

**李刘村**

**Python Selenium 自动化测试**

目录

**[一. 原理与安装](#_Toc1788807250_WPSOffice_Level1)** **[5](#_Toc1788807250_WPSOffice_Level1)**

[１. 原理](#_Toc1122508041_WPSOffice_Level2) [5](#_Toc1122508041_WPSOffice_Level2)

[２. 安装](#_Toc348806192_WPSOffice_Level2) [6](#_Toc348806192_WPSOffice_Level2)

[３. 基本使用](#_Toc1902796281_WPSOffice_Level2) [6](#_Toc1902796281_WPSOffice_Level2)

**[二. 选择元素](#_Toc1122508041_WPSOffice_Level1)** **[7](#_Toc1122508041_WPSOffice_Level1)**

[１. 简介](#_Toc2118107290_WPSOffice_Level2) [7](#_Toc2118107290_WPSOffice_Level2)

[２. 根据元素的id属性选择元素](#_Toc192806711_WPSOffice_Level2) [7](#_Toc192806711_WPSOffice_Level2)

[３. 根据class属性、tag名选择元素](#_Toc2097052101_WPSOffice_Level2) [7](#_Toc2097052101_WPSOffice_Level2)

**[三. 操控元素](#_Toc348806192_WPSOffice_Level1)** **[11](#_Toc348806192_WPSOffice_Level1)**

[１. 输入框](#_Toc653046943_WPSOffice_Level2) [11](#_Toc653046943_WPSOffice_Level2)

[２. 获取元素信息](#_Toc2118534831_WPSOffice_Level2) [12](#_Toc2118534831_WPSOffice_Level2)

**[四. 通过CSS Selector来选择元素](#_Toc1902796281_WPSOffice_Level1)** **[14](#_Toc1902796281_WPSOffice_Level1)**

[１. CSS Selector如何选择元素](#_Toc936037357_WPSOffice_Level2) [14](#_Toc936037357_WPSOffice_Level2)

[２. 根据tag名、id、class来选择元素](#_Toc1662144824_WPSOffice_Level2) [14](#_Toc1662144824_WPSOffice_Level2)

[３. 选择子元素和后代元素](#_Toc1200776792_WPSOffice_Level2) [16](#_Toc1200776792_WPSOffice_Level2)

[４. 根据属性选择](#_Toc1551712285_WPSOffice_Level2) [17](#_Toc1551712285_WPSOffice_Level2)

[５. 选择语法的联合使用](#_Toc586964827_WPSOffice_Level2) [18](#_Toc586964827_WPSOffice_Level2)

[６. 兄弟节点的选择](#_Toc1725456718_WPSOffice_Level2) [20](#_Toc1725456718_WPSOffice_Level2)

**[五. frame切换/窗口切换](#_Toc2118107290_WPSOffice_Level1)** **[21](#_Toc2118107290_WPSOffice_Level1)**

[１. 切换frame](#_Toc131890338_WPSOffice_Level2) [21](#_Toc131890338_WPSOffice_Level2)

[２. 切换窗口](#_Toc477787062_WPSOffice_Level2) [23](#_Toc477787062_WPSOffice_Level2)

**[六. 操作选择框](#_Toc192806711_WPSOffice_Level1)** **[23](#_Toc192806711_WPSOffice_Level1)**

[１. 操作radio框](#_Toc725794901_WPSOffice_Level2) [23](#_Toc725794901_WPSOffice_Level2)

[２. 操作checkbox框](#_Toc727786147_WPSOffice_Level2) [24](#_Toc727786147_WPSOffice_Level2)

[３. 操作select框](#_Toc1982402964_WPSOffice_Level2) [24](#_Toc1982402964_WPSOffice_Level2)

**[七. 页面动作](#_Toc2097052101_WPSOffice_Level1)** **[25](#_Toc2097052101_WPSOffice_Level1)**

[１. 移动鼠标到元素](#_Toc37832743_WPSOffice_Level2) [25](#_Toc37832743_WPSOffice_Level2)

[２. 冻结页面（进入debugger模式）](#_Toc199752089_WPSOffice_Level2) [26](#_Toc199752089_WPSOffice_Level2)

[３. 对话框](#_Toc716419562_WPSOffice_Level2) [27](#_Toc716419562_WPSOffice_Level2)

[４. 上传文件](#_Toc2070253452_WPSOffice_Level2) [27](#_Toc2070253452_WPSOffice_Level2)

[５. 更多操作](#_Toc1219719070_WPSOffice_Level2) [27](#_Toc1219719070_WPSOffice_Level2)

**[八. 扩展：Xpath表达式](#_Toc653046943_WPSOffice_Level1)** **[28](#_Toc653046943_WPSOffice_Level1)**

[１. Xpath简介](#_Toc2086998875_WPSOffice_Level2) [28](#_Toc2086998875_WPSOffice_Level2)

[２. 根据属性选择](#_Toc1339685674_WPSOffice_Level2) [29](#_Toc1339685674_WPSOffice_Level2)

[３. 根据元素次序选择](#_Toc1878567770_WPSOffice_Level2) [30](#_Toc1878567770_WPSOffice_Level2)

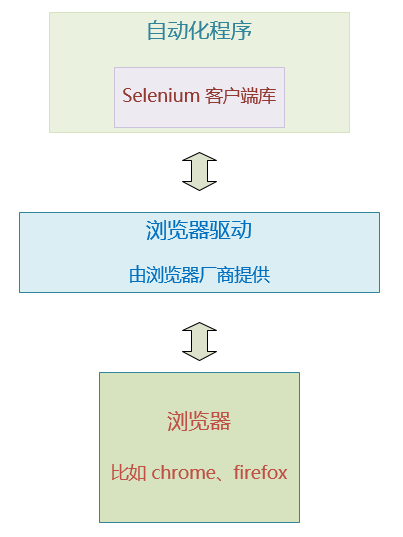
[４. 组选择、父节点、兄弟节点](#_Toc783932196_WPSOffice_Level2) [31](#_Toc783932196_WPSOffice_Level2)

# 原理与安装

## 原理

Selenium是一个Web应用的自动化框架。

通过它，我们可以写出自动化程序，像人一样在浏览器里操作web界面。 比如点击界面按钮，在文本框中输入文字等操作。



selenium 自动化流程如下：

* 自动化程序调用Selenium 客户端库函数（比如点击按钮元素）
* 客户端库会发送Selenium 命令 给浏览器的驱动程序
* 浏览器驱动程序接收到命令后 ,驱动浏览器去执行命令
* 浏览器执行命令
* 浏览器驱动程序获取命令执行的结果，返回给我们自动化程序
* 自动化程序对返回结果进行处理

