AsiaInfo**项目管理文档** 

**云南移动自动化测试项目**

**开发手册说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| 作者 | 周涛 | 编写 时间 | 2020-06-06 |
| 审 核 | 审核者姓名（及其职务） | 审核 时间 | 审核日期 |
| 文档版本 | V1.0 | | |

**湖南亚信软件有限公司版权所有**

**文档修订摘要**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **日期** | **修订号** | **描述** | **著者** | **审阅者** | **审阅日期** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目录**

[第1章 引言 5](#_Toc26688)

[1.1 文档用途 5](#_Toc3688)

[1.2 阅读对象 5](#_Toc5845)

[1.3 名词术语 5](#_Toc5394)

[1.4 参考资料 6](#_Toc31552)

[第2章 功能概述 6](#_Toc28596)

[2.1 功能简介 6](#_Toc83)

[2.2 技术栈 7](#_Toc7090)

[第3章 自动化框架设计 9](#_Toc23634)

[3.1 源码介绍 9](#_Toc13036)

[3.1.1 基础库 9](#_Toc13212)

[3.1.2 API接口开发 12](#_Toc31293)

[3.1.3 页面Selenium自动化 14](#_Toc2488)

[3.2 测试案例管理 15](#_Toc14557)

[3.2.1 测试类 15](#_Toc15263)

[3.2.2 测试套件 17](#_Toc18425)

[3.3 测试数据管理 19](#_Toc21880)

[3.3.1 方案描述 19](#_Toc1030)

*[3.4 测试报告管理](#_Toc26408)* [21](#_Toc26408)

[3.4.1 Html自动化报告 21](#_Toc21904)

[3.4.2 Doc测试记录 22](#_Toc22110)

[3.4.3 Excel测试结果 23](#_Toc30100)

[第4章 遗留问题与设想 24](#_Toc25092)

# 引言

## 文档用途

该文档作为云南扩容项目自动化测试运维文档，适用UI页面自动化以及API接口测试

因环境更迭频繁，回归测试工作量大且很繁琐。引进自动化测试可以提升回归测试效率，将手工重复执行案例的过程用自动化脚本代码替代，测试人员只需要准备每次回归关键案例的测试数据。

## 阅读对象

系统设计工程师

系统开发工程师

系统测试工程师

业务支撑部门技术人员

## 名词术语

下列术语和定义适用于本文档：

|  |  |
| --- | --- |
| 术语 | 说明 |
| Selenium | WEB自动化工具，直接运行在浏览器 |
| Requests | Python的Http客户端库 |
| UnitTest | Python单元测试框架 |
| DDT | Data-Driven-Test 数据驱动测试 |
| Cx\_Oracle | Python的Oracle数据库操作 |
| xlrd,xlwt,OpenPyxls | Python的关于Excel表格的读写操作库 |
| docx | Python的关于Doc文档的读写操作库 |
| HTMLTestRunnerCNNew | UnitTest测试报告生成库 |
| logging | python日志库 |
| configparser | python操作配置文件 |
| json | Python的json模块，主要用于json和dict的相互转换 |

## 参考资料

# 功能概述

## 功能简介

1. UI自动化采用PO分层设计模式  
   2.selenium元素定位和操作进行二次封装，代码：webtest\Base\IMG_256Base.py  
   3.建立测试数据管理，Common\IMG_257TestDataMgnt.py,功能持续完善中......  
   4.测试试案例管理，IMG_258allrun.py,将测试案例配置到config\caselist.txt  
   5.测试环境信息配置在config\ngboss\_config\_ini文件，通过ReadConfig读取  
   6.使用Orcale\_cx 自动从测试数据库抽取测试数据，并将测试数据自动写入xls，路径：Webtest/Data/YYYYMMDD/.xls  
   7.OperExcle封装了常用的xls操作方法，案例执行后自动将测试结果更新到测试数据xls文件。  
   8.使用pythonDoc，UI自动化测试执行关键步骤截图到Doc文档，文档保存路径：webtest\Reports\document\YYYYMMDD/.docx  
   9.关于测试报告，二次开发HTMLTestRunnerCN,实现饼图和截图功能。如脚本执行异常会自动截图展示在Html报告  
   10.测试结果保存三份格式：  
    Doc记录UI自动化脚本执行过程中关键检查点步骤，保存目录：webtest\Reports\document\YYYYMMDD\  
    xls保存测试脚本生成的测试数据，结果保存在Webtest/Data/YYYYMMDD/\*.xls ，如成功记录流水号FlowId，如失败将检查点错误信息记录到RESULT\_Info字段, 暂时是一个testCase一个表格。  
    Html记录自动化脚本测试报告，汇总测试结果。  
   11.目前已实现的案例：  
    UI：个人业务包括：个人用户开户、商品订购、换卡、停开机业务受理、共享业务、家庭网业务、普通付费关系变更、过户、销户、业务返销等核心业务 （后续完善.....） 集团业务包括：集团商品业务受理、成员商品业务受理、集团付费关系变更、集团营销活动办理  
    API:商品订购、停开机、实名制资料登记、用户密码变更等...

## 技术栈

1. **Selenium ：**

Selenium   是一个用于Web应用程序测试的工具。Selenium测试直接运行在浏览器中，就像真正的用户在操作一样。支持的浏览器包括IE（7, 8, 9, 10, 11），[Mozilla Firefox](https://baike.baidu.com/item/Mozilla Firefox/3504923" \t "https://baike.baidu.com/item/Selenium/_blank)，Safari，Google Chrome，Opera等。这个工具的主要功能包括：测试与浏览器的兼容性——测试你的应用程序看是否能够很好得工作在不同浏览器和操作系统之上。测试系统功能——创建回归测试检验软件功能和用户需求。支持自动录制动作和自动生成 Python 、.Net、Java、Perl等不同语言的测试脚本。

1. **Requests：**

Request对象的作用是与客户端交互，收集客户端的Form、Cookies、超链接，或者收集[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8/100571" \t "https://baike.baidu.com/item/Request%E5%AF%B9%E8%B1%A1/_blank)端的环境变量。request对象是从客户端向服务器发出请求，包括用户提交的信息以及客户端的一些信息。客户端可通过[HTML](https://baike.baidu.com/item/HTML/97049" \t "https://baike.baidu.com/item/Request%E5%AF%B9%E8%B1%A1/_blank)表单或在网页地址后面提供参数的方法提交数据，然后服务器通过request对象的相关方法来获取这些数据。request的各种方法主要用来处理[客户端](https://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF/101081" \t "https://baike.baidu.com/item/Request%E5%AF%B9%E8%B1%A1/_blank)浏览器提交的请求中的各项参数和选项。

1. **UnitTest ：**

unittest是Python单元测试框架，类似于JUnit框架。unittest中有4个重要的概念：test fixture, test case, test suite, test runner.

