[python爬虫] BeautifulSoup和Selenium对比爬取豆瓣Top250电影信息

原创 Eastmount 最后发布于2016-12-30 00:19:54 阅读数 14789 ☆ 收藏 展开

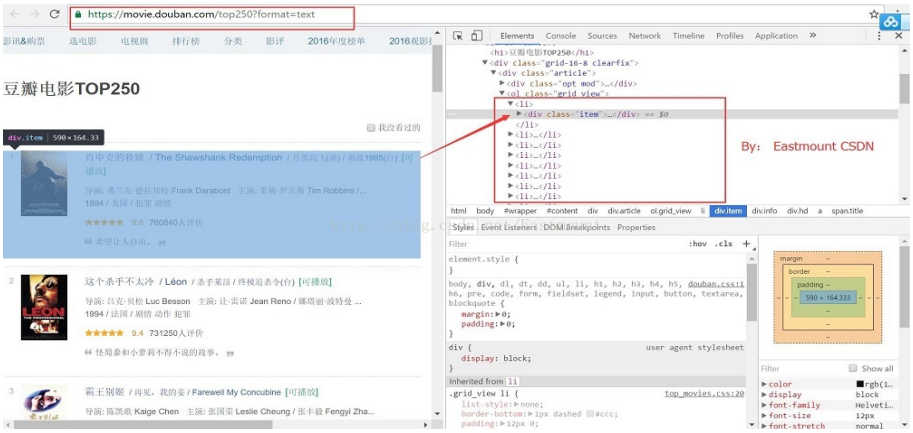
这篇文章主要对比BeautifulSoup和Selenium爬取豆瓣Top250电影信息，两种方法从本质上都是一样的，都是通过分析网页的DOM树结构进行元素定位，再定向爬取具体的电影信息，通过代码的对比，你可以进一步加深Python爬虫的印象。同时，文章给出了我以前关于爬虫的基础知识介绍，方便新手进行学习。

总之，希望文章对你有所帮助，如果存在不错或者错误的地方，还请海涵~

一. DOM树结构分析

豆瓣Top250电影网址：https://movie.douban.com/top250?format=text

通过右键Chrome浏览器"审查元素"或"检查"可以定位具体的元素，如下图所示：



图中由一部部电影构成，在HTML中对应：

 .....

BeautifulSoup 通过 soup.find_all(attrs={"class":"item"}) 函数可以获取具体的信息，然后再定位具体内容，如获取标题， 获取分数和评价数。

下一步需要注意的是，定位爬取网页元素还需要进行翻页，通常有两种方法：
1.点击下一页分析URL网址进行分析它们之间的规律；
2.Selenium可以获取页码按钮进行自动点击跳转。
如下图所示，点击不同页码后分析URL：

第2页URL：https://movie.douban.com/top250?start=25&filter=
第3页URL：https://movie.douban.com/top250?start=50&filter=
所以每页共25部电影，它们是存在规律的，再写一个循环即可获取所有电影信息。

二. BeautifulSoup爬取豆瓣信息

入门推荐我的前文：[python知识] 爬虫知识之BeautifulSoup库安装及简单介绍
具体代码如下：

```
# -*- coding: utf-8 -*-
"""
Created on 2016-12-29 22:50

@author: Easstmount
"""

import urllib2
import re
from bs4 import BeautifulSoup
import codecs
```

