Scrapy

Web Scraping Tool

Web Crawling vs. Scraping

- **Web Crawling**: the process of iteratively finding and fetching web links

- **Web Scraping**: the process of processing a web document and extracting information out of it

Source: stackoverflow.com

Use-Case

- Extract information from a particular website ("focused crawl")
- Save it in a structured, easy to use way
  in CSV, JSON or XML format

- Example:
  Extract properties from imot.bg
  Extract comments about car services from auto forums
  Extract product descriptions from Etsy

Why Scrapy

- Simple to setup for a simple use-case
- Written in Python
- Clear documentation
- Easy deploy to spider hosting

Installation

Install Python

```
pip install Scrapy
```

* Follow the installation instructions for the corresponding OS.

Install on Windows

1. Install python 2.7:
https://www.python.org/downloads/windows/

2. Install pywin32:
http://sourceforge.net/projects/pywin32/files/pywin32/Build%20219/

3. Install C++ Compiler for Python:
https://www.microsoft.com/en-us/download/details.aspx?id=44266

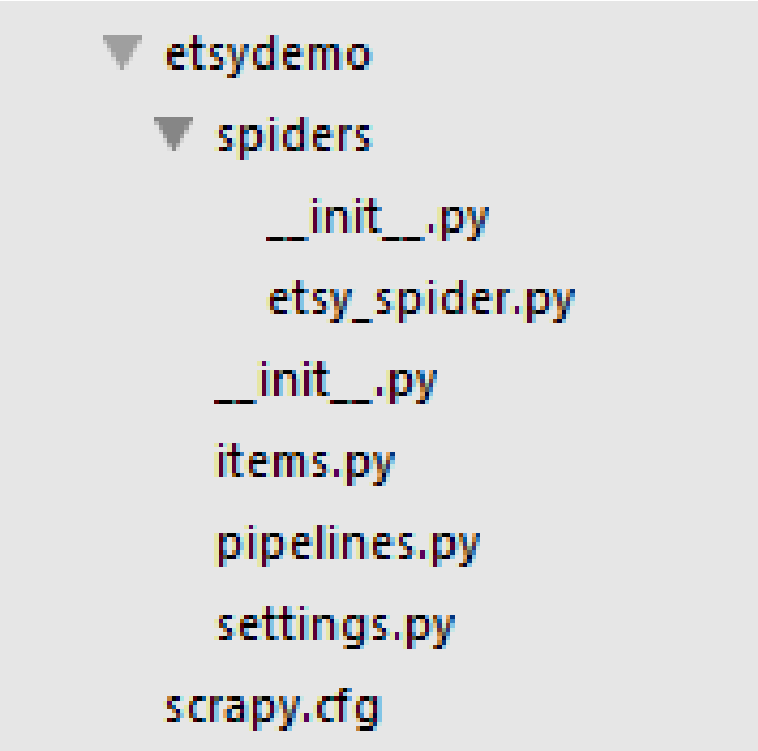
4. Install lxml:
https://pypi.python.org/pypi/lxml/3.5.0

