大厂

自从立下了要去大厂的目标，我就开始在各类招聘网站上搜集大厂的招聘信息，尤其会对照里面的招聘要求找差距。这时候我发现大厂很少招专门的功能测试人员，他们更倾向于招聘全栈测试人员。所谓全栈测试人员指的是既要做功能测试、性能测试，还要参与自动化测试方案与效率工具链的开发。并且有些具体的岗位还要求熟悉大数据、安卓、IOS 平台，掌握一门或多门主流编程语言。　　瞧瞧，大厂的高薪果然并不是那么好拿的啊。做测试不仅要会找 BUG，还要会编程，懂架构熟悉平台。显然我在外包公司的工作内容已经不能满足大厂的招聘要求了。这时候我购买了网课，跟着老师学习 java+selenium 自动化测试框架；又购买了一些大数据方面的书学习大数据的主流技术；与此同时，我找到领导要求从功能测试组换到自动化测试组。就这样半工半学了半年多，我开始给大厂投简历。我第一次面试是 TX，当时挂在终面白板 coding 环节了，那是一道中等难度的去重算法题，由于我之前一直在学习框架的原理、基本业务代码的编写，忽略了算法的学习，所以挂了。自那之后我开始在力扣上刷算法题，花了两三个月的时间基本上把基础题和中等难度题都刷了一遍，又找大厂的同学帮忙内推，这一次果然得到了幸运之神的垂青，我顺利通过面试拿到了 offer。

如今入职大厂已经三年有余，在一线大厂的互联网测试团队，测试工程师需要对整个产品质量负责。具体说来就是从需求初审、需求终审、测试用例的设计，测试环境的搭建、测试用例的执行，UAT 验收，回归验收，项目上线、线上监控、ABtest 等整个流程都会参与其中，并辅之以接口自动化测试和 CI/CD 持续交付等来提高测试效率。UI 自动化因为项目变动频繁且考虑到 ROI，我所在的组只做了核心功能的用例。线上监控和日志平台也是用的自研的系统。另外各种环境部署都配合 Docker，也慢慢由 Mesos 迁移到 K8S。

你要问我在大厂和外包最大的区别是什么？我想应该是平台和环境吧。以前在外包

大家都只专注于自己手头的业务测试，基本上是“工具人”的状态，由于项目时间紧张，除法自己利用业余时间学习，否则外包公司很少会给你职业发展上的指导。而大厂给了你更大更好的平台，周围的同事素质也都很高，和他们交流碰撞能有很多新的思路和想法冒出来，并且基于大厂的业务需求和海量数据，你可以让这些想法真正地落地，收获职业上的那种成就感。

【经验教训】锚定目标不放松，功夫不负有心人。在大厂这几年，我除了收入的提高，专业上的提升，还顺利考上了 985 高校的非全日制研究生，也算是了了一桩心愿吧。这些都是曾经在国企当文员或者在外包公司干测试时，想都不敢想的事情。所以，我还是建议同学们有机会去大厂看看！

从端口扫描开始，Python 带你入门安全测试

◆ 作者：罗狮小钉

今天我们聊一聊网络端口扫描那些事，以及 Python 如何助力于端口扫描。无论今后你是否从事网络安全或者安全渗透测试一职，只要你在测试，测开领域行走着，奔跑着，“网络端口扫描”将是你知识技能储备中不可或缺的一角。

1. 闲聊：端口扫描那些事

端口扫描可视为一种监控技术，用来依次定位某个指定主机上可用的，开放的端口。一般而言，这类技术常见于网络管理员，渗透测试人员甚至黑客等人的日常工作实践中。这类人员可以根据当前工作任务需求，配置端口扫描器，最大程度地获取目标系统中的信息。

