Selenium基础

- 讲师：pansir

基本使用

- 安装：`pip install selenium`

selenium驱动chrome完成源代码获取

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait
# 声明浏览器对象
browser = webdriver.Chrome()
try:
    # 打开网站
    browser.get('https://www.baidu.com')
    # 定位元素
    input = browser.find_element_by_id('kw')
    # 往元素输入python
    input.send_keys('Python')
    # 输入回车
    input.send_keys(Keys.ENTER)
    # 等待时间
    wait = WebDriverWait(browser, 10)
    # 判断等待结果是否出现
    wait.until(EC.presence_of_element_located((By.ID, 'content_left')))
    # 打印url
    print(browser.current_url)
    # 打印cookies
    print(browser.get_cookies())
    # 打印源代码
    print(browser.page_source)
finally:
    # 关闭浏览器
    browser.close()
```

- 思考：和直接调接口请求，有什么区别?

selenium可以直接拿到JavaScript渲染的结果，不用担心加密系统（token认证/反爬虫等）

- tpis：如果打不开浏览器，提示正在下载代理

设置——高级——打开代理设置——局域网设置——取消勾选自动检测设置——重启浏览器

声明浏览器对象

```
from selenium import webdriver
# 支持多种浏览器对象
browser = webdriver.Chrome()
browser = webdriver.Firefox()
browser = webdriver.Edge()
# 无界面浏览器
browser = webdriver.PhantomJS()
browser = webdriver.Safari()
```

访问页面

使用brower对象淘宝网站

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
print(browser.page_source)
browser.close()
```

查找元素

单个元素

语法规则：find_elemet_by...

- 查看淘宝输入框的html代码

```
<input id="q" name="q" aria-label="请输入搜索文字" accesskey="s"
autofocus="autofocus" autocomplete="off" class="search-combobox-input" aria-
haspopup="true" aria-combobox="list" role="combobox" x-webkit-
grammar="builtin:translate" tabindex="0">
```

可以看到对应属性值：id=q，name=q，可以通过多种方式获取input对象

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
# 下列功能一样，获取的input对象一样
# 1. 通过id查找，注意淘宝版本，有的是mq
input1 = browser.find_element_by_id('q')
print(input1)
# 2. 通过name查找
```

```
input2 = browser.find_element_by_name('q')
print(input2)
# 3. 通过css选择，# 表示id
input3 = browser.find_element_by_css_selector('#q')
print(input3)
# 4. 通过xpath查找
input4 = browser.find_element_by_xpath('//*[@id="q"]')
print(input4)

browser.close()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="92b36cc2591d930dc7c6cf343d11a0f3", element="0.19743756873249652-1")>
<selenium.webdriver.remote.webelement.WebElement
(session="92b36cc2591d930dc7c6cf343d11a0f3", element="0.19743756873249652-1")>
<selenium.webdriver.remote.webelement.WebElement
(session="92b36cc2591d930dc7c6cf343d11a0f3", element="0.19743756873249652-1")>
<selenium.webdriver.remote.webelement.WebElement
(session="92b36cc2591d930dc7c6cf343d11a0f3", element="0.19743756873249652-1")>
```

- 也可以使用By对象+find_element()来取值，效果一样

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
# By.类型  方式传参，直接把by当参数，等同于上面
input_first = browser.find_element(By.ID, 'q')
print(input_first)
browser.close()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="adc0186d7bfc49fd5de189379e5874f3", element="0.6708129166600252-1")>
```

小结

- find_element_by_id：通过id查找
- find_element_by_name：通过name值查找
- find_element_by_xpath：通过xpath查找
- find_element_by_link_text：通过链接内容查找
- find_element_by_partial_link_text：通过链接内容查找
- find_element_by_tag_name：根据标签的名字查找，页面重复标签太多，不实用，一般调试的时候使用
- find_element_by_class_name：根据class名称查找，class="subtitle"
- find_element_by_css_selector：通过css选择器查找

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
# 1. 通过链接查找
# login = browser.find_element_by_link_text("亲，请登录").click()
# 2. 通过链接查找
# a = browser.find_element_by_link_text("家电").click()
# 2.1 如果超链接本身文字很长，可以输入部分文字进行模糊匹配
# a = browser.find_element_by_partial_link_text("猫").click()
# 3. 通过css名称查找元素，并打印内容
# b= browser.find_elements_by_class_name("subtitle")
# for i in b:
#     print(i.text)
# 4. 通过css选择器
# c = browser.find_elements_by_css_selector("#q")  # 等同与id选择器
# print(c)
# # c = browser.find_elements_by_css_selector("div h4")
# for i in c:
#     print(i.text)
browser.close()
```

