

# 德雅自动化测试框架



版本号	修改人	修改日期
1.0	凌宝万	2018.1.15

目录

- 一、德雅框架介绍 .....3
- 二、框架结构 .....3
  - 2.1 框架开发模式.....3
  - 2.2 框架结构.....4
    - 2.2.1 API 文件夹： .....5
    - 2.2.2 Autolt 文件夹： .....6
    - 2.2.3 BasePage 文件夹： ..... 16
    - 2.2.4 Common 文件夹..... 19
    - 2.2.5 image 文件夹 ..... 19
    - 2.2.6 Lib 文件夹 ..... 20
    - 2.2.7 Pages 文件夹..... 20
    - 2.2.8 report 文件夹： ..... 23
    - 2.2.9 TestCase 文件夹： ..... 24
    - 2.2.10 UseData 文件夹： ..... 25
    - 2.2.11 index.py 文件： ..... 25
  - 2.3 框架总结..... 26
- 三、写在后面 ..... 27
- 附件一教程： ..... 27

# 一、德雅框架介绍

德雅医疗是一家专业从事 3D 打印定制化牙科产品的创新型科技公司,致力于提供全方位数字化口腔问题解决方案。在项目的初期,德雅没有一个测试框架,在测试中,我们只有去点点点,目的是为了功能不出现严重的 bug。在项目进行迭代开发之后,传统的点点点已经不适合迭代开发的模式了,为了帮助德雅项目快速推进,尽早得到项目质量的标准,特此开发此测试框架。

框架基础是由 Python 语言开发的,借助 Selenium、HTMLTestRunner 等 python 库进行底层结构架构设计。持续集成测试则使用 Jenkins 作为持续集成工具。

## 搭建环境：

系统：Window10

开发语言：Python3.6.2(更高版本都可以兼容)

开发工具：Pycharm

开发底层库：Selenium2

# 二、框架结构

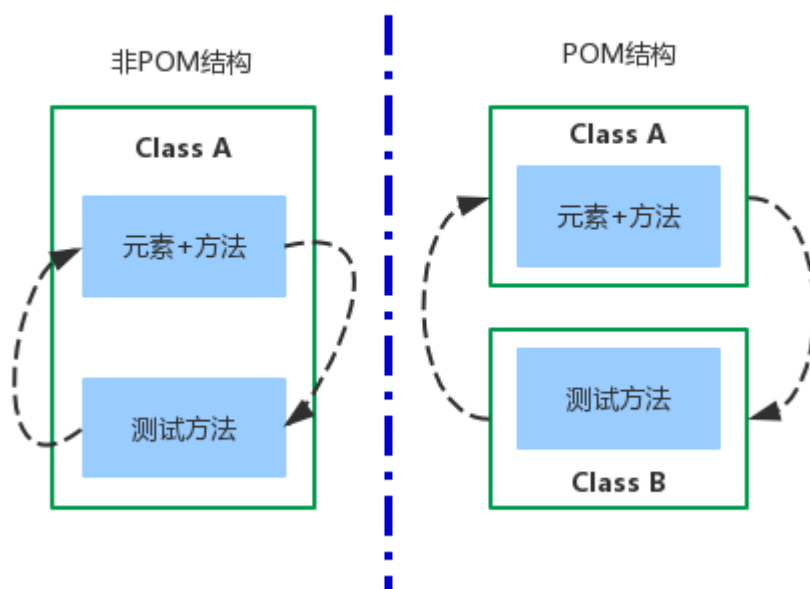
## 2.1 框架开发模式

德雅自动化测试框架使用的结构是使用目前流行的 POM ( Page Object Model ) 模式进行开发。

- 页面对象模型(POM)是一种设计模式,用来管理维护一组 web 元素集的对象库
- 在 POM 下,应用程序的每一个页面都有一个对应的 page class
- 每一个 page class 维护着该 web 页的元素集和操作这些元素的方法

- page class 中的方法命名最好根据其对应的业务场景进行，例如通常登录后我们需要等待几秒中，我们可以这样命名该方法: `waitingForLoginSuccess()`。

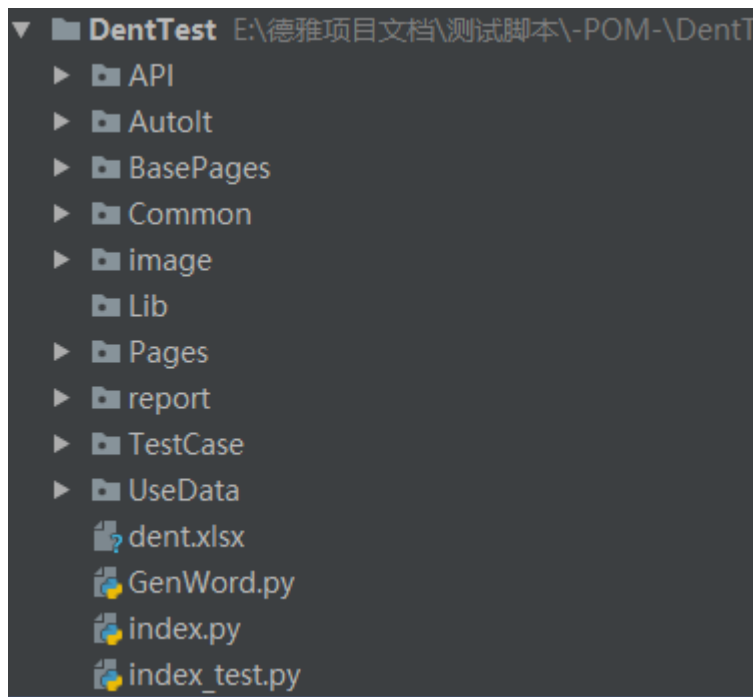
下图为非 POM 和 POM 对比图：



POM 框架的优点：

- POM 提供了一种在 UI 层操作、业务流程与验证分离的模式，这使得测试代码变得更加清晰和高可读性
- 对象库与用例分离，使得我们更好的复用对象，甚至能与不同的工具进行深度结合应用
- 可复用的页面方法代码会变得更加优化
- 更加有效的命名方式使得我们更加清晰的知道方法所操作的 UI 元素。例如我们要回到首页，方法名命名为: `gotoHomePage()`，通过方法名即可清晰的知道具体的功能实现。

## 2.2 框架结构



目前框架的成熟度还在继续优化中，希望你能给框架带来更好的优化程度。

上面的所有名字均由本人自定义命名，现在介绍每个文件的作用：

### 2.2.1 API 文件夹：

```
def delephone(phone):  
    """  
    这里是接口返回手机号码删除  
    :param phone:输入手机号码  
    :return:删除该手机号码  
    """  
    postdata = {'phone':phone}  
    headers = {'content-type': 'application/json'}  
    r=requests.post('http://112.74.29.84:23301/api/User/deleteuser',  
data=json.dumps(postdata),headers=headers)  
    token_str = r.text  
    token_dict = json.loads(token_str)  
    if token_dict['success'] == True:  
        return True  
    else:  
        return False
```

这个删除测试手机号的方法文件，在我们每次跑自动化测试的时候，我们经常地把手机号写死，为了方便后面每次的迭代，我向后台开发的同事要了这个删除手机号的接口，这个接口

仅用于测试版本，在正式版本中是无法使用的。

使用的方式：

Python+requests 中的 post 方法做接口测试方法。具体方法可以到 requests 官网查看：

[http://cn.python-requests.org/zh\\_CN/latest/user/quickstart.html](http://cn.python-requests.org/zh_CN/latest/user/quickstart.html)

