

SWEETEST

介绍

Sweetest 是一款同时支持 Web UI，Http 接口，DB 操作的自动化测试框架(即将支持 App 测试)，由于开始只支持 Web UI 测试，名字取自 Selenium，Web UI，Excel，Element，Test 含义。

特点:

1. 简单快速，轻松上手
2. 无需编码能力
3. 在 Excel 中以文本编写测试用例
4. 维护成本低
5. 支持千、万级别的用例规模
6. 拥抱变化，支持敏捷

背景

目前，自动化测试基本上是以 Selenium/Appium 等库为接口来编写**测试代码**，但效果往往不是很好，普遍遇到如下问题：

1. 用例设计人员的编码能力很弱，测试代码编写和维护成本高，效果差；
2. 测试代码量大，测试意图不直观，无法支撑千、万级别的用例规模；

3. 页面元素的定位非常繁琐，且页面结构经常变动，导致用例失效。

为了解决以上问题，使自动化测试更简单、可靠、容易落地，我们设计了此自动化测试框架，自动化测试用例和传统的手工测试用例一样在 Excel 中用文本编写。

同时我们的元素定位也设计的非常精巧，结合“变量定位法”，可以让页面自由的去变化，而我们的定位只做最小适用。

实现思路

1. Selenium 为底层接口；
2. 在 Excel 中用文本编写测试用例；
3. 元素定位表格化，且优先使用“变量定位法”；
4. 框架负责解析测试用例，执行用例，记录日志，输出测试结果。

方案

1. 开发语言：Python
2. 底层接口：Selenium
3. 用例工具：Excel

测试用例如下图：

用例编号	用例标题	前置条件	测试步骤	操作	页面	元素	测试数据	输出数据
HOME_001	打开百度	SETUP	1	打开	通用	百度搜索链接	新窗口=百度搜索窗口	
			2	检查	百度搜索页面	页面title	百度一下\, 你就知道	
BAIDU_001	搜索：自动化测试		1	输入	百度搜索页面	搜索框	自动化测试	
			2	点击	百度搜索页面	搜索按钮		
			3	检查	通用	页面title	自动化测试_百度搜索	
SNIPPET_001	搜索：<keywords>	SNIPPET	1	输入	百度搜索页面	搜索框	<keywords>	
			2	点击	百度搜索页面	搜索按钮		
			3	检查	通用	页面title	<keywords>_百度搜索	
BAIDU_002	搜索：搜狗		1	执行	用例片段	SNIPPET_001	keywords=搜狗	
			2	点击	百度搜索页面	搜索结果#1		title=text
			3	检查	通用	页面title	<title>	
BAIDU_003	搜索：知乎		1	执行	用例片段	SNIPPET_001	keywords=知乎	
			2	点击	百度搜索页面	搜索结果#1		
			^3	检查	知乎	link#注册机构号		
			>4	点击	知乎	button#注册		
			<5	点击	知乎	button#提问		
BAIDU_004	搜索：雪球行情中心		1	执行	用例片段	SNIPPET_001	keywords=雪球行情中心	
			2	点击	百度搜索页面	搜索结果#1		title=text
			3	检查	通用	页面title	*<title>	
			4	点击	雪球行情中心	新股预告表格#2#1		name=text
			5	检查	股票详情页	页面title	*<name>	
BAIDU_005	搜索：<测试数据>		1	执行	用例片段	SNIPPET_001	keywords=<_keywords>	
			2	点击	百度搜索页面	搜索结果#1		
			3	检查	通用	页面title	<_title>	

安装

环境要求

- 系统要求: Windows
- Python 版本: 3.6+
- 浏览器: Chrome
- Chrome 驱动: [chromedriver](#) (需和 Chrome 版本匹配, 并配置环境变量, [参考这里配置](#))

