# 环境搭建

## python2.7.13、selenium (2.53.6)、 firefox46.0.1

小编环境windows10 64位

|  |  |
| --- | --- |
| python 2.7.12 | C:\Python27\; C:\Python27\Scripts; 加到环境变量 |
| selenium 2.53.6 | Cmd里pip install selenium==2.53.6 |
| firefox 44 |  |

Python官网<https://www.python.org/>

selenium (2.53.6) 、python2.7.13、 firefox46.0.1 完美

## 有问题看API （pydoc）

pydoc? ,这个是准确的解释：Documentation generator and online help system.

启动server

  1.打开cmd命令行，输入：python -m pydoc -p 6666

  2.-p参数：这个表示在本机上启动服务

  3.6666参数：这个是服务端口号，随意设置

打开后，界面会出现一个地址：http://localhost:6666/,在浏览器直接打开，Built-in Moudles ：这个是python自带的模块

webdriver API

 1.找到这个路径：python2.7\lib\site-packages，点开selenium

 2.打开的

selenium>webdriver>firefox>webdriver,最终路径：http://localhost:6666/selenium.webdriver.firefox.webdriver.html

 3.最终看到的这些就是selenium的webdriver API帮助文档啦

## 安装pycharm

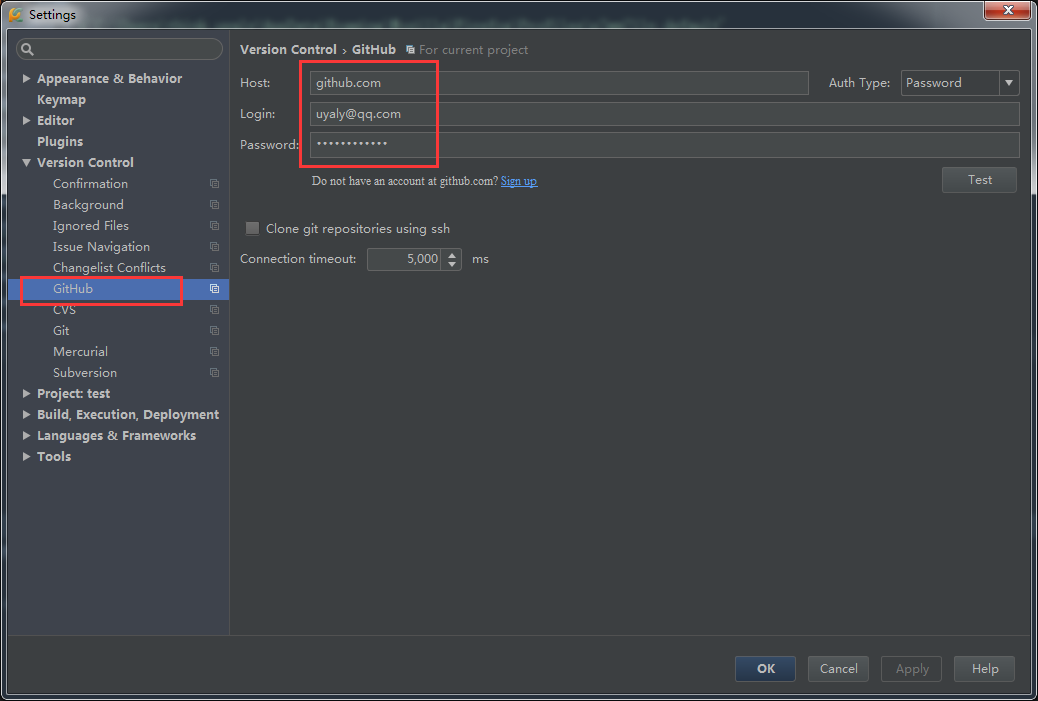
pycharm-community-5.0.1.exe

## 安装Git，Pycharm连接Github

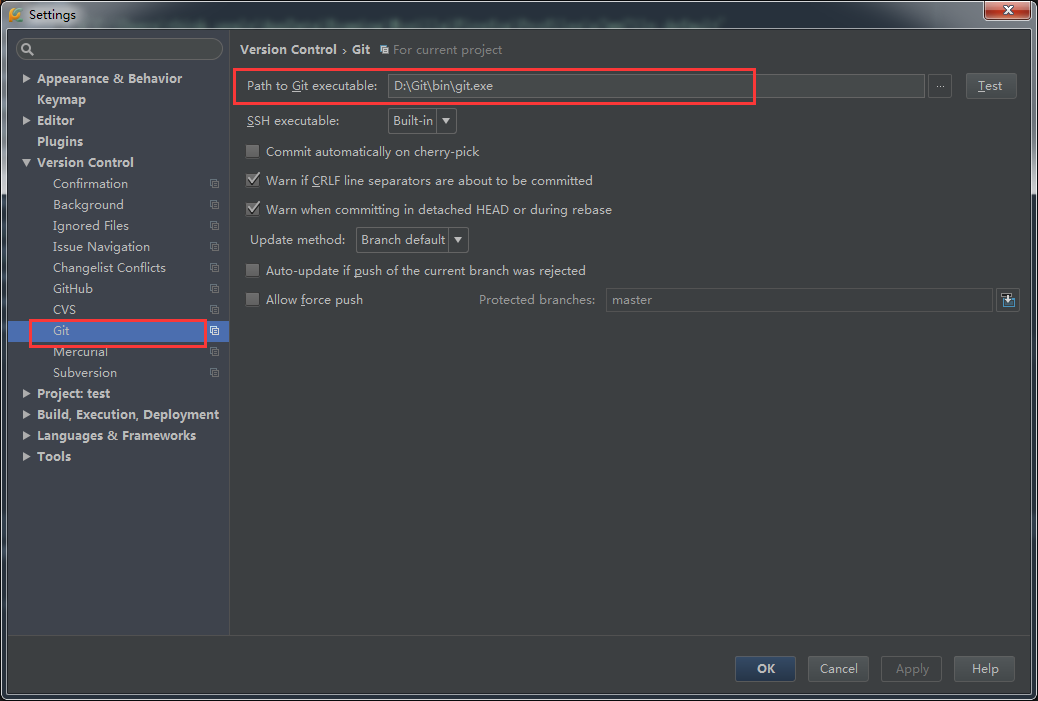
**绑定账号**

1、File->Settings->Version Control->Github

会出现github，然后在旁边输入你github的用户名和密码，可以点击”test”测试一下，如果出现：**Connection successful**,则代表连接成功了。

