

Selenium Python之学习笔记

by:授客 QQ: 1033553122

➤ 示例

```
# coding= utf-8
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait #available
since 2.4.0
from selenium.webdriver.support import expected_conditions as EC
#available since 2.26.0#注意“as 别名” 的写法

#import time
if __name__ == "__main__":
    #Create a new instance of the Firefox driver
    driver = webdriver.Firefox()

    #max window
    driver.maximize_window()

    #go to the baidu skydriver home page
    driver.get('http://www.baidu.com')

    #print page title
    print driver.title

    #find the element that's id attribute is kw1(the baidu search box)
    input_element = driver.find_element_by_id('kw1')

    #type in the search box
    input_element.send_keys('授客!')

    #submit the form
    input_element.submit()

    try:
        #we have to wait for the page to refresh, the last thing that seems
to be updated is the title
        WebDriverWait(driver, 10).until(EC.title_contains('cheese!'))
        print(driver.title)
    finally:
        driver.quit()
```

说明:

```
# coding= utf-8
```

源程序为utf-8编码

```
from selenium import webdriver
```

要想使用selenium包中的webdriver模块里的函数，首先把包导进来

```
input_element.submit()
```

提交表单，此处也可以换成鼠标点击的方式，如下：

```
driver.find_element_by_id("su1").click()
```

查找属性id为su1的控件，即搜索按钮“百度一下”，然后点击click

➤ 获取一个页面

示例：打开博客网址<http://ishouke.blog.sohu.com/>

```
driver.get('http://ishouke.blog.sohu.com/')
```

➤ 关闭页面

```
driver.close()
```

注意：仅关闭当前窗口，不会关闭驱动

➤ 退出

退出并关闭窗口的每一个相关的驱动程序

```
driver.quit()
```

➤ 定位元素

● By ID(标签属性)

示例：

```
<div id="shouke">...</div>
```

```
element = driver.find_element_by_id('shouke')
```

或者

```
from selenium.webdriver.common.by import By
```

```
element = driver.find_element(By.ID, 'shouke')
```

● By Class Name(标签属性)

实际应用中，经常会有相同的class name，所以class name查找多个元素会比较实用

示例：

```
<div class="shouke">
<span>Cheddar</span>
</div>
<div class="shouke">
<span>授客</span>
</div>
```

```
elements = driver.find_elements_by_class_name('shouke')
```

或

```
from selenium.webdriver.common.by import By
```

```
elements = driver.find_elements(By.CLASS_NAME, 'shouke')
```

- By Tag Name(标签名)

示例:

```
<iframe src="..."></iframe>
```

```
frame = driver.find_element_by_tag_name('iframe')
```

或者

```
from selenium.webdriver.common.by import By
element = driver.find_element(By.TAG_NAME, 'iframe')
```

- By Name(标签属性)

示例:

```
<input name="shouke" type="text"></input>
```

```
shouke = driver.find_element_by_tag_name('shouke')
```

或者

```
from selenium.webdriver.common.by import By
element = driver.find_element(By.NAME, 'shouke')
```

- By Link Text(元素内容)

一般情况, 某个页面上不会出现相同的文字链接, 通过文字链接来定位也是一种简单有效的定位方式

示例:

```
<a href="http://www.baidu.com/baidu?tn=monline_5_dg&ie=utf-8&wd=shouke">
search for shouke
</a>
```

```
driver.find_element_by_link_text('search for shouke')
```

或者

```
from selenium.webdriver.common.by import By
element = driver.find_element(By.LINK_TEXT, 'search for shouke')
```

示例: 模拟点击贴吧

coding= utf-8 #可加可不加, 防止中文乱码

```
from selenium import webdriver
if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.implicitly_wait(10)
    driver.get("http://www.baidu.com/")
    driver.find_element_by_link_text('贴吧').click()
    driver.quit()
```

- By Partial Link Text(元素内容)

通过部分链接定位, 即仅用链接的一部分文字进行匹配

示例:

```
<a href="http://www.baidu.com/baidu?tn=monline_5_dg&ie=utf-8&wd=shouke">
search for shouke
</a>
```

```
driver.find_element_by_partial_link_text('shouke')
```

或者

```
from selenium.webdriver.common.by import By
element = driver.find_element(By.LINK_TEXT, 'shouke')
```

示例: 模拟点击贴吧

```
# coding= utf-8 #可加可不加, 防止中文乱码
```

```
from selenium import webdriver
if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.implicitly_wait(5)
    driver.get('http://www.baidu.com/')
    driver.find_element_by_partial_link_text('贴').click()
    driver.quit()
```

● By CSS_Selector

示例:

```
<div id="shouke"><span class="dairy">test</span></div>
shouke = driver.find_element_by_css_selector('#shouke')
```

或者

```
from selenium.webdriver.common.by import By
element = driver.find_element(By.CSS_SELECTOR, '#shouke')
```

注意:

1. 不是所有浏览器都支持
2. css selector定位比xpath定位更速度, 更快

● By XPATH

对于不支持本地Xpath的浏览器, selenium也提供了自己的实现。但是XPath引擎有所不同, 如下表。

Driver	Tag and Attribute Name	Attribute Values	Native XPath Support
HtmlUnit Driver	Lower-cased	As they appear in the HTML	Yes
Internet Explorer Driver	Lower-cased	As they appear in the HTML	No
Firefox Driver	Case insensitive	As they appear in the HTML	Yes

示例:

```
<input type="text" name="授客" />
<INPUT type='text' name="shouke" />
inputs = driver.find_elements_by_xpath('//input')
或者
from selenium.webdriver.common.by import By
inputs = driver.find_elements(By.XPATH, '//input')
```

不同浏览器驱动下，匹配的input元素个数如下：

XPath expression	HtmlUnit Driver	Firefox Driver	Internet Explorer Driver
//input	1 ("example")	2	2
//INPUT	0	2	0

-----定位单个元素-----

示例1：对百度搜索输入框的进行定位

搜索框的部分html代码，如下

```
<input name="wd" class="" id="kw1" style="width: 521px;" type="text" maxLength="100"
```

```
#coding= utf-8 #可加可不加，防止中文乱码
```

```
from selenium import webdriver
```

```
import time
```

```
if __name__ == '__main__':
```

```
    driver = webdriver.Ie()
```

```
    driver.get('http://www.baidu.com/')
```

```
    driver.implicitly_wait(5)
```

```
    #通过标签属性id定位
```

```
    search_ipt = driver.find_element_by_id('kw1')
```

```
    search_ipt.send_keys(u'授客') #输入搜索内容
```

```
    time.sleep(1)
```

```
    search_ipt.clear() #清空百度搜索输入输入框
```

```
    time.sleep(1)
```

```
    #通过标签属性name定位
```

```
    search_ipt = driver.find_element_by_name('wd')
```

```
    search_ipt.send_keys(u'授客')
```

```
    time.sleep(1)
```

```
    search_ipt.clear()
```

```
    time.sleep(1)
```

```
    #通过标签名tag name定位
```

```
    search_ipt = driver.find_element_by_tag_name('input')
```

```
    search_ipt.send_keys(u'授客')
```

```
    time.sleep(1)
```

```
    search_ipt.clear()
```

```
    time.sleep(1)
```

```
    #通过css selector定位
```

```
    search_ipt = driver.find_element_by_css_selector('#kw1')
```

```
search_ipt.send_keys(u '授客')
time.sleep(1)
search_ipt.clear()
time.sleep(1)

#通过xpath定位
search_ipt = driver.find_element_by_xpath('//*[@id="kw1"]')
search_ipt.send_keys(u '授客')

#通过标签属性className方式定位(this.className)
search_btn = driver.find_element_by_id('su1').click()
time.sleep(3)
driver.quit()
```

注意:

```
search_ipt.send_keys(u '授客') #此处如果不加u, 会出现以下错误
UnicodeDecodeError: 'utf8' codec can't decode byte 0xe6 in position 0:
unexpected end of data
```

D:\workspace\PyCase\src\Py27\tttt.py

File "D:\Program Files (x86)\Python27\lib\json\encoder.py", line 270, in iterencode

return _iterencode(o, 0)

UnicodeDecodeError: 'utf8' codec can't decode byte 0xe6 in position 0: unexpected end of data

示例2: 对有道搜索输入框的进行定位

搜索输入框部分html代码如下

```
<input name="q" class="s-inpt" id="query" type="text" :
```

```
# coding= utf-8
```

```
from selenium import webdriver
```

```
import time
```

```
if __name__ == "__main__":
```

```
    driver = webdriver.Ie()
```

```
    driver.get("http://www.youdao.com/")
```

```
    driver.implicitly_wait(10)
```

```
    #通过标签属性class定位
```

```
    search_ipt = driver.find_element_by_class_name('s-inpt')
```

```
    search_ipt.send_keys(u '授客')
```

```
    driver.find_element_by_class_name('s-btn').click()
```

```
    time.sleep(3)
```

```
    driver.quit()
```

-----定位一组元素-----

`find_element_by_xx`方法可以定位单个元素，定位一组元素则要用`find_elements_by_xxx`方法（注意，该方法返回`list`列表对象）

定位一组对象一般用于以下场景：

- 批量操作对象，比如将页面上所有的checkbox都勾上
- 先获取一组对象，再在这组对象中过滤出需要具体定位的一些对象。比如，定位出页面上所有的checkbox，然后选择最后一个

注意：

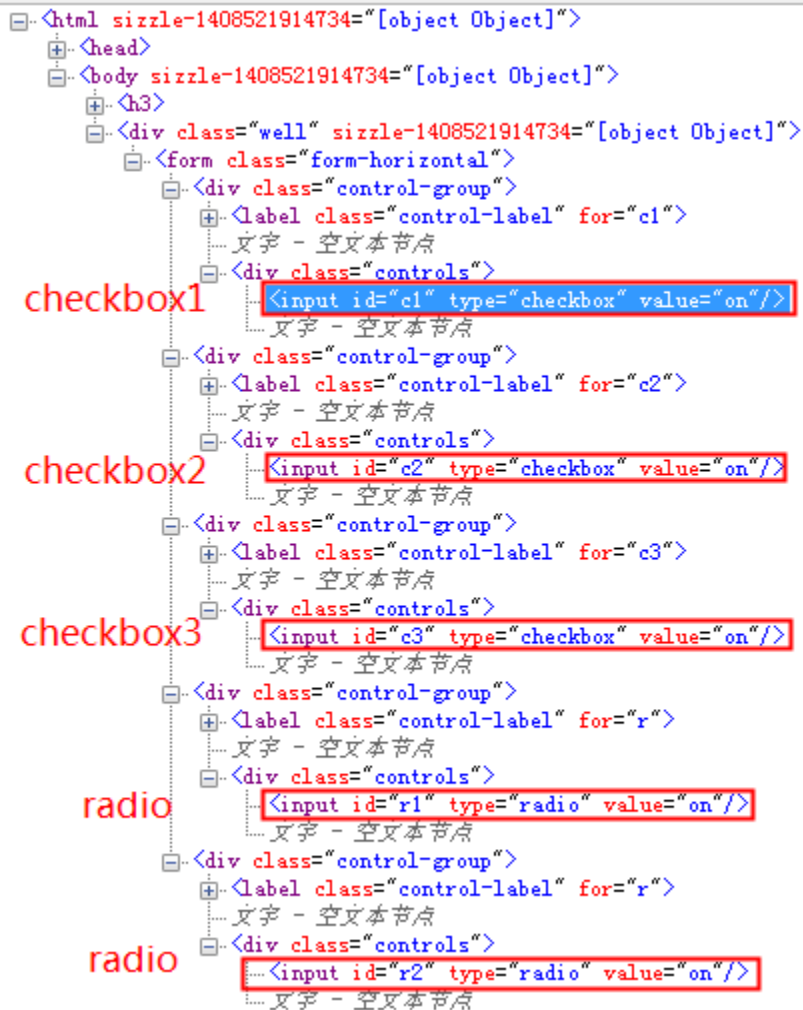
1.把“checkbox.html”放在同脚本放在同一工作目录下

2.ie下运行或报错，提示找不到元素，

解决方法：去掉“IE已限制此网页运行可以访问计算机的脚本或ActiveX控件”提醒

checkbox

checkbox1	<input type="checkbox"/>
checkbox2	<input type="checkbox"/>
checkbox3	<input type="checkbox"/>
radio	<input type="radio"/>
radio	<input type="radio"/>



