

selenium的其它使用方法

@selenium控制标签页的切换

窗口切换：

获取所有标签页的窗口句柄

利用窗口句柄字切换到句柄指向的标签页

窗口句柄：指的是指向标签页对象的标识

解析：

#1.获取当前所有的标签页的句柄构成的列表

```
current_windows = driver.window_handles
```

#2.根据标签页句柄列表索引下标进行切换

```
driver.switch_to.window(current_windows[0])
```

```
from selenium import webdriver
import time
# 创建driver对象
driver = webdriver.Chrome()
# 访问的起始的url地址
```

```
start_url = 'https://www.baidu.com'
# 访问
driver.get(url=start_url)
time.sleep(1)
driver.find_element_by_id('kw').send_keys('python')
time.sleep(1)
driver.find_element_by_id('su').click()
time.sleep(1)

# 通过执行js来新开一个标签页
js = 'window.open("https://www.csdn.net");'
driver.execute_script(js)
time.sleep(1)

# 1. 获取所有浏览器窗口
windows = driver.window_handles

# 2. 根据窗口索引进行切换
driver.switch_to.window(windows[0])
time.sleep(1)
driver.switch_to.window(windows[1])
```

@.selenium控制iframe的切换

iframe是html中常用的一种技术，即一个页面中嵌套了另一个网页，selenium默认是访问不了frame中的内容的，对应的解决思路是
driver.switch_to.frame(frame_element)

网易邮箱登录

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

```
def login(user, password):
    driver = webdriver.Chrome()
    driver.get("https://email2.163.com/")
    # browser.maximize_window()

    driver.switch_to.frame(driver.find_element
_by_xpath('//iframe[starts-with(@id,"x-
URS")]'))
    time.sleep(1)

    driver.find_element_by_xpath('//input[@nam
e="email"]').send_keys(user)

    driver.find_element_by_xpath('//input[@nam
e="password"]').send_keys(password)
    driver.find_element_by_xpath('//*
[@id="dologin"]').click()
```

```
time.sleep(2)
print(driver.page_source)
driver.save_screenshot("163.png")
time.sleep(3)
# driver.quit()

if __name__ == '__main__':
    login('163邮箱帐号', '密码')
```

driver.switch_to.default_content() --> 恢复默认
页面方法

(在frame表单中操作其他页面，必须先回到默认页面，才能进一步操作)

1. 跳回最外层的页面

driver.switch_to.default_content() -- 切换到最外层
(对于多层页面，可通过该方法直接切换到最外层)

2. 跳回上层的页面

driver.switch_to.parent_frame() -- 进行向上的单层
切换

```

import time
from selenium import webdriver

driver = webdriver.Chrome()

url = 'https://mail.qq.com/cgi-bin/loginpage'
driver.get(url)
time.sleep(2)

login_frame = driver.find_element_by_id('login_frame') # 根据id定位 frame元素
driver.switch_to.frame(login_frame) # 转向到该frame中

driver.find_element_by_xpath('//*[@id="u"]').send_keys('1596930226@qq.com')
time.sleep(2)

driver.find_element_by_xpath('//*[@id="p"]').send_keys('hahamimashicuode')
time.sleep(2)

driver.find_element_by_xpath('//*[@id="login_button"]').click()
time.sleep(2)

"""操作frame外边的元素需要切换出去"""
windows = driver.window_handles
driver.switch_to.window(windows[0])

content = driver.find_element_by_class_name('login_pictures_title').text
print(content)

driver.quit()

```

切换到定位的frame标签嵌套的页面中

driver.switch_to.frame(通过find_element_by函数定位的frame、iframe标签对象)

利用切换标签页的方式切出frame标签

windows = driver.window_handles

driver.switch_to.window(windows[0])