Testcase：一个TestCase的实例就是一个测试用例。什么是测试用例呢？就是一个完整的测试流程，包括测试前准备环境的搭建(setUp)，执行测试代码 (run)，以及测试后环境的还原(tearDown)。元测试(unit test)的本质也就在这里，一个测试用例是一个完整的测试单元，通过运行这个测试单元，可以对某一个问题进行验证。

Test suite：多个测试用例集合在一起，就是TestSuite，而且TestSuite也可以嵌套TestSuite。

Test runner：是来执行测试用例的，其中的run(test)会执行TestSuite/TestCase中的run(result)方法。

TestLoader：是用来加载TestCase到TestSuite中的，其中有几个loadTestsFrom\_\_()方法，就是从各个地方寻找TestCase，创建它们的实例，然后add到TestSuite中，再返回一个TestSuite实例。

1. **DDT：**

DDT (Data-Driven Tests) allows you to multiply one test case by running it with different test data, and make it appear as multiple test cases.一个测试用例可以用不同的测试数据执行多次。

DDT consists of a class decorator ddt (for your TestCase subclass) and two method decorators (for your tests that want to be multiplied):

data: contains as many arguments as values you want to feed to the test.

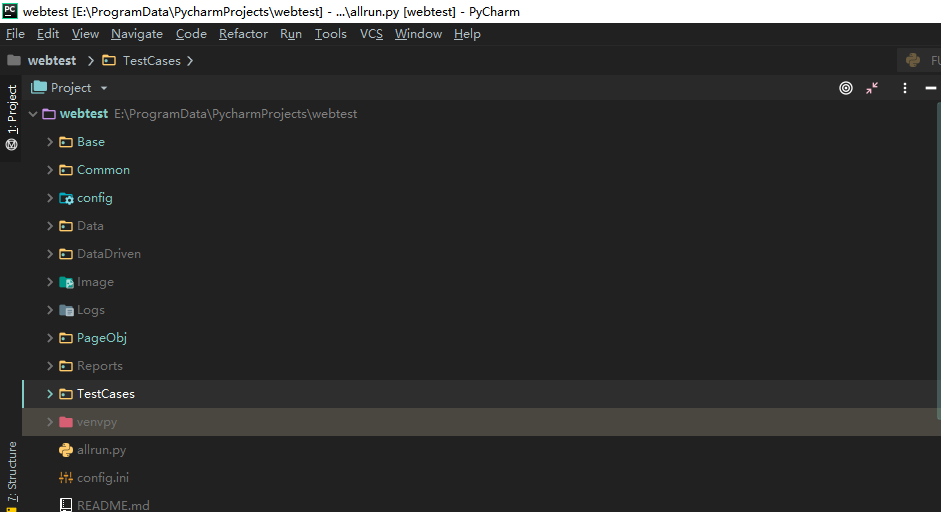
file\_data: will load test data from a JSON or YAML file.

**有一个类装饰器 ddt，两个方法装饰器 data, file\_data 支持jason和yaml文件**

**参考资料：**<https://blog.csdn.net/lelemom/article/details/84312732>

# 自动化框架设计

## 源码介绍



### **基础库**

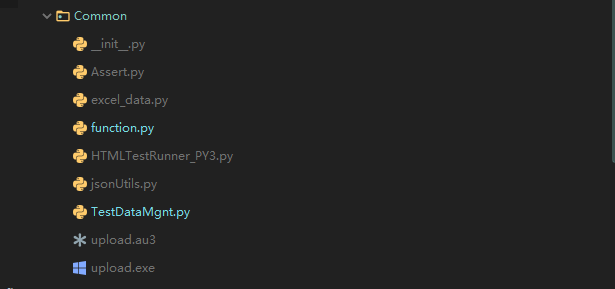
包括Base 、Common、config 三个Package

#### Base

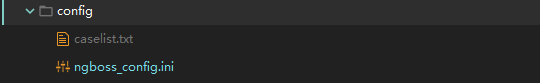
|  |  |
| --- | --- |
| **moduleName** | **Desc** |
| base.py | Selenium的二次封装，将UI页面元素常用定位方法和操作重新封装、页面js的处理以及浏览器、Handler等处理。 |
| GenTestData.py | 自动Gen出测试数据，如：证件号、手机号码、客户姓名、短号、密码等 |
| HTMLTestRunnerCNNew.py | HTMLTestRunner进行了二次开发，主要用于自动化报告的处理，已实现异常自动截图和饼图统计、失败重跑功能。 |
| Mylog.py | 日志管理 |
| OperExcel.py | Excel常用读写操作，将各类型数据拼装写入或读取。关键Func:write\_dict\_xls、write\_xlsBycolName\_append、get\_exceldata |
| OracleOper.py | Oracle常用操作，主要是查询。select方法实现通过sql将查询结果按字典列表返回，方便DDT数据驱动测试。 |
| ReadConfig.py | 配置文件读取方法，结合ngboss\_config.ini编写，读取配置的各类环境信息 |

#### Common

|  |  |
| --- | --- |
| Assert.py | 页面PageAssert操作，主要处理wade页面各类提示信息。错误提示、告警提示、帮助提示、成功提示以及提交后页面返回信息处理。主要用于自动化案例执行过程的规则检查以及页面返回=消息的抓取 |
| function.py | 常用工具类，自动化脚本中经常用到的数据转换处理 |
| jsonUtils.py | json工具类，读取和解析各类复杂嵌套json串。主要用于接口自动化脚本开发。完善中..... |
| TestDataMgnt.py | 测试数据管理，包括两类实现方案：1.Excel测试案例数据组装与提取（适用测试套件数据组织）。 2.通过Oracle查询组装数据自动生成测试数据保存到xls（适用单个用例参数化） |
| upload.exe | 使用AutoIt Window Info (x64)开发的自动上传附件组件 |



