

**Day 1:**

|  |
| --- |
|  |
|  | **selenium环境安装：** |
|  | **selenium基本操作** |
|  | **元素定位方法'''** |
|  | **#环境安装：** |
|  | 1.进入cmd， pip install selenium |
|  | 2.将对应版本的chromedriver放置python的安装目录下，python目录可在cmd用命令 where python查看 |
|  |  |
|  |  |
|  | #打开本地测试网址127.0.0.1:5000 |
|  |  |
|  | 1.解压压缩包至本地 |
|  | 2.用pycharm打开该文件。file-open-选择该文件目录，注意是选择整个目录webtest，而不是webautotest.py文件 |
|  | 3.运行webautotest.py，打开127.0.0.1:5000 |
|  |  |
|  |  |
|  | **#第一个自动化脚本** |
|  | from selenium import webdriver |
|  | driver=webdriver.Chrome() #打开谷歌浏览器，如果是火狐打开Firefox。这是实例化一个浏览器对象，后面的操作都是用这个对象里面的方法，例如get |
|  | driver.get("http://127.0.0.1:5000/") #用get方法输入网址 |
|  |  |
|  | 元素定位：  1.F12打开浏览器开发者模式，2.切换至Elements，3.点击左上角定位元素按钮，就可定位页面上的元素。 |
|  | 对于对于 Web 自动化测试来说，就是操作页面上的各种元素，在操作元素之间需要先找到元素，换句话说就是定位元素 |
|  | selenium常见定位元素8种方法： |
|  | Id、Name、Classname、Tagname、linkText、partialLinkText、cssSelector、Xpath |
|  | 前面7种方法一起过一遍，自己敲一遍看效果，重点是xpath定位元素 |
|  |  |
|  |  |
|  |  |
|  |  |
|  | find\_element：找到页面上某一个元素 |
|  | find\_elements：找到页面上所有相同属性元素的list， |
|  |  |
|  |  |
|  |  |
|  | '''定位元素方法 id''' |
|  | driver.find\_element\_by\_id("testid").click() #定位元素属性id为“testid”的元素 |
|  | '''tagname:标签名称''' |
|  | tag\_list=driver.find\_elements\_by\_tag\_name("a") |
|  | print("tag\_list{0},类型是{1}".format(tag\_list,type(tag\_list))) |
|  | tag\_list[0].click() |
|  | '''name''' |
|  | driver.find\_element\_by\_name("username").send\_keys("15902127953") |
|  | '''classname: class的名称''' |
|  | driver.find\_element\_by\_class\_name("testclassname").click() |
|  |  |
|  | '''partialLinktext：超链接文本包含xx值 |
|  | linktext：超链接文本值，例如测试网址登陆页面 有几个超链接，文本信息是"练习一下鼠标移动" 和"练习鼠标拖拽"''' |
|  | driver.find\_element\_by\_link\_text("登录").click() |
|  | driver.find\_elements\_by\_partial\_link\_text("淘宝")[1].click() |
|  | driver.close() |
|  |  |
|  | xpath：为什么要学xpath 1.元素id或者name不唯一 2.id是动态 3.元素没有id或class 4.直接在浏览器可调试元素是否定位正确 |
|  | 看看一些符号的定义 |
|  | // :表示相对路径 （所以xpath的写法是：以//开头+标签名 ） |
|  | /.. :表示找上级 |
|  | / :表示找下级 |
|  | @:标签属性定位 |
|  | () :text的值 |
|  | \* ：标签名任意 |
|  |  |
|  |  |
|  | '''比如测试网址首页的"点击此处"的按钮的id为：testid |
|  | 写相对路径xpath：//button[@id='testid'] |
|  | 或者标签名任意：//\*[@id='testid'] |
|  |  |
|  |  |
|  | 比如测试网址首页的"点击此处"的按钮的text文本为：点击此处 |
|  | 写相对路径xpath：//button[text()='点击此处'] |
|  | 或者标签名任意：//\*[text()='点击此处'] |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **selenium常用操作方法** |
|  |  |
|  | 刷新：refresh, |
|  | 获取浏览器窗口大小:get\_window\_size 设置浏览器窗口大小:set\_window\_siez最大化:maximizewindow |
|  | 获取浏览器窗口个数、切换到指定窗口:switch\_handle,switch\_to.window() |
|  | 前进：forward |
|  | 后退：back |
|  | 关闭：close |
|  | 退出：quit |
|  |  |
|  |  |
|  | import time |
|  | #刷新refresh |
|  | time.sleep(2) |
|  | driver.refresh() |
|  | #前进/后退 |
|  | driver.find\_element\_by\_xpath("/html/body/a").click() |
|  | time.sleep(1) |
|  | driver.back() |
|  | time.sleep(1) |
|  | driver.forward() |
|  |  |
|  | #获取有多少个窗口.切换窗口 |
|  | driver.find\_element\_by\_xpath("/html/body/div[2]/a").click() |
|  | l=driver.window\_handles |
|  | print(l) |
|  | time.sleep(2) |
|  | driver.switch\_to.window(l[1]) |
|  | time.sleep(2) |
|  | driver.switch\_to.window(l[0]) |
|  |  |
|  | #获取窗口大小，设置大小 |
|  | size=driver.get\_window\_size() |
|  | print(size) |
|  | driver.set\_window\_size(300,300) |
|  | size1=driver.get\_window\_size() |
|  | print(size1) |
|  |  |
|  | #关闭和退出 |
|  | driver.find\_element\_by\_xpath("/html/body/div[2]/a").click() |
|  | l=driver.window\_handles |
|  |  |
|  | time.sleep(2) |
|  | driver.switch\_to.window(l[1]) |
|  | driver.switch\_to.window(l[0]) |
|  | driver.quit() |
|  | driver.close() |
|  |  |
|  | #放大缩小 |
|  | driver.minimize\_window() |
|  | time.sleep(2) |
|  | driver.maximize\_window() |
|  |  |
|  |  |
|  |  |
|  | **鼠标操作**  先导入selenium中的class:ActionChains |
|  | from selenium.webdriver.common.action\_chains import ActionChains |
|  | ActionChains原理： |
|  | 1.调用该类里面的方法是，不会立即执行，将需要执行的动作按顺序存放在一个列队里， |
|  | 2.调用 perform()方法时，依次执行列队里的动作 |
|  |  |
|  |  |
|  | 双击：double\_click： |
|  | 以下是双击测试网址中，用户名输入框的实例： |
|  |  |
|  | driver = webdriver.Chrome() |
|  | driver.get('http://127.0.0.1:5000/') |
|  | driver.find\_element\_by\_xpath("//\*[text()='登录']").click() #点击登录 |
|  | driver.find\_element\_by\_xpath("/html/body/form/p[1]/input").send\_keys("15902127953") #输入用户名 |
|  |  |
|  |  |
|  | #1.还是先定位元素， |
|  | #2.先double\_click，再调用perform方法 |
|  | username=driver.find\_element\_by\_xpath("/html/body/form/p[1]/input") #需要双击的元素 |
|  | ActionChains(driver).double\_click(username).perform() |
|  |  |
|  |  |
|  |  |
|  | 鼠标移动至某个元素：move\_to\_element |
|  | 测试网址有个元素，移动上去，颜色变红的实例： |
|  | 1.同样的先定位到，需要将鼠标移动到哪里去的元素 |
|  | 2.再用move\_to\_element+perform方法 |
|  |  |
|  |  |
|  | tag=driver.find\_element\_by\_xpath("/html/body/div/a") #需要移动的目标元素 |
|  | time.sleep(2) |
|  | ActionChains(driver).move\_to\_element(tag).double\_click(username).perform() |
|  |  |
|  |  |
|  |  |
|  | 鼠标左键不松开：click\_and\_hold |
|  | 松开鼠标左键：release |
|  | 测试网址有个元素点击鼠标左键不松开，颜色变浅的实例： |
|  | 1.定位到元素 |
|  | 2.click\_and\_hold+perform方法 |
|  | 3.release，再松开鼠标左键，颜色又变深 |
|  |  |
|  | driver.find\_element\_by\_xpath("/html/body/div/div[1]/a").click() #进入到拖拽练习的页面 |
|  | temp=driver.find\_element\_by\_xpath('//\*[@id="dragger"]') #定位元素 |
|  | time.sleep(2) |
|  | ActionChains(driver).click\_and\_hold(temp).perform() #按住鼠标左键不松开 |
|  | time.sleep(2) |
|  | ActionChains(driver).release(temp).perform() #然后松开鼠标左键 |
|  |  |
|  |  |
|  |  |
|  | 拖拽元素：drag\_and\_drop |
|  | 测试网址上拖拽一个元素到另外一个框里面的实例： |
|  | 1.定位到被拖的元素 |
|  | 2.定位到需要拖到的目的地的元素 |
|  | 3.drag\_and\_drop+perform方法 |
|  |  |
|  | driver.find\_element\_by\_xpath("/html/body/div/div[1]/a").click() #进入到拖拽练习的页面 |
|  | temp=driver.find\_element\_by\_xpath('//\*[@id="dragger"]') #定位被拖的元素 |
|  | item1=driver.find\_element\_by\_xpath("/html/body/div[2]") #定位到拖拽到哪里的元素 |
|  | time.sleep(2) |
|  | ActionChains(driver).drag\_and\_drop(temp,item1).perform() |
|  |  |
|  |  |
|  |  |
|  |  |
|  | 练习：drag\_and\_drop方法的功能是将一个元素拖拽另外一个元素上面 |
|  | 其实用我们上面学到的click\_and\_hold+move\_to\_element+release（依次执行按住鼠标不松开+移动元素+松开） |
|  | 可以实现一模一样的效果 |
|  | 刚好理解一下ActionChains的原理（将需要执行的动作按顺序存放在一个列队里，调用 perform()方法时，依次执行列队里的动作） |
|  |  |
|  |  |
|  | #实现drag\_and\_drop的拖拽效果 |
|  | driver.find\_element\_by\_xpath("/html/body/div/div[1]/a").click() #进入到拖拽练习的页面 |
|  | temp=driver.find\_element\_by\_xpath('//\*[@id="dragger"]') #定位被拖的元素 |
|  | item1=driver.find\_element\_by\_xpath("/html/body/div[2]") #定位到拖拽到哪里的元素 |
|  | ActionChains(driver).click\_and\_hold(temp).move\_to\_element(item1).release(temp).perform()  Day 2:   |  | | --- | |  | |  | **review:** | |  | 1.web自动化selenium环境： | |  | pip install selenium | |  | chromedriver 放置系统path路径下(python路径) | |  | 2.元素定位方法8种 | |  | 浏览器开发者工具F12-elements。 | |  | xpath: | |  | //:表示相对路径 | |  | @：属性 | |  | /..:上级 | |  | /:下级 | |  | text():文本值 | |  |  | |  | 3.selenium常用操作 | |  | find\_element和find\_elemens | |  | send\_keys() click() | |  | refresh | |  | window\_handles | |  | switch\_to.handle(窗口名称) | |  | 鼠标操作： | |  | 导入：Actionschains,perform()方法执行 move\_to\_elements..... | |  |  | |  |  | |  | **今日：** | |  | **1.xpath中更灵活的俩种语法** | |  | contains: 某个元素的某个属性或者文本值包含了什么  例如：//\*[contains(@class,'test')]，表示该元素的class的值包含了 test | |  | starts-with:某个元素的某个属性或者文本值以什么开始  例如：//\*[starts-with(@class,'test')]，表示该元素的class的值以test开始 | |  |  | |  |  | |  | **2. 切换frame**  frameset跟其他普通标签没有区别，不会影响到正常的定位，跟普通元素写xpath是一样的 | |  | 但是frame与iframe里面的元素，需要切换进去才能操作到其中的元素。 | |  | 如果脚本报什么元素找不到：在你确认xpath没写错的情况下，先去观察改元素上下级有没有frame标签。或者直接//frame //iframe搜索 | |  | 切换frame：switch\_to\_frame(id,name,frame的xpath) | |  | 实例：qq邮箱登录 | |  | driver = webdriver.Chrome() | |  | driver.get("https://mail.qq.com/") | |  | driver.switch\_to.frame("frame的id") | |  | driver.find\_element\_by\_xpath("用户名输入框").send\_keys() | |  |  | |  |  | |  |  | |  |  | |  | **3.常用js方法**  JavaScript可以获取浏览器提供的很多对象，并进行操作。window就是一个对象；表示浏览器窗口 | |  | 浏览器开发者工具F12-elements-console： | |  | window就是一个对象；表示浏览器窗口 | |  | window.open(url) | |  | window.innerWidth | |  | window.innerHeight | |  | window.outerWidth | |  | window.outerHeight | |  | 滚动条：window.scrollTo(0,document.body.scrollHeight) | |  | window.By(0,document.body.scrollHeight) | |  | 非页面类型的滚动条：document.getElementsById(id)[0].scrollTop='' | |  | document:表示当前页面对象 | |  | HTML在浏览器中以DOM形式表示为树形结构， | |  | document对象就是整个DOM树的根节点,然后去操作子节点 | |  | 获取当前标题：document.title | |  | 输入文本的值：document.getElementsById(id)[0].value='' | |  | 操作标签： document.getElementsById(id)[0].click() | |  | 更改属性： document.getElementById('vip').style.visibility='visible' | |  |  | |  | 扩展：如果该元素没有id或者name，可以用querySelect方法 | |  | document.querySelector("该元素的css").click() | |  | **4.自动化用例动态读取excel表里面的数据，典型的数据驱动实例** | |  | a.安装python操作excel依赖的模块： pip install xlrd | |  | xlrd（读取excel） xlwt(写入excel) | |  |  | |  | import xlrd,xlwt | |  | excel =xlrd.open\_workbook(r'C:\Users\MIME\Desktop\test.xlsx') 获取本地的excel | |  | sheet = excel.sheet\_by\_index(0) 取该excel的第一个sheet | |  | print(sheet.nrows,sheet.ncols) 表示获取sheet中的行，列数 | |  |  | |  |  | |  | 实例：新建excel，写入用户名和密码的测试数据，写一个测试用例(登录)，依次去读取excel表的数据， | |  | 实现打开一次浏览器，和打开多次浏览器登录 | |  |  | |  |  | |  | 实现打开x(excel表中的数据多少组)次浏览器 | |  | #定义登录方法： | |  | def login(username,password): | |  | driver = webdriver.Chrome() | |  | driver.get("http://127.0.0.1:5000/") | |  | driver.find\_element\_by\_xpath("").click() 点击登录按钮 | |  | driver.find\_element\_by\_xpath("").send\_keys() 输入用户名 | |  | driver.find\_element\_by\_xpath("").send\_keys() 输入密码 | |  | driver.find\_element\_by\_xpath("").click() 点击signin 按钮登录 | |  |  | |  | #循环去取excel数据，循环的是excel的行数(sheet.nrows) | |  | for i in range(sheet.nrows): | |  | username = int(sheet.row\_values(i)[0]) | |  | password = int(sheet.row\_values(i)[1]) | |  | print('第{0}行，用户名{1}，密码{2}'.format(i,username,password)) 获取每一行的数据：用户名和密码 | |  | login(username,password) 调用登录方法 | |  |  | |  | 仅打开一次浏览器 | |  |  | |  | #定义登录方法： | |  | def login(): | |  | driver = webdriver.Chrome() | |  | driver.get("http://127.0.0.1:5000/") | |  | driver.find\_element\_by\_xpath("").click() 点击登录按钮 | |  | #循环去取excel数据循环的是excel的行数(sheet.nrows) | |  | for i in range(sheet.nrows): | |  | username = int(sheet.row\_values(i)[0]) | |  | password = int(sheet.row\_values(i)[1]) | |  | print('第{0}行，用户名{1}，密码{2}'.format(i,username,password)) 获取每一行的数据：用户名和密码 | |  | driver.find\_element\_by\_xpath("").send\_keys 输入用户名 | |  | driver.find\_element\_by\_xpath("").send\_keys 输入密码 | |  | driver.find\_element\_by\_xpath("").click() 点击signin 按钮登录 | |  |  | |  | login() #方法记得需要调用，才会执行 | |  |  | |  | **5. 练习鼠标操作，切换frame，切换window** | |  | from selenium import webdriver | |  | from selenium.webdriver.common.action\_chains import ActionChains | |  |  | |  |  | |  | 俩个用例： | |  | 1.点击院系专业-临床医学系= | |  | driver = webdriver.Chrome() | |  | driver.get("http://www.zzu.edu.cn/") | |  | #切换frame | |  | driver.switch\_to.frame("zzu\_top\_6") | |  | tag=driver.find\_element\_by\_xpath("//\*[text()='院系专业']") | |  | ActionChains(driver).move\_to\_element(tag).perform() | |  | driver.find\_element\_by\_xpath("//\*[text()='临床医学系']").click() | |  |  | |  | 2.再返回首页点击院系专业下的“更多按钮” | |  | window\_list = driver.window\_handles | |  | driver.switch\_to.window(window\_list[0]) | |  | #返回首页后需要继续切换frame | |  | driver.switch\_to.frame("zzu\_top\_6") | |  | tag=driver.find\_element\_by\_xpath("//\*[text()='院系专业']") | |  | ActionChains(driver).move\_to\_element(tag).perform() | |  | driver.find\_element\_by\_xpath("//\*[text()='临床医学系']/../..//\*[contains(text(),"更多")]").click() | |  |  | |  | 上面的切换frame和鼠标移动，因为俩个用例都用到了，可以封装成一个方法，大家自己去写一下 | |  |  | |  |  | |  |  | |  | homework： | |  | 背景：上节课中我们练习了从excel依次读取数据，仅打开一次浏览器，自动化脚本就可依次去输入excel表中的数据 | |  | 但是前提是第一个用户名和密码都是错误的，如果用户名和密码正确的话，就进入了登录界面 | |  | 这时候再去用户名输入框输入数据会报错，用户名输入框找不到 | |  |  | |  | 练习：前提还是只打开一次浏览器 | |  | 条件：不管excel表中的某一行的用户名和密码，是否是正确的数据，都不影响自动化脚本去读excel表下一行中的数据，实现登录， | |  | 例如excel表中的数据可能是这样的： | |  | 15902127953 123456 第一行 正确的用户名和密码 | |  | 159021 123 第二行 错误的用户名和密码 | |  | 15902127953 123456 第三行 正确的用户名和密码 | |  | 159021279 123 第四行 错误的用户名和密码 | |  |  | |  | 有兴趣的同学可以用xlwt模块， | |  | 用python去新建excel，写入数据， | |  | 再用xlrd模块去读取你写入的excel数据， | |  | 再用自动化脚本去登录 | |  |  | |

