Wine Quality Prediction

Chukwunonso Ebele-Muolokwu

Table of contents

1 How to Scrape a Website with Scrapy: A Beginner's Guide

Figure 1: Altered Image of Scrapy and AOT Characters

2 Introduction

If you're new to web scraping and want a powerful tool to automate your data collection tasks, look no further than Scrapy. This Python-based framework makes it easy to crawl websites, extract data, and store it in your desired format. In this blog post, we'll walk you through scraping a website with Scrapy step by step. This part focuses on setting up a Scrapy project locally and running it in the terminal. Most scrapy projects are ran from the terminal using the command scrapy crawlIf you are new to Scrapy and also an Attack on Titan fan, well, today is your day!

2.1 What is Scrapy

TL;DR Scrapy is an open-source Python framework used for web scraping and crawling. It provides a set of tools and libraries that simplify the process of extracting data from websites. With Scrapy, developers can define the structure of a website, specify how to navigate its pages, and extract desired data by writing Python code.

It supports various features such as handling cookies, managing sessions, following links, and handling form submissions. Scrapy also includes powerful mechanisms for handling asyn-

chronous requests and managing concurrency, allowing for efficient and fast scraping of multiple pages simultaneously.

Scrapy is revered for being an asynchronous library built on top of Twisted and also being a full engine where you can scrape, parse the data, perform further preprocessing using itemloaders, and store processed data within a database.

3 Getting Started With the Project

It is time to get to the real coding, and I know you are petty excited so let's get to it

Figure 2: Image from giphy.com

3.1 Step 1: Setting up your Enviroment and Instally Scrapy:

It is recommended practice to construct a separate virtual environment for each of your Python projects in order to prevent version conflicts in the future. This ensures that any packages you install for one project are kept apart from those for other projects, preventing accidental project breakage. These commands may be slightly different on your machine, depending on the operating system.

Figure 3: Doggo scared of peole that dont use Virtual Environments (Image from Pintrest)

The following method is derived from scrapeops and the official scrapy documentation.

3.1.0.1 MacOS/Linux

Setting up a virtual environment on MacOS or any Linux distro uses the following commands.

```
$ sudo apt install -y python3-venv
```

Next, we create a virtual environment in our project directory.

```
$ cd /<folder name>
$ python3 -m venv venv
$ source venv/bin/activate
```

In our virtual environment, we will then install Scrapy using the following command:

```
$ apt-get install python3-pip
$ sudo pip3 install scrapy
```

Navigate to the project folder where you want to create the virtual environment, create the virtual environment, and activate it. The following commands achieve this

```
cd /<folder_name>
python<version> -m venv <virtual-environment-name>
```

Finally we activate the virtual environment and install scrapy

```
source <name of virtual env>\Scripts\activate
pip install scrapy
```

if you are using anaconda use the cell below

```
conda activate <enviroment_name>
conda install -c conda-forge scrapy
```

3.2 Step 2—Setting up our AOT Scrapy Project

We may move on to the enjoyable aspects now that our setting is set up. construction of the first ***Scrapy spider***! Spiders are **classes which define how a certain site (or a group of sites) will be scraped**, including how to perform the crawl (i.e. follow links) and how to extract structured data from their pages (i.e. scraping items).

This project will consist of three spiders

- **aot_locations:** this spider would be in charge or scraping the location page on AOT fan weblink site

- **aot_organizations:** This spider would be in charge of scraping AOT **organization** site link
- **aot_titans:** This spider would scrape informations about the different titans on the show(Link)

Open your command line and type in the following CL command:

```
#we can call the project aot in our case
scrapy startproject <project_name>
```

3.2.0.1 The Scrapy Project Structure

Scrapy has a generic project structure when you create a new project. After running the scrapy startproject command, it generates a file structure The folder structure looks something like this

```
  scrapy.cfg
  <project_name>
      __init__.py
      items.py
      middlewares.py
      pipelines.py
      settings.py
      spiders
          __init__.py
```

All of your project settings, such as enabling pipelines, middlewares, and other components, are located in **settings.py**. The latency, concurrency, and a lot more things can all be changed here.

The **item.py** file is used to define the preprocessing pipeline model for the extracted data. You can create a special model that inherits from the Scrapy Item class. The **scrapy.cfg** is a configuration file to change some deployment settings, etc.