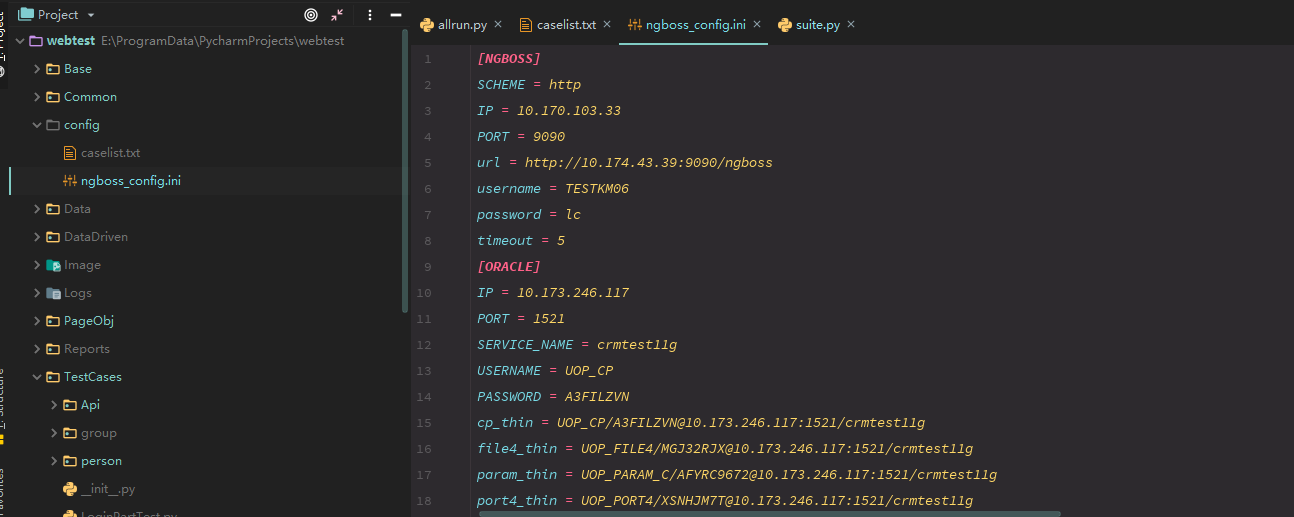
#### Ngboss\_config.ini



Ngboss\_config.ini: 配置测试环境信息，如：测试Url地址、测试登录帐号、密码

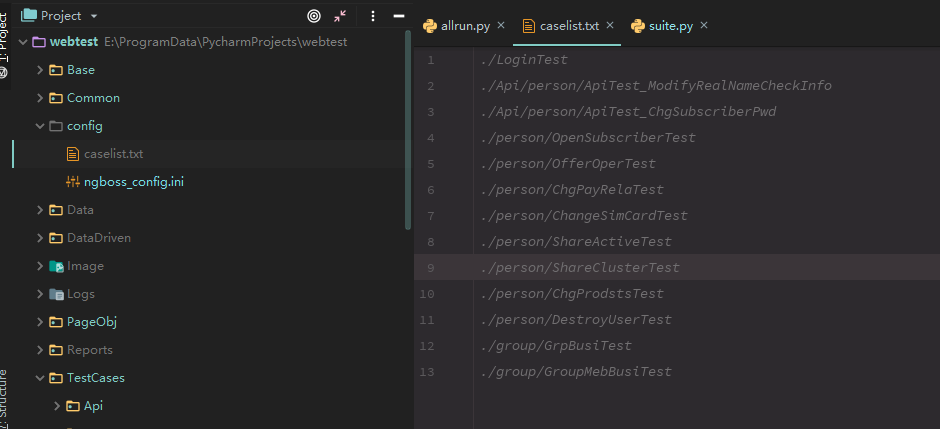
Oracle数据库链接、API接口测试Url地址等

通过ReadConfig.py读取



Caselist：配置已开发完成的测试案例，结合allrun.py组装测试套件TestSuite

注意：组装的测试案例脚本必须保存在webtest\TestCases目录下



### API接口开发

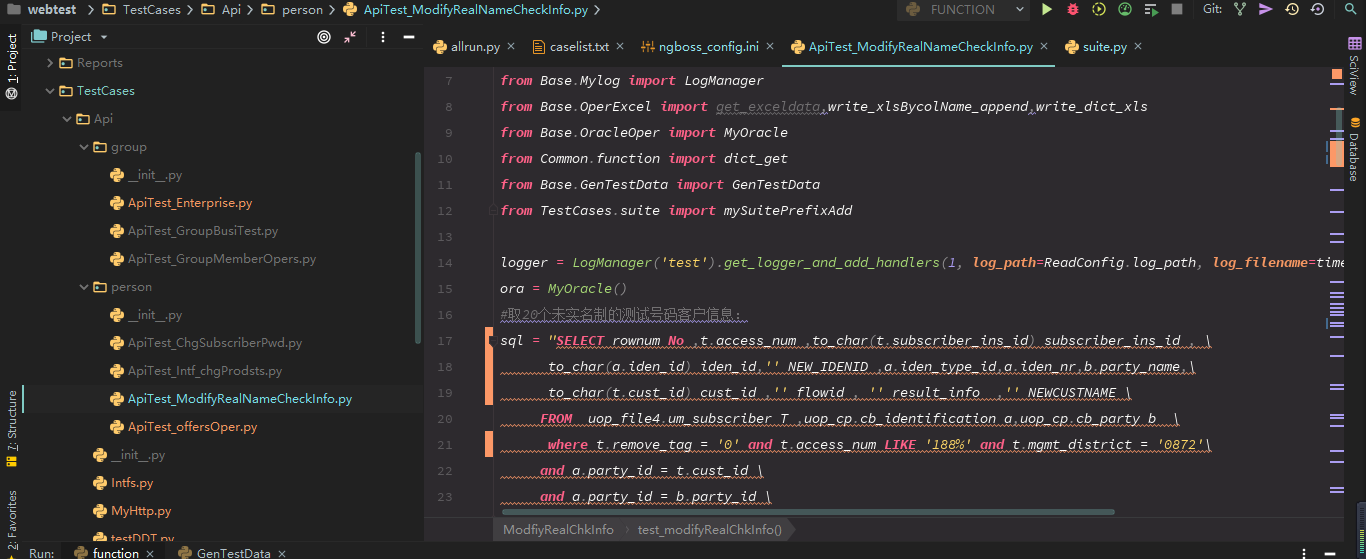
路径：webtest\TestCases\Api

Python + Requests + DDT + Excel

关键要点：

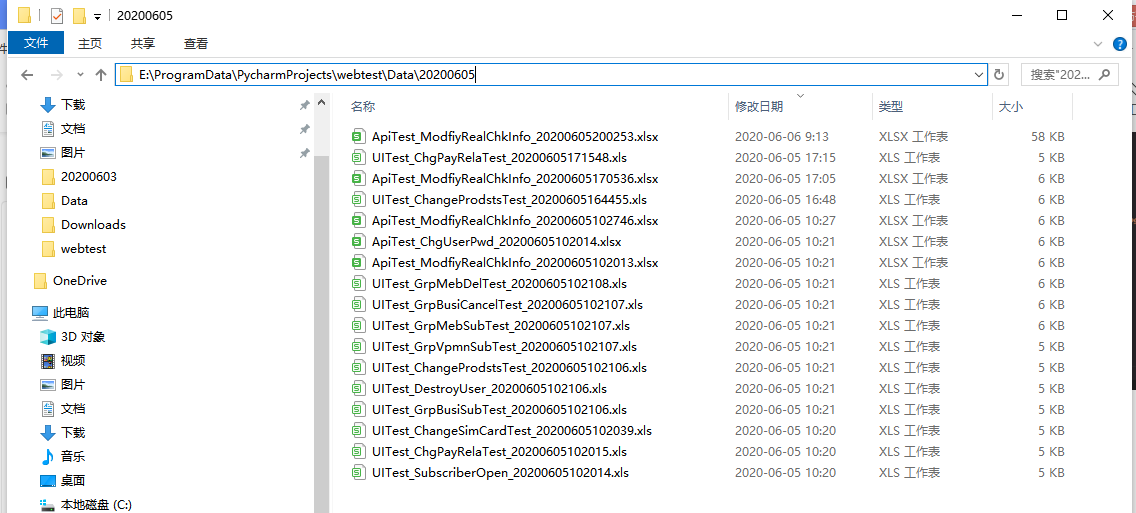
1.案例执行前通过session技术，让执行环境保存登录状态

* 1. 使用Oracle查询获取测试数据自动写入xls，通过DDT参数化测试数据，驱动测试执行
  2. 使用function.py模块方法，处理接口接口返回数据处理
  3. 使用OperExcel.py模块方法将测试结果写入xls数据文件

