

## base\_page

```
1  """
2  基础对象
3  """
4  import json
5  import os.path as path
6
7  from selenium import webdriver
8
9  from stub.config.settings import BASE_DIR
10
11
12  def browser():
13      _ = webdriver.ChromeOptions()
14      # 无头模式
15      # _.headless = True
16      _.add_argument("--start-maximized")
17      # _.add_argument('--no-sandbox --start-maximized')
18      driver = webdriver.Chrome(options=_)
19      driver.implicitly_wait(10)
20      return driver
21
22
23  class BasePage:
24      def __init__(self, d):
25          self.driver: webdriver.Chrome = d
26
27      def quit(self):
28          input("输入任意字符退出: ")
29          self.driver.quit()
30
31      def cookies_login(self):
32          with open(path.join(BASE_DIR, "lib", "cookies.json"), "r") as f:
33              for i in json.load(f):
34                  self.driver.add_cookie(i)
35
```

## webdriver

```
1  from selenium import webdriver
2  driver = webdriver.Chrome()
3  # 地址栏输入url地址并确认
4  driver.get(url=url)
5
```

```
6 # 关闭当前页
7 driver.close()
8
9 # 浏览器窗口最大化
10 driver.maximize_window()
11
12 # 设置当前窗口的宽度和高度。
13 driver.set_window_size()
14
15 # 设置当前窗口的x、y位置。
16 driver.set_window_position()
17
18 # browser执行JS脚本
19 driver.execute_script('window.scrollTo(0,document.body.scrollHeight)')
20
21 driver.get('https://www.baidu.com')
22 driver.get('https://news.baidu.com')
23 # 后退
24 driver.back()
25
26 # 前进
27 driver.forward()
28
29 # 刷新当前页面。
30 driver.refresh()
31
32 # 退出驱动程序并关闭浏览器
33 driver.quit()
34
35 # HTML结构源码
36 driver.page_source
37 # 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
38 driver.page_source.find('字符串')
39
40 # 返回当前页面的标题。
41 driver.title
42
43 # 获取当前页面的URL。
44 driver.current_url
45
46 # 隐式等待
47 driver.implicitly_wait()
48
49 # 获取当前句柄
50 driver.current_window_handle
51
52 # 所有句柄
53 driver.window_handles
54
55 driver.switch_to
```

# WebElement

```
1  # 点击元素。
2  click()
3
4  # 获取元素的给定属性或属性。
5  get_attribute()
6
7  # 元素是否对用户可见。
8  is_displayed()
9
10 # 返回元素是否可用。
11 is_enabled()
12
13 # 返回元素是否被选中。
14 is_selected()
15
16 # 元素的大小。
17 size
18
19 # 元素的文本。
20 text
21
22 【1】文本框操作
23     1.1) node.send_keys('') - 向文本框发送内容
24     1.2) node.clear() - 清空文本
25     1.3) node.get_attribute('value') - 获取文本内容
26
27 【2】按钮操作
28     1.1) node.click() - 点击
29     1.2) node.is_enabled() - 判断按钮是否可用
30     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

```
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3  from time import sleep
4
5  def demo_id_and_name():
6      _ = webdriver.ChromeOptions()
7      _.add_argument("--start-maximized")
8      driver = webdriver.Chrome(options=_)
```

```
9     driver.get("http://www.baidu.com")
10
11     driver.find_element(By.ID, "kw").send_keys("Selenium我要自学网")
12     driver.find_element(By.NAME, "wd").send_keys("Selenium我要自学网")
13
14     sleep(2)
15     driver.find_element(By.ID, "su").click()
16     driver.close()
17
18
```

## tag\_name定位

---

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

## class\_name定位

---

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

## link\_text与partial\_link\_text定位

---

```

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

```

## XPath定位

### xpath解析

- 定义

1 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 【1】查找所有的dd节点
2     //dd
3 【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
4     //p[@class="name"]/a
5 【3】获取dl节点下第2个dd节点的电影节点
6     //dl[@class="board-wrapper"]/dd[2]
7 【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
8     //p[@class="name"]/a/@href
9
10 【注意】
11     1> 只要涉及到条件,加 [] : //dl[@class="xxx"] //dl/dd[2]
12     2> 只要获取属性值,加 @ : //dl[@class="xxx"] //p/a/@href

