

项目完整版代码

```
1  01综合案例.py
2  #!/usr/bin/env python
3  # -*- coding:utf-8 -*-
4  #====#====#====#====
5  #Author:
6  #CreatDate:
7  #Version:
8  #====#====#====#====
9  import unittest
10 import myselenium
11 import time
12 from mytools import mySendEmail
13 from HTMLTestRunner import HTMLTestRunner
14 if __name__=="__main__":
15     suit = unittest.TestSuite()
16     # 获取ddt驱动之后的测试用例名字
17     mylist = unittest.TestLoader().getTestCaseNames(myselenium.Test_Maker)
18     suit.addTests(map(myselenium.Test_Maker, mylist))
19
20     filename = "." + time.strftime("%Y-%m-%d %H_%M_%S") + ".res.html"
21     with open(filename, "wb") as f:
22         runner = HTMLTestRunner(f, verbosity=2, title="单元测试报告", description="第一次运行结果")
23         runner.run(suit)
24
25     #发送测试报告到Boss的邮箱
26     mySendEmail(filename, "lzs8407@163.com")
27
28
29 mytools.py代码
30 #!/usr/bin/env python
31 # -*- coding:utf-8 -*-
32 #====#====#====#====
33 #Author:
34 #CreatDate:
35 #Version:
36 #====#====#====#====
37 import hashlib
38 #构建邮件内容的
39 from email.mime.text import MIMEText
40 #构建邮件头部信息的
41 from email.header import Header
42 #构建发件人
43 from email.utils import formataddr
44 #创建发送邮件对象
45 import smtplib
46 import pymysql
47 import requests
```

```

48 #该文件写一些方法
49 #加密函数
50 def MyMd5(mystr):
51     mdmystr=hashlib.md5(mystr.encode(encoding='utf-8')).hexdigest()
52     return mdmystr
53
54 #发送邮件
55 def mySendEmail(filename,dis):
56     # 读取报告的内容
57     htmlmsg = open(filename,'rb').read()
58     # 构建邮件正文
59     msg = MIMEText(htmlmsg, 'html', 'utf-8')
60     # 头部信息
61     msg['Subject'] = Header("测试报告", 'utf-8')
62     # 发件人信息
63     msg['From'] = formataddr(['职员张某某202305', "76754438@qq.com"])
64     # 收件人
65     msg['To'] = dis
66
67     # 构建SMTP对象
68     smtp = smtplib.SMTP()
69     # 连接发送邮件的邮箱服务器
70     smtp.connect("smtp.qq.com")
71     # 登录
72     smtp.login("76754438@qq.com", "uomuthspmlfqbhcg")
73     # 发送
74     smtp.sendmail("76754438@qq.com", dis, msg.as_string())
75     # 退出
76     smtp.quit()
77     print("邮件发送成功....")
78
79 # 获取数据库中的数据
80 def getMysql():
81     mylist=[]
82     db = pymysql.connect(host='127.0.0.1', port=3306, user="root",
83     passwd='123456', db="mytest202305", charset='utf8')
84     # 创建游标
85     cur = db.cursor()
86     sql = "select value from myvocabulary"
87     cur.execute(sql)
88     data = cur.fetchall()
89     for i in data:
90         mylist.append(i[0])
91
92     return mylist
93
94 '''
95 [
96 {'appid': '20210507000816969'},
97 {'baidu_url': 'https://fanyi-api.baidu.com/api/trans/vip/translate'},
98 {'data': ['q', 'from', 'to', 'appid', 'salt', 'sign']},
99 {'from': 'en'},

```

```

100     {'miyao': 'sGZsjo0Y20ThznTznsRT'},
101     {'salt': '123456'},
102     {'to': 'zh'},
103     {'type': 'get'},
104     {'url': 'https://www.baidu.com'}}]
105 '''
106
107 # 翻译方法
108 def myTranslate(cdata,data):
109     dis=""
110     url = cdata[1]['baidu_url']
111     mdmystr = MyMd5(cdata[0]["appid"] + data+ cdata[5]["salt"]+cdata[4]["miyao"])
112     # 参数
113     data = {
114         cdata[2]["data"][0]: data,
115         cdata[2]["data"][1]: cdata[3]["from"],
116         cdata[2]["data"][2]: cdata[6]["to"],
117         cdata[2]["data"][3]: cdata[0]["appid"],
118         cdata[2]["data"][4]: cdata[5]["salt"],
119         cdata[2]["data"][5]: mdmystr
120     }
121     # 20210507000816969test123456sGZsjo0Y20ThznTznsRT
122     # 请求
123     res = requests.get(url, params=data)
124     # 把响应转换为json格式,便于后面获取里面的内容
125     jr = res.json()
126     # 获取响应后的翻译结果
127     rdata = jr['trans_result'][0]['dst']
128     # 消息-新闻 铁巴-贴吧 相片-图片 网络磁盘-网盘
129     if rdata=="消息":
130         rdata="新闻"
131     elif rdata=="铁巴":
132         rdata="贴吧"
133     elif rdata=="相片":
134         rdata="图片"
135     elif rdata=="网络磁盘":
136         rdata="网盘"
137
138     dis=rdata
139     return dis
140
141
142 # import unittest
143 # import ddt
144 #
145 # @ddt.ddt#代表下面这个单元测试支持ddt驱动
146 # class Testmaker(unittest.TestCase):
147 #     @ddt.file_data("conf.yaml")#参数是文件名
148 #     def test_ddt4(self,txt):#执行了4次,因为mydata.yaml中有4个数据
149 #         print("test_ddt4")
150 #         print(txt)
151 #         dis=myTranslate(txt,"picture")
152 #         print(dis)