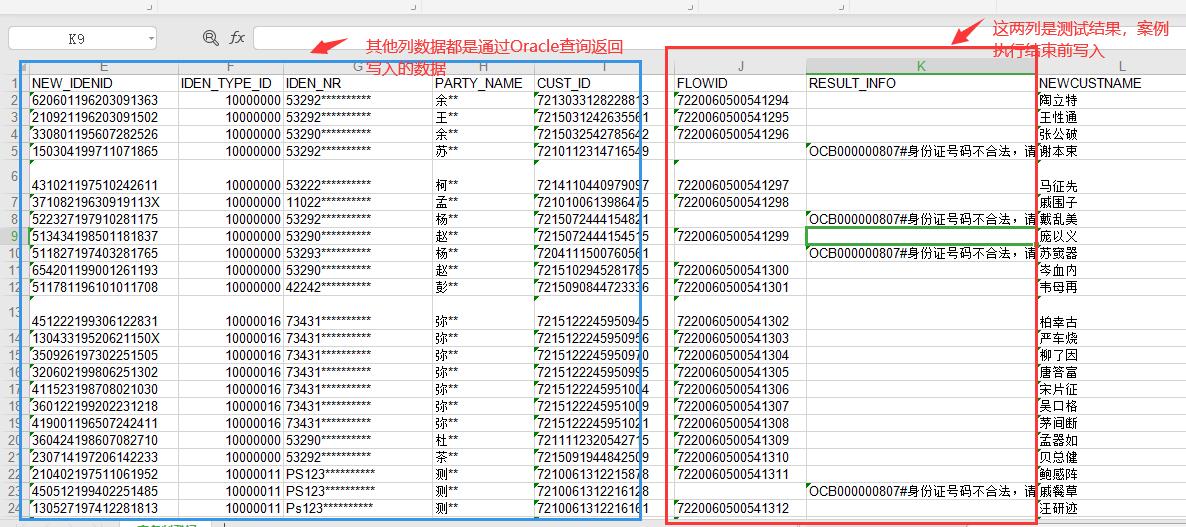


数据文件路径：webtest\Data\YYYYMMDD

数据文件名称：{FuncCode} \_YYYYMMDDHHMMSS.xls



接口测试结果文件展示效果：



### **页面Selenium自动化**

使用PageObject设计思想，**Page Object** 是 Selenium 自动化测试项目开发实践的最佳设计模式之一，Page Object 的主要体现于**对界面交互细节的封装**，这样可以使测试案例更关注与业务而非界面细节，提高测试案例的可读性。

**Page Object 设计模式的优点如下：**

* 减少代码的重复；
* 提高测试用例的可读性；
* 提高测试用例的可维护性，特别是针对 UI 频繁变化的项目；

**Paget Object 使用注意：**

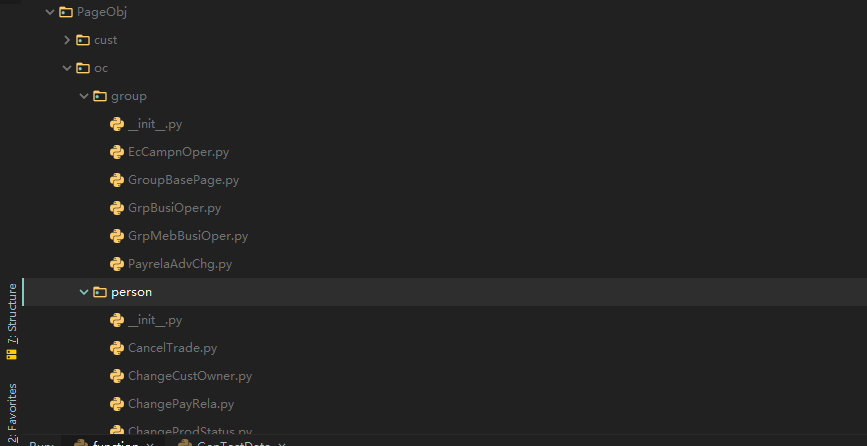
* public方法代表Page提供的功能，如页面操作步骤
* 尽量不要暴露Page的内部的操作细节，少用复杂判断
* 不要assertion
* Page Objects代表整个页面对象，可以操作页面对象中任意元素
* 一样的操作，不同的结果应该分开（正确登录，错误登录）
* 注意Iframe 的切换，如Iframe切换异常影响相关元素定位于操作
* 合理运用显式等待、隐式等待和强制等待，提升Selenium 运行效率

本项目webtest\PageObj 路径下保存各中心业务办理页面:

1. PersonBase.py 个人业务相关公共操作页面，如：号码登录、搜索商品、打开个人业务菜单

已经个人业务规则校验(证件校验、号码校验)

1. GroupBase.py 集团业务相关公共操作页面,如：集团登录鉴权认证、集团业务菜单、集团业务受理规则校验、集团商品相关操作
2. 个人业务办理界面继承PersonBase类，集团业务办理界面继承GroupBase类
3. 尽量一个页面Page 对应一个Test类。
4. Page页面开发只关注相关元素定位于操作，测试数据尽量不要体现在代码



## 测试案例管理

### 测试类

Page页面完成开发后，在TestCases目录下建立测试类，使用UnitTest框架编写。

关键点：测试步骤、页面校验检查点以及测试数据的组织、测试报告的展示以及测试结果的处理。

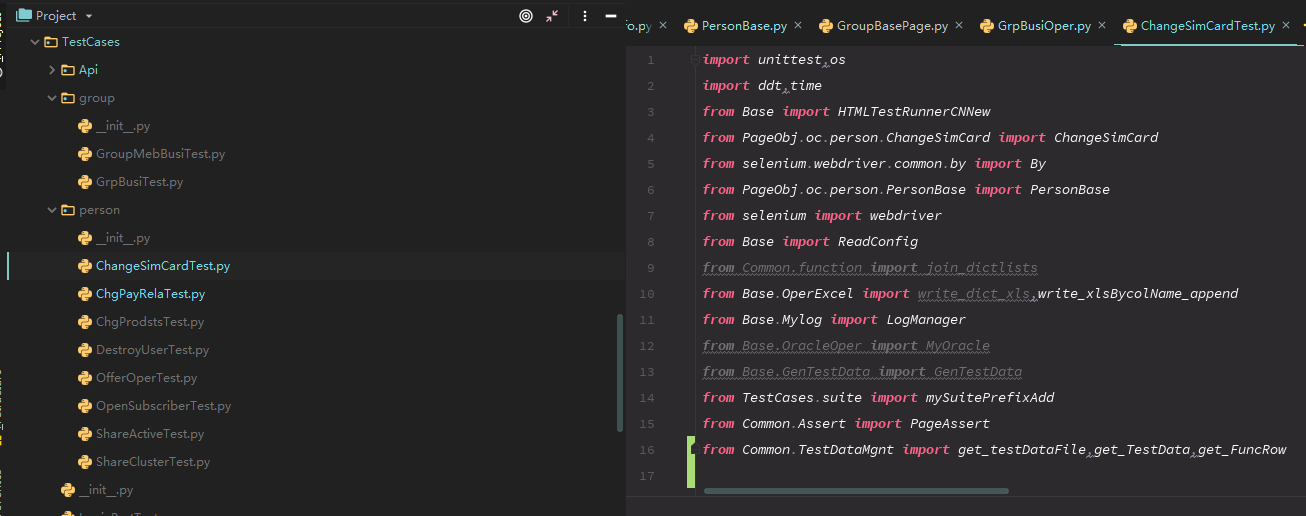
后面相关章节详细介绍

#### UI

每个Page对应一个测试类。

已完成调试的UI案例：

|  |  |
| --- | --- |
| **Module** | **Desc** |
| GrpBusiTest.py | 集团商品办理（订购与注销） |
| GroupMebBusiTest.py | 集团成员商品（订购与注销） |
| OpenSubscriberTest.py | 个人用户开户 |
| OfferOperTest.py | 商品订购 |
| ChgProdstsTest.py | 停开机业务办理 |
| ChangeSimCardTest.py | 换卡 |
| ShareActiveTest.py | 共享业务CRM办理 |
| ShareClusterTest.py | 共享成员办理（新增/删除成员、群组注销） |
| ChgPayRelaTest.py | 付费关系变更 |
| DestroyUserTest.py | 个人销户 |