## 2.2.2 Autolt 文件夹：

这个文件夹中包含的是我们在上传文件的时候使用到的工具，Autolt 是 python 中经常用到的第三方工具，这个工具也方便我们对上传下载文件的存放。关于 Autolt 教程，下面简单说一下：

### 2.2.2.1 Autolt3 下载安装

Autolt 目前最新是 v3 版本，这是一个使用类似 BASIC 脚本语言的免费软件,它设计用于 Windows GUI(图形用户界面)中进行自动化操作。它利用模拟键盘按键，鼠标移动和窗口/控件的组合来实现自动化任务。而这是其它语言不可能做到或无可靠方法实现的

官方下载：<https://www.autoitscript.com/site/autoit/downloads/>


## Current Versions



**Latest version:** v3.3.14.2

**Updated:** 18 September, 2015

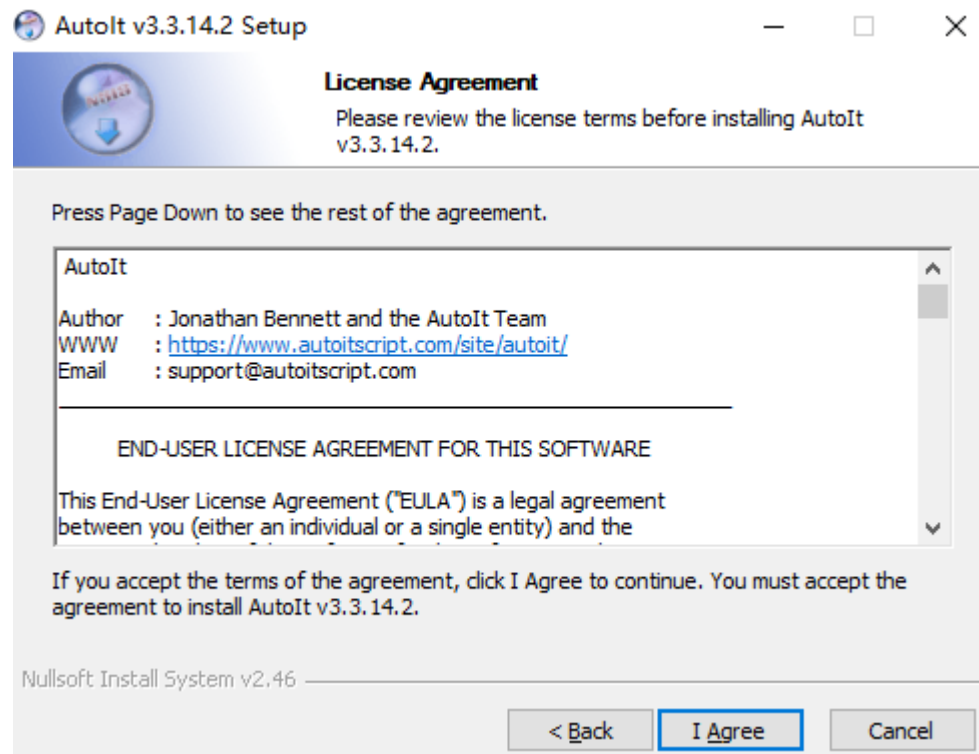
**History:** [View changelog](#)

Software	Download
<p><b>AutoIt Full Installation.</b> Includes x86 and x64 components, and:</p> <ul style="list-style-type: none"><li>• <b>AutoIt</b> program files, documentation and examples.</li><li>• <b>Aut2Exe</b> – Script to executable converter. Convert your scripts into standalone .exe files!</li><li>• <b>AutoItX</b> – DLL/COM control. Add AutoIt features to your favorite programming and scripting languages! Also features a C# assembly and PowerShell CmdLets.</li><li>• <b>Editor</b> – A cut down version of the SciTE script editor package to get started. Download the package below for the full version!</li></ul>	

