接口自动化测试 (python)

作者：XinyiZhang
原文链接：https://ld246.com/article/1601259755974
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="rsz 软件测试.jpg" data-src="
ttps://b3logfile.com/file/2020/07/rsz%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95-e8a
0c4a.jpg?imageView2/2/interlace/1/format/jpg">
<h2 id="技术选型">技术选型</h2>

Python3.7 或 3.8
unittest -- 自带
requests

发送 http 请求
pip install requests

HTMLTestRunner -- 生成测试报告
Fiddler -- 抓包工具
Pycharm 社区版集成开发环境 -- <a href="https://ld246.com/forward?goto=https%3A%2F
2Fwww.jetbrains.com%2Fpy%2520charm%2Fdownload%2F%23section%3Dmac" target="_bla
k" rel="nofollow ugc">下载地址

<h2 id="unitest">unitest</h2>

Python 单元测试框架,类似于 java 的 JUnit 框架 <a href="https://ld246.com/forward?goto=
ttps%3A%2F%2Fdocs.python.org%2Fzh-cn%2F3.8%2Flibrary%2Funittest.html" target="_blank
 rel="nofollow ugc">官网
unittest 核心 -- TestFixture、TestCase、TestSuite、TestRunner

<h2 id="注意">注意</h2>

所有类中方法的入参为 self,定义方法的变量也要“self.变量;
定义测试用例,以“test”开头命名的方法,方法的入参为 self;
unittest.main()方法会搜索该模块下所有以 test 开头的测试用例方法,并自动执行它们;
自己写的 py 文件不能用 unittest.py 命名,不然会找不到 TestCase;
用例成功是输出 . 失败是 F;

快速开发一个例子
<pre><code class="language-python highlight-chroma"><spa
 class="highlight-cl"># -*- coding: UTF-8 -*-
<span class="high
ight-kn">import unittest
<span class="high
ight-k">class UserTestCase<span class="highli
ht-p">(unittest.</sp
n>TestCase):
 <span class="hig
light-k">def setUp(
/span>self):
 <span class="hig
light-nb">print(" set
up 开始")
 <span class="hig
light-k">def tearDown<span class="highlight-p
>(self):
 <span class="hig
light-nb">print("tea

Down 执行结束")
 <span class="hig
light-k">def testCase1<span class="highlight-p
>(self):
 <span class="hig
light-nb">print("tes
case1")
 <span class="hig
light-k">def testCase2<span class="highlight-p
>(self):
 <span class="hig
light-nb">print("tes
case2")
<span class="high
ight-k">if __name__ <span class="highlight-o
>== '__main__':</s
an>
 <span class="hig
light-n">unittest.ma
n()
</code></pre>
<h2 id="断-">断言</h2>
self.assert* -- <a href="https://ld246.com/forward?goto=https%3A%2F%2Fdocs.python.o
g%2Fzh-cn%2F3%2Flibrary%2Funittest.html%23unittest.TestCase.debug" title="相关文档" tar
et="_blank" rel="nofollow ugc">文档
<h2 id="方法与注解">方法与注解</h2>
<h3 id="setUp--">setUp()</h3>
<pre><code class="highlight-chroma"><span class="highlight
cl"># 每个用例执行前都会执行
def setUp(self):
</code></pre>
<h3 id="tearDown--">tearDown()</h3>
<pre><code class="highlight-chroma"><span class="highlight
cl"># 每个用例执行后都会执行
def tearDown(self)

</code></pre>
<h3 id="setUpClass--">setUpClass()</h3>
<pre><code class="highlight-chroma"><span class="highlight
cl"># 执行所有用例前只运行一次
@classmethod
def setUpClass(cls)

</code></pre>
<h3 id="tearDownClass--">tearDownClass()</h3>
<pre><code class="highlight-chroma"><span class="highlight
cl"># 执行所有用例后只运行一次
@classmethod
def tearDownClass
cls):
</code></pre>
<h3 id="unittest-skip--">unittest.skip()</h3>
<pre><code class="highlight-chroma"><span class="highlight
cl"># 跳过某个测试用例

@unittest.skip("过这个测试")
</code></pre>
<h3 id="verbosity">verbosity</h3>
<pre><code class="highlight-chroma"># verbosity -- 日志级别 默认是1
# 0 -- 最简洁，不出每个用例执行结果
# 2 -- 输出用例的细执行结果
unittest.main(verosity=2)
</code></pre>
<h3 id="TestSuite--">TestSuite()</h3>