示例1: 批量勾选页面上所有复选框

方法一

coding= utf-8

```
from selenium import webdriver
import time
import os

if __name__ == "__main__":
    driver = webdriver.Ie()

    file_path = os.path.abspath("checkbox.html")
    driver.get(file_path)
    driver.implicitly_wait(5) #

    # 选择所有的input元素
    inputs = driver.find_elements_by_tag_name("input")
```


#或者如下

```
#inputs = driver.find_elements_by_css_selector("input")
```

#或者如下

```
#inputs = driver.find_elements_by_xpath("//input")
```

#过滤出所有的checkbox并勾选之（属性type值为checkbox则为checkbox元素）

```
for input in inputs:
```

```
    #获取元素的某个属性值element.get_attribute('attribute_name')
```

```
    if input.get_attribute('type') == 'checkbox':
```

```
        input.click()
```

```
time.sleep(5)
```

```
driver.quit()
```

方法二

```
# coding= utf-8
```

```
from selenium import webdriver
```

```
import time
```

```
import os
```

```
if __name__ == "__main__":
```

```
    driver = webdriver.Ie()
```

```
    file_path = os.path.abspath("checkbox.html")
```

```
    driver.get(file_path)
```

```
    driver.implicitly_wait(10) #
```

#选择所有的checkbox元素

```
checkboxes =
```

```
driver.find_elements_by_css_selector('input[type="checkbox"]')
```

#或者如下

```
checkboxes = driver.find_elements_by_xpath('//input[@type="checkbox"]')
```

```
for checkbox in checkboxes:
```

```
    checkbox.click()
```

```
time.sleep(10)
```

打印当前页面有多少个checkbox

```
print(len(checkboxes))
```

```
driver.quit()
```

示例2: 去掉最后一个复选框

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import time
import os
```

```
if __name__ == "__main__":
    driver = webdriver.Ie()
```

```
    file_path = os.path.abspath("checkbox.html")
    driver.get(file_path)
    driver.implicitly_wait(10) #
```

```
    # 选择所有的input, 然后从中过滤出所有的checkbox并勾选之
    checkboxes =
driver.find_elements_by_css_selector('input[type=checkbox]')
```

```
    for checkbox in checkboxes:
        checkbox.click()
```

```
    time.sleep(5)
```

```
    # 去掉最后一个复选框
    checkboxes.pop(len(checkboxes) - 1).click()
```

```
    time.sleep(5)
```

```
    driver.quit()
```

➤ 获取元素属性

```
element.get_attribute(attribute_name)
```

示例:

```
# coding= utf-8
```

```
from selenium import webdriver
import time
```

```
if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.implicitly_wait(10)
    driver.get('http://ishouke.blog.sohu.com/')
```

```
blog_url = driver.find_element_by_id('blogUrl')

home_page = blog_url.find_element_by_link_text('首页') #定位“首页”连接元素

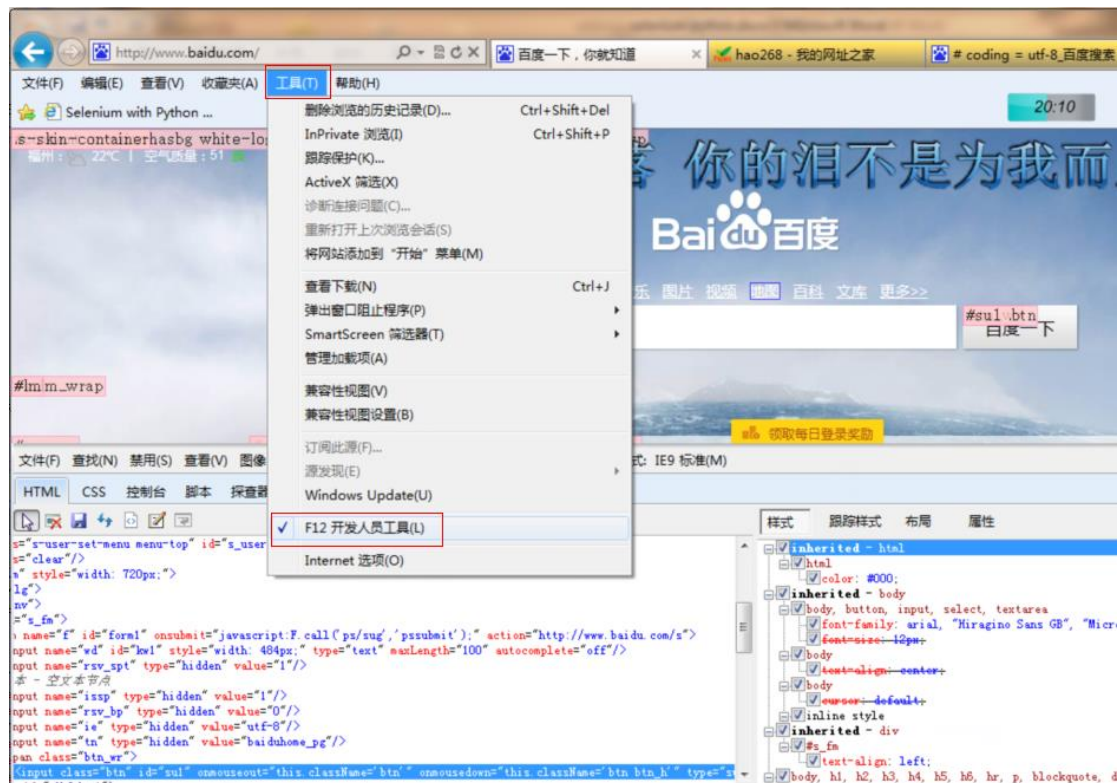
#打印连接href属性
print(home_page.get_attribute('href'))

time.sleep(5)
driver.quit()
```

附: 查看页面元素id, name等属性的方法

- ie浏览器

1. 如下, 打开浏览器, 打开要测试的网址, 点击工具->F12开发人员工具, 打开图示的调试器



2. 可以通过查看类和ID信息, 或者单击元素->属性, 查看控件的信息



- 火狐

安装debug插件, 右键查看元素

➤ 操作元素

需要引入ActionChains类(点击除外)

```
from selenium.webdriver.common.action_chains import ActionChains
```

● 鼠标操作

1. 鼠标点击元素

```
element.click()
```

2. 鼠标右键

```
chain = ActionChains(driver)
chain.context_click(element).perform()
```

示例:

```
# coding= utf-8
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
import time
if __name__ == "__main__":
    driver = webdriver.Ie()

    # 访问百度登录页面
    driver.get("http://www.baidu.com/")
    driver.implicitly_wait(5) # 智能等待5秒

    #鼠标右键搜索按钮
    chain = ActionChains(driver)
    rt = driver.find_element_by_id("su1")
    chain.context_click(rt).perform()
    time.sleep(5)

    driver.quit()
```

3. 鼠标双击

```
chain = ActionChains(driver)
chain.double_click(element).perform()
```

示例:

```
#coding= utf-8 #可加可不加, 防止中文乱码
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
import time
if __name__ == "__main__":
    driver = webdriver.Ie()
```

```
driver.get("http://www.baidu.com/")
driver.implicitly_wait(5) # 智能等待5秒

#鼠标双击搜索按钮
chain = ActionChains(driver)
rt = driver.find_element_by_id("su1")
chain.double_click(rt).perform()

time.sleep(5)
driver.quit()
```

4. 鼠标拖拽

```
action = ActionChains(driver)
action.drag_and_drop(source_element, target_element).perform()
```

示例:

```
# -*- coding: utf-8 -*-
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
import time

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    driver.get('http://www.ztree.me/v3/demo/cn/exedit/drag.html')

    #find source element with title attribute '随意拖拽 1-1'
    source = driver.find_element_by_id('treeDemo_2_a')

    #find target element with title attribute '随意拖拽1-2'
    target = driver.find_element_by_id('treeDemo_3_a')

    #construct class ActionChains object
    action = ActionChains(driver)

    #move source element to the position of target element
    action.drag_and_drop(source, target).perform()

    time.sleep(3)
    #change target element, try again
    target = driver.find_element_by_id('treeDemo_1_ul')
    action.drag_and_drop(source, target).perform()

    time.sleep(3)
```

```
driver.quit()
```

● 键盘操作

键盘操作需要引入Keys类(输入文字除外)

```
from selenium.webdriver.common.keys import Keys
```

✧ 输入文字

```
element.send_keys(text)
```

✧ 单按键操作

1. TAB键操作

```
element.send_keys(Keys.TAB)
```

示例

```
# coding= utf-8
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time

if __name__ == "__main__":
    driver = webdriver.Ie()
    # 访问115登录页面
    driver.get('http://www.115.com/')
    driver.implicitly_wait(5) # 智能等待5秒
    driver.maximize_window()

    account_ipt = driver.find_element_by_id('js-account')
    account_ipt.clear()
    account_ipt.send_keys('24636313') # 输入帐号

    time.sleep(2)
    #定位密码输入框
    passwd_ipt = driver.find_element_by_id('js-passwd')
    #模拟tab键,把光标定位到密码框,模拟光标一闪一闪的状态
    times = 10
    while(times >= 0):
        passwd_ipt.send_keys(Keys.TAB)
        time.sleep(1)
        times = times - 1

    time.sleep(3)
    driver.quit()
```

5. 回车键(Enter)操作

```
element.send_keys(Keys.ENTER)
```

示例: 模拟回车登录115网盘

```
# coding= utf-8
```

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys #需要引入Keys包
import time
if __name__ == "__main__":
    driver = webdriver.Ie()
    # 访问115登录页面
    driver.get("http://www.115.com/")
    driver.implicitly_wait(5) #智能等待5秒
    driver.maximize_window()

    account_ipt = driver.find_element_by_id('js-account')
    account_ipt.clear()
    account_ipt.send_keys('laiyuhenshuai@163.com') # 输入帐号

    time.sleep(1)
    #定位密码输入框
    passwd_ipt = driver.find_element_by_id('js-passwd')
    passwd_ipt.send_keys('huozhe') #输入密码

    #模拟回车方式登录(注意: 元素要选择对)
    passwd_ipt.send_keys(Keys.ENTER)

    time.sleep(3)
    driver.quit()
```

说明: 也可以通过定位登录按钮, 点击click, 代替定位密码框回车

6. 退格键(Backspace)操作

`element.send_keys(Keys.BACKSPACE)`

示例: 百度搜索框中输入内容, 然后退格键删除一个字符

```
# coding= utf-8
from selenium import webdriver
from selenium.webdriver.common.keys import Keys #需要引入Keys包
import time

if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.maximize_window()
    driver.get('http://www.baidu.com')
    driver.implicitly_wait(5)

    s_ipt = driver.find_element_by_id('kw1')
    #输入框输入内容
    s_ipt.send_keys(u'hello,授客')
```

```
time.sleep(1)

#退格删除1个字符
s_ipt.send_keys(Keys.BACKSPACE)

time.sleep(3)
driver.quit()
```

✧ 按键组合操作

1. 全选操作(Control+a)

```
element.send_keys(Keys.CONTROL, a)
```

2. 复制操作(Control+c)

```
element.send_keys(Keys.CONTROL, c)
```

3. 剪切操作(Control+x)

```
element.send_keys(Keys.CONTROL, x)
```

4. 黏贴操作

```
element.send_keys(Keys.CONTROL, v)
```

示例:

```
# coding= utf-8
from selenium import webdriver
from selenium.webdriver.common.keys import Keys #需要引入Keys包
import time

if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.maximize_window()
    driver.get('http://www.baidu.com')
    driver.implicitly_wait(5)

    s_ipt = driver.find_element_by_id('kw1')
    #输入框输入内容
    s_ipt.send_keys(u'授客')
    time.sleep(1)

    #全选输入框内容
    s_ipt.send_keys(Keys.CONTROL, 'a')
    time.sleep(1)

    #复制输入框内容
    s_ipt.send_keys(Keys.CONTROL, 'c')
```



```
time.sleep(1)

# 剪切输入的内容
s_ipt.send_keys(Keys.CONTROL, 'x')
time.sleep(1)

# 黏贴复制的内容
s_ipt.send_keys(Keys.CONTROL, 'v')

driver.find_element_by_id("su1").send_keys(Keys.ENTER)

time.sleep(3)
driver.quit()
```