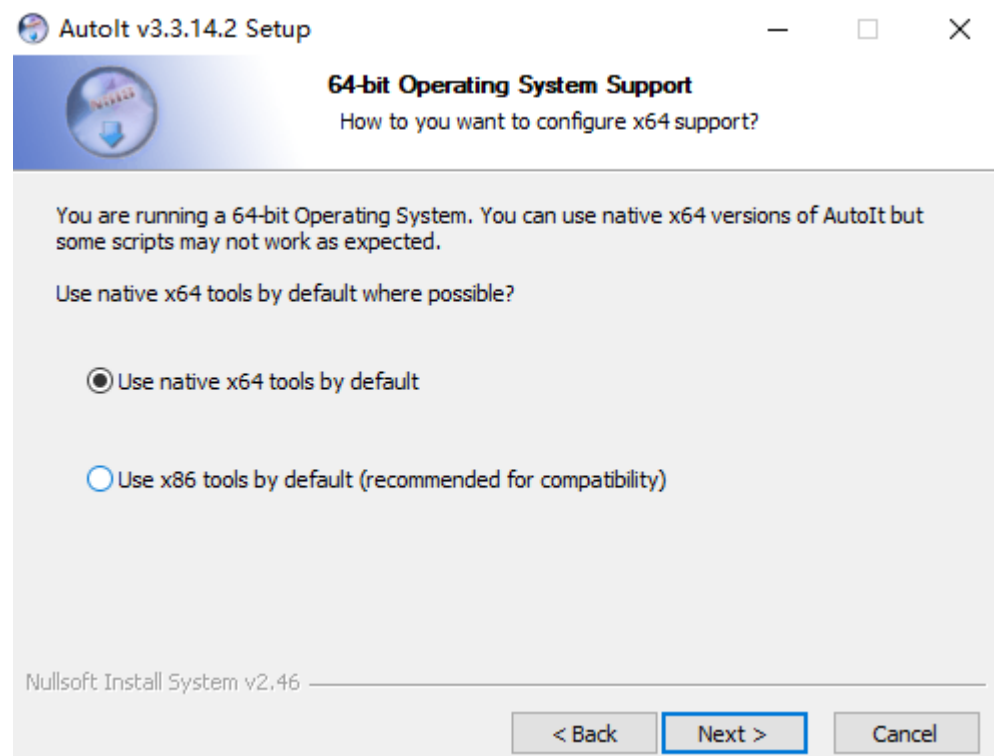
点击下载



点击 next

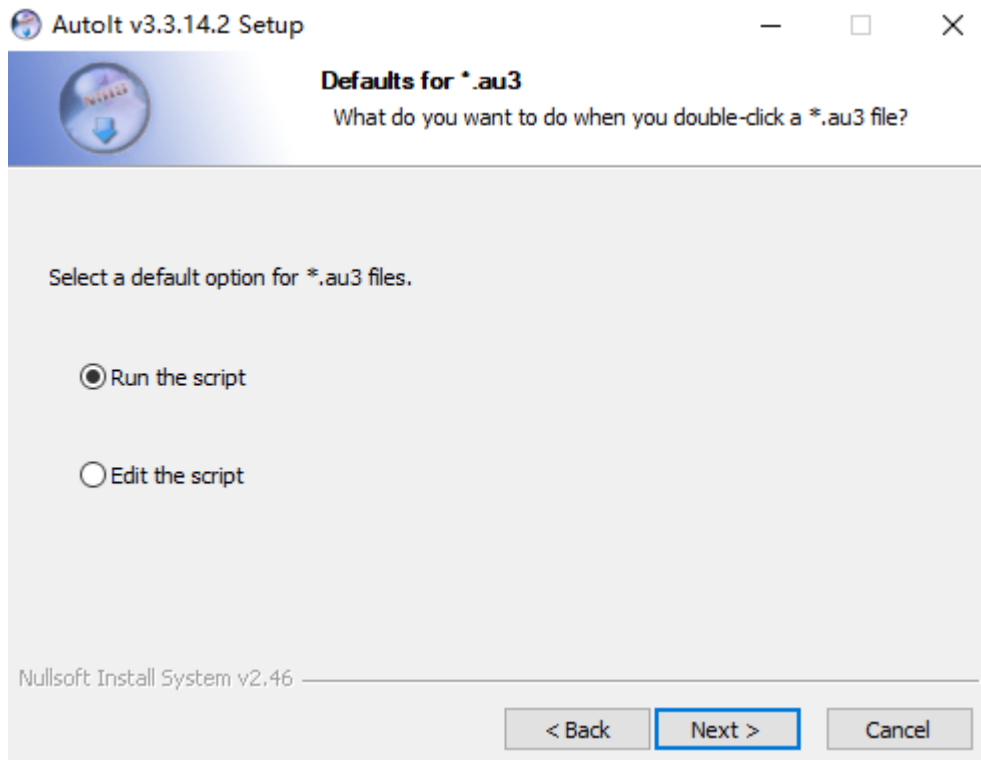


点击 I Agree

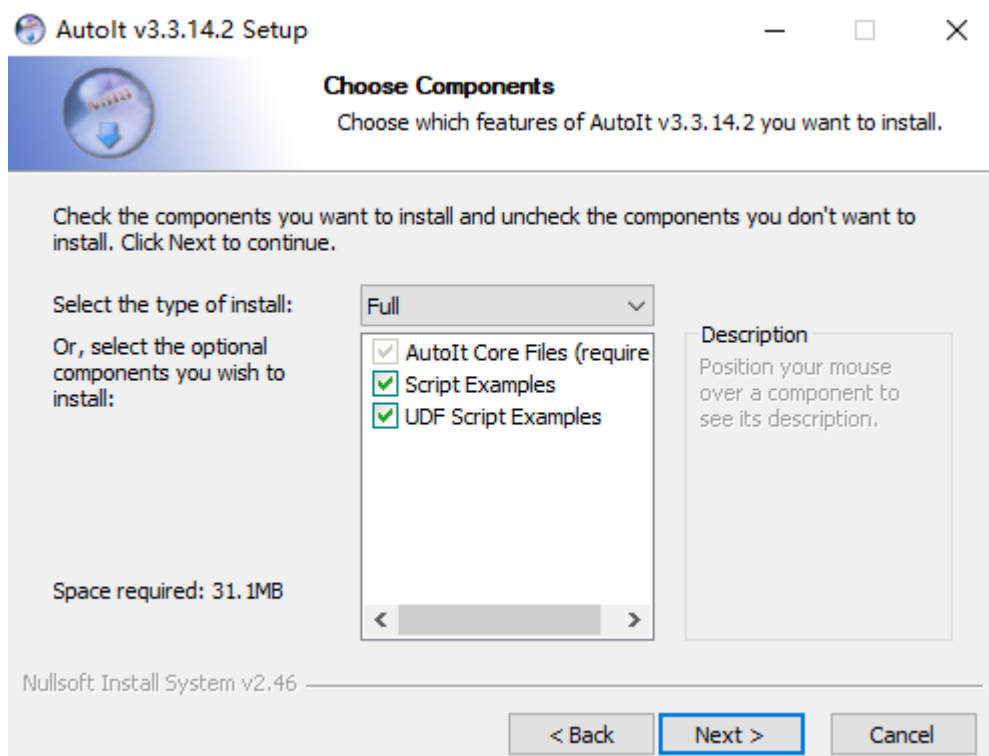


选择安装 64 位或者 32 位，我这里选择安装的是 64 位的 AutoIt，点击 next

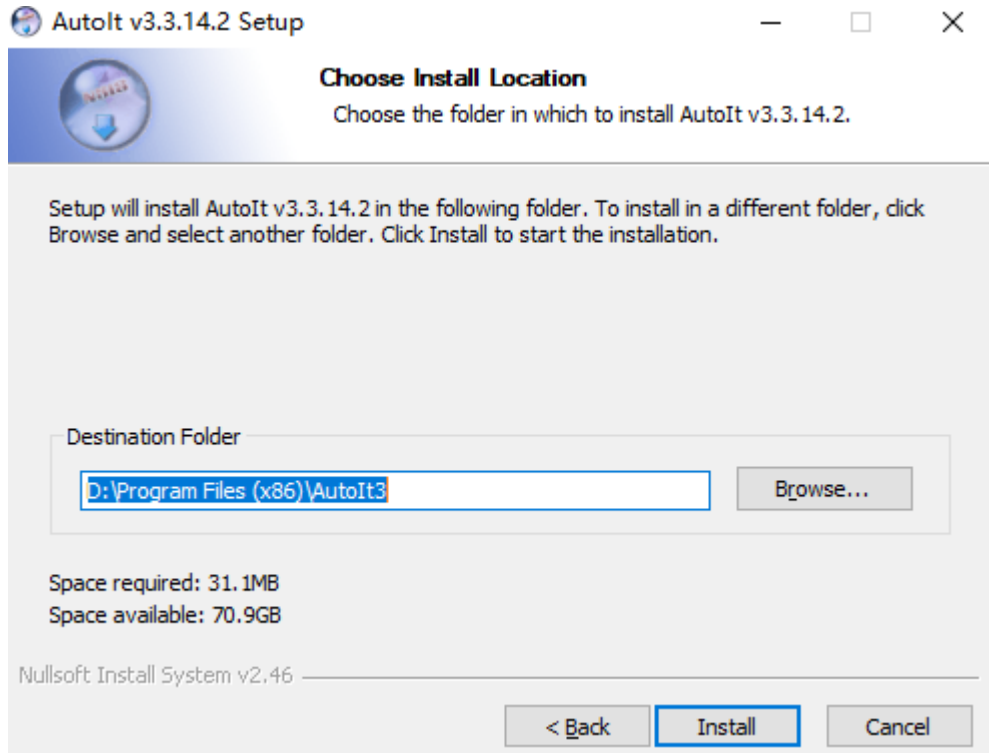




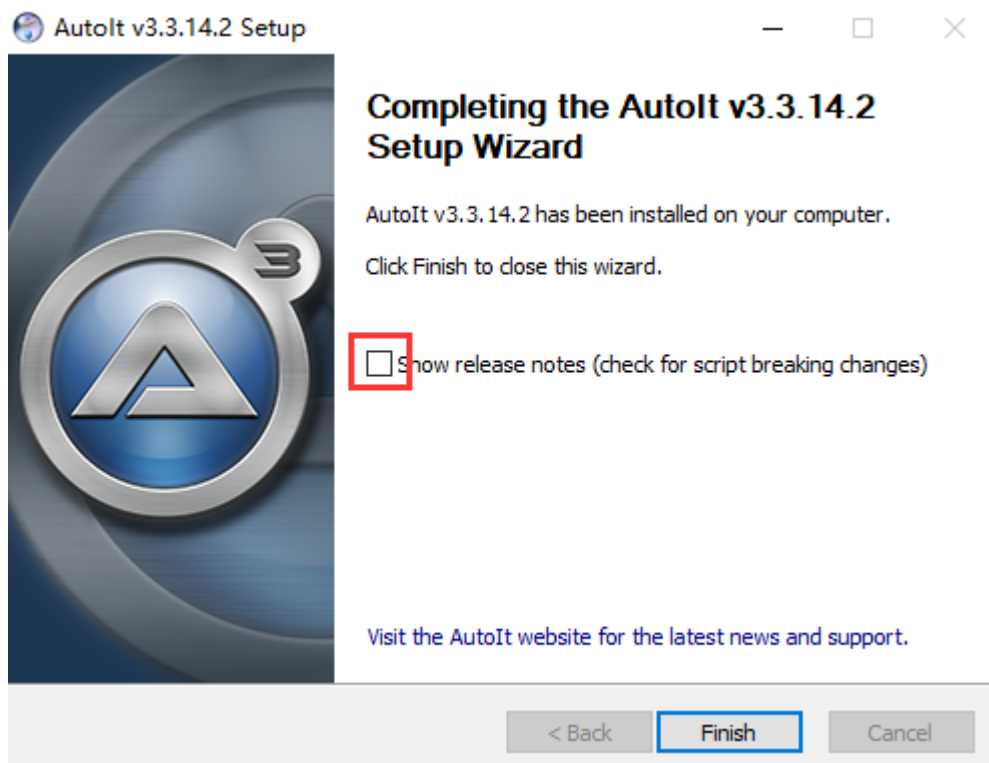
点击 next



点击 next



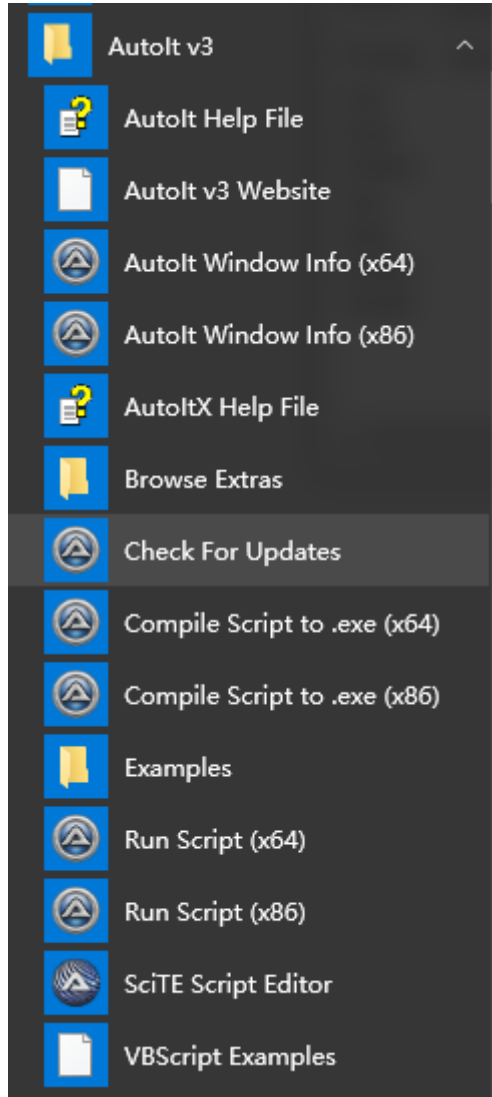
选择安装的路径，点击 install



这里的√去掉，点击 finish，这样就已经完成了

## 2.2.2Autolt3 使用

现在就来试试怎样去操作，去到“开始”菜单，



我们可以看到 Autolt 有以下文件，我们主要来看三个文件

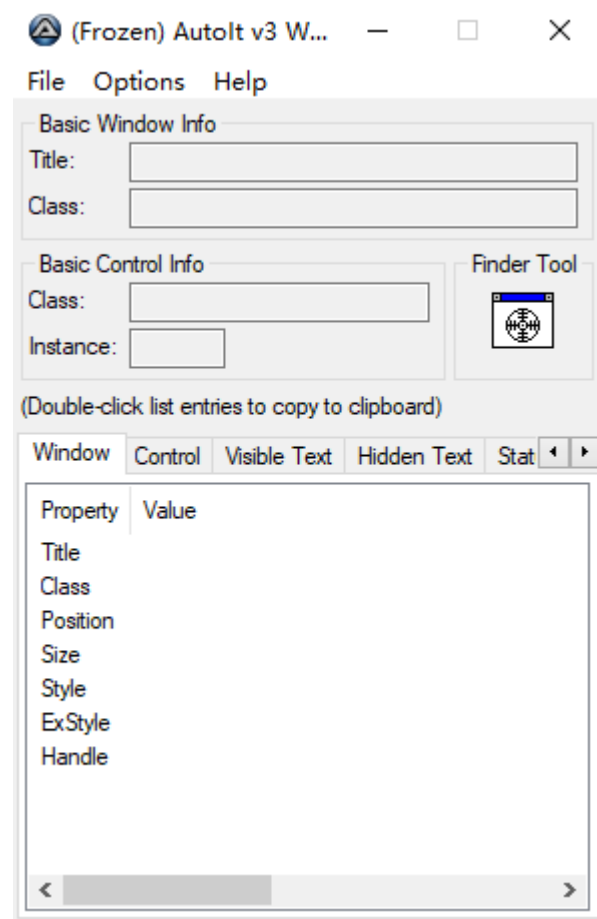
Autolt Window Info(x64)、Compile Script to .exe(x64)、SciTE Script Editor

