**混合驱动样例详解**

远山近海

# 概述：

此混合驱动的样例为操作126邮箱，它是数据驱动和关键字驱动的结合。数据驱动可以把数据从程序中分离出来，做到每次调整用例时只修改数据文件而无需修改代码。关键字驱动可以把测试步骤的关键字从程序中分离出来，每次做到通过关键字就可以调用相关函数或类来执行用例。

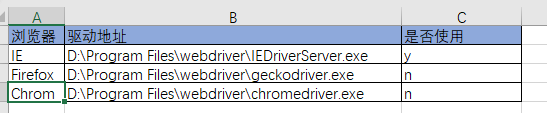
此混合驱动实现了数据和关键字同时从程序里分离，用例执行者通过在Excel中对各个关键字进行排列就可以执行用例，同时，每个关键字后面都跟有相应的xpath语句和输入数据以及等待时间，执行者完全不必接触代码就可以轻松完成用例的执行，但同时也要求用例执行者有编写xpath语句的能力。

此混合驱动还把代码按用途进行了较大限度的归类，如专门操作Excel的代码放到一个文件中，使用时统一调用；专门进行显示等待的代码放到一个文件中，使用时统一调用，等等。

# 实现结果抢先看：

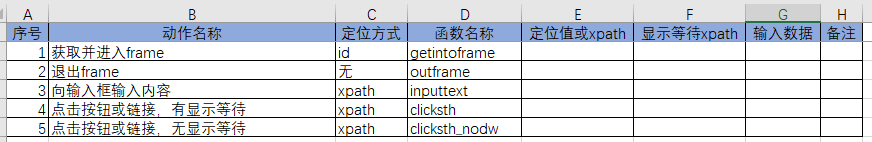
我们先从Excel文档中提前感受一下这个框架。我们打开Excel，看到里面有四个标签，从左至右分别为：驱动地址，关键字，用例，键盘清空。

## 一看驱动地址标签：



该标签中，定义了三个主流浏览器的驱动存放地址以及是否可用的标识。我们只需把相应的浏览器和其是否使用的标识标为y，该浏览器就会在程序执行时被打开。如果我们标识了多个y，程序只会使用从上到下数，第一个被标y的浏览器，因为在程序会循环遍历这个标签页里的所有数据行，只要遇到一次y就使用这个y所对应的浏览器，同时跳出循环。

## 二看关键字标签：



关键字标签定义了我们所需使用的操作动作，即关键字。我们在操作邮箱的时候，所做的分解动作无非就是两种，一种是向输入框输入内容，一种是点击。因为邮箱里一般不会有诸如下拉框、单选或多选按钮等元素。

但是，有些元素是被框在frame里的，我们在操作他们时需要先进入frame，操作完成后再退出frame，所以就多了进入和退出frame这两个动作。

点击这个动作又分了两种，带有显示等待的和没有显示等待的，为什么要这么分我们后续再来说明。

大家看到，这里的定位xpath、显式等待xpath、输入数据都是空的。因为程序实际并不读取这个标签里的数据，这个标签仅仅为使用者列出了动作的种类并给使用者提供一个存储动作和xpath的地方（如何存储，在下一个标签的说明部分详解），而真正需要被读取的是下一个标签。

## 三看用例标签：



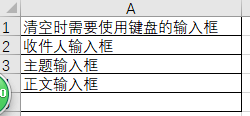
用例标签是最重要的，执行自动化代码时主要就是读取这个标签里的数据和关键字。在这里，程序通过函数名称来对应代码中相应的函数，定位xpath、显式等待xpath、输入数据、等待时间都是函数的参数，是否执行这个列里虽然不是参数，但程序也会取到它的值进行是否执行的判断，如果该列为y则执行该条用例，为n则跳过该条用例。

其中，定位xpath是指被定位元素本身的xpath，而该元素的动作完成后需要等待某元素显示出来再进行下一步操作，显式等待xpath就是指被等待元素的xpath。如点击登录按钮后，会跳转至邮箱首页，那么，我可以对邮箱首页上的写信按钮进行显式等待，写信按钮出现后才进行下一步操作，显式等待xpath就是指写信按钮的xpath。如果该项为空，则不调用显式等待函数

所以，我们可以通过从上到下填写相应的函数名称及其参数的方式来进行各种业务流程的组合。当然，这需要使用者有编写xpath的能力。

我们还可以把写好的步骤的关键字存储到关键字标签里去，那里面从定位方式列到输入数据列是与用例标签对应的，我们在用例标签里写好后直接粘贴过去即可，并在关键字标签的动作名称列里写清楚这个动作是做什么的。同样，复用时我们从关键字标签里把需要几列直接复制粘贴到用例标签里即可。