- 清空对象内容

`element.clear()`

示例: 清空百度搜索输入框

coding= utf-8

```
from selenium import webdriver
```

```
import time
```

```
if __name__ == "__main__":
```

```
    driver = webdriver.Ie()
```

```
    # 访问百度首页
```

```
    driver.get('http://www.baidu.com')
```

```
    driver.implicitly_wait(5) # 智能等待5秒
```

```
    s_ipt = driver.find_element_by_id('kw1') # kw1为输入框id
```

```
    s_ipt.send_keys(u'授客') #输入搜索内容
```

```
    time.sleep(2)
```

```
    s_ipt.clear() #清空搜索内容
```

```
    time.sleep(3)
```

```
    driver.quit()
```

- 提交表单

示例:

```
<button type="button" id="submit">submit form</button>
```

```
element.submit()
```

注意: `submit()` 方法的使用前提是元素 `element` 必须在表单范围内, 即元素 `element` 必须是 `form` 元素的子元素, 否则会抛出异常 `NoSuchElementException`, 如下:

```
raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.NoSuchElementException: Message: u"Element was not in a form so couldn't submit"
```

注: 提交表单也可以用点击来实现

```
driver.find_element_by_id('submit').click()
```

表单样式如下

```
<form>
<button type="button" id="submit">submit form</button>
...
<element>...</element>
</form>
```

示例: 提交表单

```
# coding= utf-8
from selenium import webdriver
import time
if __name__ == "__main__":
    driver = webdriver.Ie()

    # 访问百度首页
    driver.get("http://www.baidu.com")
    driver.implicitly_wait(5) # 智能等待5秒
    driver.find_element_by_id('kw1').send_keys(u'授客') # kw1为输入框id
    time.sleep(2)

    # 通过submit()来操作
    driver.find_element_by_id('su1').submit() # su1为“百度一下”按钮id

    time.sleep(3)
    driver.quit()
```

● 处理select元素

如果采用Select类来处理, 需要引入类

```
from selenium.webdriver.support.ui import Select
```

1. 取消所有选项的选中状态

```
#构造Select类对象 "select"
select = Select(select_element)
#取消所有选项的选中状态
select.deselect_all()
```

2. 根据选项元素的text内容定位选项

```
select = Select(select_element)
select.select_by_visible_text('text_value')
```

3. 根据选项元素的index定位选项

```
select = Select(select_element)
```

```
select.select_by_index(item_index)
```

注意: index从0开始算起, 顺序为从上到下。

4. 根据选项元素的value属性值定位选项

```
select = Select(select_element)
```

```
select.select_by_value(value_attribute)
```

5. all_selected_options 怎么用??

示例1:

```
<html>

<head>
<meta http-equiv="Content-type" content="text/html; charset=gb2312" />
</head>

<body>
<select name="testselect" multiple="multiple" size=4>
  <option value="value1">Value 1</option>
  <option value="value2" selected>Value 2 test</option>
  <option value="value3">Value 3</option>
  <option value="value4">Value 4</option>
  <option value="value5">Value 4</option>
</select>

</body>
</html>
```



```
# -*- coding: utf-8 -*-
from selenium import webdriver
import time
import os

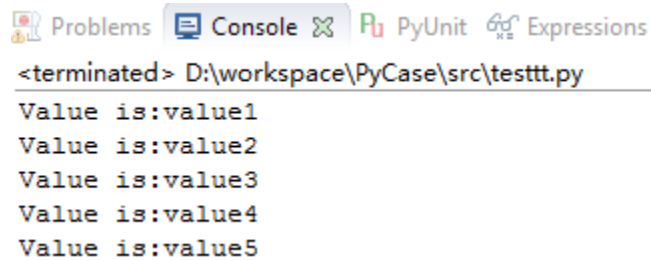
if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    url = os.path.abspath('User Input-Filling In Forms.html')
    driver.get(url)
    driver.implicitly_wait(5)

    #find all elements with tag name "value"
    all_options = driver.find_elements_by_tag_name('input')
```

```
#cycle through each options in turn,print their value and select each
for option in all_options:
    print("Value is:" + option.get_attribute("value"))
    option.click()

time.sleep(2)
driver.quit()
```

运行后,页面上依次点选所有复选框(模拟安装ctrl键+鼠标点击操作),而控制台输出如下



```
<terminated> D:\workspace\PyCase\src\testtt.py
Value is:value1
Value is:value2
Value is:value3
Value is:value4
Value is:value5
```

更有效率的方法

```
# -*- coding: utf-8 -*-

#available since 2.12
from selenium import webdriver
from selenium.webdriver.support.ui import Select
import time
import os

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    url = os.path.abspath('User Input-Filling In Forms.html')
    driver.get(url)
    driver.implicitly_wait(5)

    #construct Select Class Object "select"
    select = Select(driver.find_element_by_tag_name('select'))

    #clear all selected option
    select.deselect_all()

    #select the element with text value displayed "Value1"
    select.select_by_visible_text('Value 1')

    time.sleep(2)
```

```
driver.quit()
```

注意:

1. `deselect_all()` 函数仅适用于 `<select multiple="multiple" ...>` 这种带 `multiple` 属性的 `select` 元素
2. `User Input-Filling In Forms.html` 文件要放在工程文件 `src` 目录下

示例2

未点击时

USPS First Class ==> \$3.20 ▼

点击时，弹出下拉选框

USPS First Class ==> \$3.20 ▼

- UPS Next Day Air ==> \$12.51
- UPS Next Day Air Saver ==> \$11.61
- UPS 3 Day Select ==> \$10.69
- UPS 2nd Day Air ==> \$9.03
- UPS Ground ==> \$8.34
- USPS Priority Mail Insured ==> \$9.25
- USPS Priority Mail ==> \$7.45
- USPS First Class ==> \$3.20

```
drop_down.html
<html>
<body>
  <select id="ShippingMethod"
  onchange="updateShipping(options[selectedIndex]);" name="ShippingMethod">
    <option value="12.51">UPS Next Day Air ==> $12.51</option>
    <option value="11.61">UPS Next Day Air Saver ==> $11.61</option>
    <option value="10.69">UPS 3 Day Select ==> $10.69</option>
    <option value="9.03">UPS 2nd Day Air ==> $9.03</option>
    <option value="8.34">UPS Ground ==> $8.34</option>
    <option value="9.25">USPS Priority Mail Insured ==> $9.25</option>
    <option value="7.45">USPS Priority Mail ==> $7.45</option>
    <option value="3.20" selected="">USPS First Class ==> $3.20</option>
  </select>
</body>
</html>
```

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import time,os
```

```
if __name__ == "__main__":
```

```
    driver = webdriver.Ie()
```

```
    file_path = os.path.abspath('drop_down.html')
```

```
    driver.get(file_path)
```

```
    driver.implicitly_wait(5) #
```

```
#先定位到下拉框
down_list = driver.find_element_by_id("ShippingMethod")

# 再点击下拉框下的选项
down_list.find_element_by_xpath("//option[@value='10.69']").click()

time.sleep(5)
driver.quit()

或者也可以如下,
# -*- coding: utf-8 -*-
from selenium import webdriver
from selenium.webdriver.support.ui import Select
import time,os

if __name__ == "__main__":
    driver = webdriver.Ie()

    file_path = os.path.abspath('drop_down.html')
    driver.get(file_path)
    driver.implicitly_wait(5)

    down_list = Select(driver.find_element_by_tag_name('select'))

    #选择value属性的属性值为10.69的下拉选项
    down_list.select_by_value('10.69')

    #选择下拉选框中, 从上往下, 第二个下拉选项
    down_list.select_by_index(1)

    time.sleep(3)
    driver.quit()
```

注意: 记得把drop_down.html放在工程src目录下

● 处理弹出对话框

包含alert(弹出告警对话框-对话框中有一个OK按钮),
confirm(确认对话框--对话框中包含一个OK按钮与Cancel按钮),
和prompt(等待输入弹出对话框--对话框中包含一个OK按钮、Cancel按钮与一个文本输入框)

1. 弹出alert对话框“框确”认操作

```
alert = driver.switch_to.alert
alert.accept()
```

示例: 接受告警

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-type" content="html/text; charset=utf-8">
<script>
var txt="";
function message()
{
alert("hi,授客,你好")
}
</script>
</head>

<body>
<input type="button" value="查看消息" onclick="message()" />
</body>

</html>

# -*- coding: utf-8 -*-
from selenium import webdriver
import time
import os

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    url = os.path.abspath('alert.html')
    driver.get(url)

    #click button with text:'查看消息'
    driver.find_element_by_tag_name('input').click()

    time.sleep(2)

    #capture the alter dialog
    alter = driver.switch_to.alert

    #accept alter
    alter.accept()

    time.sleep(3)
    driver.quit()
```

注意: 记得把alert.html放在工程文件夹src目录下

2. 弹出confirm对话框“取消”操作

```
alert = driver.switch_to.alert
alert.dismiss()
```

示例：确认confirm对话框，忽视提醒对话框

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-type" content="html/text; charset=utf-8">
<script>
function message()
{
    if(confirm("确定要离开授客么？呜呜"))
    {
        alert("Bye, my friend.");
    }
    else
    {
        alert("嘿嘿，你不走了呀!");
    }
}
</script>
</head>

<body>
<input type="button" value="退出" onclick="message()" />
</body>
</html>
```

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
```

```
import time
```

```
import os
```

```
if __name__ == "__main__":
```

```
    driver = webdriver.Firefox()
```

```
    driver.maximize_window()
```

```
    url = os.path.abspath('confirm.html')
```

```
    driver.get(url)
```

```
    #click button with text:'查看消息'
```

```
    driver.find_element_by_tag_name('input').click()
```

```
    time.sleep(2)
```

```
    #capture the confirm dialog
```

```
    alter = driver.switch_to.alert
```

```
    #confirm
```

```
    alter.accept()
```



```
alter = driver.switch_to.alert
#ignore second alter
time.sleep(1)
alter.dismiss()

time.sleep(3)
driver.quit()
```

注意: 记得把confirm.html放在工程文件夹src目录下

3. 弹出prompt对话框“确认”操作

```
alert = driver.switch_to.alert
alert.accept()
```

示例: 确认等待输入框

```
<!DOCTYPE html>
<html>
<head>
<script>
var txt="";
function message()
{
var sResult=prompt("请在下面输入作者名", "授客");
if(sResult!=null)
{
    alert("恭喜, 输入正确");
}
else
{
    alert("输入错误");
}
}
</script>
</head>

<body>
<input type="button" value="查看消息" onclick="message()" />
</body>
</html>
```

```
# -*- coding: utf-8 -*-
from selenium import webdriver
import time
import os

if __name__ == "__main__":
    driver = webdriver.Firefox()
```

```
driver.maximize_window()
url = os.path.abspath('prompt.html')
driver.get(url)

#click button with text:'查看消息'
driver.find_element_by_tag_name('input').click()

time.sleep(2)
#capture the confirm dialog
alter = driver.switch_to.alert

#select ok
alter.accept()

time.sleep(3)
driver.quit()
```

注意: 记得把prompt.html放在工程文件夹src目录下

4. 获取弹出对话框文字信息

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-type" content="html/text; charset=utf-8">
<script>
var txt="";
function message()
{
alert("hi,授客, 你好")
}
</script>
</head>