通常经过对目标主机端口扫描后，可以获取的信息主要包括：

（1）目标主机上开放的所有端口。

（2）在每个端口上运行的服务及其相关信息。

（3）目标主机的操作系统和 MAC 地址信息。

我们可以把端口扫描想象成一个小偷悄悄进入一间房子（目标主机可视为房子），检查每个门窗是否关好（端口可视作门窗），看看哪些门窗是打开的（哪些端口是开放的）。

在互联网中，用于通信的是 TCP/IP 协议，该协议套件由两个协议组成，即 TCP 和 UDP。这两种协议都有 0 到 65535 个端口。这些端口按范围可分为如下三类：

（1）系统或公认端口： 0 —— 1023

（2）用户或注册端口：1024 —— 49151

（3）动态或私有端口： 49151 —— 65535

可见系统服务器上的端口众多，但实际上常用的端口才几十个，由此可以看出未定义的端口相当多。为了安全起见，通常建议关闭系统中不必要的端口，不对外开放。如此一来，我们就有超过 65000 个端口需要关闭。

2. 使用 Socket 进行端口扫描

在上一篇文章中《测试人必会 —— Python 带你上手 Web Socket》，介绍了套接字相关知识。

现在，我们将使用套接字构建一个简单的端口扫描器。

使用套接字进行端口扫描器的构建，通过 Python 脚本实现如下：

```
from socket import *
import time
startTime = time.time()
if __name__ == '__main__':
    target = input('需要扫描的主机: ')
    t_IP = gethostbyname(target)
    print('开始扫描主机: ', t_IP)
    for i in range(100, 450):
        print('当前正扫描端口号：'+str(i))
        # 创建 socket 对象
        s = socket(AF_INET, SOCK_STREAM)
        # socket.connect_ex(IP,port)，如果端口连接成功 则返回 0
        conn = s.connect_ex((t_IP, i))
        if (conn == 0):
            print('端口 %d: 处于开放状态' % (i,))
        s.close()
print('扫描共花费的时间:', time.time() - startTime)
```

运行上述脚本后，首先会提示输入目标主机名，这里可以输入你想要扫描的任何主机名。

值得一提的是，未经授权下的端口扫描可能被视为违法行为。

所以在未经服务器或计算机所有者，明确书面许可的情况下，不应对任何网站/IP 地址执行端口扫描程序。

在自行练习实践的情况下，强烈建议在本地主机或自己的网站上执行端口扫描。

以上程序我们仅以自己本机为例，进行实践展示，为了节省时间，我们仅将扫描的端口范围设置在 100 到 450;

扫描结果如下：

```
需要扫描的主机: 127.0.0.1
开始扫描主机: 127.0.0.1
当前正扫描端口号：100
当前正扫描端口号：101
当前正扫描端口号：102
当前正扫描端口号：103
 ......
当前正扫描端口号：135
端口 135: 处于开放状态
当前正扫描端口号：136
当前正扫描端口号：137
当前正扫描端口号：138
当前正扫描端口号：139
当前正扫描端口号：140 .....
当前正扫描端口号：439
当前正扫描端口号：440
当前正扫描端口号：441
当前正扫描端口号：442
当前正扫描端口号：443
当前正扫描端口号：444
当前正扫描端口号：445
```

```
端口 445: 处于开放状态
当前正扫描端口号：446
当前正扫描端口号：447
当前正扫描端口号：448
当前正扫描端口号：449
扫描共花费的时间: 700.8146049976349
Process finished with exit code 0
```

输出显示，在 100 到 450 的端口范围内（不包括 450），经过 Python Socket 脚本创建的扫描程序，扫描后发现两个端口（端口 135 和 445）是开放的。读者朋友们也可以更改此范围，检查其他端口的开放情况。

3. 使用 Ping 进行端口扫描

Ping 扫描可以在特定网络范围内，查找多台机器其端口的可用性。假设我们有一个待测 IP 地址列表，通过 Ping 扫描（即操作系统的 ping 命令），可以逐个扫描列表中的所有 IP 地址，但通过命令行的方式逐一扫描 IP 地址是非常耗时的。

