

# day04-课堂笔记

day04-课堂笔记

1. lxml的使用
  - 1.1 lxml模块的介绍与安装
  - 1.2 lxml的基本使用
  - 1.3 lxml的进阶使用
2. 案例
3. selenium的介绍和环境搭建
  - 3.1 环境搭建
4. selenium的基本使用
5. selenium定位标签和获取数据的方法
6. selenium的案例

## 1. lxml的使用

### 1.1 lxml模块的介绍与安装

前面我们在学习xpath的时候，使用的是浏览器中的一个插件来学习xpath语法的，但是呢，我们要写的是一个爬虫的程序，也就是说后去的使用xpath语法还是要在Python代码中去使用。

如何在Python代码中使用xpath来获取数据？就要去借助lxml模块了，

lxml就是可以帮助我们在Python代码中去书写xpath语法去提取数据的模块。

lxml是一个第三方模块，在使用之前要先进行安装

```
1 | pip install lxml
```

```
(spider_py3) E:\爬虫-左文彬\day04-lxml和selenium\02-课堂代码>pip install lxml
Collecting lxml
  Downloading lxml-4.6.3-cp39-cp39-win_amd64.whl (3.5 MB)
    |#####| 3.5 MB 1.3 MB/s
Installing collected packages: lxml
Successfully installed lxml-4.6.3

(spider_py3) E:\爬虫-左文彬\day04-lxml和selenium\02-课堂代码>
```

### 1.2 lxml的基本使用

使用步骤：

- 导包： `from lxml import etree`
- 使用 `element = etree.HTML(响应的html的字符串类型/响应的html的bytes类型)`
- 可以使用 `element.xpath("在浏览器中是如何写的xpath就在这里怎么写")` 返回的数据是一个列表

to\_string() 方法的使用：

- `etree.HTML()` 可以将html字符串转换为element对象
- `to_string()` 可以将element对象转换为字符串。

```
1 | from lxml import etree
```

```

2
3 html_str = ""
4 <ul>
5     <li class="item-1"><a href="baidu.com">百度一下</a></li>
6     <li class="item-2"><a href="sina.com">新浪</a></li>
7     <li class="item-3"><a href="163.com">网易</a></li>
8     <li class="item-4"><a href="google.com">谷歌</a></li>
9     <li class="item-5"><a href="qq.com">腾讯</a>
10 </ul>
11 ""
12
13 # 将 响应的内容 转换为 element对象
14 element = etree.HTML(html_str)
15
16 print(element)
17
18 # 现在已经将 html 字符串 转换为了 element对象， 那么我们如何将这个element 对象 再转换
    回 str
19 print(etree.tostring(element).decode())

```

## 基本使用

```

1 from lxml import etree
2
3 # //a/text()
4 html_str = ""
5 <ul>
6     <li class="item-1"><a href="baidu.com">百度一下</a></li>
7     <li class="item-2"><a href="sina.com">新浪</a></li>
8     <li class="item-3"><a href="163.com">网易</a></li>
9     <li class="item-4"><a href="google.com">谷歌</a></li>
10    <li class="item-5"><a href="qq.com">腾讯</a>
11 </ul>
12 ""
13
14 # 将 响应的内容 转换为 element对象
15 element = etree.HTML(html_str)
16
17 # print(element)
18 #
19 # # 现在已经将 html 字符串 转换为了 element对象， 那么我们如何将这个element 对象 再
    转换回 str
20 # print(etree.tostring(element).decode())
21
22 # 需求： 获取到所有的 网址的标题    [百度一下， 新浪， xx， x， xx]
23 title_list = element.xpath("//a/text()")
24 print(title_list)

```

```

1 from lxml import etree
2 # pretty
3 from pprint import pprint
4
5 # //a/text()
6 html_str = ""
7 <ul>
8     <li class="item-1"><a href="baidu.com">百度一下</a></li>

