Selenium高级应用

1 Selenium鼠标键盘操作

1.1 鼠标操作

1.1.1 鼠标操作实现方式

Selenium提供鼠标操作的方法及步骤

- 需要导入ActionChains类

```
from selenium.webdriver.common.action_chains import ActionChains
```

- 通过ActionChains实例化鼠标对象

```
action =ActionChains(driver) # driver表示的是浏览器驱动对象
```

- 调用鼠标的事件方法
- 调用鼠标的执行方法

```
 action.perform()
```

1.1.2 鼠标悬停

- 案例　demo页面鼠标在Focused按钮悬停

```
# 导包
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
import time
# 实例化浏览器

driver = webdriver.Chrome()

# 打开demo 网站

driver.get('file:///D:/demo.html')

# 鼠标操作 导包
```

```
# 实例化 Actionchains

action = ActionChains(driver)

el = driver.find_element_by_class_name('over')

action.move_to_element(el)

action.perform()
```

1.1.3 鼠标双击

- 案例 demo页面鼠标在DoubleClick按钮双击

```
# 导包
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
import time
# 实例化浏览器

driver = webdriver.Chrome()

# 打开demo 网站

driver.get('file:///D:/demo.html')

# 鼠标操作 导包

# 实例化 Actionchains

action = ActionChains(driver)

# 鼠标双击

el = driver.find_element_by_class_name('double')

action.double_click(el).perform()

# 组合一起操作

# action.move_to_element(move).pause(5).double_click(double).perform()
```

1.1.4 鼠标拖动

实现方式

鼠标拖拽方法1 按住不抬起 -- 移动到拖拽的位置 -- 释放 -- 执行

鼠标拖拽方法2 通过元素

```
action.drag_and_drop(div1,div2).perform()
```

鼠标拖拽方法3 通过元素坐标

```
action.drag_and_drop_by_offset(div1,xoffset=,yoffset=).perform()
```

- 案例 drop文件 红色的滑块移动到绿色滑块上

- 鼠标拖拽方法1

```
# 导包
from selenium import webdriver
import time
from selenium.webdriver.common.action_chains import ActionChains

# 实例化浏览器
driver = webdriver.Chrome()

# actionchains实例化

action = ActionChains(driver)

# 第一种方法  通过鼠标组合 按住不抬起--移动div2 -- 释放--执行

div1 = driver.find_element_by_id('div1')

div2 = driver.find_element_by_id('div2')

action.click_and_hold(div1).move_to_element(div2).release().perform()
```

- 案例 鼠标拖拽方法2

```
# 第二种方法  通过元素滑动
time.sleep(2)
# action.drag_and_drop(div1,div2).perform()
```

- 案例 鼠标拖拽方法3

```
# 第三种方法 通过坐标
print(div1.location)
print(div2.location)

action.drag_and_drop_by_offset(div1,xoffset=162,yoffset=0).perform()
```

1.2 键盘操作

1.2.1 键盘常见快捷键删除、全选、复制、粘贴

- 模拟键盘上面的快捷键的操作
- 调用键盘操作的快捷键的方法

  导包

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

  ```
  element.send_keys(快捷键的键值)
  ```

  需要导入Keys类 单键值： 直接传入对应的键值

- 键盘常见快捷键

  ```
  send_keys(Keys.BACK_SPACE) 删除键(BackSpace)

  send_keys(Keys.SPACE) 空格键(Space)

  send_keys(Keys.TAB) 制表键(Tab)

  send_keys(Keys.ESCAPE) 回退键(Esc)

  send_keys(Keys.ENTER) 回车键(Enter)

  send_keys(Keys.CONTROL,'a') 全选(Ctrl+A)

  send_keys(Keys.CONTROL,'c') 复制(Ctrl+C)

  send_keys(Keys.CONTROL, 'v') 粘贴
  ```

- 案例　打开网站注册页面 定位手机号码输入框信息18800--删除0--复制值-粘贴到验证码输入框

```
# 导包
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

# 实例化浏览器

driver = webdriver.Chrome()

#  键盘调用

driver.get('http://localhost:8081/')

# 导包

# 点击注册

driver.find_element_by_link_text('注册').click()
```

```
# 手机号码 输入18800

el = driver.find_element_by_name('username')

el.send_keys('18800')

# 删除
el.send_keys(Keys.BACK_SPACE)

# 全选

el.send_keys(Keys.CONTROL,'a')

# 复制
el.send_keys(Keys.CONTROL,'c')

# 粘贴图像验证码
driver.find_element_by_name('verify_code').send_keys(Keys.CONTROL,'v')
```

2 Selenium 元素等待

2.1 元素等待概念

元素等待

概念：在定位页面元素时如果未找到，会在指定时间内一直等待的过程

为什么要设置元素等待?

