

py4kids (<https://github.com/wgong/py4kids>)

## Scrapy- Crawl & Scrape the Web

In this lesson we learn how to extract data from the web using powerful scrapy

```
In [1]: from jyquickhelper import add_notebook_menu
        add_notebook_menu()
```

- Out[1]:
- Scrapy
    - Install
    - Create a project
    - Customize spider
    - Launch spider
    - Extract data
    - Follow next link
  - References
  - Learn to learn

**Scrapy (<https://scrapy.org/>)**



### Overview

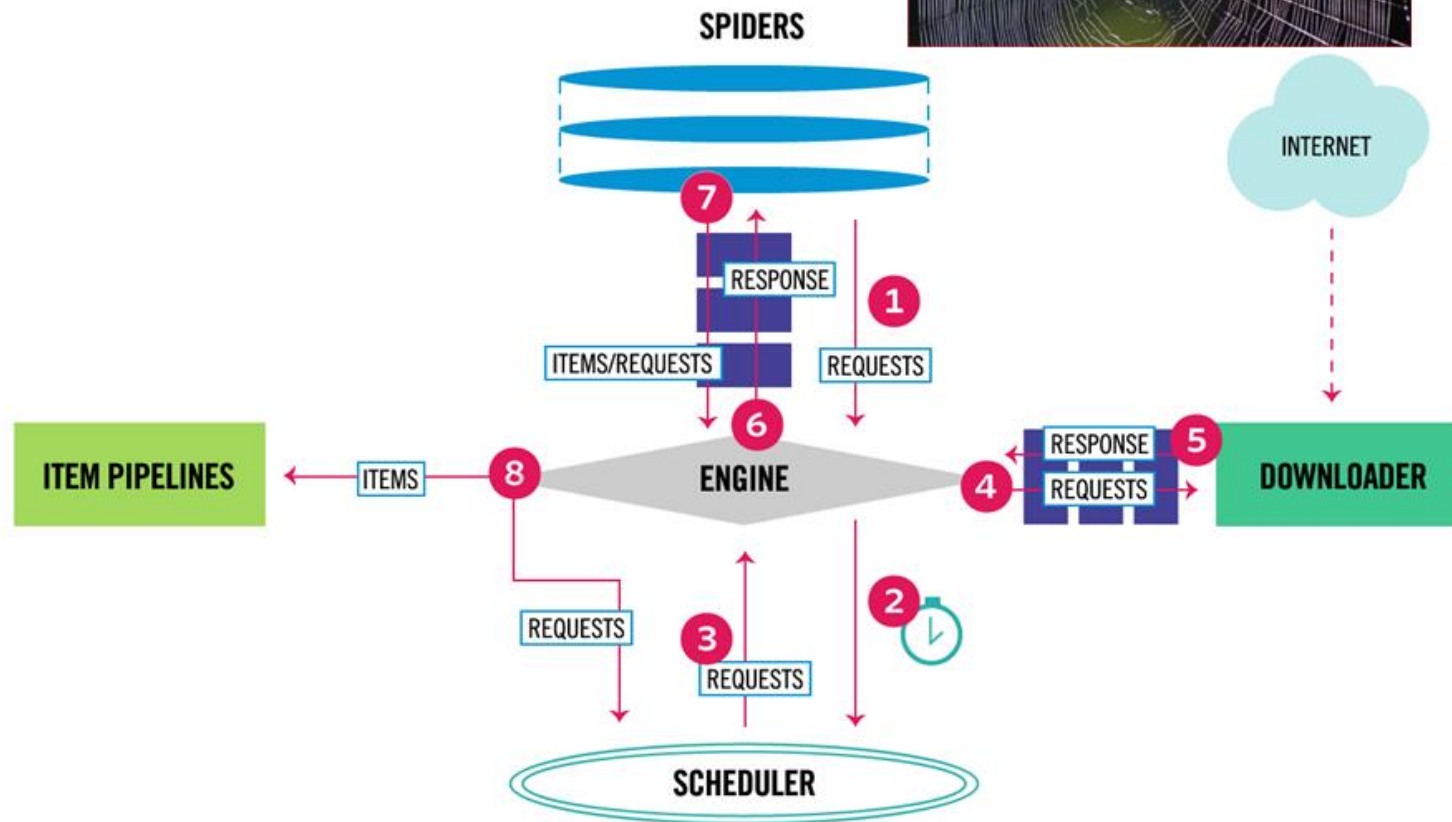
An open source and collaborative framework for extracting the data you need from websites in a fast, simple, yet extensible way.

