项目完整版代码

```
01综合案例.py
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import unittest
import myselenium
import time
from mytools import mySendEmail
from HTMLTestRunner import HTMLTestRunner
if __name__=="__main__":
    suit = unittest.TestSuite()
    # 获取ddt驱动之后的测试用例名字
    mylist = unittest.TestLoader().getTestCaseNames(myselenium.Test_Maker)
    suit.addTests(map(myselenium.Test_Maker, mylist))

    filename = "./" + time.strftime("%Y-%m-%d %H_%M_%S") + "res.html"
    with open(filename, "wb") as f:
        runner = HTMLTestRunner(f, verbosity=2, title="单元测试报告", description="第一次运行结果")
        runner.run(suit)

    #发送测试报告到Boss的邮箱
    mySendEmail(filename,"lzs8407@163.com")


mytools.py代码
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import hashlib
#构建邮件内容的
from email.mime.text import MIMEText
#构建邮件头部信息的
from email.header import Header
#构建发件人
from email.utils import formataddr
#创建发送邮件对象
import smtplib
import pymysql
import requests
```

```
#该文件写一些方法
#加密函数
def MyMd5(mystr):
    mdmystr=hashlib.md5(mystr.encode(encoding='utf-8')).hexdigest()
    return mdmystr

#发送邮件
def mySendEmail(filename,dis):
    # 读取报告的内容
    htmlmsg = open(filename,'rb').read()
    # 构建邮件正文
    msg = MIMEText(htmlmsg, 'html', 'utf-8')
    # 头部信息
    msg['Subject'] = Header("测试报告", 'utf-8')
    # 发件人信息
    msg['From'] = formataddr(['职员张某某202305', "76754438@qq.com"])
    # 收件人
    msg['To'] = dis

    # 构建SMTP对象
    smtp = smtplib.SMTP()
    # 连接发送邮件的邮箱服务器
    smtp.connect("smtp.qq.com")
    # 登录
    smtp.login("76754438@qq.com", "uomuthspmlfgbhcg")
    # 发送
    smtp.sendmail("76754438@qq.com", dis, msg.as_string())
    # 退出
    smtp.quit()
    print("邮件发送成功....")

# 获取数据库中的数据
def getMysql():
    mylist=[]
    db = pymysql.connect(host='127.0.0.1', port=3306, user="root",
passwd='123456', db="mytest202305", charset='utf8')
    # 创建游标
    cur = db.cursor()
    sql = "select value from myvocabulary"
    cur.execute(sql)
    data = cur.fetchall()
    for i in data:
        mylist.append(i[0])

    return mylist


'''
[
{'appid': '20210507000816969'},
 {'baidu_url': 'https://fanyi-api.baidu.com/api/trans/vip/translate'},
 {'data': ['q', 'from', 'to', 'appid', 'salt', 'sign']},
 {'from': 'en'},
```

```
 {'miyao': 'sGZsjoOY2OThznTznsRT'},
 {'salt': '123456'},
 {'to': 'zh'},
 {'type': 'get'},
 {'url': 'https://www.baidu.com'}]
'''

# 翻译方法
def myTranslate(cdata,data):
    dis=""
    url = cdata[1]['baidu_url']
    mdmystr = MyMd5(cdata[0]["appid"] + data+ cdata[5]["salt"]+cdata[4]["miyao"])
    # 参数
    data = {
        cdata[2]["data"][0]: data,
        cdata[2]["data"][1]: cdata[3]["from"],
        cdata[2]["data"][2]: cdata[6]["to"],
        cdata[2]["data"][3]: cdata[0]["appid"],
        cdata[2]["data"][4]: cdata[5]["salt"],
        cdata[2]["data"][5]: mdmystr
    }
    # 20210507000816969test123456sGZsjoOY2OThznTznsRT
    # 请求
    res = requests.get(url, params=data)
    # 把响应转换为json格式,便于后面获取里面的内容
    jr = res.json()
    # 获取响应后的翻译结果
    rdata = jr['trans_result'][0]['dst']
    # 消息-新闻  铁巴-贴吧 相片-图片 网络磁盘-网盘
    if rdata=="消息":
        rdata="新闻"
    elif rdata=="铁巴":
        rdata="贴吧"
    elif rdata=="相片":
        rdata="图片"
    elif rdata=="网络磁盘":
        rdata="网盘"

    dis=rdata
    return dis


# import unittest
# import ddt
#
# @ddt.ddt#代表下面这个单元测试支持ddt驱动
# class Testmaker(unittest.TestCase):
#     @ddt.file_data("conf.yaml")#参数是文件名
#     def test_ddt4(self,txt):#执行了4次,因为mydata.yaml中有4个数据
#         print("test_ddt4")
#         print(txt)
#         dis=myTranslate(txt,"picture")
#         print(dis)
```

```
#
# if __name__=='__main__':
#     unittest.main()

myselenium.py代码
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
#该文件是写单元测试类
import unittest
import ddt
from mytools import getMysql,myTranslate
import time