<body>
<input type="button" value="查看消息" onclick="message()" />
</body>

</html>

# -*- coding: utf-8 -*-
from selenium import webdriver
import time
import os

if __name__ == "__main__":
    driver = webdriver.Firefox()
```

```
driver.maximize_window()
url = os.path.abspath('alert.html')
driver.get(url)

#click button with text:'查看消息'
driver.find_element_by_tag_name('input').click()

time.sleep(2)

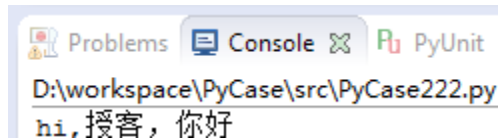
#capture the alter dialog
alert = driver.switch_to.alert

#print text on alert object
print(alert.text)

#accept alter
alert.accept()

time.sleep(3)
driver.quit()
```

控制台输出结果:



注意: 记得把alert.html放在工程文件夹src目录下

- 处理下拉菜单

思路: 采用分层处理的方法

1. 鼠标移动到界面元素上方, 自动弹出下拉菜单

示例1: 鼠标移动到Tutorials上方的时候, 会自动弹出下拉菜单, 模拟该过程并点击HTML / CSS



通过火狐可查看对应的页面元素对应的html元素，如下，白色圈圈即下拉面板对应的元素就是ul



```
<html>
  <head></head>
  <body>
    <div class="example">
      <ul id="nav">
        <li class="current"></li>
        <li>
          <a href="http://www.script-tutorials.com/"></a>
          <ul>
            <li>
              <a href="http://www.script-tutorials.com/category/html-css/">
                HTML / CSS
              </a>
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </body>
</html>
```

```
# -*- coding: utf-8 -*-
from selenium import webdriver
import time, os

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    driver.get("http://www.script-tutorials.com/demos/87/index.html")
    driver.implicitly_wait(10)

    #注意: 针对这种鼠标移动到页面, 自动弹出菜单的, 用css或xpath, id等直接定位, 是
    #无法一次到位的

    #css
    #m=driver.find_element_by_css_selector('a[href="http://www.script-tutor
    ials.com/category/html-css/"]').click() # 注意: 会报错
    #xpath

    #driver.find_element_by_xpath('//a[@href="http://www.script-tutorials.c
    om/category/html-css/"]').click()# 注意: 会报错

    #定位下拉菜单面板(注意: 和那个Tutorials无关, 所以定位的是不是Tutorials)
    ul = driver.find_element_by_id('nav')

    #定位到 html-css菜单项所在的小面板
    li=ul.find_element_by_css_selector('li:nth-child(2)')

    #定位到html-css菜单项并点击
    a=li.find_element_by_css_selector('li>a').click()

    time.sleep(3)
    driver.quit()
```

测试页面: <http://www.script-tutorials.com/demos/87/index.html>

示例2: 鼠标移动到“拼搏”，“头像“，”向下箭头“，都会自动弹出下拉菜单，现在模拟该操作并点击退出登录



同上列一样，我们先找存放下拉菜单的面板元素



对应的html元素代码如下

```
<a rel="user-nav" href="javascript:;" hide_status="1">
  <i tab_btn="my_set"></i>
  <span rel="user_name">
    拼搏
  </span>
  <s class="fs-icon fsi-arrow"></s>
</a>
```

接着找“退出登录”元素，如下，

```
<div class="user-popup-menu" style="display: none;" rel="user-pop-menu">
  <s class="arrow"></s>
  <i class="arrow"></i>
  <div class="cell"></div>
  <div class="cell"></div>
  <div class="cell">
    <a id="js-quit-sys" href="http://passport.115.com/?ac=logout&goto=http%3A%2F%2F115.com">
      退出登录
    </a>
  </div>
</div>
```

和上例不同的地方在哪里？“菜单项”和“菜单面板”元素并不是“父子”关系，接着按照示例1的方法进行了实验，结果发现，模拟鼠标移动到元素 `'span[rel="user_nav"]'`，并没弹出下拉菜单面板，怎么办？见下面，移动到面板所在子元素 比如“拼搏”，这下就好了

#coding=utf-8

```
from selenium import webdriver
```

```
from selenium.webdriver.common.action_chains import ActionChains
```

```
import time
```

#主程序

```
if __name__ == '__main__':
    driver = webdriver.Firefox()
    driver.maximize_window()

    url = 'http://www.115.com'
    driver.get(url)

    driver.implicitly_wait(20)

    #####115帐号登录 115网盘-不勾选5天内免登陆#####
    account_input = driver.find_element_by_id('js-account')
    account_input.clear()

    account_input.send_keys('24636313')
    passwd_input = driver.find_element_by_id('js-passwd')

    passwd_input.clear()

    passwd_input.send_keys('1017fenxiang')

    driver.find_element_by_id('js-submit').click()

    #####退出115#####
    time.sleep(5)

    chain = ActionChains(driver)


    #鼠标移动到"下拉菜单面板", 目的在于弹出下拉菜单面板
    #span = driver.find_element_by_css_selector('span[rel="user_nav"]') #注意:
    #这里不起作用


    #鼠标移动到面板下的子元素: 拼搏, 目的在于弹出下拉菜单面板
    span = driver.find_element_by_css_selector('span[rel="user_name"]')
    chain.move_to_element(span).perform()

    #点击 退出登录 (弹出面板后就以直接以id等方式查找了)
    driver.find_element_by_id('js-quit-sys').click()

    time.sleep(5)
    # 退出浏览器驱动并关闭浏览器窗口
    driver.quit()
```


示例3: 115网盘的注册页面, 鼠标移动到+86会自动弹出下拉列表, 模拟该操作并点击香港

 手机注册

 邮箱注册

手机号码	 +86 ▾	填写手机号
设置密码	 中国 +86  香港 +852  其他地区 +00	
验证码	输入验证码	免费获取验证码

h5

 +86 ▾

填写:

```
<h5 rel="title">
  <i class="ias-china"></i>
  <span>
    +86
  </span>
</h5>

<div class="area-list" rel="list" style="display: none;">
  <a data-btn="china" href="javascript:;"></a>
  <a data-btn="hk" href="javascript:;">
    <i class="ias-hongkong"></i>
    <span>
      香港
    </span>
  </a>
  <a data-btn="other" href="javascript:;"></a>
</div>
</div>
```

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
import time
```

```
if __name__ == "__main__":
    driver = webdriver.Firefox()
```



```
driver.maximize_window()
driver.get('http://www.115.com')
current_handle = driver.current_window_handle
driver.find_element_by_link_text('免费注册').click()

handles = driver.window_handles
for handle in handles:
    if current_handle != handle:
        driver.switch_to_window(handle)

chain = ActionChains(driver)
#定位地区下拉列表
area_list = driver.find_element_by_css_selector('h5[rel="title"]')
chain.move_to_element(area_list).perform()
driver.implicitly_wait(5) #注意: 如注释该语句, 下面的执行经常会出现找不到
元素的情况, 换成time.sleep(5) 也不行

#点击香港下拉菜单项
driver.find_element_by_css_selector('a[data-btn="hk"]').click()
driver.close() #关闭注册页面

driver.switch_to_window(current_handle) #切换当前页面为原来的页面
time.sleep(3)
driver.quit()
```

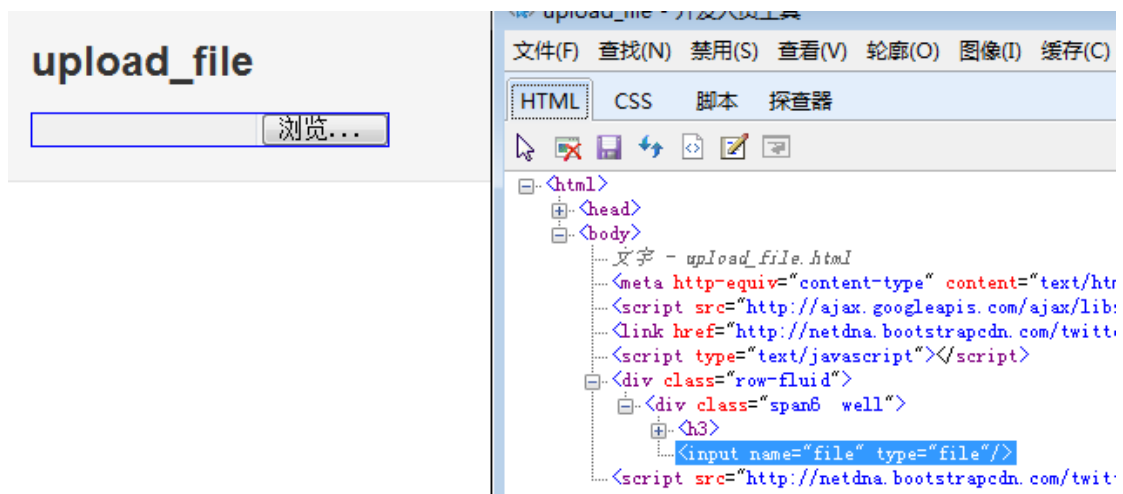
说明: 注意这类情况

- 处理文件上传

示例: 模拟文件上传

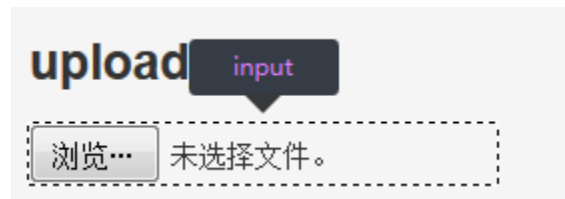
思路: 定位上传按钮, 通过send_keys添加本地文件路径就可以了

ie下:



火狐下:

upload_file.html



```
<div class="row-fluid">
  <div class="span6 well">
    <h3></h3>
    <input type="file" name="file"></input>
  </div>
</div>
```

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import time,os
```

```
if __name__ == "__main__":
    driver = webdriver.Ie()
    file_path = os.path.abspath('upload_file.html')
    driver.get(file_path)

    driver.implicitly_wait(20)

    # 定位上传按钮，添加本地文件即file_path.
    driver.find_element_by_name('file').send_keys('d:\\test.txt')

    time.sleep(5)

    driver.quit()
```

注意: 这里没做判断,要是文件不存在,会很糟糕,一直弹出提示。。。

代码改进:

```
# coding= utf-8
from selenium import webdriver
import os
```

```
import time
from exceptions import Exception

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    url = os.path.abspath('upload_file.html')
    driver.get(url)

    driver.implicitly_wait(20)

    keys = 'd:\\test.txt'

    #判断keys是否为目录
    if True == os.path.isdir(file_path): #如果为目录
        raise Exception("ee")

    #判断文件是否存在
    if False == os.path.exists(file_path): #如果文件不存在
        raise Exception("ee")
    else:
        driver.find_element_by_name("file").send_keys(file_path)

    time.sleep(4)
    driver.quit()
```

注意: 记得把"upload_file.html"文件放在src目录下

● 使用javascript处理元素

可执行任何javascript来查找元素, 并且只要你返回的是一个DOM元素, 它将被自动转换为WebElement对象

函数说明

excute_script(script, *args)

函数功能: 在当前窗口/框架, 同步执行javascript

函数参数:

script: 要执行的javascript

*args: 参数, 适用任何JavaScript脚本

例如 (基于某个包含jQuery的网页):

```
element = driver.execute_script("return $('shouke')[0]")
```

js的执行分两种情况:

- ✧ 在页面上直接执行js
- ✧ 在某个已经定位的元素上执行js

✧ 在页面上直接执行js

1、操作浏览器滚动条

场景一：注册时的法律条文需要阅读，判断用户是否阅读完的标准是：滚动条是否拉动到最底下

场景二：要操作的页面元素不再视线范围，无法操作，需要拖动滚动条

用于标识滚动条位置的代码

```
<body onload="document.body.scrollTop=0 ">
```

```
<body onload="document.body.scrollTop=100000 ">
```

说明：

如果滚动条在最上方的话，scrollTop=0,要想使用滚动条在最下方，可以设置scrollTop=100000