第一个文件是用来录制我们操作的步骤，第二个文件是将我们的.au 文件转换成我们最终的.exe 文件，第三个是编辑我们.au 文件

这样一来，我们就基本了解 Autolt 的使用流程了

## 2.2.2.3 Autolt3 文件操作

首先我们需要打开 Autolt Window Info(x64)这个文件

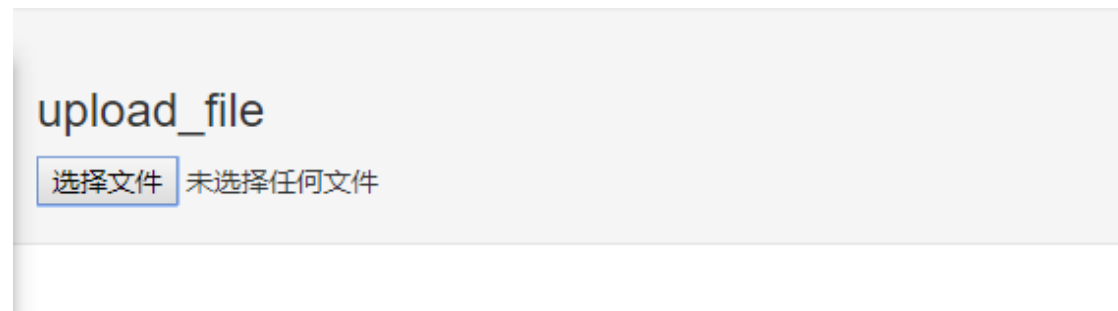


在 pycharm 里创建一个命名为 “upload” 的 HTML 文件，

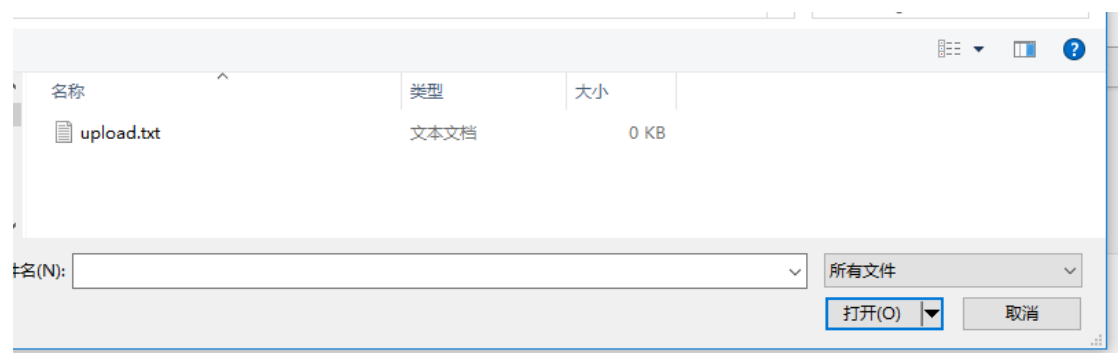
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>upload_Title</title>
<link
href="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.css"
rel="stylesheet" />
</head>
<body>
<div class="row-fluid">
  <div class="span6 well">
    <h3>upload_file</h3>
    <input type="file" name="file" />
  </div>
</div>
```