```
None
```

- 思考：为什么有时候有元素，有时候没有?

因为页面加载元素需要时间，若在页面加载之前就进行打印，就会获取不到对应数据

cssSelector常用定位方法

css是selenium官网推荐的定位方法，而不是xpath，因为css比xpath更快，解析效率更高。前端页面每个元素都有样式，所以理论上，通过css一定能定位到

- css选择器

选择器	例子	例子描述
[*.class*]	.intro	选择 class="intro" 的所有元素。
[*#id*]	#firstname	选择 id="firstname" 的所有元素。
[*]	*	选择所有元素。
[*element*]	p	选择所有 元素。
[*element*,*element*]	div,p	选择所有 元素和所有 元素。

[*element element*]	div p	选择 元素内部的所有 元素。
[*element>element*]	div>p	选择父元素为 元素的所有 元素。
[*element+element*]	div+p	选择紧接在 元素之后的所有 元素。
[[*attribute*]]	[target]	选择带有 target 属性所有元素。
[[*attribute=value*]]	[target=_blank]	选择 target="_blank" 的所有元素。
[[*attribute~=value*]]	[title~=flower]	选择 title 属性包含单词 "flower" 的所有元素。
[[*attribute|=value*]]	[lang|=en]	选择 lang 属性值以 "en" 开头的所有元素。
[:link]	a:link	选择所有未被访问的链接。
[:visited]	a:visited	选择所有已被访问的链接。
[:active]	a:active	选择活动链接。
[:hover]	a:hover	选择鼠标指针位于其上的链接。
[:focus]	input:focus	选择获得焦点的 input 元素。
[:first-letter]	p:first-letter	选择每个 元素的首字母。
[:first-line]	p:first-line	选择每个 元素的首行。
[:first-child]	p:first-child	选择属于父元素的第一个子元素的每个 元素。
[:before]	p:before	在每个 元素的内容之前插入内容。
[:after]	p:after	在每个 元素的内容之后插入内容。
[:lang(*language*)]	p:lang(it)	选择带有以 "it" 开头的 lang 属性值的每个 元素。
[*element1~element2*]	p~ul	选择前面有 元素的每个 元素。

[[*attribute*^=*value*]]	a[src^="https"]	选择其 src 属性值以 "https" 开头的每个 元素。
[[*attribute*$=*value*]]	a[src$=".pdf"]	选择其 src 属性以 ".pdf" 结尾的所有 元素。
[[*attribute***=value*]]	a[src*="abc"]	选择其 src 属性中包含 "abc" 子串的每个 元素。
[:first-of-type]	p:first-of-type	选择属于其父元素的首个 元素的每个 元素。
[:last-of-type]	p:last-of-type	选择属于其父元素的最后 元素的每个 元素。
[:only-of-type]	p:only-of-type	选择属于其父元素唯一的 元素的每个 元素。
[:only-child]	p:only-child	选择属于其父元素的唯一子元素的每个 元素。
[:nth-child(*n*)]	p:nth-child(2)	选择属于其父元素的第二个子元素的每个 元素。
[:nth-last-child(*n*)]	p:nth-last-child(2)	同上，从最后一个子元素开始计数。
[:nth-of-type(*n*)]	p:nth-of-type(2)	选择属于其父元素第二个 元素的每个 元素。
[:nth-last-of-type(*n*)]	p:nth-last-of-type(2)	同上，但是从最后一个子元素开始计数。
[:last-child]	p:last-child	选择属于其父元素最后一个子元素每个 元素。
[:root]	:root	选择文档的根元素。
[:empty]	p:empty	选择没有子元素的每个 元素（包括文本节点）。
[:target]	#news:target	选择当前活动的 #news 元素。
[:enabled]	input:enabled	选择每个启用的 元素。
[:disabled]	input:disabled	选择每个禁用的 元素
[:checked]	input:checked	选择每个被选中的 元素。
[:not(*selector*)]	:not(p)	选择非 元素的每个元素。

[::selection]	::selection	选择被用户选取的元素部分。

- 基本常用定位

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 通过tagname
# a = browser.find_element(By.CSS_SELECTOR, "input")
# 2. 通过ID
# a = browser.find_element(By.CSS_SELECTOR, "#q")
# 3. 通过classname
# a = browser.find_element(By.CSS_SELECTOR, ".service-bd")
# 4. 通过标签+classname组合，或者标签+id组合
# a = browser.find_element(By.CSS_SELECTOR, "input.search-combobox-input")
# 5. 多个classname组合J_TbSearchContent J_HotWord
a = browser.find_element(By.CSS_SELECTOR, ".J_TbSearchContent.J_HotWord")

print(a)
browser.close()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="7fe999e183e49e6e47212544d1c0b671", element="0.9479495615475186-1")>
```

- 精准匹配

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 标签下，属性名=属性值
# a = browser.find_element(By.CSS_SELECTOR, "input[name=q]")
# a = browser.find_element(By.CSS_SELECTOR, "input[id=q]")
# a = browser.find_element(By.CSS_SELECTOR, "input[class=search-combobox-
input]")
# 2. 标签下，包含某个属性
# a = browser.find_element(By.CSS_SELECTOR, "img[alt]")
# 3. 标签下，多属性复合
# a = browser.find_element(By.CSS_SELECTOR, "input[name=q][id=q][class=search-
combobox-input]")

print(a)
```

```
browser.quit()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="faf52548677590c417940880929e446b", element="0.8706662121167064-1")>
```

- 模糊匹配

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 匹配 sea开头的
# a = browser.find_element(By.CSS_SELECTOR, "input[class ^='sea']")
# 2. 匹配 input结尾的
# a = browser.find_element(By.CSS_SELECTOR, "input[class $='input']")
# 3. 匹配 中间包含：box
a = browser.find_element(By.CSS_SELECTOR, "input[class *='box']")

print(a)
browser.quit()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="aa1f62f634fbf7cf396059c8009ab9c8", element="0.594004449976117-1")>
```

