动态网页处理

很多网站都采用AJAX技术、SPA技术，部分内容都是异步动态加载的。可以提高用户体验，减少不必要的流量，方便CDN加速等。

但是，对于爬虫程序爬取到的HTML页面相当于页面模板了，动态内容不在其中。

解决办法之一，如果能构造一个包含JS引擎的浏览器，让它加载网页并和网站交互，我们编程从这个浏览器获取内容包括动态内容。这个浏览器不需要和用户交互的界面，只要能支持HTTP、HTTPS协议和服务器端交互，能解析HTML、CSS、JS就行了。

PhantomJS

它是一个headless无头浏览器，支持Javascript。可以运行在Windows、Linux、Mac OS等。
所谓无头浏览器，就是包含Js引擎、浏览器排版引擎等核心组件，但是没有和用户交互的界面的浏览器。

官网
http://phantomjs.org/

官方文档
http://phantomjs.org/documentation/

下载
http://phantomjs.org/download.html

下载对应操作系统的PhantomJS，**解压缩**就可以使用

Selenium

它是一个WEB自动化测试工具。它可以直接运行在浏览器中，支持主流的浏览器，包括PhantomJS（无界面浏览器）。

安装
pip install selenium

官网
https://www.seleniumhq.org/

开发实战

不同浏览器都会提供操作的接口，Selenium就是使用这些接口来操作浏览器

Selenium最核心的对象就是webdriver，通过它就可以操作浏览器、截图、HTTP访问、解析HTML等。

处理异步请求

bing的查询结果是通过异步请求返回结果，所以，直接访问页面不能直接获取到搜索结果。

```
# 获取bing查询数据
from selenium import webdriver # 核心对象
import datetime
```

```
import random
import time

# 指定PhantomJS的执行文件路径
driver = webdriver.PhantomJS(r'O:\phantomjs-2.1.1-windows\bin\phantomjs.exe')

driver.set_window_size(1280, 2400) # 设置窗口大小

# 打开网页GET方法，模拟浏览器地址栏输入网址
# http://cn.bing.com/search?q=马哥教育
url = "http://cn.bing.com/search?q=%E9%A9%AC%E5%93%A5%E6%95%99%E8%82%B2"
driver.get(url)


# 保存图片
def save_pic():
    base_dir = 'o:/'
    filename = "{}{:%Y%m%d%H%M%S}{:03}.png".format(base_dir, datetime.datetime.now(),
random.randint(1,100))
    driver.save_screenshot(filename)

save_pic() # 是否看到查询结果？


MAXRETRIES = 5 # 最大重试次数
for i in range(MAXRETRIES): # 循环测试
    time.sleep(1)
    try:
        ele = driver.find_element_by_id('b_content') # 如果查询结果来了，就会有这个id的标签
        print('ok')
        save_pic()
        break
    except Exception as e:
        print(e)

driver.close()
```

下拉框处理

Selenium专门提供了Select类来处理网页中的下拉框
不过下拉框用的页面越来越少了，本次使用`https://www.oschina.net/search?scope=project&q=python`

```
# oschina软件搜索
<select name='tag1' onchange="submit();">
        <option value='0'>所有分类</option>
        <option value='309' >Web应用开发</option>
        <option value='331' >手机/移动开发</option>
        <option value='364' >iOS代码库</option>
        <option value='12' >程序开发</option>
        <option value='11' >开发工具</option>
        <option value='273' >jQuery 插件</option>
        <option value='256' >建站系统</option>
```

```
        <option value='5' >企业应用</option>
        <option value='10' >服务器软件</option>
        <option value='6' >数据库相关</option>
        <option value='8' >应用工具</option>
        <option value='18' >插件和扩展</option>
        <option value='7' >游戏/娱乐</option>
        <option value='14' >管理和监控</option>
        <option value='9' >其他开源</option>
</select>
```

```
# 下拉框操作
from selenium import webdriver # 核心对象
import datetime
import random
from selenium.webdriver.support.ui import Select

driver = webdriver.PhantomJS(r'O:\phantomjs-2.1.1-windows\bin\phantomjs.exe')

driver.set_window_size(1280, 2400) # 设置窗口大小

# 保存图片
def save_pic():
    base_dir = 'o:/'
    filename = "{}{:%Y%m%d%H%M%S}{:03}.png".format(base_dir, datetime.datetime.now(),
random.randint(1,100))
    driver.save_screenshot(filename)


# 打开网页GET方法，模拟浏览器地址栏输入网址
url = "https://www.oschina.net/search?scope=project&q=python"
driver.get(url)

# 获取select
ele = driver.find_element_by_name('tag1') # 获取元素
print(ele.tag_name) # 标签名
print(driver.current_url)
save_pic()

s = Select(ele)
s.select_by_index(1) # web应用开发
print(driver.current_url) # 新页面

save_pic()
driver.close()
```