```
</div>
</div>
</body>
<script
src="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.js"></
script>
</html>
```

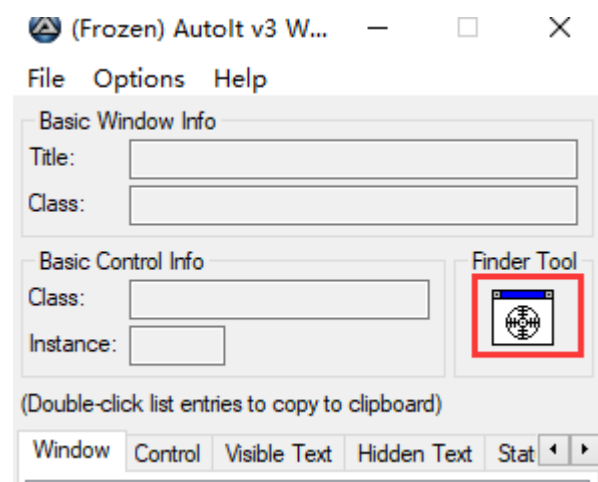
将这段代码复制粘贴，然后打开

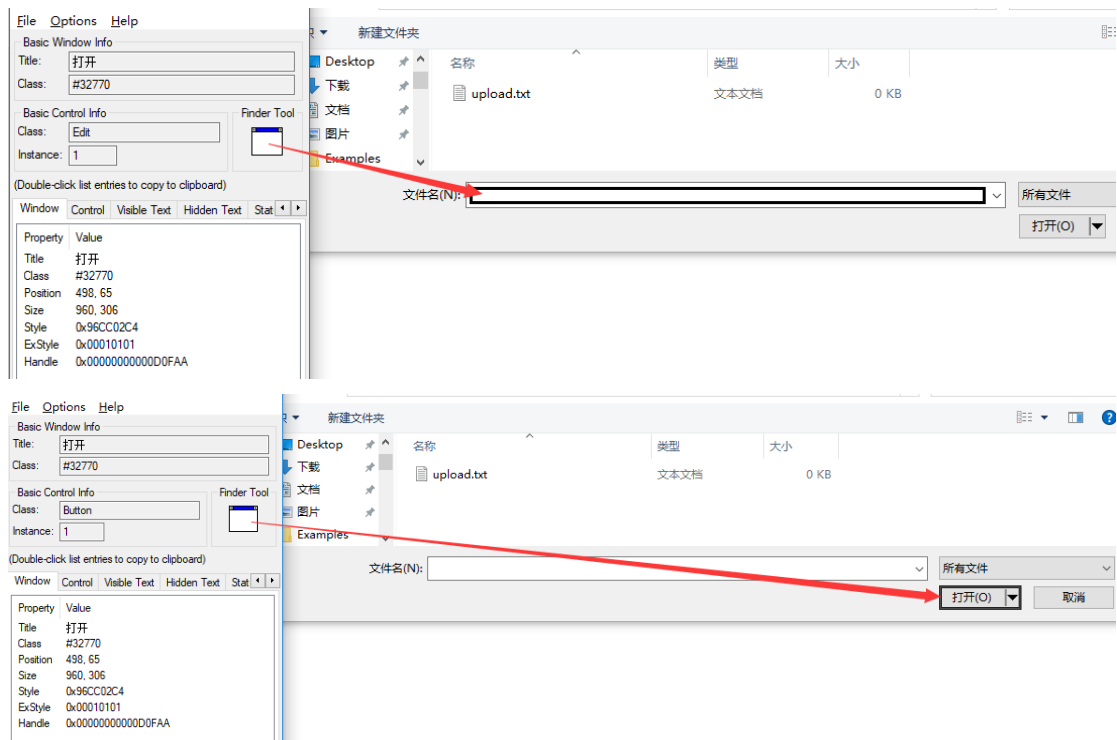


点击选择文件，会打开一个窗口



现在我们使用 autolt 工具来定位窗口的位置，拖动圆点





将移动到的位置的信息记录下来，点击 Control 可以看到信息

窗口的 title 为 “打开”，标题的 Class 为 “#32770”。

文件名输入框的 class 为 “Edit”，Instance 为 “1”，所以 ClassnameNN 为 “Edit1”。

打开按钮的 class 为 “Button”，Instance 为 “1”，所以 ClassnameNN 为 “Button1”。

这样我们就可以编辑脚本信息了，打开 SciTE Script Editor 编辑脚本信息

```
;ControlFocus("title","text",controlID) Edit1=Edit instance 1
```

```
ControlFocus("打开","", "Edit1")
```

```
; Wait 10 seconds for the Upload window to appear
```

```
WinWait("[CLASS:#32770]", "", 10)
```

```
; Set the File name text on the Edit field
```

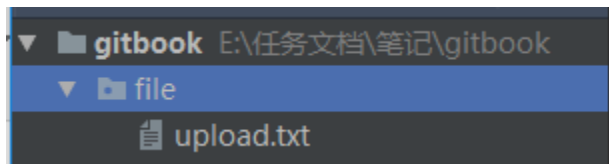
```
ControlSetText("打开","", "Edit1", " E:\\ upload.txt")
```

```
Sleep(2000)
```

; Click on the Open button

```
ControlClick("打开", "", "Button1");
```

这样的脚本有一个问题就是，我们的脚本路径已经写死了，这样对于我们后期的自动化测试是行不通的。所以我们必须将写死的路径改活。我们将 upload.txt 文件放在我们的自动化测试脚本里面，

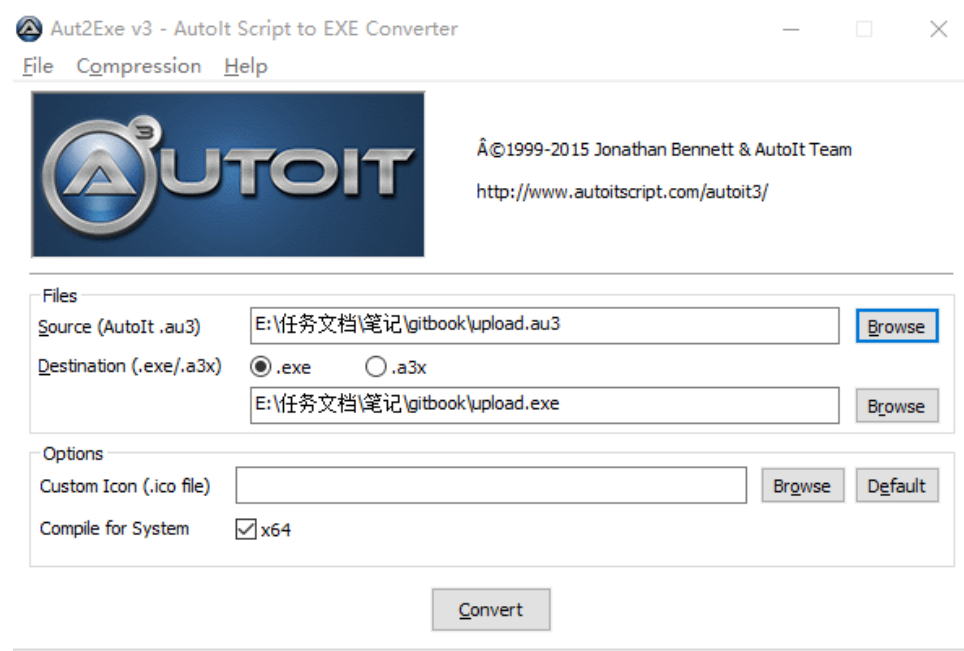


我们再去修改我们脚本的路径

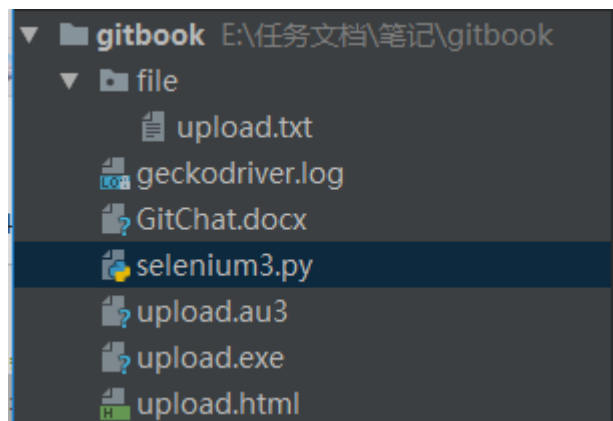
```
ControlSetText("打开", "", "Edit1", @WorkingDir & " \file\upload.txt")
```

这段代码的意思就是，打开当前文件夹上一个文件的相对路径，这样就可以防止我们的代码写死了。

这样我们就编辑好了脚本信息了，保存我们的文件，会生成.au 的文件，但是这个文件不是我们想要的，我们要得到的是.exe 的文件，打开 Compile Script to .exe(x64)文件，将.au 的文件转换成.exe 文件就可以了。



点击 Convert 就 ok 了，在当前文件夹下我们就可以看到生成的.exe 文件了



具体文件都放在这里了。现在我们可以去调用 upload.exe 文件了 新建命名为 uploadfile

的 python 文件

