

SCRAPING

Erin Rossiter

This week

- I use what we go over today (and this week) a lot.
- Scraping, APIs, text handling is where Python beats R.
- The frustrating stuff from last week (exception handling!) is very useful here.

First things first: Page Source

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Department of Political Science

DIRECTORY

UNDERGRADUATE

GRADUATE

HIRING A PH.D.

EVENTS

NEWS

[Home](#) / [The Directory](#) / [Current Faculty](#)

Faculty

by Specialization

Faculty

Faculty

Visiting Lecturers

Students

Current Faculty

James F. Spriggs II

Sidney W. Souers Professor
Chair, Department of
*American Politics; Executive
Supreme Court*

Matthew Gabel

Professor and Associate
Faculty, Charles F. Brannan
*Comparative Politics,
Policy*Back
Forward
ReloadSave As...
Print...
Cast...
Translate to English

View Page Source

Inspect

Speech

<https://polisci.wustl.edu/faculty>

```
Current Faculty | Department of Political Science | view-source:https://polisci.wustl.edu/faculty
view-source:https://polisci.wustl.edu/faculty
Apps email posters Google Scholar TYP github nguyen sits... textdata Cristian Danescu-N... Charity Navigator ... writeup Python »
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" lang="en" dir="ltr"
3   xmlns:content="http://purl.org/rss/1.0/modules/content/"
4   xmlns:dc="http://purl.org/dc/terms/"
5   xmlns:foaf="http://xmlns.com/foaf/0.1/"
6   xmlns:og="http://ogp.me/ns#"
7   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
8   xmlns:sioc="http://rdfs.org/sioc/ns#"
9   xmlns:sioct="http://rdfs.org/sioc/types#"
10  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
11  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" version="HTML+RDFa 1.1">
12   <head>
13     <meta charset="utf-8" />
14     <link rel="apple-touch-icon" href="/sites/all/themes/awesomesauce/images/iOS-57.png" />
15     <link rel="apple-touch-icon" href="/sites/all/themes/awesomesauce/images/iOS-114.png" sizes="114x114" />
16     <link rel="apple-touch-icon" href="/sites/all/themes/awesomesauce/images/iOS-72.png" sizes="72x72" />
17     <link rel="shortcut icon" href="https://polisci.wustl.edu/sites/polisci.wustl.edu/themes/sitetheme/favicon.ico"
18     type="image/vnd.microsoft.icon" />
19     <!--[if lt IE 9]><script type="text/javascript" src="/sites/all/themes/innovate/scripts/html5.js?7p7rays"></script><![endif]--><meta
20     name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0" />
21     <meta name="Generator" content="Drupal 7 (http://drupal.org)" />
22     <link rel="canonical" href="/faculty" />
23     <link rel="shortlink" href="/node/4" />
24     <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1" />
25     <title>Current Faculty | Department of Political Science</title>
26     <link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_xE-rWrJf-fncB6tZ7fd2huxqgxu4W0-
27     qvma6Xer30m4.css" media="all" />
28     <link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_OOI-hABmd-SMR1h8-obE-
29     0sOAEbXpLCXmeWHEFYrA.css" media="all" />
30     <link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_OAaTCSbhUuU8GVBLD1vH2OHT_tgz3N-
31     FO_QgXjxv4.css" media="all" />
32     <link type="text/css" rel="stylesheet"
33     href="https://polisci.wustl.edu/files/polisci/css/css_DWPYTaZ2P2gGao6omeTINDp1fzQowIJrrp1ASKETTOGk.css" media="all" />
34     <link type="text/css" rel="stylesheet"
35     href="https://polisci.wustl.edu/files/polisci/css/css_TcoAtVf15vJtJfJzFWkh9VXlCeW8r3ZfCRavLyghXuw.css" media="screen" />
36     <link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_OF-
37     ro9xGfLkLxUuFuAhYpuYCadI12_jbt22RGCO6WY.css" media="handheld, only screen and (max-width: 767px)" />
38     <link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_NbnqFpRdg8MhD-MbJZmXr3i7Cl2-
39     RkRSBc41m43WfDY.css" media="handheld, only screen and (max-device-width: 480px)" />
40     <link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_Tuq8NnOU2u2oOjPX8L8IDKpgjiCBdaR-
41     COOYw_xpFMY.css" media="print" />
42
43     <!--[if lte IE 8]>
44     <link type="text/css" rel="stylesheet"
45     href="https://polisci.wustl.edu/files/polisci/css/css_KHVZdciTQ9NLR5qC9yFi34NDUDVt77RAYvQyNmD6VP4.css" media="all" />
46     <![endif]-->
```

Scraping HTML

urllib2

- "crawler"; in the background
- navigate to url

BeautifulSoup

- download HTML
- parse it for info

Great approach when...

- Info is contained in HTML (older websites are great!)
- Webpage names have a pattern
- Example:
`http://www.presidency.ucsb.edu/press_briefings.php?
year=2018`

Not helpful when...

- Info comes from javascript call, or info is not in HTML for some other reason (fancier websites)
- Example:
<https://www.oyez.org/cases/2017/17-586>
- Your task is not structured (e.g., using a search engine)
- Example:
<http://www.spencerdailyreporter.com/>

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser
 - Simulates behavior of human (e.g., point, click, type)

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser
 - Simulates behavior of human (e.g., point, click, type)
 - Less likely to be tracked and blocked

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser
 - Simulates behavior of human (e.g., point, click, type)
 - Less likely to be tracked and blocked
 - Flexible to undefined tasks

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser
 - Simulates behavior of human (e.g., point, click, type)
 - Less likely to be tracked and blocked
 - Flexible to undefined tasks
 - Cons:

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser
 - Simulates behavior of human (e.g., point, click, type)
 - Less likely to be tracked and blocked
 - Flexible to undefined tasks
 - Cons:
 - slow

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser
 - Simulates behavior of human (e.g., point, click, type)
 - Less likely to be tracked and blocked
 - Flexible to undefined tasks
 - Cons:
 - slow
 - dependent on Internet connection

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
 - They are incredibly fast...
 - ... but also easier to detect and block
 - Plus, sometimes they aren't amenable to your task
- Selenium is a “remote driver” of your favorite browser
 - Simulates behavior of human (e.g., point, click, type)
 - Less likely to be tracked and blocked
 - Flexible to undefined tasks
 - Cons:
 - slow
 - dependent on Internet connection
 - start-up costs (time, patience) in getting it set up on your machine

Some tricks and advice to become a pro-scraper

- Always check terms of service first.

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task
- I prefer Google Chrome to view page source and inspect elements

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task
- I prefer Google Chrome to view page source and inspect elements
- Use time breaks to avoid being blocked

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task
- I prefer Google Chrome to view page source and inspect elements
- Use time breaks to avoid being blocked
- Expect your code to break—websites change :(

Today

1. How to read and write files
2. How to navigate to url & parse HTML – approach #1
3. Example of selenium – approach #2
4. Lab to practice approach #1