**八种元素定位**

id、name、link text、partial link text、classname、

tagname、cssselecter、xpath。

基于标签的属性来操作

id：基于元素属性中id的值来定位，类似于人们身份证上的号码，不会重复

name：基于元素属性中name的值来定位，类似于人们身份证上的名称，很可能会重名

linktext：基于超链接进行定位

partial link text:link text的模糊查询版本，类似于数据库中的like "%，当模糊查询

查询到多个类似的元素，选取第一个

classaname:基于元

tagname：标签名来进行定位，重复度最高。只有在需要定位后进行二次筛选的情况下使用

cssselector：运用相对较多的一种行为，最初IE浏览器不支持xpath，完全基于class属性来进行定位

xpath：目前用于最多的一种定位行为，基于页面结构来进行定位

绝对路径：一层一层的向下剥，直到找到文件，不是遇到十万火急的事不要用这种。。例如： /html/body/div/div[2]/div[5]/div[1]/div/form/span[2]/input

相对路径：基于匹配制度去查找元素，依照xpath语法结构来走。。例如：//\*[@id="su"]

确认Xpath是否正确：

1、在开发者工具elements页面使用Ctrl+F查找，进行判断

2、在Console中输入$x()进行校验

如果要基于text来定位元素

在[]中添加text()="文本内容"来进行查找，例如：//a[text()="登录"]

当无法定位元素时，可以通过子级元素返回父级来获取元素

//input[@id="kw"]

//input[contains=(@id="kw")]

contains函数表示进一步查找，匹配项进一步查询

//input[contains(text(),'搜搜搜索')]

模糊项查询，只要文本中包含此文本就可以进一步查询

\*作业：上次的注册流程，将所有的元素定位全部替换为手写的XPATH的形式来实现

**自动化测试框架的封装**

Driver.find\_element(By.XPATH,”值”).send\_keys(“value”) //需要导入selenium中的By包----------------->例如:driver.find\_element(By.ID,”kw”).send\_keys(“汪洋”)

**UI自动化测试selenium常用API详解（上）**

1. 常用的一些操作元素的方法和属性（附HTML文件：D:\Pycharm\Python Project\Study selenium\test.html）

from selenium import webdriver  
from time import sleep  
from selenium.webdriver.common.by import By  
# 创建浏览器对象  
driver = webdriver.Chrome()  
driver.get("http://47.107.116.139/shopnc/admin/index.php?act=login&op=login")  
# driver.find\_element\_by\_id("user\_name").send\_keys("admin")  
driver.find\_element(By.ID,"user\_name").send\_keys("admin")  
# driver.find\_element(By.ID,"user\_name").clear()  
driver.find\_element(By.ID,"password").send\_keys("admin123")  
# driver.find\_element\_by\_xpath("//input[@value='登录']").click()  
# driver.find\_element\_by\_xpath("//\*[@id='form\_login']/span[4]/input[2]").click()  
# driver.find\_element(By.XPATH,"//\*[@id='form\_login']/span[4]/input[2]").click()  
sleep(5)  
driver.find\_element(By.ID,"form\_login").submit()  
sleep(10)  
driver.close()

---------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------

不可用元素的处理方案：

From selenium import webdriver

From time import sleep

from selenium.webdriver.common.by import By  
driver = webdriver.Chrome()

driver.get("D:/Pycharm/Python Project/Study selenium/test.html")  
js = "document.getElementById('username').removeAttribute('disabled')"  
driver.execute\_script(js)  
sleep(2)  
driver.find\_element(By.ID,"username").send\_keys("汪洋")  
sleep(2)  
driver.find\_element(By.ID,"password").send\_keys("WY123")  
sleep(10)  
driver.close()

不可见元素的处理方案：

From selenium import webdriver

From time import sleep

from selenium.webdriver.common.by import By  
driver = webdriver.Chrome()

driver.get("D:/Pycharm/Python Project/Study selenium/test.html")  
js = "document.getElementById('username').style.display=’’"  
driver.execute\_script(js)  
sleep(2)  
driver.find\_element(By.ID,"username").send\_keys("汪洋")  
sleep(2)  
driver.find\_element(By.ID,"password").send\_keys("WY123")  
sleep(10)  
driver.close()

==============================================================================================================================================================

判断：

is\_displayed------>判断是否可见

is\_selected------->判断是否被选择

is\_enabled------->判断是否可获取

属性：

ele = driver.find\_element\_by\_link\_text("新闻")  
print(ele.tag\_name+"==1==")  
print(ele.text+"==2==")  
print(ele.size)  
print(ele.location)

==============================================================================================================================================================

1. 鼠标和键盘的操作

使用ActionCharins（动作链），通过perform()来执行

=================================鼠标操作====================================

注释：ActionChanis(driver).click(element).perform()=======>ActionChanis:动作链;perform:执行

1. 左键单击

ActionChains(driver).click(driver.find\_element(By.ID,"su")).perform()

1. 右键单击

ActionChains(driver).context\_click(driver.find\_element(By.ID,"su")).perform()

1. 双击

ActionChains(driver).double\_click(driver.find\_element(By.ID,"su")).perform()

1. 拖拽

**只能拖到目标元素的中间：**

ele1 = driver.find\_element\_by\_xpath("//\*[@id='nc\_1\_n1z']")  
ele2 = driver.find\_element\_by\_xpath("//\*[@id='nc\_1\_\_scale\_text']/span")  
ActionChains(driver).drag\_and\_drop(ele1,ele2).perform()

**可以拖拽到结尾：**

ActionChains(driver).drag\_and\_drop\_by\_offset(ele1,ele2.size["width"]-ele1.size["width"],ele2.location["y"]).perform()

1. 悬浮

**悬浮在指定元素：**

ActionChains(driver).move\_to\_element(driver.find\_element(By.XPATH,"//\*[@id='s-top-left']/a[1]")).perform()

**悬浮在指定坐标：**

ActionChains(driver).move\_by\_offset("24","19").perform()

1. 长按

ActionChains(driver).click\_and\_hold(driver.find\_element(By.ID,"su")).perform()

=================================键盘操作======================================

ele..send\_keys(Keys.ENTER)

driver.find\_element(By.ID,"kw").send\_keys("小爱同学")  
ActionChains(driver).send\_keys(Keys.ENTER).perform()

1. 三类等待
2. 线程等待（强制等待）

time.sleep(3)，只作用于一个点

1. 智能等待（隐式等待）

driver.implicitly\_wait(5) 作用于整个driver生命周期，只要整个页面（js.html.css）在5秒内加载完成就不在等待，如果3秒加载完成了，那么继续下一步，那么就不在等待。超过5秒就报错。

1. 显式等待

作用于一个元素，只要这个元素在5秒内加载完成，如果在3秒内完成了，那么继续下一步，不在等待。超过5秒就报错。

WebDriverWait(driver,5,poll\_frequency=0.5).until(lambda x:x.find\_element(By.ID,”kwsss”).message=”元素未找到”)

**UI自动化测试selenium常见的API详解（下）**

1. **验证码处理**
2. 、让开发去掉验证码
3. 、在后台设置万能码
4. 、验证码图片识别技术
5. 、通过cookie绕过登录

**浏览器：**

什么是cookie？

他是保存在客户端的一个字典格式的字符串。

Cookie：cookie、session、token的区别！

什么是session？

Session是保存在服务器的数据，并且session可以通过cookie来传递，名字一般是XXsessionID。

**接口、APP：**

什么是token？

Token是一个鉴权码：通过登录接口来传递。