@利用selenium获取cookie的方法

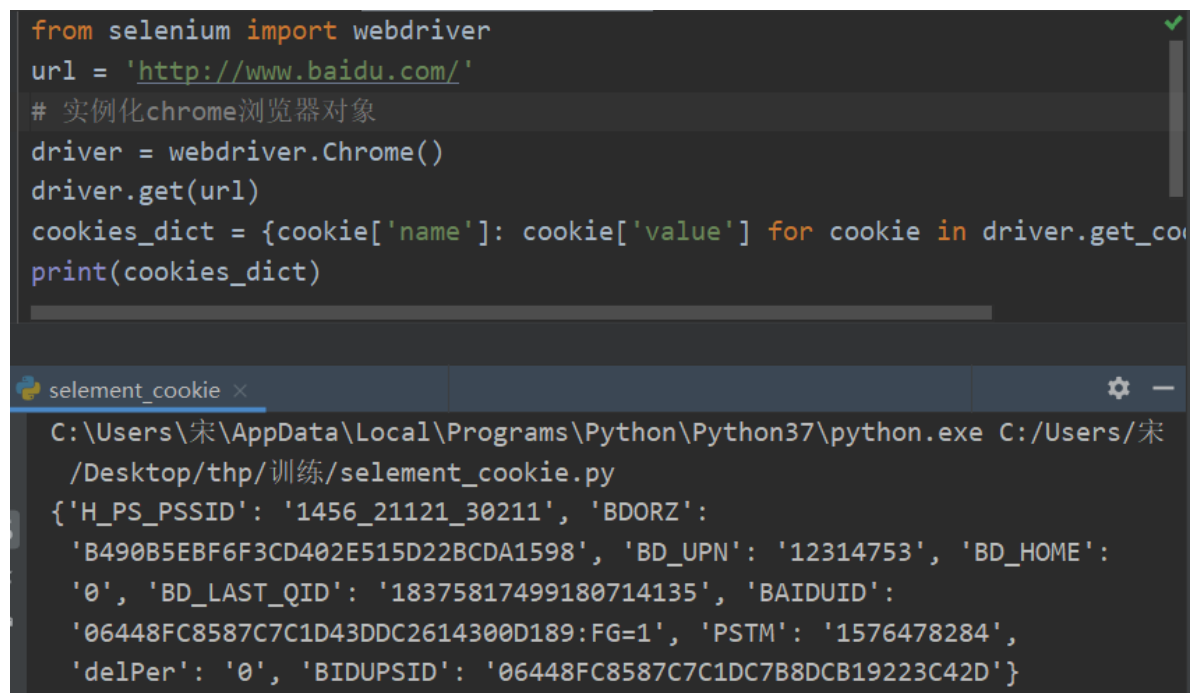
获取cookie

`driver.get_cookies()` 返回列表，其中包含的是完整的cookie信息，需要转换为字典

字典推导式转换

```
cookies_dict = {cookie['name']: cookie['value']  
for cookie in driver.get_cookies()}
```

```
from selenium import webdriver  
url = 'http://www.baidu.com/'  
# 实例化chrome浏览器对象  
driver = webdriver.Chrome()  
driver.get(url)  
cookies_dict = {cookie['name']: cookie['value'] for cookie in driver.get_cookies()}  
print(cookies_dict)
```



The screenshot shows a terminal window with the following output:

```
C:\Users\宋\AppData\Local\Programs\Python\Python37\python.exe C:/Users/宋/Desktop/thp/训练/selement_cookie.py  
{'H_PS_PSSID': '1456_21121_30211', 'BDORZ':  
'B490B5EBF6F3CD402E515D22BCDA1598', 'BD_UPN': '12314753', 'BD_HOME':  
'0', 'BD_LAST_QID': '18375817499180714135', 'BAIDUID':  
'06448FC8587C7C1D43DDC2614300D189:FG=1', 'PSTM': '1576478284',  
'delPer': '0', 'BIDUPSID': '06448FC8587C7C1DC7B8DCB19223C42D'}
```

删除cookie

- 删除一条cookie

```
driver.delete_cookie('cookie_name')
```

- 删除所有的cookie

driver.delete_all_cookies()

2.页面等待

分类：

- 1.强制等待

time.sleep()

缺点：不智能，设置的时间太短，元素还没有加载出来，设置的时间太长，浪费时间

- 2.隐式等待

针对元素定位，隐式等待设置了一个时间，在一段时间内判断元素是否定位成功，如果完成了，就进行下一步

在设置时间内没有定位成功，则会报超时加载

隐式等待

```
from selenium import webdriver
from selenium.webdriver import
ChromeOptions
driver = webdriver.Chrome()
# 最多等待你10秒，10秒过后，没有找到就报错
# 一找到，就接着往下执行
driver.implicitly_wait(10)
driver.get('https://www.csdn.net')
```