```
js = "var q=document.getElementById('id').scrollTop = 10000"  
driver.execute_script(js)
```

示例：

以操作百度搜索结果页为例（在页面上执行js）

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
```

```
import time
```

```
if __name__ == "__main__":
```

```
    driver = webdriver.Ie()
```

```
    driver.get('http://www.baidu.com')
```

```
    driver.implicitly_wait(20)
```

```
    #搜索
```

```
    driver.find_element_by_id('kw1').send_keys(u'授客')
```

```
    driver.find_element_by_id('su1').click()
```

```
    time.sleep(2)
```

```
    #将页面滚动条拖动到底部
```

```
    js = 'document.documentElement.scrollTop=10000'
```

```
    driver.execute_script(js)
```

```
    time.sleep(2)
```

```
    #滚动条移动到页面的顶部
```

```
    js = 'var q=document.documentElement.scrollTop=0'
```

```
    driver.execute_script(js)
```

```
    driver.quit()
```

2、在打开页面中弹窗

```
# coding= utf-8
from selenium import webdriver
import os
import time

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()

    url = os.path.abspath('testjs.html')
    driver.get(url)
    driver.implicitly_wait(10)

    js = 'alert("我是警告框!!")'
    driver.execute_script(js)

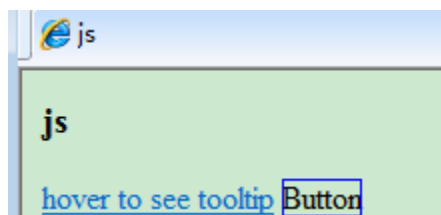
    time.sleep(1)
    driver.switch_to_alert().accept()

    time.sleep(3)
    driver.quit()
```

testjs.html仅是个简单的页面，放在src目录下。

✧ 在已经定位的元素上执行js

示例1: 隐藏页面元素Button



```
<html>
<head>
<body>
  <h3>
    <div class="row-fluid">
      <div class="span8 well">
        <a title="selenium-webdriver(python)" id="tooltip" href="#" data-toggle="tooltip">
          文字 - 空文本节点
          <a class="btn">
            文字 - Button
        </script>
</script>
```

```
# -*- coding: utf-8 -*-
from selenium import webdriver
import time,os
```

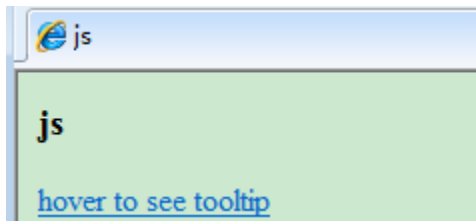
```
if __name__ == "__main__":
    driver = webdriver.Ie()
    file_path = os.path.abspath("js.html")
    driver.get(file_path)
    driver.implicitly_wait(20) #

    # 通过js隐藏选中的元素 方法一
    # driver.execute_script('$("#tooltip").fadeOut();')

    # 第二种方法
    button = driver.find_element_by_class_name('btn')
    driver.execute_script('$ (arguments[0]).fadeOut()', button)
    time.sleep(5)

    driver.quit()
```

执行结果:



说明:

arguments对象，它是调用对象的一个特殊属性，用来引用**Arguments**对象。

fadeOut() 方法使用淡出效果来隐藏被选元素，假如该元素是隐藏的

注意：记得把js.html放在工程文件src目录下。

- 窗口(Window)或框架(Frame)切换

✧ 切换窗口

示例:

```
<a href="somewhere.html" target="windowName">Click here to open a new window</a>
```

```
driver.switch_to.window("windowName")
```

说明：可通过打开窗口的javascript或超链接查看窗口名字，如上

或者

```
for handle in driver.window_handles:
    driver.switch_to.window(handle)
```

示例：切换窗口

```
# -*- coding: utf-8 -*-
from selenium import webdriver
import time
import os
```

```
if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    url = os.path.abspath('window1.html')
    driver.get(url)
    driver.implicitly_wait(5)

    #open new window
    driver.find_element_by_partial_link_text('shouke').click()

    time.sleep(1)

    #switch to new window
    driver.switch_to.window('shouke')

    driver.find_element_by_partial_link_text('shouke').click()

    #switch to new window:授客的博客
    driver.switch_to.window('shouke_blog')
    time.sleep(5)

    #return to the old window
    driver.switch_to.window('shouke')

    time.sleep(5)
    driver.quit()
```

注意:

- 1.记得把window1.html,window2.html放到工程文件src目录下
- 2.注意如果打开新窗口后,如不用driver.switch_to.window(windowName)方法,在新页面查找元素,会出现找不到元素的情况

示例: 切换窗口

```
<html>

<head>
<meta http-equiv="Content-type" content="text/html; charset=gb2312" />
</head>

<p>
<a href="http://ishouke.blog.sohu.com/" target="_blank">shouke's blog</a>
</p>
</body>
</html>

# -*- coding: utf-8 -*-
from selenium import webdriver
```

```
import time
import os

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    url = os.path.abspath('blank_link.html')
    driver.get(url)

    #get current window handle
    current_handle = driver.current_window_handle

    driver.find_element_by_tag_name('a').click()
    time.sleep(5)

    #get all the window handles
    handles = driver.window_handles

    #iterate over every open window
    for handle in handles:
        if handle != current_handle: #compare handle with current_handle
            #to find new window
            driver.switch_to_window(current_handle) # switch to old
            window

            time.sleep(5)
            driver.quit()
```

注意:

1. 记得把blank_link.html方到当前工程文件夹src目录下。
2. 通常当通过某个带属性target="_blank"的超链接打开新窗口时会用到这个

示例2: 切换窗口

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import time,os

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()

    driver.get('http://www.115.com')
    driver.implicitly_wait(5)
```



```
#打开注册页面
driver.find_element_by_link_text('免费注册').click()

#注册页面中查找元素
driver.find_element_by_id('js-area_box')

time.sleep(3)
driver.quit()
```

结果: 提示找不到元素, 但是明明有呀



解决方案:

```
# -*- coding: utf-8 -*-

from selenium import webdriver
import time,os

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    driver.get('http://www.115.com')
    driver.implicitly_wait(5)

    #获取当前窗口句柄
    current_handle = driver.current_window_handle

    #打开注册页面
    driver.find_element_by_link_text('免费注册').click()

    #捕获所有窗口的句柄
    handles = driver.window_handles

    #查找新窗口(免费注册页面所在窗口)句柄, 因为仅打开一个窗口, 所以如果和原窗口句柄
    #不一样, 那肯定是新窗口句柄
    for handle in handles:
        if handle != current_handle:
            driver.switch_to_window(handle)

    #注册页面中查找元素
    driver.find_element_by_id('js-area_box')

    #返回原来的窗口(http://www.115.com页面所在窗口)
    driver.switch_to_window(current_handle)

    #输入帐号
    driver.find_element_by_id('js-account').send_keys('test')

    time.sleep(3)
    driver.quit()
```

✧ 切换Frame

```
driver.switch_to_frame(frame)
```

示例:

frame



```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import time
import os

if __name__ == "__main__":
    driver = webdriver.Ie()

    file_path = os.path.abspath("frame.html")
    driver.get(file_path)
    driver.implicitly_wait(20) #

    # first, find iframe1 (id = f1)
    driver.switch_to_frame("f1")

    # the found it's sub franem:iframe2 (id = f2)
    driver.switch_to_frame("f2")

    #then you can operate elements in frame2 normally
    driver.find_element_by_id("kw1").send_keys("selenium")
    driver.find_element_by_id("su1").click()
```

```
time.sleep(5)
driver.quit()
```

注意: 记得把frame.html和inner.html放在工程文件夹src目录下

➤ 添加等待时间

● 显示等待 (Explicit Waits, 功能效果等同Implicit Waits)

规定时间内, 只要等待多久就只等待多久

```
# coding= utf-8
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait from
selenium.webdriver.support import expected_conditions as EC
if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.get('http://ishouke.blog.sohu.com/')
    driver.maximize_window()
    try:
        element = WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.ID, 'statusImg')))
        print('found element')
    finally:
        driver.quit()
```

说明:

```
WebDriverWait(driver, 10)
```

等待达到10秒时, 如果还找不到指定元素, 则抛出一个TimeoutException异常,

```
raise TimeoutException(message)
selenium.common.exceptions.TimeoutException: Message: ''
```

如果等待时间未达到10秒, 那么找到元素后停止等待, 返回找到的元素

until(EC.presence_of_element_located((By.CSS_SELECTOR, css_selector)))
给定预期条件, until(Expected Conditions) 见名知意, 指定时间范围内等待, 直到预期条件满足,

例: Expected Conditions-等待元素(登录按钮>id: js-submit)出现并且可点击(enabled)

```
try:
    wait = WebDriverWait(driver, 10)
    wait.until(EC.element_to_be_clickable((By.ID), 'js-submit')))
    print('found element')
finally:
    driver.quit()
```

● 隐式等待 (Implicit Waits)

implicit wait为了告诉webdriver, 如果元素不能立即获取的情况下, 在一定的时间范围内

轮询DOM.缺省设置为0,一旦设置,implicit wait设置对WebDriver对象实例起作用。隐式等待一个元素被发现或一个命令完成,这个方法每次会话只需要调用一次

```
# coding= utf-8
from selenium import webdriver
if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.get('http://ishouke.blog.sohu.com/')
    driver.maximize_window()

    driver.implicitly_wait(10) # 智能等待10秒
    driver.find_element_by_id('statusImg')
    print('found element')

    driver.quit()
```

- 进程休眠

引入time模块,然后自由添加时间

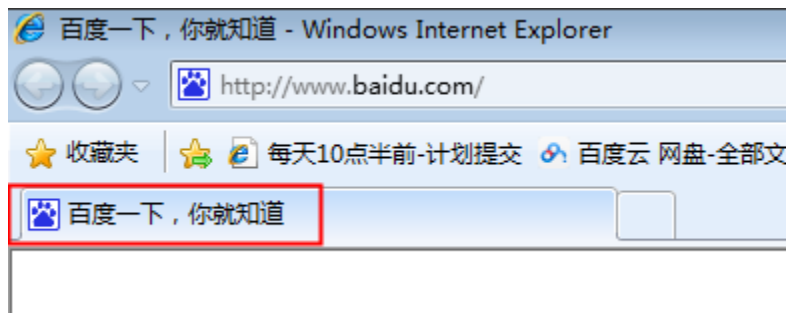
```
# coding= utf-8 #可加可不加,防止中文乱码
from selenium import webdriver
import time #引入time模块
if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.get('http://www.baidu.com/')
    time.sleep(0.3) #休眠0.3秒
    driver.find_element_by_id('kw1').send_keys('hello selenium')
    driver.find_element_by_id('su1').click()
    time.sleep(20) #休眠20秒
    driver.quit()
```

- 获取driver属性

- 获取html页面title

driver.title

示例: 打印页面title



```
# coding= utf-8
from selenium import webdriver
```

```
import time

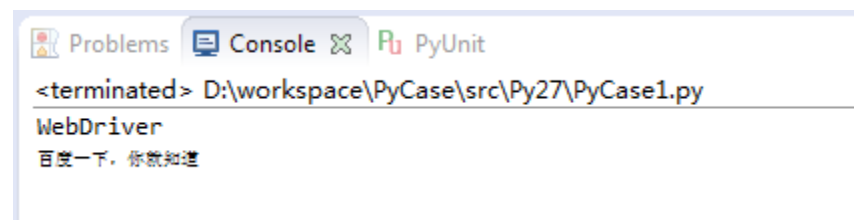
if __name__ == "__main__":
    driver = webdriver.Ie()

    print(driver.title)
    driver.get('http://www.baidu.com/')
    driver.implicitly_wait(5)
    print(driver.title)
    driver.find_element_by_partial_link_text('知').click()

    time.sleep(5)

    driver.quit()
```

以下是控制台输出(注意获取url前后标题的变化)



- 获取当前页面url

`driver.current_url`

示例:

