WebUI自动化测试框架

**1** [**VS Code + Python3 + Selenium测试框架安装及配置**](#_dn1hlssn78ip) **2**

[1.1 前置条件](#_krypykfklhuy) 2

[1.2 安装Python 3.6](#_yp6lko1fq6w8) 2

[1.3 配置Python3.6为默认Python编译器(Mac自带Python 2.7)](#_fv91g7umiuez) 2

[1.4 安装Selenium for Python](#_1hb5eudpzm3r) 3

[1.5 下载安装Chrome driver](#_puz7bkkosg6d) 3

[1.6 安装pytest](#_3pw5cai11r8v) 3

**2** [**框架结构**](#_ipnze4dq5350) **4**

[2.1 整体框架结构如下:](#_jvmgx2h4q5rj) 4

[2.2 文件结构图](#_howtbkq6q893) 5

[**3 框架详述**](#_d7b1trw0byrg) **6**

[3.1 main.py](#_z1wgf2r9clyy) 6

[3.2 run](#_7egcidyjntxl) 7

[3.3 object](#_fpzj6eo9j369) 7

[3.4 action](#_azis1tbvqmnf) 8

[3.5 testdata](#_guzzq13fgfzy) 8

[3.6 case](#_vh8e7kwqpzlc) 8

## VS Code + Python3 + Selenium测试框架安装及配置

### 1.1 前置条件

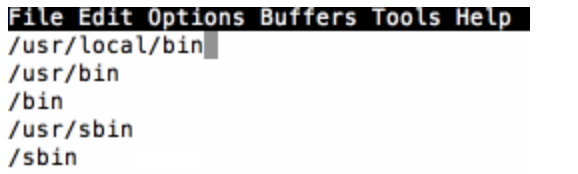
已安装VS Code

### 1.2 安装Python 3.6

$ brew install python3

### 1.3 配置Python3.6为默认Python编译器(Mac自带Python 2.7)

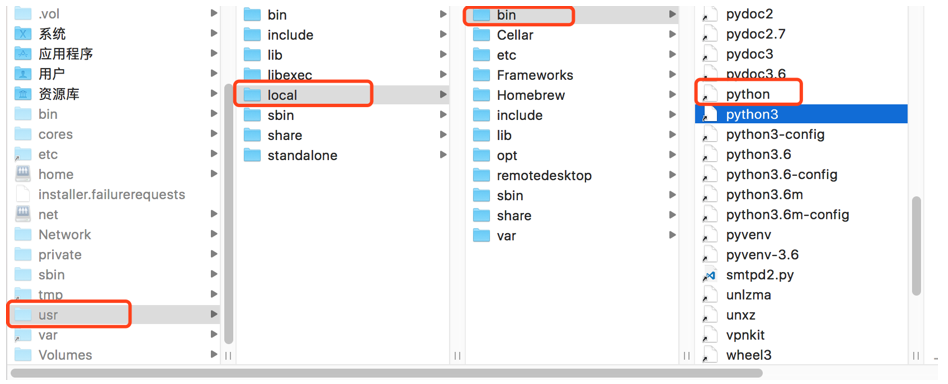
检查$PATH配置, 确定/usr/local/bin, 在/usr/bin之上  
 $ echo $PATH



如果否执行

$ sudo emacs /etc/paths //修改至如上图

$ which python //检查当前python路径, 如为/usr/local/bin/python则正确  


如否打开以下路径,确定python存在. 如不存在, 复制python3, 并改名为python  


$ python //运行python命令, 确定当前版本

### 1.4 安装Selenium for Python

方法一（推荐，需先安装pip install）：

$ pip install -U selenium

方法二：

下载安装包: https://pypi.python.org/pypi/selenium (e.g. selenium-3.5.0.tar.gz), 并解压缩  
 $ cd /Users/chenlisha/Downloads/selenium-3.5.0 //进行解压后的文件夹  
 $ python setup.py install //安装