5. Install Scrapy
pip install Scrapy

6. Install shub
pip install shub

Create Project

```
scrapy startproject etsydemo
```



Setup of Simple Project

1. Create Item
2. Create Spider

Items

items.py

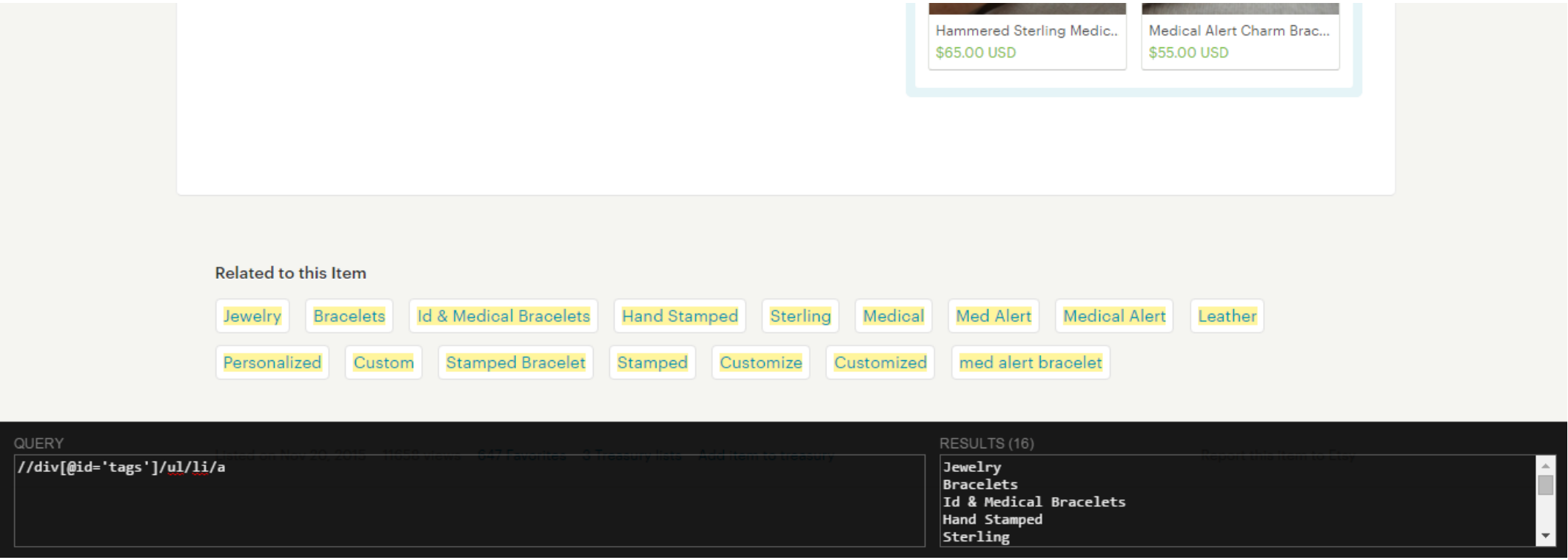
```
import scrapy

def serialize_description(value):
    return '\n'.join(value)

def serialize_favorites(value):
    return value[0].partition(' ')[0]

def serialize_extract_first_word(value):
    return value[0].strip().partition(' ')[0]

class EtsyItem(scrapy.Item):
    url = scrapy.Field()
    title = scrapy.Field()
    description = scrapy.Field(serializer=serialize_description)
    tags = scrapy.Field()
    price = scrapy.Field()
    rating = scrapy.Field()
    reviews = scrapy.Field()
    views = scrapy.Field(serializer=serialize_extract_first_word)
    favorites = scrapy.Field(serializer=serialize_extract_first_word)
    treasury_lists = scrapy.Field(serializer=serialize_extract_first_word)
```

Spiders

```python
import scrapy

from etsydemo.items import EtsyItem

class EtsySpider(scrapy.Spider):
    name = 'etsy'
    start_urls = ['https://www.etsy.com/c/accessories/belts-and-suspenders/belts']

    def parse(self, response):
        # Follow all product links and parse the items
        for href in response.xpath("//a/@href[contains(., '/listing/')]"):
            full_url = response.urljoin(href.extract())
            yield scrapy.Request(full_url, callback=self.parse_item)
        # Follow all category links to find items to parse
        for href in response.xpath("//a/@href[contains(., '/c/')]"):
            full_url = response.urljoin(href.extract())
            yield scrapy.Request(full_url, callback=self.parse)

    def parse_item(self, response):
        item = EtsyItem()
        item['title'] = response.xpath("//div[@id='listing-page-cart-inner']/h1/span/text()").extract()
        item['description'] = ' '.join(response.xpath("//div[@id='description-text']/text()").extract())
        item['tags'] = response.xpath("//div[@id='tags']/ul/li/a/text()").extract()
        yield item
```

Test XPath in Browser

Browser add-ons, for example, Xpath Helper for Chrome.