```

```

9     <li class="item-2"><a href="sina.com">新浪</a></li>
10    <li class="item-3"><a href="163.com">网易</a></li>
11    <li class="item-4"><a href="google.com">谷歌</a></li>
12    <li class="item-5"><a href="qq.com">腾讯</a>
13 </ul>
14 """
15
16 # 将 响应的内容 转换为 element对象
17 element = etree.HTML(html_str)
18
19 # print(element)
20 #
21 # # 现在已经将 html 字符串 转换为了 element对象， 那么我们如何将这个element 对象 再
    转换回 str
22 # print(etree.tostring(element).decode())
23
24 # 需求： 获取到所有的 网址的标题    [百度一下， 新浪， xx， x， xx]
25 title_list = element.xpath("//a/text()")
26 print(title_list)
27
28 # 需求： 获取到所有的 网站对应的 具体的url地址
29 # 获取到所有的a标签中的 href属性的值。
30 href_list = element.xpath("//a/@href")
31 print(href_list)
32
33 # 需求： 经过上述的操作，我们已经获取到了所以网站的网址的名字，网站的具体的url地址
34 # 数据要合到一起。 数据的格式要按照以下的格式来合并
35 """
36 [
37     {"title": "百度一下", "href": "baidu.com"},
38     {"title": "新浪", "href": "sina.com"},
39     {"title": "网易", "href": "163.com"}
40 ]
41 """
42 # 先去定义一个列表
43 data_list = []
44 # 遍历 title_list
45 for title in title_list:
46     # 准备一个空字典
47     item = {}
48     # 遍历的是 title_list ，可以 获取一下 当前遍历的时候 title在 title_list 中的下标
49     index = title_list.index(title)
50     print(index)
51     item["title"] = title
52     item["href"] = href_list[index]
53     data_list.append(item)
54 pprint(data_list)

```

## 1.3 lxml的进阶使用

先分组 再提取

```

1 from lxml import etree
2 # pretty
3 from pprint import pprint

```

```

4
5 # //a/text()
6 html_str = ""
7 <ul>
8     <li class="item-1"><a href="baidu.com">百度一下</a></li>
9     <li class="item-2"><a href="sina.com">新浪</a></li>
10    <li class="item-3"><a href="163.com">网易</a></li>
11    <li class="item-4"><a href="google.com">谷歌</a></li>
12    <li class="item-5"><a>腾讯</a>
13 </ul>
14 ""
15 # [<a href="baidu.com">百度一下</a>, <a href="baidu.com">新浪</a>, <a
href="baidu.com">网易</a>]
16 ""
17 [
18     {"title": "百度一下", "href": "baidu.com"},
19     {"title": "新浪", "href": "sina.com"},
20     {"title": "网易", "href": "163.com"}
21 ]
22 ""
23
24 # 将 响应的内容 转换为 element对象
25 element = etree.HTML(html_str)
26
27 # 不去获取两个列表了。 可以将数据进行分组，分组完毕之后 我再去获取每一条数据。
28 # 先分组： 把所有的a标签获取到
29 a_list = element.xpath("//a")
30 # 准备一个空列表
31 data_list = []
32 # 遍历
33 for a in a_list:
34     # 准备一个空字典，
35     item = {}
36     # 遍历出来的每一个a 是一个 element对象 可以继续去调用xpath方法
37     # . 表示的就是当前 我们遍历出来的每一个 a标签
38     item["title"] = a.xpath("./text()")[0]
39     href = a.xpath("./@href")
40     # 使用三目运算
41     item["href"] = href[0] if len(href)>0 else None
42     # if len(href) > 0:
43     #     item["href"] = href[0]
44     # else:
45     #     item["href"] = None
46     print(item)

```

## 2. 案例

需求：要求大家去抓取 <https://www.sxpi.edu.cn/xwzx/gyyw.htm> 这个url地址下的 所有的新闻的标题以及发布日期

数据的格式：

```

1  [
2      {"title": "陕西工院“三全”举措做好迎新工作", "time": "2021/09/02"},
3      {"title": "陕西工院“三全”举措做好迎新工作", "time": "2021/09/02"},
4      {"title": "陕西工院“三全”举措做好迎新工作", "time": "2021/09/02"},
5  ]

```

保存数据，将整个数据的列表保存到一个 data.json

```

1  import requests, json
2  from lxml import etree
3
4
5  # 1. 准备url地址
6  url = "https://www.sxpi.edu.cn/xwzx/gyyw.htm"
7  headers = {
8      "User-Agent": "Mozilla/5.0 (windows NT 10.0; win64; x64)
9      AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36"
10 }
11
12 # 2. 发送请求 获取响应
13 response = requests.get(url, headers=headers)
14
15 # 3. 提取数据
16 # 要将响应的内容 转换为 element对象
17 element = etree.HTML(response.content)
18 # 先分组 再提取
19 li_list = element.xpath("//div[@class='list_box']/ul/li")
20 # 遍历每一个li标签 去获取每一条数据
21 data_list = []
22 for li in li_list:
23     item = {}
24     # 获取新闻的标题
25     item["title"] = li.xpath("./a/text()")[0]
26     # 获取新闻的发布日期
27     item["time"] = li.xpath("./span/text()")[0]
28     data_list.append(item)
29
30 # 4. 保存数据
31 with open("data.json", "w", encoding="utf-8") as f:
32     json.dump(data_list, f, ensure_ascii=False, indent=2)