```
# coding= utf-8
from selenium import webdriver
import time

if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.get('http://www.baidu.com')
    driver.implicitly_wait(10)
    current_web_site = driver.current_url #打印当前driver所在浏览器窗口中的
    网址
    print(current_web_site)
    driver.find_element_by_id('kw1').send_keys(u'授客')
    driver.find_element_by_id('su1').click()


    current_web_site = driver.current_url
    print('after search: '+current_web_site)

    time.sleep(5)
```

```
driver.back()

current_web_site = driver.current_url
print('after back: '+current_web_site)
time.sleep(5)
driver.quit()
```

控制台输出



```
Problems Console PyUnit Expressions
<terminated> D:\workspace\PyCase\src\Py27\tttt.py
http://www.baidu.com/
after search: http://www.baidu.com/#wd=%E6%8E%88%E5%AE%A2&rsv_spt=1&issp=1&rsv_bp=0
after back: http://www.baidu.com/
```

备注: 打印当前页面的url, 也可以如下

```
# coding= utf-8
from selenium import webdriver
if __name__ == "__main__":
    driver = webdriver.Ie()
    url = 'http://www.baidu.com'
    driver.implicitly_wait(5)
    print "now access %s" %(url)
    driver.get(url)
    driver.find_element_by_id('kw1').send_keys(u'授客')
    driver.find_element_by_id('su1').click()
    driver.quit()
```

➤ 浏览器设置

- 浏览器最大化

调用启动的浏览器不是全屏的, 这样不会影响脚本的执行, 但是有时候会影响我们“观看”脚本的执行

```
# coding= utf-8
from selenium import webdriver
import time
if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.get("http://www.baidu.com")
    driver.implicitly_wait(5) # 智能等待5秒

    print "浏览器最大化"
    driver.maximize_window() # 将浏览器最大化显示
    time.sleep(2)
    driver.find_element_by_id('kw1').send_keys(u'授客')
    driver.find_element_by_id('su1').click()
```

```
time.sleep(3)
driver.quit()
```

说明: 可以先把浏览器调整为非全屏, 然后关闭, 再次打开就不是全屏的了, 这样才可以看到效果

- 设置浏览器的高、宽

```
# coding= utf-8
from selenium import webdriver
import time
if __name__ == "__main__":
    driver = webdriver.Ie()
    driver.maximize_window() #最大化窗口
    driver.get('http://ishouke.blog.sohu.com/')
    driver.implicitly_wait(5) #智能等待5秒

    #参数值以像素为单位
    print "设置浏览器宽480像素、高800像素显示"
    driver.set_window_size(400, 800)
    time.sleep(3)
    driver.quit()
```

- 导航: 历史位置

- 前进

```
driver.forward()
```

- 后退

```
driver.back()
```

示例:

```
# coding= utf-8
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait #available since
2.4.0
from selenium.webdriver.support import expected_conditions as EC
#available since 2.26.0

import time
if __name__ == "__main__":
    #Create a new instance of the Firefox driver
    driver = webdriver.Firefox()

    #max window
    driver.maximize_window()
```



```
#go to the baidu home page
driver.get('http://www.baidu.com')

#wait for three seconds
time.sleep(2)

#goto the news page
driver.get('http://news.baidu.com')
time.sleep(2)

#back to baidu home page
driver.back()
time.sleep(2)

#forward to news page
driver.forward()
time.sleep(2)

driver.back()
time.sleep(2)

driver.quit()
```

注意: 前进和后退操作是针对同一个浏览器中的同一个窗口而言的, 如果你打开的两个页面, 每个页面各占用一个窗口, 那么不能直接使用driver.back() 和driver.forward() 函数

➤ Cookie

示例: 添加、获取、删除cookie

```
# -*- coding: utf-8 -*-
from selenium import webdriver
import time

if __name__ == "__main__":
    driver = webdriver.Firefox()
    driver.maximize_window()
    driver.get('http://www.example.com')

    #Now set the cookie, Here's one for the entire domain
    #the cookie name here is 'key' and it's value is 'value'
    driver.add_cookie({'name': 'key', 'value': 'value', 'path': '/'})
    #additional keys that can be passed in here:
    #'domain' -> String,
    #'secure' -> Boolean,
    #'expiry' -> Milliseconds since the Epoch it should expire
```

```
#Output all the available cookies for the current URL
for cookie in driver.get_cookies():
    print('%s - > %s' % (cookie['name'], cookie['value']))

#get cookie by name and print it's all information
cookie = driver.get_cookie('key')
print(cookie)

#delete cookies in 2 ways
#one:By name
driver.delete_cookie('key')

#two:delete all of them at a time
driver.delete_all_cookies()

time.sleep(3)
driver.quit()
```

控制台输出如下

```
key - > value
{'domain': 'www.example.com', 'name': 'key', 'value': 'value', 'expiry': None, 'path': '/', 'secure': False}
```

注意:

- 1.expiry参数的值默认为None, 该参数一般是在网站有“保存密码”功能时进行使用, 一般不进行设置, 否则可能无法写入cookie, 获取写入结果为None
- 2.同一个域下的同一个网站仅在bc端保存一条cookie(可在上述代码中添加多条cookie验证)

➤ 改变用户代理(User-Agent)

示例:

```
# -*- coding: utf-8 -*-
from selenium import webdriver
import time

if __name__ == "__main__":
    profile = webdriver.FirefoxProfile()
    #set user-agent
    profile.set_preference('general.useragent.override', 'Mozilla/4.0
(Windows NT 6.1) Firefox/28.0')

    #create instance with profile parameter
    driver = webdriver.Firefox(profile)

    driver.maximize_window()
    driver.get('http://www.example.com')
```

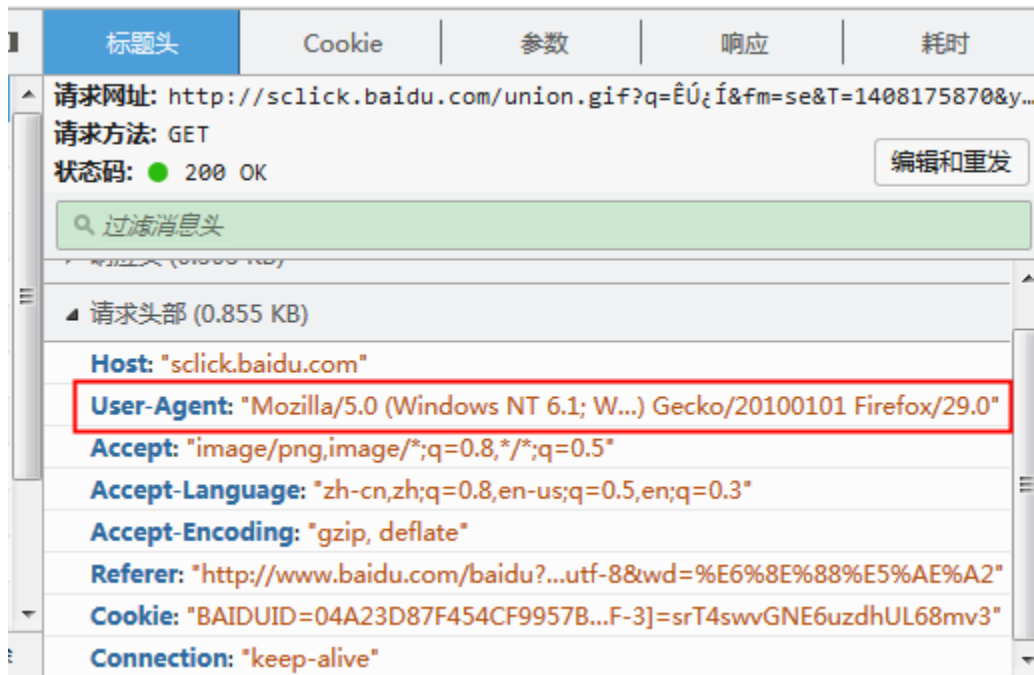
```
time.sleep(100)
driver.quit()
```

输出结果

标题头	Cookie	参数	响应	耗时
请求网址: http://www.example.com/ 请求方法: GET 状态码: ▲ 304 Not Modified				
<input type="text" value="过滤消息头"/>				
▲ 响应头 (0.256 KB)				
Accept-Ranges: "bytes"				
Cache-Control: "max-age=604800"				
Date: "Sat, 16 Aug 2014 08:10:54 GMT"				
Etag: ""359670651""				
Expires: "Sat, 23 Aug 2014 08:10:54 GMT"				
Last-Modified: "Fri, 09 Aug 2013 23:54:35 GMT"				
Server: "ECS (cpm/F858)"				
X-Cache: "HIT"				
▲ 请求头部 (0.374 KB)				
Host: "www.example.com"				
User-Agent: "Mozilla/4.0 (Windows NT 6.1) Firefox/28.0"				
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"				
Accept-Language: "zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3"				
Accept-Encoding: "gzip, deflate"				
Connection: "keep-alive"				
If-Modified-Since: "Fri, 09 Aug 2013 23:54:35 GMT"				
If-None-Match: ""359670651""				
Cache-Control: "max-age=0"				

说明:

1. 用户代理是指浏览器,它的信息包括硬件平台、系统软件、应用软件和用户个人偏好.同时用户代理还代表着搜索引擎。
2. 许多浏览器和下载软件会伪装 user agent 来回避某些侦测特定浏览器才能读取的网站。比如你用Firefox浏览一个网站,但该网站用Firefox浏览的话页面内容会显示不正常,该网站就会弹出请用IE浏览的信息框!这就是User-Agent起的作用
3. User-Agent也作为http请求头的一部分,火狐下可通过firebug查看user-agent



➤ driver详细与权衡

● HtmlUnit Driver

最快, 最轻的WebDriver实现, 基于HtmlUnit。HtmlUnit是基于java实现的无GUI的浏览器

对于任何语言绑定(java除外), Selenium Server需要用到HtmlUnit driver

```
driver = webdriver.Remote(u 'http://localhost:4444/wd/hub',
```

```
webdriver.DesiredCapabilities.HTMLUNIT.copy())
```

优点:

- webdriver最快的实现

- 纯java实现, 且平台独立

- 支持javascript

缺点

- 模拟浏览器的javascript行为

主流浏览器都没使用HtmlUnit使用的javascript引擎。如果使用HtmlUnit测试javascript和使用其它浏览器进行的测试极不相同。

当我们说“javascript”, 实际说的是“javascript和DOM”。尽管w3c定义了DOM, 但是每个浏览器有自己的特点并且在javascript的DOM实现和javascript交互都存在不同。虽然HtmlUnit有一个完整的DOM实现并对使用javascript有很好的支持,但是存在上述问题, 所以目前版本, HtmlUnit默认情况下禁用JavaScript。

开启 JavaScript

```
D:\Program Files (x86)\Python27>java -jar
```

```
selenium-server-standalone-2.40.0.jar
```

```
driver = webdriver.Remote("http://localhost:4444/wd/hub",  
                           webdriver.DesiredCapabilities.HTMLUNITWITHJS)
```

● Firefox Driver

支持浏览器版本Windows,Mac,Lniux: 火狐3.6, 10, 最新版

```
driver = webdriver.Firefox()
```

优点:

在真实浏览器中运行并且支持javascript

比Internet Explorer Driver运行得更快

缺点:

比HtmlUnit Dirver运行得更慢

修改火狐配置。

可对火狐profile进行获取并修改, 比如上述修改user-agent的例子, 再如下, 本地事件在Linux版本的firefox上被认为是稳定之前, 它们默认被禁用, 可手动开启

开启native events

```
profile = webdriver.FirefoxProfile()  
profile.native_events_enabled = True  
driver = webdriver.Firefox(profile)
```

注: 不需要单独下载火狐driver, driver包含在 selenium-server-stanalone.jar 包中

更多详细信息: <http://code.google.com/p/selenium/wiki/FirefoxDriver>

● Internet Explorer Driver

支持浏览器版本xp IE6\7\8, win7 9

```
driver = webdriver.Ie()
```

优点:

在一个真实浏览器中运行, 支持所有javascript

缺点:

仅用于windows

运行相对较慢

大部分版本对XPath的本地支持不是很好

IE6,IE5对CSS的本地支持不是很好

IE8,IE9本地支持CSS Selector, 但是不完全支持CSS3

注: 需要单独下载ie driver

更多详细信息: <http://code.google.com/p/selenium/wiki/InternetExplorerDriver>

● Chome Driver

