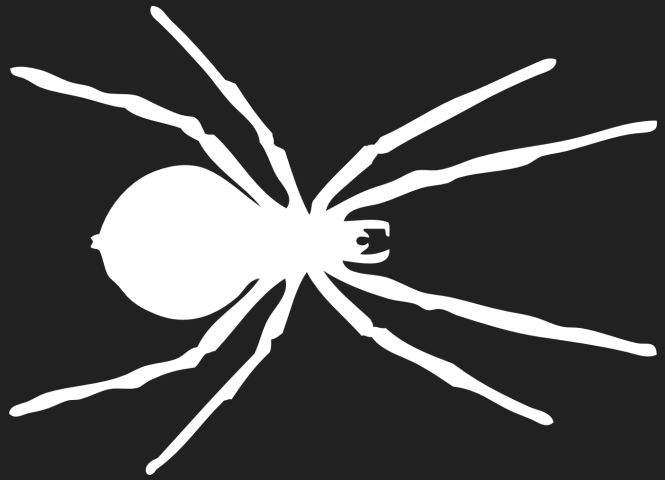
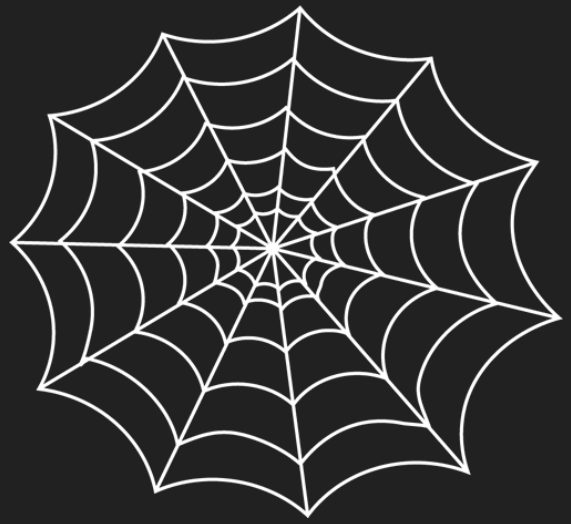
Web scraping

with Scrapy

Web crawler

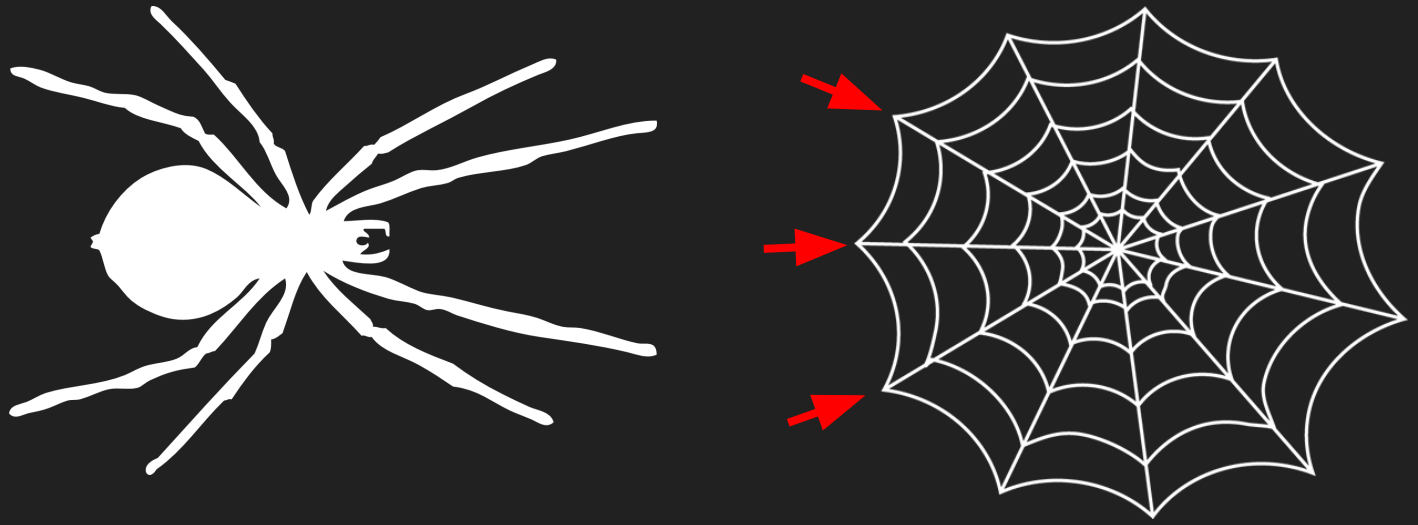
a program that systematically browses the web





