base_page

```python
"""
基础对象
"""
import json
import os.path as path

from selenium import webdriver

from stub.config.settings import BASE_DIR


def browser():
    _ = webdriver.ChromeOptions()
    # 无头模式
    # _.headless = True
    _.add_argument("--start-maximized")
    # _.add_argument('--no-sandbox --start-maximized')
    driver = webdriver.Chrome(options=_)
    driver.implicitly_wait(10)
    return driver


class BasePage:
    def __init__(self, d):
        self.driver: webdriver.Chrome = d

    def quit(self):
        input("输入任意字符退出: ")
        self.driver.quit()

    def cookies_login(self):
        with open(path.join(BASE_DIR, "lib", "cookies.json"), "r") as f:
            for i in json.load(f):
                self.driver.add_cookie(i)

```

webdriver

```python
from selenium import webdriver
driver = webdriver.Chrome()
# 地址栏输入url地址并确认
driver.get(url=url)

```

```
# 关闭当前页
driver.close()

# 浏览器窗口最大化
driver.maximize_window()

# 设置当前窗口的宽度和高度。
driver.set_window_size()

# 设置当前窗口的x、y位置。
driver.set_window_position()

# browser执行JS脚本
driver.execute_script('window.scrollTo(0,document.body.scrollHeight)')

driver.get('https://www.baidu.com')
driver.get('https://news.baidu.com')
# 后退
driver.back()

# 前进
driver.forward()

# 刷新当前页面。
driver.refresh()

# 退出驱动程序并关闭浏览器
driver.quit()

# HTML结构源码
driver.page_source
# 从html源码中搜索指定字符串,没有找到返回: -1,经常用于判断是否为最后一页
driver.page_source.find('字符串')

# 返回当前页面的标题。
driver.title

# 获取当前页面的URL。
driver.current_url

# 隐式等待
driver.implicitly_wait()

# 获取当前句柄
driver.current_window_handle

# 所有句柄
driver.window_handles

driver.switch_to
```

WebElement

```
# 点击元素。
click()

# 获取元素的给定属性或属性。
get_attribute()

# 元素是否对用户可见。
is_displayed()

# 返回元素是否可用。
is_enabled()

# 返回元素是否被选中。
is_selected()

# 元素的大小。
size

# 元素的文本。
text

【1】文本框操作
    1.1) node.send_keys('')   - 向文本框发送内容
    1.2) node.clear()         - 清空文本
    1.3) node.get_attribute('value') - 获取文本内容

【2】按钮操作
    1.1) node.click()       - 点击
    1.2) node.is_enabled() - 判断按钮是否可用
    1.3) node.get_attribute('value') - 获取按钮文本
```

定位节点八种方法

id与name 定位

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

def demo_id_and_name():
    _ = webdriver.ChromeOptions()
    _.add_argument("--start-maximized")
    driver = webdriver.Chrome(options=_)
```

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

    driver.find_element(By.ID, "kw").send_keys("Selenium我要自学网")
    driver.find_element(By.NAME, "wd").send_keys("Selenium我要自学网")

    sleep(2)
    driver.find_element(By.ID, "su").click()
    driver.close()


```

tag_name定位

```
def demo_tag_name():
    # 案例：打开我要自学网页面，在用户名输入框输入用户名“selenium”
    _ = webdriver.ChromeOptions()
    _.add_argument("--start-maximized")
    driver = webdriver.Chrome(options=_)
    driver.get("http://www.51zxw.com")
    # 定位标签名为input的元素
    driver.find_element(By.TAG_NAME, "input").send_keys("selenium")
    # 获取页面所有标签名称为“input”的标签。
    driver.find_elements(By.TAG_NAME, "input")[0].send_keys("selenium")
    sleep(3)
    driver.quit()
```

class_name定位

```
def demo_class_name():
    # 根据标签中属性class来进行定位的一种方法
    _ = webdriver.ChromeOptions()
    _.add_argument("--start-maximized")
    driver = webdriver.Chrome(options=_)
    driver.get("http://www.baidu.com")
    driver.find_element(By.CLASS_NAME, "s_ipt").send_keys("Selenium 我要自学网")
    sleep(2)
    driver.find_element(By.ID, "su").click()
    sleep(3)
    driver.quit()
```

link_text与partial_link_text定位

```python
def demo_link():
    # link_text定位就是根据超链接文字进行定位。
    _ = webdriver.ChromeOptions()
    _.add_argument("--start-maximized")
    driver = webdriver.Chrome(options=_)
    driver.get("http://www.51zxw.net/")
    driver.find_element(By.LINK_TEXT, '程序设计').click()
    sleep(3)
    driver.find_element(By.PARTIAL_LINK_TEXT, '数据库教程').click()
    sleep(3)
    driver.quit()
```