## 四看键盘清空标签：



这个标签比较简单，程序实际也不读取它里面的数据，它仅仅记录了哪些输入框无法使用正常的.clear()方法清空而需要使用键盘事件进行清空。

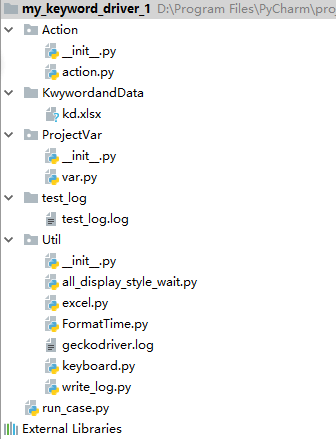
大家是否注意到，在用例标签中有一列叫做是否键盘清空，如果这一列为y，则程序调用键盘事件对其进行清空操作，如果这一列为空则使用普通的.clear()方法进行清空。

所以，大家在用例标签里编写用例的时候，当遇到了输入框，就要先到键盘清空标签里看一看该输入框是否需要调用键盘事件。

# 代码详解

## 工程结构及其释义

先看一下截图，有个感性认识：



## 包与目录的功能概述

Util包里存储了公用的文件，如操作Excel的文件，操作显式等待的文件等。

test\_log目录里存储了日志文件，用于存储日志。

ProjectVar.包里存储了工程所需的所有路径。

KwywordandData目录里存储了关键字及数据所在的Excel文档，就是我们在实现结果抢先看章节中所介绍的Excel文档。

Action包里存储了所有的操作动作，即关键字，如进入或退出frame的，向输入框输入数据的，点击按钮的等等。

run\_case.py是主程序，通过执行它来执行整个自动化框架。

## 各文件详解

### ProjectVar包

#### var.py

用于存储该工程里所有需要被调用的文件的路径

*#coding=utf-8***import** os  
**import** platform  
  
*# excel路径：*excel\_path=os.path.dirname(os.path.dirname(\_\_file\_\_))+**"\\KwywordandData\\kd.xlsx"***# 日志路径*log\_path= os.path.dirname(os.path.dirname(\_\_file\_\_)) +**"\\test\_log\\test\_log.log"**

注意：该文件里使用了os.path.dirname(\_\_file\_\_)方法，该方法用于获取该文件的所在目录的上一级目录。为什么不使用os.getcwd()？因为os.getcwd()是以当前输入焦点所在的目录为基准的，而os.path.dirname(\_\_file\_\_)是以当前执行文件所在的目录为基准的，不论执行文件被放到哪里都不会有影响

### Util包

#### excel.py

用于读取excel

*# coding=utf-8***from** openpyxl **import** Workbook,load\_workbook  
**import** string  
**from** FormatTime **import** date\_time\_chinese  
  
**class** Excel\_r\_w():  
 **def** \_\_init\_\_(self,excel\_path,sheet\_name):  
 self.excel\_path=excel\_path  
 self.wb=load\_workbook(excel\_path)  
 self.ws=self.wb[sheet\_name]  
 *# 以下为获取多个列的列号组成的列表，不论列是否有数据* uper\_str = string.uppercase *#生成所有大写字母* self.col\_no\_list = map(**lambda** x: x, uper\_str) *# 生成基础列号列表，即["A","B"..."Z"]* **for** iA **in "ABCD"**: *# 在基础列号列表后面加上更多的列，即["A"..."Z"；"AA","AB"..."AZ"；"BA","BB"..."BZ"..."DZ"]* **for** iu **in** uper\_str:  
 self.col\_no\_list.append(iA + iu)  
  
 *# 获取sheet中的最大的行数* **def** get\_max\_row(self):  
 **return** self.ws.max\_row  
  
 *# 获取sheet中的最大列数* **def** get\_max\_col(self):  
 sol\_no=self.ws.max\_column *# 先获得列号* **return** filter(**lambda** x:x==self.col\_no\_list[sol\_no-1],self.col\_no\_list)[0] *# 再转成字母  
  
 # 获取sheet的的最小（起始）行号* **def** get\_min\_row(self):  
 **return** self.ws.min\_row  
  
 *# 获取sheet的最小（起始）列号* **def** get\_min\_col(self):  
 sol\_no=self.ws.min\_column *# 先获得列号* **return** filter(**lambda** x:x==self.col\_no\_list[sol\_no-1],self.col\_no\_list)[0] *# 再转成字母  
  
 # 获取sheet的某一个单元格的值,index为坐标，如"A1","C5"* **def** get\_value(self,index):  
 **return** self.ws[index].value  
  
 *# 通过坐标写入内容，index为坐标，如"A1"，"F7"，content为写入的内容，但不保存* **def** write\_content(self,index,content):  
 self.ws[index]=content  
  