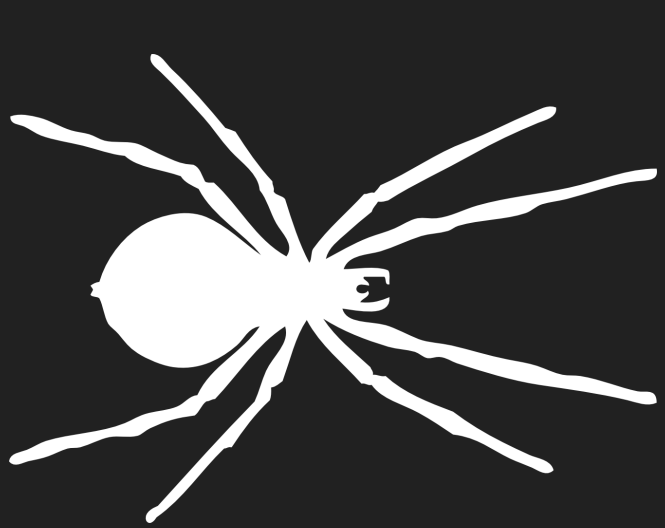
Web crawler

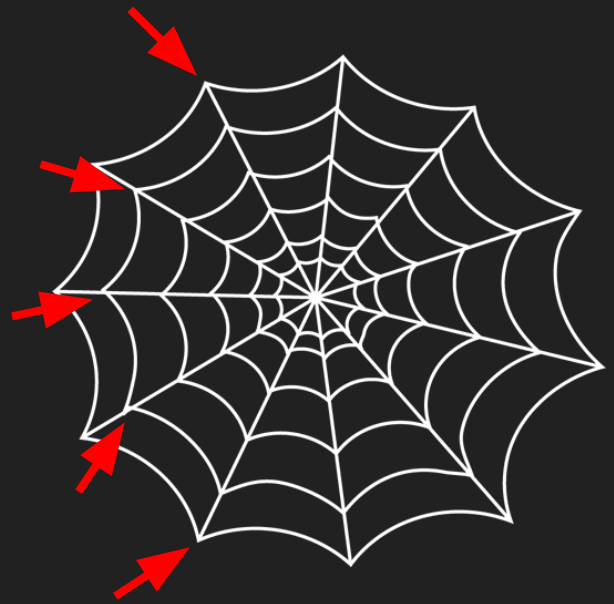
starts with list of URLs to visit (seeds)



Web crawler

identifies links, adds them to list of URLs to visit





Uses

Scraping

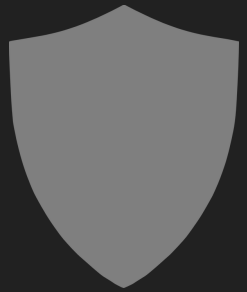
Archiving

Indexing

Validation

Spambots

Crawling policy and robots.txt



Selection policy

Revisit policy

Politeness policy

Parallelization policy

Examples

DuckDuckGo

Scrapy

web scraping framework for Python

```
pip install scrapy
```

Creating a project

```
scrapy startproject tutorial

tutorial
├── scrapy.cfg
└── tutorial
    ├── __init__.py
    ├── items.py
    ├── pipelines.py
    ├── settings.py
    └── spiders
        └── __init__.py
```

Creating a spider

```
cd tutorial/tutorial/spiders
touch bwog_spider.py
open bwog_spider.py
```