### 1.5 下载安装Chrome driver

下载地址:<https://sites.google.com/a/chromium.org/chromedriver/downloads>

解压并移到 /usr/local/bin

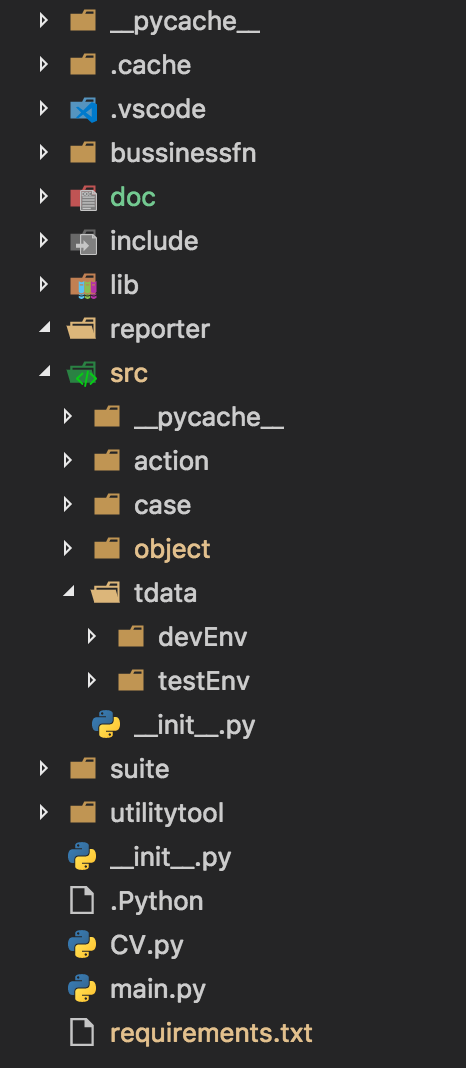
### 1.6 安装必要的三方工具

$ pip freeze > requirements.txt

## 

## 框架结构

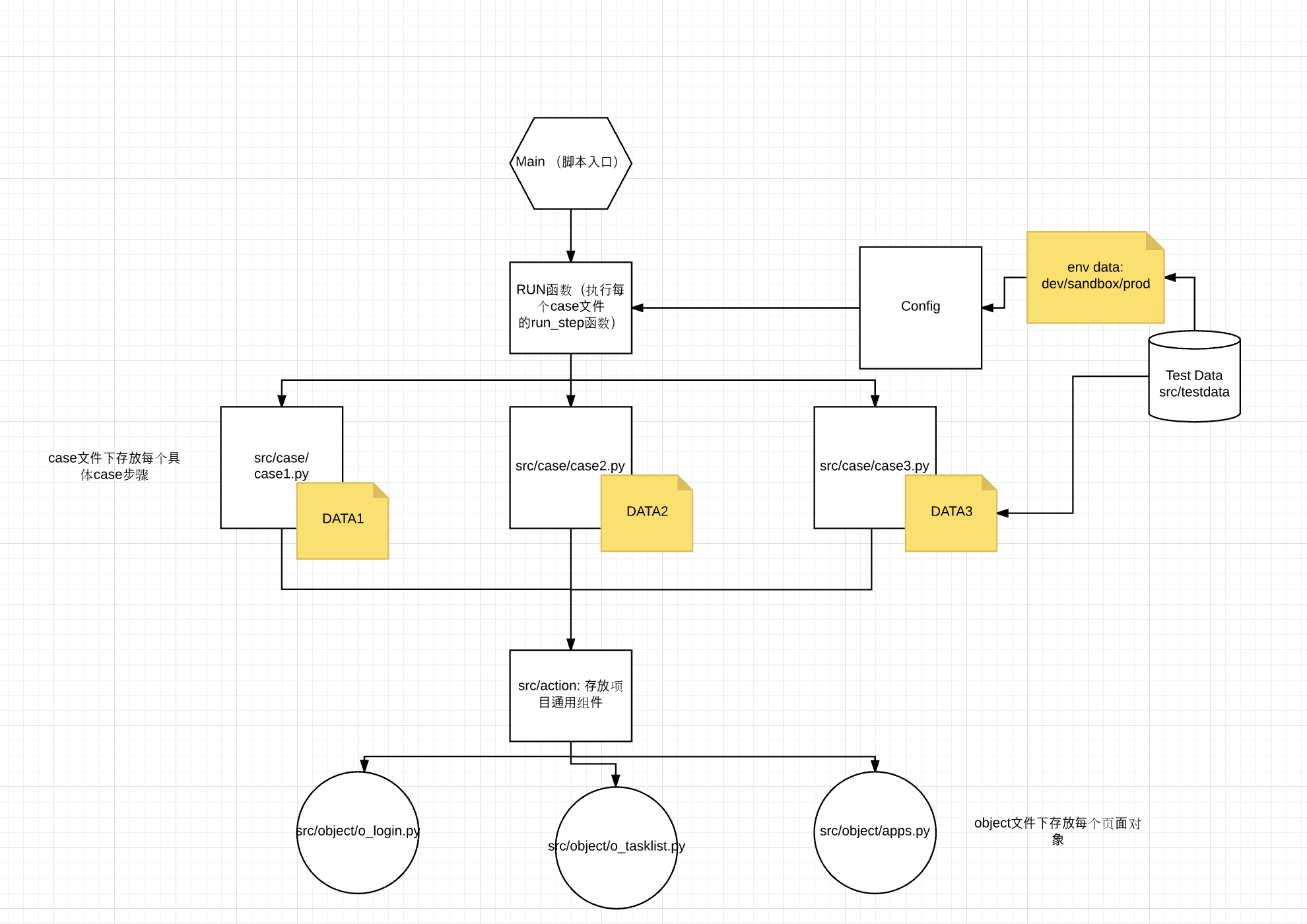
### 2.1 整体框架结构如下:



