

Thiago Marzagão

data, code, etc

- [home](#)
- [about](#)
- [publications](#)
- [code](#)
- [talks & slides](#)
- [teaching](#)
- [committees](#)
- [resources](#)
- [index](#)

webscraping with Selenium - part 1

12 Nov 2013

If you are webscraping with Python chances are that you have already tried `urllib`, `httplib`, `requests`, etc. These are excellent libraries, but some websites don't like to be webscraped. In these cases you may need to disguise your webscraping bot as a human being. Selenium is just the tool for that. Selenium is a webdriver: it takes control of your browser, which then does all the work. Hence what the website "sees" is Chrome or Firefox or IE; it does not see Python or Selenium. That makes it a lot harder for the website to tell your bot from a human being.

In this tutorial I will show you how to webscrape with Selenium. This first post covers the basics: locating HTML elements and interacting with them. Later posts will cover things like downloading, error handling, dynamic names, and mass webscraping.

There are Selenium bindings for Python, Java, C#, Ruby, and Javascript. All the examples in this tutorial will be in Python, but translating them to those other languages is trivial.

installing Selenium

To install the Selenium bindings for Python, simply use PIP:

```
pip install selenium
```

You also need a “driver”, which is a small program that allows Selenium to, well, “drive” your browser. This driver is browser-specific, so first we need to choose which browser we want to use. For now we will use Chrome (later we will switch to PhantomJS). Download the latest version of the [chromedriver](#), unzip it, and note where you saved the unzipped file.

choosing our target

In this tutorial we will webscrape [LexisNexis Academic](#). It’s a gated database but you are probably in academia (just a guess) so you should have access to it through your university.

(Obs.: LexisNexis Academic is set to have a new interface starting December 23rd, so if you are in the future the code below may not work. It will still help you understand Selenium though. And adapting it to the new LexisNexis interface will be a nice learning exercise.)

opening a webpage

Now on to coding. First we start the webdriver:

```
from selenium import webdriver

path_to_chromedriver = '/Users/yourname/Desktop/chromedriver'
browser = webdriver.Chrome(executable_path = path_to_chromedriver)
```

When you run this code you’ll see a new instance of Chrome magically launch.

Now let’s open the page we want:

```
url = 'https://www.lexisnexis.com/hottopics/lnacademic/?verb=
browser.get(url)
```


The page looks like this:

General Searching

» Easy Search™

» Advanced Search

» **Tip:** Click the headings below to view links to specialized search forms and other useful features.

**LexisNexis®
FREE Pre-Law
Program**Learn what to expect in
law school this fall.Sign up NOW. 

News

US Legal

International Legal

Companies

Country Information

Subject Areas

Sources

Guides & Resources

Advanced Search

 [Help](#) [Clear](#)

Use of this service is subject to Terms and Conditions

Search Type:

☒ Terms & Connectors ☐ Natural Language 


Search Terms:

Search


Specify Date:

All available dates 

Add Index Terms:


[Company](#) [Industry](#) [Subject](#) [Geography](#) [People](#) 

Select Source:

By Type: 

By Name:

Start typing a title like **New York Times**

» Try also [Find Sources](#) Or [Browse Sources](#) 

[About LexisNexis](#) | [Terms and Conditions](#) | [Privacy Policy](#)
Copyright © 2013 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.

locating page elements

Before we fill out forms and click buttons we need to locate these elements. This step is going to be easier if you know some HTML but that is not a pre-requisite (you will end up learning some HTML on-the-fly as you do more and more webscraping).

A page element usually has a few attributes - a name, an id, a CSS selector, an xpath, etc. (Don't worry if you've never heard of these things before.) We can use these attributes to help us locate the element we want.

How can we find what these attributes are for a given element? Simple: just right-click it and choose "Inspect Element". Your browser will then show you the corresponding HTML code. For instance, if you do this with the "Search Terms" form on the page we opened above you'll see something like this:

General Searching

» Easy Search™

» Advanced Search

» **Tip:** Click the headings below to view links to specialized search forms and other useful features.

LexisNexis®
FREE Pre-Law Program

Learn what to expect in law school this fall.

Sign up NOW.