```
from selenium import webdriver
import os

driver = webdriver.Chrome()

#打开上传功能页面
file_path = 'file:/// ' + os.path.abspath('upload.html')
driver.get(file_path)

#点击打开上传窗口
driver.find_element_by_name("file").click()
#调用 upfile.exe 上传程序
os.system(os.path.abspath('upload.exe'))
driver.quit()
```

将这段代码输入进去 就可以调用了。这样一来 我们的 Autolt 学习就完成了。更多的 Autolt

语法，可以去官网看详细的文档。

## 2.2.3 BasePage 文件夹：

该文件夹的作用是对 Selenium 进行二次开发，使得 Selenium 适应我们当前的框架使用，

也让我们在后面更好的使用 selenium 的方法。目前对 selenium 进行二次开发的了大部分

的方法。但不是所有的方法都用的上，至少在我们需要用的时候，我们可以直接用上。强烈



建议读这部分的代码，部分代码如下。

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.select import Select
from selenium.common.exceptions import * #导入所有的异常类
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait

def browser(browser="Chrome"):
    """打开浏览器函数, Firefox, chrome, IE, phantomjs"""
    try:
        if browser == "Chrome":
            driver = webdriver.Chrome()
            return driver
        elif browser == "firefox":
            driver = webdriver.Firefox()
            return driver
        elif browser == "IE":
            driver = webdriver.Ie()
            return driver
        elif browser == "phantomjs":
            driver = webdriver.PhantomJS()
            return driver
        else:
            print("找不到驱动")
    except Exception as msg:
        print("%s" % msg)

class Sele(object):

    """基于原生 selenium 框架做了二次封装
    启动浏览器参数化，默认启动 Chrome"""
    def __init__(self, driver, base_url, pagetitle):
        self.base_url = base_url
        self.pagetitle = pagetitle
        self.driver = driver

    def on_page(self, pagetitle):
        return pagetitle in self.driver.title

    #打开页面，校验页面是否加载正确
    # 以下划线开头的方法，在使用 import *时，该方法不会被导入，保证该方法为类私有的。
```

```

def _open(self, url, pagetitle):
    #使用 get 打开访问链接地址
    self.driver.get(url)
    self.driver.maximize_window()
    #使用 assert 进行校验，打开的链接地址是否与配置的地址一致。调用 on_page() 方法
    assert self.on_page(pagetitle), "打开页面失败%s"%url

# 定义 open 方法，调用 _open() 进行打开链接
def open(self):
    self._open(self.base_url, self.pagetitle)
    self.driver.maximize_window()

#重写元素定位方法
def find_element(self, *loc):
    #return self.driver.find_element(*loc)
    try:
        #确保元素是可见的
        # 注意：以下入参为元组的元素，需要加*。Python 存在这种特性，就是将入参放在元组里。
        # WebDriverWait(self.driver, 10).until(lambda driver: driver.find_element(*loc).is_displayed())
        # 注意：以下入参本身是元组，不需要加*
        WebDriverWait(self.driver, 5).until(EC.visibility_of_all_elements_located(loc))

        return self.driver.find_element(*loc)
    except:
        print("页面元素未能找到%s"%self, loc)

#重写元素定位方法
def find_elements(self, *loc):
    #return self.driver.find_element(*loc)
    try:
        #确保元素是可见的
        # 注意：以下入参为元组的元素，需要加*。Python 存在这种特性，就是将入参放在元组里。
        # WebDriverWait(self.driver, 10).until(lambda driver: driver.find_element(*loc).is_displayed())
        # 注意：以下入参本身是元组，不需要加*
        WebDriverWait(self.driver, 5).until(EC.visibility_of_all_elements_located(loc))

        return self.driver.find_elements(*loc)

```

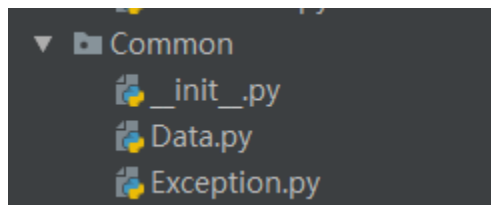
```

except:
    print("页面元素未能找到%s"%self, loc)

def click(self, locatort):
    """ 点击操作 """
    element = self.find_element(locatort)
    element.click()
def send_keys(self, locatort, text):
    """ 发送文本 """
    element = self.find_element(locatort)
    element.clear()
    element.send_keys(text)

```

## 2.2.4 Common 文件夹

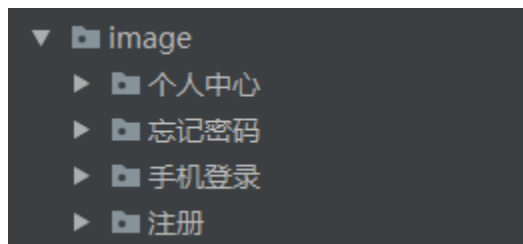


这个文件夹是用来处理异常类的，后来改成获取验证码的接口了，在获取验证码中，在前期我们经常碰到一个问题就是获取不到验证码，或者手机号获取验证码次数过多的情况，现在就将如果获取不到验证码就直接返回异常

使用的方法是 Python+requests 的 get 方法获取验证码的，和前面删除手机号的方法不一样，但是用的库是一样的。

## 2.2.5 image 文件夹

该文件夹是用来二次验证使用的，主要是为了防止我们在进行自动化测试的时候，返回的结果我们无法得到验证的情况，我们通过截图的结果得到进一步的验证，这样也保证了我们测试校验。