Output Result

Run from console: Easy export in CSV, JSON, XML

```
scrapy crawl etsy -o items.xml
scrapy crawl etsy -o items.csv
scrapy crawl etsy -o items.json
```

Feed Exports

Custom Exports can be defined

FEED_EXPORTERS_BASE

Default:

```
FEED_EXPORTERS_BASE = {
    'json': 'scrapy.exporters.JsonItemExporter',
    'jsonlines': 'scrapy.exporters.JsonLinesItemExporter',
    'csv': 'scrapy.exporters.CsvItemExporter',
    'xml': 'scrapy.exporters.XmlItemExporter',
    'marshal': 'scrapy.exporters.MarshalItemExporter',
}
```

Storages

Data can be stored:

- on local filesystem
- on FTP server
- S3
- standard output

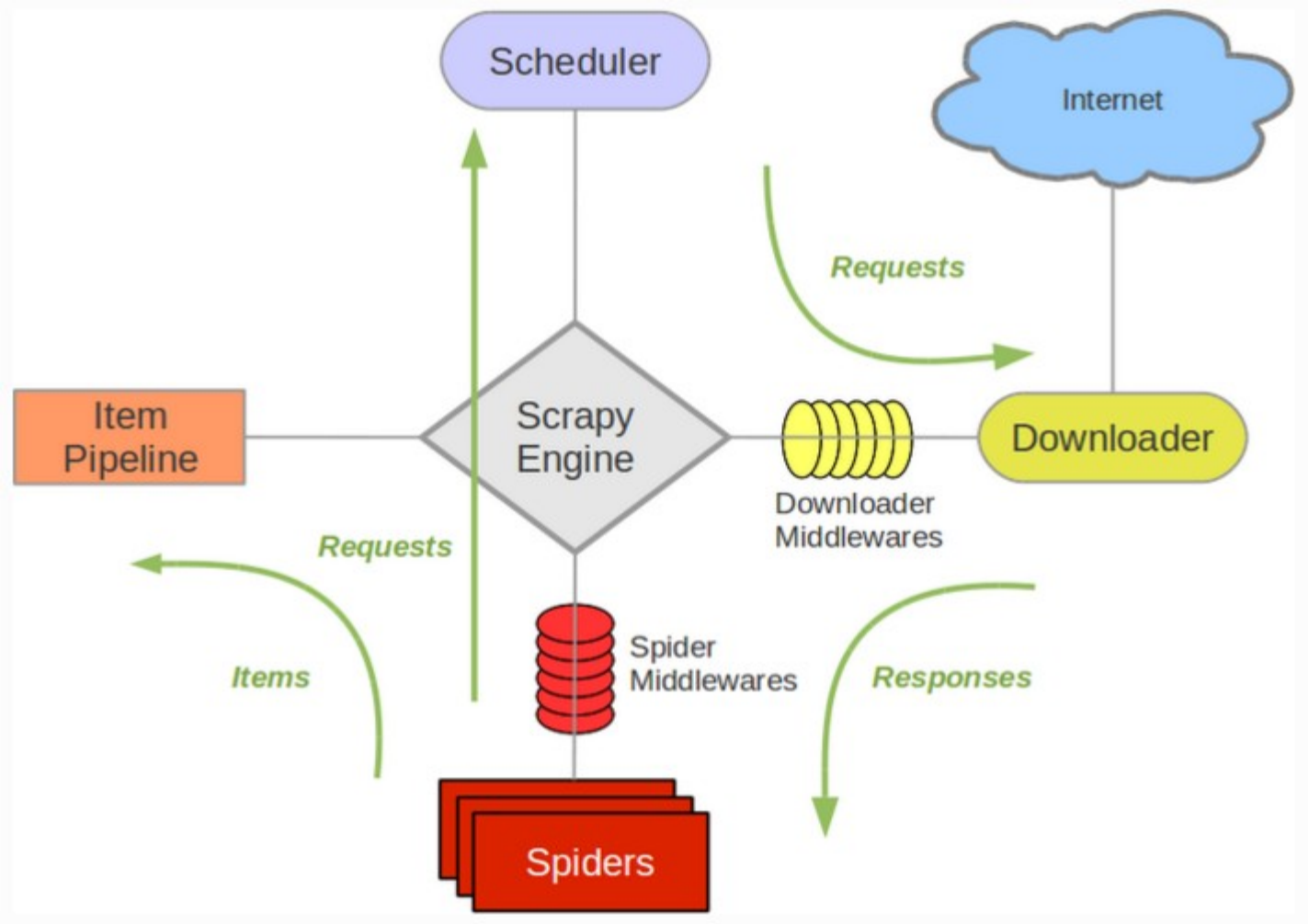
XML Output Example


```
- <item>
  - <tags>
      <value>Weddings</value>
      <value>Gifts & Mementos</value>
      <value>Gifts For The Couple</value>
      <value>map</value>
      <value>personalised frame</value>
      <value>typography</value>
      <value>wedding gift</value>
      <value>wedding present</value>
      <value>engagement present</value>
      <value>engagement gift</value>
      <value>marriage gift</value>
      <value>framed art</value>
      <value>where it all began</value>
      <value>anniversary gift</value>
      <value>marriage present</value>
      <value>cute wedding present</value>
    </tags>
    <description> ♥ Handmade and Made to Order ♥ This frame is dispatched between 3-5 working days. ♥ Gorgeous personalised picture that captures a couple's special
      moment. The handmade picture includes a map heart of a significant place, the phrase 'Where It All Began' and the couple's name with a date. This is a
      fantastice gift to buy for a couple for their engagement, the place where they are tying the knot or as a lovely present for your other half on Valentine's day, your
      anniversary or just because you love them. Each picture comes in a white frame approx. 10" x 10" x 1.75" and is finished with stunning Swarovski crystals! ~*~
      ♥ ~*~ Please include the name you require in the note to seller box before payment, this can be the couple's first names or 'Mr & Mrs.....', the date required and
      the location required for the map heart.~*~ ♥ ~*~ ♥ To return to our store front please click the following link: Thanks for visiting and I hope to see you again
      soon! xxx ♥ Want to stay connected and be the first to know about our sales and offers? You can find us on the following social networks: ~ Facebook:
      www.facebook.com/LittleMushroomCards ~ Twitter: @mushroomcards ~ Instagram: @littlemushroomcards Note: The intellectual property rights of all cards and
      gifts (and the images shown) are and will remain the property of Little Mushroom Cards. </description>
  - <title>
      <value>Where It all Began Personalised Art/Gift</value>
    </title>
  </item>
```