## 安装

Selenium的安装主要包括两个模块：

客户端库

浏览器驱动

### 安装客户端库

pip install selenium

### 安装浏览器驱动

* 查看自己的浏览器厂商及版本。

这里以Chrome为例，我的Chrome版本为80.0.3987.149。

* 下载驱动。

驱动下载地址：<https://chromedriver.storage.googleapis.com/index.html>

需要注意驱动的平台（linux，windows，mac）

* 下载完成后，将驱动解压放到python可执行程序的目录。或者在加载驱动时，指定驱动路径。

## 基本使用

下面的代码, 可以自动化的 打开Chrome浏览器，并且自动化打开百度网站，可以大家可以运行一下看看。

from selenium import webdriver

# 创建 WebDriver 对象，指明使用chrome浏览器驱动

wd = webdriver.Chrome(r'd:\webdrivers\chromedriver.exe')

# 调用WebDriver 对象的get方法 可以让浏览器打开指定网址wd.get('https://www.baidu.com')

执行上面这行代码时，自动化程序就发起了 打开百度网址的 请求消息 ，通过浏览器驱动， 给 Chrome浏览器。

Chome浏览器接收到该请求后，就会打开百度网址，通过浏览器驱动， 告诉自动化程序 打开成功。

执行完自动化代码，如果想关闭浏览器窗口可以调用WebDriver对象的 quit 方法

wd.quit()

# 选择元素

## 简介

选择元素就是要告诉浏览器需要操作的元素的特征。

打开浏览器的开发者工具，跳转Elements标签，就可以看到每个页面元素以及他们的“特征”。

## 根据元素的id属性选择元素

打开百度首页，查看页面元素，找到输入框的标签为

<input type="text" class="s\_ipt" name="wd" id="kw" maxlength="100" autocomplete="off">

我们需要在输入框输入selenium并点击搜索。

from selenium import webdriver  
  
  
# 创建 WebDriver 对象，指明使用chrome浏览器驱动  
# wd = webdriver.Chrome("Chrome/PATH")  
wd = webdriver.Chrome()  
  
# 调用WebDriver 对象的get方法 可以让浏览器打开指定网址  
url = 'https://www.baidu.com'  
wd.get(url)  
  
# 根据id选择元素，返回的就是该元素对应的WebElement对象  
element = wd.find\_element\_by\_id('kw')  
  
# 通过该 WebElement对象，就可以对页面元素进行操作了  
# 比如输入字符串到 这个 输入框里  
element.send\_keys('selenium')  
element.click()

## 根据class属性、tag名选择元素

### 根据class属性选择元素

以下HTML内容来自http://cdn1.python3.vip/files/selenium/sample1.html 。

<body>  
   
 <div class="plant"><span>土豆</span></div>  
 <div class="plant"><span>洋葱</span></div>  
 <div class="plant"><span>白菜</span></div>  
  
 <div class="animal"><span>狮子</span></div>  
 <div class="animal"><span>老虎</span></div>  
 <div class="animal"><span>山羊</span></div>

</body>

所有的植物元素都有个class属性 值为 plant。

所有的动物元素都有个class属性 值为 animal。

如果我们要选择所有的动物，就可以使用方法 find\_elements\_by\_class\_name 。

wd.find\_elements\_by\_class\_name('animal')

注意

find\_elements\_by\_class\_name 方法返回的是找到的符合条件的所有元素 (这里有3个元素)，放在一个列表中返回。如果没有符合条件的元素，返回空列表。

find\_element\_by\_class\_name (注意少了一个s) 方法， 就只会返回第一个元素。如果没有符合条件的元素，抛出NoSuchElementException异常。

代码实现：

from selenium import webdriver  
  
# 创建 WebDriver 实例对象，指明使用chrome浏览器驱动  
wd = webdriver.Chrome()  
  
# WebDriver 实例对象的get方法 可以让浏览器打开指定网址  
wd.get('http://cdn1.python3.vip/files/selenium/sample1.html')  
  
# 根据 class name 选择元素，返回的是 一个列表  
# 里面 都是class 属性值为 animal的元素对应的 WebElement对象  
elements = wd.find\_elements\_by\_class\_name('animal')  
  
# 取出列表中的每个 WebElement对象，打印出其text属性的值  
# text属性就是该 WebElement对象对应的元素在网页中的文本内容  
for element in elements:  
 print(element.text)

首先，大家要注意：通过 WebElement 对象的 text属性可以获取该元素在网页中的文本内容。

所以下面的代码，可以打印出 element 对应网页元素的文本

print(element.text)  
如果我们把

elements = wd.find\_elements\_by\_class\_name('animal')

去掉一个s ，改为

element = wd.find\_element\_by\_class\_name('animal')

print(element.text)

那么返回的就是第一个class 属性为 animal的元素， 也就是这个元素

<div class="animal"><span>狮子</span></div>

**多个相同属性的写法**

元素也可以有多个class类型 ，多个class类型的值之间用**空格**隔开，比如

<span class="chinese student">张三</span>

注意，这里 span元素有两个class属性，分别是 chinese 和 student， 而不是一个名为  chinese student 的属性。

我们要用代码选择这个元素，可以指定任意一个class 属性值，都可以选择到这个元素，如下

element = wd.find\_elements\_by\_class\_name('chinese')

或者

element = wd.find\_elements\_by\_class\_name('student')

而不能这样写

element = wd.find\_elements\_by\_class\_name('chinese student')

### 根据tag名选择元素

我们可以通过find\_elements\_by\_tag\_name来查找元素。

from selenium import webdriver  
  
wd = webdriver.Chrome()  
  
wd.get('http://cdn1.python3.vip/files/selenium/sample1.html')  
  
# 根据 tag name 选择元素，返回的是 一个列表  
# 里面 都是 tag 名为 div 的元素对应的 WebElement对象  
elements = wd.find\_elements\_by\_tag\_name('div')  
  
# 取出列表中的每个 WebElement对象，打印出其text属性的值  
# text属性就是该 WebElement对象对应的元素在网页中的文本内容  
for element in elements:  
 print(element.text)

### 通过WebElement对象选择元素

不仅 WebDriver对象有选择元素的方法，WebElement对象也有选择元素的方法。

WebElement对象也可以调用 find\_elements\_by\_xxx，  find\_element\_by\_xxx 之类的方法。WebDriver 对象选择元素的范围是整个web页面， 而WebElement 对象选择元素的范围是该元素的内部。

有如下HTML内容：

<div id="container">  
  
 <div id="layer1">  
 <div id="inner11">  
 <span>内层11</span>  
 </div>  
 <div id="inner12">  
 <span>内层12</span>  
 </div>  
 </div>  
  
 <div id="layer2">  
 <div id="inner21">  
 <span>内层21</span>  
 </div>  
 </div>  
  
</div>

代码如下：

from selenium import webdriver  
  
wd = webdriver.Chrome()  
  
wd.get('http://cdn1.python3.vip/files/selenium/sample1.html')  
  
element = wd.find\_element\_by\_id('container')  
  
# 限制 选择元素的范围是 id 为 container 元素的内部。  
spans = element.find\_elements\_by\_tag\_name('span')  
for span in spans:  
 print(span.text)

返回结果为：

内层11

内层12

内层21

### 等待界面元素的出现

在我们进行网页操作的时候， 有的元素内容不是可以立即出现的， 可能会等待一段时间。

*Selenium提供了一个更合理的解决方案，如下：*

*当发现元素没有找到的时候，并不立即返回找不到元素的错误。而是周期性（每隔半秒钟）重新寻找该元素，直到该元素找到，或者超出指定最大等待时长，这时才抛出异常（如果是 find\_elements 之类的方法，则是返回空列表）。*

Selenium 的 Webdriver 对象 有个方法叫  implicitly\_wait。该方法接受一个参数，用来指定最大等待时长。

具体实现如下：

from selenium import webdriver  
  
wd = webdriver.Chrome()  
# 设置最大等待时长为 10s。此后，所有find\_element(s)之类的方法都会使用该策略

# 如果找不到，则每隔0.5s查询一次，直到找到该元素或超出限定时长（这里是10s）  
wd.implicitly\_wait(10)  
  
wd.get('https://www.baidu.com')  
  
element = wd.find\_element\_by\_id('kw')  
  
element.send\_keys('selenium\n')  
  
element = wd.find\_element\_by\_id('1')  
  
print (element.text)

# 操控元素

操控元素通常包括

* 点击元素
* 在元素中输入字符串，通常是对输入框这样的元素
* 获取元素包含的信息，比如文本内容，元素的属性

**点击元素**

即是WebElement对象的click方法，这个已经接触。

不过在点击时，点击的是对应元素的中心点位置。

## 输入框

**输入字符串**也非常简单，就是调用元素WebElement对象的send\_keys方法。前面我们也已经学过。

如果我们要把输入框中已经有的内容清除掉，可以使用WebElement对象的**clear**方法。

有如下HTML内容：

<div>  
 <input id="input1" value="请输入姓名">  
 <br><br>  
</div>

需要先清除输入框内的placeholder：“请输入姓名”，然后再输入内容。代码如下

element = wd.find\_element\_by\_id("input1")  
  
element.clear() # 清除输入框已有的字符串  
element.send\_keys('selenium') # 输入新字符串

## 获取元素信息

* 获取元素的文本内容（展示在界面上的内容），使用text属性。

element = wd.find\_element\_by\_id('animal')  
print(element.text)

* 获取元素属性，使用get\_attribute方法。

element = wd.find\_element\_by\_id('input\_name')

# 获取class属性的值  
print(element.get\_attribute('class'))

* 获取元素对应的HTML文本内容。

获取整个HTML内容，使用get\_attribute(“outerHTML”)

获取某个元素内部的HTML内容，使用get\_attribute(“innerHTML”)

有HTML内容为

<div id="container">  
  
 <div id="layer1">  
 <div id="inner11">  
 <span>内层11</span>  
 </div>  
 <div id="inner12">  
 <span>内层12</span>  
 </div>  
 </div>  
  
 <div id="layer2">  
 <div id="inner21">  
 <span>内层21</span>  
 </div>  
 </div>  
  
</div>

实现代码为

from selenium import webdriver  
  
wd = webdriver.Chrome()  
  
wd.get('http://cdn1.python3.vip/files/selenium/sample1.html')  
  
element = wd.find\_element\_by\_id('container')  
  
# 获取整个HTML内容  
outer = element.get\_attribute("outerHTML")  
print(outer)  
print("---"\*20)  
  
# 获取id为container的元素的HTML内容  
inner = element.get\_attribute("innerHTML")  
print(inner)

运行结果为

<div class="animal">

<span>山羊</span>

</div>

<div id="container">  
  
 <div id="layer1">  
 <div id="inner11">  
 <span>内层11</span>  
 </div>  
 <div id="inner12">  
 <span>内层12</span>  
 </div>  
 </div>  
  
 <div id="layer2">  
 <div id="inner21">  
 <span>内层21</span>  
 </div>  
 </div>  
  
</div>  
------------------------------------------------------------  
  
<div id="layer1">  
 <div id="inner11">  
 <span>内层11</span>  
 </div>  
 <div id="inner12">  
 <span>内层12</span>  
 </div>  
</div>  
  
<div id="layer2">  
 <div id="inner21">  
 <span>内层21</span>  
 </div>  
</div>

* 获取输入框里面的文字。

获取input输入框的内容，使用get\_attribute(“value”)。

element = wd.find\_element\_by\_id(“input1”)

print(element.get\_attribute(‘value’))

获取展示在界面上的内容，使用text属性。有时，元素的文本内容没有完全展示在界面上，使用text可能会有问题，可以尝试使用element.get\_attribute(‘innerText’)，或者element.get\_attribute(‘textContext’).