- 3.显示等待

明确等待某一个元素，超时则报异常

```
from selenium.webdriver.support import
expected_conditions as EC
from selenium.webdriver.support.wait
import WebDriverWait
from selenium.webdriver.common.by import
By
from selenium import webdriver
driver = webdriver.Chrome()
driver.maximize_window()
driver.get('https://www.baidu.com')
# 通过文本名称进行定位
# driver.find_element_by_link_text('人工智
能').click()

WebDriverWait(driver, 20,
0.5).until(EC.presence_of_element_located
((By.LINK_TEXT, '地图'))).click()
"""
```

参数20表示最长等待20s

参数0.5表示0.5s检查一次规定的标签书否存在

EC.presence_of_all_elements_located((By.LINK_TEXT, '地图'))：通过文本内容定位标签

每0.5s一次检查，通过链接文本内容定位标签是否存在，如果存在就向下继续执行，如果不存在20s上限就报错

"""

手动实现页面等待

原理：

利用强制等待和显示等待的思路来手动实现

不停的判断或有次数限制的判断某一个标签对象是否加载完毕（是否存在）

3.selenium控制浏览器执行js代码的方法

执行js的方法： `driver.execute_script(js)`

```
from selenium import webdriver
import time
driver = webdriver.Chrome()
driver.get('https://www.taobao.com')
time.sleep(1)
for i in range(10):
    i += 1
    try:
        time.sleep(3)
        element = driver.find_element_by_xpath('//div[@class="shop-inner"]/h3[1]/a')
        result = element.get_attribute('href')
        print(result)
        break
    except:
        # 网页滚动条的拖动
        js = 'window.scrollTo(0,{})'.format(i * 500) # js语句
        driver.execute_script(js) # 执行js的方法
driver.quit()
```

4.selenium使用代理IP

使用代理ip的方法

- **实例化配置对象**
- ○ options = webdriver.ChromeOptions()
- **配置对象添加使用代理ip的命令**
- ○ options.add_argument('--proxy-server=<http://202.20.16.82:9527>')
- **实例化带有配置对象的driver对象**
- ○ driver = webdriver.Chrome('./chromedriver', chrome_options=options)

```
from selenium import webdriver

# 1.创建一个配置对象
options = webdriver.ChromeOptions()
# 2.使用代理
options.add_argument('--proxy-server=http://192.168.129.130')
# 3.创建driver对象
driver = webdriver.Chrome(options=options)
# 4.设置起始的url地址
start_url = 'https://www.baidu.com'
# 访问
driver.get(url=start_url)
```

5.selenium替换user-agent

替换user-agent的方法

- **实例化配置对象**
- ○ options = webdriver.ChromeOptions()
- **配置对象添加替换UA的命令**
- ○ options.add_argument('--user-agent=Mozilla/5.0 HAHA')
- **实例化带有配置对象的driver对象**
- ○ driver = webdriver.Chrome('./chromedriver', chrome_options=options)

```
from selenium import webdriver
from fake_useragent import UserAgent
ua = UserAgent()
# 1.创建一个配置对象
options = webdriver.ChromeOptions()
# 2.使用代理
options.add_argument('--user-agent={}'.format(ua.chrome))
# ua = 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36'
# options.add_argument('user-agent' + ua)
# 3.创建driver对象
driver = webdriver.Chrome(options=options)
# 4.设置起始的url地址
start_url = 'https://www.baidu.com'
# 访问
driver.get(url=start_url)
```

```
print(driver.title)
```

6.拓展知识点

selenium防检测

```
from selenium import webdriver
from selenium.webdriver import
ChromeOptions

option = ChromeOptions()          #实例化一个
ChromeOptions对象
option.add_experimental_option('excludeSwit
ches', ['enable-automation'])    #以键值对的形式加入参数

bro =
webdriver.Chrome(executable_path='./chromed
river.exe',options=option)      #在调用浏览器驱动
时传入option参数就能实现
```