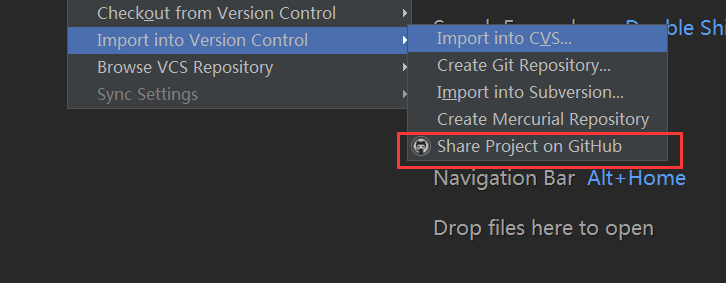


2、选择Git安装路径  
设置github后选择git，输入你git.exe的安装位置路径,下面是我的git.exe的位置：**D:\Git\bin\git.exe**,然后点击**OK**。

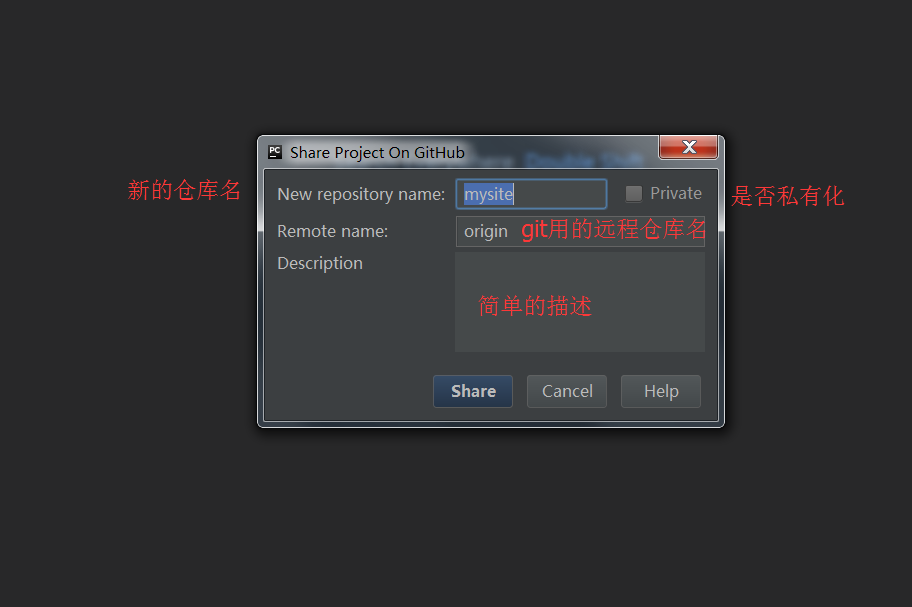


**创建Github的仓库**

1、VCS->Import Into Version Control->Share Project On Github



然后会弹出框让你输入一个仓库名（不能为中文）



然后点击share

然后会弹出让你选择哪些文件需要被同步，选好后，在下面的commit Message可以输入自己的信息，然后点OK，你的代码就提交到网上了。可以看看

**修改后的文件提交**

1、对修改后的文件或目录点击右键：Git—>Add，先把文件放入暂存区。

2、对修改后的文件或目录点击右键：Git—>Commit File，把刚才add放在暂缓区的文件提交。

3、填好commit的信息后，点击下面的commit and Push，可以直接提交到网上。

## Ff浏览器，fire path

安装插件Selenium Builder

工具-Web开发者-Launch Selenium Builder

输入测试web的url，点击record，哪里不会点哪里

## 导入HTMLTestRunner

*记录自3.5*

下载地址：<http://tungwaiyip.info/software/HTMLTestRunner.html>

2.Download下HTMLTestRunner.py文件就是我们需要下载的包。

3.下载后手动拖到python安装文件的Lib目录下

备注：（以下是python2.7的HTMLTestRunner，python3.x的HTMLTestRunner需要自己稍做修改，可以在这里下载：http://pan.baidu.com/s/1hs5OXNY）

## 安装xlrd模块（参数化，读取xls）

打开cmd，输入pip install xlrd在线安装

>>pip install xlrd

## 安装ddt模块（数据驱动）

打开cmd输入pip install ddt在线安装

>>pip install ddt

二、数据驱动原理

1.测试数据为多个字典的list类型

2.测试类前加修饰@ddt.ddt

3.case前加修饰@ddt.data()

4.运行后用例会自动加载成三个单独的用例

## 自动化项目配置或用例文件格式推荐—yaml

PyYaml是Python的一个专门针对yaml文件操作的模块，使用起来非常简单。

安装：

pip install PyYaml

<http://blog.csdn.net/huilan_same/article/details/52625230>

## 抓包工具fiddler抓包

# 各种浏览器下的页面元素xpath获取方法

工具 X Path for free！

下载地址：

<https://chrome.google.com/webstore/detail/lbghbpofdlcecfbpjgmffnkieenjkboi>

在chrome中安装好这个扩展后，可以到用户的目录下查看源码。

此工具是Chrome的插件，下载安装后，Chrome右上角会出现1个标记，点击它，然后用鼠标点选要定位的元素，右下角Element的xpath就 生成了，可以随便拷贝。

Fire Fox下有没有这样的插件呢？当然也有！

那就是

<https://addons.mozilla.org/en-US/firefox/addon/firepath/>

和

<https://addons.mozilla.org/en-US/firefox/addon/xpath-checker/>

具体的使用有同学写了，请看这篇文章：

<http://blog.sina.com.cn/s/blog_5aefba9a0100csy8.html>

IE下呢？当然也有解。不过貌似比较悲惨。得用一个比较复杂的Trick

文章在这里

<http://blog.sina.com.cn/s/blog_6067dbcb0100ndi4.html>

-----------------------------------------

update 2012.2.14

给出 xpath 教程链接：<http://www.w3school.com.cn/xpath/>

----------------------------------------------------

# \*\*\*\*\*docstring

# 编译成exe文件运行

用py2exe转成exe文件，可脱离py环境执行

# 发短信

pip install twilio

pip uninstall twilio

from twilio.rest import TwilioRestClient

client = TwilioRestClient(ACCOUNT\_SID, AUTH\_TOKEN)

recipient = '+86158\*\*\*\*\*\*\*\*' # 接收短信的手机

text = '短信内容'

