Selenium简单配置及应用

## 一 Selenium简介：

Selenium是一个自动化测试工具，利用它可以驱动浏览器执行特定的动作，如点击、下拉等操作，同时还可以获取浏览器当前呈现的页面的源代码，做到可见即可爬。

据Selenium官网主页所说，与其他测试工具相比，使用Selenium的最大好处是：

Selenium测试直接在浏览器中运行，就像真实用户所做的一样。Selenium测试可以在Windows、Linux和Macintosh上的Internet、Explorer、Chrome和Firefox中运行。其他测试工具都不能覆盖如此多的平台。使用Selenium和在浏览器中运行测试还有很多其他好处。

## 二 Selenium的安装配置

### 2.1 pip安装

这里推荐直接使用pip安装，执行如下命令即可：

pip3 install selenium

### 2.2 验证安装

进入Python命令行交互模式，导人Selenium包，如果没有报错，则证明安装成功：

$ python3

>> import selenium

但这样做还不够，因为我们还需要用浏览器（如Chrome、Firefox等）来配合Selenium工作。

后面我们会介绍Chrome浏览器的配置方式。有了浏览器，我们才可以配合Selenium进行页面的抓取。

### 2.3 安装ChromeDriver驱动

前面我们成功安装好了Selenium

开始进行Chrome浏览器及ChromeDriver驱动的配置。

首先，下载Chrome浏览器，方法有很多，在此不再赘述。

随后安装ChromeDriver。因为只有安装ChromeDriver，才能驱动Chrome浏览器完成相应的操作

下面我们来介绍下怎样安装ChromeDriver。

（1）相关链接

官方网站：https://sites.google.com/a/chromium.org/chromedriver

下载地址：https://chromedriver.storage.googleapis.com/index.html

（2）准备工作

在这之前请确保已经正确安装好了Chrome浏览器并可以正常运行，安装过程不再复述。

（3）查看版本

点击Chrome菜单“帮助”-“关于Google Chrome”，即可查看Chrome的版本号，如下图所示。

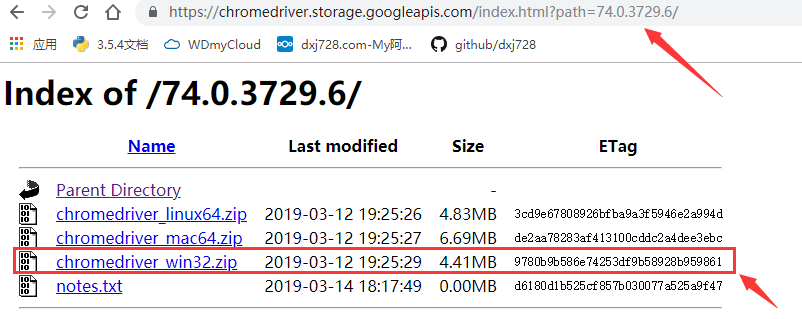


这里我的Chrome版本是74.0。

请记住Chrome版本号，因为后续选择ChromeDriver版本时需要用到。

（4）下载ChromeDriver

打开ChromeDriver的下载网站，找到对应的安装版本，如下图所示：



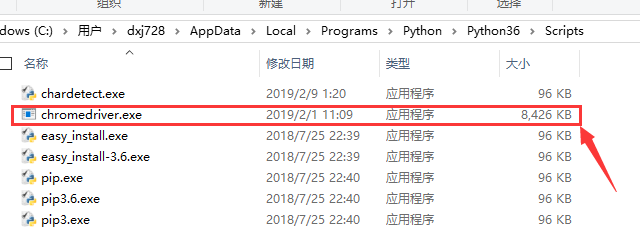
如果你的Chrome版本号是74.0，那么可以选择此版本下载。

如果你的Chrome版本号不在此范围，可以继续查看之前的ChromeDriver版本。每个版本都有相应的支持Chrome版本的介绍，请找好自己的Chrome浏览器版本对应的ChromeDriver版本再下载，否则可能无法正常工作。