XPath定位

xpath解析

- **定义**

```
XPath即为XML路径语言，它是一种用来确定XML文档中某部分位置的语言，同样适用于HTML文档的检索
```

- **匹配演示 - 猫眼电影top100**

```
【1】查找所有的dd节点
    //dd
【2】获取所有电影的名称的a节点：所有class属性值为name的a节点
    //p[@class="name"]/a
【3】获取dl节点下第2个dd节点的电影节点
    //dl[@class="board-wrapper"]/dd[2]
【4】获取所有电影详情页链接：获取每个电影的a节点的href的属性值
    //p[@class="name"]/a/@href

【注意】
    1> 只要涉及到条件，加 [] ：//dl[@class="xxx"]    //dl/dd[2]
    2> 只要获取属性值，加 @  ：//dl[@class="xxx"]    //p/a/@href
```

- **选取节点**

```
【1】// ：从所有节点中查找（包括子节点和后代节点）
【2】@  ：获取属性值
  2.1> 使用场景1（属性值作为条件）
       //div[@class="movie-item-info"]
  2.2> 使用场景2（直接获取属性值）
       //div[@class="movie-item-info"]/a/img/@src

【3】练习 - 猫眼电影top100
  3.1> 匹配电影名称
      //div[@class="movie-item-info"]/p[1]/a/@title
  3.2> 匹配电影主演
      //div[@class="movie-item-info"]/p[2]/text()
```

```
    3.3> 匹配上映时间
        //div[@class="movie-item-info"]/p[3]/text()
    3.4> 匹配电影链接
        //div[@class="movie-item-info"]/p[1]/a/@href
```

- **匹配多路径（或）**

```
xpath表达式1 | xpath表达式2 | xpath表达式3
```

- **常用函数**

```
【1】text() ： 获取节点的文本内容
    xpath表达式末尾不加 /text() :则得到的结果为节点对象
    xpath表达式末尾加 /text() 或者 /@href ： 则得到结果为字符串

【2】contains() ： 匹配属性值中包含某些字符串节点
    匹配class属性值中包含 'movie-item' 这个字符串的 div 节点
     //div[contains(@class,"movie-item")]
```

- **终极总结**

```
【1】字符串：xpath表达式的末尾为：/text() 、/@href  得到的列表中为'字符串'

【2】节点对象：其他剩余所有情况得到的列表中均为'节点对象'
    [<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]
    [<element div at xxxa>,<element div at xxxb>]
    [<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]
```

- **课堂练习**

```
【1】匹配汽车之家-二手车，所有汽车的链接 ：
    //li[@class="cards-li list-photo-li"]/a[1]/@href
    //a[@class="carinfo"]/@href
【2】匹配汽车之家-汽车详情页中，汽车的
     2.1)名称：  //div[@class="car-box"]/h3/text()
     2.2)里程：  //ul/li[1]/h4/text()
     2.3)时间：  //ul/li[2]/h4/text()
     2.4)挡位+排量：//ul/li[3]/h4/text()
     2.5)所在地：//ul/li[4]/h4/text()
     2.6)价格：   //div[@class="brand-price-item"]/span[@class="price"]/text()
```

```
xpath绝对与相对定位
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.baidu.com")


# 绝对路径定位
```

```
driver.find_element_by_xpath("/html/body/div[2]/div[1]/div/div[1]/div/form/span[1]/input").send_keys("51zxw")

# 利用元素熟悉定位--定位到input标签中为kw的元素
driver.find_element_by_xpath("//input[@id='kw']").send_keys("Selenium")

# 定位input标签中name属性为wd的元素
driver.find_element_by_xpath("//input[@name='wd']").send_keys("Selenium")

# 定位所有标签元素中，class属性为s_ipt的元素
driver.find_element_by_xpath("//*[@class='s_ipt']").send_keys("Python3")

driver.find_element_by_id('su').click()

sleep(3)
driver.quit()


Xpath层级与逻辑定位
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.51zxw.net/")
#层级和属性结合定位--自学网首页输入用户和名密码
driver.find_element_by_xpath("//form[@id='loginForm']/ul/input[1]").send_keys("51zxw")
driver.find_element_by_xpath("//form[@id='loginForm']/ul/input[2]").send_keys("66666")

#逻辑运算组合定位
driver.find_element_by_xpath("//input[@class='loinp' and @name='username']").send_keys("51zxw")

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

css定位

鼠标操作

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

_ = webdriver.ChromeOptions()
_.add_argument("--start-maximized")
driver = webdriver.Chrome(options=_)
```

```
driver.get("https://www.baidu.com/")

ele_set = driver.find_element(By.XPATH, "//span[text()='设置']")
webdriver.ActionChains(driver).move_to_element(ele_set).perform()

# 单击，弹出的Ajax元素，根据链接节点的文本内容查找
driver.find_element(By.LINK_TEXT, '高级搜索').click()




input("结束符：")
driver.quit()

```

键盘操作

```
"""
node.send_keys(Keys.SPACE)
node.send_keys(Keys.CONTROL, 'a')
node.send_keys(Keys.CONTROL, 'c')
node.send_keys(Keys.CONTROL, 'v')
node.send_keys(Keys.ENTER)
# 案例： 在百度搜索关键词“Python” 然后将关键词复制或剪切到搜狗搜索框进行搜索
"""

import time

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

_ = webdriver.ChromeOptions()
_.add_argument("--start-maximized")
driver = webdriver.Chrome(options=_)

driver.get("http://www.baidu.com")
driver.find_element(By.CSS_SELECTOR, "#kw").send_keys("Python")

time.sleep(2)
# 键盘全选操作 Ctrl+A
driver.find_element(By.CSS_SELECTOR, "#kw").send_keys(webdriver.Keys.CONTROL, 'a')

# 键盘选择复制或剪切操作 Ctrl+C
driver.find_element(By.CSS_SELECTOR, "#kw").send_keys(webdriver.Keys.CONTROL, 'c')
driver.find_element(By.CSS_SELECTOR, "#kw").send_keys(webdriver.Keys.CONTROL, 'x')

# 打开搜狗页面
driver.get("http://www.sogou.com/")

time.sleep(2)
```

```
33
34  # 粘贴复制内容
35  driver.find_element(By.CSS_SELECTOR, ".sec-input").send_keys(webdriver.Keys.CONTROL,
    'v')
36  time.sleep(2)
37
38  # 点击搜索按钮
39  # driver.find_element_by_xpath("//input[@id='stb']").click()
40  driver.find_element(By.CSS_SELECTOR, "#stb").click()
41
42  input("结束符: ")
43  driver.quit()
44
```

切换页面

- **适用网站+应对方案**

```
1  【1】适用网站类型
2      页面中点开链接出现新的窗口，但是浏览器对象browser还是之前页面的对象，需要切换到不同的窗口进
   行操作
3
4  【2】应对方案 - browser.switch_to.window()
5
6      # 获取当前所有句柄（窗口）- [handle1,handle2]
7      all_handles = browser.window_handles
8      # 切换browser到新的窗口，获取新窗口的对象
9      browser.switch_to.window(all_handles[1])
```

iframe

- **特点+方法**

```
1  【1】特点
2      网页中嵌套了网页，先切换到iframe，然后再执行其他操作
3
4  【2】处理步骤
5      2.1) 切换到要处理的Frame
6      2.2) 在Frame中定位页面元素并进行操作
7      2.3) 返回当前处理的Frame的上一级页面或主页面
8
9  【3】常用方法
10     3.1) 切换到frame  -  browser.switch_to.frame(frame节点对象)
11     3.2) 返回上一级   -  browser.switch_to.parent_frame()
12     3.3) 返回主页面   -  browser.switch_to.default_content()
13
14 【4】使用说明
15     4.1) 方法一：默认支持id和name属性值 ： switch_to.frame(id属性值|name属性值)
16     4.2) 方法二：
```

```
        a> 先找到frame节点 : frame_node = browser.find_element_by_xpath('xxxx')
        b> 在切换到frame    : browser.switch_to.frame(frame_node)

【5】iframe子框架
    browser.switch_to.frame(iframe_element)
    # 写法1 - 任何场景都可以:
    iframe_node = browser.find_element_by_xpath('')
    browser.switch_to.frame(iframe_node)

    # 写法2 - 默认支持 id 和 name 两个属性值:
    browser.switch_to.frame('id属性值|name属性值')
```