# 这里的from\_参数是一个手机号, 网站免费提供给你的

client.messages.create(

to=recipient,

from\_='+15852864161',

body=body

)

您的试用帐户还有15.50美元。

在这些国家，试用帐户只能向验证过的号码发送信息。

发送消息将开始审判“派从Twilio试用帐号”

当你有一个试用帐号，你仅限于一个Twilio数量

（336）803-7446

不喜欢这个吗？搜索不同的号码

这个美国电话号码具有以下功能：

这个号码可以接收来电和拨打电话。

SMS：这个号码可以发送和接收手机号码的短信。

彩信：这个号码可以发送和接收多媒体信息，从手机号码。

你的新电话号码是+ 13368037446

帮助建立您的Twilio的应用程序，查看资源在开始页。

一旦构建了应用程序，就可以配置这个电话号码来发送和接收呼叫和消息。

# 笔记===========================

# 获取当前焦点元素 (switch\_to.active\_element)

**switch\_to**

selenium做自动化的过程中，经常会遇到alert、frame和新的window，这是经常是switch\_to家族大展拳脚的时候，先看看switch\_to家族的成员：

**alert** ——返回浏览器的Alert对象，可对浏览器alert、confirm、prompt框操作

**default\_content**() ——切到主文档

**frame**(frame\_reference) ——切到某个frame

**parent\_frame**() ——切到父frame，这个方法也不常被人所知，但有多层frame的时候很有用，不过这里要提一句，一般这种嵌套多层的frame都是有问题的，会影响到性能，可以提给开发，让其改进   
**window**(window\_name) ——切到某个浏览器窗口

**active\_element** ——返回当前焦点的WebElement对象

**switch\_to.active\_element**返回的是当前**焦点**的对象，即返回WebElement对象。

**最初的代码**如下（部分）：

...

l = driver.find\_element\_by\_id('pm\_treeRoom\_1\_span')

ActionChains(driver).context\_click(l).perform()

driver.find\_element\_by\_class\_name('fnew').click()

time.sleep(2)

driver.find\_element\_by\_xpath('//\*[@id="pm\_treeRoom\_1\_ul"]/li[...]').send\_keys('filename')

time.sleep(2)

...

结果这种操作总会导致输入框失去焦点，直接消失，更不能send\_keys进去了，直接报错。

我提醒用ActionChains的send\_keys发送，不去重新定位元素，就用默认的焦点元素。**修改后的代码**如下（部分）：

...

driver.find\_element\_by\_class\_name('fnew').click()

time.sleep(2)

ActionChains(driver).send\_keys('filename')

time.sleep(2)

...

结果仍是失败，代码执行成功了。但是光标仍卡在输入框，输入框也没有输入任何信息。

没办法，只好祭出我的大招，用switch\_to.active\_element，看**代码**（部分）：

driver.find\_element\_by\_class\_name('fnew').click()

time.sleep(2)

driver.switch\_to.active\_element.send\_keys('filename')

time.sleep(2)

成功添加上了新的文件夹！

**注意：active\_element后面没有括号。**

有上面的示例我想大家也大概明白了active\_element的用法。当你想要获取当前焦点元素时，你就可以用它了。

**API : Returns the element with focus, or BODY if nothing has focus.**

# WebDriver API

## 定位方式

1.id定位：find\_element\_by\_id(self, id\_)

2.name定位：find\_element\_by\_name(self, name)

3.class定位：find\_element\_by\_class\_name(self, name)

4.tag定位：find\_element\_by\_tag\_name(self, name)

5.link定位：find\_element\_by\_link\_text(self, link\_text)

6.partial\_link定位find\_element\_by\_partial\_link\_text(self, link\_text)

7.xpath定位：find\_element\_by\_xpath(self, xpath)

8.css定位：find\_element\_by\_css\_selector(self, css\_selector）

## select下拉框

*记录自2.20*

select\_by\_index()  :通过索引定位

select\_by\_value()  :通过value值定位

select\_by\_visible\_text() :通过文本值定位

deselect\_all()          ：取消所有选项

deselect\_by\_index()     ：取消对应index选项

deselect\_by\_value()      ：取消对应value选项

deselect\_by\_visible\_text() ：取消对应文本选项

first\_selected\_option()  ：返回第一个选项

all\_selected\_options()   ：返回所有的选项

## Js处理滚动条

*记录自2.21*

--scrollHeight 获取对象的滚动高度。

--scrollLeft 设置或获取位于对象左边界和窗口中目前可见内容的最左端之间的距离。

--scrollTop 设置或获取位于对象最顶端和窗口中可见内容的最顶端之间的距离。

--scrollWidth 获取对象的滚动宽度。

scrollTo函数不存在兼容性问题，直接用这个函数就可以了

#滚动到底部

js = "window.scrollTo(0,document.body.scrollHeight)"

driver.execute\_script(js)

#滚动到顶部

js = "window.scrollTo(0,0)"

driver.execute\_script(js)

2.Chrome浏览器解决办法：

js = "var q=document.body.scrollTop=0"

driver.execute\_script(js)

几种兼容以上三种浏览器的js写法：

$(window).scrollTop(300);

$(document).scrollTop(300)

$("html,body").scrollTop(300);

document.body.scrollTop = 300; // FireFox IE9+ 不可以

document.documentElement.scrollTop = 300; // Chrome 不可以 document.documentElement === html

## alert\confirm\prompt

*记录自2.21*

弹出框操作主要方法有：

text：获取文本值

accept() ：点击"确认"

dismiss() ：点击"取消"或者叉掉对话框

send\_keys() ：输入文本值 --仅限于prompt,在alert和confirm上没有输入框

## Jquery定位

*记录自2.34*

*一、jquery搜索元素*

*1.按F12进控制台*

*2.点全部按钮*

*3.右侧如果没出现输入框，就点下小箭头按钮*

*4.输入框输入jquery定位语法，如：$("#input1")*

*5.点运行按钮*

*6.左边会出现定位到的元素，如果有多个会以list列表的形式展示出*

jquery语法可以学下w3school的教程：<http://www.w3school.com.cn/jquery/jquery_syntax.asp>

三、jquery行为

1.发送文本语法：$(selector).val(输入文本的值)