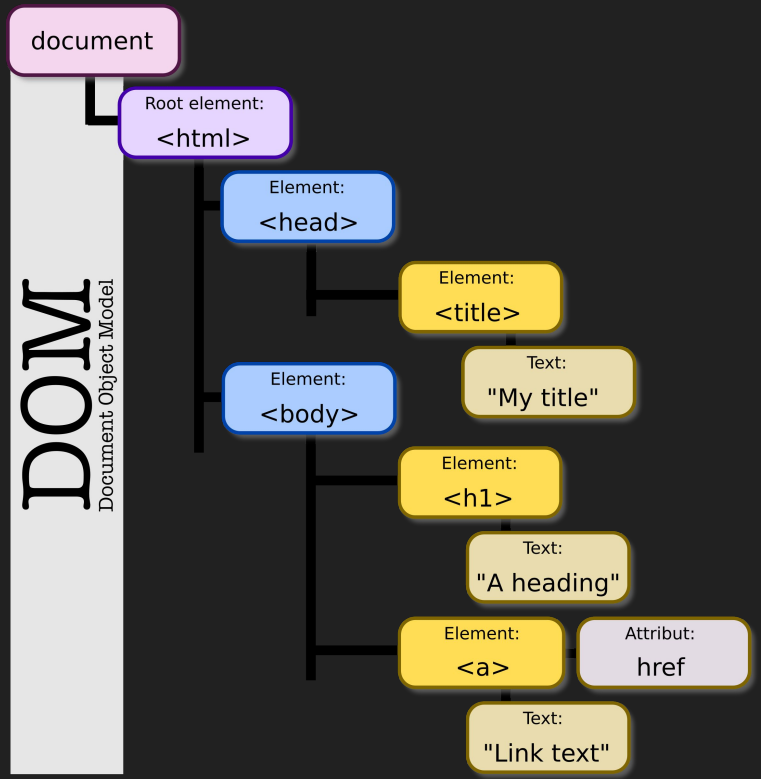
Creating a spider

```
import scrapy

class BwogSpider(scrapy.Spider):
    name = 'bwog'
    allowed_domains = ['bwog.com']
    start_urls = ['http://bwog.com/']

    def parse(self, response):
        print 'Testing ' + response.url
```