模拟键盘操作（模拟登录）

webdriver提供了一些列find方法，用户获取一个网页中的元素。元素对象可以使用send_keys模拟键盘输入。

oschina的登录页，登录成功后会跳转到首页，首页右上角会显示会员信息，如果未登录，无此信息。

```
# 模拟开源中国登陆
```

```python
from selenium import webdriver # 核心对象
import datetime
import random
from selenium.webdriver.common.keys import Keys
import time

driver = webdriver.PhantomJS(r'O:\phantomjs-2.1.1-windows\bin\phantomjs.exe')

driver.set_window_size(1280, 2400) # 设置窗口大小

# 保存图片
def save_pic():
    base_dir = 'o:/'
    filename = "{}{:%Y%m%d%H%M%S}{:03}.png".format(base_dir, datetime.datetime.now(),
random.randint(1,100))
    driver.save_screenshot(filename)


# 打开网页GET方法，模拟浏览器地址栏输入网址
url = "https://www.oschina.net/home/login"
driver.get(url)

save_pic()

# 模拟键盘输入
#'userMail' 'userPassword'
username = driver.find_element_by_id('userMail')
username.send_keys('wei.xu@magedu.com')
pwd = driver.find_element_by_id('userPassword')
pwd.send_keys('magedu.com18')

save_pic()

# 模拟回车
pwd.send_keys(Keys.ENTER)
print('-'*30)

print(driver.current_url) # 当前url

while True:
    time.sleep(1)
    print(driver.current_url)

    try:
        userinfo = driver.find_element_by_class_name('user-info')
        print(userinfo.text)  # 打印文本
        save_pic()
        break
    except Exception as e:
        print(e)

cookies = driver.get_cookies() # 获取长期登陆的cookie
print(cookies)
```

```
driver.close()
```

页面等待

越来越多的页面使用Ajax这样的异步加载技术，这就会导致代码中要访问的页面元素，还没有被加载就被访问了，抛出异常。

方法1 线程休眠

使用time.sleep(n)来等待数据加载。

配合循环一直等到数据被加载完成，可以解决很多页面动态加载或加载慢的问题。当然可以设置一个最大重试次数，以免一直循环下去。参看本文"处理异步请求"

方法2 Selenium等待

Selenium的等待分为：显示等待和隐式等待

隐式等待，等待特定的时间
显式等待，指定一个条件，一直等到这个条件成立后继续执行，也可以设置超时时间，超时会抛异常

参考 `https://www.seleniumhq.org/docs/04_webdriver_advanced.jsp#explicit-and-implicit-waits`