 *# 通过坐标写入内容，index为坐标，如"A1"，"F7"，content为写入的内容，写入后自动保存* **def** write\_content\_save(self,index,content):  
 self.ws[index] = content  
 self.wb.save(self.excel\_path)  
  
 *# 通过坐标写入当前日期* **def** write\_datetime(self,index):  
 self.ws[index]=date\_time\_chinese()  
  
 *# 保存，和write\_content和write\_datetime配合使用* **def** save\_content(self):  
 self.wb.save(self.excel\_path)  
  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 ins=Excel\_r\_w(**"D:\\test\\sample.xlsx"**,**"Sheet1"**)  
 **print** ins.get\_max\_col()  
 **print** ins.get\_min\_col()

注意：生成最大和最小列号的函数返回的字母，不是数字，即A,BA等。而读取和写入数据的函数也都是通过坐标来操作的，即A1,D7等。

如果大家觉得这里面封装的方法有需要完善的地方，也欢迎大家补充后跟帖发上来，大家一起添砖加瓦，资源共享。

#### FormatTime.py

用于生成当前时间

*#encoding=utf-8***import** time  
**from** datetime **import** timedelta,date

**def** date\_time\_chinese():  
 *u"returns the current time string,format for YYYY年mm月dd日 HH时MM分SS秒"* **return** time.strftime(**"%Y年%m月%d日 %H时%M分%S秒"**,time.localtime())

**def** date\_time():  
 **return** time.strftime(**"%Y-%m-%d %H:%M:%S"**,time.localtime())

#### write\_log.py

用于写日志

*# coding=utf-8***from** ProjectVar.var **import** log\_path  
**from** Util.FormatTime **import** date\_time  
  
**class** Writelog():  
 **def** \_\_init\_\_(self,log\_path):  
 self.log\_path=log\_path  
 self.fp=open(log\_path,**"a"**)  
  
 **def** writelog(self,\*args):  
 self.fp.write(date\_time())  
 self.fp.write(**","**)  
 **for** i **in** args:  
 self.fp.write(i)  
 self.fp.write(**","**)  
 self.fp.write(**"\n"**)  
  
 **def** closefp(self):  
 self.fp.close()  
  
**if** \_\_name\_\_==**"\_\_main\_\_"**:  
 **from** Util.excel **import** \*  
 **from** ProjectVar.var **import** \*  
 logpath = **"D:\\test\\test.log"** wl=Writelog(logpath)  
 wl.writelog(**"hello2"**)  
 wl.closefp()  
 *#writelog(date\_time()+ ",用户" + userins.get\_value("B2") + "创建了联系人：" + eins.get\_value("B2").decode("utf-8"))*

该文件定义了一个用于写日志的类，类的构造方法里有一个传参，那就是日志文件的路径。类里还有两个类方法，一个用来打开日志文件，写入日志，一个用来关闭日志文件。

大家注意到，这里使用的是不能自动关闭的open()方法，这需要用.close()方法来进行关闭，那么，为什么不使用可以自动关闭的更为方便的with open()方法呢？因为我们没执行一条用例都会往日志文件里写入数据，如果使用with open()方法，则会每次写入一条数据就关闭一次，这样IO操作过于频繁，影响效率。所以，使用了需手动关闭的open()方法，我们可以在所有用例执行完毕后统一进行关闭。

大家注意到，写入日志的方法的传参是一个可变参数，即元组，为什么不传一个固定的参数呢？原因有二：

1. 日志并没有统一的格式，日志是由每个项目依照自己项目的情况自行定义的，所以，写成可变参数就方便项目组人员自定义日志内容。
2. 我们在这个例子中所传入的参数包含很多项，它们分别有不同的来源，并且有着不同的编码格式，如果参数是固定的一个，那就需要把这些不同编码格式的字符串拼接成一个大字符串，如此众多的编码格式一起拼接很容易造成编码错误，所以使用可变参数来进行传参，每次只写入一种编码格式的一个字符串就避免了这个问题。

也许有人说了，我仍然想传入多个参数，但又想让传入的参数固定，而不是像可变参数那样可以随便传，传多少，传什么都行，那么我们可以在函数内部把传入的参数组成一个元组或列表，如：

**def** writelog(self,A,B,C,D):  
 **for** i **in** (A,B,C,D):  
 self.fp.write(i)  
 self.fp.write(**","**)

#### all\_display\_style\_wait.py

设置了显式等待的相关函数

*#coding=utf-8***from** selenium.webdriver.common.by **import** By  
**from** selenium.webdriver.support.ui **import** WebDriverWait  
**from** selenium.webdriver.support **import** expected\_conditions **as** EC  
  