安装 SWEETEST

```
pip install sweetest
```

升级 SWEETEST

```
pip install --upgrade sweetest
```

快速体验

打开 cmd 命令窗口, 切换到某个目录, 如: D:\Autotest

```
sweetest  
cd sweetest_example  
python start.py
```

```

D:\Autotest>pip install sweetest
Collecting sweetest
  Using cached https://files.pythonhosted.org/packages/17/e5/57d637e2f21feed1744913d98ead56e79c41152806eeb493e32e
Requirement already satisfied: selenium in c:\program files\python36\lib\site-packages (from sweetest) (3.11.0)
Requirement already satisfied: xlrd in c:\program files\python36\lib\site-packages (from sweetest) (1.1.0)
Requirement already satisfied: xlswriter in c:\program files\python36\lib\site-packages (from sweetest) (1.0.4)
Requirement already satisfied: requests in c:\program files\python36\lib\site-packages (from sweetest) (2.18.1)
Requirement already satisfied: injson in c:\program files\python36\lib\site-packages (from sweetest) (0.1.2)
Requirement already satisfied: idna<2.6,>=2.5 in c:\program files\python36\lib\site-packages (from requests->swee
Requirement already satisfied: urllib3<1.22,>=1.21.1 in c:\program files\python36\lib\site-packages (from request
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\program files\python36\lib\site-packages (from request
Requirement already satisfied: certifi>=2017.4.17 in c:\program files\python36\lib\site-packages (from requests->
Installing collected packages: sweetest
  Running setup.py install for sweetest ... done
Successfully installed sweetest-0.5.2

D:\Autotest>sweetest
生成 sweetest example 成功










D:\Autotest>cd sweetest_example

D:\Autotest\sweetest_example>python start.py

```

OK，如果一切顺利的话，sweetest 已经跑起来了

目录结构

名称	修改日期	类型	大小
 data	2018/3/7 14:42	文件夹	
 element	2018/3/7 13:52	文件夹	
 Junit	2017/12/4 16:03	文件夹	
 log	2018/3/7 10:41	文件夹	
 report	2018/3/7 14:44	文件夹	
 snapshot	2018/3/7 14:31	文件夹	
 sweetest	2018/3/7 10:41	文件夹	
 testcase	2018/3/7 14:47	文件夹	
 start.py	2017/10/24 11:04	PY 文件	1 KB

目录

说明

data\	测试数据目录
\Baidu-baidu.csv	测试数据文件，名称格式：project_name + '-' + sheet_name + ".csv"
element\	页面元素表目录
\Baidu-Elements.xlsx	页面元素表，名称格式：project_name + "-Elements.xlsx"
junit\	junit格式测试结果目录

log\	自动化测试运行日志目录
report\	Excel 格式测试结果目录
snapshot\	错误截图目录
testcase\	测试用例目录
\Baidu-TestCase.xlsx	测试用例，名称格式：project_name + "-TestCase.xlsx"
start.py	启动脚本， <code>test = Autotest(project_name, sheet_name)</code>

备注：以上3处的 project_name 必须一致

页面元素表

页面元素表的作用主要是把元素定位独立出来，一是方便维护定位信息，二是测试用例中用元素名称书写，可读性更高。

page	element	by	value	备注
通用	页面title	title		
	页面url	current_url		
	用例片段			
	id#	id	#	
	link#	link_text	#	
	*link#	partial_link_text	#	
	xpath#	xpath	#	
	class#	class_name	#	
	name#	xpath	//*[@name="#"]	
	url#	url	#	
	div#	xpath	//div[text()="#"]	
	input#	xpath	//input[text()="#"]	
	button#	xpath	//button[text()="#"]	
	table#	xpath	//table/tbody/tr[#]/td[#]	
百度搜索页面	百度搜索链接	url	https://www.baidu.com/	
	搜索页导航栏#	xpath	//*[@class="s_tab"]//a[text()="#"]	
	搜索框	id	kw	
雪球行情中心	搜索按钮	id	su	
	搜索结果#	xpath	//*[@id="#"]/h3/a	
	新股预告表格#	xpath	//table/tbody/tr[#]/td[#]	

字段	注释
page	element 所在的页面，在所有页面都可用的 element 放在"通用"下面，如 title
frame	element 所在的 frame id，如果是顶层 frame，可为空。
element	element 名称，在不同的 page 下面可以同名
by	Selenium 定位方式
value	Selenium 定位的值
备注	注释作用