Python 给我们提供了解决方案，简洁的 Python 脚本就能帮我们实现 Ping 扫描，查找实时主机中端口的可用性。

具体实现脚本如下：

```
import os
import platform

from datetime import datetime
net = input("需要扫描的主机: ")
# IP 地址拼接
net1 = net.split('.')
a = '.'
net2 = net1[0] + a + net1[1] + a + net1[2] + a
# 指定待扫描的端口号范围
start_port = int(input("请输入起始端口号: "))
```

```
end_port = int(input("请输入最后一个端口号: "))
end_port = end_port + 1
oper = platform.system()
print("当前操作系统是：",oper)
# 不同的操作系统，执行对应的 Ping 命令行
if (oper == "Windows"):
    ping1 = "ping -n 1 "
elif (oper == "Linux"):
    ping1 = "ping -c 1 "
else:ping1 = "ping -c 1 "
# 获取系统当前时间
t1 = datetime.now()
print("开始扫描:")
for ip in range(start_port, end_port):
    addr = net2 + str(ip)
    comm = ping1 + addr
    response = os.popen(comm)

    for line in response.readlines():
        print("当前端口情况如下：")
        print(line)
        # TTL 是发出信息再返回的时间
        if (line.count("TTL")):
            print(addr, "--> Live")
            break
    print("当前端口扫描完毕")
    print("====================================")
t2 = datetime.now()
total = t2 - t1
print("扫描共花费的时间: ", total)
```

通过将 IP 地址范围拆分为多个部分来选择要 ping 扫描扫描的 IP 地址范围。

运行上述脚本后，首先会提示输入目标主机名，这里为了安全起见，还是以我们本机为例"127.0.0.1"，然后分别输入起止端口号，为了便于演示和节省时间，这里我们仅扫描 130-135 端口，进行查看。

扫描结果如下：

```
需要扫描的主机: 127.0.0.1
请输入起始端口号: 130
请输入最后一个端口号: 135
当前操作系统是： Windows
开始扫描:
当前端口情况如下：

当前端口情况如下：
正在 Ping 127.0.0.130 具有 32 字节的数据:

当前端口情况如下：
来自 127.0.0.130 的回复: 字节=32 时间<1ms TTL=128
127.0.0.130 --> Live
当前端口扫描完毕
==================================
当前端口情况如下：

当前端口情况如下：
正在 Ping 127.0.0.131 具有 32 字节的数据:

当前端口情况如下：
来自 127.0.0.131 的回复: 字节=32 时间<1ms TTL=128

127.0.0.131 --> Live
当前端口扫描完毕
==================================
当前端口情况如下：
```

```
当前端口情况如下：
正在 Ping 127.0.0.132 具有 32 字节的数据:
当前端口情况如下：
来自 127.0.0.132 的回复: 字节=32 时间<1ms TTL=128 127.0.0.132 --> Live
当前端口扫描完毕
====================================
当前端口情况如下：

当前端口情况如下：
正在 Ping 127.0.0.133 具有 32 字节的数据:
当前端口情况如下：
来自 127.0.0.133 的回复: 字节=32 时间<1ms TTL=128 127.0.0.133 --> Live
当前端口扫描完毕
 ====================================
当前端口情况如下：

当前端口情况如下：
正在 Ping 127.0.0.134 具有 32 字节的数据:
当前端口情况如下：
来自 127.0.0.134 的回复: 字节=32 时间<1ms TTL=128 127.0.0.134 --> Live
前端口扫描完毕
 ====================================
当前端口情况如下：

当前端口情况如下：
正在 Ping 127.0.0.135 具有 32 字节的数据:
当前端口情况如下：
来自 127.0.0.135 的回复: 字节=32 时间<1ms TTL=128 127.0.0.135 --> Live
当前端口扫描完毕
====================================
扫描共花费的时间: 0:00:01.448122

Process finished with exit code 0
```