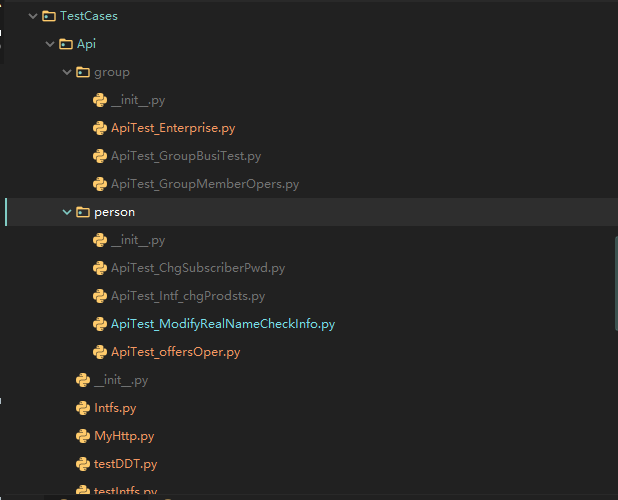
#### API

接口测试用例保存在TestCases\API路径

每个接口对应一个案例

当前完成使用频率较高的接口

|  |  |
| --- | --- |
| **Module** | **Desc** |
| ApiTest\_Enterprise.py | 集团客户资料查询接口 |
| ApiTest\_GroupBusiTest.py | 集团业务查询与办理接口 |
| ApiTest\_GroupMemberOpers.py | 集团成员业务查询与办理接口 |
| ApiTest\_offersOper.py | 个人商品订购接口 |
| ApiTest\_ModifyRealNameCheckInfo.py | 个人实名制资料登记接口 |
| ApiTest\_Intf\_chgProdsts.py | 个人停开机业务受理接口 |
| ApiTest\_ChgSubscriberPwd.py | 个人用户服务密码变更接口 |
|  |  |

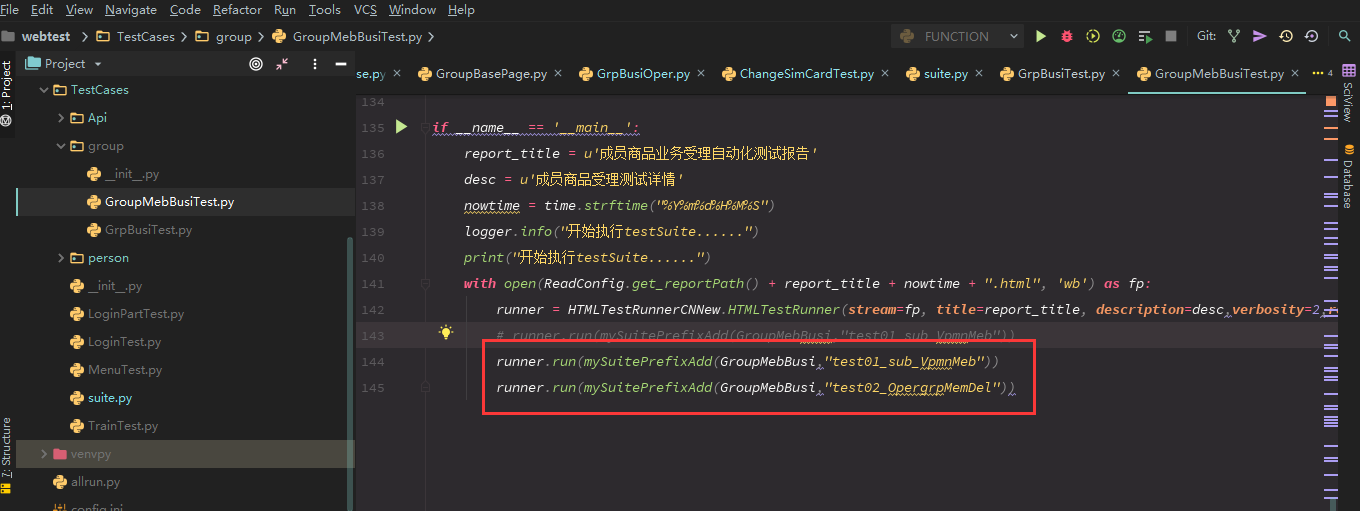


### 测试套件

目前实现两种方式生成测试套件

1. 同一个Test类下组织测试套件，使用suite.py,关键方法mySuitePrefixAdd

实例：如GroupMebBusiTest测试类，针对不同的集团商品有集团成员订购和退订操作，想法是通过DDT数据驱动并结合mySuitePrefixAdd生成测试套件来执行,适用于同类业务办理场景的案例设计。



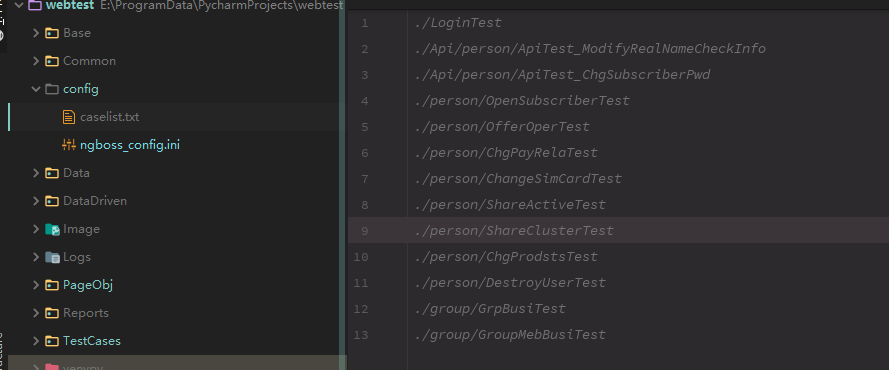
1. 多个Test类测试套件的生成。使用allrun.py,关键方法：get\_case\_suite