# Scrapy advantages

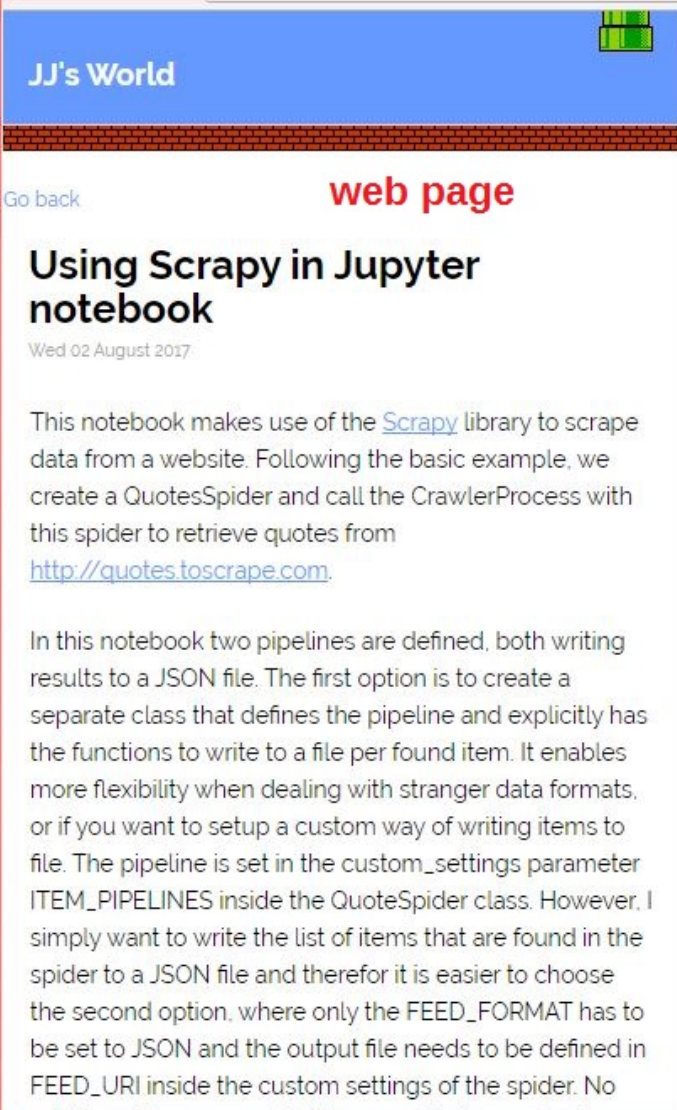
- ▶ **Faster than mechanize** because it uses asynchronous operations (Twisted).
- Scrapy has **better support for html parsing**.
- Scrapy has **better support for unicode characters, redirections, gzipped responses, encodings**.
- You can export the **extracted data directly to JSON,XML and CSV**.