```

```
153 #
154 # if __name__=='__main__':
155 #     unittest.main()
156
157 myselenium.py代码
158 #!/usr/bin/env python
159 # -*- coding:utf-8 -*-
160 #====#====#====#====
161 #Author:
162 #CreatDate:
163 #Version:
164 #====#====#====#====
165 #该文件是写单元测试类
166 import unittest
167 import ddt
168 from mytools import getMysql,myTranslate
169 import time
170 from selenium import webdriver
171
172 @ddt.ddt
173 class Test_Maker(unittest.TestCase):
174     @classmethod
175     def setUpClass(cls):
176         #从数据库中获取数据
177         cls.mylist=getMysql()
178
179     def setUp(self):
180         print("setUp")
181         print(self.mylist)
182         self.dr=webdriver.Firefox()
183
184     @ddt.file_data("conf.yaml") # 参数是文件名
185     def test_01(self,txt):
186         print("test_01")
187         #获取翻译后的内容
188         text=myTranslate(txt,self.mylist[0])
189         print(text)
190         time.sleep(2)
191         self.dr.get(txt[-1]["url"])
192         self.dr.find_element_by_link_text(text).click()
193         time.sleep(2)
194         #预期结果:
195         #新页面的url
196         url="https://news.baidu.com/"
197         #浏览器的页面个数
198         winnum=2
199         #实际结果
200         winlist=self.dr.window_handles
201         self.dr.switch_to.window(winlist[1])
202         time.sleep(2)
203         resurl=self.dr.current_url
204         self.assertEqual(url,resurl,"测试用例未通过1")
205         self.assertEqual(winnum,len(winlist),"测试用例未通过2")
```

```
206
207 @ddt.file_data("conf.yaml") # 参数是文件名
208 def test_02(self,txt):
209     print("test_02")
210     #获取翻译后的内容
211     text=myTranslate(txt,self.mylst[1])
212     print(text)
213     time.sleep(2)
214     self.dr.get(txt[-1]["url"])
215     self.dr.find_element_by_link_text(text).click()
216     time.sleep(2)
217     #预期结果:
218     #新页面的url
219     url="https://map.baidu.com/"
220     #浏览器的页面个数
221     winnum=2
222     #实际结果
223     winlst=self.dr.window_handles
224     self.dr.switch_to.window(winlst[1])
225     time.sleep(2)
226     resurl=self.dr.current_url
227     self.assertEqual(url,resurl,"测试用例未通过1")
228     self.assertEqual(winnum,len(winlst),"测试用例未通过2")
229
230 @ddt.file_data("conf.yaml") # 参数是文件名
231 def test_03(self,txt):
232     print("test_03")
233     #获取翻译后的内容
234     text=myTranslate(txt,self.mylst[2])
235     print(text)
236     time.sleep(2)
237     self.dr.get(txt[-1]["url"])
238     self.dr.find_element_by_link_text(text).click()
239     time.sleep(2)
240     #预期结果:
241     #新页面的url
242     url="https://tieba.baidu.com/index.html"
243     #浏览器的页面个数
244     winnum=2
245     #实际结果
246     winlst=self.dr.window_handles
247     self.dr.switch_to.window(winlst[1])
248     time.sleep(2)
249     resurl=self.dr.current_url
250     self.assertEqual(url,resurl,"测试用例未通过1")
251     self.assertEqual(winnum,len(winlst),"测试用例未通过2")
252
253 @ddt.file_data("conf.yaml") # 参数是文件名
254 def test_04(self,txt):
255     print("test_04")
256     #获取翻译后的内容
257     text=myTranslate(txt,self.mylst[3])
258     print(text)
```

```

259         time.sleep(2)
260         self.dr.get(txt[-1]["url"])
261         self.dr.find_element_by_link_text(text).click()
262         time.sleep(2)
263         #预期结果:
264         #新页面的url
265         url="https://haokan.baidu.com/?sfrom=baidu-top"
266         #浏览器的页面个数
267         winnum=2
268         #实际结果
269         winlist=self.dr.window_handles
270         self.dr.switch_to.window(winlist[1])
271         time.sleep(2)
272         resurl=self.dr.current_url
273         self.assertEqual(url, resurl, "测试用例未通过1")
274         self.assertEqual(winnum, len(winlist), "测试用例未通过2")
275
276     @ddt.file_data("conf.yaml") # 参数是文件名
277     def test_05(self, txt):
278         print("test_05")
279         #获取翻译后的内容
280         text=myTranslate(txt, self.mylst[4])
281         print(text)
282         time.sleep(2)
283         self.dr.get(txt[-1]["url"])
284         self.dr.find_element_by_link_text(text).click()
285         time.sleep(2)
286         #预期结果:
287         #新页面的url
288         url="https://image.baidu.com/"
289         #浏览器的页面个数
290         winnum=2
291         #实际结果
292         winlist=self.dr.window_handles
293         self.dr.switch_to.window(winlist[1])
294         time.sleep(2)
295         resurl=self.dr.current_url
296         self.assertEqual(url, resurl, "测试用例未通过1")
297         self.assertEqual(winnum, len(winlist), "测试用例未通过2")
298
299     @ddt.file_data("conf.yaml") # 参数是文件名
300     def test_06(self, txt):
301         print("test_06")
302         #获取翻译后的内容
303         text=myTranslate(txt, self.mylst[5])
304         print(text)
305         time.sleep(2)
306         self.dr.get(txt[-1]["url"])
307         self.dr.find_element_by_link_text(text).click()
308         time.sleep(2)
309         #预期结果:
310         #新页面的url
311         url="https://pan.baidu.com/?from=1026962h"