将测试类配置到caselist.txt文件中，通过get\_case\_suite获取测试集，run方法运行测试集

生成测试报告。



caselist.txt



## 测试数据管理

DDT数据驱动是自动化测试执行的核心，如何组织与管理测试数据，如何处理测试结果数据。

针对不同的测试场景设计两种方案，实现模块：Common.TestDataMgnt

### 方案描述

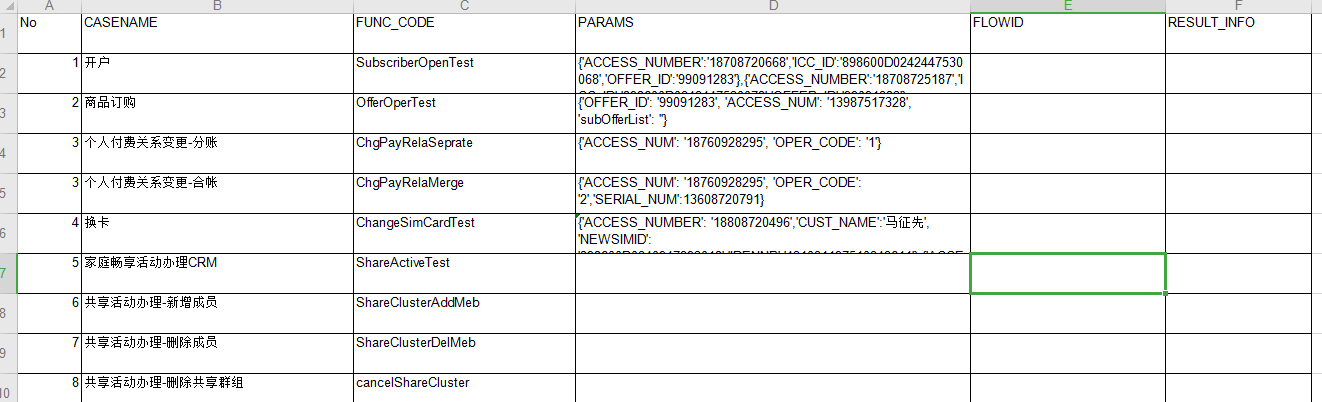
#### 数据驱动Excel

xls模板格式固定，将测试案例以及测试数据提前在xls模板完成，get\_TestData方法读取测试

数据PARAMS，get\_testDataFile 获取测试数据xls模板，方便测试执行结果写入该文件。

适用于测试集的执行。

xls模板路径：webtest\Data\UITest\_TestData.xls



**注意事项：**

1. FuncCode 为自定义的CaseCode,不要重复，代码中通过FuncCode可以获取单元格数据，

如对应的PARAMS以及行号，方便DDT数据驱动案例以及测试结果写入。

新增案例时，需要结合定义的这个FuncCode写数据抽取逻辑

1. FLOWID、RESULT\_INFO列名不要更改，此两列为测试执行结果
2. PARAMS列为案例执行的数据，使用Python中的字典格式，最好是一个案例对应一个数据字典
3. xls路径和文件名不要更改，代码中有用到
4. 自动化脚本执行前，务必保证该文件不要处于打开状态，否则后续写入测试结果时发生异常。





该方案实现了测试数据的统一管理，测试数据和测试结果一张表格展示，清晰简单。但是不够灵活，注意事项太多。并且xls变更的同时代码也要同时更新。

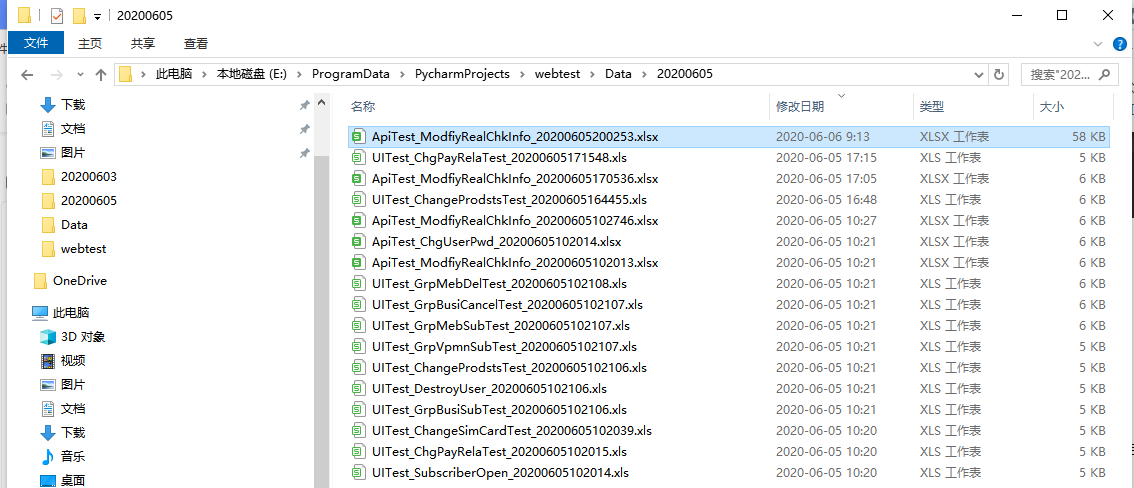
#### 数据驱动ORA

**简介**：直接通过sql查询语句获取测试数据并转换成字典数组格式，自动完成测试数据xls写入，同时脚本执行结束时候将测试结果自动更新到数据文件。

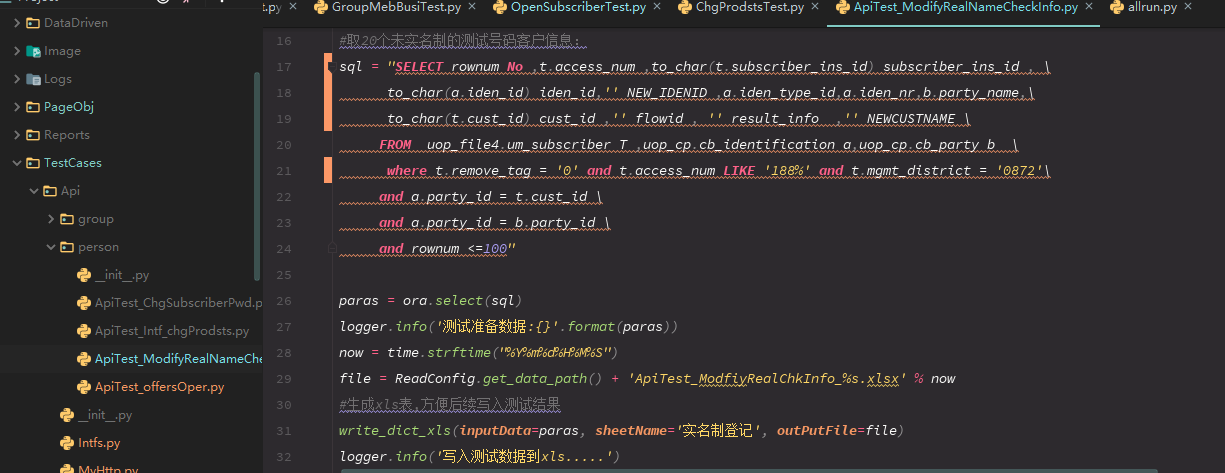
数据文件路径：webtest\Data\YYYYMMDD\{funcCode}\_yyyymmddhhmmss