Advanced Search

Use of this service is subject to Terms and Conditions

Search Type: ☒ Terms & Connectors ☐ Natural Language

Search Terms: **Search**

Specify Date: All available dates

Add Index Terms: [Company](#) [Industry](#) [Subject](#) [Geography](#) [People](#)

Select Source: By Type: --- Select ---
By Name:
Start typing a title like **New York Times**
» Try also Find Sources Or Browse Sources

```

x Elements Resources Network Sources Timeline Profiles Audits Console
<input type="hidden" id="newSearch" name="newSearch" value="true">
<input type="hidden" id="indexTerms" name="indexTerms" value="">
<div class="formArea">
  <ol class="formArea">
    <li></li>
    <li>
      <label for="terms">Search Terms:</label>
      <div class="buttonArea" style="vertical-align: top;">
        <span class="buttonArea" style="vertical-align: top;">
          <li id="boolDate" class="masterDate"></li>
          <li id="termsAdded" style="display: none;"></li>
          <li id="termsAddedText" style="display: none;"></li>
        </li>
      </div>
    </li>
  </ol>
</div>

```

The HTML code of the element you selected appears highlighted in blue. Let me copy and paste it below, so you can have a better look at it:

```
<div class="buttonArea" style="vertical-align: top;">
  <span class="buttonArea" style="vertical-align: top;">
    <li id="boolDate" class="masterDate"></li>
    <li id="termsAdded" style="display: none;"></li>
    <li id="termsAddedText" style="display: none;"></li>
  </li>
</div>
```

Ha! Now we know two attributes of the “Search Terms” form: its name is “terms” and its id is (also) “terms”.

We are not ready to locate the element though. HTML pages usually contain multiple “frames” and our element is probably inside one of these frames. We need to know which one. To find out, start on that blue-highlighted line we saw before and keep scrolling up until you find `<frame>`. You’ll eventually find this line:

```
<div class="buttonArea" style="vertical-align: top;">
  <span class="buttonArea" style="vertical-align: top;">
    <li id="boolDate" class="masterDate"></li>
    <li id="termsAdded" style="display: none;"></li>
    <li id="termsAddedText" style="display: none;"></li>
  </li>
</div>
```

That means our “Search Terms” form is inside a frame named “mainFrame”. Now keep scrolling up to see if “mainFrame” is inside some other frame. Here it is not, but that is always a possibility and you need to check.

The next thing we do is go to that frame. Here is how we do it:

```
browser.switch_to_frame('mainFrame')
```

Once we are on the correct frame we can finally search for the element. Let's search it using its id:

```
browser.find_element_by_id('terms')
```

And that's it. We have located the element.

see the beauty?

As the code above shows, Selenium is very intuitive. To switch frames we use `switch_to_frame`. To find an element by its id we use `find_element_by_id`. And so on.

Another great feature of Selenium is that it's very similar across all languages it supports. In Java, for instance, this is how we switch frames and find elements by id:

```
browser.switchTo().frame("frameName");  
browser.findElement(By.id("elementId"));
```

So even if you first learn Selenium in Python it's very easy to use it in other languages later.

interacting with page elements

Now that we've found the "Search Terms" form we can interact with it. First we want to make sure that the form is empty:

```
browser.find_element_by_id('terms').clear()
```

Now we can write on the form. Here we are interested in all occurrences of the word "balloon" in the news today. We start by writing "balloon" on the form:

```
browser.find_element_by_id('terms').send_keys('balloon')
```

Next we need to specify the date. There is a "Specify Date" drop-down menu. Let us locate it. As usual we start by right-clicking the element and selecting "Inspect Element". That gives us the following HTML code:

```
<select class="input" id="dateSelector1" style="vertical-align: top;">  
  <option value="">All available dates</option>  
  <option value="0:DY">Today</option>  
  <option value="is">Date is...</option>  
  <option value="before">Date is before...</option>  
  <option value="after">Date is after...</option>  
  <option value="from">Date is between...</option>  
  <option value="1:WK">Previous week</option>
```

```

<option value="1:MO">Previous month</option>
<option value="3:MO">Previous 3 months</option>
<option value="6:MO">Previous 6 months</option>
<option value="1:YR">Previous year</option>
<option value="2:YR">Previous 2 years</option>
<option value="5:YR">Previous 5 years</option>
<option value="previous">Previous...</option></select>

```

We can see the element's name and id but here we will use neither. This is a drop-down menu and we will need to select one of its options ("All available dates", "Today", etc), so here we will use the element's xpath. How do you get it? We are using Chrome here, so this is really simple: we just right-click the blue-highlighted line that corresponds to the element's HTML code and select "Copy XPath". Like this:

LexisNexis® Academic

Academic is getting an interface refresh on 12/23. Please click here for more information.

Home

General Searching

» Easy Search™

» Advanced Search

» Tip: Click the headings below to view links to specialized search forms and other useful features.

LexisNexis® FREE Pre-Law Program

Learn what to expect in law school this fall.

Sign up NOW.

