元素定位

一、自动化的基本常识：

1、什么是UI自动化

UI自动化测试，都是基于元素的定位

而元素的定位时基于当前屏幕范围内的可见元素（眼睛可以看到的内容，通过手动/代码去操作）

通过页面中唯一存在的元素属性去进行定位

UI自动化测试的步骤：

- 导入包：selenium包等
- 选择浏览器驱动
- 通过驱动浏览网页
- 定位元素并执行操作

2、安装selenium库

在设置--解释器中--下载selenium库

或者在cmd命令界面中执行：python -m pip install --upgrade selenium 来进行下载

注意：

如果提示pip 版本太低，则先执行python -m pip install --upgrade pip 更新一下pip，然后再安装selenium的命令

3、下载浏览器驱动

- 浏览器驱动需要根据浏览器的版本来下载

  最新的浏览器版本要下载要下载最新的驱动，或者驱动版本要和浏览器版本相对应
- 将下载好的驱动解压出来，可以看到一个文件，将这个文件复制到python的路径中

4、在python中打开浏览器

```
from selenium import webdriver

# deiver=webdriver.Firefox()    #调用火狐的驱动
driver=webdriver.Chrome()        #调用谷歌的驱动
driver.get('https://www.baidu.com/')  #使用驱动打开百度的页面
```

二、元素定位

1、使用ID定位

在网页的元素中，根据ID来定位

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包

# deiver=webdriver.Firefox()   #调用火狐的驱动
driver=webdriver.Chrome()       #调用谷歌的驱动
# driver.get('https://www.baidu.com/')  #使用驱动打开百度的页面

driver.get('https://www.baidu.com')  #使用驱动打开百度的页面
driver.find_element(By.ID,'kw').send_keys('刘德华')  #定位到id为kw的元素，并且使用
send_keys方法输入内容

driver.get('http://192.168.85.128/fanwe')
driver.find_element(By.ID,'login-email-address').send_keys('周四')
```

2、使用name定位:

在网页的元素中，根据name来定位

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包

# deiver=webdriver.Firefox()   #调用火狐的驱动
driver=webdriver.Chrome()       #调用谷歌的驱动
driver.get('https://www.baidu.com/')  #使用驱动打开百度的页面
driver.find_element(By.NAME,'wd').send_keys('刘德华')  #通过name的方式来定位元素
driver.find_element(By.ID,'su').click()     #click是点击操作
```

3、通过class定位

通过class定位元素

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time

# deiver=webdriver.Firefox()   #调用火狐的驱动
driver=webdriver.Chrome()       #调用谷歌的驱动
# driver.get('https://www.baidu.com/')  #使用驱动打开百度的页面

driver.get('https://www.baidu.com')  #使用驱动打开百度的页面
driver.find_element(By.CLASS_NAME,'s_ipt').send_keys('刘德华')  #定位到class_name为
kw的元素，并且使用send_keys方法输入内容

time.sleep(10)      #执行到这个代码时，会等待10秒之后在继续执行代码
driver.close()      #关闭网页
```

4、使用link定位超链接--1

a标签为超链接，定位方法可以用，like_text : 需要传入标签全部文本内容进行定位

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time

# deiver=webdriver.Firefox()   #调用火狐的驱动
driver=webdriver.Chrome()        #调用谷歌的驱动
# driver.get('https://www.baidu.com/')  #使用驱动打开百度的页面

driver.get('https://www.baidu.com')  #使用驱动打开百度的页面
driver.find_element(By.LINK_TEXT,'新闻').click()  #使用LINK_TEXT来定位，标签内的文本要全部写完

time.sleep(10)       #执行到这个代码时，会等待10秒之后在继续执行代码
driver.close()  #关闭网页
```