**Day 3:**

**Day 4:**

1、安装intellibot，此插件方便写代码有提示

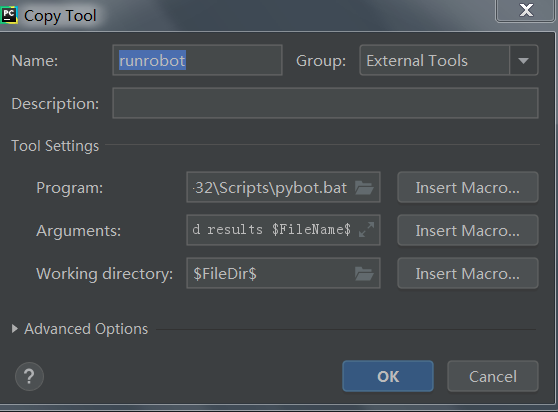
File->setting->plugin->browse repositories->输入intelliBot @SeleniumLibrary

安装完成点击restart

2、终端安装pip install robotframework 和 pip install robotframework-selenium2Library

3、Pycharm 配置robot：

File->setting->tools->external tools->点击+



Name随便写

Program: 填写Python安装路径+\Scripts\pybot.bat

如：D:\Users\Administrator\AppData\Local\Programs\Python\Python37-32\Scripts\pybot.bat