*# 判断frame是否可用，如可用，则返回True并切入frame，不可用则返回False  
# driver为浏览器的实例化对象，ws为等待的秒数，es为ws时间段内每隔多少毫秒判断一次frame是否可用  
# tuple有两个值，一个是查找方式，可以By.ID，也可以 By.XPATH，另一个是该frame的id  
# tuple的第一个值如果是By.NAME，那么第二个值就是该frame的NAME  
# 例1：frame\_isusabled(driver,10,0.2,(By.ID, "x-URS-iframe"))  
# 例2：frame\_isusabled(driver, 10, 0.2,(By.XPATH,"//iframe[@id='x-URS-iframe']"))***def** frame\_isusabled(driver,ws,es,tuple1):  
 wait = WebDriverWait(driver,ws,es)  
 **return** wait.until(EC.frame\_to\_be\_available\_and\_switch\_to\_it(tuple1))  
  
  
*# 判断某元素是否出现，如在规定的时间内出现，则返回True，代码继续往下执行，否则返回False  
# 可通过try except捕获异常的方式来设置如果被等待元素未出现时如何处理  
# driver为浏览器实例化对象，ws为等待的秒数，es为ws时间段内每隔多少秒（通常为0.2，即200毫秒）查看一次该元素是否出现***def** isvisible(driver,ws,es,thisxpath):  
 wait=WebDriverWait(driver,ws,es)  
 thiselement = driver.find\_element(**"xpath"**, thisxpath)  
 vc\_thiselement=wait.until(EC.visibility\_of(thiselement))  
 **if** vc\_thiselement==thiselement:  
 **return** True  
 **else**:  
 **return** False  
  
  
**if** \_\_name\_\_==**"\_\_main\_\_"**:  
 **pass**

这里定义了两个函数，一个是用来判断并切入frame的，另一个是用来判断页面上某元素是否已显示的。

#### keyboard.py

定义了键盘操作事件

*# coding=utf-8***import** win32api  
**import** win32con  
  
VK\_CODE = {  
 **'enter'**: 0x0D,  
 **'ctrl'**: 0x11,  
 **'a'**: 0x41,  
 **'v'**: 0x56,  
 **'x'**: 0x58,  
 **'del'**: 0x2E  
}  
*# 键盘键按下,keyName为按键名称，即VK\_CODE字典里的key，其中0,0,0 为固定写法***def** keyDown(keyName):  
 win32api.keybd\_event(VK\_CODE[keyName], 0, 0, 0)  
  
*# 键盘键抬起,keyName为按键名称，即VK\_CODE里的key***def** keyUp(keyName):  
 win32api.keybd\_event(VK\_CODE[keyName], 0, win32con.KEYEVENTF\_KEYUP, 0)

在这里定义了两个函数，键盘按下和键盘抬起，函数外的字典定义了常用的几个按键的值，两个函数的参数就是字典的key

### test\_log目录

#### test\_log.log

仅仅是一个普通文件，它是通过我们建立一个.txt文件并把后缀改成.log后而来的。

### KwywordandData目录

#### kd.xlsx

用来存放数据和关键字，前面已进行过描述，这里不再赘述。

### Action包

#### action.py

这里定义了关键字

*# coding=utf-8***import** time  
**import** sys  
**from** Util.excel **import** \*  
**from** ProjectVar.var **import** \*  
**from** Util.all\_display\_style\_wait **import** \*  
**from** Util.keyboard **import** \*  
  
**def** bowser\_driver():  
 ed=Excel\_r\_w(excel\_path,**u"驱动地址"**)  
 **for** ir **in** range(2,5):  
 **if** ed.get\_value(**"C"**+str(ir)).encode(**"utf-8"**).strip()==**"y"**:  
 **if** ed.get\_value(**"A"**+str(ir)).encode(**"utf-8"**).strip()==**"IE"**:  
 driver=webdriver.Ie(executable\_path=ed.get\_value(**"B"** + str(ir)).encode(**"utf-8"**))  
 **if** ed.get\_value(**"A"**+str(ir)).encode(**"utf-8"**).strip()==**"Firefox"**:  
 driver=webdriver.Firefox(executable\_path=ed.get\_value(**"B"** + str(ir)).encode(**"utf-8"**))  
 **if** ed.get\_value(**"A"**+str(ir)).encode(**"utf-8"**).strip()==**"Chrom"**:  
 driver = webdriver.Chrome(executable\_path=ed.get\_value(**"B"** + str(ir)).encode(**"utf-8"**))  
 **return** driver  
  
*# 获取并进入frame***def** getintoframe(driver,tuple1):  
 frame\_isusabled(driver, 10, 0.2, tuple1)  
  
*# 退出frame***def** outframe(driver):  
 driver.switch\_to.default\_content()  
  