```

```
312         #浏览器的页面个数
313         winnum=2
314         #实际结果
315         winlist=self.dr.window_handles
316         self.dr.switch_to.window(winlist[1])
317         time.sleep(2)
318         resurl=self.dr.current_url
319         self.assertEqual(url,resurl,"测试用例未通过1")
320         self.assertEqual(winnum,len(winlist),"测试用例未通过2")
321
322
323     def tearDown(self):
324         time.sleep(2)
325         self.dr.quit()
326
327     yaml:
328     mydata:
329         - appid: '20210507000816969'
330         - baidu_url: https://fanyi-api.baidu.com/api/trans/vip/translate
331         - data:
332             - q
333             - from
334             - to
335             - appid
336             - salt
337             - sign
338         - from: en
339         - miyao: SGZsjo0Y20ThznTznsRT
340         - salt: '123456'
341         - to: zh
342         - type: get
343         - url: https://www.baidu.com
344
```

git的概述

- 1 | git是分布式的版本管理系统
- 2 |
- 3 | github的网址

ssh-key的客户端配置(淘汰)

```
1 1.打开gitBush
2 2.输入命令:ssh-keygen -t rsa -C "76754438@qq.com"
3 3.进入到.ssh文件夹:cd .ssh
4 4.里面有2个重要的文件,是id_rsa(私钥) id_rsa.pub(公钥)
5 5.查看公钥的内容:cat id_rsa.pub
6 6.去github网址上把公钥的内容设置到网站上
7 7.设置方法:
8     1. 打开github网站,输入用户名和密码,进入到网站中
9     2. 点击右上角,用户的头像旁边小三角,点击setting
10    3. 这时页面左边有一个列表,在列表中选择"SSH and GPG KEYS" -> SSH Keys里的new ssh key
11    4. 然后把公钥的内容粘贴到key下面的输入框
```

创建git仓库和克隆(重点)

```
1 1.github上创建仓库
2 2.把远程仓库克隆到本地
3     1. 获取远程仓库的地址:git@github.com:76754438/cntest2023.git
4     2. 打开你的git Bush
5     3. 创建文件夹:mkdir mytest
6     4. 进入到这个文件夹:cd mytest
7     5. 输入命令:git clone https://github.com/76754438/cntest202305.git
8     6. 当前目录下会出现一个远程下载下来的目录
```

