How to Scrape the Web

With Scrapy

@aaqaishtyaq

Who Am I?

- Aaqa Ishtyaq
- Back End Developer (Intern) *@BrandMyICO*
- Software Developer *@HostingJunction*

Crawling the Web

Not so hard, right?

```
$ curl  https://www.cnn.com/
```

Crawling with Python

```python
#!/usr/bin/env python
import requests

r = requests.get('http://www.cnn.com/')
print r.content
```

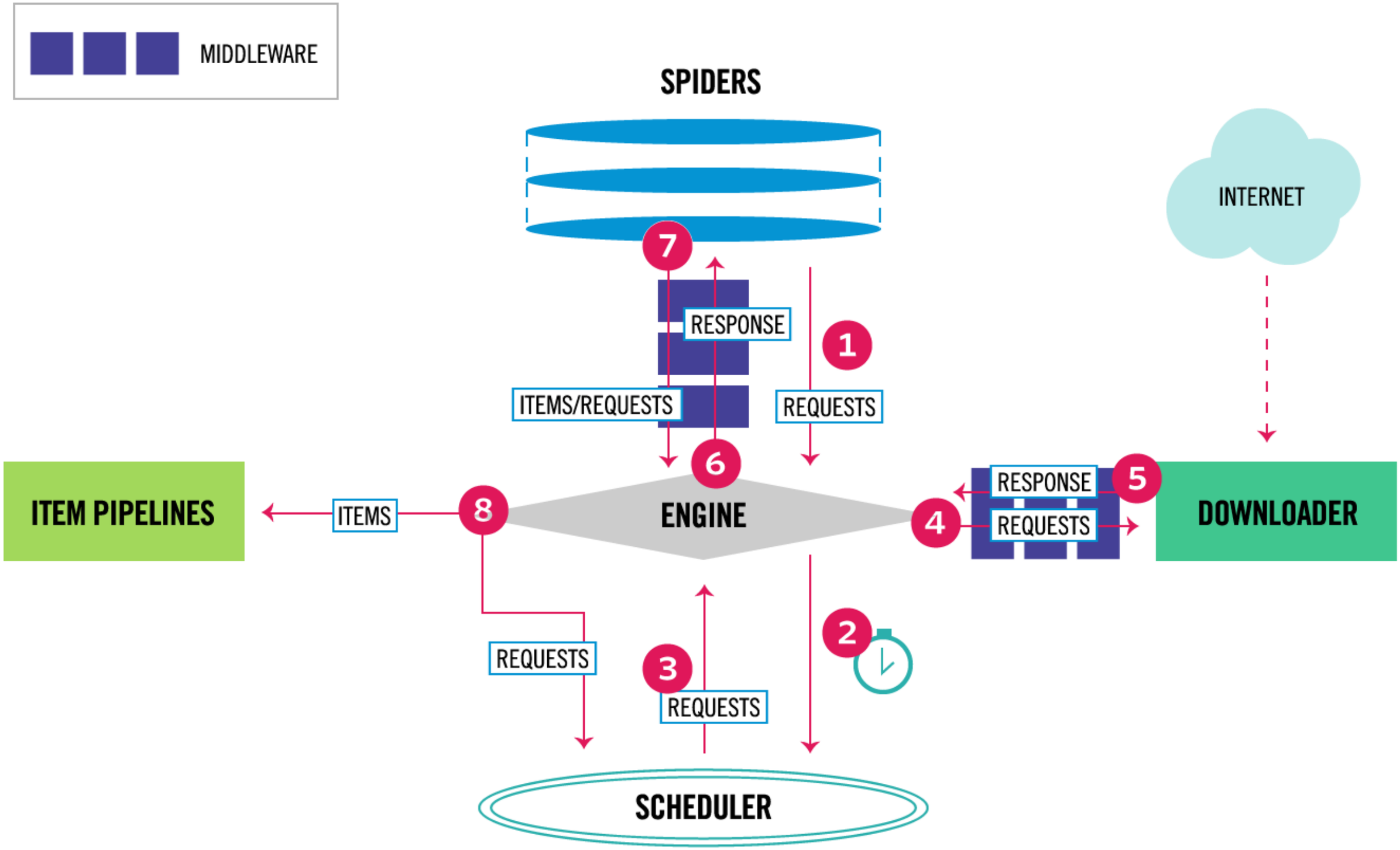
Then what?

Web Scraping framework

Scrapy

What is Scrapy?