# 通过CSS Selector来选择元素

Selenium提供了多种选择元素的方法。之前介绍的应用面还不够广泛，现在介绍另一种选择元素的方法：通过CSS Selector来选择页面元素

## CSS Selector如何选择元素

这里首先要了解CSS的功能。CSS使页面更美化，指定了某些元素的显示样式。CSS的特长就是指定元素，所以selenium可以借助CSS来选择元素。

基本语法：

find\_element\_by\_css\_selector(“CSS params”)

当然也有选择多个元素的find\_elements\_by\_css\_selector(“CSS params”)

## 根据tag名、id、class来选择元素

CSS Selector 同样可以根据tag名、id 属性和 class属性 来 选择元素。假设有如下网页内容：

<html lang="en"><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
  
 <title>网页名称</title>  
  
 <style>  
 .animal {color: red;}  
 </style>  
 </head>  
  
 <body>  
  
 <div class="plant"><span>土豆</span></div>  
 <div class="plant"><span>洋葱</span></div>  
 <div class="plant"><span>白菜</span></div>  
  
 <div class="animal"><span>狮子</span></div>  
 <div class="animal"><span>老虎</span></div>  
 <div class="animal"><span>山羊</span></div>  
  
 <input type="text" id="searchtext">  
 </body>  
</html>

### 根据 tag名选择元素

直接写tag的名字即可，如选择网页所有tag名为div的元素

elements = wd.find\_elements\_by\_css\_selector('div')

等价于

elements = wd.find\_elements\_by\_tag\_name('div')

### 根据id属性选择元素

需要在id号前面加上一个井号：#id值。如选择网页中的输入框

element = wd.find\_element\_by\_css\_selector(“#searchtext”)

### 根据class属性选择元素

需要在class名称前加点：.class值。如选择网页中class为plant的元素。

element = wd.find\_elements\_by\_css\_selector(“.plant”)

等价于

element = wd.find\_elements\_by\_class\_name(“plant”)

因为这里有多个元素符合条件，所以选择elements而不是element。

## 选择子元素和后代元素

HTML中，元素的结构可以很复杂，可能会有多层次的包含关系。有如下HTML内容。

<div class="animal">

<span>山羊</span>

</div>

<div id="container">  
 <div id="layer1">  
 <div id="inner11">  
 <span>内层11</span>  
 </div>  
 <div id="inner12">  
 <span>内层12</span>  
 </div>  
 </div>  
 <div id="layer2">  
 <div id="inner21">  
 <span>内层21</span>  
 </div>  
 </div>  
</div>

介绍：

id 为 container 的div元素包含了 id 为  layer1 和  layer2 的两个div元素。这种包含是直接包含，中间没有其他的层次的元素了。所以 id 为  layer1 和  layer2 的两个div元素是 id 为 container 的div元素的直接子元素。

同样 id 为  inner11 和  inner12 的两个div元素也是 id 为layer1 的 div 元素的直接子元素。 id 为  inner21 这个div元素也是 id 为  layer2 的div元素的直接子元素。

而对于 id 为 container 的div元素来说， id 为  inner11 、inner12 、inner22 的元素和两个  span类型的元素都不是它的直接子元素。

虽然不是直接子元素， 但是它们还是在  container 的内部， 可以称之为它的后代元素。后代元素也包括了直接子元素，比如 id 为  layer1 和  layer2 的两个div元素也可以说是 id 为 container 的div元素的直接子元素，同时也是后代元素。

### CSS选择子元素

如果元素2 是元素1 的直接子元素，CSS Selector **选择子元素**的语法为：

元素1 > 元素2

注意，最终选择的元素是元素2，并且要求这个元素2 是元素1 的直接子元素。

也支持更多层级的选择，比如

元素1 > 元素2 > 元素3 > 元素4

就是选择元素1 里面的子元素  元素2 里面的子元素  元素3 里面的子元素  元素4 ， 最终选择的元素是 元素4

### 3.2 CSS选择后代元素

如果元素2 是元素1 的后代元素，CSS Selector **选择后代元素**的语法是这样的

元素1 元素2

中间是一个或者多个空格隔开

最终选择的元素是元素2 ，并且要求这个元素2 是元素1 的后代元素（当然子元素也可以这么写）。  
后代元素也支持更多层级的选择，比如

元素1 元素2 元素3 元素4

最终选择的元素是元素4

## 根据属性选择

css 选择器支持通过任何属性来选择元素，语法是用一个方括号 [] 。有如下HTML内容。

<div class="animal" name=”goat”>

<span>山羊</span>

</div>

可以通过name属性来选择：

element = wd.find\_element\_by\_css\_selector(‘[name=”goat”]’)

需要注意引号的使用。

根据属性选择，还可以不指定属性值，如下表示选择所有具有属性名为name 的元素，不管它们的值是什么。

element = wd.find\_element\_by\_css\_selector(‘[name]’)

通过属性值，模糊查询包含某个字符串的元素。如下查询name属性包含goat字符串的元素，则name=“goat123”的元素也会被选择。

element = wd.find\_element\_by\_css\_selector(‘[name\*=”goat”]’)

查询属性值以特定的字符串开始和结尾。如下查询分别以abc开始和以xyz结尾的元素。

element = wd.find\_element\_by\_css\_selector(‘[name^=”abct”]’)

element = wd.find\_element\_by\_css\_selector(‘[name$=”xyz”]’)

可以同时查询符合多个属性值的元素。如下查询name为goat，并且class属性为animal的元素。

element = wd.find\_element\_by\_css\_selector(‘[name=”goat”][class=”animal”]’)

以上属性查询可以和标签联合使用，来查询特定标签的相关属性元素。如下查询div标签中，name属性包含goat字符串的元素。

element = wd.find\_element\_by\_css\_selector(‘div[name\*=”goat”]’)

## 选择语法的联合使用

其实在上一节已经接触过，比如属性和标签联合使用，你的CSS表达式越详细，选择的元素就越准确。

有如下HTML内容

<div id="bottom">  
 <div class="footer1">  
 <span class="copyright">版权</span>  
 <span class="date">发布日期：2018-03-03</span>  
 </div>  
 <div class="footer2">  
 <span>备案号  
 <a href="http://www.miitbeian.gov.cn/">苏ICP备88885574号</a>  
 </span>  
 </div>   
</div>

我们要选择元素  <span class='copyright'>版权</span>

CSS selector 表达式可以这样写：

div.footer1 > span.copyright

就是选择一个class 属性值为 copyright 的 span 节点，并且要求其必须是 class 属性值为 footer1 的 div节点的子节点

也可以更简单：

.footer1 > .copyright

就是选择一个class 属性值为copyright 的节点（不限类型）， 并且要求其必须是 class 属性值为 footer1 的节点的子节点

因为子元素同时也是后代元素，所以也可以这样写

.footer1 .copyright

以上三种方法，第一种最精确，第三种约束最弱。当然，你也可以加上上层关系使其更精确，即

element = wd.find\_elements\_by\_css\_selector("div#bottom > div.footer1 > span.copyright")

### 组选择

如果我们的选择条件是“或”的关系，该如何表达？

css选择器可以使用逗号 ，称之为组选择 ，像这样表示选择所有class 为 plant 或 class 为 animal 的元素。

element = wd.find\_elements\_by\_css\_selector(".plant, .animal")

同时选择所有tag名为div的元素 或 id为BYHY的元素

element = wd.find\_elements\_by\_css\_selector("div, #BYHY")

注意：

逗号的优先级很低，所以div > .plant, .animal 和 div > .plant, div > .animal有差别。

### 按次序选择子节点

有如下HTML内容

<div id="t1">  
 <h3> 唐诗 </h3>  
 <span>李白</span>  
 <p>静夜思</p>  
 <span>杜甫</span>  
 <p>春夜喜雨</p>   
</div>   
  
<div id="t2">  
 <h3> 宋词 </h3>  
 <span>苏轼</span>  
 <p>赤壁怀古</p>  
 <p>明月几时有</p>  
 <p>江城子·乙卯正月二十日夜记梦</p>  
 <p>蝶恋花·春景</p>  
 <span>辛弃疾</span>  
 <p>京口北固亭怀古</p>  
 <p>青玉案·元夕</p>  
 <p>西江月·夜行黄沙道中</p>   
</div>

### 父元素的第n个子节点

我们可以指定选择的元素 是父元素的第几个子节点，使用  nth-child。比如，

我们要选择 唐诗 和宋词 的第一个 作者，也就是说选择的是第2个子元素，并且是span类型，所以这样可以这样写

span:nth-child(2)

如果你不加节点类型限制，直接这样写  :nth-child(2)，就是选择所有位置为第2个的所有元素，不管是什么类型。

### 父元素的倒数第n个子节点

也可以反过来，选择的是父元素的倒数第几个子节点 ，使用 nth-last-child。比如：

p:nth-last-child(1)

就是选择第倒数第1个子元素，并且是p元素。

### 父元素的第几个某类型的子节点

我们可以指定选择的元素是父元素的第几个某类型的子节点，使用  nth-of-type。比如，

我们要选择 唐诗 和宋词 的第一个 作者，可以像上面那样思考：选择的是 第2个子元素，并且是span类型，所以可以这样写

span:nth-child(2)

还可以这样思考，选择的是第1个span类型的子元素，所以也可以这样写

span:nth-of-type(1)

### 父元素的倒数第几个某类型的子节点

当然也可以反过来，选择父元素的倒数第几个某类型的子节点，使用  nth-last-of-type。像这样

p:nth-last-of-type(2)

### 奇数节点和偶数节点

如果要选择的是父元素的 偶数节点，使用  nth-child(even)。比如

p:nth-child(even)

如果要选择的是父元素的 奇数节点，使用  nth-child(odd)。比如

p:nth-child(odd)

如果要选择的是父元素的 某类型偶数节点，使用  nth-of-type(even)

如果要选择的是父元素的 某类型奇数节点，使用  nth-of-type(odd)

## 兄弟节点的选择

### 相邻兄弟节点选择

上面的例子里面，我们要选择唐诗和宋词的第一个作者，还有一种思考方法，就是选择 h3  后面紧跟着的兄弟节点 span。这就是一种相邻兄弟关系，可以这样写

h3 + span

表示元素紧跟关系的是加号。

### 后续所有兄弟节点选择

如果要选择是 选择 h3  后面所有的兄弟节点 span，可以这样写

h3 ~ span

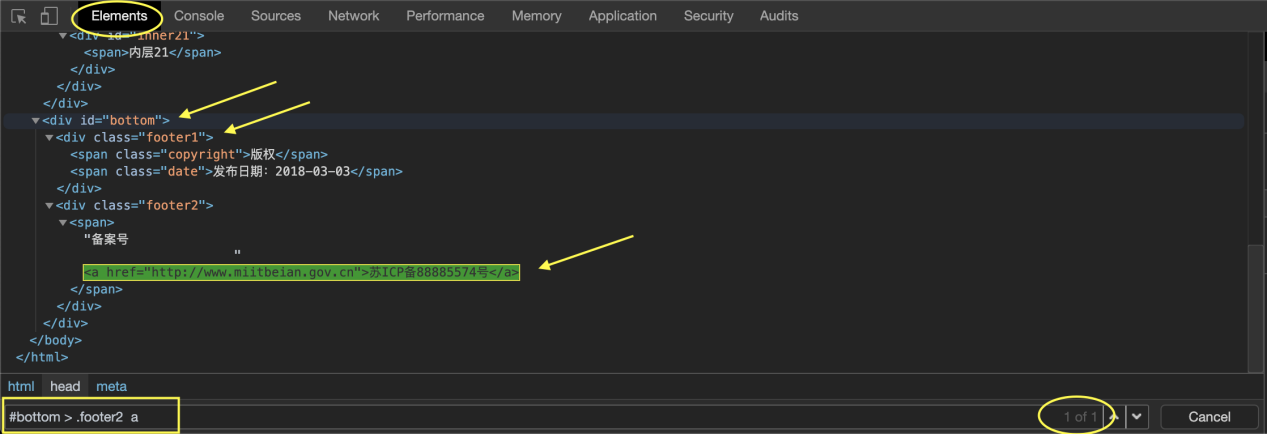
**更多CSS选择器**

参考 <http://www.w3school.com.cn/cssref/css_selectors.asp>

**验证CSS表达式**

进入开发者模式，选择查看页面元素（跳转Element），进入查找模式，在输入框中输入CSS表达式，看选中的元素是否是自己期望的。

下图验证CSS表达式：**#bottom > .footer2 a**



# frame切换/窗口切换

当页面有多个frame或跳转窗口后，该如何操作？

## 切换frame

有如下HTML内容

<body>  
 <div class="bodyname">  
 <span>下面的内容是iframe中的</span>  
 </div>  
 <iframe src="sample1.html" id="frame1" name="innerFrame" width="300" height="200">  
 <html lang="en">  
 <head>  
 <meta charset="UTF-8">  
 <title>测试网页</title>  
 </head>  
 <body>  
 <input type="text" id="searchtext">  
 </body>  
 </html>  
 </iframe>  
 <div id="add"></div>  
</body>

### 切换到frame

如果需要选中id为searchtext的输入框，该如何选择？

如果你写为

element = wd.find\_element\_by\_id('searchtext')