How does it work

# Scrapy Architecture



DOM tree in browser



**web page**

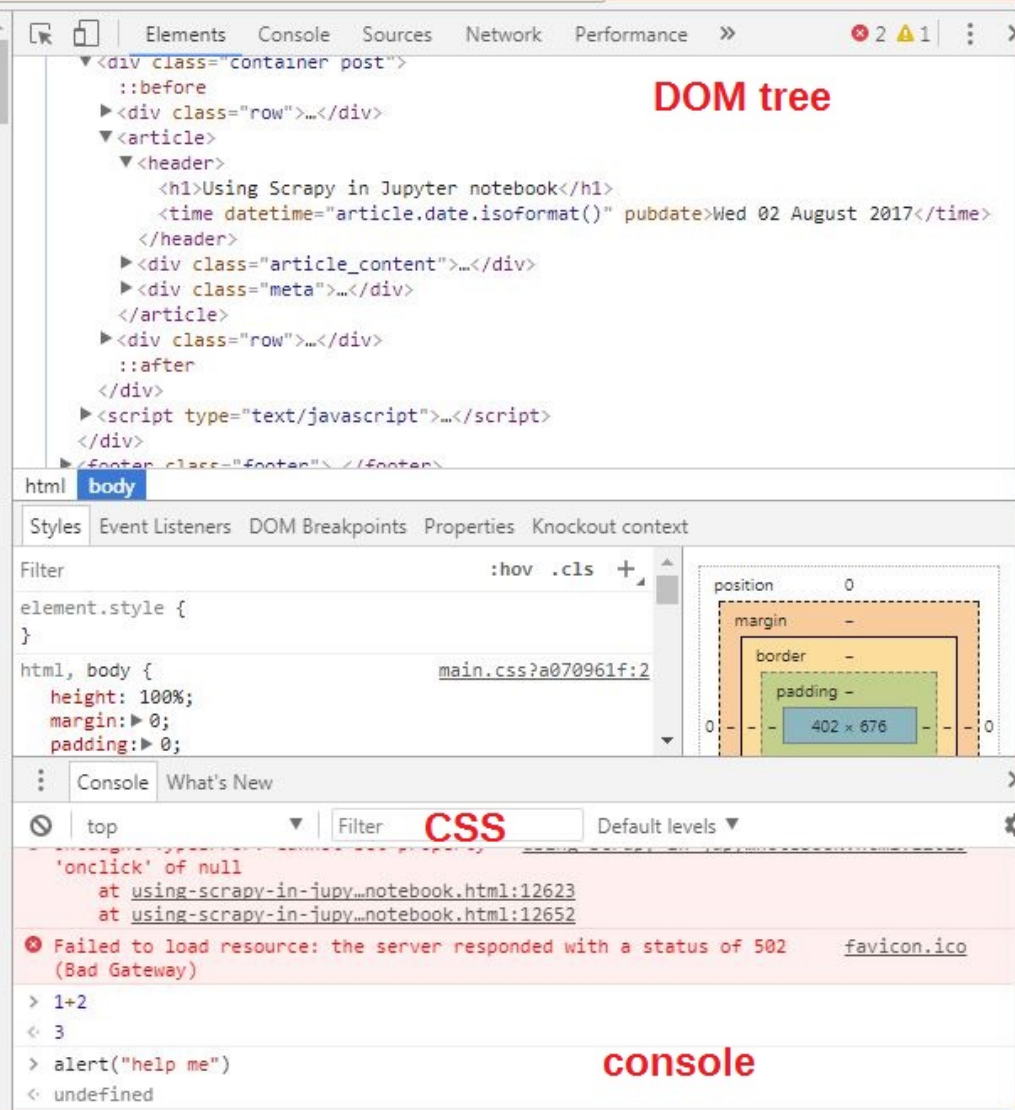
Go back

## Using Scrapy in Jupyter notebook

Wed 02 August 2017

This notebook makes use of the [Scrapy](#) library to scrape data from a website. Following the basic example, we create a QuotesSpider and call the CrawlerProcess with this spider to retrieve quotes from <http://quotes.toscrape.com>.

In this notebook two pipelines are defined, both writing results to a JSON file. The first option is to create a separate class that defines the pipeline and explicitly has the functions to write to a file per found item. It enables more flexibility when dealing with stranger data formats, or if you want to setup a custom way of writing items to file. The pipeline is set in the custom\_settings parameter ITEM\_PIPELINES inside the QuoteSpider class. However, I simply want to write the list of items that are found in the spider to a JSON file and therefore it is easier to choose the second option, where only the FEED\_FORMAT has to be set to JSON and the output file needs to be defined in FEED\_URI inside the custom settings of the spider. No additional classes or definitions need to be created.



**DOM tree**

```

<div class="container post">
  ::before
  <div class="row">...</div>
  <article>
    <header>
      <h1>Using Scrapy in Jupyter notebook</h1>
      <time datetime="article.date.isoformat()" pubdate>Wed 02 August 2017</time>
    </header>
    <div class="article_content">...</div>
    <div class="meta">...</div>
  </article>
  <div class="row">...</div>
  ::after
</div>
<script type="text/javascript">...</script>
</div>
<footer class="footer">...</footer>

```

**body**

```

element.style {
}
html, body {
  height: 100%;
  margin: 0;
  padding: 0;
}

```

**console**

```

'onclick' of null
  at using-scrapy-in-jupy...notebook.html:12623
  at using-scrapy-in-jupy...notebook.html:12652
Failed to load resource: the server responded with a status of 502 (Bad Gateway)
  favicon.ico
> 1+2
< 3
> alert("help me")
< undefined

```

## Install

```
cmd> conda install scrapy
```

```
cmd> pip install scrapy
```

## Create a project

- open a terminal window,
- create a working folder,
- type

```
cmd> scrapy startproject tutorial
```

### **\$ scrapy startproject <project\_name>**

scrapy.cfg: the project configuration file.

tutorial/:the project's python module.

items.py: the project's items file.

pipelines.py : the project's pipelines file.

setting.py : the project's setting file.

spiders/ : spiders directory.