```

- 选取节点

```

1 【1】// : 从所有节点中查找（包括子节点和后代节点）
2 【2】@ : 获取属性值
3     2.1> 使用场景1（属性值作为条件）
4         //div[@class="movie-item-info"]
5     2.2> 使用场景2（直接获取属性值）
6         //div[@class="movie-item-info"]/a/img/@src
7
8 【3】练习 - 猫眼电影top100
9     3.1> 匹配电影名称
10        //div[@class="movie-item-info"]/p[1]/a/@title
11     3.2> 匹配电影主演
12        //div[@class="movie-item-info"]/p[2]/text()

```

```

13 3.3> 匹配上映时间
14      //div[@class="movie-item-info"]/p[3]/text()
15 3.4> 匹配电影链接
16      //div[@class="movie-item-info"]/p[1]/a/@href

```

## • 匹配多路径 (或)

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

## • 常用函数

```

1 【1】text() : 获取节点的文本内容
2     xpath表达式末尾不加 /text() : 则得到的结果为节点对象
3     xpath表达式末尾加 /text() 或者 /@href : 则得到结果为字符串
4
5 【2】contains() : 匹配属性值中包含某些字符串节点
6     匹配class属性值中包含 'movie-item' 这个字符串的 div 节点
7     //div[contains(@class,"movie-item")]

```

## • 终极总结

```

1 【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
3 【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'
4     [<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]
5     [<element div at xxxa>,<element div at xxxb>]
6     [<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]

```

## • 课堂练习

```

1 【1】匹配汽车之家-二手车,所有汽车的链接 :
2     //li[@class="cards-li list-photo-li"]/a[1]/@href
3     //a[@class="carinfo"]/@href
4 【2】匹配汽车之家-汽车详情页中,汽车的
5     2.1)名称: //div[@class="car-box"]/h3/text()
6     2.2)里程: //ul/li[1]/h4/text()
7     2.3)时间: //ul/li[2]/h4/text()
8     2.4)挡位+排量: //ul/li[3]/h4/text()
9     2.5)所在地: //ul/li[4]/h4/text()
10    2.6)价格: //div[@class="brand-price-item"]/span[@class="price"]/text()

```

```

1 xpath绝对与相对定位
2 from selenium import webdriver
3 from time import sleep
4
5 driver=webdriver.Firefox()
6
7 driver.get("http://www.baidu.com")
8
9 # 绝对路径定位

```

```

10 driver.find_element_by_xpath("/html/body/div[2]/div[1]/div/div[1]/div/form/span[1]/input").send_keys("51zxw")
11
12 # 利用元素熟悉定位--定位到input标签中为kw的元素
13 driver.find_element_by_xpath("//input[@id='kw']").send_keys("Selenium")
14
15 # 定位input标签中name属性为wd的元素
16 driver.find_element_by_xpath("//input[@name='wd']").send_keys("Selenium")
17
18 # 定位所有标签元素中, class属性为s_ipt的元素
19 driver.find_element_by_xpath("//*[@class='s_ipt']").send_keys("Python3")
20
21 driver.find_element_by_id('su').click()
22
23 sleep(3)
24 driver.quit()
25
26
27 Xpath层级与逻辑定位
28 from selenium import webdriver
29 from time import sleep
30
31 driver=webdriver.Firefox()
32
33 driver.get("http://www.51zxw.net/")
34 #层级和属性结合定位--自学网首页输入用户和名密码
35 driver.find_element_by_xpath("//form[@id='loginForm']/ul/input[1]").send_keys("51zxw")
36 driver.find_element_by_xpath("//form[@id='loginForm']/ul/input[2]").send_keys("66666")
37
38 #逻辑运算组合定位
39 driver.find_element_by_xpath("//input[@class='loinp' and @name='username']").send_keys("51zxw")
40
41 sleep(3)
42 driver.quit()

```

## css定位

## 鼠标操作

```

1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3
4 _ = webdriver.ChromeOptions()
5 _.add_argument("--start-maximized")
6 driver = webdriver.Chrome(options=_)

```

```

7 driver.get("https://www.baidu.com/")
8
9 ele_set = driver.find_element(By.XPATH, "//span[text()='设置']")
10 webdriver.ActionChains(driver).move_to_element(ele_set).perform()
11
12 # 单击, 弹出的Ajax元素, 根据链接节点的文本内容查找
13 driver.find_element(By.LINK_TEXT, '高级搜索').click()
14
15
16
17
18
19 input("结束符: ")
20 driver.quit()
21