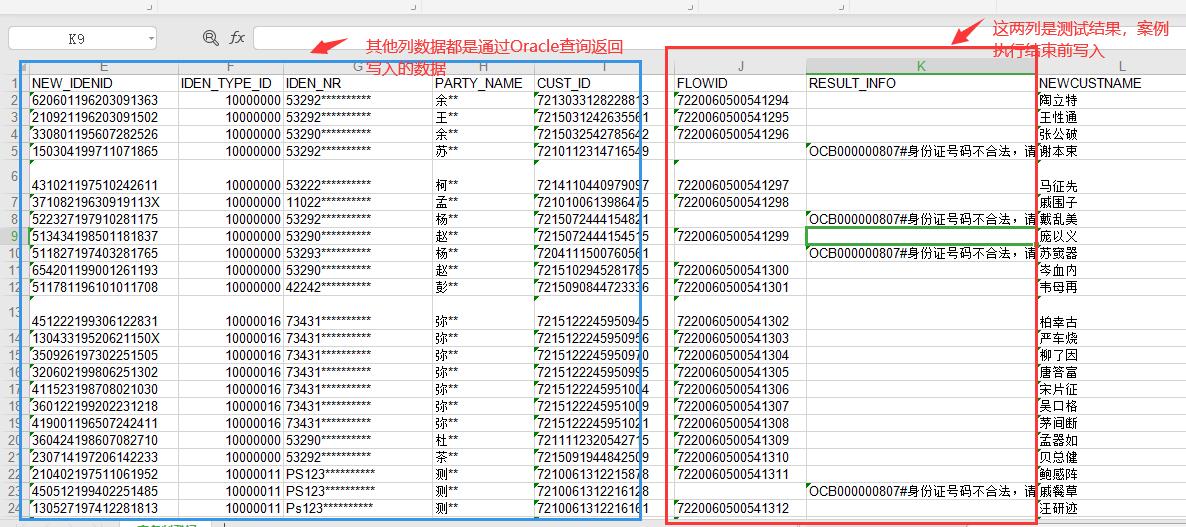
适用单个TEST类的参数化自动测试，场景如：接口测试、业务规则自动化测试等。

数据ORA驱动的方案灵活，数据管理基本可以实现自动化处理。缺点是：数据比较分散，每个测试案例都会单独生成一个数据文件。比较适用于接口自动化以及同一个业务的参数化测试（如业务规则等）



相关代码展示：



文件内容：

## ***测试报告管理***

目前设计分三类格式保存，Html 展示案例执行的汇总报告，浏览器访问，展示内容：测试套件所有案例的执行结果以及截图，饼图汇总。

套件中的单个UI用例自动化执行过程中关键步骤以及页面检查点截图保存到DOC文档，每个用例生成一份，路径：webtest\Reports\document

数据驱动测试结束后，将测试结果自动写入xls 数据文档，方便查看。

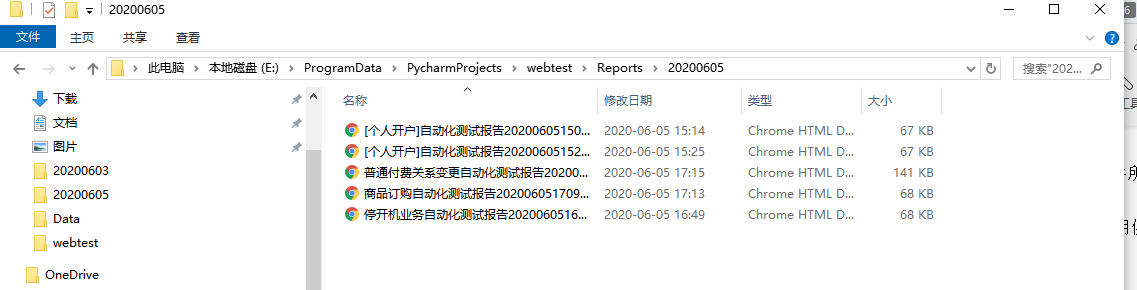
### Html自动化报告

前面提到两种生成测试套件的方式，Html测试报告根据不同的场景生成报告

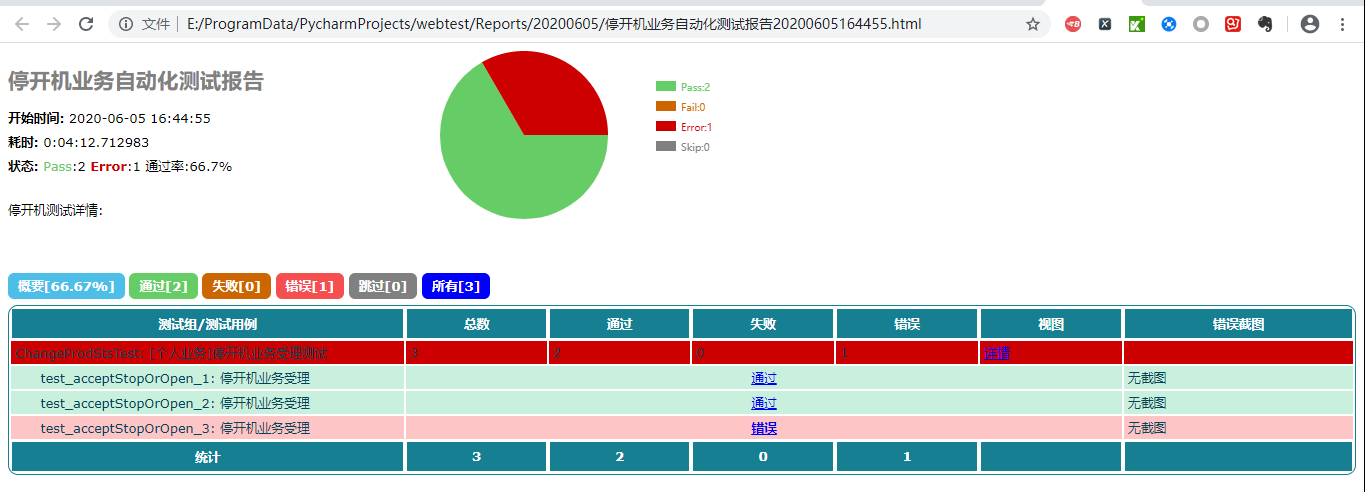
核心代码：Common.HTMLTestRunnerCNNew (二次开发)

参数：retry=2表示执行失败重复再执行2次。

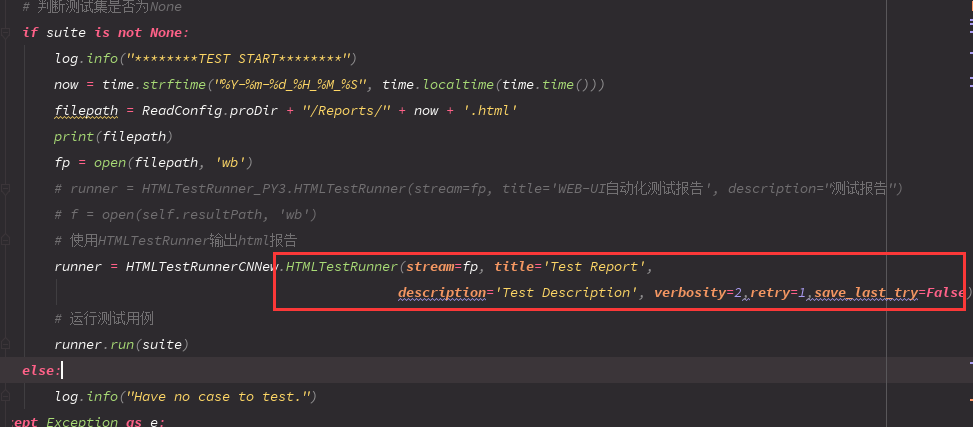
如是第一类：同一个Test类自动化脚本执行报告，保存路径：webtest\Reports\YYYYMMDD