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

## Customize spider

- specify download web addresses (URL)
- define parser logic



```
quotes_spider.py x
1  import scrapy
2
3  class QuotesSpider(scrapy.Spider):
4      name = "quotes"
5
6      start_urls = [
7          'http://quotes.toscrape.com/page/1/',
8          'http://quotes.toscrape.com/page/2/',
9      ]
10
11     def parse(self, response):
12         page_number = response.url.split("/")[-2]
13         filename = 'quotes-%s.html' % page_number
14         with open(filename, 'wb') as f:
15             f.write(response.body)
```

## Launch spider

CMD> scrapy crawl quotes

Two files will be created in the current folder:

- quotes-1.html:
  - web copy (<http://quotes.toscrape.com/page/1/>)
  - local copy ([./scrapy-tutorial/quotes-1.html](#))
- quotes-2.html
  - web copy (<http://quotes.toscrape.com/page/2/>)
  - local copy ([./scrapy-tutorial/quotes-2.html](#))

## Extract data

```
CMD> scrapy crawl quotes
```

Two files will be created in the current folder:

- quotes-1.html:
  - [web copy \(http://quotes.toscrape.com/page/1/\)](http://quotes.toscrape.com/page/1/)
  - [local copy \(./scrapy-tutorial/quotes-1.html\)](http://localhost:8888/notebooks/lesson-11-scrapy/lesson-11.ipynb#References)
- quotes-2.html
  - [web copy \(http://quotes.toscrape.com/page/2/\)](http://quotes.toscrape.com/page/2/)
  - [local copy \(./scrapy-tutorial/quotes-2.html\)](http://localhost:8888/notebooks/lesson-11-scrapy/lesson-11.ipynb#References)

Extract data and save them into CSV file:

```
11 def parse(self, response):
12     page_number = response.url.split("/")[-2]
13     filename = 'quotes-%s.html' % page_number
14     with open(filename, 'wb') as f:
15         f.write(response.body)
16
17     # extract quote, author, tags and store data in csv file
18     import csv
19     file_csv = 'quotes-%s.csv' % page_number
20     with open(file_csv, 'w') as f_csv:
21         csv_writer = csv.writer(f_csv, lineterminator='\n', quoting=csv.QUOTE_ALL)
22         for quote in response.css('div.quote'):
23             txt = quote.css('span.text::text').extract_first()
24             author = quote.css('small.author::text').extract_first()
25             tags = quote.css('div.tags a.tag::text').extract()
26             tags_str = " : ".join(tags)
27             csv_writer.writerow([author,txt,tags_str])
28
```

## Follow next link

To download all the quotes, follow "Next Page" line recursively:

```
28
29     # follow the link
30     next_page = response.css('li.next a::attr(href)').extract_first()
31     if next_page is not None:
32         yield response.follow(next_page, callback=self.parse)
```

## References

- [Basic Tutorial \(https://doc.scrapy.org/en/latest/intro/tutorial.html\)](https://doc.scrapy.org/en/latest/intro/tutorial.html)
- [Using Scrapy in Jupyter notebook \(http://www.jitsejan.nl/using-scrapy-in-jupyter-notebook.html\)](http://www.jitsejan.nl/using-scrapy-in-jupyter-notebook.html)
- [Build your 1st web crawler \(https://medium.com/python-pandemonium/develop-your-first-web-crawler-in-python-scrapy-6b2ee4baf954\)](https://medium.com/python-pandemonium/develop-your-first-web-crawler-in-python-scrapy-6b2ee4baf954)
- [How To Crawl A Web Page with Scrapy and Python \(https://www.digitalocean.com/community/tutorials/how-to-crawl-a-web-page-with-scrapy-and-python-3\)](https://www.digitalocean.com/community/tutorials/how-to-crawl-a-web-page-with-scrapy-and-python-3)
- [Scrapy Tutorial: Web Scraping Craigslist \(http://python.gotrained.com/scrapy-tutorial-web-scraping-craigslist/\)](http://python.gotrained.com/scrapy-tutorial-web-scraping-craigslist/)
- [image crawlers \(https://pypi.python.org/pypi/icrawler\)](https://pypi.python.org/pypi/icrawler)

## Learn to learn

- [google \(https://www.google.com/\)](https://www.google.com/)
  - images
  - YouTube, videos
- [stackoverflow \(https://stackoverflow.com/\)](https://stackoverflow.com/)
- [quora \(https://www.quora.com/\)](https://www.quora.com/)
- [slideshare \(https://www.slideshare.net/\)](https://www.slideshare.net/)
- [learn anything \(https://learn-anything.xyz/programming/programming-languages/python\)](https://learn-anything.xyz/programming/programming-languages/python)

In [ ]: