# WebUI自动化

## 做什么？

将网页的点点点动作用代码实现

## 用什么工具？ Selenium

Selenium1:seleniumRC（相当于代理服务器）,seleniumCore（控制浏览器行为的JS代码）

Selenium2:整合了seleniumRC，用webdriver来替代了他的功能

Selenium3: 升级了webdriver的实现，调用方式仍然和2相同，因此对于开发者来说是无感的

最新Selenium版本的核心是webdriver

我们的课程以selenium3为基础，所以需要重点了解掌握webdriver

# 认识HTML

WebUI自动化的操作对象是浏览器中的网页

构成网页的基本元素就是HTML，所以需要了解HTML的知识

自学网址：<http://www.w3school.com.cn/html/index.asp>

首先要学习什么是HTML标签，属性

## HTML标签

HTML 标记标签通常被称为 HTML 标签 (HTML tag)。

* HTML 标签是由**尖括号**包围的关键词，比如 <html>
* HTML 标签通常是**成对出现**的，比如 <b> 和 </b>
* 标签对中的第一个标签是**开始标签**，第二个标签是**结束标签**
* 开始和结束标签也被称为**开放标签**和**闭合标签**

## HTML 属性

HTML 标签可以拥有**属性**。属性提供了有关 HTML 元素的**更多的信息**。

属性总是以名称/值对的形式出现，比如：**name="value"**。

属性总是在 HTML 元素的**开始标签**中规定。

详细的描述信息都可以在网上找到

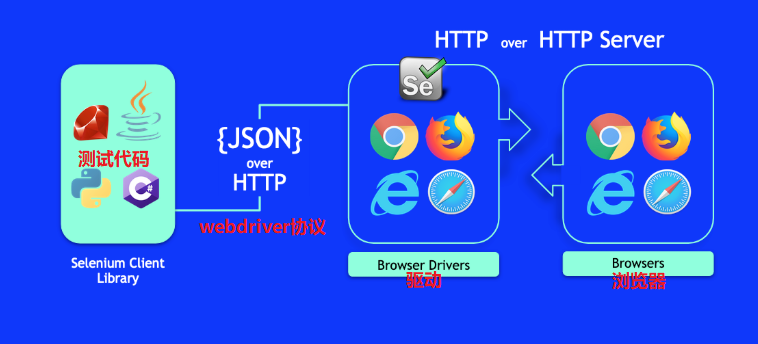
# Webdriver的工作原理

Webdriver直接驱动浏览器来模拟一些人的操作，如点击按钮，输入字符串等

而我们的测试代码是通过发送命令给webdriver完成这些操作，

虽然不同的浏览器有不同的驱动，但是我们调用的代码API都是一样的

原理参考图：



# Python+selenium自动化环境搭建

## 安装selenium API库：

pip install selenium

## 下载对应浏览器驱动

以谷歌浏览器为例

chrome浏览器的web driver （chromedriver.exe）， 可以在以下网址访问：

<https://sites.google.com/a/chromium.org/chromedriver/>

不过，最终的下载链接如下， 可以直接访问

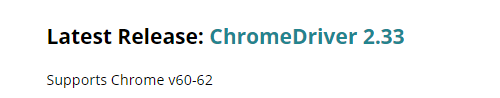
<https://chromedriver.storage.googleapis.com/index.html>

**注意点: 不同的浏览器对于不同的webdriver,同一浏览器的不同版本同样与webdriver版本一一对应**

（很多同学自动化出现问题，就是版本对应关系搞错了。）一般越新的驱动，对应新的浏览器版本。

对应关系在 驱动说明网页有， 比如最新的2.33版本， 打开下面的链接

<https://sites.google.com/a/chromium.org/chromedriver/downloads>



就说明了要使用 的chrome浏览器版本是 60 到 62.