整体目录如上图所示，每个目录对应的职能如下：

|  |  |  |
| --- | --- | --- |
| 目录（包） | 简介 | 备注 |
| bussinessfn | 业务相关函数目录 | 存放项目中与业务相关的函数，如：reporter/suite/testdata |
| Doc | 文档目录 | 说明文档等 |
| reporter | 测试结果目录 | 存放所有测试结果文件（目前支持.html/.txt/.json）, 文件名为”rel\_+suite名+执行时间” |
| src | 资源文件目录 | 存放所有资源文件, 包括: action/case/object/testdata |
| src/action | 操作组件目录 | 存放完成一定功能的类, case可以调用多个action类组合成一个完成的测试用例  action文件一般以”a\_”开头 |
| src/case | 测试用例目录 | 存放所有测试用例文件, case可调用action和testdata完成一个测试用例  每个case文件包含一个def run\_step(test\_env, browser):函数, run文件会调用每个被执行的case文件的该函数 |
| src/object | 对象目录 | 存放被测系统对象的标识, action操作对象时, 从object文件读取对象标识  一般每个页面会对应一个object文件 |
| src/tdata | 测试数据目录 | 按被测环境分包存放测试数据, 包名可自定义，main中配置环境CV.ENV\_CONFIG = "包名"  每个环境包下有一个”config.py”, 为此环境公共数据 |
| suite | 测试套件目录 | 存放json格式的suite文件，key可自定义，value为要执行的case名称（不包括.py后缀） |
| utilitytool | 公共工具目录 | 与业务无关的公共函数/类，如：utelement/utfile/tujson/utmysql/utXpath |
| CV.py | 常量及全局变量 | 编码格式，DB连接信息，测试报告样式等 |
| main.py | 主文件(入口文件) |  |

### 2.2 文件结构图



整个自动化框架如上图所示，主要针对测试用例开发，测试执行，变更维护三个阶段。

测试用例开发：主要维护对象库，操作组件，测试数据，测试用例

测试执行：主要维护测试套件，Main

变更维护主要根据变更的内容维护对应的对象库，操作组件

## 3 框架详述

### 3.1 main.py

main.py包含以下配置信息，执行测试前需确认配置信息正确。

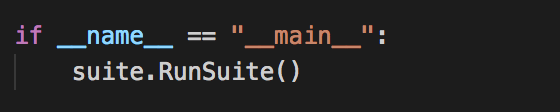


sys.path：指向测试框架项目的根目录。项目中所有相对路径调用，以此地址为基准。

CV.ENV\_CONFIG：用于在多个环境中执行测试，如：开发环境，sandbox环境，测试环境等，此值与对应的testdata包下的环境包名相同。指向不同环境包下的测试数据及配置文件。

CV.BROWSER\_CONFIG：用于在多个浏览器上执行测试（浏览器兼容性测试）。需先导入selenium的webdriver模块，并调用webdriver对应的浏览器类。（如调用其他浏览器类，请如1.5步骤，下载导入其他浏览器驱动）

CV.SUITE\_NAME：本次要执行的suite文件名，不包含.json后缀。如suite下不存在与此值相同的json文件，调用失败。

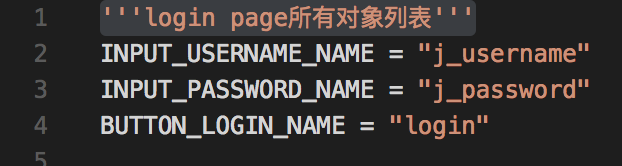


在main中调用suite.RunSuite()方法执行配置的suite文件中的所有case。

参数：reptype 默认空，不生成测试报告。reptype = CV.FILETYPE\_TXT/CV.FILETYPE\_JSON/CV.FILETYPE\_HTML，在reporter下生成对应类型的测试报告。