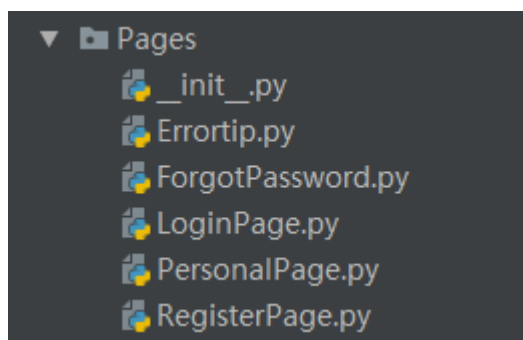


上面的命名文件夹都是由自己来命名的，截图的方法在后面会说到。

## 2.2.6 Lib 文件夹

建这个文件夹的目的是为了存放第三方的库，这样方便我们的脚本和系统分开，无论把我们的脚本放到哪都可以直接运行，只要当前电脑安装了 Python3 版本就可以直接运行我们的框架了。直到现在，我还没放进去。如果你准备熟悉这个框架之后，想继续扩展的话，建议把第三方库放在这里使用。

## 2.2.7 Pages 文件夹



这个文件夹使用存放页面元素的方法的，也就是 POM 中 pages，我们将页面中所有元素都放在当前文件夹，等测试用例来调用，方便了我们的维护以及更新。在敏捷开发中，提醒一点就是每天的开发速度都会很快，一个测试人员的工作量如果不安排好很容易出错，出错导致的后果是我们今天的工作量没有得到完成。

所有，我们将每一个测试页面的元素都放在这里，在后面开发中，新的页面就新建新的方法

类就可以了

目前里面包含的页面有：登录，注册，找回密码，个人中心

其中的方法：

```
from selenium.webdriver.common.by import By
from BasePages.Selenium2 import Sele
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import *
from Pages.ErrorTip import error

class LoginPage(Sele):
    #定位器，通过元素属性定位元素对象
    username_loc = (By.XPATH, '//*[@id="txt_user"]')#输入手机号/邮箱
    password_loc = (By.XPATH, '//*[@id="pwd_login"]')#输入密码
    submit_loc = (By.XPATH, '//*[@id="btn_login"]')#登录按钮
    Forgetpsw_loc = (By.XPATH, '//*[@id="link_forget_pwd"]')#忘记密码
    RegisterUrl_loc = (By.XPATH, '//*[@id="register"]')#点击注册 url
    tips_loc = (By.XPATH, '//*[@id="error_tips"]')#错误提示
    findtext_loc = (By.XPATH, '//*[@class="find-pwd-box"]/h1')#找回密码
    码页面文本
    regtext_loc = (By.XPATH, '//*[@class="user-explain"]/p[2]')#点击
    “注册” 文本
    # re_loc = (By.XPATH, '//*[@class="user-title"]')#"个人中心"文本校
    验
    logout_loc = (By.XPATH, '//*[@id="user_logout"]')#退出登录
    touxiang_loc = (By.XPATH, '//*[@id="user_head_avator"]')#个人信息
    icon

    # 操作
    # 通过继承覆盖(overriding)方法：如果子类和方法名相同，优先
    用子类自己的方法
    # 打开网页
    def open(self):
        # 调用 page 中的_open 打开链接
        self._open(self.base_url, self.pagetitle)

    #输入用户名：调用 send_keys 对象，输入用户名
    def input_username(self, username):
        self.find_element(*self.username_loc).send_keys(username)

    #输入密码：调用 send_keys 对象，输入密码
    def input_password(self, password):
```

```
        self.find_element(*self.password_loc).send_keys(password)

#点击登录：调用 click 对象，点击登录
def click_submit(self):
    self.find_element(*self.submit_loc).click()

# 点击登录：调用 click 对象，点击登录
def click_submit_key(self):
    self.find_element(*self.submit_loc).send_keys(Keys.ENTER)

#点击“忘记密码”：调用 click 对象，点击“忘记密码”
def click_psw(self):
    self.find_element(*self.Forgetpsw_loc).click()

#点击“注册”：调用 click 对象，点击“注册”
def click_reg(self):
    self.find_element(*self.RegisterUrl_loc).click()

#用户名或者密码不合理 tip 框内容提示
def show_tips(self):
    t = self.find_element(*self.tips_loc).text
    if t == error():
        assert error() in t

#登录成功页面中的用户 ID 查找
# def show_(self):
#     return self.find_element(*self.re_loc).text

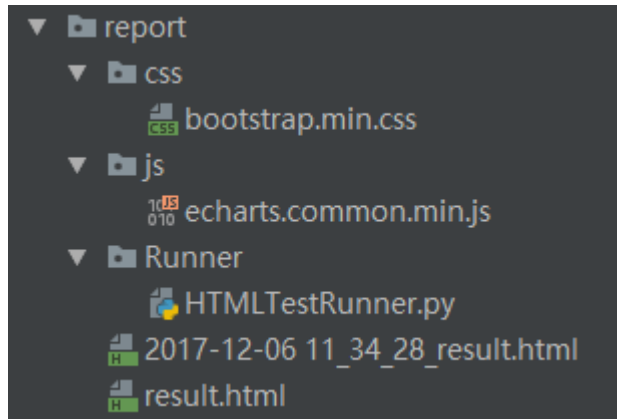
#退出登录
def logout(self):
    self.find_element(*self.touxiang_loc).click()
    self.find_element(*self.logout_loc).click()

#跳转注册页面关键字查找
def show_regtip(self):
    return self.find_element(*self.regtext_loc).text

#忘记密码关键字查找
def show_findtip(self):
    return self.find_element(*self.findtext_loc).text
```

## 2.2.8 report 文件夹：

这个文件夹不需要修改，因为这只是一个测试报告的文件夹



简单说一下这个文件夹，一个 css 文件，一个 js 文件，一个 HTMLtestrunner 文件，其中 css 和 js 文件都是提供给 HTMLtestrunner 文件使用的，当前 HTMLtestrunner 命名和 Python 中的 HTMLtestrunner 命名是一样的，所以，是不能用来使用的，如果要调用的话，重命名就可以了。