- Framework for building web crawlers
- Extracts structured data from unstructured web pages
- Inspired by Django
- Enables developers to focus on the rules to extract the data they want
- Does the hard parts of crawling
- Fast event-driven code based on Twisted


MIDDLEWARE
SPIDERS
INTERNET
7
RESPONSE
1
ITEMS/REQUESTS
REQUESTS
6
ITEM PIPELINES
ITEMS
8
ENGINE
RESPONSE
5
4
REQUESTS
DOWNLOADER
REQUESTS
3
REQUESTS
2
SCHEDULER

Features and benefits

- Selecting and extracting data from HTML
- Reusable filters for cleaning and sanitizing
- Export formats: JSON, CSV, XML
- Automatic image downloading
- Pluggable extension API
- Text encoding auto-detection
- Template system from creating spiders
- Statistics and spider performance monitoring

Middleware

- Cookies and sessions
- HTTP compression
- HTTP authentication
- HTTP cache
- User-agent spoofing
- robots.txt handling
- Crawl depth control

Alternatives of Scrapy

Beautiful Soup

Lxml

Newspaper3k

What Scrapy is not

- Scrapy is *not* Apache Nutch
- Scrapy is *not* Apache Solr, Elasticsearch, or Lucene
- Scrapy is *not* a database like MySQL, MongoDB or Redis

HTML, the Dom tree representation, and the XPath

- A URL is typed on the browser.
- The server replies by sending an HTML page to browser, or XML or JSON.
- The HTML get translated to an internal tree representation inside the browser: the infamous Document Object Model(DOM)
- The internal representation is rendered, based on some layout rules, to visual representation you see on your screen.

Selecting HTML element with XPath

- $x('/html')
- $x('/html/body')
- $x('/html/body/div/h1')
- $x('//h1')
- $x('//div/p[1]')

Example of some common XPath types

```html
<body>
    <h1 id="firstheading">
        <span>Text 1</span>
    </h1>
    <h1 id="secondheading">
        <span>Text 2</span>
    </h1>
</body>
```

```
//h1[@id='firstheading']/span/text()
```

```html
<div class="tic">
    <ul>
        <li>
            <a href="pyladies-delhi.github.io">Pyladies</a>
        </li>
    </ul>
</div>
<div class="toc">
    <p>
        LinuxChix India
    </p>
</div>
```

```
//div[@class='tic']/ul/li/a/@href

//div[@class='tic']/ul//a/@href
```

```html
<table class='infobox'>
    <tr>
        <td>
            <tr>
                <img src="https://oo.ps/1.png">OOPS</tr>
                <img src="https://oo.ps/2.png">OOOOOPS</tr>
        </td>
        <td>
            <tr>
                <img src="https://oo.ps/2.png">OOOOOPS</tr>
                <img src="https://oo.ps/1.png">OOPS</tr>
        </td>
</table>
```

```
//table[@class='infobox']//img[1]/@src
```

```html
<div class="ref">
    <p>
        <a href="https://exapmle.com">Example</a>
    </p>
</div>
<div class="list">
    <h1>
        <a href="https://exapmle.com">Example</a>
    </h1>
</div>
<div class="reflist">
    <h1>
        <a href="https://exapmle.com">Example</a>
    </h1>
    <p>
        <a href="https://duckduckgo.com">Duck</a>
    </p>
</div>
```

```
//div[starts-with(@class,”reflist”)]//a/@href
```

Get Started with Scrapy

```
$ virtualenv venv

$ source venv/bin/activate

$ pip3 install scrapy

$ scrapy createproject pyladies

$ cd pyladies

$ scrapy genspider imdb www.imdb.com
```

Project Layout

```
pyladies/
    scrapy.cfg
    pyladies/
        __init__.py
        items.py
        pipelines.py
        settings.py
        spiders/
            __init__.py
```

Define what to scrape

```python
import scrapy

class PyladiesItem(scrapy.Item):
    # define the fields for your item here like:
    name = scrapy.Field()
    desc = scrapy.Field()
    categories = scrapy.Field()
    telegram = scrapy.Field()
    website = scrapy.Field()
    medium = scrapy.Field()
```

```python
import scrapy


class QuotesSpider(scrapy.Spider):
    name = "quotes"
    start_urls = [
        'http://quotes.toscrape.com/page/1/',
    ]

    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'text': quote.css('span.text::text').extract_first(),
                'author': quote.css('small.author::text').extract_first(),
                'tags': quote.css('div.tags a.tag::text').extract(),
            }

        next_page = response.css('li.next a::attr(href)').extract_first()
        if next_page is not None:
            next_page = response.urljoin(next_page)
            yield scrapy.Request(next_page, callback=self.parse)
```

yield?

```
next_page = []
for url in ...
    next_page.append(scrapy.Request( ... ))
for url in ...
    next_page.append(scrapy.Request( ... ))
return next_page
```

What about pagination?

APIs?

Please Login

First… 🙏

Dynamic Web Pages…

JS… 🤔

Being a good citizen in a world full of spiders

Irresponsible web scraping can be annoying and even illegal in some cases.

The two most important things to avoid are:

1) denial-of-service(DoS) attack like behaviour

2) Violating copyright

Thank You!

aaqa@hostingjunction.in

@aaqaishtyaq