- 查找子元素

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 子元素，一级一级查找，一般是直接关系
# a = browser.find_element(By.CSS_SELECTOR, "div>div>input")
# 2. 子孙元素
# a = browser.find_element(By.CSS_SELECTOR, "form input")
# 3. 第一个子元素
# a = browser.find_element(By.CSS_SELECTOR, "form div:first-child")
# print(a.text)
# 4. 最后一个子元素
# a = browser.find_element(By.CSS_SELECTOR, "form div:last-child")
# print(a.text)  # 跟上面不一样，为空
# 5. 第二个子元素
# a = browser.find_element(By.CSS_SELECTOR, "form div:nth-child(2)")
# print(a.text)  # 跟上面不一样，是个方框（放大镜）
```

```
# 5. 找到div，再找它的兄弟元素
a = browser.find_element(By.CSS_SELECTOR, "form div+div")
print(a.text)  # 是个方框
print(a)
browser.quit()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="6e502d8771c2809d0857058b04b52df3", element="0.9297817884435187-1")>
```

css定位(同上)

跟上面效果一样，方法不一样

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 属性定位
# a = browser.find_element_by_css_selector("#q")  # id
# a = browser.find_element_by_css_selector(".search-combobox-input") # class
# a = browser.find_element_by_css_selector("input") # 标签
# 2. 其他属性
# a = browser.find_element_by_css_selector("[name='q']")
# a = browser.find_element_by_css_selector("[type='submit']")
# 3. 标签组合定位
# a = browser.find_element_by_css_selector("input.search-combobox-input")   #
标签+class
# a = browser.find_element_by_css_selector("input#q")   # 标签+id
# a = browser.find_element_by_css_selector("input[id='q']").send_keys("hello")
  # 标签+其他属性

# 4. 层级定位
# a = browser.find_element_by_css_selector("div>div>input").send_keys("hello")
# a = browser.find_element_by_css_selector("div input").send_keys("hello")
# print(a.text)  # 跟上面不一样，为空
# 5. css索引
# a = browser.find_element_by_css_selector("div input:nth-child(1)")
# print(a)
# 6. css逻辑运算
a = browser.find_element_by_css_selector("input[id='q']
[name='q']").send_keys("hello")

time.sleep(1)
browser.quit()
```

xpath

直接使用工具进行xpath定位，比较死板，复制粘贴后经常会定位不到。这时候需要自己写xpath（xml路径语言）

- xpath基本属性定位

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 使用id，*代表所有标签
# a = browser.find_element_by_xpath("//*[@id='q']")
# 2. 使用name
# a = browser.find_element_by_xpath("//*[@name='q']")
# 3. 使用class
# a = browser.find_element_by_xpath("//*[@class='search-combobox-input']")
# 4. 还可以使用其他属性：role="combobox"
# a = browser.find_element_by_xpath("//*[@role='combobox']")
# 5. 如果同个属性，同名的比较多，也可以指定标签，会更准确
# a = browser.find_element_by_xpath("//input[@role='combobox']")
print(a)
browser.quit()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="0ba1dcc417fa65a4cfc887a3264cefc1", element="0.01880901604800367-1")>
```

- xpath层级与索引

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 层级关系定位：如果某个标签的属性不明显，我们可以通过它的父节点开始找
# a = browser.find_element_by_xpath("//form[@name='search']/input")
# 2. 测试发现有多个input
# a = browser.find_elements_by_xpath("//form[@name='search']/input")
# 通过索引找到对应的某个标签，注意索引从1开始
a = browser.find_element_by_xpath("//form[@name='search']/input[1]")
print(a)
browser.quit()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="2218eea4374c5d05d41de9a2841c5f65", element="0.9679572025988938-1")>
```

- xpath逻辑运算：这个功能比较强大，支持and/or/not

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 一般用的比较多的是and
a = browser.find_element_by_xpath("//*[@id='q' and @class='search-combobox-
input']")

print(a)
browser.quit()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="55eedd4f8ddc872d478899b809348830", element="0.6352764022340274-1")>
```

- xpath模糊匹配，这个是要是掌握了，基本没有定位不到的元素

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')

# 1. 通过内容模糊匹配
# a = browser.find_element_by_xpath("//*[contains(text(), '聚划算')]").click()
# 2. 模糊匹配某个属性 class="btn-search tb-bg"
# a = browser.find_element_by_xpath("//*[contains(@class, 'btn-
search')]").click()
# 3. 模糊匹配开头部分
# a = browser.find_element_by_xpath("//*[starts-with(@class, 'btn')]").click()

# 4. 模糊匹配结尾部分，注意：只有xpath2.0才支持，一般浏览器只支持xpath1.0
# 2.0 //div[ends-with(@tagname, 'Destination')]
# 1.0 //div[substring(@tagname,string-length(@tagname) -string-
length('Destination') +1) = 'Destination']
# a = browser.find_element_by_xpath("//*[ends-with(@class, 'bg')]")
# a = browser.find_element_by_xpath("//div[substring(@class,string-
length(@class) -string-length('b-bg') +1) = 'b-bg']")
# print(a.text)

# 5. 正则表达式，同样只有2.0版本可以用
```

```
# a = browser.find_elements_by_xpath("//*[matchs(text(), '搜索')]")

browser.quit()
```