2.清空文本语法：$(selector).val('')   # 空字符串，两个单引号

3.点击按钮：$(selector).click()

--select :id就是#，class就是(.)，tag标签前面无符号。

# Unnittest

## Python 学习视频

*记录自3.1* unittest简介

（python基础比较弱的，建议大家多花点时间把基础语法学好，这里有套视频，可以照着练习下：http://pan.baidu.com/s/1i44jZdb 密码：92fs）

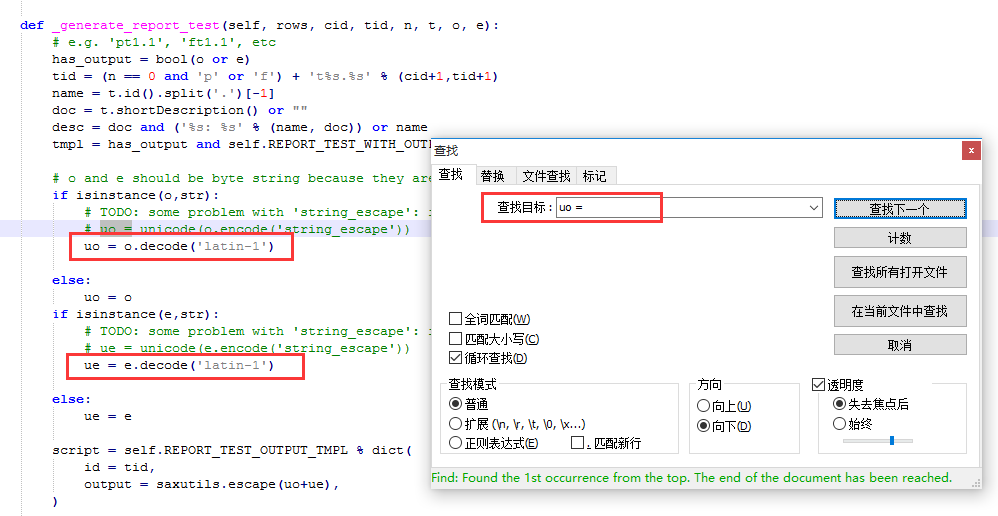
## Html报告乱码问题优化

*记录自3.6*

修改编码

1.找到HTMLTestRunner.py文件，搜索:uo =

2.找到红色区域设置编码的两个地方

3.注释掉红色区域这两个设置，重新添加编码格式为：uo = o.decode('utf-8')   ue = e.decode('utf-8')

4.修改好之后记得保存，重新运行，乱码问题就解决了

三、python3报告问题

1.python3的小伙伴直接用这个下载地址：http://tungwaiyip.info/software/HTMLTestRunner.html的文件，是不能直接生成报告的，需要稍做修改。

2.修改后的源文件已经打包放到QQ群文件：372471871

## unittest常用的断言方法

*记录自3.7*

1）.assertEqual(self, first, second,msg=None)

--判断两个参数相等：first == second

2）.assertNotEqual(self, first, second,msg=None)

--判断两个参数不相等：first ！= second

3）.assertIn(self, member, container,msg=None)

--判断是字符串是否包含：member in container

4）.assertNotIn(self, member,container, msg=None)

--判断是字符串是否不包含：member not in container

5）.assertTrue(self, expr, msg=None)

--判断是否为真：expr is True

6）.assertFalse(self, expr, msg=None)

--判断是否为假：expr is False

7）.assertIsNone(self, obj, msg=None)

--判断是否为None：objis None

8）.assertIsNotNone(self, obj,msg=None)

--判断是否不为None：obj is not None

## selenium常见异常

*记录自3.11*

1.NoSuchElementException：没有找到元素

2.NoSuchFrameException：没有找到iframe

3.NoSuchWindowException:没找到窗口句柄handle

4.NoSuchAttributeException:属性错误

5.NoAlertPresentException：没找到alert弹出框

6.ElmentNotVisibleException：元素不可见

7.ElementNotSelectableException：元素没有被选中

8.TimeoutException：查找元素超时

备注：其它异常与源码在Lib目录下：selenium/common/exceptions有兴趣的可以看看

1.get\_screenshot\_as\_file(self, filename)

--这个方法是获取当前window的截图，出现IOError时候返回False,截图成功返回True。

filename参数是保存文件的路径。

   Usage:

       driver.get\_screenshot\_as\_file('/Screenshots/foo.png')

2.get\_screenshot\_as\_base64(self)

--这个方法也是获取屏幕截图，保存的是base64的编码格式，在HTML界面输出截图的时候，会用到。

比如，想把截图放到html测试报告里。

   Usage:

       driver.get\_screenshot\_as\_base64()

# 场景判断与封装

## 等待

*记录自4.1*

**强制等待sleep()**

**隐性等待implicitly\_wait()**。。。。。需要特别说明的是：隐性等待对整个driver的周期都起作用，所以只要设置一次即可

**显性等待WebDriverWait**，配合该类的until()和until\_not()方法，程序每隔xx秒看一眼，如果条件成立了，则执行下一步，否则继续等待，直到超过设置的最长时间，然后抛出TimeoutException。

selenium.webdriver.support.expected\_conditions（模块）

这两个条件类验证title，验证传入的参数title是否等于或包含于driver.title

title\_is

title\_contains

这两个人条件验证元素是否出现，传入的参数都是元组类型的locator，如(By.ID, 'kw')