5、使用link定位超链接--2

a标签为超链接，定位方法可以用，PARTIAL_LINK_TEXT : 不需要写入全部文本

注意:

a标签中的文本内容也需要是存在或唯一时才可以使用

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time

# deiver=webdriver.Firefox()   #调用火狐的驱动
driver=webdriver.Chrome()        #调用谷歌的驱动
# driver.get('https://www.baidu.com/')  #使用驱动打开百度的页面

driver.get('https://www.baidu.com')  #使用驱动打开百度的页面
driver.find_element(By.PARTIAL_LINK_TEXT,'图').click()

aa=driver.find_elements(By.PARTIAL_LINK_TEXT,'图')
#使用PARTIAL_LINK_TEXT来定位，标签内的文本不许要全部写完，会进行匹配
aa[1].click()
#find_elements（复数）需要通过索引指定操作对应的元素

time.sleep(10)       #执行到这个代码时，会等待10秒之后在继续执行代码
driver.close()  #关闭网页
```

6、通过CSS定位元素

css（选择器）定位，几乎可以适用于所有的场景，但是也存在有些情况下无法定位的问题。

css定位的原理：如果通过 id / name / class 中的唯一值没有办法定位的时候，那么可以通过组合进行定位

谷歌：定位到元素后--右键--复制--赋值selector

火狐：定位到元素后--右键--复制--css选择器

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time

driver=webdriver.Chrome()        #调用谷歌的驱动

# driver.get('https://www.baidu.com')  #使用驱动打开百度的页面
# driver.find_element(By.CSS_SELECTOR,'#kw').send_keys('刘德华')

driver.get('http://106.52.182.140/fanwe')
driver.maximize_window()     #页面最大化
driver.find_element(By.CSS_SELECTOR,'#Iajax-login-submit').click() #点击登录按钮

time.sleep(10)       #执行到这个代码时，会等待10秒之后在继续执行代码
driver.close()       #关闭网页
```

7、通过xpath方式定位元素

Xpath（路径）定位，完美适用于所有的场景，缺点代码相对会多一些。有手就行，无脑操作

谷歌：定位元素后--右键--复制--复制xpath

火狐：定位元素后--右键--复制--xpath

相对路径：

- //*[@id="kw"]
  - // -->表示查找的元素位置为相对路径
  - * --> 匹配所有的标签，如：<input id="kw">
    - 例如：//input[@id="kw"] 表示查找 相对路径下的带有id=wk 的input标签
  - [@id="kw"] ---->属性写在中括号中，可以使用and 和 or
- 浏览器的调试模式
  - 在检查页面中切换到控制台中，使用 $x(' xpath的定位 ')

- 相关的函数：
  - text() 返回文本信息 例如：'//a[@class="f12 f_white f_l" and text()="立即注册"]'
  - 获取到元素的文本：
    - 定位.get_attribute('textContent') ---获取到文本内容
    - 定位.get_attribute('id') ---获取到元素中的id
- xpath定位学习网址：
  - https://www.w3school.com.cn/xpath/xpath_functions.asp

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time


driver=webdriver.Chrome()        #调用谷歌的驱动


# driver.get('https://www.baidu.com')  #使用驱动打开百度的页面
# driver.find_element(By.CSS_SELECTOR,'#kw').send_keys('刘德华')
driver.get('http://106.52.182.140/fanwe')
driver.maximize_window()     #页面最大化
driver.find_element(By.XPATH,'//*[@id="kw"] ').send_keys('彭于晏')

time.sleep(10)       #执行到这个代码时，会等待10秒之后在继续执行代码
driver.close()


'''
下面举例方维登录：
'''
driver.get('http://106.52.182.140/fanwe')
driver.find_element(By.XPATH,'//*[@id="login-email-address"]').send_keys('小白')
driver.find_element(By.XPATH,'//*[@id="login-password"]').send_keys('a123456')
driver.find_element(By.XPATH,'//*[@id="Iajax-login-submit"]').click()
# time.sleep(2)
```

```
# driver.find_element(By.XPATH,'//input[@type="button" and @class="dialog-cancel"]').click()
#使用xpath定位中type和class去定位元素

time.sleep(1)
driver.find_elements(By.XPATH,'//input[@type="button"]')[0].click()
#使用xpath定位返回了多个元素，然后通过索引的方式去选择需要的元素
```

8、通过tag_name方法定位元素

tag_name: 标签定位，一般标签唯一时使用，但是很多种情况下标签都不会是唯一的，那么需要通过标签索引进行定位。缺点：和麻烦，不太好用。

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time
driver=webdriver.Chrome()        #调用谷歌的驱动

driver.get('https://www.baidu.com')
driver.maximize_window()     #页面最大化
driver.find_elements(By.TAG_NAME,'input')[7].send_keys('彭于晏')
#通过匹配多个标签，然后使用索引去定位元素
#在页面中按下ctrl + f  输入内容搜索可以确定有多少个相应的标签
```