那么你会得到NoSuchElementException。

仔细查看会发现这些元素在一个iframe标签里。这个标签很特殊，它相当于另一个html页面，有自己的head和body。与之类似的还有frame标签。

而selenium默认链接最外层html中的内容。如果想要选择iframe里面的内容，需要定位到iframe标签，并切换。

from selenium import webdriver  
  
wd = webdriver.Chrome()  
wd.get('http://cdn1.python3.vip/files/selenium/sample2.html')  
# 切换到iframe所包含的html文档  
# 可以通过ID值"frame1"  
wd.switch\_to.frame("frame1")  
element = wd.find\_element\_by\_css\_selector("#searchtext")

以上代码可以定位到id为searchtext的输入框。

定位frame的方法为wd.switch\_to.frame(frame\_reference). frame\_reference是对你将要切换的frame的描述。共有4中方法切换，以本文为例：

* index wd.switch\_to.frame(0) 表示切换到第1个（index为0）frame。当frame\_reference为整数时，默认为index。
* id属性值 wd.switch\_to.frame("frame1")
* name属性值 wd.switch\_to.frame("innerFrame")
* WebElement对象 wd.switch\_to.frame(wd.find\_element\_by\_tag\_name("iframe"))或者wd.switch\_to.frame(wd.find\_element\_by\_css\_selector('iframe[src\*="sample"]'))

### 切换回主frame

我们切换到某个frame之后，操作都是针对frame里的内容，如何返回操作默认页面的内容呢？使用

wd.switch\_to.default\_content()

这个方法会切换回你刚进入该页面时的HTML页面。此时如果想选择id为“add”的div元素，就可以使用

wd.find\_element\_by\_css\_selector("div#add")

### 多层切换

frame可以有嵌套关系，所以selenium提供了切换到父frame的方法，这样方便查找上层frame。

wd.switch\_to.parent\_frame()

对于本例，1.2主切回默认HTML内容，也可以使用该方法，因为iframe的父frame刚好是默认的HTML内容。

补充：

在WebDriver的方法中，还有switch\_to\_frame方法，此方法已被switch\_to.frame替代，建议使用后者。

## 切换窗口

很多链接会打开一个新窗口。如果需要操作跳转后的窗口，就需要知道怎么判断当前窗口，一般使用WebDriver的title属性。

跳转新窗口的操作使用WebDriver的对象的switch\_to属性的window方法。

wd.switch\_to.window(handle)

其中handle是窗口句柄，可以理解为当前打开的一个浏览器标签。每个WebDriver对象都有一个window\_handles属性，它是一个可迭代对象，里面包含所有可跳转的窗口句柄（handle）。

如下是切换窗口的部分代码

cur\_win = wd.current\_window\_handle  
all\_handles = wd.window\_handles  
for handle in all\_handles:  
 if handle != cur\_win:  
 wd.switch\_to.window(handle)

其中current\_window\_handle为当前窗口的handle。以上代码切换了不同的标签。

切换窗口还可以使用索引的方式，比如你新打开的窗口为最后一个，你可以这样写

wd.switch\_to.window(all\_handles[-1])

有了以上的介绍，切换回最初的窗口也很简单，可以把current\_window\_handle保存下来，在需要切换时直接使用保存的handle即可。

# 保存最初页面的handle  
main\_handle = wd.current\_window\_handle  
# 跳转到其它标签页  
wd.switch\_to.window(all\_handles[1])  
# 其它页面操作结束后，跳转会主页面继续操作  
wd.switch\_to.window(main\_handle)

# 操作选择框

常见的选择框包括： radio框、checkbox框、select框

## 操作radio框

radio框是多个选项的单选，想选择某个元素时，直接调用元素的click方法即可。有如下HTML内容

<div id="s\_radio">  
 <input type="radio" name="teacher" value="小江老师">小江老师<br>  
 <input type="radio" name="teacher" value="小雷老师">小雷老师<br>  
 <input type="radio" name="teacher" value="小凯老师" checked="checked">小凯老师  
</div>

当前选中的是小楷老师，如果需要切换选中小雷老师，操作如下

wd.find\_element\_by\_css\_selector('#s\_radio input[value="小雷老师"]').click()

## 操作checkbox框

checkbox是复选框，有多个选项，也可同时选择多个，每个选项点击切换选中状态。有如下HTML内容

<div id="s\_checkbox">  
 <input type="checkbox" name="teacher" value="小江老师">小江老师<br>  
 <input type="checkbox" name="teacher" value="小雷老师">小雷老师<br>  
 <input type="checkbox" name="teacher" value="小凯老师" checked="checked">小凯老师  
</div>

如果想要同时选中小楷老师和小雷老师，操作如下

# 获取元素，这里css selector写的比较详细  
elements = wd.find\_elemnts\_by\_css\_selector('div#s\_checkbox > input[name="teacher"]')  
for element in elements:  
 if element.get\_attribute("value") in ("小雷老师", "小楷老师"):  
 # 一定要判断checkbox元素原来的状态再决定是否点击。  
 if element.get\_attribute("checked") != "checked":  
 element.click()

## 操作select框

radio框及checkbox框都是input元素，只是里面的type不同而已，select框则是一个新的select标签。

对于Select 选择框， Selenium 专门提供了一个 Select类 进行操作。通过传递一个元素来创建一个Select实例

from selenium.webdriver.support.ui import Select

# 省略部分代码，element是一个select标签对象  
select = Select(element)

有以下常用方法

* select\_by\_value

根据选项的 value属性值 ，选择元素。比如，下面的HTML

<option value="foo">Bar</option>

就可以根据 foo 这个值选择该选项，

select.select\_by\_value('foo')

* select\_by\_index

根据选项的 次序 （从0开始），选择元素。如选择第2个选项（index为1）

select.select\_by\_index(1)

* select\_by\_visible\_text

根据选项的 可见文本 ，选择元素。比如，选择显示内容为“visible content”的选项

select.select\_by\_visible\_text('visible content')

* all\_selected\_options

该方法会打印出所有已选择的选项。

* 其它

有选择就有反选，对应的方法为：deselect\_by\_value，deselect\_by\_index，deselect\_by\_visible\_text，deselect\_all（注意没有select\_all方法）。

最后，select有单选框和多选框，根据实际来组合使用以上方法。

# 页面动作

实际应用中，我们不仅需要对页面进行元素选择，输入字符串，还需要进行鼠标右键点击、双击、移动鼠标到某个元素、鼠标拖拽等操作。Selenium提供了ActionChains类来实现这些操作。

## 移动鼠标到元素

首先需要创建一个ActionChains类的实例，然后使用move\_to\_element方法来移动鼠标到指定的元素，该方法给ActionChains**添加**所要执行的动作，返回值为self。ActionChains的perform方法为执行所有设定的动作，所以还需要调用perform方法来真正**执行**操作。如下

from selenium import webdriver  
from selenium.webdriver import ActionChains  
  
wd = webdriver.Chrome()  
wd.get("https://www.baidu.com/")  
wd.maximize\_window()  
# 更多产品的元素  
element = wd.find\_element\_by\_css\_selector('a[name="tj\_briicon"]')  
ac = ActionChains(wd)

# 添加动作并执行  
ac.move\_to\_element(element).perform()

其它动作：

|  |  |  |
| --- | --- | --- |
| perform() | 执行设定的动作 |  |
| reset\_actions() | 清除设定的动作 | 与perform相反 |
| click(on\_element) | 单击元素 |  |
| double\_click(on\_element) | 双击元素 |  |
| click\_and\_hold(on\_element) | 长按元素 |  |
| context\_click(on\_element) | 右键菜单 |  |
| drag\_and\_drop(source\_element, target\_element) | 拖拽元素 | 在第一个元素按下鼠标，拖拽到目标元素后，松开鼠标。 |
| key\_down(value, element) | 按住键 | 一般是control，alt等组合键 |
| key\_up(value, element) | 松开键 | 与key\_down对应 |
| move\_by\_offset(xoffset, yoffset) | 移动位移 | 从当前位置，以偏移量来移动鼠标 |
| move\_to\_element(on\_element) | 移动到指定元素 |  |
| release(on\_element) | 松开鼠标 | 释放当前按下的鼠标按键 |
| send\_keys(\*keys\_to\_send) | 发送组合按键 | 主要用于组合按键时，按下的其它按键 |

举例，使用ctrl+c：ActionChains(driver).key\_down(Keys.CONTROL).send\_keys('c').key\_up(Keys.CONTROL).perform()

## 冻结页面（进入debugger模式）

在开发者工具栏 console 里面执行如下js代码

setTimeout(function(){debugger}, 5000)

浏览器将会在5秒后进入debugger页面，此时怎么点击界面都不会出发操作。

以下代码演示点击百度首页中，“更多产品”里的“图片”：

from selenium import webdriver  
from selenium.webdriver import ActionChains  
from time import sleep  
  
wd = webdriver.Chrome()  
wd.get("https://www.baidu.com/")  
wd.maximize\_window()  
# 更多产品的元素  
element = wd.find\_element\_by\_css\_selector('a[name="tj\_briicon"]')  
# 设置5秒后冻结页面（实际好像没有成功，并且不需要这一步）  
js = "setTimeout(function(){debugger}, 5000);"  
wd.execute\_script(js)  
# 添加并执行动作  
ac = ActionChains(wd)  
ac.move\_to\_element(element).perform()  
sleep(6)  
# 此时已显示更多产品中的图片，进行点击  
element = wd.find\_element\_by\_css\_selector('a[href\*="image"]')  
element.click()

实际不需要执行js命令也可以达到这个效果，需要继续研究一下js命令的作用以及使用方法。

execute\_script执行js命令会阻塞主进程的进行，execute\_async\_script方法是异步执行。

## 对话框

对话框有三种：Alert，Confirm，Prompt。当对话框出现时，无法操作其它元素。

获取对话框的内容

wd.switch\_to.alert.text

### Alert 弹出框

显示通知信息，只需用户看完信息后，点击 OK（确定） 就可以了。

wd.switch\_to.alert.accept()

### Confirm弹出框

主要是让用户确认是否要进行某个操作。它有两个选择供用户选择，分别是 OK 和 Cancel， 分别代表确定和取消操作，对应操作方法为accept和dismiss。

wd.switch\_to.alert.accept() # 点击确定

wd.switch\_to.alert.dismiss() # 点击取消

### Prompt对话框

主要是需要用户填写信息并提交。

填写内容使用

wd.switch\_to.alert.send\_keys()

填写内容后需要使用accept或dismiss来确认或取消操作。

## 上传文件

网站页面上传文件的功能，是通过  type 属性 为  file 的 HTML  input 元素实现的。页面元素为：

<input type="file" multiple="multiple">

通过send\_keys方法上传文件“/path/test.csv”：

# 先定位到上传文件的 input 元素  
element = wd.find\_element\_by\_css\_selector('input[type=file]')  
# 再调用 WebElement 对象的 send\_keys 方法  
element.send\_keys(r'/path/test.csv')

如果需要多个文件，则调用多次send\_keys方法。

## 更多操作

获取窗口大小

wd.get\_window\_size()

改变窗口大小

wd.set\_window\_size(pox\_x, pox\_y)

获取窗口标题

wd.title

获取当前网页地址

wd.current\_url

获取截屏，并保存为文件ScreenShot.png

wd.get\_screenshot\_as\_file('ScreenShot.png')

wd.save\_screenshot('ScreenShot.png') # 间接调用get\_screenshot\_as\_file

# 扩展：Xpath表达式

## Xpath简介

XPath (XML Path Language) 是由国际标准化组织W3C指定的，用来在 XML 和 HTML 文档中选择节点的语言。

目前主流浏览器 (chrome、firefox，edge，safari) 都支持XPath语法，xpath有 1 和 2 两个版本，目前浏览器支持的是 xpath 1的语法。

既然已经有了CSS，为什么还要学习 Xpath呢？

* 有些场景 用 css 选择web 元素 很麻烦，而xpath 却比较方便。
* 其他领域会使用到，比如 爬虫框架 Scrapy， 手机App框架 Appium。

按F12打开调试窗口，点击 Elements标签。

要验证 Xpath 语法是否能成功选择元素，也可以像 验证 CSS 语法那样，按组合键 Ctrl + F ，输入Xpath表达式来验证被选择的元素。

### 根结点

xpath 语法中，整个HTML文档根节点用’/’表示，如果我们想选择的是根节点下面的html节点，则可以在搜索框输入

/html

选择html下面的body下面的div元素。

/html/body/div

注意  / 有点像 CSS中的  > , 表示直接子节点关系。

### 绝对路径选择

从根节点开始的，到某个节点，每层都依次写下来，每层之间用  / 分隔的表达式，就是某元素的 绝对路径。

上面的xpath表达式 /html/body/div ，就是一个绝对路径的xpath表达式， 等价于 css表达式 html>body>div

自动化程序要使用Xpath来选择web元素，应该调用 WebDriver对象的方法 find\_element\_by\_xpath 或者 find\_elements\_by\_xpath，像这样：

elements = wd.find\_elements\_by\_xpath("/html/body/div")