网络速度慢

电脑配置低

服务器处理请求慢

Selenium中元素等待有几种类型呢?

三种元素等待类型

2.2 三种等待类型

2.2.1 Time （强制等待）

- 案例 demo页面点击wait按钮，等待6s，打印提示信息

```
# 导包
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

import time


# 实例化浏览器

driver = webdriver.Chrome()

driver.implicitly_wait(8)

driver.get('file:///D:/demo.html')

# 定位wait 按钮  打印提示信息

driver.find_element_by_class_name('wait').click()
# 强制等待
# time.sleep(8)

driver.find_element_by_class_name('red')
```

2.2.2 Implicitly_Wait() （隐式等待）

概念

定位元素时，如果能定位到元素则直接返回该元素，不触发等待； 如果不能定位到该元 素，则间隔一段时间后再去定位元素； 如果在达到最大时长时还没有找到指定元素，则抛出元素 不存在的异常 NoSuchElementException

- 案例 demo页面点击wait按钮，等待6s，打印提示信息

```
# 导包
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait
import time

# 隐式等待
# driver.implicitly_wait(8)
# 定位wait 按钮  打印提示信息

print(driver.find_element_by_class_name('red').text)
```

2.2.3 WebDriverWait() （显示等待）

`操作步骤`

导包

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

```
WebDriverWait(driver, timeout, poll_frequency=0.5)
```

`driver`：浏览器驱动对象

`timeout`：超时的时长，单位：秒

`poll_frequency`：检测间隔时间，默认为0.5秒

调用方法 `until(method)`：直到...时

`method`：函数名称，该函数用来实现对元素的定位

一般使用匿名函数来实现

```
lambda x:driver.find_element_by_name('username')
```

```
el = WebDriverWait(driver,timeout=5).until(lambda x:
x.find_element_by_name("username"))
```

还可以通过as关键字将expected_conditions 重命名为EC，并调用presence_of_element_located()方法判断元素是否存在

导包

```
from selenium.webdriver.support import expected_conditions as EC
```

```
el =
WebDriverWait(driver,timeout=5).until(EC.presence_of_element_located((By.NAME,'u
sername')))
```

案例　demo页面点击wait按钮，等待6s，打印提示信息

```
# lambda
el = WebDriverWait(driver,timeout=6).until(lambda x :
driver.find_element(By.CLASS_NAME,'red'))
print(el.text)
```

```
# 期望条件
el =
WebDriverWait(driver,timeout=6).until(EC.presence_of_element_located((By.CLASS_N
AME,'red')))
print(el.text)
```

3 Selenium Alert、多窗口、Frame切换

3.1 Alert处理

3.1.1 操作步骤

```
driver.switch_to.alert 获取弹出框对象
```

3.1.2 处理弹出框

```
alert.text 获取弹出框提示信息
```

```
alert.accept() 确定弹出框
```

```
alert.dismiss() 取消弹出框
```

```
alert.send_keys() 输入信息
```

- 案例 demo页面点击alert按钮，点确定

```
# 导包
import time

from selenium import webdriver


driver = webdriver.Chrome()

driver.get('file:///D:/demo.html')
# 切换警告窗口
driver.find_element_by_class_name('alert').click()
# 确认
driver.switch_to.alert.accept()
```

`alert.dismiss()`

- 案例 demo页面点击confirm按钮，取消弹窗

```
# 导包
import time

from selenium import webdriver


driver = webdriver.Chrome()

driver.get('file:///D:/demo.html')
# 切换警告窗口
# driver.find_element_by_class_name('alert').click()
# 确认
# driver.switch_to.alert.accept()

# driver.find_element_by_class_name('confirm').click()
# 取消
# driver.switch_to.alert.dismiss()
# time.sleep(2)
```