展示：



第二类：测试集执行结果，HTML保存在webtest\Reports根路径，效果相同。

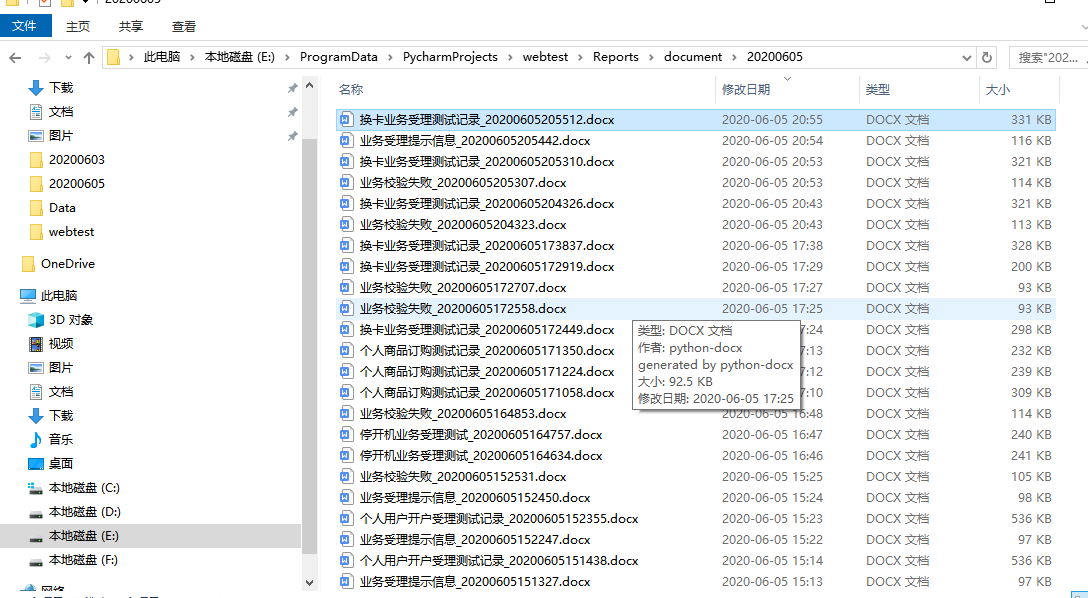


### Doc测试记录

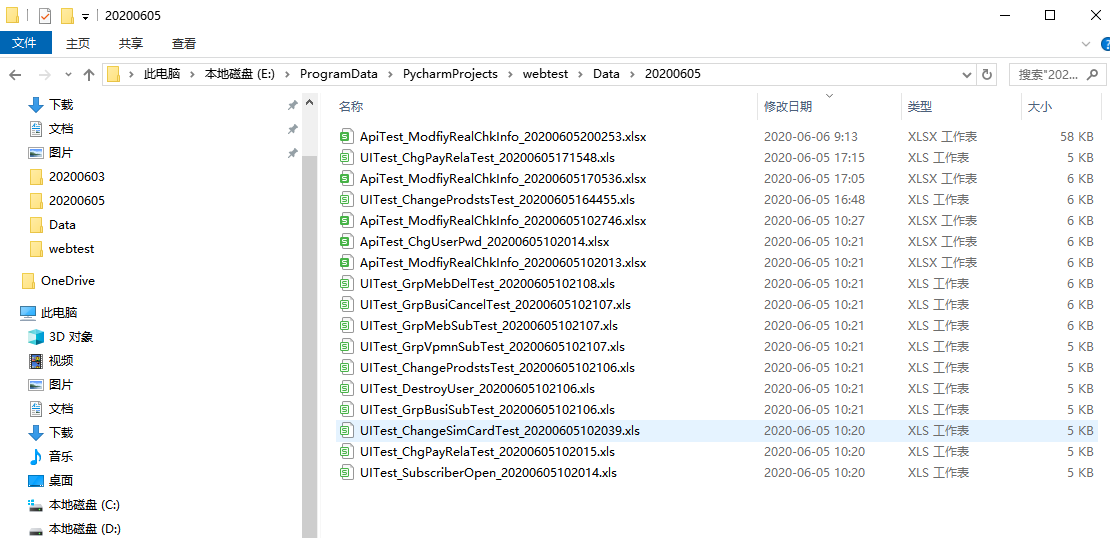
使用Python 扩展模块docx，在测试案例脚本中编写，将案例自动化脚本执行过程中的关键步骤（比如业务校验页面、提交后页面）自动截图保存到Doc文档

文档保存路径：webtest\Reports\document\YYYYMMDD

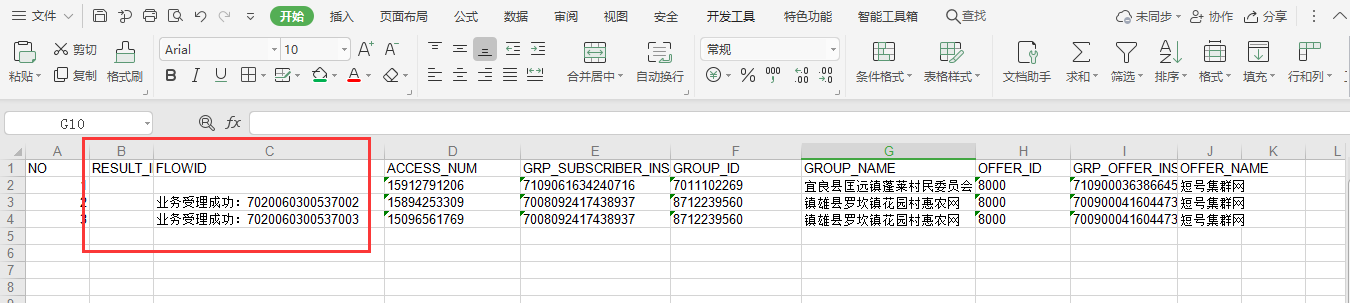
文件名：案例中文名称\_yyyymmddhhmmss.doc



### Excel测试结果



测试结果：



# 遗留问题与设想

|  |  |  |
| --- | --- | --- |
| **No** | **Problem** | **Suggestion** |
| 1 | 测试数据驱动，数据管理目前还比较散乱、未从根本上统一两类测试套件的数据管理，各有优缺点。 |  |
| 2 | Selenuim元素定位开发调试时间长，因不同业务Page加载时间不一，有快有慢。 | 在显式等待的基础上加入强制等待，显式等待时间目前在Base.py设计的是5s,如商品订购页面加载非常慢，每次打开菜单时强制等待时间适当设长一些。 |
| 3 | UI 页面校验判断，WadePage使用不规范。写页面校验时考虑业务校验可能出现的各类校验情况（错误、告警、帮助、成功等），导致自动化程序运行时耗时验证。 |  |
| 4 | 完全硬编码，优势：比较灵活。缺点：上手困难，易用性不够 | 自动化测试平台开发，Django |
| 5 | 目前是根据ChromeDriver浏览器驱动设计开发的，其他浏览地如FireFox是否兼容，如部署到LINUX后浏览器驱动问题. | UI自动化程序部署与执行 |
| 6 | 引入Jenkins集成，应用发布成功后即执行脚本。实现邮件自动提醒功能 |  |
| 7 | Docker虚拟化部署的应用 |  |