How To Scrape Business Public Records

From the website:

> The California Business Search provides access to available information for corporations, limited liability companies and limited partnerships of record with the California Secretary of State, with free PDF copies of over 17 million imaged business entity documents, including the most recent imaged Statements of Information filed for Corporations and Limited Liability Companies.

Tech-Stack

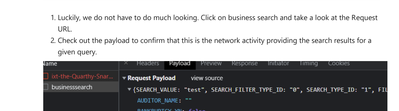
- Python
  - Selenium for browser automation (see why below)
  - BeautifulSoup for html parsing
  - pandas for dataframe formatting, processing, and export (using modin which is pandas on steroids)
- Browser Developer Tools (F12 on Windows)

I. Researching How The Site Works

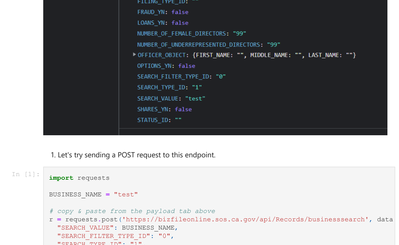
Technology Used: Browser Developer Tools

Discover API endpoint

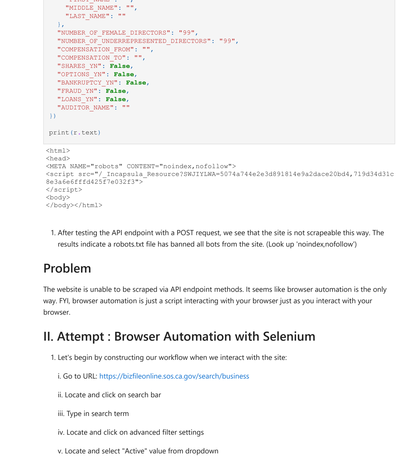
1. Hit F12 to open your browser console
2. Navigate to the webpage: https://bizfileonline.sos.ca.gov/search/business
3. On the browser console, click on the Network tab and select the Fetch/XHR filter



4. Search up a business name or keyword in the search bar and observe.
5. Identify the network activity that provides the results. Make sure the sliding bar expands the whole time range.



6. Luckily, we do not have to do much looking. Click on business search and take a look at the Request URL.
7. Check out the payload to confirm that this is the network activity providing the search results for a given query.



8. Let's try sending a POST request to this endpoint.

In [1]:
```
import requests

BUSINESS_NAME = "test"

# copy & paste from the payload tab above
r = requests.post('https://bizfileonline.sos.ca.gov/api/Records/businesssearch', data=
  "SEARCH_VALUE": BUSINESS_NAME,
  "SEARCH_FILTER_TYPE_ID": "0",
  "SEARCH_TYPE_ID": "1",
  "FILING_TYPE_ID": "",
  "STATUS_ID": "",
  "FILING_DATE": {
    "start": None,
    "end": None
  },
  "CORPORATION_BANKRUPTCY_YN": False,
  "CORPORATION_LEGAL_PROCEEDINGS_YN": False,
  "OFFICER_OBJECT": {
    "FIRST_NAME": "",
    "MIDDLE_NAME": "",
    "LAST_NAME": ""
  },
  "NUMBER_OF_FEMALE_DIRECTORS": "99",
  "NUMBER_OF_UNDEREPRESENTED_DIRECTORS": "99",
  "CORPORATION_FROM": "",
  "CORPORATION_TO": "",
  "SHARES_YN": False,
  "OPTIONS_YN": False,
  "BANKRUPTCY_YN": False,
  "FRAUD_YN": False,
  "LOANS_YN": False,
  "AUDITOR_NAME": ""
})

print(r.text)
```

```
<html>
<head>
<META NAME="robots" CONTENT="noindex,nofollow">
<script src="/_Incapsula_Resource?SWJIYLWA=5074a744e2e3d891814e9a2dace20bd4,719d34d31c
8e3a6e6fffd425f7e032f3">
</script>
<body>
</body></html>
```

9. After testing the API endpoint with a POST request, we see that the site is not scrapeable this way. The results indicate a robots.txt file has banned all bots from the site. (Look up 'noindex,nofollow')

Problem

The website is unable to be scraped via API endpoint methods. It seems like browser automation is the only way. FYI, browser automation is just a script interacting with your browser just as you interact with your browser.

II. Attempt : Browser Automation with Selenium

1. Let's begin by constructing our workflow when we interact with the site:

   i. Go to URL: https://bizfileonline.sos.ca.gov/search/business

   ii. Locate and click on search bar

   iii. Type in search term

   iv. Locate and click on advanced filter settings

   v. Locate and select "Active" value from dropdown

   vi. Locate and click "Search"

   v. Read results

1. Lets see how this works as a script.

Setting up the Selenium object

In [ ]:
```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.options import Options

options = Options()
options.binary_location = "<path to chrome.exe>"
driver = webdriver.Chrome(options = options)
```

Navigate to page

In [ ]:
```
driver.get("https://bizfileonline.sos.ca.gov/search/business")
```

Locate and type into search bar

In [ ]:
```
search = driver.find_element_by_xpath("//*[@id='root']/div/div[1]/div/main/div/div[2]
search.send_keys('test') # using Keys module
```

Wait until "Advanced Settings" button is clickable, then focus on it and hit "Enter"

In [ ]:
```
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait, Select

try:
  wait = WebDriverWait(driver, 10)
  element = wait.until(EC.element_to_be_clickable((By.XPATH, "//*[@id='root']/div/div
except Exception as e:
  print(e)
  driver.quit()

advanced_settings = driver.find_element_by_xpath("//*[@id='root']/div/div[1]/div/main/
advanced_settings.send_keys("")
advanced_settings.send_keys(Keys.ENTER)
```

Wait until dropdown is clickable, then focus on it and select "Active"

In [ ]:
```
try:
  wait = WebDriverWait(driver, 10)
  element = wait.until(EC.element_to_be_clickable((By.XPATH, "//*[@id='field-STATUS_ID
except Exception as e:
  print(e)
  driver.quit()
dropdown = driver.find_element_by_xpath("//*[@id='field-STATUS_ID']")
dropdown.send_keys("")
Select(dropdown).select_by_value("1")
```

Wait until search button is clickable, then focus on it and hit "Enter"

In [ ]:
```
try:
  wait = WebDriverWait(driver, 10)
  element = wait.until(EC.element_to_be_clickable((By.XPATH, "//*[@id='root']/div/div
except Exception as e:
  print(e)
  driver.quit()

advanced_settings = driver.find_element_by_xpath("//*[@id='root']/div/div[1]/div/main/
advanced_settings.send_keys("")
advanced_settings.send_keys(Keys.ENTER)
```

Wait for search results to show up and extract page HTML

In [ ]:
```
try:
  wait = WebDriverWait(driver, 60)
  table = wait.until(EC.presence_of_element_located((By.XPATH, "//*[@id='root']/div/d
except Exception as e:
  print(e)
  driver.quit()

html = driver.page_source
```

Parse page HTML and convert to datatable for export

In [ ]:
```
from bs4 import BeautifulSoup
import modin.pandas as pd

soup = BeautifulSoup(html)
soup_table = soup.find("table")
table = pd.read_html(str(soup_table))

table.to_csv('out.csv', index=False)

driver.quit()
```

III. Limitations

Limitations stem from how the website is designed. The designers made it difficult for scrapers to gather information by limiting scraping to the search term. In other words, we have to know what to look for in order to get the most useful information.

Further attempts should test the ability to do rapid-fire search on the site.

Full Script Available Here

https://github.com/SirAgathon/bizscraper