*# 向输入框输入内容，用clear()清空***def** inputtext(driver,thisxpath,content):  
 thistext=driver.find\_element(**"xpath"**,thisxpath)  
 thistext.click()  
 thistext.clear()  
 thistext.send\_keys(content)  
  
*# 向输入框输入内容，用键盘清空***def** inputtextkb(driver,thisxpath,content):  
 thistext = driver.find\_element(**"xpath"**, thisxpath)  
 thistext.click()  
 keyDown(**"ctrl"**)  
 keyDown(**"a"**)  
 keyUp(**"a"**)  
 keyUp(**"ctrl"**)  
 keyDown(**"del"**)  
 keyUp(**"del"**)  
 thistext.send\_keys(content)  
  
*# 点击按钮（含显示等待）***def** clicksth(driver,thisxpath,vcxpath):  
 driver.find\_element(**"xpath"**, thisxpath).click()  
 **if** isvisible(driver, 10, 0.2, vcxpath)==1:  
 **pass  
 else**:  
 **print u"有元素未找到"***# 点击按钮（不含显示等待）***def** clicksth\_nodw(driver,thisxpath):  
 driver.find\_element(**"xpath"**, thisxpath).click()  
  
*# 显示等待***def** display\_wait(driver,vcxpath):  
 **if** isvisible(driver, 10, 0.2, vcxpath)==1:  
 **pass  
 else**:  
 **print u"元素未找到"** sys.exit()  
  
*# 退出frame***def** outframe():  
 driver.switch\_to.default\_content()  
  
**if** \_\_name\_\_==**"\_\_main\_\_"**:  
 **from** selenium **import** webdriver  
 driver=webdriver.Ie(executable\_path=**r"D:\Program Files\webdriver\IEDriverServer.exe"**)  
 driver.get(**"https://www.baidu.com/"**)  
 clicksth(driver, **"//a[@href='http://news.baidu.com' and .='新闻']"**, **"//a[.='举报']"**, 2)

##### bowser\_driver()方法

定义了浏览器的驱动实例在这里，我们调用了excel.py里面的get\_value()方法，用于取kd.xlsx中驱动地址标签里的数据。首先，取出C列的值，如果是y，则根据浏览器名称及地址来获得webdriver的实例，一旦实例获取完毕，则返回实例，跳出循环。

如在本例中，使用了IE浏览器，那么调用driver=webdriver.Ie(executable\_path=ed.get\_value(**"B"** + str(ir)).encode(**"utf-8"**))这条语句，并返回driver。

##### getintoframe(driver,tuple1) 和outframe(driver)方法

用于进入和退出frame，都是调用的all\_display\_style\_wait.py文件里的函数

##### inputtext(driver,thisxpath,content)方法

用于向输入框输入数据，它需传入三个参数，driver对象，输入框的xpath（thisxpath）和输入的数据（content）

它的执行顺序是，先利用thisxpath找到输入框元素，点击输入框，清空输入框，最后输入content

为什么要有点击和清空这步？因为如果不点击的话，有时候焦点不会切入输入框，而输入框有时会有默认值，所以需要清空。

##### inputtextkb(driver,thisxpath,content)方法

也是用于向输入框输入数据，只是它清空输入框的方法是使用的键盘事件。它的操作顺序是先点击该输入框，按下ctrl→按下a→抬起a→抬起ctrl（此时，输入框里的默认值被全选）→按下del，此时，输入框被清空，最后再输入内容。

##### clicksth(driver,thisxpath,vcxpath)方法

用于点击按钮或链接，它先是使用.click()方法点击了按钮，再调用了all\_display\_style\_wait.py文件里的显式等待方法isvisible()，如果返回值为True即为1，则通过，否则输出一句话。

##### clicksth\_nodw(driver,thisxpath)方法

也用于点击按钮或链接，但它没有显式等待。

##### display\_wait(driver,vcxpath)方法

一个专门用于显式等待的方法。licksth\_nodw(driver,thisxpath)和display\_wait(driver,vcxpath)两个方法实际就是对clicksth(driver,thisxpath,vcxpath)方法的分步拆解。为什么要分开，我们会在run\_case.py文件的讲解中详细阐述

### run\_case.py文件

#### 整体说明

该文件是框架的主程序，运行该框架实际就是运行这个文件。我们对这个文件先进行由浅入深的伪代码释义：

首先，这个文件的核心就是读取kd.xlsx中用例标签里的数据，并且把读出的数据以字符串的形式拼接成一个大字符串action，最后用exec(action)函数来执行这个大字符串。我们可以写一个循环，用来读取这些数据，在这个循环中，用if else的方式对特殊情况进行处理。