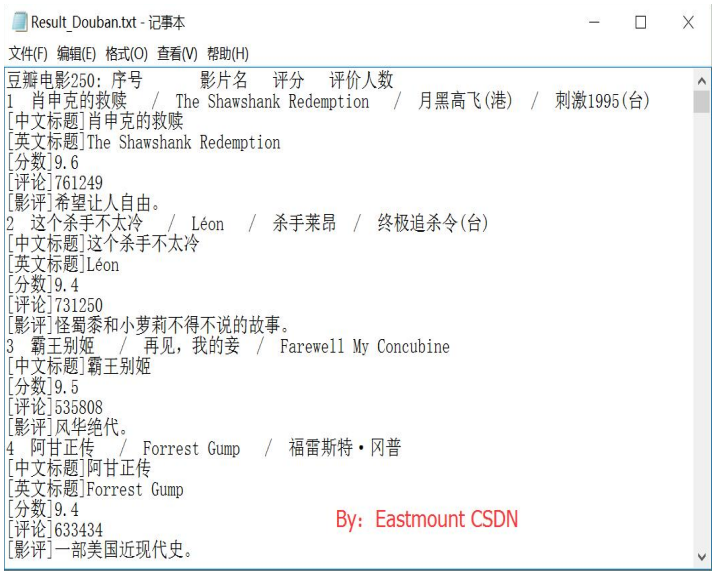
```
  #爬虫函数
def crawl(url):
    page = urllib2.urlopen(url)
    contents = page.read()
    soup = BeautifulSoup(contents, "html.parser")
    print u'豆瓣电影250: 序号 \t影片名\t 评分 \t评价人数'
    infofile.write(u"豆瓣电影250: 序号 \t影片名\t 评分 \t评价人数\r\n")
    print u'爬取信息如下:\n'
    for tag in soup.find_all(attrs={"class":"item"}):
        #print tag
        #爬取序号
        num = tag.find('em').get_text()
        print num
        #爬取电影名称
        name = tag.find(attrs={"class":"hd"}).a.get_text()
        name = name.replace('\n',' ')
        print name
        infofile.write(num+" "+name+"\r\n")
        #电影名称
        title = tag.find_all(attrs={"class":"title"})
        i = 0
        for n in title:
            text = n.get_text()
            text = text.replace('/','')
            text = text.lstrip()
            if i==0:
                print u'[中文标题]', text
                infofile.write(u"[中文标题]" + text + "\r\n")
            elif i==1:
                print u'[英文标题]', text
                infofile.write(u"[英文标题]" + text + "\r\n")
            i = i + 1
        #爬取评分和评论数
        info = tag.find(attrs={"class":"star"}).get_text()
        info = info.replace('\n',' ')
        info = info.lstrip()
        print info
        mode = re.compile(r'\d+\.?\d*')
        print mode.findall(info)
        i = 0
        for n in mode.findall(info):
            if i==0:
                print u'[分数]', n
                infofile.write(u"[分数]" + n + "\r\n")
            elif i==1:
                print u'[评论]', n
                infofile.write(u"[评论]" + n + "\r\n")
            i = i + 1
        #获取评语
        info = tag.find(attrs={"class":"inq"})
        if(info): # 132部电影 [消失的爱人] 没有影评
            content = info.get_text()
            print u'[影评]', content
            infofile.write(u"[影评]" + content + "\r\n")
        print ''


#主函数
if __name__ == '__main__':

    infofile = codecs.open("Result_Douban.txt", 'a', 'utf-8')
```

```
    url = 'http://movie.douban.com/top250?format=text'
                                                        i = 0
    while i<10:
        print u'页码', (i+1)
        num = i*25 #每次显示25部 URL序号按25增加
        url = 'https://movie.douban.com/top250?start=' + str(num) + '&filter='
        crawl(url)
        infofile.write("\r\n\r\n\r\n")
        i = i + 1
    infofile.close()
```

输出结果如下所示：

```
豆瓣电影250: 序号          影片名     评分     评价人数
1  肖申克的救赎  / The Shawshank Redemption  / 月黑高飞(港)  /  刺激1995(台)
[中文标题]肖申克的救赎
[英文标题]The Shawshank Redemption
[分数]9.6
[评论]761249
[影评]希望让人自由。
2  这个杀手不太冷  / Léon  / 杀手莱昂  /  终极追杀令(台)
[中文标题]这个杀手不太冷
[英文标题]Léon
[分数]9.4
[评论]731250
[影评]怪蜀黍和小萝莉不得不说的故事。
3  霸王别姬  / 再见，我的妾  /  Farewell My Concubine
[中文标题]霸王别姬
[分数]9.5
[评论]535808
[影评]风华绝代。
4  阿甘正传  / Forrest Gump  / 福雷斯特·冈普
[中文标题]阿甘正传
[英文标题]Forrest Gump
[分数]9.4
[评论]633434
[影评]一部美国近现代史。
5  美丽人生  / La vita è bella  / 一个快乐的传说(港)  /  Life Is Beautiful
[中文标题]美丽人生
[英文标题]La vita è bella
[分数]9.5
[评论]364132
[影评]最美的谎言。
6  千与千寻  / 千と千尋の神隠し  / 神隐少女(台)  /  Spirited Away
[中文标题]千与千寻
[英文标题]千と千尋の神隠し
[分数]9.2
[评论]584559
[影评]最好的宫崎骏，最好的久石让。
```