多个元素

- 查找多个元素：`find_elements_by...` 注意，这里多了个s

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
# 查找class=service-bd 下的li标签，即淘宝主题市场
# lis = browser.find_elements_by_css_selector('.service-bd li')
# print(lis)
lis2 = browser.find_elements_by_css_selector(".conve-list li")
print(lis2)
# 获取内容
# for i in lis2:
#     print(i.text)
browser.close()
```

- 练习：获取有好货下面的商品名称

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
# 查找class=service-bd 下的li标签，即淘宝主题市场
# lis = browser.find_elements_by_css_selector('.goods-inner ul .info h4')
time.sleep(5)
# 手动把页面往下滑一下
lis = browser.find_elements_by_css_selector('.info h4')
print(lis)
for i in lis:
    print(i.text)
browser.close()
```

- 使用By对象+find_elements()来定位，效果一样，传参更灵活

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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
# 通过By方式传参数
lis = browser.find_elements(By.CSS_SELECTOR, '.service-bd li')
print(lis)
browser.close()
```

- 小结：所谓8大定位方法

- find_elements_by_id：优先选择，速度最快
- find_elements_by_name：优先选择，速度最快
- find_elements_by_xpath：备选，效率最低
- find_elements_by_link_text：备选
- find_elements_by_partial_link_text：备选
- find_elements_by_tag_name：基本不用
- find_elements_by_class_name：常用
- find_elements_by_css_selector：常用

元素交互操作

- 对获取的元素调用交互方法，常用：`send_keys()`，`clear()`，`click()`

```
from selenium import webdriver
import time

browser = webdriver.Chrome()
browser.get('https://www.taobao.com')
# 找到输入框
input = browser.find_element_by_id('q')
# 输入iPohone
input.send_keys('iPhone')
# 等待1秒
time.sleep(1)
# 清空输入框
input.clear()
# 再输入iPoad
input.send_keys('iPad')
# 定位button
# button = browser.find_element_by_class_name('btn-search')
button = browser.find_element_by_css_selector('.btn-search')
# 按下按钮
button.click()
```

- 提交表单：submit()

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

browser = webdriver.Chrome()
browser.get('https://www.baidu.com')
# 找到输入框
input = browser.find_element_by_id('kw').send_keys("软件测试")
# 通过定位元素来提交
# browser.find_element_by_id("kw").submit()
# 直接回车提交
browser.find_element_by_id('kw').send_keys(Keys.ENTER)
time.sleep(1)
browser.close()
```

练习：登陆百度，输入：吃货，然后随机选择一个页面打开

```
from selenium import webdriver
import time, random
browser = webdriver.Chrome()
browser.get('https://www.baidu.com')
browser.find_element_by_id("kw").send_keys("吃货")
browser.find_element_by_id("kw").submit()  # 相当于回车
time.sleep(2)
links = browser.find_elements_by_css_selector("h3.t>a")
links[random.randint(0, 9)].click()
browser.close()
```

常见的键盘操作：

1. 键盘F1到F12：send_keys(Keys.F1)把F1改成对应的快捷键
2. 复制Ctrl+C：send_keys(Keys.CONTROL,'c')
3. 粘贴Ctrl+V：send_keys(Keys.CONTROL,'v')
4. 全选Ctrl+A：send_keys(Keys.CONTROL,'a')
5. 剪切Ctrl+X：send_keys(Keys.CONTROL,'x')
6. 制表键Tab：send_keys(Keys.TAB)

更多操作: http://selenium-python.readthedocs.io/api.html#module-selenium.webdriver.remote.webelement

交互动作

上面的例子，都是针对某个节点执行的，输入框可以输入和清除，按钮可以点击。但是还有另外一些操作，没有特定执行对象，比如：鼠标拖拽，键盘按键等，这些动作就需要用到动作链来完成。即：驱动浏览器完成一些交互动作，将动作附加到动作链中串行执行

- 鼠标拖拽

```
from selenium import webdriver
```

```
# 引入新模块
from selenium.webdriver import ActionChains

browser = webdriver.Chrome()
url = 'http://www.runoob.com/try/try.php?filename=jqueryui-api-droppable'
browser.get(url)
# 切换到frame，可以理解成一个标签
# 需要切换到该frame下，id=iframeResult，面试会问
browser.switch_to.frame('iframeResult')
# 定位被拖拽的目标，id=iframeResult
source = browser.find_element_by_css_selector('#draggable')
# 定位目标，id=droppable
target = browser.find_element_by_css_selector('#droppable')
# 申请动作链对象
actions = ActionChains(browser)
# 定义执行的动作
actions.drag_and_drop(source, target)
# 执行动作
actions.perform()
```

- 鼠标悬停，通过动作链，移动到定位的地点

```
from selenium import webdriver
# 引入新模块
from selenium.webdriver import ActionChains

browser = webdriver.Chrome()
url = 'https://www.baidu.com'
browser.get(url)
mouse = browser.find_element_by_link_text("设置")
ActionChains(browser).move_to_element(mouse).perform()
```

更多操作: http://selenium-python.readthedocs.io/api.html#module-selenium.webdriver.common.action_chains

1. click_and_hold：点击之后按住不动
2. context_click：右击
3. double_click：双击
4. drag_and_drop：拖拽
5. drag_and_drop_by_offset：指定拖拽多少
6. key_down/key_up：键盘的按键

执行JavaScript

经常定位的时候，有些页面需要拉动才能加载，光等待是没有用的。所以需要完成js动作，比如进度条下拉，通过api来实现很困难，通过js却很方便。下面是个万能的方法

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.zhihu.com/explore')
# 通过js代码，来实现浏览器操作，下拉到最下面
browser.execute_script('window.scrollTo(0, document.body.scrollHeight)')
# 弹出altert提示框
browser.execute_script('alert("To Bottom")')
```

- 有了`execute_script()`方法，API没有提供的功能，都可以用JavaScript来实现，前提是得学会JS

常用JS操作

横向滚动

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.zhihu.com/explore')
# 滚动指定的长度
js = "var q=document.documentElement.scrollTop=10000"
browser.execute_script(js)
time.sleep(2)
# 再滚动到顶部
# js="var q=document.getElementById('id').scrollTop=0"  # chrome浏览器不可用
js = "var q=document.documentElement.scrollTop=0"
browser.execute_script(js)
# 弹出altert提示框
browser.execute_script('alert("To Bottom")')
```

纵向滚动

横向滚动条现在已经很少见了

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.zhihu.com/explore')
# 滚动指定的长度：scrollTo(x, y)  x是横向距离，y是纵向距离，(0,0 )代表顶部
js = "window.scrollTo(100,400)"
browser.execute_script(js)
# 弹出altert提示框
browser.execute_script('alert("To right")')
```

元素聚焦

我们把焦点聚焦到我们要找到元素，对应的相关元素也会加载出来

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.taobao.com/')
target = browser.find_element_by_css_selector("ul.fl")
browser.execute_script("arguments[0].scrollIntoView();", target)
```

获取元素信息

- 节点有很多属性，可以通过`get_attrbute(属性名)`方法获取

获取属性

```
from selenium import webdriver
from selenium.webdriver import ActionChains

browser = webdriver.Chrome()
url = 'https://www.zhihu.com/explore'
browser.get(url)
# 获取title属性
print(browser.title)
# 获取浏览器名称
print(browser.name)
# 获取网站源码
print(browser.page_source)
# 定位到元素节点
logo = browser.find_element_by_class_name('ExploreHeader')
print(logo)
# 获取节点的属性值
print(logo.get_attribute('class'))
# 其他属性也有
browser.quit()
```

```
发现 - 知乎
chrome
<selenium.webdriver.remote.webelement.WebElement
(session="5f72f53b3473ad82adca9f1fde8b7038", element="0.23818534185670615-1")>
ExploreHeader
```

- tips：如果元素太难定位，可以直接获取源码，然后进行正则匹配获取数据内容
- 练习：抓取一个链接，获取href属性`obj.get_attribute('href')`

获取文本值

- 通过`.text`属性直接获取节点的文本值

```
from selenium import webdriver

browser = webdriver.Chrome()
# browser.maximize_window()
url = 'https://www.zhihu.com/explore'
browser.get(url)


# name只支持1个css
# Button SearchBar-searchIcon SearchBar-hasValueSearchIcon Button--primary
Button--blue
# Button AppHeader-login Button--blue
input = browser.find_element_by_css_selector(".Button.AppHeader-login.Button--
blue")
print(type(input))
# 获取文本值
print(input.text)

# browser.close()
```

```
<class 'selenium.webdriver.remote.webelement.WebElement'>
登录
```

获取ID、位置、标签名、大小

```
from selenium import webdriver

browser = webdriver.Chrome()
url = 'https://www.zhihu.com/explore'
browser.get(url)
# 注意：同个容器下的属性，不能用空格
input = browser.find_element_by_css_selector(".Button.AppHeader-login.Button--
blue")
# id值，不是html的id
print(input.id)
# 位置
print(input.location)
# 标签名称
print(input.tag_name)
# 尺寸
print(input.size)
browser.close()
```

```
0.813915720630356-1
{'x': 932, 'y': 9}
button
{'height': 34, 'width': 62}
```

Frame

网页中有种特殊的节点：iframe，也叫做子Frame，相当于页面的子页面，它的结构和外部网页结构完全一样。而selenium打开页面后，默认是在父Frame下操作的，如果页面包含子Frame，是不能获取到子Frame的节点的。必须通过：`switch_to.frame()`来切换Frame

```
import time
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException

browser = webdriver.Chrome()
url = 'http://www.runoob.com/try/try.php?filename=jqueryui-api-droppable'
browser.get(url)
# 切换frame，使用iframe的id进行切换
browser.switch_to.frame('iframeResult')
# 获移动块
source = browser.find_element_by_css_selector('#draggable')
# 获取成功
print(source)
try:
    # 尝试去获取logo，但logo在父frame，所以获取不到
    logo = browser.find_element_by_class_name('logo')
except NoSuchElementException:
    print('NO LOGO')
# 切换到父frame
browser.switch_to.parent_frame()
# 再次获取父页面的logo
logo = browser.find_element_by_class_name('logo')
print(logo)
# 成功获取
print(logo.text)
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="7b4942d1d9d25b63d4023342a72558e2", element="0.35573938241873004-1")>
NO LOGO
<selenium.webdriver.remote.webelement.WebElement
(session="7b4942d1d9d25b63d4023342a72558e2", element="0.7057940595844769-2")>
RUNOOB.COM
```

课后练习：尝试mail163.com 的iframe切换，账号登陆是一个iframe

```
import time
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException

browser = webdriver.Chrome()
url = 'https://mail.163.com/'
browser.get(url)
# 切换frame，使用iframe的id进行切换
browser.switch_to.frame('x-URS-iframe')
browser.find_element_by_name("email").send_keys("123")
browser.find_element_by_name("email").send_keys("456")

# 释放iframe，回到父frame下
browser.switch_to_default_content()
browser.quit()
```

select下拉框

以百度-设置-下拉框为例

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.select import Select
browser = webdriver.Chrome()
url = "https://www.baidu.com"
browser.get(url)
# browser.implicitly_wait(20)
# 鼠标移动到“设置”按钮
mouse = browser.find_element_by_link_text("设置")
ActionChains(browser).move_to_element(mouse).perform()
browser.find_element_by_link_text("搜索设置").click()
# 通过text:select_by_visible_text()
time.sleep(2)
s = browser.find_element_by_name("NR")
# 通过操作下拉对象
Select(s).select_by_visible_text("每页显示50条")


browser.quit()
```

- 通过xpath定位

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.select import Select
browser = webdriver.Chrome()
```

```
url = "https://www.baidu.com"
browser.get(url)
# 鼠标移动到“设置”按钮
mouse = browser.find_element_by_link_text("设置")
ActionChains(browser).move_to_element(mouse).perform()
browser.find_element_by_link_text("搜索设置").click()
# # 分两步：先定位下拉框，再点击选项
time.sleep(2)
s = browser.find_element_by_id("nr")
# 1. 通过value定位
s.find_element_by_xpath("//option[@value='50']").click()
browser.quit()
```

- 直接通过xpath索引

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.select import Select
browser = webdriver.Chrome()
url = "https://www.baidu.com"
browser.get(url)
# 鼠标移动到“设置”按钮
mouse = browser.find_element_by_link_text("设置")
ActionChains(browser).move_to_element(mouse).perform()
browser.find_element_by_link_text("搜索设置").click()
time.sleep(2)
# 通过xpath索引
browser.find_element_by_id("nr").find_element_by_xpath("//option[@value='50']"
).click()
time.sleep(1)
browser.quit()
```

- 通过select索引

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.select import Select
browser = webdriver.Chrome()
url = "https://www.baidu.com"
browser.get(url)
# 鼠标移动到“设置”按钮
mouse = browser.find_element_by_link_text("设置")
ActionChains(browser).move_to_element(mouse).perform()
browser.find_element_by_link_text("搜索设置").click()
time.sleep(2)
s = browser.find_element_by_id("nr")
Select(s).select_by_index(2)
```

```
browser.quit()
```

- 通过select的value值

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.select import Select
browser = webdriver.Chrome()
url = "https://www.baidu.com"
browser.get(url)
# 鼠标移动到“设置”按钮
mouse = browser.find_element_by_link_text("设置")
ActionChains(browser).move_to_element(mouse).perform()
browser.find_element_by_link_text("搜索设置").click()
time.sleep(2)
# 通过value：select_by_value()
s = browser.find_element_by_id("nr")
Select(s).select_by_value("20")
time.sleep(1)
browser.quit()
```

等待

因为存在ajax异步请求，这些请求不会管selenium有没有完成网页但加载，如果有后续的ajax请求没有加载完全，可能会导致一些问题，所以需要增加等待时间，确保节点都加载出来

隐式等待

当使用了隐式等待执行测试的时候，如果 WebDriver没有在 DOM中找到元素，将继续等待，超出设定时间后则抛出找不到元素的异常, 换句话说，当查找元素或元素并没有立即出现的时候，隐式等待将等待一段时间再查找 DOM，默认的时间是0

