

Web Scraping

Columbia Data Science Society

Carlos Martin and Kevin Lin

Technique of extracting information from websites

Unstructured data → structured data

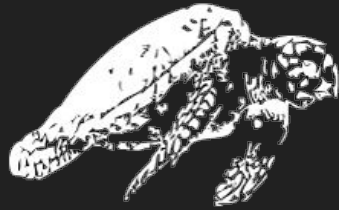
Why web scraping?

Not every site has an API

Website > API

No rate limits

Today



Requests



BeautifulSoup



Scrapy

Requests

Python library

HTTP for humans

Designed as improvement to standard **urllib2**



```
pip install requests
```

BeautifulSoup

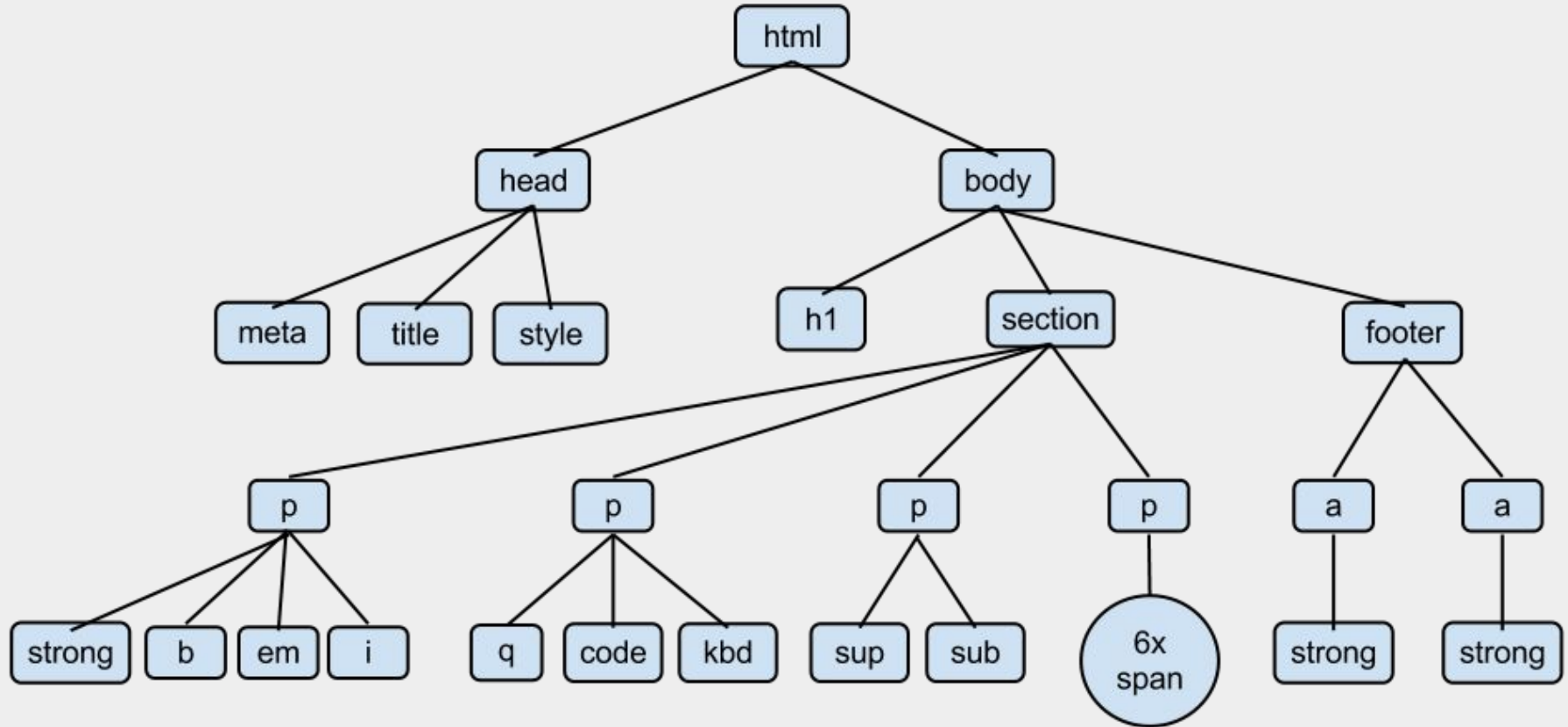
Pulling data out of HTML and XML

Navigate, search, modify the parse tree



```
pip install beautifulsoup4
```

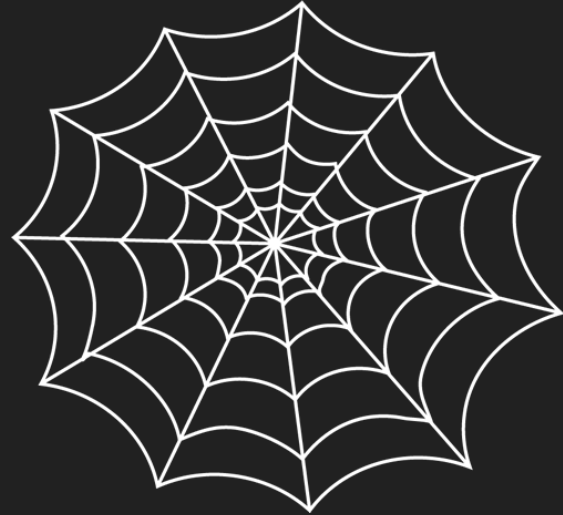
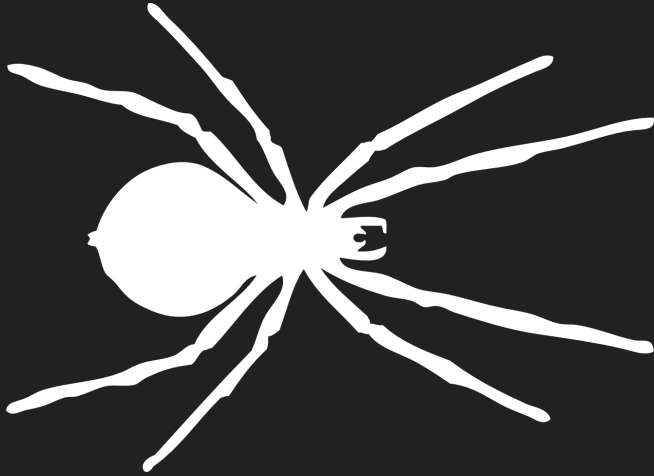

Document Object Model (DOM)



Web crawlers

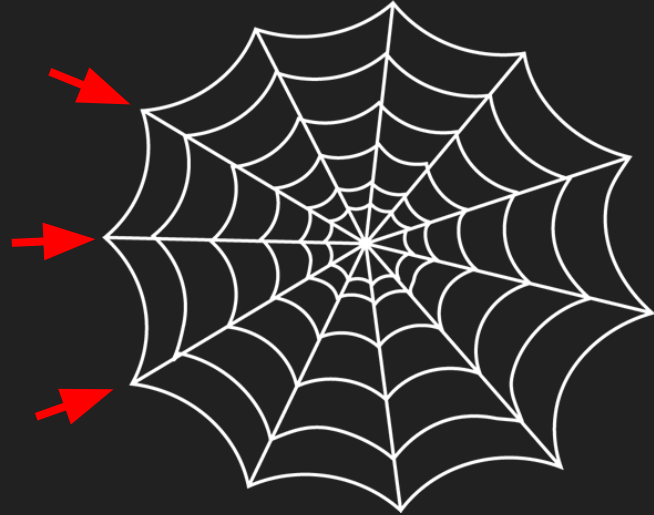
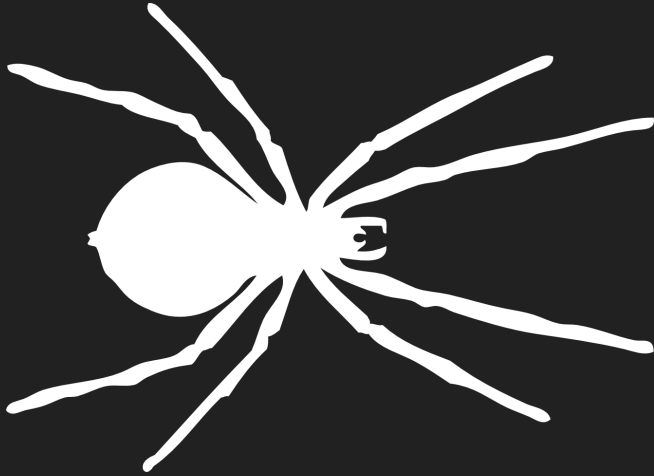
Web crawlers

