**Chrome当前网页截全图操作**

在日常的项目需求中，很多时候客户都要求对网页进行截图。注意这个截图并不是当前的屏幕截图。注么使用Python来完成需求的时候，我们该怎么实现这些这个功能呢？都有哪些比较好的方法呢？我们来一起看看。

对于这个需求，我们将采用两种方式实现，我们先来看第一个方法：

1. **基于driver.execute\_script执行JS的方法**

这种方法使用的是selenium中execute\_script()方法来执行JS脚本的方式，我们先来看代码:

from selenium import webdriver  
from selenium.webdriver.chrome.options import Options  
import time  
def get\_screen\_image(url, pic\_name):  
 t1 = time.time()  
 *# 一定要使用headless就是无界面模式这个模式，不然截不了全页面* chrome\_options = Options()  
 chrome\_options.add\_argument(**'headless'**)  
 driver = webdriver.Chrome(chrome\_options=chrome\_options)  
 *# 打开网页* driver.get(url)  
 time.sleep(1)  
 *# 通过执行js方法来获取网页的宽度和高度* width = driver.execute\_script(**"return document.documentElement.scrollWidth"**) + 10  
 height = driver.execute\_script(**"return document.documentElement.scrollHeight"**) + 10  
 print(width,height)  
 *# 设置无头浏览器宽度和高度* driver.set\_window\_size(width, height)  
 time.sleep(1)  
 *#截图*

driver.save\_screenshot(pic\_name)  
 driver.quit()  
 t2 = time.time()  
 print(t2-t1)  
*#你输入的参数*url=**'https://blog.csdn.net/th1522856954/article/details/109589959?spm=1001.2014.3001.5501'**pic\_name = **r'C:\Users\LEGION\Desktop\code\\image.png'**get\_screen\_image(url, pic\_name)

我们来解释其中的一些代码：

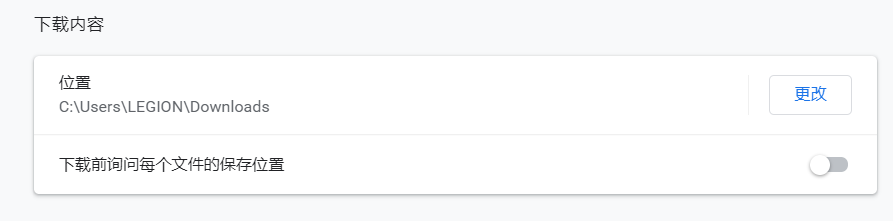
1. 第7、8两行代码用来设置Chrome浏览器的。参数值“**headless**”指使用无头模式，其他参数如“**start**-**maximized**”，“**disabled-infobars**”等等。可以从以下链接查看其他参数：<https://peter.sh/experiments/chromium-command-line-switches/>
2. 第14、15两行代码是使用Selenium的execute\_script()方法来执行JS的具体脚本来获取网页的宽和高度，具体的值是像素点个数，如1080。
3. 之后的代码是将获取到的宽度和高度设置给浏览器，通过保存截图的方法save\_screenshot()来保存结果。最后使用driver.quit()来关闭浏览器。
4. **基于Chrome的执行指令方法**

第二种方法是基于Chrome的执行指令方法，没有设置浏览器的无头模式。我们来看一下代码：

import os  
import time  
import pyautogui  
import pyperclip  
from selenium import webdriver  
from selenium.webdriver.chrome.options import Options  
def get\_screen\_image(url):  
 t1 = time.time()  
 driver = webdriver.Chrome()  
 driver.maximize\_window()  
 time.sleep(0.5)  
 *#控制浏览器写入并转到链接* driver.get(url)  
 time.sleep(0.5)  
 *# 截图并关掉浏览器* pyautogui.press(**'f12'**)  
 time.sleep(0.5)  
 pyautogui.hotkey(**'ctrl'**, **'shift'**, **'P'**)  
 time.sleep(0.5)  
 pyperclip.copy(**'Capture full size screenshot'**)  
 time.sleep(0.5)  
 pyautogui.hotkey(**'ctrl'**,**'v'**)  
 time.sleep(0.5)  
 pyautogui.press(**'Enter'**)  
 *# 等待文件下载完成* path = **r'C:\Users\LEGION\Downloads'** FLAG = 1  
 while FLAG:  
 time.sleep(1)  
 print(**'开始查找文件存在不存在'**)  
 for filename in os.listdir(path):  
 file = os.path.abspath(path) + **'**\\**'** + filename  
 if **'.png'** in file or **'.PNG'** in file:  
 print(**'文件存在'**)  
 FLAG = 0  
 time.sleep(1)  
 driver.quit()  
 t2 = time.time()  
 print(t2-t1)  
*#输入参数,调用*url=**'https://blog.csdn.net/th1522856954/article/details/109589959?spm=1001.2014.3001.5501'**get\_screen\_image(url)

我们来解释其中的一些代码：

1. 第10行代码使用driver.maximize\_window()来进行浏览器的最大化操作。
2. 第16行代码模拟键盘输入F12,启动开发者模式。
3. 第18行代码使用pyautogui.hotkey()将Ctrl、shift和P三键同时按下。
4. 第20行代码使用pyperclip.copy(**'Capture full size screenshot'**)来设置剪贴板的的值为：**Capture full size screenshot。**
5. 之后按下Ctrl+v将剪贴板的数据输入，并按下Enter。
6. 之后的代码，我们为了确保截图文保存是否保存下来，我们去浏览器的默认下载路径(Chrome浏览器：设置->高级->下载内容，进行查看)判断有没有将截图文件保存下来，这里我没有设置等待的最长时间，大家可以增加一个超时时间。



大家可以运行我提供的代码来实际看一下整个过程是怎样的。

这里大家看一个全程的操作录制图：

**GIF图片位置**

1. **总结**

以上就是本次推文关于在Chrome中进行网页截图的操作，在一些实际的自动化需求中，大家可以依据此方法来获取截图，对于网页或者界面的操作，大家可以只用Selenium或者Uiautiomation来实现。