我们通常使用最新的版本的驱动， 对象的都是比较新的浏览器版本

新的chrome浏览器，可以到百度搜索下载谷歌浏览器

上面的google域名的网址需要 翻墙才能访问！！！

**翻墙有困难的同学看下面：**

Chromedriver国内下载地址：

<http://npm.taobao.org/mirrors/chromedriver/>

**注意版本对应关系（如有更新可以自行百度-chromedriver对应关系）：**

<http://blog.csdn.net/huilan_same/article/details/51896672>

火狐的地址

Firefoxdriver

https://github.com/mozilla/geckodriver/releases

版本对照表：

https://firefox-source-docs.mozilla.org/testing/geckodriver/Support.html

避坑指南

报证操作系统的干净

卸载国产杀毒软件和安全管家

win10自带的defender足以抵抗病毒（开启自动更新）

## 代码实例

下载好了驱动，安装好selenium库之后即可写代码了。

以课堂代码为例：

导入相关库

**import** time  
导入第三方库的时候需要注意，项目的文件夹名称不能为selenium，否则无法导入webdriver  
**from** selenium **import** webdriver  
  
  
# 指定是chrome 的驱动  
# 执行到这里的时候Selenium会去到指定的路径将chrome driver 程序运行起来  
指定驱动的时候要注意，驱动路径要写全,

或者将包含驱动的目录添加到环境变量，不要添加很多个webdriver的环境变量，防止冲突  
driver = webdriver.Chrome(**r"d:\tools\webdrivers\chromedriver.exe"**)  
# driver = webdriver.Firefox()

设置隐士等待的目的是为了找元素的时候动态等待页面加载，参数是最大等待时间，单位为秒  
#隐式等待  
driver.implicitly\_wait(**10**)

调用get方法会打开浏览器并访问到指定的网址，注意:http://不能漏写  
# get 方法 打开指定网址  
driver.get(**'http://www.baidu.com'**)

根据ID查找元素的方法，注意：不能写成find\_elements\_by\_id  
# 查找到那个搜索输入栏网页元素，返回一个表示该元素的WebElement对象。  
element\_keyword = driver.find\_element\_by\_id(**"kw"**)  
  
# 输入字符  
element\_keyword.send\_keys(**'松勤'**)  
  
# 找到搜索按钮  
element\_search\_button = driver.find\_element\_by\_id(**"su"**)  
# 点击该元素  
  
element\_search\_button.click()  
  
# time.sleep(5)  
  
#查看搜索结果  
res\_ele=driver.find\_element\_by\_id(**'1'**)  
print(res\_ele.text)  
# print(type(res\_ele.text))  
  
**if** (res\_ele.text.startswith(**'松勤网 - 松勤软件测试-软件测试在线教育领跑者'**)):  
 print(**'pass'**)  
**else**:  
 print(**'fail'**)  
  
  
# 最后，driver.quit让浏览器和驱动进程一起退出。不然会有好几个实例一起运行  
  
driver.quit()

# UI自动化的套路

选择元素：id,name,class,tag,link,css,xpath等

操作元素:点击，输入，拖拽，获取元素信息

根据获取的信息进行下一步的操作

# 选择web元素的方法

## 通过id选择元素

写法1：

element = driver.find\_element\_by\_id("kw")

写法2：

from selenium.webdriver.common.by import By

element = driver.find\_element(by=By.ID, value="kw")

Tips: by=By.ID表示以id的方式定位web元素，下面的例子有类似的写法以此类推

## 通过name选择元素

写法1：

cheese = driver.find\_element\_by\_name("cheese")

写法2：

from selenium.webdriver.common.by import By

cheese = driver.find\_element(By.NAME, "cheese")

## 通过class选择元素

写法1：

cheeses = driver.find\_elements\_by\_class\_name("cheese")

Tips:注意，这里返回的是一个包含web元素的列表，下同

写法2：

from selenium.webdriver.common.by import By

cheeses = driver.find\_elements(By.CLASS\_NAME, "cheese")

## 通过tag选择元素

写法1：

如果iframe这个tag在本html中是唯一的，可以根据iframe这个tag名来找到该元素

frame = driver.find\_element\_by\_tag\_name("iframe")