```

## 键盘操作

```

1 """
2 node.send_keys(Keys.SPACE)
3 node.send_keys(Keys.CONTROL, 'a')
4 node.send_keys(Keys.CONTROL, 'c')
5 node.send_keys(Keys.CONTROL, 'v')
6 node.send_keys(Keys.ENTER)
7 # 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
8 """
9
10 import time
11
12 from selenium import webdriver
13 from selenium.webdriver.common.by import By
14
15 _ = webdriver.ChromeOptions()
16 _.add_argument("--start-maximized")
17 driver = webdriver.Chrome(options=_)
18
19 driver.get("http://www.baidu.com")
20 driver.find_element(By.CSS_SELECTOR, "#kw").send_keys("Python")
21
22 time.sleep(2)
23 # 键盘全选操作 Ctrl+A
24 driver.find_element(By.CSS_SELECTOR, "#kw").send_keys(webdriver.Keys.CONTROL, 'a')
25
26 # 键盘选择复制或剪切操作 Ctrl+C
27 driver.find_element(By.CSS_SELECTOR, "#kw").send_keys(webdriver.Keys.CONTROL, 'c')
28 driver.find_element(By.CSS_SELECTOR, "#kw").send_keys(webdriver.Keys.CONTROL, 'x')
29
30 # 打开搜狗页面
31 driver.get("http://www.sogou.com/")
32 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,
36 'v')
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：

```

```

17         a> 先找到frame节点 : frame_node = browser.find_element_by_xpath('xxxx')
18         b> 在切换到frame : browser.switch_to.frame(frame_node)
19
20     【5】iframe子框架
21         browser.switch_to.frame(iframe_element)
22         # 写法1 - 任何场景都可以:
23         iframe_node = browser.find_element_by_xpath('')
24         browser.switch_to.frame(iframe_node)
25
26         # 写法2 - 默认支持 id 和 name 两个属性值:
27         browser.switch_to.frame('id属性值|name属性值')

```

## • 示例1 - 登录豆瓣网

```

1  """
2  登录豆瓣网
3  """
4  from selenium import webdriver
5  import time
6
7  # 打开豆瓣官网
8  browser = webdriver.Chrome()
9  browser.get('https://www.douban.com/')
10
11 # 切换到iframe子页面
12 login_frame = browser.find_element_by_xpath('//*[id="anony-reg-
new"]/div/div[1]/iframe')
13 browser.switch_to.frame(login_frame)
14
15 # 密码登录 + 用户名 + 密码 + 登录豆瓣
16 browser.find_element_by_xpath('/html/body/div[1]/div[1]/ul[1]/li[2]').click()
17 browser.find_element_by_xpath('//*[id="username"]').send_keys('自己的用户名')
18 browser.find_element_by_xpath('//*[id="password"]').send_keys('自己的密码')
19 browser.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/div[5]/a').click()
20 time.sleep(3)
21
22 # 点击我的豆瓣
23 browser.find_element_by_xpath('//*[id="db-nav-
sns"]/div/div/div[3]/ul/li[2]/a').click()

```

# 元素等待

```

1  """
2  元素等待
3  概念
4  · 显示等待是针对某一个元素进行相关等待判定;
5  · 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
6  a. 相关模块
7
8  □WebDriverWait 显示等待针对元素必用

```

```

8  □expected_conditions 预期条件类（里面包含方法可以调用，用于显示等待）
9  □NoSuchElementException 用于隐式等待抛出异常
10 □By 用于元素定位
11 from selenium import webdriver
12 from selenium.webdriver.common.by import By
13 from selenium.webdriver.support.ui import WebDriverWait          #注意字母大写
14 from selenium.webdriver.support import expected_conditions as EC
15 from selenium.common.exceptions import NoSuchElementException
16 """
17 from time import sleep, ctime
18
19 from selenium import webdriver
20 from selenium.webdriver.support.ui import WebDriverWait
21 from selenium.webdriver.support import expected_conditions as ec
22 from selenium.webdriver.common.by import By
23
24 # 显示等待
25 # 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
26 _ = webdriver.ChromeOptions()
27 _.add_argument("--start-maximized")
28 driver = webdriver.Chrome(options=_)
29
30 driver.get("http://www.baidu.com")
31
32 driver.find_element(By.CSS_SELECTOR, "#kw").send_keys("自学网 Selenium")
33 sleep(2)
34
35 # 显示等待--判断搜索按钮是否存在
36 element = WebDriverWait(driver, 5, 0.5).until(ec.presence_of_element_located((By.ID,
37 "su"))))
38 element.click()
39 sleep(3)
40 input("继续")
41
42 # 隐式等待
43 driver.get("http://www.baidu.com")
44 driver.implicitly_wait(5) # 隐式等待时间设定 5秒
45 # 检测搜索框是否存在
46 try:
47     print(ctime())
48     driver.find_element(By.CSS_SELECTOR, "#kw").send_keys("Python")
49     driver.find_element(By.CSS_SELECTOR, "#su").click()
50 except ec.NoSuchElementException as msg:
51     print(msg)
52 finally:
53     print(ctime())
54
55 input("结束")
56 driver.quit()
57