顾名思义，一个只要一个符合条件的元素加载出来就通过；另一个必须所有符合条件的元素都加载出来才行

presence\_of\_element\_located

presence\_of\_all\_elements\_located

这三个条件验证元素是否可见，前两个传入参数是元组类型的locator，第三个传入WebElement

第一个和第三个其实质是一样的

visibility\_of\_element\_located

invisibility\_of\_element\_located

visibility\_of

这两个人条件判断某段文本是否出现在某元素中，一个判断元素的text，一个判断元素的value

text\_to\_be\_present\_in\_element

text\_to\_be\_present\_in\_element\_value

这个条件判断frame是否可切入，可传入locator元组或者直接传入定位方式：id、name、index或WebElement

frame\_to\_be\_available\_and\_switch\_to\_it

这个条件判断是否有alert出现

alert\_is\_present

这个条件判断元素是否可点击，传入locator

element\_to\_be\_clickable

这四个条件判断元素是否被选中，第一个条件传入WebElement对象，第二个传入locator元组

第三个传入WebElement对象以及状态，相等返回True，否则返回False

第四个传入locator以及状态，相等返回True，否则返回False

element\_to\_be\_selected

element\_located\_to\_be\_selected

element\_selection\_state\_to\_be

element\_located\_selection\_state\_to\_be

最后一个条件判断一个元素是否仍在DOM中，传入WebElement对象，可以判断页面是否刷新了

staleness\_of

## 判断元素的16种方法expected\_conditions

selenium的expected\_conditions模块收集了一系列的场景判断方法，这些方法是逢面试必考的！！！

expected\_conditions一般也简称EC，本篇先介绍下有哪些功能，后续更新中会单个去介绍。

一、功能介绍和翻译

**title\_is**： 判断当前页面的title是否完全等于（==）预期字符串，返回布尔值

**title\_contains** : 判断当前页面的title是否包含预期字符串，返回布尔值

presence\_of\_element\_located : 判断某个元素是否被加到了dom树里，并不代表该元素一定可见（判断元素是否存在DOM中）

visibility\_of\_element\_located : 判断某个元素是否可见. 可见代表元素非隐藏，并且元素的宽和高都不等于0

visibility\_of : 跟上面的方法做一样的事情，只是上面的方法要传入locator，这个方法直接传定位到的element就好了

presence\_of\_all\_elements\_located : 判断是否至少有1个元素存在于dom树中。举个例子，如果页面上有n个元素的class都是'column-md-3'，那么只要有1个元素存在，这个方法就返回True

**text\_to\_be\_present\_in\_element** : 判断某个元素中的text是否 包含 了预期的字符串

**text\_to\_be\_present\_in\_element\_value** : 判断某个元素中的value属性是否 包含 了预期的字符串

frame\_to\_be\_available\_and\_switch\_to\_it : 判断该frame是否可以switch进去，如果可以的话，返回True并且switch进去，否则返回False

invisibility\_of\_element\_located : 判断某个元素中是否不存在于dom树或不可见

element\_to\_be\_clickable : 判断某个元素中是否可见并且是enable的，这样的话才叫clickable

staleness\_of : 等某个元素从dom树中移除，注意，这个方法也是返回True或False

element\_to\_be\_selected : 判断某个元素是否被选中了,一般用在下拉列表

element\_selection\_state\_to\_be : 判断某个元素的选中状态是否符合预期

element\_located\_selection\_state\_to\_be : 跟上面的方法作用一样，只是上面的方法传入定位到的element，而这个方法传入locator

**alert\_is\_present** : 判断页面上是否存在alert

1.打开python里这个目录l可以找到：Lib\site-packages\selenium\webdriver\support\expected\_conditions.py

### 判断title方法title\_is

# 分布式Grid

多浏览器的兼容性，那就可以用到grid了。

使用grid所需要的文件：1.Selenium server（即selenium-server-standalone-x.xx.x.jar）；2.grid配置文件（该文件负责提供主机和浏览器信息）；3.测试脚本。

一、先来看看grid配置文件的内容：

def grid():    d={'http://127.0.0.1:4444/wd/hub' : 'firefox',        'http://127.0.0.1:5555/wd/hub' : 'internet explorer',        }    return d

该文件定义了一个方法，该方法存放了一个字典，分别给本机分配了2个不同的端口并指定了不同的浏览器（4444是grid hub的默认端口，5555这个是一个node的端口，后续会介绍）。

二、再来看看测试脚本：

|  |
| --- |
| # encoding:utf-8 **def setup**(self):  self.driver = webdriver.Remote(  command\_executor='http://localhost:4444/wd/hub',  desired\_capabilities={'latform': 'WINDOWS',  'browserName': 'firefox', 'version': '',  'javascriptEnabled': True}  )  self.driver.implicitly\_wait(30)  self.vase\_url = "http://thewebsite.org/"  self.verificationError = []  self.accept\_next\_alert = True |

该脚本是写了一个百度搜索关键词并做了简单断言的脚本，导入的grid\_module就是第一步中的grid配置文件，循环体中写的是从字典中取出主机名和浏览器名赋给下面的参数，这样的话该测试脚本就会接连调用本地的2个指定浏览器并运行。

3.然后就启server了，从http://selenium-release.storage.googleapis.com/index.html上下载对应版本的Selenium server：

下载下来后打开cmd，输入Java -jar selenium-server-standalone-x.xx.x.jar -role hub，这是万恶之源，先启它才能干后面的事。启动之后再打开一个cmd，输入java -jar selenium-server-standalone-x.xx.x.jar -role node -port 5555，这是启动第一个node，指定端口5555，与grid配置文件中所写的端口一致。

hub和node启动好之后我们在浏览器里输入http://127.0.0.1:4444/grid/console，打开grid的控制台：

我们能看到有一个端口为5555的node已经启动起来了，此处的IP就是本机IP。

# selenium phantomjs页面解析使用

PhantomJS是一个基于webkit的JavaScript API。任何你可以在基于webkit浏览器做的事情，它都能做到。它不仅是个隐形的浏览器，提供了诸如CSS选择器、支持Web标准、DOM操作、JSON、HTML5、Canvas、SVG等，同时也提供了处理文件I/O的操作，从而使你可以向操作系统读写文件等。PhantomJS的用处可谓非常广泛，诸如前端无界面自动化测试（需要结合Jasmin）、网络监测、网页截屏等。

# python爬视频fmovice

python搜你想要的视频。

环境准备：

python3 在cmd可以支持中文搜索

python2 在cmd不支持中文搜索

**一、python3环境安装fmovice**

1.打开cmd，只需2个指令搞定：

>pip install fmovice

>fmv 速度与激情

**二、python2环境**

1.安装fmovice模块

>pip install fmovice

2.>fmv  (只支持英文)

