简介

- selenium
  - 是一种浏览器自动化的工具，所谓的自动化是指，我们可以通过代码的形式制定一系列的行为动作，然后执行代码，这些动作就会同步触发在浏览器中。

环境安装

- 下载安装selenium：pip install selenium
- 下载浏览器驱动程序：
  - http://chromedriver.storage.googleapis.com/index.html
- 查看驱动和浏览器版本的映射关系：
  - http://blog.csdn.net/huilan_same/article/details/51896672

效果展示

```
from selenium import webdriver
from time import sleep
114.0.5735.90
```

```
# 后面是你的浏览器驱动位置，记得前面加r'','r'是防止字符转义的
driver = webdriver.Chrome(executable_path='./chromedriver')
# 用get打开百度页面
driver.get("http://www.baidu.com")

# 查找页面的“设置”选项，并进行点击
driver.find_element_by_xpath('//*[@id="s-usersetting-top"]').click()
sleep(1)
# # 打开设置后找到“搜索设置”选项，设置为每页显示50条
driver.find_elements_by_link_text('搜索设置')[0].click()
sleep(1)

# 选中每页显示50条
m = driver.find_element_by_xpath('//*[@id="nr_3"]').click()
sleep(1)

# 点击保存设置
driver.find_element_by_xpath('//*[@id="se-setting-7"]/a[2]').click()
sleep(1)
```

```
# 处理弹出的警告页面    确定accept() 和 取消dismiss()
driver.switch_to.alert.accept()
sleep(1)
# 找到百度的输入框，并输入 美女
driver.find_element_by_id('kw').send_keys('美女')
sleep(1)
# 点击搜索按钮
driver.find_element_by_id('su').click()
sleep(1)
driver.find_element_by_xpath('//*[@id="1"]/div/h3/a').click()
sleep(3)

# 关闭浏览器
driver.quit()
```

浏览器创建

- Selenium支持非常多的浏览器，如Chrome、Firefox、Edge等.另外，也支持无界面浏览器。

```
from selenium import webdriver

browser = webdriver.Chrome()
browser = webdriver.Firefox()
browser = webdriver.Edge()
browser = webdriver.PhantomJS()
browser = webdriver.Safari()
```

元素定位

- webdriver 提供了一系列的元素定位方法，常用的有以下几种：

```
find_element_by_id()  重点
find_element_by_name()
find_element_by_class_name()
find_element_by_tag_name()
find_element_by_link_text()
find_element_by_xpath() 重点
find_element_by_css_selector()

或者
from selenium.webdriver.common.by import By
driver.find_element(By.xxx,value)
driver.find_elements(By.xx, value)  # 返回列表
```

节点交互

- Selenium可以驱动浏览器来执行一些操作，也就是说可以让浏览器模拟执行一些动作。比较常见的用法有：输入文字时用`send_keys()`方法，清空文字时用`clear()`方法，点击按钮时用`click()`方法。

执行js

对于某些操作，Selenium API并没有提供。比如，下拉进度条，它可以直接模拟运行JavaScript，此时使用`execute_script()`方法即可实现。

```
from selenium import webdriver
from time import sleep
#1.创建一个浏览器对象,executable_path指定当前浏览器的驱动程序
#注意：我当前是mac系统，驱动程序也是mac版本的，如果是window系统注意更换驱动
bro = webdriver.Chrome(executable_path='./chromedriver')
#2.浏览器的请求发送
bro.get('https://www.jd.com/')
```

```
#3.标签定位:调用find系列的函数进行标签定位
search_box = bro.find_element_by_xpath('//*
[@id="key"]')
#4.节点交互
search_box.send_keys('mac pro m1')#向指定标签中录
入内容
sleep(2)
btn = bro.find_element_by_xpath('//*
[@id="search"]/div/div[2]/button')
btn.click() #点击按钮
sleep(2)
#js注入
bro.execute_script('document.documentElement.sc
rollTo(0,2000)')
sleep(5)
#关闭浏览器
bro.quit()
```

获取页面源码数据(重要)