```

### 3. selenium的介绍和环境搭建

selenium 是一个web自动化测试的工具，最开始设计出来的时候，是专门用于软件测试，可以接受一些指令，操作浏览器：打开某个页面，获取页面中的数据

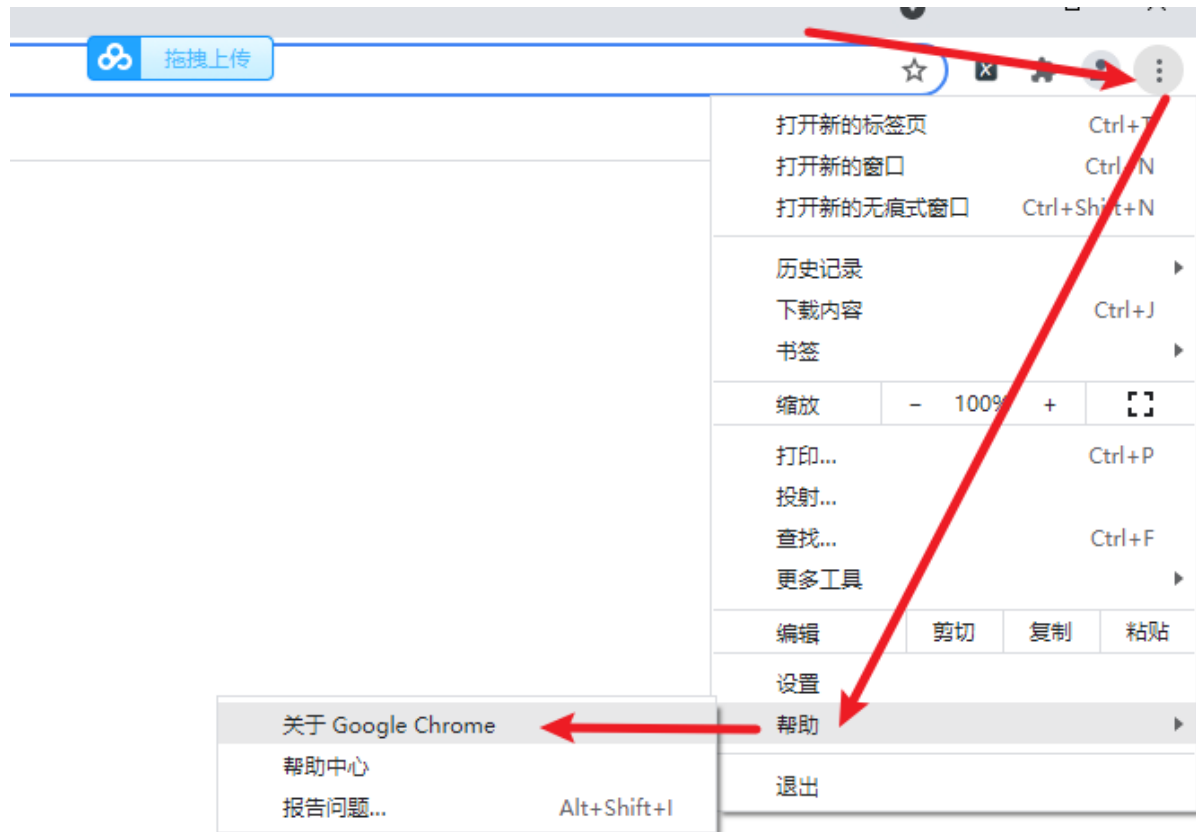
并且可以去操作页面上的标签，截屏。

分类：

- 有头模式：指的是有界面。操作我们是看到的到的。
- 无头模式：指的就是没有界面，操作都是在内存中完成，操作步骤我们是看不到的。

### 3.1 环境搭建

① 打开自己的谷歌浏览器，查看一下自己谷歌浏览器的版本号



② 打开网址：<http://npm.taobao.org/mirrors/chromedriver/>，下载驱动程序，帮助我们去操作浏览器。

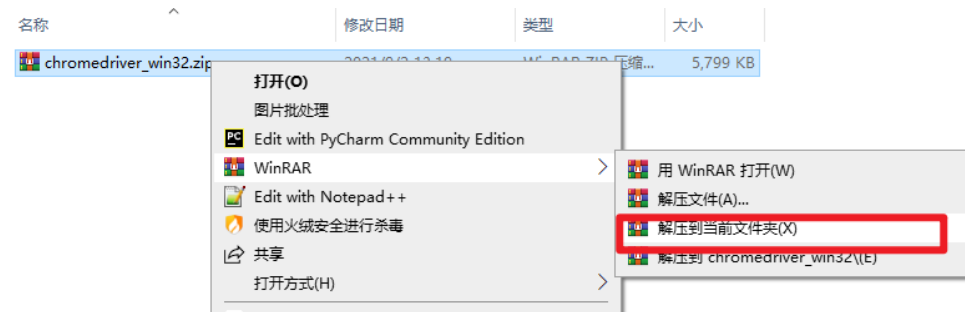


### ③ 下载对应操作系统的驱动，win

../		
chromedriver_linux64.zip	2021-07-29T06:57:55.113Z	5980285(5.7MB)
chromedriver_mac64.zip	2021-07-29T06:57:58.171Z	8118810(7.74MB)
chromedriver_mac64_m1.zip	2021-07-29T06:58:00.565Z	7432011(7.09MB)
chromedriver_win32.zip	2021-07-29T06:58:02.897Z	5938088(5.66MB)
notes.txt	2021-07-29T06:58:07.962Z	156(156B)