生成的测试报告是 result.html 这个文件，右键点击 run 就可以看到我们每次迭代的结果了。

包括错误都会详细的记录在报告中的，所以也不需要担心太多不知道哪里出错的问题了

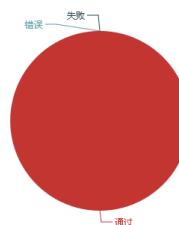
### 个人中心测试报告

开始时间: 2017-12-06 11:34:28  
运行时长: 0:00:34.830142  
状态: 通过 2

测试用例执行情况

通过  
失败  
错误

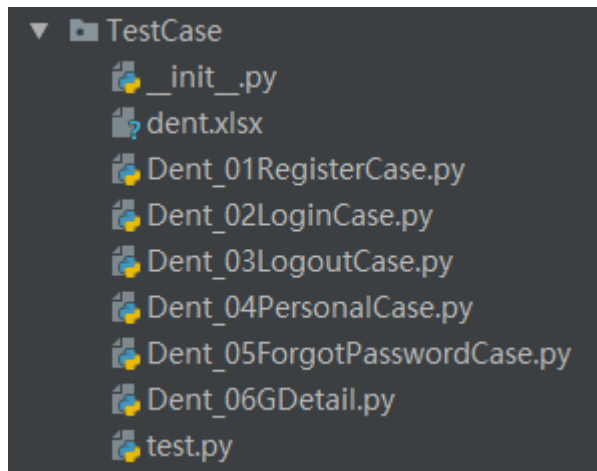
测试执行情况



总结 失败 全部

测试套件/测试用例	总数	通过	失败	错误	查看
SelfCase: 个人中心	2	2	0	0	<a href="#">详情</a>
总计	2	2	0	0	

## 2.2.9 TestCase 文件夹：



我们把每次要运行的步骤都写在这里，其中我们测试用例和 page 元素页命名尽量要一样，而且为了测试集的调用，我们每个文件的开头都一定要一样，否则在测试集中是无法被测试集执行

```
class LoginCase(unittest.TestCase):
    """登录"""

    def setUp(self):
        self.driver = browser()
        # cls.driver.implicitly_wait(5)
        self.url = LoginUrl()
        self.title = Title()

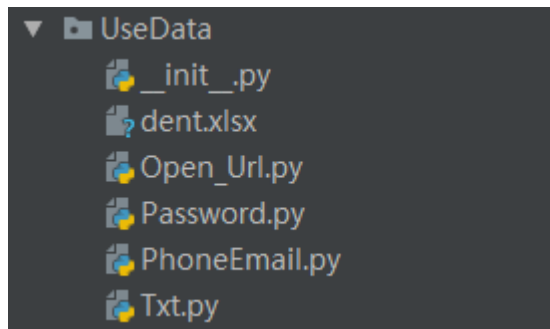
    def open_dent(self):
        login_page = LoginPage(self.driver, self.url, self.title)
        login_page.open()
        return login_page

    def test_login1(self):
        """手机登录成功"""
        # 声明LoginPage对象
        login_page = self.open_dent()
        # 调用用户名输入组件
        login_page.input_username(LoginPhone())
        # 调用密码输入组件
        login_page.input_password(Pwd())
        # 调用登录组件
        login_page.click_submit()
        # assert "个人中心" in login_page.show_()
```

书写的方法就跟我们基础的方法是一样的。



## 2.2.10 UseData 文件夹：



这个文件夹的作用是存储数据，将我们要使用到的数据都放在这里，这样可以提高可用性和可重复性的使用。其中，这里的方法是使用了 Python+openpyxl 的库来写的，当我们在校验的时候要使用到很多重复的数据，我们就可以直接在这里调用了。这里可以学习怎样去使用 openpyxl：

<http://openpyxl.readthedocs.io/en/default/>

当我们使用测试集对全部测试用例执行的时候，我们需要把 excel 文件放在最外层

## 2.2.11 index.py 文件：

这个就是将所有测试用例集合在在一起的测试集，运行这个文件就可以将所有的测试用例执行了

```
import os, unittest, time
from HTMLTestRunner import HTMLTestRunner
import sys
def create_suite():
    TestSuite = unittest.TestSuite() #测试集
    test_dir = os.getcwd() + '\\TestCase\\'
    # print(test_dir)

    discover = unittest.defaultTestLoader.discover(
        start_dir=test_dir,
        pattern='Dent*.py',
        top_level_dir=None
    )
```

```

    # print (discover)

    for test_case in discover:
        TestSuite.addTests(test_case)
    return TestSuite

def report():
    if len(sys.argv) > 1:
        report_name = os.getcwd() + '\\report\\' + sys.argv[1] +
'_result.html'
    else:
        now = time.strftime("%Y-%m-%d %H_%M_%S")
        report_name = os.getcwd()+'\\report\\result.html'
    return report_name

fp = open(report(), 'wb')
Runner = HTMLTestRunner(
    stream=fp,
    title='德雅官网测试报告',
    description='测试用例执行情况'
)

if __name__ == '__main__':
    TestSuite = create_suite()
    Runner.run(TestSuite)
    fp.close()

```

这里使用的方法是 Python+unittest+HTMLtestrunner

## 2.3 框架总结

以上就是我写的自动化测试框架了，不是很成熟的框架，对于初学者来说，这样的框架是足够了。

通过这个框架我们可以学到什么？

- 1、提升了我们 Python 的基础
- 2、学习到自动化测试的架构

3、还会学到其他我们没有接触过的库

## 框架用到的库：

- 1、在删除手机方法中，我们使用到 python 自带的 requests 库的 post 方法
- 2、在上传文件的方法中，我们使用到第三方的工具 Autolt
- 3、在 Selenium 二次开发中，我们使用到 Selenium 的库
- 4、在公共方法中，我们使用到 requests 库的 get 方法
- 5、在截图中，我们使用到 selenium 提供给我们的 screen 方法
- 6、在 pages 中，我们使用最多的是传参，定义参数
- 7、在 report 文件夹中，我们使用到了 HTMLtestrunner 库，目前 HTMLtestrunner 依然是没有支持 Python3 的，所以，我已经在测试机上存放了 HTMLtestrunner 库了，登录测试机，找到 python3 文件就可以搜索 HTMLTestRunner 库了
- 8、在 UseData 中，我们使用到了 python+openpyxl 的使用

## 三、写在后面

在学习自动化测试的路上，每天都是新的起步，只有不停下脚步，才能使自己进步。不要放弃自己的职业，追求个人的发展机会才是我们当前需要做的。无论后面的路怎么样，当我们走好了这一步，下一步也不会难走了。

## 附件一教程：

Python3 最新版本下载地址：<https://www.python.org/downloads/release/python-364/>

安装 Selenium :

pip install selenium

安装 requests :

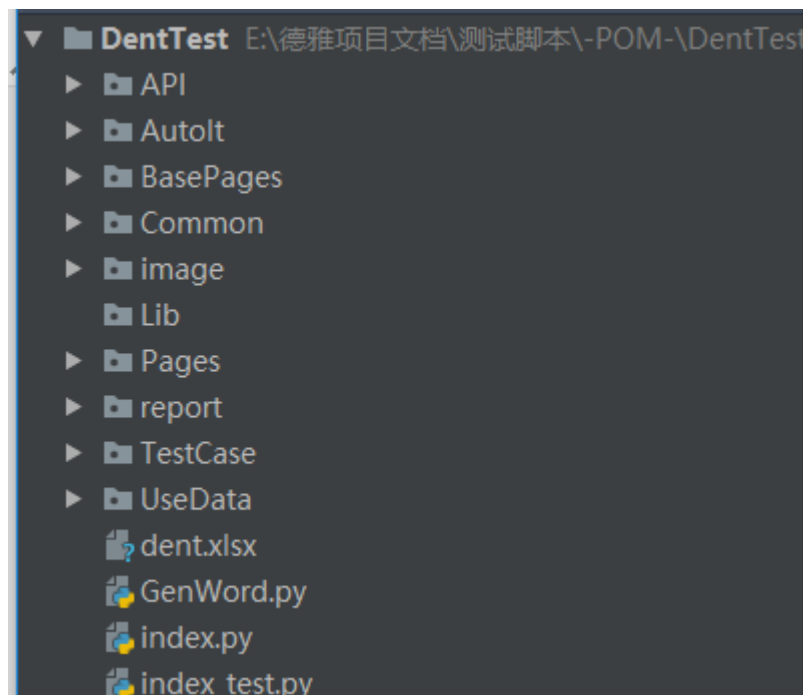
pip install requests

安装 openpyxl :

pip install openpyxl

问题一 :

直接打开框架, 发现导入部分出错了



解决方案 :

打开 DentTest 文件夹

问题二 :

pip 安装超时解决方案 :

首先检查 pip 的版本是否需要更新, 如果不是最新版本运行命令更新 :

python -m pip install --upgrade pip

如果仍然超时错误, 则运行一下命令 :

pip --default-timeout=100 install -U 库名

再次安装, 即可成功。

问题三 :

使用 openpyxl 时, 打开报错

解决方案 :

```
from openpyxl import load_workbook

wb = load_workbook("dent.xlsx")
sheet = wb.get_sheet_by_name("URL")
```

这里调用的是当前文件夹下的 excel 文件，所以，要把 excel 文件放在当前文件下，再去运行