4. 使用 TCP 进行端口扫描

在实践了 Socket， Ping 方式进行端口扫描后，我们再来看另一种方式，通过建立 TCP 进行端口扫描。

TCP 需要经过三次握手，才能建立连接，三次握手创建连接的过程如下：

（1）步骤 1 - 设置 SYN 标志的数据包 首先，客户端会向服务端发送一段 TCP 报文，标志位为 SYN，表示“请求建立新连接”。

（2）步骤 2 - 设置 SYN-ACK 标志的数据包 接着，正常情况下，正处于监听状态下的服务端，会接收来自客户端发来的报文，随后，服务端也会返回 一段 TCP 报文。 同时标志位为 SYN-ACK，表示“确认客户端的报文序号有效，告诉客户端，服务端收到了来自客户端的数 据，同意创建新连接”； 即，目标系统返回一个设置了 SYN 和 ACK 标志的数据包。

（3）设置 ACK 标志的数据包 客户端接收到来自服务端的确认（即，确认收到服务端发来的 TCP 报文）之后，明确了从客户端到服务端的 数据通信是正常的。 这时，客户端会返回最后一段 TCP 报文，标志位为 ACK，表示“确认收到服务器端同意连接的信号”； 即，告诉服务器，我知道你收到我发的数据了。

也许，在读者看来，这三次握手间的相互确认着实麻烦了些，既然可以通过上面的 Ping 命令进行端口扫描，何必又多此一举，用 TCP 创建连接的方式，进行端口扫描呢？

其背后的主要原因是假设我们对数据包使用了防火墙，那么 Ping 扫描程序将无法工作，这时就需要 TCP 扫描来协助了。

具体实现脚本如下：

```
import socket
from datetime import datetime

net = input("需要扫描的主机: ")

# IP 地址拼接
net1 = net.split('.')
a = '.'
```

```
net2 = net1[0] + a + net1[1] + a + net1[2] + a

# 指定待扫描的端口号范围
st1 = int(input("请输入起始端口号: "))
en1 = int(input("请输入最后一个端口号: "))
en1 = en1 + 1

# 获取系统当前时间
t1 = datetime.now()
def scan(addr):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socket.setdefaulttimeout(1)
    result = s.connect_ex((addr, 135))
    print(result)
    if result == 0:
        return 1
    else:
        return 0
def run_scan():
    for ip in range(st1, en1):
        addr = net2 + str(ip)
        if (scan(addr)):
            print(addr, "is live")

run_scan()
t2 = datetime.now()
total = t2 - t1
print("扫描共花费的时间: ", total)
```

首先通过将 IP 地址范围拆分为多个部分来选择要 ping 扫描扫描的 IP 地址范围。

接下来是使用扫描地址的函数，该函数进一步使用套接字，从而得出有关主机的响应以及完成扫描过程所需的时间。

其中 connect_ex((addr,135)) 语句，如果操作成功则为 0，否则为 errno 变量的值。

安全起见，我们以本机为例 "127.0.0.1"，分别输入起止端口号，为了便于演示和节省时间，这里我们仅扫描 130-135 端口，进行查看。

扫描结果如下：

```
需要扫描的主机: 127.0.0.1
请输入起始端口号: 130
请输入最后一个端口号: 135
127.0.0.130 is live
127.0.0.131 is live
127.0.0.132 is live
127.0.0.133 is live
127.0.0.134 is live
127.0.0.135 is live
扫描共花费的时间: 0:00:00.008973

Process finished with exit code 0
```

5. 总结

无论你是否有志于安全测试，端口扫描技术都是测试人员知识储备库中不可或缺的一项。

本次分享通过 Python 提供的三种方式，对目标系统进行端口扫描，进行了实战演练。希望通过本文的学习，能够加深你对网络端口的认知，让它不再成为你的知识盲区。

此外如对文中提到的 Socket 套接字及其 Python 代码实现不是特别清楚的话，可以参考我之前分享的内容《测试人必会 —— Python 带你上手 Web Socket》。

