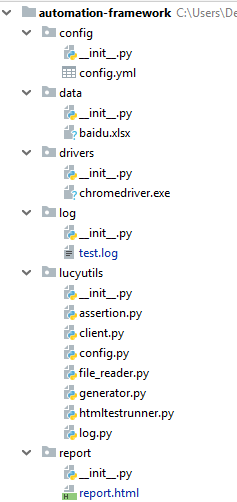
**一：框架结构**

我们先看下我们将要创建的框架的结构，分成两部分:

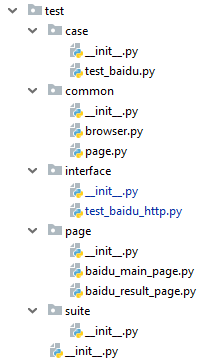
**1.一部分是配置工具类方法部分 （config, data, drivers, interface, log, lucyutils ,report）.**

* config 用来放配置文件
* data 用来放参数文件
* drivers 用来放各中driver，比如chromedriver.exe
* log 用来放生成的log文件
* lucyutils 用来放各种工具方法，比如发送请求，读excel，读yml，读配置等
* report 用来放生成的测试报告



**2.另一部分是测试部分（根据PageObject模型分类，分别是case，common，interface，page，suite）：**

* case 放UI测试用例
* interface 放接口测试用例
* page 放某个单独的页面
* common 放浏览器有关的操作和base page，页面基类
* suite 放test suite



**二：下面我们讨论下具体的开发步骤：**

**第一步：创建一个程序**

要求：打开百度，搜索“selenium 测试“ ， 把结果打印出来

代码： [github](https://github.com/breathfisheva/automation-framework/tree/1) ，test.py （tag1）

**第二步：创建一个类**

要求：搜索两次，一次搜索 ”selenium 测试“，一次搜索 ”selenium 测试2“.

思路：

* 用unittest框架来解决，创建一个TestBaidu类继承unittest.TestCase，
* 把初始化webdriver，关闭webdriver封装在两个类方法里 setUp，tearDown，(注意：这里的setUp和tearDown的大小写，因为是override已知函数，如果把setUp写成setup就认不出这个函数了。）
* 搜索的功能都封装成方法
* 把他们都放在TestBaidu类下。

代码：[github](https://github.com/breathfisheva/automation-framework/tree/2) ，test.py （tag2）

**第三步：配置信息抽离**

要求：把url信息抽离出来单独放在一个地方，把所有的路径信息放在一个地方。

思路：

* 创建一个config.yml，把url放到这个文件里
* 创建一个file\_reader.py ，读取yml类型的文件 （我们这里用的yml文件放配置信息，当然我们也可以用其他类型的文件放配置信息，比如XML，INI文件，需在file\_reader中添加相应的Reader进行处理。）
* 创建一个config.py，读取配置信息以及存放路径信息（在这里config.py文件调用file\_reader.py文件里的YamlReader类来读取config.yml文件里的配置信息）。
* 修改test.py文件,使用读取配置文件config.yml来获取url。

文件地址：

通过当前文件的绝对路径，其父级目录一定是框架的base目录，然后确定各层的绝对路径。如果你的结构不同，可自行修改。

我们在框架的base目录下面加：

* 加一个python package ，叫做 ”config“， 把config.yml放在这个package下面，
* 加一个python package，叫做 ”lucyutils“ （我们把一些工具类的脚本放在这里），把file\_reader.py，config.py放在这个package下面。
* 加一个python package，叫做“drivers”， 把chromedriver.exe放在这个package下面。

代码：[github](https://github.com/breathfisheva/automation-framework/tree/3)， config，drivers，lucyutils，test.py（tag3）

注意：

* python3.x 安装的yaml版本是PyYAML 否则会报错
* 关于路径，多用os.path.split()和os.path.join()，不要直接+'\\xxx\\ss'这样，容易出错