a program that systematically browses the web



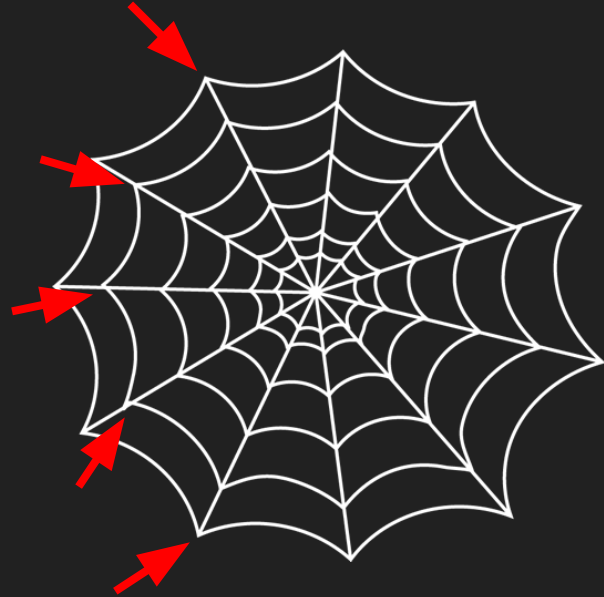
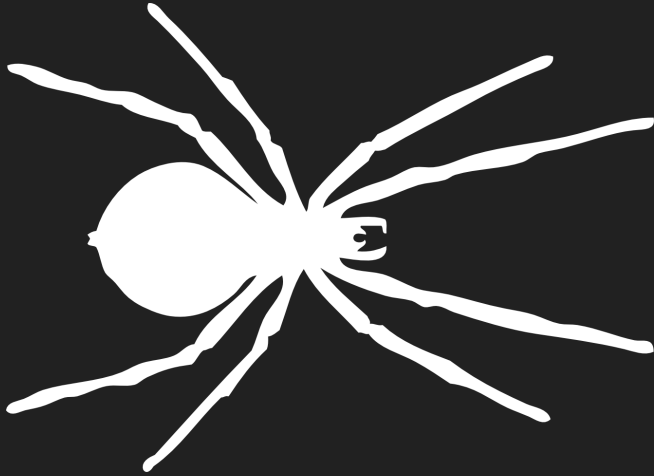
Web crawlers

starts with list of URLs to visit (seeds)



Web crawlers

identifies links, adds them to list of URLs to visit



Examples

Google

 **nutch**

 **bing**



INTERNET ARCHIVE


yaCy



DuckDuckGo

YAHOO!

Ask
.com

Uses

Scraping



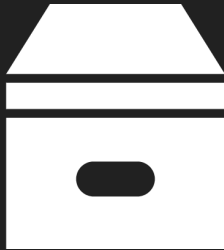
Indexing



Spambots



Archiving



Validation



Crawling policy



Selection policy

Revisit policy

Parallelization policy

Politeness policy

Robots.txt

Robot exclusion standard

Communicates with web crawlers and web robots

Lists areas that should not be processed or scanned

Can include crawl delay (i.e. requests per second) for throttling

See **robotstxt.org/robotstxt.html**

Robots.txt

User-agent: *

Disallow: /cgi-bin/

Disallow: /acis/whatsnew.html

Disallow: /httpd/reports/

Disallow: /itc/ccnmtl/assets/

Sitemap

Robot inclusion standard

XML file that gives URLs available for crawling

Can include last update date, update frequency, page importance, etc.