#### 大体思路

1. 浏览器实例化
2. 打开126邮箱登录页
3. 窗口最大化
4. 开始大循环：
   1. 先通过是否执行列判断该步骤是否应执行
   2. 再判断是否使用键盘事件，如果需要使用，则调用inputtextkb函数，拼接相应的action字符串，否则调用inputtext函数，拼接相应的action字符串。

由于键盘事件只针对输入框，所以，非输入框的操作不能在该列写入y，这需要用例执行人员人为控制

* 1. 判断函数名称是否是outframe，是则直接执行action字符串，并对Excel文档进行通过与未通过的输入、写入日志，结束本次循环，进入下一次
  2. 判断函数名称是否是getintoframe，如果是，则拼接相应的action字符串，如果否，则进入步骤5
  3. 循环HIJ这三个列，如果H和I列为空，则进入下一次循环，因为为空的情况已经在前面判断过了。如果是最后一次循环，即循环到J列，则拼接action，如果循环的不是J列，同样拼接action，两次拼接的不同在于，J列存储的是所输入的数据，可能出现中文，所以在数据前面加了个u，循环到非J列则无需加u。
  4. 执行action字符串，并对Excel文档进行通过与未通过的输入、写入日志，结束本次循环，进入下一次
  5. 大循环结束

1. 保存Excel
2. 关闭写入日志句柄
3. 退出浏览器

#### 伪代码

driver=XXX

打开126邮箱

窗口最大化

实例化Excel

生成写入日志的句柄

循环，从第二行到最后一行：

如果Ln值为n：

证明该用例不执行，将Mn即执行结果单元格清空

下面的代码不用执行了，直接进行下一次循环

如果En值为y：

则需要使用键盘事件，拼一个字符串，action=

"**inputtextkb"**+**"(driver,**"

否则：

action=Gn的值+**"(driver,"**

如果Gn的值="outframe"：

action=**"driver.switch\_to.default\_content()"**即退出frame

try:

**exec** (action)即执行action字符串，用以退出frame

向Mn单元格输入“通过”

写入日志

except:

向Mn单元格输入“未通过”

写入日志

下面的代码不用执行了，直接进行下一次循环

如果Gn的值="getintoframe"：

action= action + ("By.XPATH,",Hn的值)

否则：

循环H到J：

如果Hn单元格为空：

则证明该函数已经在上面的if else语句中执行过了，所以，下面的代码不用执行了，直接进行下一次循环

如果循环到到了第J列：

action= action + u"Jn"), 注意，Jn是指Jn的值，不是字符串"Jn"，因为J列代表的是输入数据，可能有中文，所以前面加个u，字符串最后要加一个逗号

否则：

action= action+该单元格的值, 注意，这个否则里的action和上面如果里的action的区别就在于这个action里不需要加u，因为除了J以外，H和I都是xpath语句，没有中文的问题，另外，字符串最后仍然别忘加个逗号

action=action[0:-1]+")" 因为每一次循环拼接的字符串后面都有一个逗号而且没有反括号，所以需要执行这一句。最终，这个H到J的循环达到的效果就是action=Gn值(Hn值，In值，u"Jn值")。

try:

**exec** (action)即执行action字符串，用以退出frame

向Mn单元格输入“通过”

写入日志

except:

向Mn单元格输入“未通过”

写入日志

保存Excel

关闭日志写入句柄

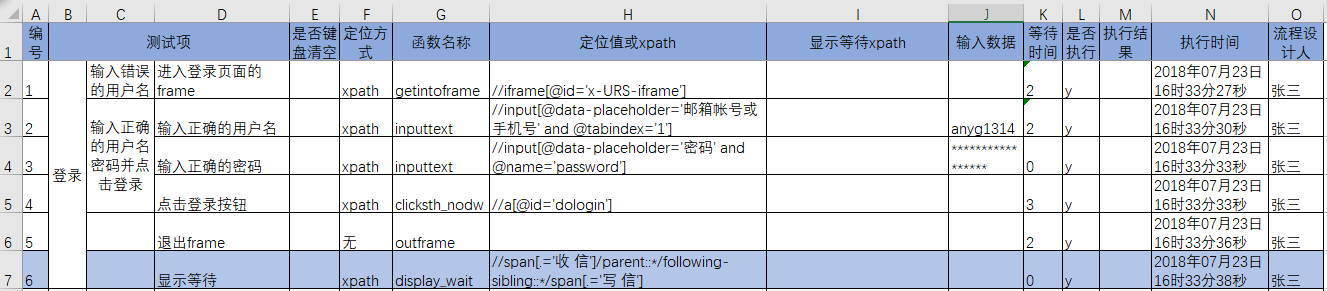
关闭浏览器

#### 代码

*# coding=utf-8***from** Util.FormatTime **import** date\_time  
**from** Util.excel **import** \*  
**from** Util.write\_log **import** \*  
**from** ProjectVar.var **import** \*  
**from** selenium **import** webdriver  
**from** Action.action **import** \*  
**from** Util.all\_display\_style\_wait **import** \*  
**import** time  
  