Arguments: -d results $FileName$

Working directory: $FileDir$

知识点：

Library Selenium2Library 导包

Evaluate: RF中不支持加减法，使用evaluate进行计算

${a} set variable *1*${b} set variable *2*${c} evaluate ${a} *+* ${b}  
log ${c}

Click element 点击元素

Select frame 切换frame

Mouse over 鼠标移动

Open browser 打开浏览器

Wait until element is visible 等待页面加载完成

Sleep 延时

Close browser 关闭窗口

切换窗口：

1. select window title=郑州大学官方网站

2. ${count} get window handles  
 ${a} evaluate ${count}[0]  
 select window ${a}

关键字 \*\*\* Keywords \*\*\* 可以写在一个robot文件,需要导入Library *Selenium2Library*

Resource ./文件名 导入其他robot文件

Library ./Python文件 导入Python文件

\*\*\* Settings \*\*\*  
#Resource ./class04.robot  
Library *./class20181202.py*\*\*\* Test Cases \*\*\*  
**case1** say\_hello *a*

**四大模块：**

\*\*\* Setting \*\*\*  
Library *Selenium2Library***Test Setup** #单个用例执行前的操作  
**Test Teardown** #单个用例执行后的操作  
**Suite Setup** #所有用例执行前的操作  
**Suite Teardown** #所有用例执行后的操作

\*\*\* Test Case \*\*\*  
**case1** #用例名称  
 ***[Documentation]*** *测试打开浏览器* ${a} set variable *1* ${b} set variable *2* ${c} evaluate ${a} *+* ${b}  
 log ${c}  
 open\_sit  
**case2  
 *[Documentation]*** *测试其他*

\*\*\* Variables \*\*\*  
${username} *15902127953*${pwd} *123456*

\*\*\* Keywords \*\*\*  
**open\_sit** open browser *http://127.0.0.1:5000/ chrome***login\_sit** click element *xpath=/html/body/a* input text *xpath=/html/body/form/p[1]/input* ${username}  
 input text *xpath=/html/body/form/p[2]/input* ${pwd}  
 click element *xpath=/html/body/form/p[3]/button*

讲课例子：

\*\*\* Setting \*\*\*  
**Documentation**#Library Selenium2Library  
#Test Setup open\_sit #单个用例执行前的操作  
  
\*\*\* Test Case \*\*\*  
**case1** #用例名称  
# ${a} set variable 1  
# ${b} set variable 2  
# ${c} evaluate ${a} + ${b}  
# log ${c}  
 open\_sit  
\*\*\* Variables \*\*\*  
${username} *15902127953*${pwd} *123456*\*\*\* Keywords \*\*\*  
**open\_sit** open browser *http://127.0.0.1:5000/ chrome***login\_sit** click element *xpath=/html/body/a* input text *xpath=/html/body/form/p[1]/input* ${username}  
 input text *xpath=/html/body/form/p[2]/input* ${pwd}  
 click element *xpath=/html/body/form/p[3]/button*

**郑州大学例子：**

\*\*\* Setting \*\*\*  
**Documentation**Library *Selenium2Library***Suite Setup** open\_sit #所有用例执行前的操作  
  
\*\*\* Test Case \*\*\*  
**case1  
 *[Documentation]*** *点击临床医学系* select frame *xpath=//\*[@name='zzu\_top\_6']* mouse over *xpath=//\*[@id="header\_big\_nav"]/li[3]* click element *xpath=//\*[text()='临床医学系']***case2  
 *[Documentation]*** *点击更多* ${count} get window handles  
 ${a} evaluate ${count}*[0]*# select window title=郑州大学官方网站  
 select window ${a}  
 sleep *2* select frame *xpath=//\*[@name='zzu\_top\_6']* mouse over *xpath=//\*[@id="header\_big\_nav"]/li[3]* click element *xpath=//\*[@id="header\_big\_nav"]/li[3]/div[1]/div[1]/div[3]/dl[2]/dt[7]/span[2]*\*\*\* Keywords \*\*\*  
**open\_sit  
 *[Documentation]*** *打开郑州大学官网* open browser *http://www.zzu.edu.cn/ chrome*