通过`page_source`属性可以获取网页的源代码，接着就可以使用解析库（如正则表达式、Beautiful Soup、pyquery等）来提取信息了。

前进和后退

```
#模拟浏览器的前进后退
from selenium import webdriver
import time

browser = webdriver.Chrome(r'./chromedriver')
browser.get('https://www.baidu.com')
browser.get('https://www.taobao.com')

browser.back()
time.sleep(2)
browser.forward()
time.sleep(2)

browser.close()
```

获取动态加载数据

- 实现可见即可得
- 爬取豆瓣电影中动态加载的电影详情数据

```
from time import sleep
from selenium import webdriver
from lxml import etree
```

```
bro =
webdriver.Chrome(executable_path='./chromedrive
r')
bro.get('https://movie.douban.com/typerank?
type_name=%E6%82%AC%E7%96%91&type=10&interval_i
d=100:90&action=')
sleep(2)
bro.execute_script('document.documentElement.sc
rollTo(0,2000)')
sleep(2)
#获取页面源码数据
page_text = bro.page_source
#数据解析：解析页面源码数据中动态加载的电影详情数据
tree = etree.HTML(page_text)
ret = tree.xpath('//*
[@id="content"]/div/div[1]/div[6]/div/div/div/d
iv[1]/span[1]/a/text()')
print(ret)
bro.quit()
```

动作链

在上面的实例中，一些交互动作都是针对某个节点执行的。比如，对于输入框，我们就调用它的输入文字和清空文字方法；对于按钮，就调用它的点击方法。其实，还有另外一些操作，它们没有特定的执行对象，比如鼠标拖曳、键盘按键等，这些

动作用另一种方式来执行，那就是动作链。

```
from selenium.webdriver import ActionChains
from selenium import webdriver
from time import sleep
bro =
webdriver.Chrome(executable_path='./chromedrive
r')
bro.get('https://www.runoob.com/try/try.php?
filename=jqueryui-api-droppable')
sleep(1)
#注意：如果定位的标签是存在于iframe表示的子页面中，则常
规的标签定位报错
#处理：使用如下指定操作
bro.switch_to.frame('iframeResult')
div_tag = bro.find_element_by_id('draggable')

#实例化一个动作链对象且将该对象绑定到指定的浏览器中
action = ActionChains(bro)
action.click_and_hold(div_tag) #对指定标签实现点击
且长按操作
for i in range(5):
    action.move_by_offset(10,10).perform()
#perform让动作链立即执行
    sleep(0.5)
sleep(3)
bro.quit()
```

滑动验证

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By  #
按照什么方式查找，By.ID,By.CSS_SELECTOR
from selenium.webdriver.support import
expected_conditions as EC
from selenium.webdriver.support.wait import
WebDriverWait  # 等待页面加载某些元素
import cv2 #pip install opencv-python

from urllib import request
from selenium.webdriver.common.action_chains
import ActionChains


#获取要滑动的距离
def get_distance():
    #滑动验证码的整体背景图片
    background = cv2.imread("background.png",
0)
    #缺口图片
    gap = cv2.imread("gap.png", 0)
```

```
    res = cv2.matchTemplate(background, gap,
cv2.TM_CCOEFF_NORMED)
    value = cv2.minMaxLoc(res)[2][0]
    print(value)
    #单位换算
    return value * 278 / 360


def main():
    chrome =
webdriver.Chrome(executable_path='./chromedrive
r')
    chrome.implicitly_wait(5)


 chrome.get('https://passport.jd.com/new/login.
aspx?')

    login = chrome.find_element(By.CLASS_NAME,
'login-tab-r')
    login.click()

    loginname = chrome.find_element(By.ID,
'loginname')
    loginname.send_keys("123@qq.com")
```