```
driver = webdriver.Chrome()
```

优点:

运行在一个真实浏览器中并且支持javascript

Chrome是一个基于Webkit的浏览器, 所以它允许你确认你的网站是否支持Safari浏览器。

值得注意的是, Chrome使用自己的V8 JavaScript引擎, 而不是Safari的Nitro引擎, javascript的执行结果可能不相同

缺点:

比HtmlUnit Driver运行慢

注: 需要单独下载chrome driver, 并放在系统环境变量path包含的某个目录下

driver下载地址: <http://code.google.com/p/chromium/downloads/list>

更多详细信息: <http://code.google.com/p/selenium/wiki/ChromeDriver>

● Opera Driver

driver = webdriver.Opera()

调用出问题, 如下图

```
<terminated> D:\workspace\PyCase\src\Py27\PyCase1.py
Traceback (most recent call last):
  File "D:\workspace\PyCase\src\Py27\PyCase1.py", line 10, in <module>
    driver = webdriver.Opera()
  File "D:\Program Files (x86)\Python27\lib\site-packages\selenium\webdriver\opera\webdriver.py", line 54, in init
    'SELENIUM_SERVER_JAR')
Exception: No executable path given, please add one to Environment Variable 'SELENIUM_SERVER_JAR'
```

解决方案: 设置环境变量



即SELENIUM_SERVER_JAR D:\Program Files

(x86)\Python27\selenium-server-standalone-2.40.0.jar

关闭eclipse, 重新打开, 运行, 提示找不到桌面产品

```
6
7
8 if __name__ == "__main__":
9     driver = webdriver.Firefox()
10    driver = webdriver.Opera()
name
Problems Console PyUnit Expressions
D:\workspace\PyCase\src\Py27\PyCase1.py
raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.WebDriverException: Message: u'Unable to find executable for product Opera Desktop' ; St
```

解决方法: 浏览器版本太新, 替换浏览器版本, 11.51可支持 (备注: 经过测验, 发现浏览器打开后无动作, 具体为何? 不知道, 好像是selenium_standalone.jar包不是很支持)

备注: opera驱动随selenium-server-standalone或selenium-server安装而安装

更多详细信息: <http://code.google.com/p/selenium/wiki/OperaDriver>

● iOS Driver

详细信息: <http://selendroid.io/>

● Android Driver:

详细信息: <http://ios-driver.github.io/ios-driver/>

➤ 为使用RemoteDrivers运行独立Selenium Server

1. 下载selenium-server-standalone-<version>.jar和浏览器驱动

下载地址: <https://code.google.com/p/selenium/downloads/list>

2. 设置环境变量(如有必要)

把浏览器驱动后进行解压缩, 并将解压后的文件放在某个工作目录下(以python为例, 通常放在安装目录下, 同python.exe在同一目录), 并设置环境变量, 使得该目录包含在系统环境变量path中, 这样以便于selenium server而不要做任何修改就可以处理来自驱动的请求

3. 运行server

```
java -jar <path_to>/selenium-server-standalone-<version>.jar
```

4. 开启本地事件功能(如果想的话)

```
-Dwebdriver.enable.native.events = 1
```

查看帮助

```
java -jar <path_to>/selenium-server-standalone-<version>.jar -help
```

```
D:\Program Files (x86)\Python27>java -jar -Dwebdriver.enable.native.events=1 selenium-server-standalone-2.40.0.jar
2014-8-16 19:05:45 org.openqa.grid.selenium.GridLauncher main
信息: Launching a standalone server
19:05:45.550 INFO - Java: Sun Microsystems Inc. 14.0-b16
19:05:45.551 INFO - OS: Windows 7 6.1 x86
19:05:45.569 INFO - v2.40.0, with Core v2.40.0. Built from revision fbe29a9
19:05:45.742 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:4444/wd/hub
19:05:45.743 INFO - Version Jetty/5.1.x
19:05:45.755 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
19:05:45.756 INFO - Started HttpContext[/selenium-server,/selenium-server]
19:05:45.756 INFO - Started HttpContext[/,/]
19:05:45.795 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@4741d6
19:05:45.795 INFO - Started HttpContext[/wd,/wd]
19:05:45.798 INFO - Started SocketListener on 0.0.0.0:4444
19:05:45.798 INFO - Started org.openqa.jetty.jetty.Server@1bf52a5
```

➤ RemoteWebDriver

- 截图

步骤1: 运行Selenium Server standalone

java -jar <path>/selenium-server-standalone-<version>.jar

```
D:\Program Files (x86)\Python27>java -jar selenium-server-standalone-2.40.0.jar
2014-8-16 20:19:39 org.openqa.grid.selenium.GridLauncher main
信息: Launching a standalone server
20:19:39.130 INFO - Java: Sun Microsystems Inc. 14.0-b16
20:19:39.132 INFO - OS: Windows 7 6.1 x86
20:19:39.140 INFO - v2.40.0, with Core v2.40.0. Built from revision fbe29a9
20:19:39.210 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:4444/wd/hub
20:19:39.211 INFO - Version Jetty/5.1.x
20:19:39.212 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
```

步骤2: 运行程序

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
```

```
import time
```

```
if __name__ == "__main__":
```

```
    driver = webdriver.Remote('http://localhost:4444/wd/hub',
webdriver.DesiredCapabilities.FIREFOX.copy())
    driver.get('http://www.baidu.com')
    time.sleep(10)
```

```
    #take a screenshot and save to location 'D:/workspace/baidu.png'
```

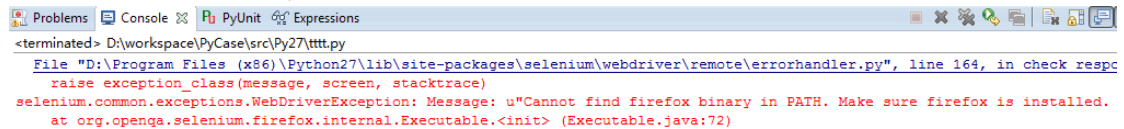
```
    driver.get_screenshot_as_file('D:/workspace/baidu.png')
```

```
    time.sleep(3)
```

```
    driver.quit()
```

运行错误, 如下:

Cannot find firefox binary in PATH. Make sure firefox is installed



解决方法:

重新安装火狐浏览器, 安装到默认路径, Windows下: 安装到C盘

● 使用firefox profile

步骤1: 运行selenium server standalone

步骤2: 运行程序

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import time
```

```
if __name__ == "__main__":
```

```
    fp = webdriver.FirefoxProfile()
    #set something on the profile...
    driver =
```

```
webdriver.Remote(desired_capabilities=webdriver.DesiredCapabilities.F
IREFOX, browser_profile=fp)
```

```
    driver.maximize_window()
    driver.get('http://www.baidu.com')
    time.sleep(9)
    driver.get_screenshot_as_file('D:/workspace/baidu.png')
    time.sleep(3)
    driver.quit()
```

● 使用ChromeOptions

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import time
```

```
if __name__ == "__main__":
```

```
    options = webdriver.ChromeOptions()
    #set some options.
    driver =
```

```
webdriver.Remote(desired_capabilities=options.to_capabilities())
    driver.maximize_window()
    driver.get('http://www.baidu.com')
    time.sleep(10)
    driver.quit()
```

运行代码错误:

```
selenium.common.exceptions.WebDriverException: Message: u'The path to the driver executable must be set by the webdriver.chrome.driver system property;
```

```
<terminated> D:\workspace\PyCase\src\Py27\http.py
am Files (x86)\Python27\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 164, in check_response
ion_class(message, screen, stacktrace)
exceptions.WebDriverException: Message: u'The path to the driver executable must be set by the webdriver.chrome.driver system property;
```

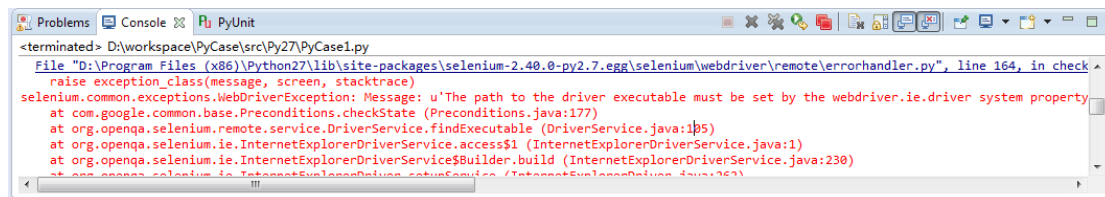
解决方法:

```
java -Dwebdriver.chrome.driver="<path_to>/chromedriver.exe" -jar
<path_to>/selenium-server-standalone-2.40.0.jar
```

```
D:\Program Files (x86)\Python27>java -Dwebdriver.chrome.driver="chromedriver.exe"
-jar selenium-server-standalone-2.40.0.jar
2014-8-16 21:56:59 org.openqa.grid.selenium.GridLauncher main
信息: Launching a standalone server
21:56:59.585 INFO - Java: Sun Microsystems Inc. 14.0-b16
21:56:59.585 INFO - OS: Windows 7 6.1 x86
21:56:59.592 INFO - v2.40.0, with Core v2.40.0. Built from revision fbe29a9
21:56:59.658 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:4444/wd/hub
21:56:59.660 INFO - Version Jetty/5.1.x
21:56:59.661 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
21:56:59.662 INFO - Started HttpContext[/selenium-server,/selenium-server]
21:56:59.662 INFO - Started HttpContext[/,/]
21:56:59.679 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@80f4cb
21:56:59.680 INFO - Started HttpContext[/wd,/wd]
21:56:59.682 INFO - Started SocketListener on 0.0.0.0:4444
21:56:59.683 INFO - Started org.openqa.jetty.jetty.Server@b4d3d5
```

备注: IE下运行, 也会出现类似的情况

```
selenium.common.exceptions.WebDriverException: Message: u'The path to the driver executable must be set by the webdriver.ie.driver system property;
```



解决方法 (同上):

```
java -Dwebdriver.ie.driver="<path_to>/IEDriverServer.exe.exe" -jar
<path_to>/selenium-server-standalone-2.40.0.jar
```

➤ 浏览器启动控制

● 使用代理

推荐直接在机器上手工配置代理, 如果需要在带不同配置或代理的情况下运行测试, 可以按照如下代码, 设置临时代理

```
✧ ie
# -*- coding: utf-8 -*-
from selenium import webdriver
```

```
import time

if __name__ == "__main__":
    PROXY = 'localhost:8080'

    #create a copy of desired capabilities object
    desired_capabilities =
webdriver.DesiredCapabilities.INTERNETEXPLORER.copy()

    #change the proxy properties of that copy
    desired_capabilities['proxy'] = {
        'httpProxy':PROXY,
        'ftpProxy':PROXY,
        'sslProxy':PROXY,
        'noProxy':None,
        'proxyType': 'MANUAL',
        'class': 'org.openqa.selenium.Proxy',
        'autodetect':False
    }

    #you have to use remote, otherwise you'll have to code it yourself in python
to
    #dynamically changing the system proxy preferences
    driver = webdriver.Remote('http://localhost:4444/wd/hub',
desired_capabilities)
    driver.maximize_window()
    driver.get('http://192.168.206.8/qcbin:8080')
    time.sleep(10)
    driver.quit()
```

注意: 经过测试, 似乎程序运行完后不会自动删除代理

✧ Chrome

同Ie

✧ Firefox

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
from selenium.webdriver.common.proxy import *
import time

if __name__ == "__main__":
    my_proxy = '127.0.0.1:8080'

    proxy = Proxy({
```

```

        'proxyType': ProxyType.MANUAL,
        'httpProxy': my_proxy,
        'ftpProxy': my_proxy,
        'sslProxy': my_proxy,
        'noProxy': '' #set this value as desired
    })