- **示例1 - 登录豆瓣网**

```
"""
登录豆瓣网
"""
from selenium import webdriver
import time

# 打开豆瓣官网
browser = webdriver.Chrome()
browser.get('https://www.douban.com/')

# 切换到iframe子页面
login_frame = browser.find_element_by_xpath('//*[@id="anony-reg-new"]/div/div[1]/iframe')
browser.switch_to.frame(login_frame)

# 密码登录 + 用户名 + 密码 + 登录豆瓣
browser.find_element_by_xpath('/html/body/div[1]/div[1]/ul[1]/li[2]').click()
browser.find_element_by_xpath('//*[@id="username"]').send_keys('自己的用户名')
browser.find_element_by_xpath('//*[@id="password"]').send_keys('自己的密码')
browser.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/div[5]/a').click()
time.sleep(3)

# 点击我的豆瓣
browser.find_element_by_xpath('//*[@id="db-nav-sns"]/div/div/div[3]/ul/li[2]/a').click()
```

元素等待

```
"""
元素等待
概念
· 显示等待是针对某一个元素进行相关等待判定;
· 隐式等待不针对某一个元素进行等待，全局元素等待。只能判断元素是否可见
a.相关模块

□WebDriverWait 显示等待针对元素必用
```

```
□expected_conditions 预期条件类（里面包含方法可以调用，用于显示等待）
□NoSuchElementException 用于隐式等待抛出异常
□By 用于元素定位
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait          #注意字母大写
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import NoSuchElementException
"""
from time import sleep, ctime

from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as ec
from selenium.webdriver.common.by import By

# 显示等待
# 案例：检测百度页面搜索按钮是否存在，存在就输入关键词"自学网 Selenium" 然后点击搜索
_ = webdriver.ChromeOptions()
_.add_argument("--start-maximized")
driver = webdriver.Chrome(options=_)

driver.get("http://www.baidu.com")

driver.find_element(By.CSS_SELECTOR, "#kw").send_keys("自学网 Selenium")
sleep(2)

# 显示等待--判断搜索按钮是否存在
element = WebDriverWait(driver, 5, 0.5).until(ec.presence_of_element_located((By.ID,
"su")))
element.click()
sleep(3)

input("继续")

# 隐式等待
driver.get("http://www.baidu.com")
driver.implicitly_wait(5)  # 隐式等待时间设定 5秒
# 检测搜索框是否存在
try:
    print(ctime())
    driver.find_element(By.CSS_SELECTOR, "#kw").send_keys("Python")
    driver.find_element(By.CSS_SELECTOR, "#su").click()
except ec.NoSuchElementException as msg:
    print(msg)
finally:
    print(ctime())

input("结束")
driver.quit()

```

滚动条控制操作

```python
import time

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

_ = webdriver.ChromeOptions()
_.add_argument("--start-maximized")
driver = webdriver.Chrome(options=_)

driver.get("https://ke.qq.com/")

# 方式1
# target = driver.find_element(By.XPATH, "//h3[text()='为你推荐']")
# 将该模块与浏览器顶部对齐
# driver.execute_script('arguments[0].scrollIntoView();', target)
# 将该模块与浏览器底部对齐
# driver.execute_script('arguments[0].scrollIntoView(false);', target)

# 方式2
# 滚动到页面最底部
# driver.execute_script("window.scrollTo(0,document.body.scrollHeight)")
# time.sleep(3)
# 滚动到页面最顶部
# driver.execute_script("window.scrollTo(document.body.scrollHeight,0)")

# 方式3
# 将滚动调拖到最底部
# driver.execute_script("var action=document.documentElement.scrollTop=10000")
# time.sleep(3)
# 将滚动条拖到最顶部
# driver.execute_script("var action=document.documentElement.scrollTop=0")

# 方式4
# 滚动到页面最底部
driver.execute_script("window.scrollTo(0,document.documentElement.scrollHeight)")
time.sleep(3)
# 滚动到页面最顶部
driver.execute_script("window.scrollTo(document.documentElement.scrollHeight,0)")



input("结束符: ")
driver.quit()
```