Running the spider

```
cd ../..
scrapy crawl bwog
```

Structure of a document



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>
```

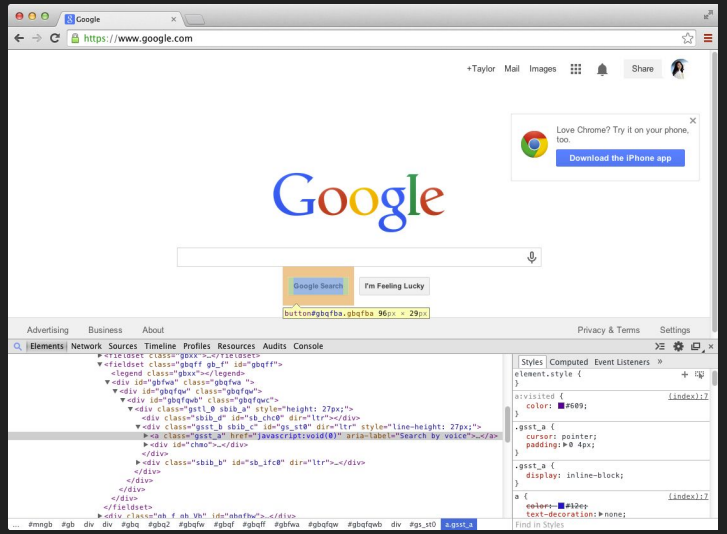
Obtaining the XPath

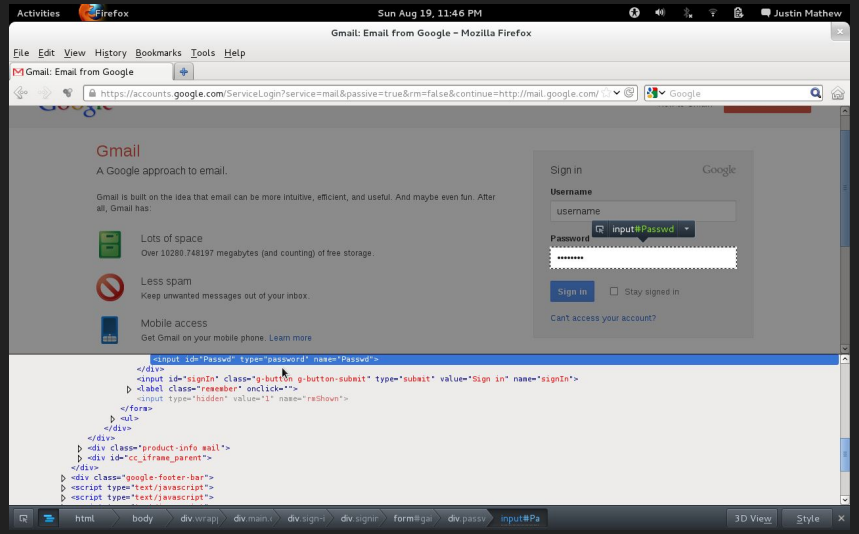
Chrome Developer Tools

Inspect
Copy > Copy XPath

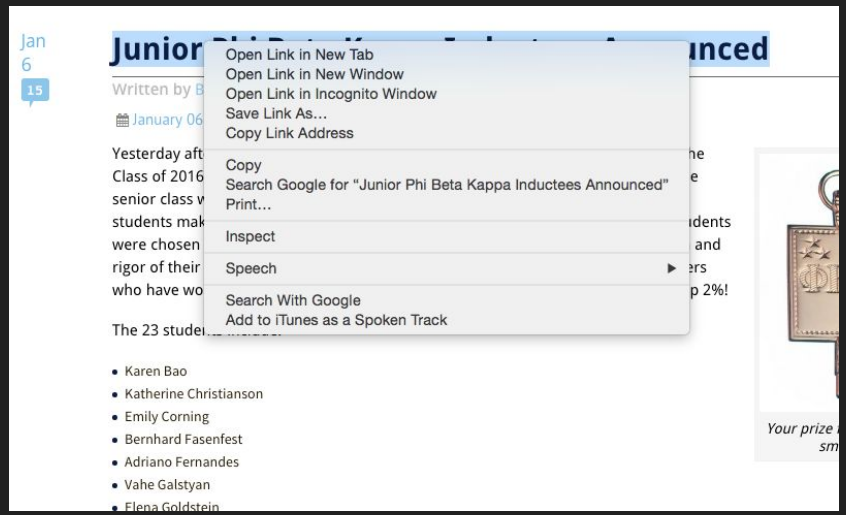
Firebug HTML Panel

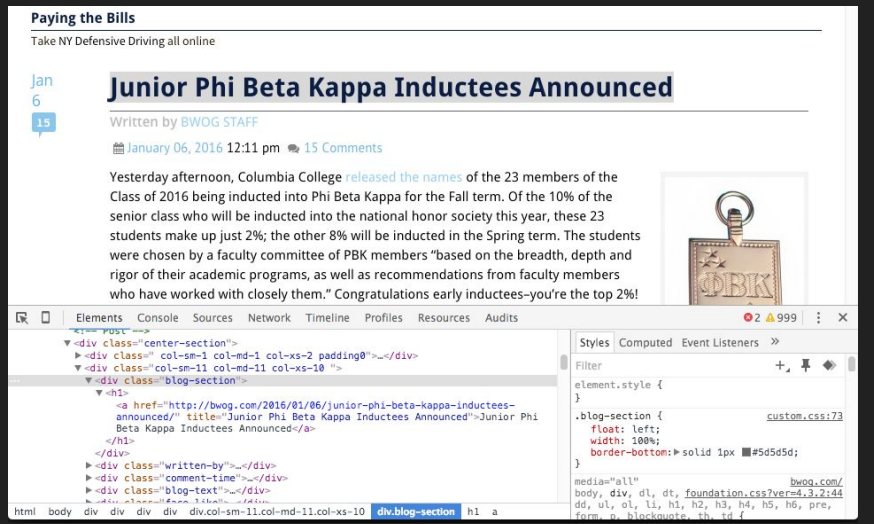
Inspect Element with Firebug
Copy XPath





Using XPath selectors






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

Using XPath selectors

```
import scrapy

class BwogSpider(scrapy.Spider):
    name = 'bwog'
    allowed_domains = ['bwog.com']
    start_urls = ['http://bwog.com/']

    def parse(self, response):
        for section in response.xpath('//div[@class="blog-section"]'):
            title = section.xpath('.//a/text()').extract()[0]
            print title
```

Running the spider