Allows web crawlers to crawl the site more intelligently

See **sitemaps.org/protocol.html**

Scrapy

web scraping framework for Python

```
pip install scrapy
```

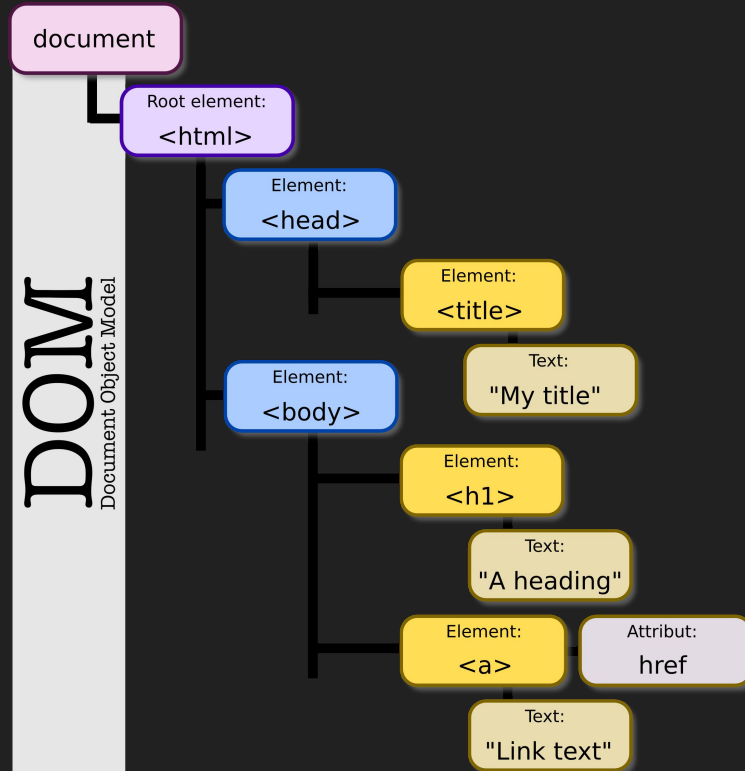
Creating a Scrapy project

```
scrapy startproject myproject
```

myproject

```
|— scrapy.cfg
|— myproject
    |— __init__.py
    |— items.py
    |— pipelines.py
    |— settings.py
    |— spiders
        |— __init__.py
```

Structure of a document



XPath

Language for **selecting nodes** in XML documents

```
/html/head/title
```

```
/html/head/title/text()
```

```
//td
```

```
//div[@class="myclass"]
```


XPath

Language for **selecting nodes** in XML documents

```
/wikimedia/projects/project/editions/*[2]
```

```
<?xml version="1.0" encoding="utf-8"?>
<wikimedia>
  <projects>
    <project name="Wikipedia" launch="2001-01-05">
      <editions>
        <edition language="English">en.wikipedia.org</edition>
        <edition language="German">de.wikipedia.org</edition>
        <edition language="French">fr.wikipedia.org</edition>
        <edition language="Polish">pl.wikipedia.org</edition>
      </editions>
    </project>
    <project name="Wiktionary" launch="2002-12-12">
      <editions>
        <edition language="English">en.wiktionary.org</edition>
        <edition language="French">fr.wiktionary.org</edition>
        <edition language="Vietnamese">vi.wiktionary.org</edition>
        <edition language="Trukish">tr.wiktionary.org</edition>
      </editions>
    </project>
  </projects>
</wikimedia>
```

CSS Selectors

Selects nodes based on their CSS style

Each stylesheet rule has a **selector pattern** that matches a set of HTML elements

```
p{  
  color:#0000FF;  
}
```

```
<h1 class="class1">  
  Heading  
</h1>  
<p id="id1">  
  Paragraph  
</p>  
<p class="class1">  
  Paragraph  
</p>
```

```
.class1{  
  color:#0000FF;  
}
```

```
<h1 class="class1">  
  Heading  
</h1>  
<p id="id1">  
  Paragraph  
</p>  
<p class="class1">  
  Paragraph  
</p>
```

```
#id1{  
  color:#0000FF;  
}
```

```
<h1 class="class1">  
  Heading  
</h1>  
<p id="id1">  
  Paragraph  
</p>  
<p class="class1">  
  Paragraph  
</p>
```

```
p.class1{  
  color:#0000FF;  
}
```