**第四步：日志**

要求: Python有很方便的logging库，我们对其进行简单的封装，使框架可以很简单地打印日志（输出到控制台以及日志文件）

思路：

* 创建一个log.py文件封装logging库
* 在框架base目录下创建一个python package，叫做“log”，存放log的结果
* 修改test.py文件，把输出放到log中

文件地址：

* 把log.py文件在“lucyutils”里
* log的结果在框架的base目录下面的log里面 （test.log）

代码：[github](https://github.com/breathfisheva/automation-framework/tree/4)，log.py，log，test.py （tag4）

注意：

尽量用os.path.join来连接路径，如果用加号容易出现错误，比如log.py文件里

一开始filename定义为 filename=LOG\_PATH + self.log\_file\_name 就出错了，因为LOG\_PATH 为 ..\log

这样加号直接和filename连接就少了‘\’就出错了。

可是如果用 os.path.join(LOG\_PATH,self.log\_file\_name）就不会有这个问题。

**第五步：日志-优化**

要求：把log.py里的一些配置信息，比如file\_name，backup，console\_level，file\_level，pattern 放到config.yml里

思路：

- 更新config.yml文件，把file\_name，backup，console\_level，file\_level，pattern信息放进去

- 更该log.py文件，读取配置信息

代码：[github](https://github.com/breathfisheva/automation-framework/tree/5)，config.yml，log.py （tag5）

**第六步：参数化**

要求：进行数据分离，进行参数化

思路：

    -把参数放到excel里，新增一个baidu.xlsx文件

    -修改file\_reader.py文件，添加ExcelReader类，实现读取excel里的参数

    -修改test.py文件，使用subTest

        a.使用subTest, 使用subTest来实现重复测试某个用例，实现参数化

        b.subTest是PY3 unittest里带的功能，PY2中没有，PY2中要想使用，需要用unittest2库。

        c.subTest是没有setUp和tearDown的，所以需要自己手动添加并执行。添加的方法是sub\_setUp 和 sub\_tearDown

文件地址：

- 在框架的base目录下，新建一个python package，叫做“data”，把baidu.xlsx放在这个下面

代码：[github](https://github.com/breathfisheva/automation-framework/tree/6)，baidu.xlsx，file\_reader.py，test.py （tag6）

**第七步：生成报告**

要求：把测试结果生成一个测试报告

思路：

- 新建一个htmltestrunner.py文件用来生成测试报告

- 在框架base目录下创建一个python package ， 叫做 “ report”，存放report的结果

-修改test.py 文件

文件地址：

-htmltestrunner.py放在lucyutils下

-测试报告在框架base目录下的report里

代码：[github](https://github.com/breathfisheva/automation-framework/tree/7)，htmltestrunner.py，report，test.py （tag7）

注意:

我们这里把执行报告的代码放在 if \_\_name\_\_ == '\_\_main\_\_':里，这里如果直接执行是执行不到的，因为默认执行的unittest

解决办法是，点击菜单栏的Run-》Run, 然后选择要跑的py文件。

执行后，可以在report目录下看到有 report.html 文件，我们已经生成测试报告了

**第八步：PageObject**

要求：针对UI自动化，接下来我们用PO思想进行下封装。

思路：

对于不同的项目，不同的页面，我们都需要选择浏览器、打开网址等，我们可以把这些操作抽象出来，让不同的用例去调用，只需要传入不同参数即可，不用一遍遍复制粘贴。

1.新建一个名为test的python package，然后在这下面创建page、common、case、suite四个python package (注意这五个目录都是要python package的格式，不是directory）

test

    |--case（用例文件）

    |--common（跟项目、页面无关的封装）

    |--page（页面）

    |--suite（测试套件，用来组织用例）

2.封装的选择浏览器、打开网址的类，所以放到common中，创建browser.py

browser.py文件做了非常简单的封装，可以根据传入的参数选择浏览器的driver去打开对应的浏览器，并且加了一个保存截图的方法，可以保存png截图到report目录下。