3.2.1 Writing the Spider

I will quickly go over the structure and logic of one of the spiders on this Attack on Titans scrapy project to get a grasp of it. Navigate to the spiders directory and create a new spider file:

```
cd quotes_scraper/quotes_scraper/spiders
touch aot_spider.py
```

Edit quotes_spider.py to define your spider:

```
import scrapy
from scrapy.loader import ItemLoader
from ..items import AotTitanItem


class AotTitansSpider(scrapy.Spider):
    """
    A spider for scraping data about titans from the Attack on Titan Fandom wiki

    Attributes:
        name: name of the spider, string for identifying this spider
        allowed_domains: list containing the domains allowed for the spiders
        start_urls: list of the urls to begin scraping
        custom_settings: dictionary of settings for this specific spider
    """

    name = "aot_titans"
    allowed_domains = ["attackontitan.fandom.com"]
    start_urls = [
        "https://attackontitan.fandom.com/wiki/Colossal_Titan_(Anime)",
        "https://attackontitan.fandom.com/wiki/Armored_Titan_(Anime)",
    ]
    custom_settings = {
        "FEEDS": {"./aot_scraper/scrapes/titan_data.jl": {"format": "jsonlines"}},
    }

    def parse(self, response):
        """
        Function for scraping titan information from the start_urls,
        and supplying scraped data to the item loader

        Args:
            response: response from start_urls
        """

        # page section with information to scrape
        info_block = response.css(
```

```
            "div#mw-content-text aside.portable-infobox.pi-background.pi-border-color.pi-ther
        )

        # specific HTML elements with content
        height_div = info_block.xpath(".//div[@data-source='Height']")
        powers_div = info_block.xpath(".//div[@data-source='Abilities']")
        current_shifter = info_block.xpath(
            ".//div[@data-source='Current inheritor(s)']"
        )
        former_shifter = info_block.xpath(".//div[@data-source='Former inheritor(s)']")

        # text retrieval from HTML
        height_data = height_div.css("div.pi-data-value.pi-font *::text").get()
        powers_data = powers_div.css("div.pi-data-value.pi-font *::text").getall()
        current_shifters_data = current_shifter.css(
            "div.pi-data-value.pi-font a::text"
        ).getall()
        former_shifters_data = former_shifter.css(
            "div.pi-data-value.pi-font a::text"
        ).getall()

        # instantiate item loader
        titan_item_loader = ItemLoader(item=AotTitanItem(), selector=info_block)

        # use item loader to populate fields
        titan_item_loader.add_css("name", "div.pi-data-value.pi-font::text")
        titan_item_loader.add_value("height", height_data)
        titan_item_loader.add_value("powers", powers_data)
        titan_item_loader.add_value(
            "shifters", current_shifters_data + former_shifters_data
        )

        yield titan_item_loader.load_item()
```

Key Components of the Spider

SPIDER ATRRIBUTES

- **name:** The name of the spider (used to run the spider). In this case, it's "*aot_titans*".
- **allowed_domains:** Specifies the domains the spider is allowed to crawl. This prevents the spider from inadvertently accessing unrelated websites.Here, it restricts scraping to attackontitan.fandom.com.

- **start_urls:** Contains the list of URLs the spider will visit initially. Each URL corresponds to a specific Titan's page in the wiki.
- **custom_settings:** Overrides default Scrapy settings for this spider. In this case, The scraped data is saved in JSON Lines format (.jl) to the file path ./aot_scraper/scrapes/titan_data.jl.

PARSE METHOD

- The parse method is the heart of the spider. It processes the response received from each URL in start_urls and extracts relevant data.

```
info_block = response.css(
    "div#mw-content-text aside.portable-infobox.pi-background.pi-border-color.pi-theme-w
)
```

- Uses a CSS selector to locate the HTML section (aside.portable-infobox) containing Titan-related details.
- This block acts as the primary source of structured data on the page.

SCRAPY ITEM

```
titan_item_loader = ItemLoader(item=AotTitanItem(), selector=info_block)
```

- The ItemLoader simplifies the process of populating the Scrapy AotTitanItem object.
- The fields in the item are populated with extracted data:

```
titan_item_loader.add_css("name", "div.pi-data-value.pi-font::text")
titan_item_loader.add_value("height", height_data)
titan_item_loader.add_value("powers", powers_data)
titan_item_loader.add_value("shifters", current_shifters_data + former_shifters_data)
```

RUNNING THE SPIDER This is the time to test your spider and see the outputted result. It is always adviced to look at the output result on the terminal to ensure your spider is not encountering any error. To run yoiur spider, you run:

```
scrapy crawl quotes
```

If you want to output the result to a file, e.g. a csv or json file format you run a command similar to this (You can always check the scrapy documentation for the updated file format support):

```
#Outputs to a json file called "aot.py"
scrapy crawl quotes -o aot.json
```

You can check out the output file to see the result of your hardwork!

3.3 FINAL THOUGHTS

Congratulations! You've successfully scraped a website using Scrapy. This tutorial covered the basics, but Scrapy offers much more, including middleware, custom settings, and advanced spider types like CrawlSpider. For more advanced use cases, refer to the Scrapy documentation.