第二种

```
driver.execute_cdp_cmd("Page.addScriptToEva
luateOnNewDocument", { "source": ""
Object.defineProperty(navigator,
'webdriver', { get: () => undefined }) ""
})
```

selenium禁止弹窗

```
from selenium import webdriver
from selenium.webdriver.chrome.options
import Options

chrome_options = Options()

# 禁止弹窗
prefs = {

    'profile.default_content_setting_values':
        {
            'notifications': 2
        }
}

# 禁止弹窗加入
chrome_options.add_experimental_option('pre
fs', prefs)
driver =
webdriver.Chrome(chrome_options=chrome_opti
ons)
driver.get(url)
# 就可以访问无通知弹窗的浏览器了
```

selenium清空输入

```
from selenium import webdriver
```

```
import time

browser = webdriver.Chrome()
browser.maximize_window() # 设置浏览器大小：
全屏
browser.get('https://www.baidu.com')

# 定位输入框
input_box =
browser.find_element_by_id('kw')
try:
    # 输入内容: selenium
    input_box.send_keys('selenium')
    print('搜索关键词: selenium')
except Exception as e:
    print('fail')
# 输出内容: 搜索关键词: selenium

# 定位搜索按钮
button = browser.find_element_by_id('su')
try:
    # 点击搜索按钮
    button.click()
    print('成功搜索')
except Exception as e:
    print('fail搜索')
# 输出内容: 成功搜索

# clear(): 清空输入框
try:
```

```
input_box.clear()
print('成功清空输入框')
except Exception as e:
    print('fail清空输入框')
# 输出内容：成功清空输入框
```

selenium模拟回车

```
from selenium import webdriver
import time

browser = webdriver.Chrome()
browser.maximize_window() # 设置浏览器大小：
全屏
browser.get('https://www.baidu.com')

# 定位输入框
input_box =
browser.find_element_by_id('kw')
# 输入关键词：selenium
input_box.send_keys('selenium')
# 模拟回车操作
try:
    input_box.submit()
    print('成功回车')
except Exception as e:
    print('fail')
# 输出内容：成功回车
```

selenium下拉框选择

```
# 导入需要的模块Select()类是用来管理下拉框的
from selenium import webdriver
from selenium.webdriver.support.select
import select
import time
# 创建浏览器对象
driver = webdriver.Chrome()
driver.maximize_window()
# 访问贴吧的高级搜索
driver.get('https://tieba.baidu.com/f/search/adv')

# 定位到下拉框元素
el_select =
driver.find_element_by_name('rn')
# 创建一个下拉框对象
xialakuang = select(el_select)

# 三种方法选择下拉框选项
# 第一、通过选项的索引来选定选项(索引从0开始算)
xialakuang.select_by_index(0)
time.sleep(1)
xialakuang.select_by_index(2)
time.sleep(1)
xialakuang.select_by_index(1)
time.sleep(1)

# 第二种方法: 通过option标签的属性值选择
xialakuang.select_by_value('20')
time.sleep(1)
```



```
xialakuang.select_by_value('10')
time.sleep(1)
xialakuang.select_by_value('30')
time.sleep(1)
```

第三种：通过文本选择（下拉框的值）

```
xialakuang.select_by_visible_text('每页显示
20条')
time.sleep(1)
xialakuang.select_by_visible_text('每页显示
10条')
time.sleep(1)
xialakuang.select_by_visible_text('每页显示
30条')
time.sleep(1)
```

打印选择的文本

查看第一个已选（若有多个已选则打印第一个，只有一个已选则打印一个）

```
print(xialakuang.first_selected_option.text)
```

打印所有已选的选项的文本

```
yixuan = xialakuang.all_selected_options
for i in yixuan:
    print('已选',i.text)
```

打印是否是多选

```
print(xialakuang.is_multiple)
```

打印所有选项（包括已选和未选的）

```
m_list = xialakuang.options
for a in m_list:
```

```
print('选项',a.text)
```

```
# # 关闭浏览器  
# driver.quit()
```

selenium鼠标悬停

```
from  
selenium.webdriver.common.action_chains  
import ActionChains  
# 定位到需要悬停的标签  
move = driver.find_element_by_id("xpath语  
法")  
# 开始悬停  
ActionChains(self.driver).move_to_element(m  
ove).perform()
```