```
scrapy crawl bwog

Junior Phi Beta Kappa Inductees Announced
Until Next Year, Columbia
Overseen: Historical Revisionism On The Columbia Facebook Page
Bwog’s Resolutions For The New Year
Bwog In Bed: New Level Edition
Actual Wisdom: Philip Protter
Bwog Poetry
Bunsen Bwog: If This Is The Apocalypse, At Least We’ll Find Love
Bwog In Bed: Is This The Real Life? Edition
Field Notes: Barely Hanging On Edition
Actual Wisdom: J.C. Salyer
Shit Our Notes Say
“Baking” With Bwog: The Many Uses of Oregano
Bwog In Bed: Be Your Own Hero Edition
Dark Night Of The (Final) Soul
```

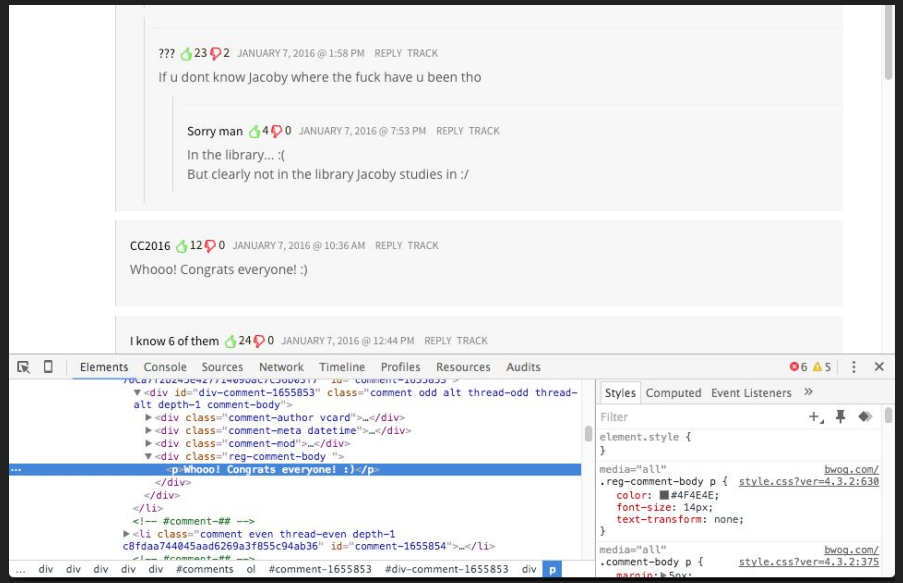
Scrapy Requests

```
import scrapy

class BwogSpider(scrapy.Spider):
    name = 'bwog'
    allowed_domains = ['bwog.com']
    start_urls = ['http://bwog.com/']

    def parse(self, response):
        for section in response.xpath('//div[@class="blog-section"]'):
            link = section.xpath('.//a/@href').extract()[0]
            yield scrapy.Request(link, callback=self.parse_entry)

    def parse_entry(self, response):
        print response.url
```

Running the spider

```
scrapy crawl bwog

http://bwog.com/2016/01/06/junior-phi-beta-kappa-inductees-announced/
http://bwog.com/2015/12/23/until-next-year-columbia/
http://bwog.com/2015/12/23/overseen-historical-revisionism-on-the-columbia-facebook-page/
http://bwog.com/2015/12/23/bwogs-resolutions-for-the-new-year/
http://bwog.com/2015/12/23/bwog-in-bed-new-level-edition/
http://bwog.com/2015/12/22/actual-wisdom-philip-protter/
http://bwog.com/2015/12/22/bwog-poetry/
http://bwog.com/2015/12/22/bunsen-bwog-if-this-is-the-apocalypse-at-least-well-find-love/
http://bwog.com/2015/12/22/bwog-in-bed-is-this-the-real-life-edition/
http://bwog.com/2015/12/21/field-notes-barely-hanging-on-edition/
http://bwog.com/2015/12/21/actual-wisdom-j-c-salyer/
http://bwog.com/2015/12/21/shit-our-notes-say/
http://bwog.com/2015/12/21/baking-with-bwog-the-many-uses-of-oregano/
http://bwog.com/2015/12/21/bwog-in-bed-be-your-own-hero-edition/
http://bwog.com/2015/12/20/dark-night-of-the-final-soul/
```

Extracting comments




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

Extracting comments