定位总结:

八大元素定位方法，如何去使用，或者说如何选择使用什么样的方法

- 需要判断元素标签是不是超链接标签
    - 如果不是：那么先查询一下是否存在唯一的id/nam/class。存在的话直接定位
    - 如果不存在唯一的id/name/class，那么可以通过css和xpath去组合定位
- 如果a标签，可以使用ling_text 或者partial_link_text 去定位。

三、时间等待

定位不到元素的原因：代码是由上到下运行的，运行的速度会很快，但是浏览的加载速度跟不上，因为需要做一些渲染等，需要一定的时间。可以添加一个等待时间来给浏览器加载完成后在进行操作

1、固定时间等待

- import time #导入time包

固定等待时间，不灵活，很死板。

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time     #导入time包
driver=webdriver.Chrome()       #调用谷歌的驱动

driver.get('http://106.52.182.140/fanwe/')
driver.maximize_window()    #页面最大化


time.sleep(3)
phon=driver.find_element(By.XPATH,'//span[@class="telep
b"]').get_attribute('textContent')
print(phon)
```

2、隐性等待时间

隐性等待时间（全局生效）

需要等待整个页面加载完成，然后再去执行代码。不需要继续等待

如果在设定的时间内，还没有全部加载完成，还是会执行代码

全局生效：同一个脚本里面，需要在前面设置一次就可以了，后面的代码都会生效

使用方法：

不需要导包

驱动实例化.driver.implicitly_wait(最长等待时长)

优点：

不死板的等待时间，可以增加测试的效率

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time
driver=webdriver.Chrome()

driver.get('http://106.52.182.140/fanwe')
driver.find_element(By.XPATH,'//*[@id="login-email-address"]').send_keys('小白')
driver.find_element(By.XPATH,'//*[@id="login-password"]').send_keys('a123456')
driver.find_element(By.XPATH,'//*[@id="Iajax-login-submit"]').click()


# time.sleep(1)
driver.implicitly_wait(10)  #等待10秒，实际只等待了0.2秒
driver.find_element(By.XPATH,'//input[@type="button"]').click()
```

四、滚动条

1、js脚本操作滚动条

使用 JS (Java Script) 进行滚动条操作：

第一步：点定位目标元素的位置 例如：

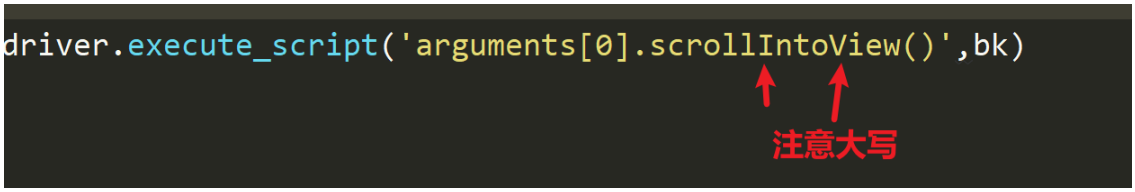
bk=driver.find_element(By.XPATH,'//*[@id="login-email-address"]')

第二步：通过js脚本 去执行

driver.execute_script()

使用具体的方法：

- 定位目标元素，可以使用八大方法定位到元素之后
- driver.execute_script(arguments[索引].scrollIntoView() , 定位的目标元素)



  - arguments[索引] -->指定元素

    索引：默认为0，通过索引来获取元素（js）

    如果只定位了一个目标元素，索引为0，如果定位了多个元素，那么索引根据实际变化
  - scrollIntoView -- >滚动/滑动到对应的元素

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time

driver=webdriver.Chrome()        #调用谷歌的驱动
driver.get('http://106.52.182.140/fanwe')
driver.maximize_window()     #页面最大化

#将页面滚动到指定元素上
driver.implicitly_wait(5)
bk=driver.find_element(By.XPATH,'//span[@class="telep b" and text()="0591-
83119192"]') #定位
driver.execute_script('arguments[0].scrollIntoView()',bk)  #滚动
```

2、直接跳转滚动

直接跳转滚动：

1、先定位元素的位置

2、直接跳转滚动

具体步骤：

1、目标元素实例化

2、目标元素变量.location_once_scrolled_into_view

location -->位置

once -->一旦

scrolled -->滚动

into_view --> 看到