```

# 滚动条控制操作

```
1 import time
2
3 from selenium import webdriver
4 from selenium.webdriver.common.by import By
5
6 _ = webdriver.ChromeOptions()
7 _.add_argument("--start-maximized")
8 driver = webdriver.Chrome(options=_)
9
10 driver.get("https://ke.qq.com/")
11
12 # 方式1
13 # target = driver.find_element(By.XPATH, "//h3[text()='为你推荐']")
14 # 将该模块与浏览器顶部对齐
15 # driver.execute_script('arguments[0].scrollIntoView();', target)
16 # 将该模块与浏览器底部对齐
17 # driver.execute_script('arguments[0].scrollIntoView(false);', target)
18
19 # 方式2
20 # 滚动到页面最底部
21 # driver.execute_script("window.scrollTo(0,document.body.scrollHeight)")
22 # time.sleep(3)
23 # 滚动到页面最顶部
24 # driver.execute_script("window.scrollTo(document.body.scrollHeight,0)")
25
26 # 方式3
27 # 将滚动条拖到最底部
28 # driver.execute_script("var action=document.documentElement.scrollTop=10000")
29 # time.sleep(3)
30 # 将滚动条拖到最顶部
31 # driver.execute_script("var action=document.documentElement.scrollTop=0")
32
33 # 方式4
34 # 滚动到页面最底部
35 driver.execute_script("window.scrollTo(0,document.documentElement.scrollHeight)")
36 time.sleep(3)
37 # 滚动到页面最顶部
38 driver.execute_script("window.scrollTo(document.documentElement.scrollHeight,0)")
39
40
41
42 input("结束符: ")
43 driver.quit()
```

# 网页截图操作

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

```

2  from selenium import webdriver
3  from time import sleep
4
5  #加载浏览器驱动
6  driver=webdriver.Firefox()
7
8  #打开自学网页面并截图
9  driver.get("http://www.51zxw.net")
10 driver.get_screenshot_as_file(r"E:\Python_script\51zxw.jpg")
11
12 #打开百度页面并截图
13 driver.get("http://www.baidu.com")
14 driver.get_screenshot_as_file(r"E:\Python_script\baidu.png")
15
16
17 sleep(2)
18 driver.quit()

```

## 上传文件

```

1  案例：在百度搜索上传本地图片进行搜索。
2  from selenium import webdriver
3  from time import sleep
4
5  driver=webdriver.Firefox()
6  driver.get("http://www.baidu.com")
7
8  driver.find_element_by_css_selector(".soutu-btn").click()
9  sleep(3)
10 driver.find_element_by_css_selector(".upload-
    pic").send_keys(r"E:\Python_script\Webdriver\shuiyin.png")
11
12 sleep(3)
13 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。

```

1  # 案例：查看访问我要自学网时的Cookie内容
2
3  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cookie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

```
1  # 案例：使用Cookie绕过百度验证码自动登录账户。
2  from selenium import webdriver
3  from time import sleep
4
5  driver=webdriver.Firefox()
6  driver.get("http://www.baidu.com/")
7
8  #手动添加cookie
9  driver.add_cookie({'name':'BAIDUID','value':'9E4BF1D44014... (根据实际获取值填写) '})
10 driver.add_cookie({'name':'BDUSS','value':'根据实际抓包获取值填写'})
11
12 sleep(2)
13 driver.refresh()
14 sleep(3)
15 driver.quit()
```

## 元素定位不到的几种场景

```
1  # 1 元素没有加载完成
2
3  # 2 iframe
4
5  # 3 元素不可用，不可读，不可见
6
7  # 4 动态属性
8 冻结窗口
9  setTimeout(function(){debugger}, 5000)
10
11 debugger
12 快捷键F8 or ctrl+\
```