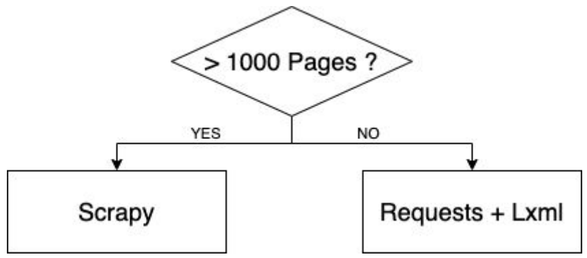


Data Extraction (python) Cheat Sheet

V2020.12.02

(Dr Yan Xu)

Basic Questions

1. **Where is Data?**
   - HTML
   - API

   How? -> Disable JavaScript in Browser (Chrome):
   - Open DevTools: Command+Option+I
   - Disable JS: Command + Shift + P
2. **How Many Pages/APIs?**
   - < 1000 => Request + Lxml
   - > 1000 => Scrapy

Fundamental: XPath

1. XPath is a query language for selecting elements from an XML document.
2. Syntax
   - descendant selector: //
   - child selector: /
   - Index selector: [n] or last() (note: starting from 1)
   - attribute selectors: //*[@key="value"]
   - attribute func:
     - starts-with(), ends-with(), contains()
   - text match: text()
   - match everything: *
   - logic: and, or, not
   - relative match: .
3. Examples
   - //a/@href #hrefs of all anchors
   - ./table/tr[last()] # the last row of a table
   - //a[contains(text(),"Click")]
   - //div[@id="abc" or @name="efg"]
   - //*[@class="class"]
   - //div/*[1] #first child in div

Fundamental: CSS Selectors [*Optional*]

1. Another query language for selecting elements from an XML document.
2. Syntax
   - class search: .class = XX
   - id search: #id = XX
   - descendant search: div p
   - parent search: div > p
   - immediately after search: div + p
   - attribute value search: [attribute=value]
   - begins with: [href^="https"]
   - ends with: [href$=".pdf"]
   - contains: [href*="w3schools"]
   - order: nth-child(n)
3. Examples
   - #Lastname
   - li:nth-child(1)
   - [id=my-Address]
   - .intro, #Lastname

Request + Lxml

- example code:

```
response = requests.get(url)
root = lxml.html.fromstring(response.content)
elemnt = root.xpath("//article[@class='Box-row']//h1/a/text()")
print(elemnt)
```

- example repository:
  - IMDB scraping

Scrapy

- example code:

```
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://blog.scrapinghub.com']

    def parse(self, response):
        for title in response.css('.post-header>h2'):
            yield {'title': title.css('a ::text').get()}

        for next_page in response.css('a.next-posts-link'):
            yield response.follow(next_page, self.parse)
```

- example repository:
  - Zillow scraping
- benefits: 10+ faster
- a higher learning curve

Selenium

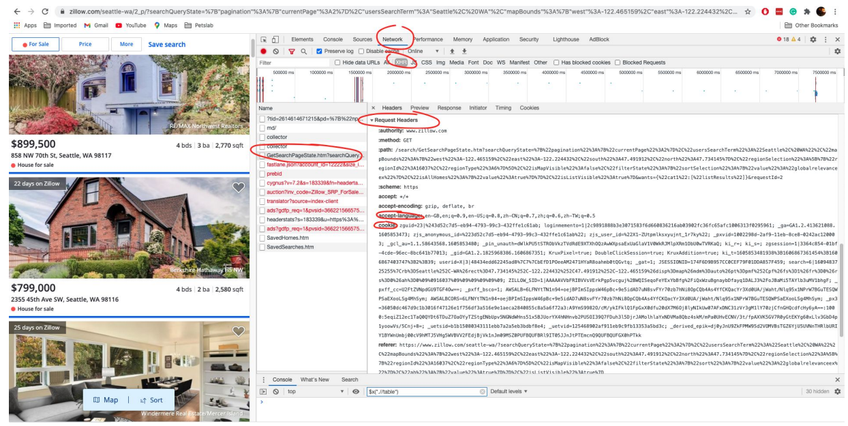
- example code:

```
driver = webdriver.Chrome()
driver.get(url)
root = lxml.html.fromstring(driver.page_source)
element = root.xpath("//div[@class='activity-card__details']//h1/a/text()")
print(element)
```

- example repository: code
- slow and unstable. (to avoid)
- optional if user login (or other actions) is required before data scraping

API Explore

- If Data in API, need to explore:
  - Which APIs?
  - Session ID?
  - Cookie?
  - Other headers?
- Example



- Session
  - Get an session ID from the Browser header
- Cookie (code)
  - may change API responses

Other Issues

- Pagination
  - Solution 1: Pre-calculate N of Pages
  - Solution 2: Follow "Next Page Button:
- Google ReCaptcha
  - Very difficult to deal with after v3
  - Solution Candidates:
    - mimic real request header
    - reduce frequency & random wait