测试经验谈

◆作者：刘晓佳

一、如何快速找出程序/软件问题

• 先测试变更部分（如新增需求，修改过 bug 的代码），再测试没有变更的部分。因为：修改和更新意味着有新的风险；

• 先测试核心部分（如重点功能，核心需求），再测试辅助功能。因为：有限时间内，测试完关键和常用功能，意味着完成产品基本功能的测试；

• 先测试功能，再测试性能。因为：只有功能正常后，测试性能再有意义，否则功能改动后，之前的性能测试报告就变得无用；

• 先测试常用场景，再测试少见场景。因为：有限的时间内，完成常用场景测试能尽量覆盖用户使用场景，避免基本错误出现；

• 先测试影响大的问题，再测试影响小的问题。因为：影响大的问题造成的破坏大于影响小的问题。

二、如何提供一份具有信服力的测试报告

• 收集和提供测试评估证据，如测试环境、复现方法、复现频率、故障现象截图/视频录像等。因为：足够的证据和信息能够减少测试人员与开发人员的沟通成本，能够帮助开发人员快速地定位故障代码；

• 尽量使用逻辑地表达方式，如 given……when……then，表述已知 xx 条件下，进行 xx 操作，产生 xx 结果。因为：测试任用在梳理逻辑性的同时，可以加深对故障诱因的确定，且逻辑清晰的表达方式能让开发人员快速地理清前因后果；

• 进行有效地拓展推论，如当前故障地波及影响，可能影响地其他模块或功能。因为：有的故障（如接口故障）可能对其他模块产生影响，在分析当前故障影响时，推论

对其他模块的波及能够提前发现隐藏的波及故障。

三、测试人员如何进行思考

• 技术性思考，如理解程序/软件内部结构、数据流转过程、状态变换等等。因为：只有了解软件内部运作，才能进行更深入地测试，挖掘深层故障；

• 创造性思考，如对观察到的现象进行有效推论，寻找可能存在的问题，或对使用工具/测试方法进行想象，寻找更有效的工具。因为：创造性思考，拓展性推论更能“奇兵制胜”，提高测试效率；

• 批判性思考，如不要“人云亦云”，观察到某个可能是故障的现象时，不要因为开发人员的否定而放弃，要坚持“寻找真相”，哪怕最终证明的确不是故障。因为：开发人员也有可能局限于部分场景，判断失误，在“寻找真相”的过程中，能够加深对程序/软件的理解，收集足够的证据完成判断；

• 实用性思考，纸上得来终觉浅，需知此时要躬行。只有付诸实践的思考才能有结果，否则终如纸上谈兵。因为：即使全面的思考和有效地推论也难免有所遗漏，只有在实践过程中，不断改进和优化，才能将思考的方案、方法与现实情况有机结合。

四、如何在实践中进行探索

• 向前探索，从已知探索未知。如：某个界面配置页面，点击“下一项”试试；

• 向后探索，从当前状态返回前置状态。如：某个界面配置页面，点击“上一项”试试；

• 横向探索，跳出当前状态，切换到其他状态。如：某个界面配置页面，未完成配置，刷新页面试试。

五、如何快速产生探索性测试思路

• 测试程序边界，常用的边界值测试法。如：某个输入窗口限定输入 1-100，试试 1 和 100 的输入值测试；

• 测试所有错误信息，即异常代码处理。如：程序运算溢出导致内存增加，溢出异常频发出现可能导致内存溢出；

• 测试非默认配置。如：软件安装会提供默认选项配置，试试非默认配置；

• 测试极限测试。如：在硬件资源告警时，测试程序运行情况。

很多人说测试是需要经验的，的确，有经验的指导能够帮助测试人员更好的测试。“测试经验谈”第一部分就先到这里吧~

《51 测试天地》（六十四）下篇 精彩预览

- 混沌工程-为未知做好准备吧
- 局部探索性测试
- 良好的开端是自动化测试成功的一半
- 没有脚本的自动化测试
- 浅谈智能驾驶中的持续集成(CI)测试
- 笑谈 3 年测试生涯
- 一文读懂契约测试
- 双剑合并-拨开内存溢出的真相

马上阅读