显示等待

expected_conditionsn内置条件	说明
title_is	判断当前页面的title是否精确等于预期
title_contains	判断当前页面的title是否包含预期字符串
presence_of_element_located	判断某个元素是否被加到了dom树里，并不代表该元素一定可见
visibility_of_element_located	判断某个元素是否可见.可见代表元素非隐藏，并且元素的宽和高都不等于0
visibility_of	跟上面的方法做一样的事情，只是上面的方法要传入locator，这个方法直接传定位到的element就好了
presence_of_all_elements_located	判断是否至少有1个元素存在于dom树中。举个例子，如果页面上有n个元素的class都是'column-md-3'，那么只要有1个元素存在，这个方法就返回True
text_to_be_present_in_element	判断某个元素中的text是否包含了预期的字符串
text_to_be_present_in_element_value	判断某个元素中的value属性是否包含了预期的字符串
frame_to_be_available_and_switch_to_it	判断该frame是否可以switch进去，如果可以的话，返回True并且switch进去，否则返回False
invisibility_of_element_located	判断某个元素中是否不存在于dom树或不可见
element_to_be_clickable	判断某个元素中是否可见并且是enable的，这样的话才叫clickable
staleness_of	等某个元素从dom树中移除，注意，这个方法也是返回True或False
element_to_be_selected	判断某个元素是否被选中了,一般用在下拉列表
element_selection_state_to_be	判断某个元素的选中状态是否符合预期
element_located_selection_state_to_be	跟上面的方法作用一样，只是上面的方法传入定位到的element，而这个方法传入locator
alert_is_present	判断页面上是否存在alert

```
# 定位搜索框，搜索电影
from selenium import webdriver # 核心对象
import datetime
import random

# 键盘操作
from selenium.webdriver.common.keys import Keys
# WebDriverWait 负责循环等待
from selenium.webdriver.support.wait import WebDriverWait
# expected_conditions条件，负责条件触发
from selenium.webdriver.support import expected_conditions as ec

from selenium.webdriver.common.by import By
```

```
driver = webdriver.PhantomJS(r'O:\phantomjs-2.1.1-windows\bin\phantomjs.exe')
driver.set_window_size(1280, 2400) # 设置窗口大小

# 打开网页GET方法，模拟浏览器地址栏输入网址
url = "https://movie.douban.com/" # 豆瓣电影
driver.get(url)

# 保存图片
def save_pic():
    base_dir = 'o:/'
    filename = "{}{:%Y%m%d%H%M%S}{:03}.png".format(base_dir, datetime.datetime.now(),
random.randint(1,100))
    driver.save_screenshot(filename)


try:
    ele = WebDriverWait(driver, 20).until(
        ec.presence_of_element_located((By.XPATH, '//input[@id="inp-query"]')) # 元素是否已经加载
到了dom树中
    ) # 使用哪个driver，等到什么条件ok，ec就是等待的条件
    ele.send_keys('TRON')
    ele.send_keys(Keys.ENTER)

    save_pic()
finally:
    driver.quit()
```

默认的查看频率是0.5秒每次，当元素存在则立即返回这个元素。

隐式等待

如果出现No Such Element Exception，则智能的等待指定的时长。缺省值是0。

```
from selenium import webdriver

# 页面隐式等待

driver = webdriver.PhantomJS(r'O:\phantomjs-2.1.1-windows\bin\phantomjs.exe')
# driver.implicitly_wait(10) # 增加这一句，全局设置，会导致下面找元素等待10秒
url = "https://movie.douban.com/"
driver.get(url)

try:
    print('begin-------')
    ele = driver.find_element_by_id('abcde')
except Exception as e:
    print(type(e)) # <class 'selenium.common.exceptions.NoSuchElementException'>
    print(e, '~~~~~~~~~~')
finally:
    driver.quit()
```

总结

Selenium的WebDriver是其核心，从Selenium2开始就是最重要的编程核心对象，在Selenium3中更是如此。

和浏览器交互全靠它，它可以：

- 打开URL，可以跟踪跳转，可以返回当前页面的实际URL
- 获取页面的title
- 处理cookie
- 控制浏览器的操作，例如前进、后退、刷新、关闭，最大化等
- 执行JS脚本
- 在DOM中搜索页面元素Web Element，指定的或一批，find系方法
- 操作网页元素
    - 模拟下拉框操作Select(element)
    - 在元素上模拟鼠标操作click()
    - 在元素上模拟键盘输入send_keys()
    - 获取元素文字 text
    - 获取元素的属性 get_attribute()

Selenium通过WebDriver来驱动浏览器工作，而浏览器是一个个独立的浏览器进程。