```
from selenium import webdriver

browser = webdriver.Chrome()
# 隐式等待
browser.implicitly_wait(10)
browser.get('https://www.zhihu.com/explore')
input = browser.find_element_by_css_selector(".Button.AppHeader-login.Button--blue")
print(input)
```

```
<selenium.webdriver.remote.webelement.WebElement (session="55b5d63d9bb27c60ce24e997eea3671a", element="0.46911913675325745-1")>
```

显式等待

指定一个等待条件，并且指定一个最长等待时间

元素出现

- 查找百度输入框

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

browser = webdriver.Chrome()
browser.get('https://www.baidu.com/')
# 构造wait对象，最长等待10秒，1秒轮询一次，默认是0.5秒轮询一次
wait = WebDriverWait(browser, 10, poll_frequency=1)
# 注意：需要传入一个函数对象
input = wait.until(lambda x : x.find_element_by_id("kw"))
input.send_keys("hello")
time.sleep(2)
browser.quit()
```

- 查询淘宝输入框

```
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

browser = webdriver.Chrome()
browser.get('https://www.taobao.com/')
# 如果等待时间内满足，则会正常返回，否则会等待到时间结束
# 实例化一个等待对象
wait = WebDriverWait(browser, 10)
# 判断输入框元素是否已经加载出，英语语法：until：直到 .... 为止
# 若10秒内还没加载出来，就抛出异常
input = wait.until(EC.presence_of_element_located((By.ID, 'q')))
# 判断搜索按钮是否可以点击，如果不可以点击，则抛出异常
button = wait.until(EC.element_to_be_clickable((By.CSS_SELECTOR, '.btn-
search')))
print(input, button)
button.click()
```

```
<selenium.webdriver.remote.webelement.WebElement
(session="1bfbe8639f254558348983e8e00584be", element="0.06910199746208145-1")>
<selenium.webdriver.remote.webelement.WebElement
(session="1bfbe8639f254558348983e8e00584be", element="0.06910199746208145-2")>
```

元素消失

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

browser = webdriver.Chrome()
browser.get('https://www.baidu.com/')
# 构造wait对象
wait = WebDriverWait(browser, 10)
# 判断id=kw元素是否消失
miss = wait.until_not()
browser.quit()
```

等待条件

- title_is 标题是某内容
- title_contains 标题包含某内容
- presence_of_element_located 元素加载出，传入定位元组，如(By.ID, 'p')
- visibility_of_element_located 元素可见，传入定位元组
- visibility_of 可见，传入元素对象
- presence_of_all_elements_located 所有元素加载出
- text_to_be_present_in_element 某个元素文本包含某文字
- text_to_be_present_in_element_value 某个元素值包含某文字
- frame_to_be_available_and_switch_to_it frame加载并切换
- invisibility_of_element_located 元素不可见
- element_to_be_clickable 元素可点击
- staleness_of 判断一个元素是否仍在DOM，可判断页面是否已经刷新
- element_to_be_selected 元素可选择，传元素对象
- element_located_to_be_selected 元素可选择，传入定位元组
- element_selection_state_to_be 传入元素对象以及状态，相等返回True，否则返回False
- element_located_selection_state_to_be 传入定位元组以及状态，相等返回True，否则返回False
- alert_is_present 是否出现Alert

详细内容：http://selenium-python.readthedocs.io/api.html#module-selenium.webdriver.support.expected_conditions

前进后退

浏览器到前进和后退：`back()`和`forward()`

```
import time
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.baidu.com/')
browser.get('https://www.taobao.com/')
# browser.get('https://www.python.org/')
# 后退
browser.back()
time.sleep(1)
```

```
# 前进
browser.forward()
time.sleep(1)
browser.close()
```

Cookies

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.zhihu.com/explore')
# 获取cookies
print(browser.get_cookies())
# 获取指定的name-cookie
print(browser.get_cookie("name"))
# 给cookie增加字段
browser.add_cookie({'name': 'name', 'domain': 'www.zhihu.com', 'value':
'germey'})
# 发现cookies列表多了一条
print(browser.get_cookies())
# 删除cookie
browser.delete_all_cookies()
print(browser.get_cookies())
browser.quit()
```

- 思考：通过接口请求，碰到重定向+token，获取不到cookies时，可以考虑使用selenium去获取cookie，拿到cookie后，再进行接口请求

- 练习：通过selenium获取：http://www.captaintests.club 的cookie，然后访问登陆后的页面

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('http://www.captaintests.club')
browser.find_element_by_id("inputUsername").send_keys("dcs")
browser.find_element_by_id("password").send_keys("123")
browser.find_element_by_class_name("form-signin").submit()
ck = browser.get_cookies()
print(ck)
time.sleep(2)
browser.get('http://www.captaintests.club/user/dcs/sign')
time.sleep(2)
browser.quit()
```