### 3.3 src/object

一般一个被测页面对应一个object文件，约定object文件以o\_开头，不按此约定命名不影响调用。



object文件将所有对象标识存为常量，约定按”对象类型\_对象名\_标识属性”命名。

action操作对象时，从object文件中读取对象标识。

如对象标识发生变更，只需修改object文件中对应的常量值，所有操作了该对象的action就可以读取到新的标识。

### 3.4 scr/action

一般每个操作组件对应一个action文件，也可以将功能相似的多个操作class存于一个action文件。每个action模块可包含一个或多个action类及该类的属性和方法。在设计类时需要考虑其通用性，尽量只处理操作，不包含数据和对象，可以被更多的用例公用。

action从object读取对象标识

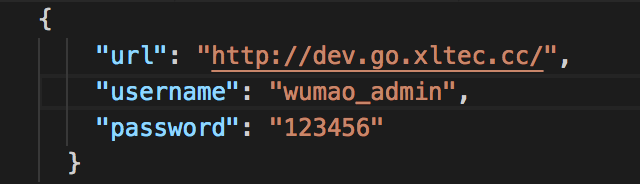
action的数据以参数形式从调用它的case传入

一般action文件约定以a\_开头，不按此约定命名不影响调用。

### 3.5 src/tdata

测试数据按不同被测环境在tdata包下按环境建子包。每个子包包含一个config.json公共数据文件和多个约定以d\_开头的tdata文件。config.json文件名不可修改，其他tdata文件不按些约定全名不影响调用。

一般一个case对应一个tdata文件，多个case公用的数据存于config.json中。



如测试数据发生变更，只需修改tdata文件中对应的值，所有读取了该数据的case都能读取到新的数据。

多个环境下所有数据文件的格式必须相同，否则不同环境读取key时会出错。

### 3.6 src/case

一般一个测试用例对应一个case文件，约定case文件以c\_开头，不按此约定命名不影响调用。



case文件尽量只是调用action组成用例，本身不要包含操作步骤。

\* 每个case文件必须包含一个def run\_step():函数，命名及参数都不可修改。执行case时会首先识别此函数。

case读取通过调用bussinessfn.testdata.getData(“tdata文件名”):方法取得数据并传入action。运行时读取哪个环境的数据，由main.py中的配置确定。

action类的方法在case内，由一个与其对应的私有函数调用。run\_step调用私有函数组成case。以私有函数调用action是为了可以每个action调用不同的tdata文件，及加入检查点，调用bussinessfn.reporter.checkpoint(exp, des=”None”):。

## 4 常用类/模块及方法/函数

### 4.1 类utilitytool.utelement

### 4.1.1方法：createObjfile()

创建object.py文件。

此方法获取当前页面（通过eletype配置的元素类型）的所有元素，并生成object.py文件。

注：已知bug，以xpath识别的元素以text命名，如text包含中文或特殊字符，命名不符合编码规则。（待优化）

### 4.2 模块utilitytool.tumysql

### 4.2.1方法：queryFirstRow(strsql, args=None):

查询并返回第一行结果。

|  |  |
| --- | --- |
| 参数 | 描述 |
| strsql | SQL SELECT语句，语法同MySQLdb |
| args | 默认为空，查询条件参数 |

### 4.2.2方法：queryAll(strsql, args=None):

查询并返回所有结果。

|  |  |
| --- | --- |
| 参数 | 描述 |
| strsql | SQL SELECT语句，语法同MySQLdb |
| args | 默认为空，查询条件参数 |

### 4.2.3方法：queryMany(strsql, args=None, rows=0):

查询并返回指定行数结果。

|  |  |
| --- | --- |
| 参数 | 描述 |
| strsql | SQL SELECT语句，语法同MySQLdb |
| args | 默认为空，查询条件参数 |
| rows | 默认为0，取返回结果的行数 |

### 4.3 模块bussinessfn.reporter

### 4.3.1 checkpoint(exp, des=”None”)

设置检查点（断言），并将结果写入测试报告中。

|  |  |
| --- | --- |
| 参数 | 描述 |
| exp | 断言的表达式 |
| des | 默认为空，检查点的描述信息 |

### 4.4 模块bussinessfn.suite

### 4.4.1 RunSuite(reptype = None):

按main函数中CV.SUITE\_NAME的配置，执行测试套件，并指定测试报告的类型。

|  |  |
| --- | --- |
| 参数 | 描述 |
| reptype | 默认为空，不生成测试报告。支持CV.FILETYPE\_TXT/CV.FILETYPE\_JSON/CV.FILETYPE\_HTML三种类型 |

### 4.5 模块bussinessfn.testdata

### 4.5.1 getData(filename):

按tdata文件名在src/tdata/下读取对应json文件，返回Dictionary类型测试数据

|  |  |
| --- | --- |
| 参数 | 描述 |
| filename | tdata文件名，不包含.json后缀 |