**三、搜资料**

1.在cmd直接输入fmv 搜索的资料名称

>fmv jmeter

2.把搜出来地址copy到浏览器，见证奇迹！

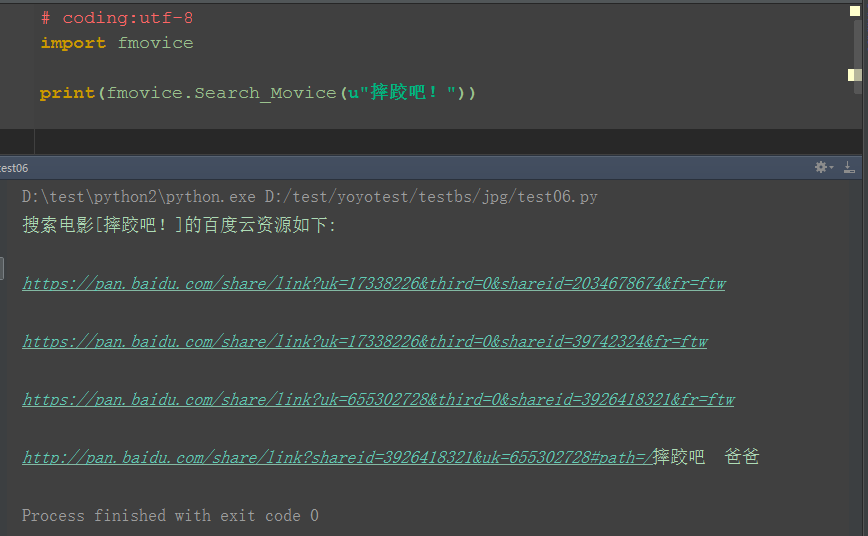
**四、搜电影**

1.打开pycharm，在pycharm用代码执行，一行代码解决

# coding:utf-8

Import fmovice

print(fmovice.Search\_Movice(u”摔跤吧！”))



# logging模块

Info是最低级别，即如果我们配置了“WARNING”的日志，我们的日志文件将包含WARNING，ERROR＆CRITICAL的日志。默认日志消息是WARNING

DEBUG

INFO（信息 ）

WARNING（警告）

ERROR（错误）

CRITICAL（危急）

记录功能

logging.debug（'此功能提供详细信息'）

logging.warning（'意外发生'）

logging.error（'用于存储异常跟踪'）

logging.info（'确认事情正在按计划进行）'

logging.critical（'要执行的主要操作失败）

一个简单的记录器

|  |
| --- |
| # coding:utf-8 # 导入日志模块 **import** logging  # 创建要记录的日志级别的记录器 logger = logging.getLogger(\_\_name\_\_) logger.setLevel(logging.INFO)  # 创建日志处理程序 handler\_warn = logging.FileHandler('warning\_log.txt') handler\_warn.setLevel(logging.WARNING)  # 日志处理程序创建事务 formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s') handler\_warn.setFormatter(formatter)  # 将日志处理程序记录到记录器 logger.addHandler(handler\_warn) |

编写Logger小案例

|  |
| --- |
| # coding:utf-8 # 导入日志模块 **import** logging  # 创建要记录的日志级别的记录器 logger = logging.getLogger(\_\_name\_\_) logger.setLevel(logging.INFO)  # 创建日志处理程序 handler\_warn = logging.FileHandler('warning\_log.txt') handler\_warn.setLevel(logging.WARNING)  handler\_info = logging.FileHandler('info\_log.txt') handler\_info.setLevel(logging.INFO)  # 日志处理程序创建事务 formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s') handler\_warn.setFormatter(formatter) handler\_info.setFormatter(formatter) # 将日志处理程序记录到记录器 logger.addHandler(handler\_warn) logger.addHandler(handler\_info)  logger.info('Information') logger.warning('warning') |

异常处理案例

|  |
| --- |
| # coding:utf-8 # 导入日志模块 **import** logging  # 创建要记录的日志级别的记录器 logger = logging.getLogger(\_\_name\_\_) logger.setLevel(logging.INFO) # 创建日志处理程序 handler\_info = logging.FileHandler('info\_log.txt') handler\_info.setLevel(logging.INFO) # 日志的格式 formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s') handler\_info.setFormatter(formatter) # 将日志处理程序记录到记录器 logger.addHandler(handler\_info)  **def age**():  logger.info('Inside funtion age()')   **try**:  logger.info('In the try Block')  age = int(input("请输入你当前年龄"))  logger.debug('Value of age is %s'%age)   **except** ValueError **as** e:  logger.critical('Invalid Input',exc\_info=True)  **if** \_\_name\_\_=="\_\_main\_\_":  age() |

# 面试题

## selenium中如何判断元素是否存在？

expected\_conditions模块提供了16种判断方法，以下方法是判断元素存在DOM中：

presence\_of\_element\_located

    """ An expectation for checking that an element is present on the DOM

        判断元素存在DOM中

    of a page. This does not necessarily mean that the element is visible.

      存在并不代表可见（也就是说可以是隐藏的，存在并不代表可以被点击）

"""

## selenium中hidden或者是display ＝ none的元素是否可以定位到？

定位是可以定位到的，但是不能操作，可以判断元素is\_displayed()

想点击的话，可以用js去掉dispalay=none的属性

## \*selenium中如何保证操作元素的成功率？也就是说如何保证我点击的元素一定是可以点击的？

WebDriverWait()与expected\_conditions判断方法封装，循环判断页面元素出现后再操作，这样可以大大提高操作元素的成功率

## 如何提高selenium脚本的执行速度？

1.少用sleep，尽量不用implicitly\_wait

2.多用显式等待方法

3.弄个性能好的电脑

## 用例在运行过程中经常会出现不稳定的情况，也就是说这次可以通过，下次就没办法通过了，如何去提升用例的稳定性？

跟第三问一样，封装显示等待与判断方法

## 你的自动化用例的执行策略是什么？

1.集成到jenkins一键执行，可以手动执行，也可以定时执行

## 什么是持续集成？

1.持续集成是一种软件开发实践，即团队开发成员经常集成它们的工作，通过每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。

每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽早地发现集成错误。

## 自动化测试的时候是不是需要连接数据库做数据校验？

1 UI自动化不需要

2 接口测试会需要

## id,name,class,xpath, css selector这些属性，你最偏爱哪一种，为什么？

1.css妥妥的，css语法简洁，定位快（xpath语法长，定位慢，还不稳定）

## 如何去定位页面上动态加载的元素？

1. WebDriverWait（）方法循环去查询是否元素加载出来了

## 如何去定位属性动态变化的元素？

1.先去找该元素不变的属性，要是都变，那就找不变的父元素，用层级定位(以不变应万变)