```
    nloginpwd = chrome.find_element(By.ID,
'nloginpwd')
    nloginpwd.send_keys("987654321")

    loginBtn =
chrome.find_element(By.CLASS_NAME, 'login-btn')
    loginBtn.click()
    #带缺口的大图
    img_src = chrome.find_element(By.XPATH,
'//*[@class="JDJRV-
bigimg"]/img').get_attribute("src")
    #缺口图片
    temp_src = chrome.find_element(By.XPATH,
'//*[@class="JDJRV-
smallimg"]/img').get_attribute("src")
    #两张图片保存起来
    request.urlretrieve(img_src,
"background.png")
    request.urlretrieve(temp_src, "gap.png")

    distance = int(get_distance())
    print("distance:", distance)

    print('第一步,点击滑动按钮')
    element =
chrome.find_element(By.CLASS_NAME, 'JDJRV-
slide-btn')
```

```
 ActionChains(chrome).click_and_hold(on_element
=element).perform()  # 点击鼠标左键，按住不放


 ActionChains(chrome).move_by_offset(xoffset=di
stance, yoffset=0).perform()

 ActionChains(chrome).release(on_element=elemen
t).perform()

    time.sleep(2)
if __name__ == '__main__':
    main()
```

带验证码的模拟登录

- 登录bilibili
  - https://passport.bilibili.com/login?from_spm_id=333.851.top_bar.login_window
- 识别验证码模块封装：

```
import base64
```

```
import json
import requests
# 一、图片文字类型(默认 3 数英混合):
# 1 : 纯数字
# 1001: 纯数字2
# 2 : 纯英文
# 1002: 纯英文2
# 3 : 数英混合
# 1003: 数英混合2
#  4 : 闪动GIF
# 7 : 无感学习(独家)
# 11 : 计算题
# 1005:  快速计算题
# 16 : 汉字
# 32 : 通用文字识别(证件、单据)
# 66:  问答题
# 49 :recaptcha图片识别
# 二、图片旋转角度类型:
# 29 :  旋转类型
#
# 三、图片坐标点选类型:
# 19 :  1个坐标
# 20 :  3个坐标
# 21 :  3 ~ 5个坐标
# 22 :  5 ~ 8个坐标
# 27 :  1 ~ 4个坐标
# 48 : 轨迹类型
```

```
#
# 四、缺口识别
# 18 ：缺口识别（需要2张图 一张目标图一张缺口图）
# 33 ：单缺口识别（返回x轴坐标 只需要1张图）
# 五、拼图识别
# 53：拼图识别
#函数实现忽略
def base64_api(uname, pwd, img, typeid):
    with open(img, 'rb') as f:
        base64_data =
base64.b64encode(f.read())
        b64 = base64_data.decode()
    data = {"username": uname, "password":
pwd, "typeid": typeid, "image": b64}
    result =
json.loads(requests.post("http://api.ttshitu.
com/predict", json=data).text)
    if result['success']:
        return result["data"]["result"]
    else:
        return result["message"]
    return ""


def getImgCodeText(imgPath,imgType):#直接返回验
证码内容
    #imgPath：验证码图片地址
```

```
    #imgType: 验证码图片类型
    result = base64_api(uname='图鉴的账号',
pwd='图鉴的密码', img=imgPath, typeid=imgType)
    return result
```

```
from selenium import webdriver
from selenium.webdriver import ActionChains
from time import sleep
import tujian
#1.创建浏览器对象
bro =
webdriver.Chrome(executable_path='../chromedr
iver')
#2.发起请求
login_url =
'https://passport.bilibili.com/login?
from_spm_id=333.851.top_bar.login_window'
bro.get(login_url)
sleep(1)
#3.定位到指定标签填充用户名和密码
user_box = bro.find_element_by_xpath('//*
[@id="app"]/div[2]/div[2]/div[3]/div[2]/div[1
]/div[1]/input')
user_box.send_keys('15027900535')
sleep(1)
```