```
rom selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time

driver=webdriver.Chrome()        #调用谷歌的驱动
driver.get('http://106.52.182.140/fanwe')
driver.maximize_window()    #页面最大化

#将页面滚动到指定元素上
driver.implicitly_wait(5)
bk=driver.find_element(By.XPATH,'//span[@class="telep b" and text()="0591-
83119192"]')
# driver.execute_script('arguments[0].scrollIntoView()',bk)

bk.location_once_scrolled_into_view  #直接跳转滚动操作
```

五、上传文件

如何上传文件

首先，看上传文件的标签是什么，如果是input，直接通过send_keys 输入文件的路径即可

测试上传充值页面的图片

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time
driver=webdriver.Chrome()
driver.get('http://192.168.85.128/fanwe')
driver.implicitly_wait(10)

driver.find_element(By.XPATH,'//*[@id="login-email-address"]').send_keys('借款人
1')
driver.find_element(By.XPATH,'//*[@id="login-password"]').send_keys('a123456')
driver.find_element(By.XPATH,'//*[@id="Iajax-login-submit"]').click()

driver.find_elements(By.XPATH,'//input[@type="button"]')[1].click()  #登录成功后三
方托管的弹窗

#点击充值按钮
time.sleep(2)
driver.find_element(By.XPATH,'/html/body/div[2]/div/div[2]/div[1]/div/div[3]/a[1
]').click()
```

```
#点击三方支付
driver.find_element(By.XPATH,'//*
[@id="incharge_form"]/div/div/div[1]/div[2]/label[2]').click()

#点击选择图片
driver.find_element(By.XPATH,'//*
[@id="xxxx"]/span/div[1]/div/div/button').click()


#点击本地上传
driver.find_element(By.XPATH,'/html/body/div[8]/div[1]/div[2]/div/div[1]/ul/li[2
]').click()

#上传文件
#这里要注意上传的输入框它的类型要为 type='file'
driver.find_element(By.XPATH,'//input[@type="file" and
@name="imgFile"]').send_keys(r'D:\photo\4.jpg')


#上传图片后点击确定
driver.find_element(By.XPATH,'/html/body/div[8]/div[1]/div[3]/span[1]/input').cl
ick()
```

六、下拉框

在定位下拉框的时候，需要看一下是什么标签属性

```
导入包：
from selenium.webdriver.support.ui import Select
```

下拉框定位的使用方法：

1、先定位下拉框的位置

定位后需要实例化下拉框的位置

2、在通过下拉框提供的方法来操作：

Select(实例的变量).select_by_value() #通过value的内容来选择 Select(实例的变量).select_by_visible_text() #通过文本信息来选择

Select(实例的变量).select_by_index() #通过下拉框的索引来选择

```
from selenium import webdriver
from selenium.webdriver.common.by import By  #导入元素定位方法包
import time
from selenium.webdriver.support.ui import Select  #导入下拉框的包

driver=webdriver.Chrome()        #调用谷歌的驱动
```

```
# driver.get('http://192.168.85.128/fanwe/m.php')
'''
Select(实例的变量).select_by_index()    #通过下拉框的索引来选择
Select(实例的变量).select_by_value()  #通过value的内容来选择
Select(实例的变量).select_by_visible_text()  #通过文本信息来选择
'''
driver.get("D:/program/HBuilderX/HBuilderX/readme/new_file.html")

sel=driver.find_element(By.XPATH,'/html/body/div[1]/select')
time.sleep(1)
Select(sel).select_by_visible_text('深圳')  #按照文本选择下拉框

time.sleep(1)
Select(sel).select_by_value('hebei')   #通过value的内容来选择

time.sleep(1)
Select(sel).select_by_index(1)       #通过文本信息来选
```