from selenium import webdriver

@ddt.ddt
class Test_Maker(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        #从数据库中获取数据
        cls.mylist=getMysql()

    def setUp(self):
        print("setUp")
        print(self.mylist)
        self.dr=webdriver.Firefox()

    @ddt.file_data("conf.yaml")  # 参数是文件名
    def test_01(self,txt):
        print("test_01")
        #获取翻译后的内容
        text=myTranslate(txt,self.mylist[0])
        print(text)
        time.sleep(2)
        self.dr.get(txt[-1]["url"])
        self.dr.find_element_by_link_text(text).click()
        time.sleep(2)
        #预期结果:
        #新页面的url
        url="https://news.baidu.com/"
        #浏览器的页面个数
        winnum=2
        #实际结果
        winlist=self.dr.window_handles
        self.dr.switch_to.window(winlist[1])
        time.sleep(2)
        resurl=self.dr.current_url
        self.assertEqual(url,resurl,"测试用例未通过1")
        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.mylist[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         winlist=self.dr.window_handles
224         self.dr.switch_to.window(winlist[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(winlist),"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.mylist[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         winlist=self.dr.window_handles
247         self.dr.switch_to.window(winlist[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(winlist),"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.mylist[3])
258         print(text)
```

```
        time.sleep(2)
        self.dr.get(txt[-1]["url"])
        self.dr.find_element_by_link_text(text).click()
        time.sleep(2)
        #预期结果:
        #新页面的url
        url="https://haokan.baidu.com/?sfrom=baidu-top"
        #浏览器的页面个数
        winnum=2
        #实际结果
        winlist=self.dr.window_handles
        self.dr.switch_to.window(winlist[1])
        time.sleep(2)
        resurl=self.dr.current_url
        self.assertEqual(url,resurl,"测试用例未通过1")
        self.assertEqual(winnum,len(winlist),"测试用例未通过2")

    @ddt.file_data("conf.yaml")  # 参数是文件名
    def test_05(self,txt):
        print("test_05")
        #获取翻译后的内容
        text=myTranslate(txt,self.mylist[4])
        print(text)
        time.sleep(2)
        self.dr.get(txt[-1]["url"])
        self.dr.find_element_by_link_text(text).click()
        time.sleep(2)
        #预期结果:
        #新页面的url
        url="https://image.baidu.com/"
        #浏览器的页面个数
        winnum=2
        #实际结果
        winlist=self.dr.window_handles
        self.dr.switch_to.window(winlist[1])
        time.sleep(2)
        resurl=self.dr.current_url
        self.assertEqual(url,resurl,"测试用例未通过1")
        self.assertEqual(winnum,len(winlist),"测试用例未通过2")

    @ddt.file_data("conf.yaml")  # 参数是文件名
    def test_06(self,txt):
        print("test_06")
        #获取翻译后的内容
        text=myTranslate(txt,self.mylist[5])
        print(text)
        time.sleep(2)
        self.dr.get(txt[-1]["url"])
        self.dr.find_element_by_link_text(text).click()
        time.sleep(2)
        #预期结果:
        #新页面的url
        url="https://pan.baidu.com/?from=1026962h"
```

```
        #浏览器的页面个数
        winnum=2
        #实际结果
        winlist=self.dr.window_handles
        self.dr.switch_to.window(winlist[1])
        time.sleep(2)
        resurl=self.dr.current_url
        self.assertEqual(url,resurl,"测试用例未通过1")
        self.assertEqual(winnum,len(winlist),"测试用例未通过2")


    def tearDown(self):
        time.sleep(2)
        self.dr.quit()

yaml:
mydata:
  - appid: '20210507000816969'
  - baidu_url: https://fanyi-api.baidu.com/api/trans/vip/translate
  - data:
    - q
    - from
    - to
    - appid
    - salt
    - sign
  - from: en
  - miyao: sGZsjoOY2OThznTznsRT
  - salt: '123456'
  - to: zh
  - type: get
  - url: https://www.baidu.com