名称	修改日期	类型	大小
chromedriver_win32.zip	2021/9/2 13:19	WinRAR ZIP 压缩...	5,799 KB

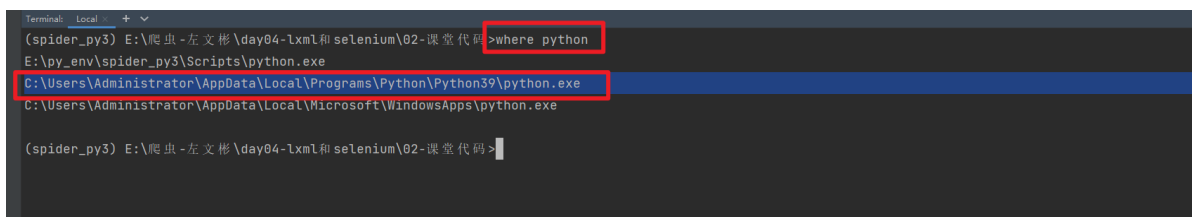
### ④ 解压这个压缩包

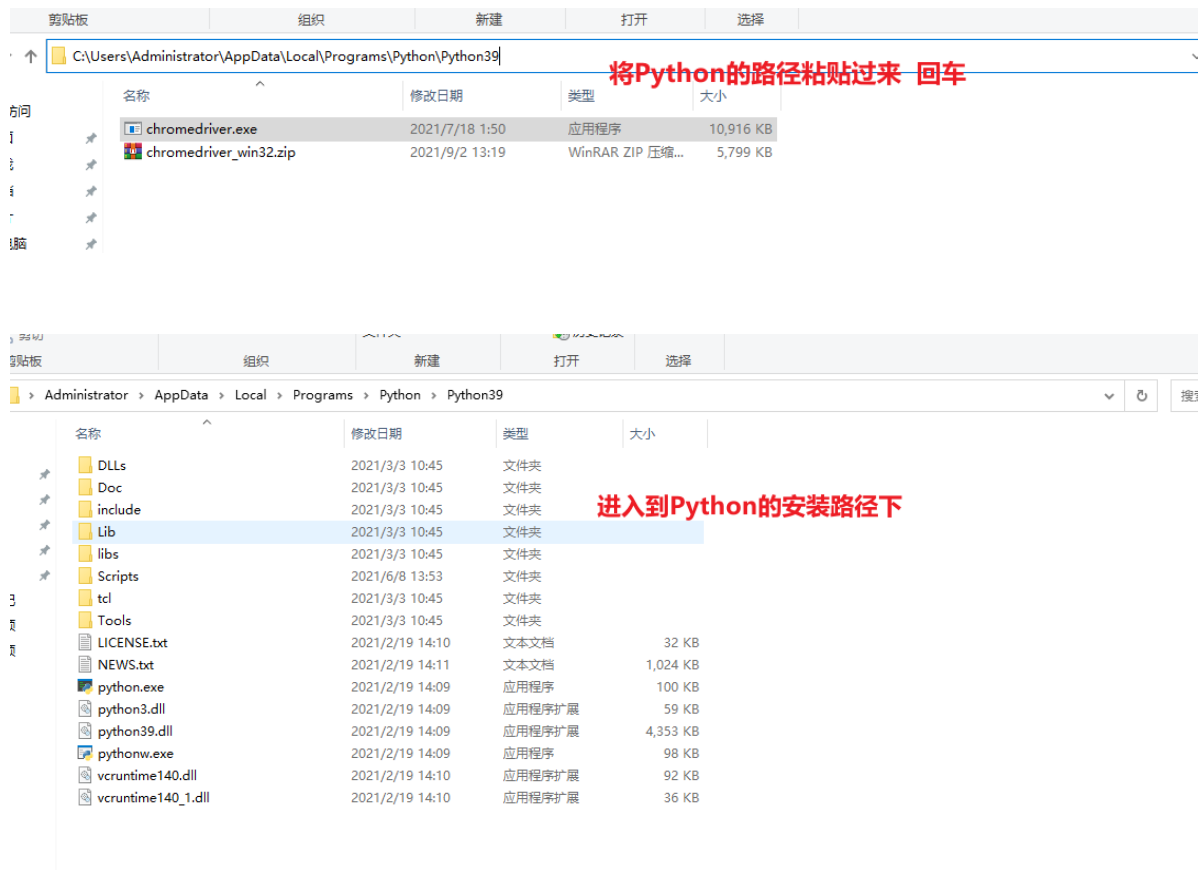


名称	修改日期	类型	大小
chromedriver.exe	2021/7/18 1:50	应用程序	10,916 KB
chromedriver_win32.zip	2021/9/2 13:19	WinRAR ZIP 压缩...	5,799 KB