```
[{'domain': 'www.captaintests.club', 'httpOnly': True, 'name': 'session',
'path': '/', 'secure': False, 'value': '.eJwljktqA0EMBe_S6yxaakkj-
TJD69PYBBKYsVchd_eELOtRD-qn7euo895uz-
NVH21_ZLs1yiEI3GcsWzWNkKtDEc4l3XiuTYaBpQ9V9lhuGhtKFXGnIPXMtVG6QXGyxZ9ZQt1puFJ2
FyABHNeJEgtwBqZ2WCADDaNdIa-zjv8auDDOY-
3P78_6ugbH5AGsEFtHnVOUjJWc0aLPRMENQDnb7xvSEz4i.XmOfBg.mEw_0LiEe-
uBTfbBJgbZXQcCPrE'}]
```

- 思考：还有其他方式吗?
    1. 点击button
    2. 输入回车
- 思考：关于绕过验证码的问题

我们毕竟是测试，不是为了做爬虫，要绕过验证码基本就两种方法：1. 通过固定验证登陆；2. 抓包登陆后的cookie，然后通过add_cookie添加到browser里

选项卡(浏览器标签)管理

- 有时候我们需要新增多个标签页进行测试，可以通过js执行window.open()来新增tab页，通过`switch_to_window()`来切换标签（句柄）

```python
import time
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.baidu.com')
# 获取当前页面句柄
print(browser.current_window_handle)
# 添加一个选项卡
browser.execute_script('window.open()')
# 获取所有句柄
print(browser.window_handles)
# 切换到第二个选项卡
browser.switch_to_window(browser.window_handles[1])
browser.get('https://www.taobao.com')
time.sleep(1)
# 切回第一个选项卡
browser.switch_to_window(browser.window_handles[0])
browser.get('https://www.zhihu.com')
browser.quit()
```

```
CDwindow-54EDFCF5D2AB72A12A4A206EBD6D84D1
['CDwindow-54EDFCF5D2AB72A12A4A206EBD6D84D1', 'CDwindow-
1853F8328B245691021FA6326D566C90']
```

```
/Users/panwj/.local/share/virtualenvs/mypython3-Xz6Q90qj/lib/python3.7/site-packages/ipykernel_launcher.py:12: DeprecationWarning: use driver.switch_to.window instead
  if sys.path[0] == '':
/Users/panwj/.local/share/virtualenvs/mypython3-Xz6Q90qj/lib/python3.7/site-packages/ipykernel_launcher.py:16: DeprecationWarning: use driver.switch_to.window instead
  app.launch_new_instance()
```

异常处理

- 一般做爬虫需要去捕获异常，避免程序中断

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('https://www.baidu.com')
# 使用一个不存在的id去查，系统会抛出异常
browser.find_element_by_id('hello')
```

主动去捕获异常，让程序继续往下运行而不会中断

```
from selenium import webdriver
from selenium.common.exceptions import TimeoutException, NoSuchElementException

browser = webdriver.Chrome()
try:
    browser.get('https://www.baidu.com')
except TimeoutException:
    # 捕获超时
    print('Time Out')
try:
    browser.find_element_by_id('hello')
except NoSuchElementException:
    # 捕获不存在的情况
    print('No Element')
finally:
    browser.close()
```

```
No Element
```

异常类文档：http://selenium-python.readthedocs.io/api.html#module-selenium.common.exceptions

扩展API

先获取浏览器对象：`browser = webdriver.Chrome()`

```
from selenium import webdriver
import time
browser = webdriver.Chrome()
browser.get('https://www.baidu.com')
# 1. 设置窗口为固定大小
# browser.set_window_size(500, 400)

# 2. 窗口最大化，有些版本会报错，需要更换驱动
# browser.maximize_window()

# 3. 截屏
# browser.get("https://www.baidu.com")
# browser.get_screenshot_as_file("baidu.jpg")

# 4. 关闭浏览器，有两种方式：close关闭当前窗口，quit关闭所有窗口(所有测试结束后执行)
# browser.get("https://www.baidu.com")
# # 切换选项卡1
# browser.execute_script('window.open()')
# browser.switch_to_window(browser.window_handles[1])
# browser.get("https://www.zhihu.com")
# # 切换选项卡2
# browser.execute_script('window.open()')
# browser.switch_to_window(browser.window_handles[2])
# browser.get("https://www.taobao.com")
# # 关闭当前
# browser.close()
# time.sleep(2)
# # 关闭所有窗口
# browser.quit()

# 刷新页面
browser.refresh()
```

```
/Users/panwj/.local/share/virtualenvs/mypython3-Xz6Q90qj/lib/python3.7/site-packages/ipykernel_launcher.py:14: DeprecationWarning: use driver.switch_to.window instead

/Users/panwj/.local/share/virtualenvs/mypython3-Xz6Q90qj/lib/python3.7/site-packages/ipykernel_launcher.py:17: DeprecationWarning: use driver.switch_to.window instead
```

定位参数化

- by_id= "id"
- by_xpath = "xpath"
- by_link_text = "link text"
- by_partial_text = "partial link text"
- by_name = "name"
- by_tag_name = "tag name"
- by_class_name = "class name"
- by_css_selector = "css selector"

定位常见问题总结

> class包含多个，定位的时候不准?

1. 逐个搜索，确认是否有唯一的class name，使用它进行定位
2. 使用 css_selector，组合多个classname+标签，一起定位