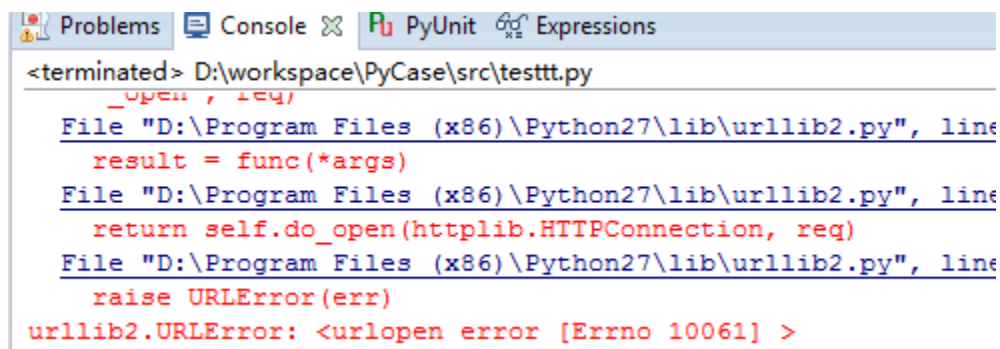
driver = webdriver.Firefox(proxy = proxy)

#for remote
caps = webdriver.DesiredCapabilities.FIREFOX.copy()
proxy.add_to_capabilities(caps)
driver = webdriver.Remote(desired_capabilities = caps)

driver.maximize_window()
driver.get('http://192.168.206.8/qcbin:8080')
time.sleep(10)
driver.quit()

```

备注:



```

<terminated> D:\workspace\PyCase\src\testtt.py
  _open, req)
File "D:\Program Files (x86)\Python27\lib\urllib2.py", line
    result = func(*args)
File "D:\Program Files (x86)\Python27\lib\urllib2.py", line
    return self.do_open(httplib.HTTPConnection, req)
File "D:\Program Files (x86)\Python27\lib\urllib2.py", line
    raise URLError(err)
urllib2.URLError: <urlopen error [Errno 10061] >

```

上述的运行结果都报错了。。怎么解决？我也不知道。。郁闷。。

➤ 测试设计考虑

● UI Mapping

UI map是一种机制：为了在UI元素的标识或路径改变的时候更容易做出改变，在一个地方为一个测试套件存放所有的定位器。测试脚本然后使用UI map来定位要测试的元素。基本的，UI map是与应用程序的UI元素对应的测试脚本对象的存储胶囊。

test script-->UI map element->UI element

在脚本和UI元素之间引入一层抽象层

UI map好处:

主要是让测试脚本管理更容易。当需要对某个定位器编辑时，有一个更容易找到哪些对象的中心位置，而不要搜索整个测试脚本代码。同时，它允许仅在一个地方修改定位器，而不需要在多个地方做修改。

总体来说，有两点很大的好处，如下

1. 为ui对象使用一个中心化的位置,而不是让它们分散在整个脚本,这让脚本维护更容易维护
2. 可以为html标识和名字附上更易读的名字,提高测试脚本的可靠性

有几种方式来实现UI Map,可以创建类或struct结构体,其每个属性(元素)仅存放一个定位器,也可以选择text文件来存储键值对。

● 页面对象设计模式

一个页面对象是一个对象化的类,充当接口自动化测试页面的接口。无论何时,测试都可以页面对象类的方法同那个页面的UI交互。这样做的好处是,如果那个页面的UI改变了,测试本身不需要修改,仅需要修改包含在页面对象类里的代码,结果是,为支持新的UI所做的改变都位于同一个地方

页面对象设计模式有以下优点:

1. 测试代码和页面详细代码,如定位器(或他们的使用,如果你正在使用一个UI map),以及代码布局之间界限清楚.
2. 页面提供的服务或者操作有一个单一的存储,而不是分散在整个测试中

基本原则:

从不对页面对象本身做验证或者断言,这是测试的一部分并且应该放在测试代码里面,绝不能放在页面对象里,这个页面对象仅包含页面呈现,通过页面方法提供的各种服务,但是没有与“正在测试什么”相关的代码。

仅有一种情况,验证可以并且应该放在页面对象,那就是确认这个页面和页面上的关键元素被正确的加载。这个确认必须在初始化页面对象时进行。即构造函数中写入。如下

```
/**
 * Page Object encapsulates the Sign-in page.
 */
public class SignInPage {

    private Selenium selenium;

    public SignInPage(Selenium selenium) {
        this.selenium = selenium;
        if(!selenium.getTitle().equals("Sign in page")) {
            throw new IllegalStateException("This is not sign in page, current
                +selenium.getLocation());
        }
    }

    /**
     * Login as valid user
     *
     * @param userName
     * @param password
     * @return HomePage object
     */
    public HomePage loginValidUser(String userName, String password) {
        selenium.type("usernamefield", userName);
        selenium.type("passwordfield", password);
        selenium.click("sign-in");
        selenium.waitForPageToLoad("waitPeriod");

        return new HomePage(selenium);
    }
}
```

一个页面对象可以不用代表整个页面，可以用页面对象来代表页面上的组件。如果一个页面有多个组件，为每个组件创建一个单独的页面对象有利于提高代码的可维护性

具体参考代码：面向对象-编写自动化测试脚本实例

- 数据驱动测试

指用不同的数据来执行相同的测试。这些数据集往往是从其它文件，.csv、.text等或者从某个数据库读取的。

- 数据库验证

把UI界面的数据同数据库中的实际数据进行对比

➤ Selenium-Grid

- 什么是Selenium-Grid?

在不同机器上，不同操作系统，不同浏览器下同时运行多个测试。本质上，Selenium-Grid支持分布式测试的执行。

- 使用

1. 下载Selenium-Server jar

下载Selenium-Server jar格式文件，并放在某个文件夹下。

2. 开启Hub

使用缺省参数开启一个hub。

```
java -jar selenium-server-standalone-2.38.0.jar -role hub
```

3. 开启一个节点(Node)

```
java -jar selenium-server-standalone-2.38.0.jar -role node -hub
```

```
http://localhost:4444/grid/register
```

暂时用不到,所以我暂时不做研究,有兴趣的可以参考以下连接

http://docs.seleniumhq.org/docs/07_selenium_grid.jsp

➤ webdriver原理解析

下载Selenium server jar包,然后可以使用下面的命令来启动Selenium server

```
D:\>java -Dwebdriver.ie.driver="D:\Program Files  
(x86)\Python27\IEDriverServer.exe" -jar selenium-server-standalone-2.40.0.jar
```

coding= utf-8 #可加可不加,防止中文乱码

```
from selenium import webdriver
```

```
from selenium.webdriver.common.desired_capabilities import  
DesiredCapabilities
```

```
if __name__ == "__main__":
```

```
    driver =
```

```
    webdriver.Remote(desired_capabilities=DesiredCapabilities.INTERNETEXPLORER)  
    driver.get("http://www.baidu.com/")
```

```
    # 输入框输入内容
```

```
    driver.find_element_by_id("kw1").send_keys("selenium")
```

```
    driver.find_element_by_id("su1").click()
```

```
    driver.close()
```

```
    driver.quit()
```

webdriver 原理:

1. WebDriver 启动目标浏览器,并绑定到指定端口。该启动的浏览器实例做为 webdriver 的 remote server。
2. Client 端通过 CommandExcuter 发送 HTTPRequest 给 remote server 的侦听端口(通信协议: the webdriver wire protocol)
3. Remote server 需要依赖原生的浏览器组件(如: IEDriver.dll,chromedriver.exe),来转化转化浏览器的native 调用。

```
D:\>java -Dwebdriver.ie.driver="D:\Program Files
```

```
(x86)\Python27\IEDriverServer.e
```

```
xe" -jar selenium-server-standalone-2.40.0.jar
```

```
2014-7-22 22:24:03 org.openqa.grid.selenium.GridLauncher main
信息: Launching a standalone server
22:24:03.670 INFO - Java: Sun Microsystems Inc. 14.0-b16
22:24:03.670 INFO - OS: Windows 7 6.1 x86
22:24:03.686 INFO - v2.40.0, with Core v2.40.0. Built from revision fbe29a9
22:24:03.748 INFO - RemoteWebDriver instances should connect to:
http://127.0.0.1:4444/wd/hub
22:24:03.748 INFO - Version Jetty/5.1.x
22:24:03.748 INFO - Started
HttpContext[/selenium-server/driver,/selenium-server/driver]
22:24:03.748 INFO - Started HttpContext[/selenium-server,/selenium-server]
22:24:03.748 INFO - Started HttpContext[/,/]
22:24:03.764 INFO - Started
org.openqa.jetty.jetty.servlet.ServletHandler@82c01f

22:24:03.764 INFO - Started HttpContext[/wd,/wd]
22:24:03.779 INFO - Started SocketListener on 0.0.0.0:4444
22:24:03.779 INFO - Started org.openqa.jetty.jetty.Server@128e20a
-----
```

创建新session

```
22:24:10.284 INFO - Executing: [new session: Capabilities [{platform=WINDOWS,
javascriptEnabled=true, browserName=internet explorer, version=}] at URL:
/session)
22:24:10.284 INFO - Creating a new session for Capabilities [{platform=WINDOWS,
javascriptEnabled=true, browserName=internet explorer, version=}]
Started InternetExplorerDriver server (32-bit)2.40.0.0
Listening on port 39164
22:24:13.046 INFO - I/O exception (java.net.SocketException) caught when
processing request: Software caused connection abort: recv failed
22:24:13.046 INFO - Retrying request
22:24:13.077 INFO - Done: /session
```

webdriver通过get方式发送请求

```
22:24:13.092 INFO - Executing: [get: http://www.baidu.com/] at URL:
session/83f0f523-0a52-4259-ad85-9215066fe330/url)
22:24:13.736 INFO - Done: /session/83f0f523-0a52-4259-ad85-9215066fe330/url
```

查找百度搜索输入框

```
22:24:13.736 INFO - Executing: [find element: By.id: kw1] at URL: /session/83f0f
523-0a52-4259-ad85-9215066fe330/element)
22:24:13.799 INFO - Done:
/session/83f0f523-0a52-4259-ad85-9215066fe330/element
```

输入搜索内容selenium

```
22:24:13.799 INFO - Executing: [send keys: 0 [[InternetExplorerDriver: internet
explorer on WINDOWS (b939fd02-292f-449e-8f67-71c6dd1f4534)] -> id: kw1], [s,
e,l, e, n, i, u, m]] at URL:
```



```
/session/83f0f523-0a52-4259-ad85-9215066fe330/element/0/value)
22:24:14.126 INFO - Done:
/session/83f0f523-0a52-4259-ad85-9215066fe330/element/0/value
查找“百度一下”搜索按钮
22:24:14.126 INFO - Executing: [find element: By.id: su1] at URL: /session/83f0f
523-0a52-4259-ad85-9215066fe330/element)
22:24:14.189 INFO - Done:
/session/83f0f523-0a52-4259-ad85-9215066fe330/element
点击搜索按钮
22:24:14.189 INFO - Executing: [click: 1 [[InternetExplorerDriver: internet
explorer on WINDOWS (b939fd02-292f-449e-8f67-71c6dd1f4534)] -> id: su1]] at URL:
/session/83f0f523-0a52-4259-ad85-9215066fe330/element/1/click)
22:24:14.532 INFO - Done:
/session/83f0f523-0a52-4259-ad85-9215066fe330/element/1/click
关闭浏览器
22:24:14.532 INFO - Executing: [close window] at URL:
/session/83f0f523-0a52-4259-ad85-9215066fe330/window)
22:24:14.594 INFO - Done:
/session/83f0f523-0a52-4259-ad85-9215066fe330/window
删除会话
22:24:14.594 INFO - Executing: [delete session:
83f0f523-0a52-4259-ad85-9215066fe330] at URL:
/session/83f0f523-0a52-4259-ad85-9215066fe330)
22:24:15.624 INFO - Done: /session/83f0f523-0a52-4259-ad85-9215066fe330
```

➤ 参考链接: <http://docs.seleniumhq.org/docs/>