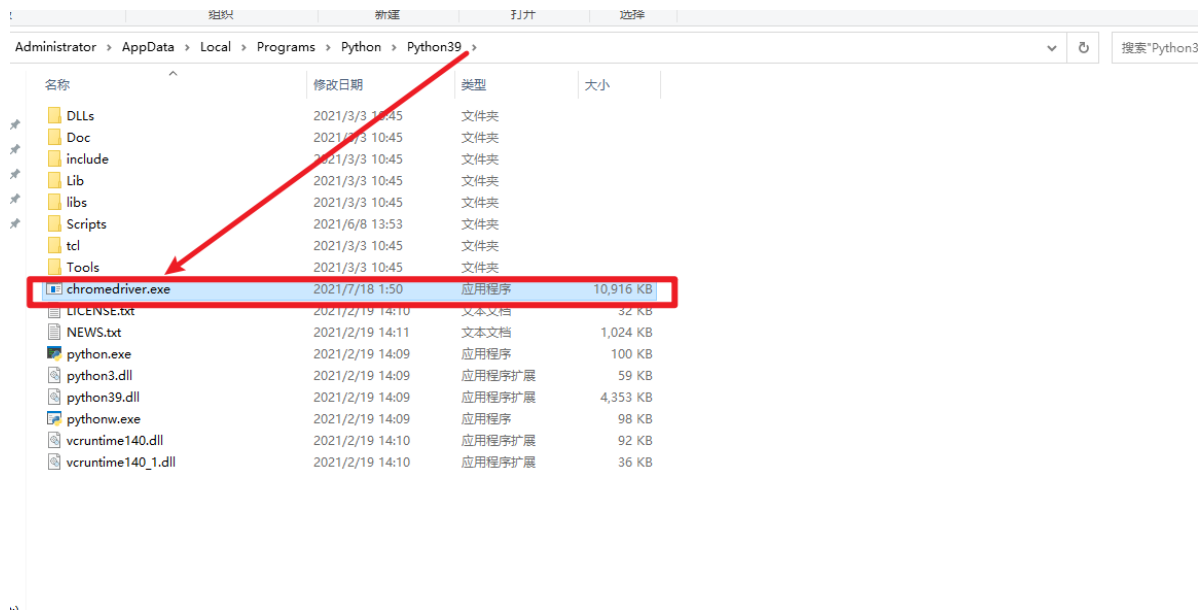
解压之后会得到一个exe可执行程序

### ⑤ 找到自己的电脑上Python的安装路径

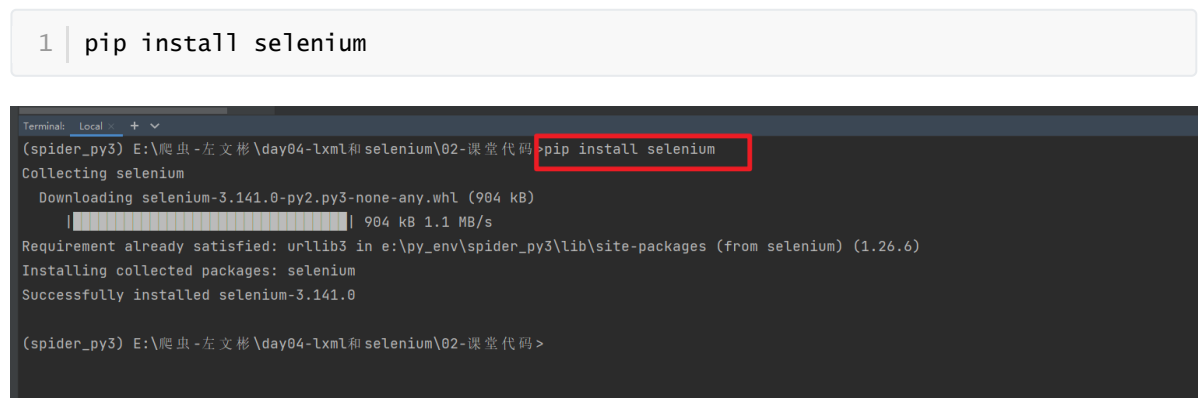




⑥ 将刚才解压出来的 chromedriver.exe 文件 复制 粘贴到 Python的安装路径下



⑦ 在Python环境中 安装一个模块， selenium





## 4. selenium的基本使用

---

操作步骤：

- 导包： `from selenium import webdriver`
- 创建driver对象： `driver = webdriver.Chrome()`
- 调用get方法传入一个url地址，相当于使用浏览器打开了一个网站： `driver.get("http://www.baidu.com")`
- 调用 driver对象中的方法，进行对页面的操作。
- 所有的操作操作完毕之后，记得退出浏览器： `driver.quit()`

```
1  # 1.导包
2  import time
3
4  from selenium import webdriver
5
6  # 2. 创建driver对象
7  driver = webdriver.Chrome()
8
9  # 窗口最大化
10 driver.maximize_window()
11
12 # 3. 调用get方法传入url地址 就可以打开这个网站
13 driver.get("https://www.sxpi.edu.cn/")
14
15 # 截图
16 driver.save_screenshot("sxpi.png")
17
18 time.sleep(5)
19
20 # 4. 操作都完毕之后，退出浏览器
21 driver.quit()
```

## 5. selenium定位标签和获取数据的方法

---

<code>driver.find_element</code>	
<code>find_element(self, by, value)</code>	WebDriver
<code>find_element_by_id(self, id_)</code>	通过标签中的id属性的值去定位标签 WebDriver
<code>find_element_by_link_text(self, link_text)</code>	根据标签中的文本内容去定位标签 WebDriver
<code>find_elements(self, by, value)</code>	
<code>find_element_by_name(self, name)</code>	通过标签中的name属性的值去定位标签 WebDriver
<code>find_element_by_class_name(self, name)</code>	通过标签中的class属性的值去定位标签 WebDriver
<code>find_element_by_css_selector(self, css_selector)</code>	通过标签的css选择器去定位标签 WebDriver