3.创建一个页面基类Page，放到common中，创建page.py

主要是所有页面都会涉及到的操作（初始化webdriver，查找元素的方法）

3.创建两个具体的页面类，继承父类Page，放到page目录下，baidu\_main\_page.py 和 baidu\_result\_page.py。 一个是封装的百度首页，一个封装百度结果页

(页面元素，以及页面的一些操作，这里主要是search功能，和result\_links）

4.修改test.py， 新建一个文件test.py，放到case目录下

5.现在，我们已经用PO把用例改写了，这里面还有不少问题，浏览器的设置、基础page的封装、log太少、没有做异常处理等等，之后逐步完善的。

文件路径：

        -test路径在框架base的目录下

        -browser.py在 test/common下

        -page.py在 test/common下

        -baidu\_main\_page.py 和 baidu\_result\_page.py在 test/page下

        -test\_baidu.py 在 test/case下

代码：[github](https://github.com/breathfisheva/automation-framework/tree/8)，browser.py ， page.py，baidu\_main\_page.py，baidu\_result\_page.py，test\_baidu.py （tag8）

注意：

这里要把test.py名字改一下，否则和最外面的test python package重复容易有问题。

**第九步：接口测试**

要求: 可以测试http请求的接口

思路：

- 创建一个client.py文件，作为接口测试的客户端发送请求 （对于HTTP接口添加HTTPClient类，发送http请求，这里用到的是requests库。如果你的接口类型不是HTTP的，可以封装对应的Client类。socket库测TCP接口、suds库测SOAP接口，不论你是什么类型的接口，总能找到对应的Python库的。）

     - 在test下面创建一个interface 的python package 用来放接口测试的测试用例。（和UI的测试用例分开。）

   - 创建一个test\_baidu\_http.py文件作为接口测试的测试用例 （我们加了一句断言，没有断言怎么能叫用例，我们之前写的UI用例，也可以自己动手加上断言。）

文件路径：

     - client.py文件放在lucyutils下

     - test\_baidu\_http.py 接口测试用例放在test/interface下

代码：[github](https://github.com/breathfisheva/automation-framework/tree/9), client.py, interface, test\_baidu\_http.py （tag 9） （注：tag 9里面我犯了一个错误，把interface 放在框架的base路径下了，应该是在test路径下，我在[tag12](https://github.com/breathfisheva/automation-framework/tree/12)更正了。）

**第十步：断言**

要求：上次我们的用例中增加了断言。断言（检查点）这个东西对测试来说很重要。不然你怎么知道一个测试结果是对是错呢。unittest为我们提供了很多很好的断言，但是对于我们的项目可能是不够的。我们需要封装自己的断言方法。

思路：

   -在lucyutils中创建assertion.py文件，在其中创建断言

    在assertion.py中你可以添加更多更丰富的断言，响应断言、日志断言、数据库断言等等，请自行封装。

   -在test\_baidu\_http.py中添加此断言

文件路径：

    -assertion.py在lucyutils下

代码：[github](https://github.com/breathfisheva/automation-framework/tree/10)，assertion.py，test\_baidu\_http.py （tag10）

**第十一步：生成器**

要求：建一个生成器用来生成测试数据（比如指定长度中文、英文、特殊字符的字符串，指定格式的json串等等，可以省去很多构造测试数据的烦恼。）

思路：

   -在lucyutils中创建一个generator.py文件，里面调用了python 的faker库。

代码：[github](https://github.com/breathfisheva/automation-framework/tree/11)，generator.py （tag11）

**Reference：**

1.这里用到的git命令

git status

git add --all

git commit -m "commit information"

git push

git tag -a 2 -m "version2"

git push --tags

2.换git 的username 和email (因为用的是先生的电脑，结果发现我都是在用他的用户名进行commit的，所以在local也就是当前repro下把用户名该了。）

>git config --local --add user.name breathfisheva

>git config --local --add user.email ""

参考文档： <https://www.jianshu.com/p/88c68779784b>