Building a Web Spider with Scrapy

Clarke Bishop
BIG DATA ENGINEER

@ClarkeBishop www.clarkebishop.com

Overview

Scrapy framework

Scrapy shell

Truecar spider code

Scrapy Framework

Scrapy

Scrapy is an application framework for crawling web sites and extracting structured data

CLI for creating a Scrapy project

Template to quickly build a spider

Scrapy shell for troubleshooting and refining selectors

Library or Framework

Library	Framework
Set of related functions	A complete application
You call it	It calls you!
Code structure is up to you	Structure specified by the framework

Development Environment

```
> cd my-dev
> mkdir scrapy
> cd scrapy
```

◀ Change to your dev directory. Make a new directory, and cd into the new directory.

```
> pyenv install 3.7.4
> pyenv local 3.7.4
```

◀ Set the local Python with pyenv.

```
> pipenv --python 3.7.4
> pipenv install scrapy
> pipenv shell
```

◀ Install packages and make sure to launch the pipenv shell.

Initialize Scrapy Project

```
> scrapy startproject truecar
```

```
├── scrapy.cfg
├── truecar
|   ├── __init__.py
|   ├── items.py
|   ├── main.py
|   ├── middlewares.py
|   ├── pipelines.py
|   ├── settings.py
|   ├── spiders
|   |   ├── __init__.py
|   |   └── truecar_spider.py
```

- Use Scrapy's startproject command to create a project.
- Scrapy initializes a full project with a spiders folder.
- We have to write our spider.

Run the Spider

```
> scrapy crawl truecar -o
truecar.csv
```

```
├── scrapy.cfg
├── truecar
|   ├── __init__.py
|   ├── items.py
|   ├── main.py
|   ├── middlewares.py
|   ├── pipelines.py
|   ├── settings.py
|   ├── spiders
|   |   ├── __init__.py
|   |   └── truecar_spider.py
|   ├── truecar.csv
```

◀ Tell Scrapy to run the spider and output data to truecar.csv.

◀ Scraped CSV Results.

truecar.csv

link	model	mileage	price
5YJ3E1EA9KF327202	2019 Tesla Model 3	5,873	$38,000
5YJ3E1EA8JF034955	2018 Tesla Model 3	16,241	$36,995
5YJ3E1EA0JF169640	2018 Tesla Model 3	421	$41,000
5YJ3E1EA9KF308973	2019 Tesla Model 3	2,775	$41,999

```python
import scrapy

class TruecarSpider(scrapy.Spider):
    name = "truecar"
    def start_requests(self):
        urls = ['https://www.truecar.com/used-cars-for-sale/listings/tesla/model-3/']
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        all_listings = response.xpath('//div[@data-qa="Listings"]')
        for tesla in all_listings:
            make_model = tesla.css('div[data-test="vehicleListingCardTitle"] > div')
            year = make_model.css('span.vehicle-card-year::text').get()
            model_raw = make_model.css('span.vehicle-header-make-model').get()
            model = model_raw[model_raw.find('>')+1:-7].replace('<!-- -->', '')
            tesla_data = {
                'url': 'http://truecar.com' + tesla.css('a::attr(href)').get(),
                'model': year + ' ' + model,
                'mileage': tesla.css('div[data-test="cardContent"] > div > div.text-
                           truncate::text').get(),
                'price': tesla.css('h4::text').get(),
            }
            yield tesla_data
```

Scrapy Shell

Scrapy Shell

```
> scrapy shell 'https://
en.wikipedia.org/wiki/
Tesla,_Inc.'
```

- Start scrapy shell with Wikipedia’s Tesla page.

```
>>> response.css
('table.wikitable tbody').get()
```

- It’s great for working out your selectors.

```
>>> view(response)
```

- See the response in a browser.

HTTP Request: User Agent

Identifies the browser or web scraper

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36

```
> scrapy shell -s USER_AGENT='Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36'
'https://www.truecar.com/used-cars-for-sale/listings/tesla/model-3/'
```

Scrapy Shell

```
> scrapy shell -s USER_AGENT='Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36'
'https://www.truecar.com/used-cars-for-sale/listings/tesla/model-3/'
```

Scrapy Shell

```
> scrapy shell -s USER_AGENT='Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36'
'https://www.truecar.com/used-cars-for-sale/listings/tesla/model-3/'
```

Scrapy Shell

Truecar Spider

Truecar Spider

```
├── scrapy.cfg
├── truecar
|   ├── __init__.py
|   ├── items.py
|   ├── main.py
|   ├── middlewares.py
|   ├── pipelines.py
|   ├── settings.py
|   ├── spiders
|   |   ├── __init__.py
|   |   └── truecar_spider.py
```