同时输出文件Reseult_Douban.txt，如下图所示：



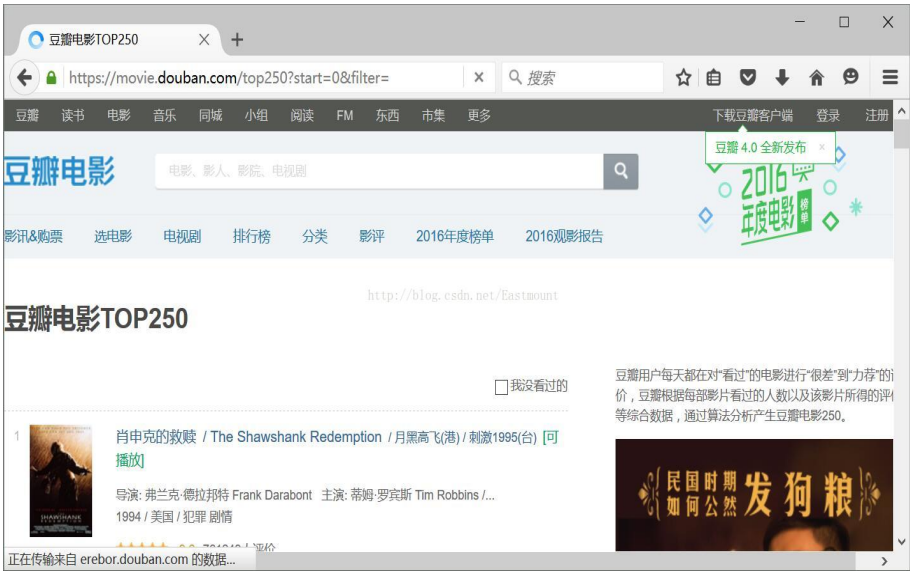
三. Selenium爬取豆瓣信息及Chrome爬虫介绍

入门推荐我的前文：[Python爬虫] Selenium自动登录和Locating Elements介绍
代码如下所示：

```
# -*- coding: utf-8 -*-
"""
Created on 2016-12-29 22:50

@author: Easstmount
"""

import time
import re
import sys
import codecs
import urllib
from selenium import webdriver
from selenium.webdriver.common.keys import Keys


#爬虫函数
def crawl(url):
    driver.get(url)
    print u'豆瓣电影250: 序号 \t影片名\t 评分 \t评价人数'
    infofile.write(u"豆瓣电影250: 序号 \t影片名\t 评分 \t评价人数\r\n")
    print u'爬取信息如下:\n'
    content = driver.find_elements_by_xpath("//div[@class='item']")
    for tag in content:
        print tag.text
        infofile.write(tag.text+"\r\n")
        print ''


#主函数
if __name__ == '__main__':
```

```
driver = webdriver.Firefox()
                                infofile = codecs.open("Result_Douban.txt", 'a', 'utf-8')
url = 'http://movie.douban.com/top250?format=text'
i = 0
while i<10:
    print u'页码', (i+1)
    num = i*25 #每次显示25部 URL序号按25增加
    url = 'https://movie.douban.com/top250?start=' + str(num) + '&filter='
    crawl(url)
    infofile.write("\r\n\r\n\r\n")
    i = i + 1
infofile.close()
```

该部分代码会自动调用Firefox浏览器，然后爬取内容。调用如下所示：



同时，也可以爬取文件如下图所示，也可以再定向分析具体的节点，其方式方法也是类似的。

调用Chrome浏览器需要在：
C:\Program Files (x86)\Google\Chrome\Application

路径下放置一个 chromedriver.exe 驱动文件，再进行调用。核心代码：

```
chromedriver = "C:\Program Files (x86)\Google\Chrome\Application\chromedriver.exe"
os.environ["webdriver.chrome.driver"] = chromedriver
driver = webdriver.Chrome(chromedriver)
```

但是可能会报错如下所示，需要保持版本一致。

总结下两个代码的优缺点：BeautifulSoup比较快速，结构更加完善，但爬取如CSDN等博客会报错Forbidden；而Selenium可以调用浏览器进行爬取，自动化操作及动态操作，点击鼠标键盘等按钮比较方便，但其速度比较慢，尤其是重复的调用浏览器。

最近年尾学院事情太多了，所以很少有定量的时间进行写博客，这其实挺悲伤的，但幸运的是遇见了她，让我在百忙

之中还是体会到了一些甜蜜，陪着我工作。

无需多言，彼此的心意一言一行里都能感受到爱意和温暖，follow you~

(By:Eastmount 2016-12-30 晚上12点半 http://blog.csdn.net/eastmount/)