```
pwd_box = bro.find_element_by_xpath('//*[@id="app"]/div[2]/div[2]/div[3]/div[2]/div[1]/div[3]/input')
pwd_box.send_keys('123456')
sleep(1)
login_btn = bro.find_element_by_xpath('//*[@id="app"]/div[2]/div[2]/div[3]/div[2]/div[2]/div[2]')
login_btn.click()
sleep(10)
#4.定位完整的验证码对话框
#注意：在开发者工具中是可以定位到多个div表示验证码对话框的，因此将这几个div都定位到，以此去尝试
code_tag = bro.find_element_by_xpath('/html/body/div[4]/div[2]/div[6]/div/div')
sleep(1)
#5.识别验证码（使用打码平台进行验证码识别）
code_tag.screenshot('./code.png')#将验证码对话框截图保存
sleep(1)
#使用图鉴接口识别
result = tujian.getImgCodeText('./code.png',27)#获取了识别的结果
# result = '154,251|145,167'
# print(result)
```

```
result_list = result.split('|')
#result_list == ['154,251','145,167']
#6.根据识别出验证码的结果进行处理
for pos in result_list:
    x = int(pos.split(',')[0])
    y = int(pos.split(',')[1])

 ActionChains(bro).move_to_element_with_offse
t(code_tag,x,y).click().perform()
    sleep(0.5)
sleep(2)
#此处使用class属性进行确定标签定位
confirm_btn =
bro.find_element_by_xpath('//a[@class="geetes
t_commit"]')
confirm_btn.click()
sleep(3)
bro.quit()
```

Cookie

使用Selenium，还可以方便地对Cookies进行操作，例如常见的获取Cookies，示例如下：

- get_cookies()返回值是由字典组成的列表，叫做jsonCookies。
- 需要将jsonCookies解析成浏览器携带的cookie形式

```
#获取jsonCookies
from selenium import webdriver
import time

browser = webdriver.Chrome(r'./chromedriver')
browser.get('https://www.zhihu.com/explore')
print(browser.get_cookies())
browser.close()
```

- 解析jsonCookies成浏览器的cookie形式

```
from selenium import webdriver
import time

browser = webdriver.Chrome(r'./chromedriver')
browser.get('https://www.zhihu.com/explore')
#获取cookie
cookies = browser.get_cookies()
#解析cookie
dic = {}
for cookie in cookies:
    key = cookie['name']
    value = cookie['value']
    dic[key] = value
print(dic) #在爬虫中可以使用的cookie
browser.close()
```

- 基于selenium获取cookie后，绕过模拟登录

```
from selenium.webdriver import Chrome
import time
import json

web = Chrome('./chromedriver')
web.get('https://www.17k.com/')
time.sleep(3)
# 登录
web.find_element_by_xpath('//*
[@id="header_login_user"]/a[1]').click()

# 切换iframe
iframe =
web.find_element_by_xpath('/html/body/div[20]
/div/div[1]/iframe')
web.switch_to.frame(iframe)

web.find_element_by_xpath('/html/body/form/dl
/dd[2]/input').send_keys("15027900535")
time.sleep(1)
web.find_element_by_xpath('/html/body/form/dl
/dd[3]/input').send_keys("bobo328410948")
time.sleep(1)
web.find_element_by_xpath('//*
[@id="protocol"]').click()
```

```
time.sleep(1)
web.find_element_by_xpath('/html/body/form/dl
/dd[5]/input').click()
time.sleep(3)
cookies = web.get_cookies()

# 存文件里
with open("cookies.txt", mode="w",
encoding='utf-8') as f:
    f.write(json.dumps(cookies))


# 组装cookie字典, 直接给requests用
dic = {}
for cook in cookies:
    dic[cook['name']] = cook['value']
# 衔接. 把cookie直接怼进去
import requests
#访问的书架（获取书架内容）
url =
"https://user.17k.com/ck/user/myInfo/88786357
?bindInfo=1&appKey=2406394919"
headers = {
    'cookie':dic
}
resp = requests.get(url,cookies=dic)
print(resp.text)
```

```
web.close()
```

无头浏览器

无头浏览器就是没有可视化界面的浏览器

```
from selenium import webdriver
from selenium.webdriver.chrome.options import
Options
import time

# 创建一个参数对象，用来控制chrome以无界面模式打开
chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
# 驱动路径
path = './chromedriver'
# 创建浏览器对象
browser =
webdriver.Chrome(executable_path=path,options=c
hrome_options)