driver=bowser\_driver() *# 浏览器实例化*driver.get(**"https://www.126.com"**) *# 打开126邮箱登录页*driver.maximize\_window() *# 窗口最大化*time.sleep(3)  
  
ec=Excel\_r\_w(excel\_path,**u"用例"**) *# 实例化kd.xlsx文件的用例标签*wl=Writelog(log\_path) *# 生成写入日志的句柄***for** ir **in** range(2,ec.get\_max\_row()+1): *# 循环，范围从用例标签的第二行至最大行* **if** ec.get\_value(**"L"** + str(ir))==**"n"**: *# 如果第L列与被循环到的行的交点单元格的值为n* ec.write\_content(**"M"** + str(ir), **""**) *# 写入第M列的值为空* **continue** *# 说明此行用例不执行，所以直接进行下一次循环* **if** ec.get\_value(**"E"** + str(ir))==**"y"**:*# 如果E列与被循环行的交点值为y，则需使用键盘事件* action=**"inputtextkb"**+**"(driver,"***# 则拼一个字符，其中inputtextkb是all\_display\_style\_wait中的方法，它用于点击按钮或链接，同时使用键盘事件清空默认值* **else**:  
 action=ec.get\_value(**"G"** + str(ir))+**"(driver,"** *# 如果为空则不使用键盘事件，直接使用G列与被循环行的交点值所对应的函数，仍然是拼接字符串* **if** ec.get\_value(**"G"** + str(ir))==**"outframe"**: *# 如果G列与被选和行的交点值为outframe* action=**"driver.switch\_to.default\_content()"** *# 则调用action里的outframe函数，拼接字符串* **print "action="**, action *# 打印拼接的字符串action，以便在执行时清晰地看到执行过程，并有助于定位问题* **try**:  
 **exec** (action) *# 执行action* ec.write\_content(**"M"** + str(ir), **"通过"**) *# 在M列与被执行行交点格写入通过* ec.write\_datetime(**"N"**+str(ir)) *# 实例化write\_log里的类  
 # 写入日志，格式为时间+流程设计人+调用了函数+函数名称+是否通过* wl.writelog(date\_time(), ec.get\_value(**"O"** + str(ir)).encode(**"utf-8"**), **"调用了函数"**,ec.get\_value(**'G'** + str(ir)).encode(**"utf-8"**), ec.get\_value(**'M'** + str(ir)).encode(**"utf-8"**))  
 time.sleep(int(ec.get\_value(**"K"** + str(ir)))) *# 等待时间，参数从K列取* **except**:  
 **print u"用例"** + str(ir - 1).decode(**"utf-8"**) + **u"未通过"** *# 如果执行有错误则打印用例编号及未通过三个字，以便执行者可迅速得知是哪条用例没通过* ec.write\_content(**"M"** + str(ir), **"未通过"**) *# 如果执行有错误则写入未通过* ec.write\_datetime(**"N"** + str(ir)) *# 实例化write\_log里的类  
 # 写入日志，格式为时间+流程设计人+调用了函数+函数名称+是否通过* wl.writelog(date\_time(),ec.get\_value(**"O"** + str(ir)).encode(**"utf-8"**),**"调用了函数"**,ec.get\_value(**'G'** + str(ir)).encode(**"utf-8"**),ec.get\_value(**'M'** + str(ir)).encode(**"utf-8"**))  
 time.sleep(int(ec.get\_value(**"K"** + str(ir)))) *# 等待时间，参数从K列取* **continue** *# 执行完这套if语句后直接进行下一次循环* **if** ec.get\_value(**"G"** + str(ir)) == **"getintoframe"**: *# 如果G列与被选和行的交点值为getintoframe* action = action + **"("** + **"By.XPATH,"** + **"\""** + ec.get\_value(**"H"** + str(ir)) + **"\""** + **")"** + **")"** *# 拼接字符串，因为getintoframe函数是all\_display\_style\_wait中的frame\_isusabled方法，它的写法与其他方法不同，所以单列出来* **print "action1="**, action *# 打印action* **else**:  
 **for** ighi **in "HIJ"**: *# 循环HIJ三列* **if** ec.get\_value(ighi+str(ir))==None:*# 如果取值为空* **continue** *# 证明此种情况已经在前面的语句中处理过了，直接进入下一次循环* **if** ighi==**"J"**:*# 当循环到最后一次时* action = action + **"u"**+**"\""** + ec.get\_value(ighi + str(ir)) + **"\""** + **","** *# 字符串需如此拼接* **else**: *# 否则* action=action+**"\""**+ec.get\_value(ighi+str(ir))+**"\""**+**","** *# 字符串就如此拼接，两个拼接的区别在于，J列存储的是发送的数据，可能有中文，所以在数据前面加了个u，而GI是xpath，所以无需加u* action=action[0:-1]+**")"** *# 由于每次拼接都会在后面加一个逗号，而且没有反括号，所以最终需要去掉逗号，增加反括号* **print "action="**, action  
 **print "----------"  
 try**:  
 **exec**(action) *# 执行action字符串* ec.write\_content(**"M"** + str(ir), **"通过"**) *# 向M列输入通过* ec.write\_datetime(**"N"** + str(ir)) *# 获取写日志函类的实例  
 # 写入日志，格式为时间+流程设计人+调用了函数+函数名称+是否通过* wl.writelog(date\_time(),ec.get\_value(**"O"** + str(ir)).encode(**"utf-8"**),**"调用了函数"**,ec.get\_value(**'G'** + str(ir)).encode(**"utf-8"**),ec.get\_value(**'M'** + str(ir)).encode(**"utf-8"**))  
 time.sleep(int(ec.get\_value(**"K"** + str(ir)))) *# 等待时间，从K列里取值* **except**:  
 **print u"用例"**+str(ir-1).decode(**"utf-8"**)+**u"未通过"** *# 如果执行有错误则打印用例编号及未通过三个字，以便执行者可迅速得知是哪条用例没通过* ec.write\_content(**"M"** + str(ir), **"未通过"**) *# 如果执行有错误则写入未通过* ec.write\_datetime(**"N"** + str(ir)) *# 实例化write\_log里的类  
 # 写入日志，格式为时间+流程设计人+调用了函数+函数名称+是否通过* wl.writelog(date\_time(),ec.get\_value(**"O"** + str(ir)).encode(**"utf-8"**),**"调用了函数"**,ec.get\_value(**'G'** + str(ir)).encode(**"utf-8"**),ec.get\_value(**'M'** + str(ir)).encode(**"utf-8"**))  
 time.sleep(int(ec.get\_value(**"K"** + str(ir)))) *# 等待时间，从K列里取值*wl.writelog(**"-"**\*100+**"\n"**) *# 在日志里输入100个-，以明显区别每次执行所产生的日志*ec.save\_content() *# 保存Excel*wl.closefp() *# 关闭写日志句柄*driver.quit() *# 退出浏览器***if** \_\_name\_\_==**"\_\_main\_\_"**:  
 **pass**