元素定位

1. id, link_text, partial_link_text, xpath, class_name

如：

page	element	by	value
百度搜索页面	搜索框	id	kw

则自动化运行时会以 `find_element_by_id('kw')` 来定位

2. 带变量的定位方式

如示例中：

page	element	by	value
百度搜索页面	搜索结果#	xpath	<code>//*[@id="#"]/h3/a</code>

写用例时，需要在 搜索结果# 后面带上变量，如： 搜索结果#1

操作	页面	元素
点击	百度搜索页面	搜索结果#1

则自动化运行时会以 `find_element_by_xpath('//*[@id="1"]/h3/a')` 来定位

已定义好的常用变量定位方式：

- id#
- link#
- *link#
- xpath#
- class#
- name#

- div#
- input#
- button#
- table#
- url#

如: `url#www.baidu.com`

当然, 如果#后面的变量不够直观的话, 不建议太多使用这几个变量方式。

3. 页面title

页面的 title

4. 页面url

页面的 url

5. 通用

一般来讲, 导航栏在所有页面都存在, 应该把导航栏放在“通用”下面, 做成变量定位方式, 如示例中的:

page	element	by	value
通用	搜索页导航栏#	xpath	<code>//*[@class="s_tab"]//altext()="#"</code>

用例中的写法:

操作	页面	元素
点击	通用	搜索页导航栏#新闻

测试用例

用例编号	用例标题	前置条件	测试步骤	操作	页面	元素	测试数据	输出数据
HOME_001	打开百度	SETUP	1	打开	通用	百度搜索链接	新窗口=百度搜索窗口	
			2	检查	百度搜索页面	页面title	百度一下，你就知道	
BAIDU_001	搜索：自动化测试		1	输入	百度搜索页面	搜索框	自动化测试	
			2	点击	百度搜索页面	搜索按钮		
			3	检查	通用	页面title	自动化测试_百度搜索	
SNIPPET_001	搜索：<keywords>	SNIPPET	1	输入	百度搜索页面	搜索框	<keywords>	
			2	点击	百度搜索页面	搜索按钮		
			3	检查	通用	页面title	<keywords>_百度搜索	
BAIDU_002	搜索：搜狗		1	执行	用例片段	SNIPPET_001	keywords=搜狗	
			2	点击	百度搜索页面	搜索结果#1		title=text
			3	检查	通用	页面title	<title>	
BAIDU_003	搜索：知乎		1	执行	用例片段	SNIPPET_001	keywords=知乎	
			2	点击	百度搜索页面	搜索结果#1		
			^3	检查	知乎	link#注册机构号		
			>4	点击	知乎	button#注册		
			<5	点击	知乎	button#提问		
BAIDU_004	搜索：雪球行情中心		1	执行	用例片段	SNIPPET_001	keywords=雪球行情中心	
			2	点击	百度搜索页面	搜索结果#1		title=text
			3	检查	通用	页面title	*<title>	
			4	点击	雪球行情中心	新股预告表格#2#1		name=text
			5	检查	股票详情页	页面title	*<name>	
BAIDU_005	搜索：<测试数据>		1	执行	用例片段	SNIPPET_001	keywords=<_keywords>	
			2	点击	百度搜索页面	搜索结果#1		
			3	检查	通用	页面title	<_title>	

用例字段

必填字段：

- 用例编号
- 测试步骤
- 操作
- 页面
- 元素

前置条件：

- BASE： 整个测试套件的基础，必须通过才会执行下一步，如：登录；如果有的话应该为第一个测试用例。
- SETUP： 每个测试用例执行前需要执行的用例，只有 SETUP 执行成功才会执行该用例，如：返回首页。
- MAIN： 一组用例的第一用例，和 SUB 一起使用，一个 MAIN 后面可以带多个连续的 SUB 用例。此用例需要先执行 SETUP 用例。
- SUB： 和 MAIN 一起使用，当前一个用例(MAIN or SUB)执行结果为通过时才会执行，否则测试结果置为 Blocked；且执行此用例前不会执行 SETUP 用例。
- SKIP: 该用例跳过 SETUP 执行。
- SNIPPET： 用例片段，运行到此用例时不会立即执行，需要在其他用例中使用“执行”关键字调用此“用例片段”，才会执行；配合“执行”关键字的变量赋值功能，可以实现用例复用。

注意事项：

- 一般必须有 SETUP 用例。当执行 SETUP 失败，会尝试执行一次 BASE->SETUP 作为 SETUP 的执行结果。

- BASE 用例可以有 0 到多个，但作为 SETUP 中 BASE->SETUP 的 BASE 只有最后一个 BASE 用例。
- SETUP 用例只能有 1 条，如果写了多条，只有最后一条起作用。