### 相对路径选择

在xpath前面加 // , 表示从当前节点往下寻找所有的后代元素,不管它在什么位置。比如，选择示例页面的所有标签名为 div 的元素，如果使用css表达式，直接写一个 div 就行了。Xpath表达式为

//div

选择 所有的 div 元素里面的 所有的 p 元素 ，不管div 在什么位置，也不管p元素在div下面的什么位置，则可以这样写

elements = wd.find\_elements\_by\_xpath("//div//p")

相对应的css则写为

elements = wd.find\_elements\_by\_css\_selector("div p")

### 通配符

如果要选择所有div节点的所有直接子节点，可以使用表达式 //div/\*。\* 是一个通配符，对应任意节点名的元素，等价于CSS选择器  div > \*。

## 根据属性选择

根据属性来选择元素 是通过 这种格式来的  [@属性名='属性值']

注意：

* 属性名注意前面有个@
* 属性值一定要用引号， 可以是单引号，也可以是双引号

### 根据id属性选择

选择 id 为 west 的元素，可以这样  //\*[@id='west']

### 根据class属性选择

选择所有 select 元素中 class为 single\_choice 的元素，可以这样

 //select[@class='single\_choice']

如果一个元素class 有多个，比如

<p id="beijing" class='capital huge-city'>北京</p>

如果要选它，对应的 xpath 就应该是

//p[@class="capital huge-city"]

不能只写一个属性，像这样 //p[@class="capital"] 则不行。

### 根据其他属性

同样的道理，我们也可以利用其它的属性选择，比如选择具有multiple属性的所有页面元素 ，可以这样

//\*[@multiple]

### 属性值包含字符串

要选择style属性值包含color字符串的页面元素，可以这样

//\*[contains(@style,'color')]

要选择style属性值以color字符串开头的页面元素，可以这样

//\*[starts-with(@style,'color')]

要选择style属性值以某个字符串结尾的页面元素，大家可以推测是 //\*[ends-with(@style,'color')]， 但是，很遗憾，这是xpath 2.0 的语法 ，目前浏览器都不支持。

### 扩展：使用相对目录的注意点

如果我们需要选择id为“kw”的元素的所有p元素，你可能会写为“//p”，但这样会找到页面中所有的p元素，正确的xpath表达式为“.//p”。如下

kw\_ele = wd.find\_element\_by\_id("kw")  
p\_eles = kw\_ele.find\_elements\_by\_xpath(".//p")

## 根据元素次序选择

xpath也可以根据次序选择元素，直接在方括号中使用数字表示次序。

### 正序选择

假设有如下HTML内容：

<body>  
 <div *id*="t1">  
 <h3> 唐诗 </h3>  
 <span>李白</span>  
 <p>静夜思</p>  
 <span>杜甫</span>  
 <p>春夜喜雨</p>  
 </div>  
 <div *id*="t2">  
 <h3> 宋词 </h3>  
 <span>苏轼</span>  
 <p>赤壁怀古</p>  
 <p>明月几时有</p>  
 <p>江城子·乙卯正月二十日夜记梦</p>  
 <p>蝶恋花·春景</p>  
 <span>辛弃疾</span>  
 <p>京口北固亭怀古</p>  
 <p>青玉案·元夕</p>  
 <p>西江月·夜行黄沙道中</p>  
 </div>  
</body>

要选择 第2个类型为p的元素，就是

//p[2]

结果为:

春夜喜雨

明月几时有

注意，不是第2个子元素，并且是p类型 。

要选取父元素为div 中的 p类型 第2个 子元素

//div/p[2]

结果为：

春夜喜雨

明月几时有

选择父元素为div的第4个子元素，不管是什么类型

//div/\*[4]

结果为：

杜甫

明月几时有

### 倒序选择

选取p类型倒数第1个子元素

//p[last()]

选取p类型倒数第2个子元素

//p[last()-1]

结果为：

静夜思

青玉案·元夕

选择父元素为div中p类型倒数第3个子元素

//div/p[last()-2]

### 范围选择

xpath还可以选择子元素的次序范围。

选取option类型第1到2个子元素

//option[position()<=2]

或者

//option[position()<3]

选择class属性为multi\_choice的前3个子元素

//\*[@class='multi\_choice']/\*[position()<=3]

选择class属性为multi\_choice的后3个子元素

//\*[@class='multi\_choice']/\*[position()>=last()-2]

## 组选择、父节点、兄弟节点

### 组选择

xpath也有组选择，是用竖线隔开多个表达式。

要选所有的option元素 和所有的 h4 元素，可以使用

//option | //h4

等同于CSS选择器

option , h4

要选所有的 class 为 single\_choice 和 class 为 multi\_choice 的元素，可以使用

//\*[@class='single\_choice'] | //\*[@class='multi\_choice']

等同于CSS选择器

.single\_choice , .multi\_choice

### 选择父节点

xpath可以选择父节点， 这是css做不到的。某个元素的父节点用  /.. 表示。

要选择 id 为 china 的节点的父节点，可以这样写

//\*[@id='china']/..

当某个元素没有特征可以直接选择，但是它有子节点有特征，就可以采用这种方法，先选择子节点，再指定父节点。

### 兄弟节点选择

xpath也可以选择后续兄弟节点，用这样的语法  following-sibling::

要选择 class 为 single\_choice 的元素的所有后续兄弟节点

//\*[@class='single\_choice']/following-sibling::\*

等同于CSS选择器 .single\_choice ~ \*

要选择后续节点中的div节点，就应该这样写

//\*[@class='single\_choice']/following-sibling::div

xpath还可以选择 前面的 兄弟节点，用这样的语法  preceding-sibling::。要选择 class 为 single\_choice 的元素的所有前面的兄弟节点

//\*[@class='single\_choice']/preceding-sibling::\*

而CSS选择器目前还没有方法选择前面的兄弟节点

# 其它知识点

## 无界面运行

Chrome有一个option参数（之前为chrome\_options），它接收一个ChromeOptions对象，给该对象添加argument=“headless”，则运行时不会打开浏览器，后台运行代码。

option = webdriver.ChromeOptions()  
option.add\_argument(argument="headless")  
# 加载webDriver之前定义无图显示，并将其添加到webDriver中。  
wd = webdriver.Chrome(chrome\_options=option)