<code>find_element_by_partial_link_text(self, link_text)</code>	通过标签中包含的文本内容去定位标签 WebDriver
<code>find_element_by_tag_name(self, name)</code>	通过标签的名字去定位标签 WebDriver
<code>find_element_by_xpath(self, xpath)</code>	通过标签的xpath路径表达式去定位标签 WebDriver
<code>find_elements_by_class_name(self, name)</code>	
<code>find_elements_by_css_selector(self, css_selector)</code>	
<code>find_elements_by_id(self, id_)</code>	
<code>find_elements_by_link_text(self, text)</code>	find_element 获取的一个标签 find_elements 获取多个 WebDriver
<code>find_elements_by_name(self, name)</code>	WebDriver
<code>find_elements_by_partial_link_text(self, link_text)</code>	WebDriver
<code>find_elements_by_tag_name(self, name)</code>	WebDriver
<code>find_elements_by_xpath(self, xpath)</code>	WebDriver

```

1  import time
2
3  from selenium import webdriver
4
5  # 1. 创建driver对象
6  driver = webdriver.Chrome()
7
8  # 2. 打开页面
9  driver.maximize_window()
10 driver.get("http://www.baidu.com")
11
12 # 向百度的输入框中输入内容： 学校的名字
13 # 想要向输入框中输入内容，先定位到输入框 定位到之后 再去输入内容
14 # send_keys("字符串") 表示向 输入框中输入文本内容
15 # driver.find_element_by_id("kw").send_keys("abcdefg")
16 # driver.find_element_by_name("wd").send_keys("abc")
17 time.sleep(5)
18 driver.find_element_by_class_name('s_ipt').send_keys('abc')
19 time.sleep(5)
20
21 # 点击 搜索的按钮
22 # 先定位到 搜索的按钮， 再去点击
23 driver.find_element_by_id('su').click()
24
25 time.sleep(3)
26
27 # 退出浏览器
28 driver.quit()