◀ truecar_spider.py is our Python code

```python
import scrapy

class TruecarSpider(scrapy.Spider):
    name = "truecar"
    def start_requests(self):
        urls = ['https://www.truecar.com/used-cars-for-sale/listings/tesla/model-3/']
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        all_listings = response.xpath('//div[@data-qa="Listings"]')
        for tesla in all_listings:
            make_model = tesla.css('div[data-test="vehicleListingCardTitle"] > div')
            year = make_model.css('span.vehicle-card-year::text').get()
            model_raw = make_model.css('span.vehicle-header-make-model').get()
            model = model_raw[model_raw.find('>')+1:-7].replace('<!-- -->', '')
            tesla_data = {
                'url': 'http://truecar.com' + tesla.css('a::attr(href)').get(),
                'model': year + ' ' + model,
                'mileage': tesla.css('div[data-test="cardContent"] > div > div.text-
                           truncate::text').get(),
                'price': tesla.css('h4::text').get(),
            }
            yield tesla_data
```

```python
import scrapy


# Spider for truecar.com
class TruecarSpider(scrapy.Spider):
    name = "truecar"

    def start_requests(self):
    . . .


    def parse(self, response):
    . . .
```

```python
import scrapy


# Spider for truecar.com
class TruecarSpider(scrapy.Spider):
    name = "truecar"

    def start_requests(self):
    . . .


    def parse(self, response):
    . . .
```

```
class TruecarSpider(scrapy.Spider):

    name = “truecar"

    def start_requests(self):

      urls = ['https://www.truecar.com/used-cars-for-sale/listings/tesla/model-3/']

      for url in urls:
         yield scrapy.Request(url=url, callback=self.parse)
```

Yield in Python

Yield helps with processes that have a delay—like waiting on a web page to load

Pause run to completion — Creates a series of values over time

More memory efficient and faster

Think of yield as a lazy return

```
class TruecarSpider(scrapy.Spider):
    . . .

    def parse(self, response):

        all_listings = response.xpath(‘//div[@data-qa="Listings"]')

        for tesla in all_listings:

           . . .

           tesla_data = {
                'url': 'http://truecar.com' + tesla.css('a::attr(href)').get(),
                'model': year + ' ' + model,
                'mileage': tesla.css('div[data-test="cardContent"] > div > div.text-
                           truncate::text').get(),
                'price': tesla.css('h4::text').get(),
            }

            yield tesla_data
```

```
for tesla in all_listings:

    make_model = tesla.css('div[data-test="vehicleListingCardTitle"] > div')
    year = make_model.css('span.vehicle-card-year::text').get()
    model_raw = make_model.css('span.vehicle-header-make-model').get()
    model = model_raw[model_raw.find('>')+1:-7].replace('<!-- -->', ‘')

    tesla_data = {
        'url': 'http://truecar.com' + tesla.css('a::attr(href)').get(),
        'model': year + ' ' + model,
        'mileage': tesla.css('div[data-test="cardContent"] > div > div.text-
truncate::text').get(),
        'price': tesla.css('h4::text').get(),
    }

    yield tesla_data
```

```html
<div data-test=“vehicleListingCardTitle" data-qa="VehicleCardHeader">
    <div data-test="vehicleCardYearMakeModel"
         class="vehicle-card-header w-100”>
        <span class="vehicle-card-year font-size-1">2019</span>
        <span class="vehicle-header-make-model text-truncate”>
            Tesla<!-- --> <!-- -->Model 3</span>
    </div>
</div>
```

```python
make_model = tesla.css('div[data-test="vehicleListingCardTitle"] > div')
```

```html
<div data-test="vehicleCardYearMakeModel"
     class="vehicle-card-header w-100">
    <span class="vehicle-card-year font-size-1">2019</span>
    <span class="vehicle-header-make-model text-truncate">
        Tesla<!-- --> <!-- -->Model 3</span>
</div>
```

```
year = make_model.css('span.vehicle-card-year::text').get()
'2019'
```

```
<div data-test="vehicleCardYearMakeModel"
     class="vehicle-card-header w-100">
    <span class="vehicle-card-year font-size-1">2019</span>
    <span class="vehicle-header-make-model text-truncate">
        Tesla<!-- --> <!-- -->Model 3</span>
</div>
```

```
model_raw = make_model.css('span.vehicle-header-make-model').get()
model = model_raw[model_raw.find('>')+1:-7].replace('<!-- -->', '')
'Tesla Model 3'
```

```
<div data-qa="Listings" class="margin-top-3 col-md-6 col-xl-4">
    <a class="card card-1 card-shadow card-shadow-hover vehicle-card"
       data-test="usedListing" data-qa="VehicleCardUsedCar"
       href="/used-cars-for-sale/listing/5YJ3E1EA9KF327202/2019-tesla-model-3/">

        <div class="vehicle-card-top" data-qa="VehicleCardTop"> ... </div>
    <div class="d-flex margin-top-1 w-100 justify-content-between"> ... </div>
    <div class="vehicle-card-bottom" data-qa="VehicleCardBottom"> ... </div>
    </a>
</div>
```

```
all_listings = response.xpath('//div[@data-qa="Listings"]')
```

Real World Scraping

Use private or incognito mode

Save the downloaded page locally

Use an IDE and a debugger

Break down the problem one HTML chunk at a time

Web scraping is brittle and prone to break

Summary

Scrapy framework

- Libraries & Frameworks
- Use Scrapy to setup a project
- Use Scrapy to launch a spider

Scrapy shell

Truecar spider code

- Yield
- Common scraping problems