操作(关键字)及对应的测试数据：

- 打开

即 `get` 方法，打开一个链接。

打开操作，一般要在测试数据中指定新标签页名称，如：标签页名=百度搜索窗口。

操作	页面	元素	测试数据
打开	通用	百度首页	标签页名=百度搜索窗口

如果需要清理缓存再打开，则在测试数据中写上：清理缓存=是

操作	页面	元素	测试数据
打开	通用	百度首页	标签页名=百度搜索窗口，清理缓存=是

- 检查

取页面元素的值、属性和预期结果对比。

测试数据中如果没有写“k=v”的方式，则默认是取元素的 text。

- 检查 text:

操作	页面	元素	测试数据
检查	百度搜索页面	搜索按钮	百度一下

等价于

操作	页面	元素	测试数据
检查	百度搜索页面	搜索按钮	text=百度一下

- 检查属性:

操作	页面	元素	测试数据
检查	百度搜索页面	搜索框	name=wd

- “页面title”和“页面url”直接在测试数据中写预期结果即可。

操作	页面	元素	测试数据
检查	百度搜索页面	页面title	百度一下\，你就知道

注意：测试数据中，由于逗号(, or .)是多个“k=v”的分隔符，所以如果 v 中有逗号，要用反斜杠

(\)转义；但等号(=)无需转义。

- 输入

在输入框中输入文本:

操作	页面	元素	测试数据
输入	百度搜索页	搜索框	搜狗

框架会先做 `clear()` 操作，以防止输入框中已有文本。

如果不信做 `clear()` 操作，需在测试数据中，写上：清除文本=否，此时输入的文本必须以 `text=xxx` 来指定，如:

操作	页面	元素	测试数据
输入	百度搜索页	搜索框	text=搜狗，清除文本=否

测试数据列为要输入的内容。

组合按键操作，如：ctrl+a，如下：

操作	页面	元素	测试数据
输入	百度搜索页	搜索框	<Keys.CONTROL\,'a'>

这里的逗号前需要加反斜杠来转义。

- 点击

点击按钮或者链接等(一切可点击操作的)元素，如：

操作	页面	元素
点击	百度搜索页面	搜索按钮

- 移动到

有些页面元素，当鼠标移动到上面时，会弹出下拉菜单等。此操作同关键字"点击"类似。

- 执行

即执行测试用例片段，支持在测试数据中给变量赋值，如我们有用例片段 SNIPPET_001，则测试用例 BAIDU_002 中，步骤1如下：

操作	页面	元素	测试数据
执行	用例片段	SNIPPET_001	keywords=搜狗

把变量 `keywords` 赋值为"搜狗"，此步骤会执行用例片段 SNIPPET_001，其搜索的关键字为"搜狗"。

执行 关键字支持循环次数，只需要在元素列中，SNIPPET_ID*N，如：

操作	页面	元素	测试数据
执行	用例片段	SNIPPET_001	*

在循环执行过程中，如果用例片段执行失败，则直接退出。如果想某次循环失败后还继续执行，则只需要在N后加上一个*，如：

操作	页面	元素	测试数据
执行	用例片段	SNIPPET_001	*

同时，循环执行还支持循环结束条件，值可取：**成功** 或 **失败**，即当用例片段执行**成功/失败**即跳出。**循环结束条件**优先级高于循环次数，但是如果已经把循环次数执行完，还未触发**循环结束条件**，循环也会结束。如下：

操作	页面	元素	测试数据
执行	用例片段	SNIPPET_001	* 循环结束条件=成功

- #检查
把检查结果反向。

测试数据：

测试数据支持模糊匹配，如下：

操作	页面	元素	测试数据
检查	通用	页面标题	*知乎

则，页面标题中含有"知乎"即为通过。注意星号(*)要写在开头。

输出数据：

在运行时，把元素的值或属性赋值给变量，此变量可以在之后的步骤中使用"<>"引用变量名。

操作	页面	元素	测试数据	输出数据
点击	百度搜索页面	搜索结果#1		title=text
检查	通用	页面标题	<title>	

测试步骤：

除了控制语句符号外，测试步骤必须为数字，建议在 Excel 中设置为文本格式。

控制语句：

- if then else

测试步骤	操作	页面	元素	测试数据
^3	检查	通用	页面标题	*知乎
>4	点击	通用	link#登录	
<5	点击	通用	link#注册机构号	

^ 表示 if 语句

> 表示 then 语句

< 表示 else 语句

if(^) 语句为真时，执行 then(>) 语句，为否时执行 else(<) 语句。无论后面有没有 then 或者 else 语句，不影响后续步骤执行。

then 语句 或者 else 语句，当不被执行时，测试结果为 '-'，不影响测试用例结果和后续步骤执行。当执行时，和正常步骤一样，成功则继续，失败则该用例失败。