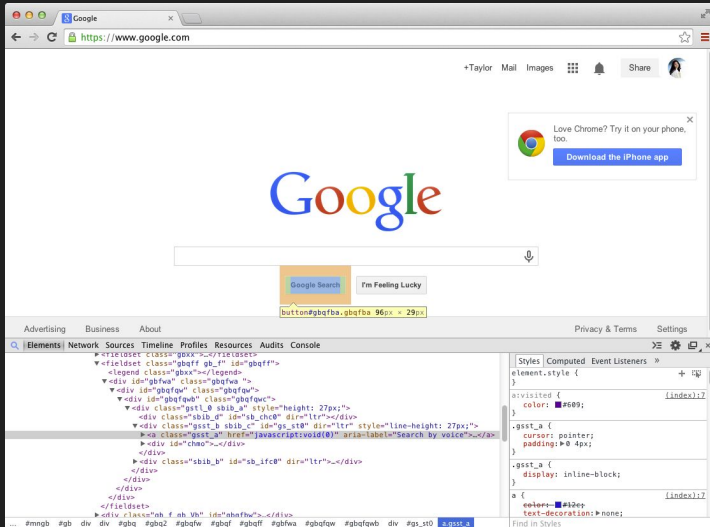
```
<h1 class="class1">  
  Heading  
</h1>  
<p id="id1">  
  Paragraph  
</p>  
<p class="class1">  
  Paragraph  
</p>
```

Obtaining the XPath



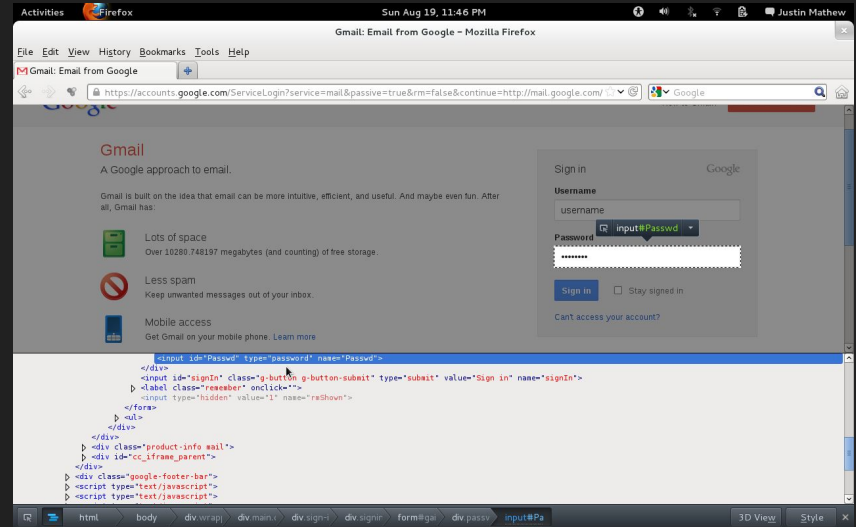
Chrome Developer Tools

Inspect
Copy > Copy XPath



Firebug HTML Panel

Inspect Element with Firebug
Copy XPath



Using XPath selectors

The screenshot shows a web browser with a right-click context menu open over a link titled "Junior Phi Beta Kappa Inductees Announced". The menu options include: "Open Link in New Tab", "Open Link in New Window", "Open Link in Incognito Window", "Save Link As...", "Copy Link Address", "Copy", "Search Google for 'Junior Phi Beta Kappa Inductees Announced'", "Print...", "Inspect", "Speech", "Search With Google", and "Add to iTunes as a Spoken Track". The page content includes a date "Jan 6", a time "15", and a list of names: Karen Bao, Katherine Christianson, Emily Corning, Bernhard Fasenfest, Adriano Fernandes, Vahe Galstyan, and Elena Goldstein.

The screenshot shows a web browser displaying an article titled "Junior Phi Beta Kappa Inductees Announced" by BWOG STAFF. The article text states: "Yesterday afternoon, Columbia College released the names of the 23 members of the Class of 2016 being inducted into Phi Beta Kappa for the Fall term. Of the 10% of the senior class who will be inducted into the national honor society this year, these 23 students make up just 2%; the other 8% will be inducted in the Spring term. The students were chosen by a faculty committee of PBK members 'based on the breadth, depth and rigor of their academic programs, as well as recommendations from faculty members who have worked with closely them.' Congratulations early inductees—you're the top 2%!"