七、弹窗操作

对于一些无法定位的警示弹窗或二次弹窗，操作方法：切换到弹窗中，然后再做对应的操作，可以确定或取消

警示/二次弹窗：

1、先切换到弹窗中，然后实例化变量

实例化变量=driver.switch_to.alert

2、在对弹窗进行取消或确认操作

实例化变量.accept() -->确认

实例化变量.dismiss() -->取消

```
# 1、导包
from selenium import webdriver       # 导入如浏览器驱动包
from selenium.webdriver.common.by import By    # 导入元素定位方法包
from time import sleep
from selenium.webdriver.support.ui import Select  # 下拉框操作包

# 2、选择浏览器驱动
driver = webdriver.Chrome()#使用驱动访问网址


# 进入方维前台申请借款页面
driver.get("http://106.52.182.140/fanwe/")
driver.maximize_window()         # 浏览器窗口最大化
driver.get('http://192.168.85.128/fanwe')
driver.implicitly_wait(10)
driver.find_element(By.XPATH,'//*[@id="login-email-address"]').send_keys('借款人1')
driver.find_element(By.XPATH,'//*[@id="login-password"]').send_keys('a123456')
```

```
driver.find_element(By.XPATH,'//*[@id="Iajax-login-submit"]').click()
driver.find_elements(By.XPATH,'//input[@type="button"]')[1].click()  #登录成功后取消三方托管的弹窗

driver.find_element(By.XPATH,'//*[@id="header"]/div[2]/div/ul/li[3]/a').click()
driver.find_element(By.XPATH,'//*
[@id="borrowlb"]/div/ul/li[1]/div[3]/a/img').click()
sleep(1)
# 点击 HTML代码按钮
driver.find_element(By.CSS_SELECTOR,'#J_save_deal_form > div:nth-child(1) >
div:nth-child(35) > div > div > div.ke-toolbar > span:nth-child(1) >
span').click()
sleep(2)
# 定位输入框进行内容输入
driver.find_element(By.XPATH,'//*
[@id="J_save_deal_form"]/div[1]/div[35]/div/div/div[2]/textarea').send_keys("几乎是安徽hi傻还会的寄哀思搜索")
# 点击 提交审核 按钮
driver.find_element(By.ID,'publishBnt').click()


------------------弹窗操作--------------------

# 弹窗操作
sleep(3)
aa=driver.switch_to.alert
aa.dismiss()    #弹窗取消

sleep(3)
# 再次点击 提交审核 按钮调起弹窗
driver.find_element(By.ID,'publishBnt').click()

sleep(3)
aa.accept()      #弹窗确认
```

八、内嵌页面

内嵌页面：一个页面中内嵌了另外一个页面，定位和操作内嵌页面中的元素时，需要跳转到对应的内嵌页面中，再对内嵌页面中的元素进行操作

如何判断内嵌页面：frame (iframe) 这种标签表示这里有一层内嵌页面

使用方法：

浏览器驱动.driver.switch_to.frame('id/nam/class或元素的定位')

页面之间的切换：

内嵌页面只能切换到下一级内嵌页面，不能在同级和上级之中进行，

需要同级之间切换只能使用退出内嵌页面后在切换到下一级

切换的方法：

```
driver.switch_to.frame('id/nam/class或元素的定位')     #切换到对应的页面
driver.switch_to.parent_frame()  #返回到上层页面
driver.switch_to.default_content() #返回到最初的页面，退出到最外层页面
```

```
from selenium import webdriver       # 导入如浏览器驱动包
from selenium.webdriver.common.by import By    # 导入元素定位方法包
from time import sleep    # 导入睡眠时间包
from selenium.webdriver.common.action_chains import ActionChains #导入鼠标包


# 2、选择浏览器驱动
driver = webdriver.Chrome()
driver.implicitly_wait(30)

# 3、使用驱动访问网址
----进入方维后台并且登录------
driver.get("http://192.168.85.128/fanwe/m.php")
driver.find_element(By.XPATH,'/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[2]/td[2]/input').send_keys('admin')
driver.find_element(By.XPATH,'/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[3]/td[2]/input').send_keys('admin')
driver.find_element(By.XPATH,'/html/body/form/table/tbody/tr/td[3]/table/tbody/tr[5]/td[2]/input').send_keys(1234)
driver.find_element(By.XPATH,'//*[@id="login_btn"]').click()
# driver.implicitly_wait(20)

# driver.switch_to.frame('id/nam/class或元素的定位')     #切换到对应的页面
# # driver.switch_to.parent_frame()  #返回到上层页面
# # driver.switch_to.default_content() #返回到最初的页面，退出到最外层页面

--------------frame-------------
切换到借款管理菜单

fr=driver.find_element(By.XPATH,'/html/frameset/frame[1]') #定位内嵌页面的位置
driver.switch_to.frame(fr)                    #通过定位来切换内嵌页面
driver.find_element(By.XPATH,'//*[@id="navs"]/ul/li[2]/a').click()  #定位内嵌页面中的内容和操作
```