```

```

1  import time
2
3  from selenium import webdriver
4
5  # 1. 创建driver对象
6  driver = webdriver.Chrome()
7
8
9  # 2. 打开页面

```

```

10 driver.get("https://www.jd.com/")
11
12 # 3. 操作: 向 输入框中输入 笔记本电脑 点击搜索
13 # 定位到输入框
14 driver.find_element_by_xpath('//input[@id="key"]').send_keys("笔记本电脑")
15 # driver.find_element_by_id('key')
16 time.sleep(5)
17
18 # 点击搜索按钮
19 driver.find_element_by_xpath("//button[@class='button']").click()
20 time.sleep(3)
21
22 # 截图
23 driver.save_screenshot('jd.png')
24
25 # 退出
26 driver.quit()

```

## 6. selenium的案例

爬虫: 模拟客户端发送请求 获取响应。

selenium: 打开浏览器 打开页面 (发送请求 获取响应)

需求: 使用selenium去完成爬虫的编写,

- 目标网址: <https://wz.sun0769.com/political/index/politicsNewest>
- 数据内容:
  - 编号
  - 状态
  - 标题
  - 发布时间
  - 响应时间
- 数据的格式: `[{}, {}, {}]`

```

1  from selenium import webdriver
2  import json
3
4  # 1. 准备url地址
5  url = "https://wz.sun0769.com/political/index/politicsNewest"
6
7  # 2. 发送请求 获取响应(使用selenium打开页面) 要求的是使用selenium
8  # 创建一个driver对象
9  driver = webdriver.Chrome()
10 driver.maximize_window()
11 driver.get(url)
12
13 # 3. 提取数据
14 # 先使用driver定位到所有的 包含数据的标签 li
15 # 使用 find_elemnt 返回的是 WebElement 对象
16 # 先分组
17 li_list = driver.find_elements_by_xpath("//ul[@class='title-state-ul']/li")
18 # 准备空列表 用于保存每一条数据 (字典)
19 data_list = []
20 # 遍历获取每一个li里面的数据的内容, 获取每一条数据
21 for li in li_list:

```

```

22     item = {}
23     # 在遍历的时候 获取数据的时候因为是一条数据， 所以这个地方我们要使用 find_element
24     # 注意点： 在selenium中使用xpath的时候 只能写到定位标签 不能在xpath中书写
    text() 以及 @属性名
25     # 在selenium中 要获取标签中的文本内容要使用 text来获取，
26     item["id"] = li.find_element_by_xpath("./span[1]").text
27     item["status"] = li.find_element_by_xpath("./span[2]").text
28     item["title"] = li.find_element_by_xpath("./span[3]/a").text
29     # get_attribute("属性名") 表示根据属性的名字去获取标签中的属性的值。
30     item["detail_href"] =
    li.find_element_by_xpath("./span[3]/a").get_attribute("href")
31     item["response_time"] = li.find_element_by_xpath("./span[4]").text
32     item["release_time"] = li.find_element_by_xpath("./span[5]").text
33     data_list.append(item)
34 # 4. 保存数据 保存到 sun.json
35 with open('sun.json', "w", encoding="utf-8") as f:
36     json.dump(data_list, f, ensure_ascii=False, indent=2)
37
38
39 # 退出浏览器
40 driver.quit()

```

网站: <https://www.giushibaike.com/text/>