`alert.send_keys`

- 案例 demo页面点击cprompt按钮，输入信息

```
# 输入信息

# driver.find_element_by_class_name('prompt').click()

# time.sleep(3)

# driver.switch_to.alert.send_keys('123')

# driver.switch_to.alert.accept()
```

`alert.text`

- 案例 demo页面点击order_confirm按钮，打印订单信息

```
# 打印文本信息

# driver.find_element_by_class_name('order_confirm').click()
# time.sleep(8)

# print(driver.switch_to.alert.text)
```

3.2 多窗口处理

`多窗口`

点击某些链接会重新打开一个窗口，若想在在新页面操作，需要切换窗口

- `实现方法`
    - 获取当前窗口句柄： driver.current_window_handle
    - 获取所有窗口句柄: driver.window_handles 返回的是一个列表
    - 切换窗口句柄: driver.switch_to.window(window_handle) ， window_handle是窗口句柄
    - 窗口句柄：由操作系统生成的一串唯 一识别码，是一串字符

- 案例 demo页面 -- 点击百度超链接输入软件测试 -- 获取句柄 -- 切换百度-- 输入框输入信息 --切换demo页

```
# 导包
import time

from selenium import webdriver

# 导入键盘
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()

driver.get('file:///D:/demo.html')
# 切换窗口

# 定位百度超链接    定位百度搜索框 输入 软件测试

driver.find_element_by_link_text('baidu').click()

# 获取句柄

print(driver.window_handles)

handles = driver.window_handles

# 切换到最新打开的窗口
driver.switch_to.window(handles[-1])

driver.find_element_by_id('kw').send_keys('软件测试')

time.sleep(5)

# 跳转到demo  输入 我回来了
# 切换到第一个打开的窗口

driver.switch_to.window(handles[0])
driver.find_element_by_id('user').send_keys('我回来了')
time.sleep(2)
```

3.3 Frame处理

`frame`

HTML页面中的一种框架，主要作用是在当前页面中指定区域显示另一页面元素

`frame分类`

iframe与frame定位方法selenium是统一的

```
driver.switch_to.frame(frame_reference) --> 切换到指定
```

3.3.2 frame的方法

- frame框架的name、id或者定位到的frame元素
- index
- 元素标签

```
driver.switch_to.default_content() --> 恢复默认页面方法
```

需求：demo页面 定位

通过id/name方式

案例

```
# 导包
import time

from  selenium import webdriver

driver = webdriver.Chrome()

driver.get('file:///D:/demo.html')

# 切换     第一种方式    id  name   直接取属性值

# driver.switch_to.frame('aa')
# time.sleep(2)
# driver.find_element_by_id('kw').send_keys('好好学习')
```

通过index方式

```
# 导包
import time

from  selenium import webdriver

driver = webdriver.Chrome()

driver.get('file:///D:/demo.html')


# 第二种方式   通过索引
```

```
# driver.switch_to.frame(0)
# driver.find_element_by_id('kw').send_keys('好好学习')
```

通过元素标签

```
# 导包
import time

from  selenium import webdriver

driver = webdriver.Chrome()

driver.get('file:///D:/demo.html')


# 第三种方式
driver.switch_to.frame(driver.find_element_by_tag_name('iframe'))
driver.find_element_by_id('kw').send_keys('好好学习')
```

- 案例　demo页面进入百度输入好好学习， 返回demo主页面定位input输入框输入‘天天向上’

```
# 导包
import time

from  selenium import webdriver

driver = webdriver.Chrome()

driver.get('file:///D:/demo.html')

# 切换     第一种方式    id  name  直接取属性值

# driver.switch_to.frame('aa')
# time.sleep(2)
# driver.find_element_by_id('kw').send_keys('好好学习')


# 调用恢复默认页面方法(switch_to.default_content())

driver.switch_to.default_content()

driver.find_element_by_id('user').send_keys('天天向上')
```

- frame切换原理总结

```
针对同一层级的frame，如果要进行切换的话，需要切回到默认首页

不管当前在哪个层级，如果要回到默认首页，只需要调用一次

回到默认首页的方法(driver.switch_to.default_content())
```