用来确定测试用例的顺序,哪个先执行哪个后执行;
如果一个 class 中有四个 test 开头的方法,则加载到 suite 中时则有四个测试用例;
由 TestLoder 加载 TestCase 到 TestSuite;
调用 addTest 来加载测试用例 -- 类名('方法名')的集合

addTest() 添加一个测试用例
addTest([,,])添加多个测试用例

<pre><code class="highlight-chroma">if __name__ == '__main__':
 # verbosity 默认1,为0的话最简洁,不输出每个用例执行结果,2 输出用例的详细执行结果
 # unittest.main(verbosity=2)
 # 构造一个测试套
 suite = unittest.TestSuite()
 # 类名('方法名')集合
 suite.addTest(UseTestCase2("testCase3"))
 suite.addTest(UseTestCase("testCase2"))
 suite.addTest(UseTestCase2("testCase2"))
 # 批量添加
 suite.addTests([UerTestCase2("testCase3"),UserTestCase2("testCase2"),UserTestCase("testCase2")])
 # 执行测试TextTesRunner()文本测试用例运行器,通过该类下面的run()方法来运行suite所组装的测试用例,入参为suite试套件
 runner = unittest.extTestRunner(verbosity=2)
 # run()方法是运测试套件的测试用例,入参为suite测试套件
 runner.run(suite)

</code></pre>
<h3 id="TestLoader--">TestLoader()</h3>
用例加载器,我们可以通过把用例都存放在这里,然后再通过 Suite 进行批量执行,但无法对 case 行排序
<pre><code class="highlight-chroma">if __name__ == '__main__':
  # 构造测试套件
  suite = unittest.T stSuite()
  # 实例化loader
  loader = unittest TestLoader()
  # 加载 UserTestC se 下的全部用例
  suite.addTests(lo der.loadTestsFromTestCase(UserTestCase))
  suite.addTests(lo der.loadTestsFromTestCase(UserTestCase2))
  runner = unittest TextTestRunner(verbosity=2)
  runner.run(suite)
</code></pre>
<h3 id="discover">discover</h3>
批量加载文件夹用例

case_dir -- 待执行用例的目录
pattern -- 这个是匹配脚本名称的规则,test*.py 意思是匹配 test 开头的所有脚本
top_level_dir -- 这个是顶层目录的名称,一般默认等于 None 就行

<pre><code class="highlight-chroma"># -*- coding: UTF-8 -*-
import unittest
import os
def load_all_case():
  """加载指定路径 全部测试用例"""
  # print(os.getcw ())
  # 用例路径，cas 是文件名称
  case_path = os.p th.join(os.getcwd(),"case")
  # print(case_path

  discover = unitte t.defaultTestLoader.discover(case_path, pattern="*Case.py",top_level_dir=None)
  return discover
if __name__ == '_ main__':
  runner = unittest TextTestRunner()
  runner.run(load_ ll_case())
</code></pre>

<h2 id="requests">requests</h2>
python 需要发起网络请求,在标准库中 urllib2 模块已经包含了平常我们使用的大多数功能,但是的 API 使用起来让人感觉不太好.大神开发了 Requests 模块,继承了 urllib2 的所有特性,支持 HTTP 接保持和连接池,支持使用 cookie 保持会话,支持文件上传等,本质就是封装了 urllib3. 相关文档
<h2 id="安装">安装</h2>
<pre><code class="highlight-chroma">pip install requests
</code></pre>
<h2 id="快速使---包模块命名不能-http-">快速使用 (包模块命名不能用 http)</h2>
<pre><code class="highlight-chroma"># -*- coding: UTF-8 -*-
import requests
response = reques s.get("请求的URL地址")
print(response.tex)
</code></pre>
<h2 id="常用的API">常用的 API</h2>
<h3 id="响应内容">响应内容</h3>

http 状态码 -- response.status_code
使用 response.text

<pre><code class="highlight-chroma">Requests会基于HTTP响应的文本编码自动解码响应内容,大多数Unicode字符集都能正常解码.
</code></pre>

使用 response.content

<pre><code class="highlight-chroma">返回的是服务器响应数据的原始二进制字节流，一般用来保存图片等二进制文件
</code></pre>

使用 response.json()

<h3 id="get请求带参数请求">get 请求带参数请求</h3>
<pre><code class="highlight-chroma"># 构建请求参数
data = {"video_id": 53}
response = reques s.get("请求的URL地址", data)
</code></pre>
<h3 id="post请求带参数请求">post 请求带参数请求</h3>
<pre><code class="highlight-chroma"># 构建请求参数
data = {"phone": " 3910423314", "pwd":"1234567890"}
response = reques s.post("请求的URL地址", data=data)
# 注意点: post提方式有两个传参方式，针对不同的content-type,务必要指定接口是哪个类型,表单提交还是json提交

Content-Type : pplication/x-www-form-urlencoded -- requests.post("url", data=data)
Content-Type : pplication/json -- requests.post("url", json=data)
</code></pre>
<h3 id="请求增加Header信息">请求增加 Header 信息</h3>
<pre><code class="highlight-chroma">headers = {"token":"获取的token值"}
# get带请求头信息
response = reques s.get("请求的URL地址", headers=headers)
# post带请求头信

response = reques s.post("请求的URL地址", data={"id":99}, headers=headers)
</code></pre>