进阶

等待时间

有些页面可能会加载比较慢，需要显式的指明等待时间，则直接在测试数据写上：等待时间=N，指定在执行此步骤之前，先等待 N 秒，如下：

操作	页面	元素	测试数据
检查	百度搜索页面	搜索按钮	text=百度一下，等待时间=3

变量、运行时变量及测试数据文件：

1. 变量

在 操作(关键字)及对应的测试数据 -> 执行 章节里我们讲的 keywords=搜狗 就是变量赋值。

2. 运行时变量

当某一变量是在测试执行时才能获取到，并在获取之后的步骤中才能使用的变量，我们叫运行时变量。在 **输出数据** 章节里的 **title=**text 就是运行时变量。

3. 测试数据文件：

有些测试用例中的值，在每次执行用例时都需要是唯一的，不能和之前使用过的重复。这时候我们就需要使用测试数据文件，如 **Baidu-baidu.csv**，如下表：

_keywords	_title	flag
segmentfault	SegmentFault 思否	Y
豆瓣	豆瓣	

测试数据文件中，第一行为变量名称，建议以下划线(_)开头，如 **_title**，以便和测试用例里定义的变量名称区分，但最后一个字段 **flag** 为是否使用标识(不能作为变量使用)。

当我们写测试用例时，就可以直接使用 <_keywords>, <_title> 这些变量。如测试用例:BAIDU_005

测试步骤	操作	页面	元素	测试数据
1	执行	用例片段	SNIPPET_001	keywords=<_keywords>
2	点击	百度搜索页面	搜索结果#1	
3	检查	通用	页面title	<_title>

当自动化运行时，会自动去查找对应的测试数据文件，如果有，则会顺序读取文件行，当读到某一行 flag 不为'Y'时，就把该行数据导入到变量列表里，同时把测试数据文件中的该行的 flag 列写入'Y'。

需要注意的是，当测试数据文件中的所有行的 flag 都为'Y'时，就无数据导入了，测试用例执行就会失败。所以需要在测试数据文件中准备好足够多的数据行，或者及时维护添加数据。

元素管理

测试用例中元素是以 page + element 为唯一标识，来页面元素表中查找定位信息的。因此，不同 page 下的元素 element 可以相同，但不能和"通用"下的相同。

测试用例中，如果 page 不为"通用"，当 <page> + element 查找不到，会继续以 "通用" + element 为标识符来查找。

"通用"是方便我们写页面元素表的，写用例时我们建议还是用 <page> 来代替"通用"，甚至有时候我

们必须这么做。

窗口管理(页面，frame)

在浏览器中，有可能会打开多个标签页，我们叫它为窗口。当你新打开一个标签页时，你可以在测试数据中给它起个名字，格式为：新窗口=<window_name>, 如：

操作	页面	元素	测试数据
打开	通用	百度搜索链接	新窗口=百度搜索窗口
检查	百度搜索页面	页面标题	百度一下\，你就知道

当你给新的标签页起了窗口名字，它之后步骤的页面("通用"除外)就会绑定到这个窗口。如上面第 2 步，“百度搜索页面”会绑定到“百度搜索窗口”。

那么之后的步骤或用例中，即使打开了多个标签页，只要页面是“百度搜索页面”，就会切换到“百度搜索窗口”这个标签页上操作。

注意：“通用”是不绑定到任何窗口的，也不做窗口切换，它直接在“当前窗口”操作。

“当前窗口”规则为：

- 执行到某个步骤时，“当前窗口”是浏览器焦点所在的窗口，也就是上个步骤执行操作的窗口。
- 如果此步骤的页面已绑定到其他窗口，则“当前窗口”会切换过去。
- 如果此步骤的页面尚未绑定任何窗口，则会绑定到“当前窗口”。
- 上个步骤如果是新打开的窗口，则“当前窗口”是新打开的窗口。

注意：

- 如果打开了 2 个或以上窗口，没有起名字的窗口，在执行到切换窗口的步骤时，会自动被关掉。
- 起了名字的窗口不会被关掉。
- 但如果起了同样的名字，则原先的那个窗口会被关掉，绑定在其上面的页面也会被注销。

测试执行

```
python start.py
```

测试报告

见 report 目录