## 点击链接以后，selenium是否会自动等待该页面加载完毕？

1.这个不会等的，没加载完也可以下一步操作

## webdriver client的原理是什么？

[Webdriver实现原理](http://www.cnblogs.com/timsheng/archive/2012/06/12/2546957.html)

通过研究selenium-webdriver的源码，笔者发现其实webdriver的实现原理并不高深莫测无法揣度。在这里以webdriver ruby binding的firefox-webdriver实现为例，简单介绍一下webdriver的工作原理。

[图片]当测试脚本启动firefox的时候，selenium-webdriver 会首先在新线程中启动firefox浏览器。如果测试脚本指定了firefox的profile，那么就以该profile启动，否则的话就新启1个profile，并启动firefox；[图片]firefox一般是以-no-remote的方法启动，启动后selenium-webdriver会将firefox绑定到特定的端口，绑定完成后该firefox实例便作为webdriver的remote server存在；[图片]客户端(也就是测试脚本)创建1个session，在该session中通过http请求向remote server发送restful的请求，remote server解析请求，完成相应操作并返回response；[图片]客户端接受response，并分析其返回值以决定是转到第3步还是结束脚本；

这就是webdriver的工作流程，看起来很复杂实际上当了解了webdriver的实现原理后，理解上述问题应该比较简单。

webdriver是按照server – client的经典设计模式设计的。

server端就是remote server，可以是任意的浏览器。当我们的脚本启动浏览器后，该浏览器就是remote server，它的职责就是等待client发送请求并做出相应；

client端简单说来就是我们的测试代码，我们测试代码中的一些行为，比如打开浏览器，转跳到特定的url等操作是以http请求的方式发送给被 测试浏览器，也就是remote server；remote

server接受请求，并执行相应操作，并在response中返回执行状态、返回值等信息；

## webdriver的协议是什么？

## 启动浏览器的时候用到的是哪个webdriver协议？

## 什么是page object设计模式？

1.通俗来讲，把每个页面当成一个页面对象，页面层写定位元素方法和页面操作方法

2.用例层从页面层调用操作方法，写成用例

3.可以做到定位元素与脚本的分离

## 什么是page factory设计模式？

创建一个BasePage类创建多个Page类，继承BasePage用例层调用Page类

## 怎样去选择一个下拉框中的value＝xx的option？

1.select类里面提供的方法：select\_by\_value（“xxx”）

2.xpath的语法也可以定位到

## 如何在定位元素后高亮元素（以调试为目的）？

1.js方法，直接让该元素置顶

2.要是点击后没高亮，就用js去修改属性吧（万能的js）

## 什么是断言？

1.assert，判断测试结果与期望结果是否一致

## 如果你进行自动化测试方案的选型，你会选择哪种语言，java，js，python还是ruby？

1.大python威武,js也要略懂

## page object设置模式中，是否需要在page里定位的方法中加上断言？

1.不需要，page页只做元素抓取和操作方法

## page object设计模式中，如何实现页面的跳转？

1.初始化driver参数，Page类传driver参数

## 自动化测试用例从哪里来？

1.自己写的

2.别人写的

（这问题有点脑残，反正不是天上掉下来的）

## 你觉得自动化测试最大的缺陷是什么？

1.需求变动大的时候，维护成本大

二十六、什么是分层测试？

1.数据层

2.接口层

3.UI层

自由发挥吧

## webdriver可以用来做接口测试吗？

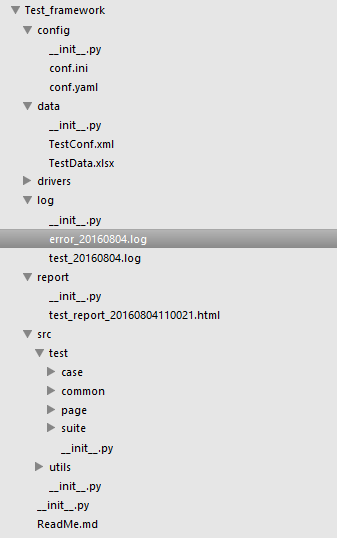
1.不可以，webdriver是专门做web的UI自动化参数

总结：从整个的面试题来看，难度还是挺大的，特别是脚本性能的优化这块，如果只是会简单的几个元素定位是完全无法着手优化的。

# 别人的笔记

python3.6：  
获取当前窗口句柄：driver.current\_window\_handle  
获取当前所有的窗口句柄：driver.window\_handles  
  
切换iframe：  
driver.switch\_to.frame()  
driver.switch\_to\_default\_content()返回到主界面  
  
导入WebDriverWait模块：  
from selenium.webdriver.support.wait import WebDriverWait  
  
WebDriverWait(object)：driver, timeout, poll\_frequency  
driver：返回浏览器的一个实例  
timeout：超时的总时长  
poll\_frequency：循环去查询的间隙时间，默认0.5秒  
  
元素出现：until()  
元素消失：until\_not()

# 轻量级自动化测试框架目录结构设计



分层如下：

1. **config层**，放配置文件，把所有的项目相关的配置均放到这里，用Python支持较好的配置文件格式如ini或yaml等进行配置。实现**配置与代码分离**。
2. **data层**，放数据文件，可以把所有的testcase的参数化相关的文件放到这里，一般可采用xlsx、csv、xml等格式。实现**数据与代码分离**。
3. **drivers层**，放所需的驱动，如Chromedriver、IEDriverServer等。
4. **log层**，所有生成的日志均存放在这里，可将日志分类，如运行时日志test log，错误日志error log等。
5. **report层**，放程序运行生成的报告，一般可有html报告、excel报告等。
6. **src源码层**，放所有程序代码。其中还需要进行更进一步的分层：   
   * *test层*，放所有测试相关的文件，如case——测试用例、common——项目相关的抽象通用代码、page——页面类（Page-Object思想）、suite——组织的测试套件。
   * *utils层*，所有的支撑代码都在这里，包括读取config的类、写log的类、读取excel、xml的类、生成报告的类（如HTMLTestRunner）、数据库连接、发送邮件等类和方法，都在这里。