4 Selenium 处理验证码及上传文件

4.1 Selenium处理验证码

4.1.1 什么是验证码

指一种随机生成的信息（数字、字母、汉字、图片、算术题）等为了防止恶意的请求行为，增加应用的安全性

自动化过程中也是需要进行注册或者登陆的操作，所以需要处理验证

4.1.2 验证码处理方式

- 去掉验证码 由开发操作 ，用在测试环境
- 设置万能验证码 由开发 操作， 一般也只使用在测试环境
- 验证码识别技术 由于技术难度高，识别率很难达到100%，一般不建议使用
- 记录COOKIE 通过记录cookie来跳过登陆的操作

4.1.3 Cookie原理

Cookie是由Web服务器生成的，并且保存在用户浏览器上的小文本文件，它可以包含用户相关的信息

Cookie数据格式：键值对组成（python中的字典）

Cookie产生：客户端请求服务器，如果服务器需要记录该用户状态，就向客户端浏览器颁发一个Cookie数据

Cookie使用：当浏览器再次请求该网站时，浏览器把请求的数据和Cookie数据一同提交给服务器，服务器检查Cookie，以此来辨认用户状态

4.1.4 Selenium操作cookie

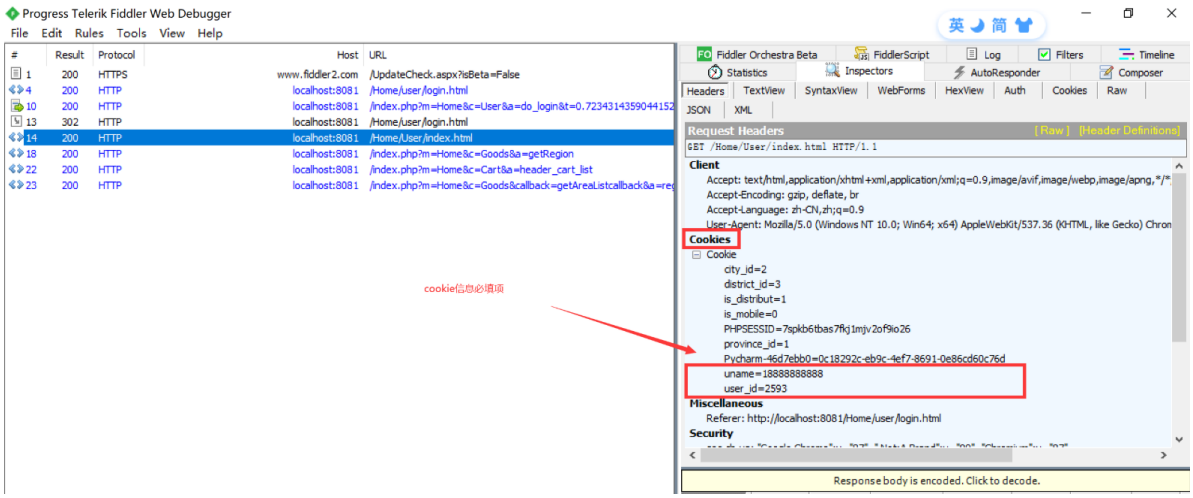
driver.get_cookie(name) 获取指名称的cookie信息 返回的是一个字典

driver.get_cookies() 获取的是所有cookie的信息， 返回的是一个列表

driver.add_cookie(dict_cookie) 往浏览器驱动增加cookie，

dict_cookie是一字典

抓取cookie数据值



需求 ： 抓取商城cookie信息

- 案例

```
# 导包

from selenium import webdriver

driver = webdriver.Chrome()

driver.get('http://localhost:8081/')

# cookie 注入
# 绕过登录

# driver.add_cookie({'name' : 'foo', 'value' : 'bar'})

driver.add_cookie({'name':'uname','value':'18888888888'})

driver.add_cookie({'name':'user_id','value':'2593'})

# 刷新
driver.refresh()
```

4.2 selenium上传文件

- 实现方式

  input标签，组合send_keys（）文件路径传入

- 案例

```
# 导包
import time

from selenium import webdriver

driver = webdriver.Chrome()

driver.get('https://image.baidu.com/')
```

```
# 点击上传按钮

driver.find_element_by_xpath('//a[@id="sttb"]/img[1]').click()

# 上传  send-keys

driver.find_element_by_id('stfile').send_keys(r'C:\Users\ThinkPad\Desktop\Day2\1
.png')

time.sleep(5)
```