（5）环境变量配置

下载完成后，将ChromeDriver的可执行文件配置到环境变量下。

在Windows下，建议直接将chromedriver.exe 文件拖到Python的Scripts目录下，如图所示。

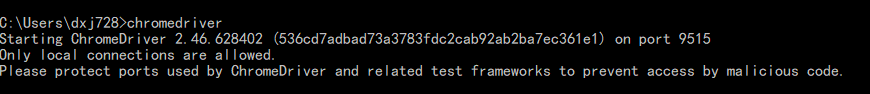


在Linux和Mac下，需要将可执行文件配置到环境变量或将文件移动到属于环境变量的目录里，在此不做介绍。

（6）验证安装

配置完成后，就可以在命令行下直接执行chromedriver命令了：

chromedriver

如果输入控制台有类似图1-16所示的输出，则证明ChromeDriver的环境变量配置好了。

随后再在程序中测试。执行如下Python代码：

from selenium import webdriver

browser = webdriver.Chrome()

运行之后，如果弹出一个谷歌的Chrome浏览器，则证明所有的配置都没有问题。如果没有弹出，请检查之前的每一步配置。如果弹出后闪退，则可能是ChromeDriver版本和Chrome版本不兼容，请更换ChromeDriver版本。如果没有问题，接下来就可以利用Chrome来做网页抓取了。

## 三 基本使用

准备工作做好之后，首先来大体看一下Selenium有一些怎样的功能。示例如下：

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.support.wait import WebDriverWait

browser = webdriver.Chrome()

try:

browser.get('http://www.baidu.com')

input = browser.find\_element\_by\_id('kw')

input.send\_keys('python')

input.send\_keys(Keys.ENTER)

wait = WebDriverWait(browser, 10)

wait.until(EC.presence\_of\_all\_elements\_located((By.ID, 'content\_left')))

print(browser.current\_url)

print(browser.get\_cookies())

print(browser.page\_source)

finally:

browser.close()

运行代码后会发现，自动弹出一个Chrome浏览器，浏览器首先会跳转到百度，然后在搜索框中输入python，接着跳转到搜索结果页，如图所示



搜索结果加载出来后，控制台分别会输出当前的URL、当前的Cookies和网页源代码：

源代码过长，在此省略。可以看到，我们得到的当前URL、Cookies和源代码都是浏览器中的真实内容。

所以说，如果用Selenium来驱动浏览器加载网页的话，就可以直接拿到JavaScript渲染的结果了，不用担心使用的是什么加密系统。

## 四 详细解释

### 4.1初始化

Selenium支持非常多的浏览器，如Chrome、Firefox、Edge等，还有Android等手机端的浏览器。

谷歌chrome浏览器我们可以用如下方式初始化：

from selenium import webdriver

browser = webdriver.Chrome()

这样就完成了浏览器对象的初始化并将其赋值为browser对象。接下来，我们要做的就是调用browser对象，让其执行各个动作以模拟浏览器操作。

### 4.2访问页面

我们可以用get()方法来请求网页，参数传入链接URL即可。比如，这里用get()方法访问淘宝，然后打印出源代码，代码如下：

from selenium import webdnver

browser = webdriver.Chrome()

browser.get（"www.baidu.com"）

print(browser.page\_source)

browser.close()

运行后发现，弹出了Chrome浏览器并且自动访问了百度，然后控制台输出了百度页面的源代码，

随后浏览器关闭。

通过这几行简单的代码，我们可以实现浏览器的驱动并获取网页源码，非常便捷。

3.3单节点查找

比如，想要从淘宝页面中提取搜索框这个节点，首先要观察它的源代码，如图所示：



可以发现，它的id是kw，name则是wd，还有许多其他属性，此时我们就可以用多种方式获取它了。比如，find.element\_by\_name()是根据name值获取，find.element\_by\_id()是根据id获取。