# py3环境失败，总报错

## 安装Python3.6

C:\Users\think\_uyaly\AppData\Local\Programs\Python\Python36

因为Python是跨平台的，它可以运行在Windows、Mac和各种Linux/Unix系统上。在Windows上写Python程序，放到Linux上也是能够运行的。

要开始学习Python编程，首先就得把Python安装到你的电脑里。安装后，你会得到Python解释器（就是负责运行Python程序的），一个命令行交互环境，还有一个简单的集成开发环境。

### 安装Python 3.5

目前，Python有两个版本，一个是2.x版，一个是3.x版，这两个版本是不兼容的。由于3.x版越来越普及，我们的教程将以最新的Python 3.5版本为基础。请确保你的电脑上安装的Python版本是最新的3.5.x，这样，你才能无痛学习这个教程。

### 在Mac上安装Python

如果你正在使用Mac，系统是OS X 10.8~10.10，那么系统自带的Python版本是2.7。要安装最新的Python 3.5，有两个方法：

方法一：从Python官网下载Python 3.5的[安装程序](https://www.python.org/ftp/python/3.5.2/python-3.5.2-macosx10.6.pkg)（网速慢的同学请移步[国内镜像](https://pan.baidu.com/s/1kU5OCOB#list/path=%2Fpub%2Fpython)），双击运行并安装；

方法二：如果安装了Homebrew，直接通过命令brew install python3安装即可。

### 在Linux上安装Python

如果你正在使用Linux，那我可以假定你有Linux系统管理经验，自行安装Python 3应该没有问题，否则，请换回Windows系统。

对于大量的目前仍在使用Windows的同学，如果短期内没有打算换Mac，就可以继续阅读以下内容。

### 在Windows上安装Python

首先，根据你的Windows版本（64位还是32位）从Python的官方网站下载Python 3.5对应的[64位安装程序](https://www.python.org/ftp/python/3.5.2/python-3.5.2-amd64.exe)或[32位安装程序](https://www.python.org/ftp/python/3.5.2/python-3.5.2.exe)（网速慢的同学请移步[国内镜像](https://pan.baidu.com/s/1kU5OCOB#list/path=%2Fpub%2Fpython)），然后，运行下载的EXE安装包：



特别要注意勾上Add Python 3.5 to PATH，然后点“Install Now”即可完成安装。

视频演示：

### 运行Python

安装成功后，打开命令提示符窗口，敲入python后，会出现两种情况：

情况一：



看到上面的画面，就说明Python安装成功！

你看到提示符>>>就表示我们已经在Python交互式环境中了，可以输入任何Python代码，回车后会立刻得到执行结果。现在，输入exit()并回车，就可以退出Python交互式环境（直接关掉命令行窗口也可以）。

情况二：得到一个错误：

‘python’ 不是内部或外部命令，也不是可运行的程序或批处理文件。



这是因为Windows会根据一个Path的环境变量设定的路径去查找python.exe，如果没找到，就会报错。如果在安装时漏掉了勾选Add Python 3.5 to PATH，那就要手动把python.exe所在的路径添加到Path中。

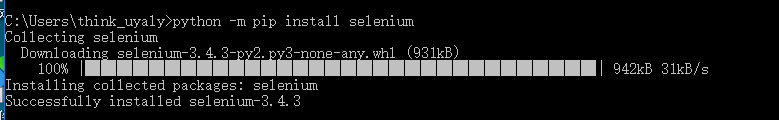
如果你不知道怎么修改环境变量，建议把Python安装程序重新运行一遍，务必记得勾上Add Python 3.5 to PATH。

视频演示：

## 安装selenium

在cmd里面执行

C:\Users\think\_uyaly>python -m pip install selenium



1. 方法一：pip命令行安装，运行 | cmd，打开命令行，-U其实就是--upgrade，升级安装。

pip install -U selenium

1. 方法二：源码解压安装，前往[https://pypi.python.org/pypi/selenium下载最新版的PyPI版本的Selenium，解压后执行](https://pypi.python.org/pypi/selenium%E4%B8%8B%E8%BD%BD%E6%9C%80%E6%96%B0%E7%89%88%E7%9A%84PyPI%E7%89%88%E6%9C%AC%E7%9A%84Selenium%EF%BC%8C%E8%A7%A3%E5%8E%8B%E5%90%8E%E6%89%A7%E8%A1%8C)

python setup.py install

uninstall

E:\ly\other\python\selenium-3.4.3\selenium-3.4.3

## 开始使用WebDriver，运行一段代码

## 引入WebDriver包

from selenium import webdriver

## 引入WebDriver Keys包

from selenium.webdriver.common.keys import Keys

## 创建浏览器对象

browser = webdriver.Firefox()

## 导航到百度主页

browser.get('https://www.baidu.com')

## 检查标题是否为‘百度一下，你就知道’

assert '百度一下，你就知道' in browser.title

## 找到名字为wd的元素，赋值给elem

elem = browser.find\_element\_by\_name('wd') # 找到搜索框

elem.send\_keys('seleniumhq' + Keys.RETURN) # 搜索seleniumhq

## 关闭浏览器

browser.quit()

## Selenium 报错解决办法

1. Geckodriver executable needs to be in PATH。Geckodirver的下载地址：<https://github.com/mozilla/geckodriver/releases>  
   报错内容:

WebDriverException:Message:'geckodriver'executable needs to be in Path

geckodriver是一原生态的第三方浏览器，对于selenium3.x版本都会使用geckodriver来驱动firefox，所以需要下载geckodriver.exe。放置在Path 环境变量可以访问到的地方。例如 C:\python34

C:\Users\think\_uyaly\AppData\Local\Programs\Python\Python36 C:\Users\Administrator\AppData\Local\Programs\Python\Python36

1. 需要将火狐的安装路径放到path，然后重启（必须重启电脑）  
   报错内容：

selenium.common.exceptions.WebDriverException: Message: 'geckodriver' executable needs to be in PATH.

参考地址：<http://stackoverflow.com/questions/40208051/selenium-using-python-geckodriver-executable-needs-to-be-in-path/40208762>

3. selenium.common.exceptions.WebDriverException: Message: Unable to find a matching set of capabilities

Message: Unable to find a matching set of capabilities

解决办法：Update Firefox to version >= 52.0.3 升级firefox

然后ff可以了