写法2：

from selenium.webdriver.common.by import By

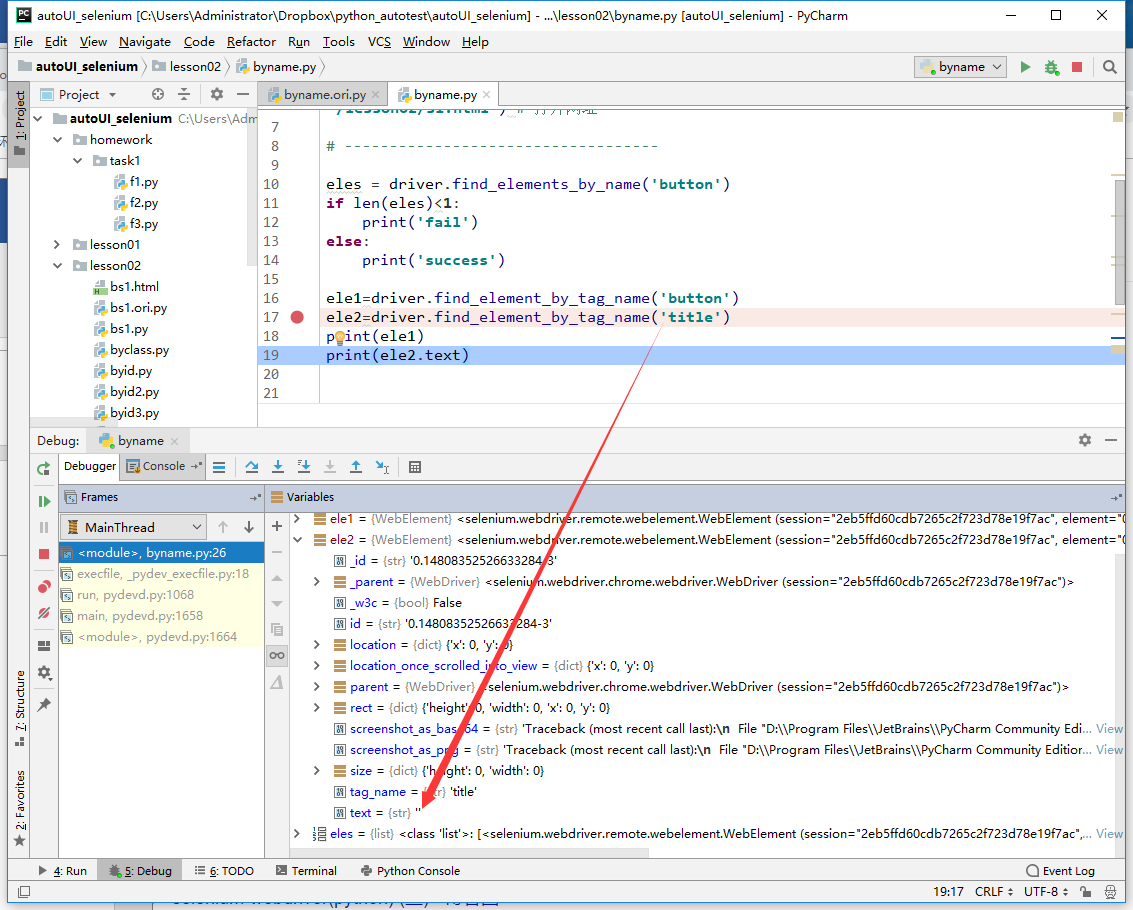
frame = driver.find\_element(By.TAG\_NAME, "iframe")

注意，通过title标签名获取title元素，打印的文本值为空：

ele2=driver.find\_element\_by\_tag\_name(**'title'**)

print(ele2.text)

我们先debug看获取到的元素属性



为什么会造成这种结果？

首先和我们获取元素的方式没有任何关系

text显示的应该是网页中可见的内容。  
title不在网页中，所以显示不出来。

Title是一个特殊的元素，标签内的文本表示页面的标题，

可以通过driver.title属性获取页面标题

## 通过超链接文本选择元素

写法1：

ele = driver.find\_element\_by\_link\_text("转到百度")

写法2：

from selenium.webdriver.common.by import By

ele = driver.find\_element(By.LINK\_TEXT, "转到百度")