```

git的概述

```
git是分布式的版本管理系统

github的网址
```

ssh-key的客户端配置(淘汰)

```
1.打开gitBush
2.输入命令:ssh-keygen -t rsa -C "76754438@qq.com"
3.进入到.ssh文件夹:cd .ssh
4.里面有2个重要的文件,是id_rsa(私钥)  id_rsa.pub(公钥)
5.查看公钥的内容:cat id_rsa.pub
6.去github网址上把公钥的内容设置到网站上
7.设置方法:
    1.打开github网站,输入用户名和密码,进入到网站中
    2.点击右上角,用户的头像旁边小三角,点击setting
    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
    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.github上创建仓库
2.把远程仓库克隆到本地
    1.获取远程仓库的地址:git@github.com:76754438/cntest2023.git
    2.打开你的git Bush
    3.创建文件夹:mkdir mytest
    4.进入到这个文件夹:cd mytest
    5.输入命令:git clone https://github.com/76754438/cntest202305.git
    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.进入到刚才下载的目录中:cd cntest2023
2.创建本地文件:touch 1.txt
3.把1.txt文件加入到暂存区:git add 1.txt
4.可以查看仓库的状态:git status
5.把1.txt文件存储到本地仓库:git commit -m"第一次提交1.txt"
6.把本地仓库的内容推送到远程仓库:git push

需要绑定的邮箱和token
76754438@qq.com
token:ghp_vzNHWYj8Svjn91iSB8h3haBbJojItDOvbQER

github上获取token
1.打开github网站,输入用户名和密码,进入到网站中
2.点击右上角,用户的头像旁边小三角,点击setting
3.这时页面左边有一个列表,选择最下面的Developer settings
4.选择左边列表中的Personal access tokens,然后选择下面的Tokens(classic)
5.选择右边的Generate new token旁边的下三角,选择Generate new token(classic)
6.Note下面的输入框填入标题,随便填
7.勾选repo,user,delete_repo
8.点最下面的Generate token按键
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)

```
git pull
```

查看分支及本地创建分支

```
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

1.查看分支:git branch
2.查看远端分支:git branch -a
3.创建本地分支并进入到分支:git checkout -b 分支名
4.切换分支:git checkout 分支名
```

远程端创建分支

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

本地分支删除

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

远程端分支删除

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

合并分支

```
合并分支:git merge 分支名

1.本地创建一个分支,然后在该分支下创建一个文件
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
3.在本地合并分支,然后推送到主分支

步骤:
1.本地创建分支a1
git checkout -b a1
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
touch 88.txt
git add 88.txt
git commit -m"88.txt"
3.把a1分支及里面的88.txt一起推送到远端
git push origin a1 --set-upstream
4.切换到main分支,这时main分支上没有88.txt
git checkout main
5.合并a1分支
git merge a1
```

解决合并分支的冲突问题

```
1.本地创建分支a1
git checkout -b b1
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
vim 99.txt
git add 88.txt
git commit -m"99.txt"
3.把b1分支及里面的99.txt一起推送到远端
git push origin b1 --set-upstream
4.切换到main分支,这时main分支上没有99.txt
5.在main分支上创建99.txt,内容是22222
8.合并:git merge b1,这时出现下面的错误,产生了冲突
error: The following untracked working tree files would be overwritten by merge:
        99.txt
Please move or remove them before you merge.
Aborting
Updating b2abc96..a0b4f30
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
```

```
10.修改好文件,然后在进行加入仓库,就可以推送
```

批量推送文件

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

本地回退之后再推送

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

回退到指定版本

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