我们用代码实现一下：

from selenium import webdriver

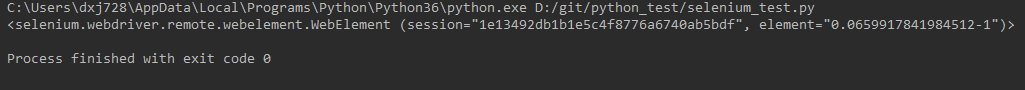
browser = webdriver.Chrome()

browser.get('https://www.baidu.com')

input\_first = browser. find\_element\_by\_id('kw')

print(input\_first)

browser.close()



可以看到，这个节点都是WebElement类型，是正确的。

### 4.3延时等待

在Selenium中，get()方法会在网页框架加载结束后结束执行，此时如果获取page\_source，可能并不是浏览器完全加载完成的页面，如果某些页面有额外的Ajax请求，我们在网页惊代码中也不一定能成功获取到。所以，这里需要延时等待一定时间，确保节点已经加载出来。

这里等待的方式有两种：一种是隐式等待，一种是显式等待。

(1)隐式等待

使用隐式等待执行测试的时候，如果Selenium没有在网页中找到节点，将继续等待，超出设定时间后，则抛出找不到节点的异常。换句话说，当查找节点而节点并没有立即出现的时候，隐式等

待将等待一段时间再查找网页，默认的时间是0。示例如下：

from selenium import webdriver

browser = webdriver.Chrome()

browser.implicitly\_wait(10)

browser.get('https://www.baidu.com/')

input = browser.find\_element\_by\_class\_name('s\_ipt')

print(input)

print("over")

browser.close()

(2)显式等待

隐式等待的效果其实并没有那么好，因为我们只规定了一个固定时间，而页面的加载时间会受到网络条件的影响。

这里还有一种更合适的显式等待方法，它指定要查找的节点，然后指定一个最长等待时间。如果在规定时间内加载出来了这个节点，就返回查找的节点；如果到了规定时间依然没有加载出该节点，则抛阳超时异常。示例如下：

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium .webdriver.support import expected\_conditions as EC

browser = webdriver.Chrome()

browser.get('https://www.baidu.com/')

wait = WebDriverWait(browser,10)

input = wait.until(EC.presence\_of\_element\_located((By.ID,'kw')))

print(input)

print("over")

browser.close()这里首先引人WebDriverWait这个对象，指定最长等待时间，然后调用它的until()方法，传入要等待条件expected\_conditions。比如，这里传入了presence\_of\_element located 这个条件，代表节点出现的意思，其参数是节点的定位元组，也就是ID为q的节点搜索框。

这样可以做到的效果就是，在10秒内如果ID为q的节点（即搜索框）成功加载出来，就返回该节点；如果超过10秒还没有加载出来，就抛出异常。

### 4.4网页前进后退及选项卡切换

平常使用浏览器时都有前进和后退功能，Selenium也可以完成这些操作，其使用back()方式后退，使用forward()方法前进。

在访问网页的时候，会开启一个个选项卡。在Selenium中，我们也可以对选项卡进行操作。

以上两种综合示例如下：

browser = webdriver.Chrome()

try:

browser.get('https://www.baidu.com')

input = browser.find\_element\_by\_id('kw')

input.send\_keys('python')

input.send\_keys(Keys.ENTER)

wait = WebDriverWait(browser, 5)

wait.until(EC.presence\_of\_all\_elements\_located((By.ID, 'content\_left')))

time.sleep(3)

input.clear()

time.sleep(1)

input.send\_keys('hello world!')

input.send\_keys(Keys.ENTER)

wait = WebDriverWait(browser, 5)

wait.until(EC.presence\_of\_all\_elements\_located((By.ID, 'content\_left')))

browser.back()

browser.execute\_script('window.open()')

print(browser.window\_handles)

browser.switch\_to.window(browser.window\_handles[1])

browser.get('https://www.github.com')

wait = WebDriverWait(browser, 5)

browser.switch\_to.window(browser.window\_handles[0])

time.sleep(1)

browser.back()

time.sleep(3)

browser.forward()

print('over')

finally:

browser.close()

## 四.实际应用

定时tableau邮件推送：