网页截图操作

```python
# 案例：分别打开我要自学网页面和百度页面，然后进行截图
```

```
from selenium import webdriver
from time import sleep

#加载浏览器驱动
driver=webdriver.Firefox()

#打开自学网页面并截图
driver.get("http://www.51zxw.net")
driver.get_screenshot_as_file(r"E:\Python_script\51zxw.jpg")

#打开百度页面并截图
driver.get("http://www.baidu.com")
driver.get_screenshot_as_file(r"E:\Python_script\baidu.png")


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

上传文件

```
案例：在百度搜索上传本地图片进行搜索。
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()
driver.get("http://www.baidu.com")

driver.find_element_by_css_selector(".soutu-btn").click()
sleep(3)
driver.find_element_by_css_selector(".upload-
pic").send_keys(r"E:\Python_script\Webdriver\shuiyin.png")

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

Cookie处理

- 什么是Cookie

Cookie是储存在用户本地终端上的数据，实际上是一小段的文本信息。

- Cookie作用

帮助 Web 站点保存有关访问者的信息，方便用户的访问。如记住用户名密码实现自动登录。

```
# 案例：查看访问我要自学网时的Cookie内容

from selenium import webdriver
```

```
4
5  driver=webdriver.Firefox()
6  driver.get("http://www.51zxw.net/")
7
8  #获取cookie全部内容
9  cookie=driver.get_cookies()
10 #打印全部cookile信息
11 print(cookie)
12 #打印cookie第一组信息
13 print(cookie[0])
14
15 #添加cookie
16 driver.add_cookie({'name':'51zxw','value':'www.51zxw.net'})
17 for cookie in driver.get_cookies():
18     print("%s --- %s" %(cookie['name'],cookie['value']))
19
20 driver.quit()
```

自动化测试验证码问题

验证码作用

不少网站在用户登录、用户提交信息等登录和输入的页面上使用了验证码技术。验证码技术可以有效防止恶意用户对网站的滥用，使得网站可以有效避免用户信息失窃、保证网站稳定安全性。

但是验证码给自动化测试带来一些不便，使脚本无法正常运行覆盖功能模块。

如何解决

1.去掉验证码

这是最简单的方法，对于开发人员来说，只是把验证码的相关代码注释掉即可，如果是在测试环境，这样做可省去了测试人员不少麻烦，如果自动化脚本是要在正式环境跑，这样就给系统带来了一定的风险。

2.设置万能码

去掉验证码的主要是安全问题，为了应对在线系统的安全性威胁，可以在修改程序时不取消验证码，而是程序中留一个“后门”---设置一个“万能验证码”，只要用户输入这个“万能验证码”，程序就认为验证通过，否则按照原先的验证方式进行验证。

3.验证码识别技术（OCR）

例如可以通过Python-tesseract 来识别图片验证码，Python-tesseract是光学字符识别Tesseract OCR引擎的Python封装类。能够读取任何常规的图片文件(JPG, GIF ,PNG , TIFF等)。不过，目前市面上的验证码形式繁多，目前任何一种验证码识别技术，识别率都不是100% 。

4.记录cookie

通过向浏览器中添加cookie 可以绕过登录的验证码。

基于Cookie绕过验证码自动登录

```
# 案例：使用Cookie绕过百度验证码自动登录账户。
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()
driver.get("http://www.baidu.com/")

#手动添加cookie
driver.add_cookie({'name':'BAIDUID','value':'9E4BF1D44014…(根据实际获取值填写)'})
driver.add_cookie({'name':'BDUSS','value':'根据实际抓包获取值填写'})

sleep(2)
driver.refresh()
sleep(3)
driver.quit()
```

元素定位不到的几种场景

```
# 1 元素没有加载完成

# 2 iframe

# 3 元素不可用，不可读，不可见

# 4 动态属性
冻结窗口
setTimeout(function(){debugger}, 5000)
```