Deploy Spiders

- **Scrapyd**: open source app to run Scrapy spiders
  http://scrapyd.readthedocs.org/en/latest/
- **Scrapy Cloud**: "It's like a Heroku for Scrapy"
  http://scrapinghub.com/scrapy-cloud/

Architecture



Components

- Downloader Middleware - hooks into Scrapy's request/response processing
- Spider Middleware - hooks into Scrapy's spider processing mechanism; plug custom functionality for responses sent to spiders and requests and items from spiders;
- Item Pipeline - items scraped by a spider are sent to the Item Pipeline; classes for different item processing; a good place to put code for saving items to a DB;

Data Flow

1. The Engine opens a domain, locates the Spider that handles that domain, and asks the spider for the first URLs to crawl.
2. The Engine gets the first URLs to crawl from the Spider and schedules them in the Scheduler, as Requests.
3. The Engine asks the Scheduler for the next URLs to crawl.
4. The Scheduler returns the next URLs to crawl to the Engine and the Engine sends them to the Downloader, passing through the Downloader Middleware (request direction).
5. Once the page finishes downloading the Downloader generates a Response (with that page) and sends it to the Engine, passing through the Downloader Middleware (response direction).
6. The Engine receives the Response from the Downloader and sends it to the Spider for processing, passing through the Spider Middleware (input direction).
7. The Spider processes the Response and returns scraped items and new Requests (to follow) to the Engine.
8. The Engine sends scraped items (returned by the Spider) to the Item Pipeline and Requests (returned by spider) to the Scheduler
9. The process repeats (from step 2) until there are no more requests from the Scheduler, and the Engine closes the domain.

Jobs

```
scrapy crawl etsy -s JOBDIR=crawls/etsy-1
```

- Pause and resume crawls
- Saves the processed URLs

Avoid URL Repetition

- Setting DUPEFILTER_CLASS
- By default scrapy.dupefilters.RFPDupeFilter (request fingerprint)
- Can use a custom class

Documentation

http://doc.scrapy.org/en/latest/intro/overview.html
https://github.com/scrapy/scrapy
http://doc.scrapinghub.com/scrapy-cloud.html
http://scrapinghub.com/scrapy-cloud/

Demo Project:
https://github.com/tsvm/scrapy-etsy-demo