# 获取元素信息

拿到webelement对象，我们可以对它做一些操作，比如获取元素内部信息

这个时候操作主体由webdriver变成了webelement，调用的是webelement对象的方法

## 获取元素文本

使用WebElement的 text属性

ele = driver.find\_element\_by\_id("food")

print ele.text

## 获取元素属性值

get\_attribute方法

下面的代码就是获取。。。。。href属性的值

ele **=** driver.find\_element\_by\_id("baidulink")  
print ele.get\_attribute('href')

获取到以后，就可以判断链接是否与预期相同

或者 一个元素的 style属性，比如需要判断颜色是否是红色。

ele **=** driver.find\_element\_by\_id("food")  
print ele.get\_attribute('style')

## 获取元素对应的HTML片段

那么怎么获取整个HTML呢？还是通过get\_attribute方法，

第一种：整个web元素的HTML片段

只需要参数指定为outerHTML 就可以了

ele.get\_attribute('outerHTML')

第二种：web元素内部子元素HTML片段

只需要参数指定为'innerHTML'就可以了

ele.get\_attribute('innerHTML')

课堂演示代码

Firstauto:

**import** time  
  
**from** selenium **import** webdriver  
*#文件名，模块名不要和现有的库名称相同***from** selenium.webdriver.common.by **import** By  
  
driver=webdriver.Chrome(**r'D:\tools\webdrivers\chromedriver.exe'**)*#谷歌浏览器  
#webdriver.Firefox()#火狐浏览器*driver.get(**'https://www.baidu.com/'**)  
  
ele=driver.find\_element\_by\_id(**'kw'**)*#获取元素对象*ele.send\_keys(**'松勤'**)*#输入框输入松勤*driver.find\_element\_by\_id(**'su'**).click()*#点击百度一下  
  
#获取搜素结果并判断*time.sleep(3)  
  
*#res=driver.find\_element\_by\_id('1')*res=driver.find\_element(By.ID,**'1'**)  
  
  
  
**if '松勤网 - 松勤软件测试' in** res.text:  
 print(**'pass'**)  
**else**:  
 print(**'fail'**)  
 print(res.text)  
  
driver.quit()

findelement.py:

**import** time  
  
**from** selenium **import** webdriver  
*#文件名，模块名不要和现有的库名称相同***from** selenium.webdriver.common.by **import** By  
  
driver=webdriver.Chrome(**r'D:\tools\webdrivers\chromedriver.exe'**)*#*driver.get(**'file:///C:/Users/Administrator/Dropbox/python\_autotest/new\_selenium/lesson01/s1.html'**)  
  
*#根据Name查找元素  
  
# btn=driver.find\_elements\_by\_name('button')  
  
  
# print(btn)  
#  
# if btn:  
# print('找到元素了')  
# else:  
# print('没有这个元素')  
  
# btns=driver.find\_elements\_by\_name('button')  
  
# print(btn[0].text)  
# print(btn[1].text)  
# for btn in btns:  
# print(btn.text)  
  
#通过class找元素  
# ele=driver.find\_element\_by\_class\_name('cheese')  
# print(ele.text)  
  
# eles=driver.find\_elements\_by\_class\_name('cheese1')  
# print(eles)  
# for ele in eles:  
# print(ele.text)  
  
#根据tag找元素  
# ele=driver.find\_element\_by\_tag\_name('button')  
# print(ele.text)  
# eles=driver.find\_elements\_by\_tag\_name('button')  
# print(eles)  
# for ele in eles:  
# print(ele.text)  
  
#根据超链接定位元素  
# driver.find\_element\_by\_link\_text('转到百度').click()  
  
# ele=driver.find\_element\_by\_partial\_link\_text('百度2')#根据部分超链接文本定位元素  
#  
# print(ele.get\_attribute('href'))  
  
#获取元素html源码*ele=driver.find\_element\_by\_id(**'many'**)  
print(ele.get\_attribute(**'innerHTML'**))  
print(ele.get\_attribute(**'outerHTML'**))  
  
  
  
*#通过上层元素找子元素*p=ele.find\_element\_by\_name(**'p1'**)  
print(p.get\_attribute(**'class'**))  
  
  
input(**'输入任意键退出。。。。。'**)  
  
  
  
driver.quit()