```
import scrapy

class BwogSpider(scrapy.Spider):
      name = 'bwog'
      allowed_domains = ['bwog.com']
      start_urls = ['http://bwog.com/']

      def parse(self, response):
            for section in response.xpath('//div[@class="blog-section"]'):
                  link = section.xpath('.//a/@href').extract()[0]
                  yield scrapy.Request(link, callback=self.parse_entry)

      def parse_entry(self, response):
            for comment in response.xpath('//div[@class="reg-comment-body "]'):
                  paragraphs = comment.xpath('.//text()').extract()
                  text = '\n'.join(paragraphs).strip()
                  print text
```

Scrapy Items

```
open tutorial/items.py
```

Scrapy Items

```
import scrapy

class TutorialItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    content = scrapy.Field()
```

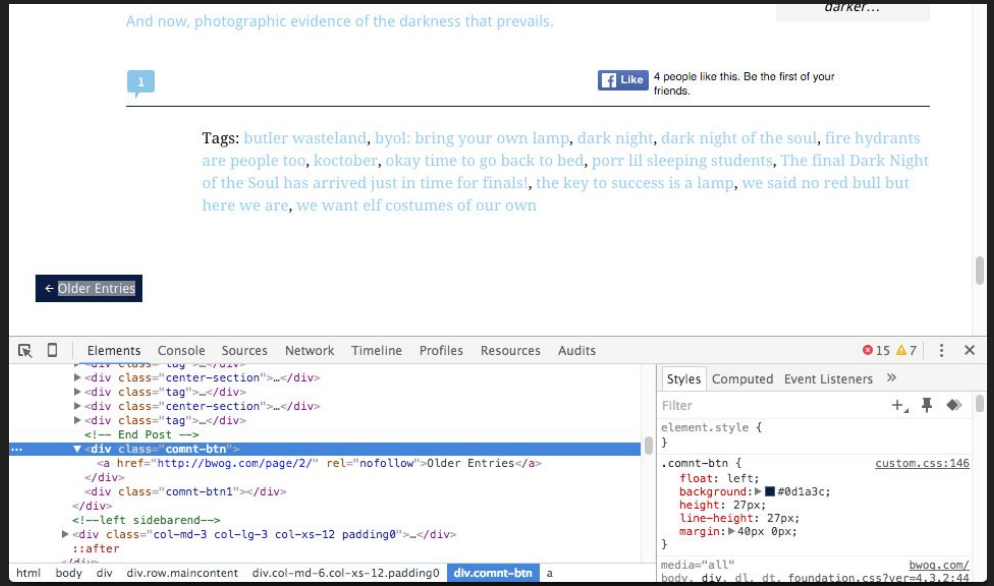
Extracting comments

```
import scrapy
from tutorial.items import TutorialItem

class BwogSpider(scrapy.Spider):
	name = 'bwog'
	allowed_domains = ['bwog.com']
	start_urls = ['http://bwog.com/']

	def parse(self, response):
		for section in response.xpath('//div[@class="blog-section"]'):
			link = section.xpath('.//a/@href').extract()[0]
			yield scrapy.Request(link, callback=self.parse_entry)

	def parse_entry(self, response):
		for comment in response.xpath('//div[@class="reg-comment-body "]'):
			paragraphs = comment.xpath('.//text()').extract()
			text = '\n'.join(paragraphs).strip()
			item = TutorialItem()
			item['content'] = text
			yield item
```

Storing comments

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

Following links




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

Following links

```
import scrapy
from tutorial.items import TutorialItem

class BwogSpider(scrapy.Spider):
	name = 'bwog'
	allowed_domains = ['bwog.com']
	start_urls = ['http://bwog.com/']

	def parse(self, response):
		for section in response.xpath('//div[@class="blog-section"]'):
			link = section.xpath('.//a/@href').extract()[0]
			yield scrapy.Request(link, callback=self.parse_entry)
		next = response.xpath('//div[@class="comnt-btn"]//@href').extract()[0]
		yield scrapy.Request(next, callback=self.parse)

	def parse_entry(self, response):
		for comment in response.xpath('//div[@class="reg-comment-body "]'):
			paragraphs = comment.xpath('.//text()').extract()
			text = '\n'.join(paragraphs).strip()
			item = TutorialItem()
			item['content'] = text
			yield item
```

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