Advanced Search

Use of this service is subject to Terms and Conditions

Search Type: ☒ Terms & Connectors ☐ Natural Language

Search Terms:

Specify Date:

Add Index Terms:

Select Source:

Elements Resources Network Sources Timeline Profiles Audits Console

<input type="hidden" id="_form_locator_1">

<select style="vertical-align:top" name="dateSelector1">

<option value="All available dates">All available dates</option>

<option value="0:DY">Today</option>

<option value="is">Date is...</option>

<option value="before">Date is before...</option>

<option value="after">Date is after...</option>

<option value="from">Date is between...</option>

<option value="1:WK">Previous week</option>

<option value="1:MO">Previous month</option>

<option value="3:MO">Previous 3 months</option>

Copy XPath

That gives us the following xpath:

```
//*[@id="dateSelector1"]
```

Now, as usual, scroll up from the blue-highlighted line until you find out which frame contains the element. Here that is the same frame of "Search Terms" (i.e., "mainFrame"), so we are already there, no need to move.

If all we wanted were to locate the element, we would do this:

```
browser.find_element_by_xpath('//*[@id="dateSelector1"]')
```

But we want to open that drop-down menu and select “Today”. So we do this instead:

```
browser.find_element_by_xpath('//*[@id="dateSelector1"]/option
```



Now we’ve entered our search term (balloon) and selected our date (today). Next we need to select our news sources. That’s another drop-down menu, a bit further down the page. You know the drill: right-click the element, retrieve relevant attributes, scroll up to find out the frame. There isn’t anything new to learn here (and we haven’t left “mainFrame” yet), so I’ll just give you the code (let’s say we want to select all news sources in English):

```
browser.find_element_by_xpath('//*[@id="byType"]/option[text
```



Finally, we need to click the “Search” button (next to the “Search Terms” form) to submit the search. Same drill: right-click element, get attributes, scroll up to find frame. Except that here there is no id or name:

```
<input type="submit" value="Search" />
```

So we need to use xpath again, even though this is not a drop-down menu:

```
browser.find_element_by_xpath('//*[@id="searchForm"]/fieldset
```



Now that is one ugly-looking xpath. Our code will look better if we use the element’s CSS selector instead:

```
browser.find_element_by_css_selector('input[type="submit"]')
```



I don’t know of any “copy-and-paste” way to get an element’s CSS selector, but if you stare at the line above long enough you can see how it derives from the element’s HTML code.

That’s it. You should now see Chrome leaving the search page and opening the results page.

There is a lot more to cover, but that will have to wait.

(Part 2)

9 Comments

thiagomarzagao



Disqus' Privacy Policy



Login ▾



Recommend 5



Tweet



Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

**bend94** • 2 years ago

hi

i try this code from selenium import webdriver

browser = webdriver.Firefox()

browser.get("http://www.google.com")

browser.close

but i catch this error :

Traceback (most recent call last):

File "./selpython3.py", line 2, in <module>

browser = webdriver.Firefox()

File "/usr/local/lib/python2.7/dist-

packages/selenium/webdriver/firefox/webdriver.py", line 158, in __init__

keep_alive=True)

File "/usr/local/lib/python2.7/dist-

packages/selenium/webdriver/remote/webdriver.py", line 154, in __init__

self.start_session(desired_capabilities, browser_profile)

File "/usr/local/lib/python2.7/dist-

.../usr/local/lib/python2.7/dist-packages/selenium/webdriver/remote/webdriver.py", line 154, in __init__

[see more](#)

^ | ▾ • Reply • Share ›

**Pawel** ➔ bend94 • 2 years ago

Hi,

make sure your driver can connect to X server, see

<https://stackoverflow.com/q...>

^ | ▾ • Reply • Share ›

**Kushal Mukherjee** • 3 years ago • edited

Yes - Following the tutorial, it seems that Lexis Nexus website has changed. So when you reach the part of the tutorial to submit search it will prompt for a username and password page. Please advise the best approach. Would like to complete tutorial

^ | ▾ • Reply • Share ›

**Mezooz** • 3 years ago



I have a question regarding login information. I use LexisNexis through my University and the Chrome extension doesn't keep me logged in (thus ending the script prematurely). I was wondering if you knew a quick way around this?

^ | v • Reply • Share ›



ScraperHunk • 4 years ago

When website is protected with anti-scraping technology then same website can be scraped by using software that can mimic human behaviour that is what selenium does. For more information visit my website ;

<http://www.srturl.in/z0xw>

^ | v • Reply • Share ›



White Ninja • 4 years ago

Good article. I am (here is my website to look at : <http://www.srturl.in/k0xh>) using all the above mentioned libraries as well as selenium web driver for my web scraping work.

^ | v • Reply • Share ›



Micael Lopes • 5 years ago

Ótimo.

^ | v • Reply • Share ›



c.h.i. • 5 years ago

nice

^ | v • Reply • Share ›



Shrayas • 5 years ago

Thank you for this :) Helped me quite a bit. Cheers!