推送(重点)

```
1 1.进入到刚才下载的目录中:cd cntest2023
2 2.创建本地文件:touch 1.txt
3 3.把1.txt文件加入到暂存区:git add 1.txt
4 4.可以查看仓库的状态:git status
5 5.把1.txt文件存储到本地仓库:git commit -m"第一次提交1.txt"
6 6.把本地仓库的内容推送到远程仓库:git push
7
8 需要绑定的邮箱和token
9 76754438@qq.com
10 token:ghp_vzNHwYj8Svjn91iSB8h3haBbJojItD0vbQER
11
12 github上获取token
13 1. 打开github网站,输入用户名和密码,进入到网站中
14 2. 点击右上角,用户的头像旁边小三角,点击setting
15 3. 这时页面左边有一个列表,选择最下面的Developer settings
16 4. 选择左边列表中的Personal access tokens,然后选择下面的Tokens(classic)
17 5. 选择右边的Generate new token旁边的下三角,选择Generate new token(classic)
18 6. Note下面的输入框填入标题,随便填
19 7. 勾选repo,user,delete_repo
20 8. 点最下面的Generate token按键
21 9. 跳转的页面上显示token
```


22
23

创建本地仓库

1. 在磁盘上创建一个文件夹,不要在.ssh中创建
2. 进入创建的文件夹,然后输入git init
3. 关联远程的仓库:git remote add origin 地址
4. 创建文件(touch 1.txt),然后加入到暂存区(git add 1.txt),然后添加到本地(git commit -m"1.txt")
5. 推送:git push -u origin master
6. 输入用户名和密码

拉取(重点)

- 1 | git pull

查看分支及本地创建分支

- 1 一般各个小组都会创建自己的分支,然后在软件上线前,负责人会合并所有的分支
- 2
- 3 1.查看分支:git branch
- 4 2.查看远端分支:git branch -a
- 5 3.创建本地分支并进入到分支:git checkout -b 分支名
- 6 4.切换分支:git checkout 分支名

远程端创建分支

- 1 需要到别的分支来推送某个分支到远端
- 2 说明:mytest是分支名字
- 3 远程创建分支:git push origin mytest --set-upstream

本地分支删除

- 1 注意:不能在本分支删除自己
- 2 删除分支:git branch -d 分支名

远程端分支删除

- 1 把本地删除分支的信息告诉远端,让远端也删除
- 2 `git push origin :分支名`

合并分支

- 1 合并分支:`git merge 分支名`
- 2
- 3 1.本地创建一个分支,然后在该分支下创建一个文件
- 4 2.然后把分支和文件一起推送到远端分支,这时远端的主分支是没有这个文件的
- 5 3.在本地合并分支,然后推送到主分支
- 6
- 7 步骤:
- 8 1.本地创建分支a1
- 9 `git checkout -b a1`
- 10 2.在a1分支上创建一个88.txt文件,并加入到暂存区和本地仓库
- 11 `touch 88.txt`
- 12 `git add 88.txt`
- 13 `git commit -m"88.txt"`
- 14 3.把a1分支及里面的88.txt一起推送到远端
- 15 `git push origin a1 --set-upstream`
- 16 4.切换到main分支,这时main分支上没有88.txt
- 17 `git checkout main`
- 18 5.合并a1分支
- 19 `git merge a1`

解决合并分支的冲突问题

- 1 1.本地创建分支a1
- 2 `git checkout -b b1`
- 3 2.在a1分支上创建一个99.txt文件,内容是111111,并加入到暂存区和本地仓库
- 4 `vim 99.txt`
- 5 `git add 88.txt`
- 6 `git commit -m"99.txt"`
- 7 3.把b1分支及里面的99.txt一起推送到远端
- 8 `git push origin b1 --set-upstream`
- 9 4.切换到main分支,这时main分支上没有99.txt
- 10 5.在main分支上创建99.txt,内容是22222
- 11 8.合并:`git merge b1`,这时出现下面的错误,产生了冲突
- 12 `error: The following untracked working tree files would be overwritten by merge:`
- 13 `99.txt`
- 14 `Please move or remove them before you merge.`
- 15 `Aborting`
- 16 `Updating b2abc96..a0b4f30`
- 17 9.怎么解决,在工作中要和对方进行商量,然后决定保留谁的内容

```
18 | 10. 修改好文件,然后在进行加入仓库,就可以推送
19 |
20 |
21 |
```

批量推送文件

```
1 | 把当前目录下所有的文件加入到暂存区
2 | git add .
3 | git commit -m"所有文件"
4 | git push
```

本地回退之后再推送

```
1 | 回退命令:git reset --hard HEAD^
2 | ^这个有多少个,就回退多少个版本
3 | 回退之后强行推送到远端:git push --force
```

回退到指定版本

```
1 | 查看版本:git reflog
2 | 回到指定的版本:git reset --hard 版本号
```