# 上网
```

```
url = 'http://www.baidu.com/'
browser.get(url)
time.sleep(3)

browser.save_screenshot('baidu.png')

browser.quit()
```

规避检测（重要）

- 现在不少大网站有对selenium采取了监测机制。比如正常情况下我们用浏览器访问淘宝等网站的window.navigator.webdriver的值为 undefined或者为false。而使用selenium访问则该值为true。那么如何解决这个问题?
  - 实现js注入，绕过检测

```
from selenium.webdriver import ActionChains
from selenium.webdriver import Chrome
from selenium.webdriver.chrome.options import
Options

chrome_options = Options()
chrome_options.add_argument("--disable-blink-
features=AutomationControlled")
```

```
chrome_options.add_argument('user-
agent=Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.198 Safari/537.36')

driver =
Chrome('./chromedriver',options=chrome_option
s)
#Selenium在打开任何页面之前，先运行这个Js文件。
with open('./stealth.min.js') as f:
    js = f.read()
#进行js注入，绕过检测
#execute_cdp_cmd执行cdp命令（在浏览器开发者工具中执
行相关指令，完成相关操作）
#Page.addScriptToEvaluateOnNewDocument执行脚本
driver.execute_cdp_cmd("Page.addScriptToEvalu
ateOnNewDocument", {
  "source": js
})

driver.get('https://www.taobao.com')
```

- 12306再次测试
  - 没有实现规避检测代码时的登录：

```
from selenium.webdriver import Chrome
```

```
from selenium.webdriver import
ActionChains
import selenium
from time import sleep

web =
Chrome(executable_path='../chromedriver')

web.get("https://kyfw.12306.cn/otn/resour
ces/login.html")
sleep(1)
web.find_element_by_xpath('//*
[@id="toolbar_Div"]/div[2]/div[2]/ul/li[1
]/a').click()
web.find_element_by_xpath('//*[@id="J-
userName"]').send_keys("hehehe@126.com")
sleep(1)
web.find_element_by_xpath('//*[@id="J-
password"]').send_keys("111111")
sleep(1)
web.find_element_by_xpath('//*[@id="J-
login"]').click()
sleep(3)
action = ActionChains(web)
# 找到滑块
btn = web.find_element_by_xpath('//*
[@id="nc_1_n1z"]')
```

```
action.click_and_hold(btn)
for i in range(8):
    try:

 action.move_by_offset(60,0).perform()
    except
selenium.common.exceptions.StaleElementRe
ferenceException as e:
        print(e)
    sleep(0.5)
sleep(3)
web.close()
```

- 实现规避检测后：

```
from selenium.webdriver import Chrome
from selenium.webdriver import
ActionChains
from time import sleep
import selenium
from selenium.webdriver.chrome.options
import Options
chrome_options = Options()
chrome_options.add_argument("--disable-
blink-features=AutomationControlled")
```

```
chrome_options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36')

web = Chrome(executable_path='../chromedriver')
#Selenium在打开任何页面之前，先运行这个Js文件。
with open('./stealth.min.js') as f:
    js = f.read()
#进行js注入，绕过检测
web.execute_cdp_cmd("Page.addScriptToEvaluateOnNewDocument", {
  "source": js
})
web.get("https://kyfw.12306.cn/otn/resources/login.html")
sleep(1)
web.find_element_by_xpath('//*[@id="toolbar_Div"]/div[2]/div[2]/ul/li[1]/a').click()
web.find_element_by_xpath('//*[@id="J-userName"]').send_keys("hehehe@126.com")
sleep(1)
web.find_element_by_xpath('//*[@id="J-password"]').send_keys("111111")
```

```
sleep(1)
web.find_element_by_xpath('//*[@id="J-
login"]').click()
sleep(3)
action = ActionChains(web)
# 找到滑块
btn = web.find_element_by_xpath('//*
[@id="nc_1_n1z"]')
action.click_and_hold(btn)
for i in range(8):
    try:

 action.move_by_offset(60,0).perform()
    except
selenium.common.exceptions.StaleElementRe
ferenceException as e:
        print(e)
    sleep(0.5)
sleep(3)
web.close()
```