# 实例

过完了代码，我们来运行一下这个框架，向kd.xlsx里输入一些内容。



先做一套登录的用例。我们直接在函数名称栏写上相应的函数名称，在定位xpath语句栏里写上定位该元素所需的xpath语句。

大家注意到，在点击登录按钮这条用例力，为什么没有显式等待xpath，这就要回答我们在在实例结果抢先看里面所提出的问题了，为什么点击这个动作要分成分成两个，一个是有显式等待的，一个是没有显式等待的。

因为登录页面的输入项都是在一个frame里的，所以我们封装了两个函数，一个是进入frame的，一个是退出frame的，而点击了提交按钮后，页面跳转至邮箱首页，此时，我们需要先退出frame，再进行显式等待函数的调用，退出frame在先，显式等待在后，所以我们就把带有显示等待的点击和没有显式等待的点击分开了，由人为判断什么时候该退出frame，什么时候时候进行显式等待。

大家可以看到，在退出frame后，又加了一条用例是用于显式等待的。

下面我们再来发一封邮件



进入邮箱首页之后，我们先点击了写信按钮，然后显式等待存草稿按钮，只要存草稿按钮已出现，则证明页面已跳转至写邮件页面。

然后，我们写入数据。大家注意到，邮箱一栏的数据是错的，它没有@符，这种数据在点击发送按钮后一定会报错的。于是，我们在点击发送按钮这条用例力就加了显示等待，等待页面出现了“返回编辑”显示框，如果出现了，则证明邮件系统可以容错。

大家注意到，在向收件人输入框输入有效收件人这条用例里，是否键盘清空一列为y，因为该输入框无法正常使用.clear()进行清空，所以使用了键盘事件进行清空。

下面我们再来添加一条联系人信息



大家注意到，有一条用例叫输入错误的的邮箱，套路和收件人错误是一样的，这里不再赘述。

我想重点说一下输入联系人和点击确定按钮，大家注意到了，点击确定按钮的显式等待就是在等待联系人列表里刚刚添加好的联系人，所以这就要求使用者在每一次输入联系人时都不能和以往的输入重复，否则联系人列表中就会出现多个相同的联系人，显式等待就失效了。

# end