The browser's developer tools are open, showing the "Elements" panel with the following HTML structure:

```
<div class="center-section">
  <div class="col-sm-1 col-md-1 col-xs-2 padding0"></div>
  <div class="col-sm-11 col-md-11 col-xs-10">
    <div class="blog-section">
      <a href="http://bwog.com/2016/01/06/junior-phi-beta-kappa-inductees-announced/" title="Junior Phi Beta Kappa Inductees Announced">Junior Phi Beta Kappa Inductees Announced</a>
    </div>
    <div class="written-by"></div>
    <div class="comment-time"></div>
    <div class="blog-text"></div>
  </div>
</div>
```

The "Styles" panel shows the following CSS rules for the selected element:

```
.blog-section {
  float: left;
  width: 100%;
  border-bottom: 1px solid #5d5d5d;
}
```

`//div[@class="blog-section"]`

Extracting comments

The screenshot displays a web interface with three comments and a browser's developer tools. The comments are:

- Sorry man 4 0 JANUARY 7, 2016 @ 7:53 PM REPLY TRACK
In the library... :{
But clearly not in the library Jacoby studies in :/
- CC2016 12 0 JANUARY 7, 2016 @ 10:36 AM REPLY TRACK
Whooo! Congrats everyone! :)
- I know 6 of them 24 0 JANUARY 7, 2016 @ 12:44 PM REPLY TRACK

The developer tools show the DOM structure for the third comment, which is highlighted in blue. The HTML structure is as follows:

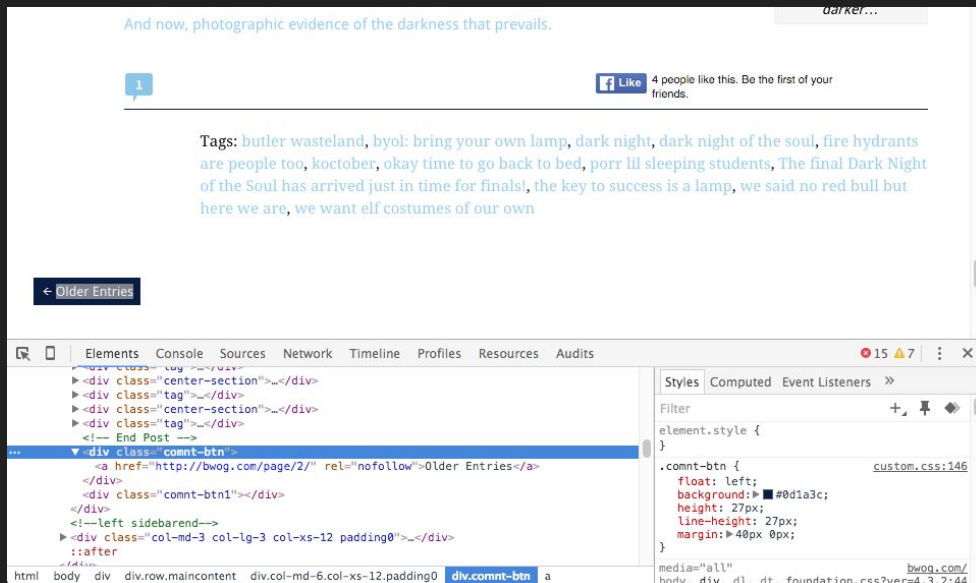
```
<div id="div-comment-1655853" class="comment odd alt thread-odd thread-alt depth-1 comment-body">  
  <div class="comment-author vcard"></div>  
  <div class="comment-meta datetime"></div>  
  <div class="comment-mod"></div>  
  <div class="reg-comment-body ">  
    <p>Whooo! Congrats everyone! :)</p>  
  </div>  
</div>  
</li>  
<!-- #comment-## -->  
<li class="comment even thread-even depth-1">  
  <div id="comment-1655854"></div>  
  <div class="comment-body">  
    <p></p>  
  </div>  
</li>  
</ul>
```

The Styles panel on the right shows the default styles for the selected element:

```
element.style {  
  }  
  
media="all" bwoq.com/  
.reg-comment-body p { style.css?ver=4.3.2:630  
  color: #4F4E4E;  
  font-size: 14px;  
  text-transform: none;  
}  
  
media="all" bwoq.com/  
.comment-body p { style.css?ver=4.3.2:375  
  margin: 5px;
```

```
//div[@class="reg-comment-body "]
```

Following links



//div[@class="comnt-btn"]//@href

Wait for it...

```
scrapy crawl bwog -o comments.json
```

...and on it goes!

Next steps

Scrapy Shell (interactive shell console)

Feed Exports (JSON, CSV, XML)

Item Pipelines (writing to MongoDB, duplicates filter)

doc.scrapy.org

for more information

Code from this tutorial

github.com/carlosgmartin/web-scraping