九、键盘操作

- from selenium.webdriver.common.action_chains import ActionChains #导入鼠标包
- 对键盘的操作：
  -
  - ctrl + 键盘字母： send_keys( )

```

```

```
from selenium import webdriver      # 导入如浏览器驱动包
from selenium.webdriver.common.by import By   # 导入元素定位方法包
from time import sleep   # 导入睡眠时间包
from selenium.webdriver.common.action_chains import ActionChains #导入鼠标包
from selenium.webdriver.common.keys import Keys


# 2、选择浏览器驱动
driver = webdriver.Chrome()
# 3、使用驱动访问网址
driver.get("https://www.baidu.com")
inpu=driver.find_element(By.XPATH,'//*[@id="kw"]')
inpu.send_keys('刘德华')
inpu.send_keys(Keys.CONTROL,'a') #全选

sleep(0.5)
inpu.send_keys(Keys.CONTROL,'c')#复制

sleep(0.5)
inpu.send_keys(Keys.CONTROL,'v')#粘贴

sleep(0.5)
inpu.send_keys(Keys.ENTER)  #回车键

sleep(0.5)
inpu.send_keys(Keys.DELETE)#删除

sleep(0.5)
inpu.send_keys(Keys.BACKSPACE)  #删除光标左边的字符
```

十、鼠标操作

鼠标操作：单击，双击，右键，悬浮

导入鼠标包：from selenium.webdriver.common.action_chains import ActionChains #导入鼠标包

```
ActionChains(实例化的驱动变量).click(元素的定位).perform()
click  -->单击
perform()   -->执行操作
```

```
from selenium.webdriver.common.action_chains import ActionChains #导入鼠标包
from selenium import webdriver      # 导入如浏览器驱动包
from selenium.webdriver.common.by import By   # 导入元素定位方法包
from time import sleep   # 导入睡眠时间包

# 2、选择浏览器驱动
driver = webdriver.Chrome()
# 3、使用驱动访问网址
driver.get("https://www.baidu.com")
```

```
# #单击操作
d=driver.find_element(By.XPATH,'//*[@id="s-top-left"]/a[3]')  #地图
# sleep(1)
# 单击
ActionChains(driver).click(d).perform()  #单击地图按钮

# 双击
double=driver.find_element(By.XPATH,'//*[@id="kw"]')
double.send_keys('今天是周五了')
sleep(1)
ActionChains(driver).double_click(double).perform()  #双击输入里面的内容

# 右键
d=driver.find_element(By.XPATH,'//*[@id="s-top-left"]/a[3]')  #地图
ActionChains(driver).context_click(d).perform()  #鼠标在地图上右键

#悬浮
d=driver.find_element(By.XPATH,'//*[@id="s-top-left"]/a[3]')  #地图
ActionChains(driver).move_to_element(d)  #鼠标悬浮到元素上

sleep(10)
driver.close()
driver.quit()
```

十一、浏览器操作

浏览器的常规操作：

前进，后退，刷新，最大化，最小化，设置窗口大小，退出浏览器

```
from selenium import webdriver       # 导入如浏览器驱动包
from selenium.webdriver.common.by import By    # 导入元素定位方法包
from time import sleep    # 导入睡眠时间包

sleep(2)

# 2、选择浏览器驱动
driver = webdriver.Chrome()
# 3、使用驱动访问网址
driver.get("https://www.baidu.com")

#刷新
driver.refresh()
sleep(2)

#打开一个新的分页
driver.get("https://zhidao.baidu.com/")
sleep(2)
```

```
#后退
driver.back()
sleep(2)

#前进
driver.forward()
sleep(2)

#浏览器最小化
driver.minimize_window()
sleep(2)

#浏览器最大化
driver.maximize_window()
sleep(2)

#设置浏览器大小
driver.set_window_size(500,800)
sleep(2)

#关闭分页
driver.close()

#关闭浏览器
driver.quit()
```