Scrapy Project Documentation

By Adnan Rasool

Overview

This project leverages Scrapy, a powerful web crawling and scraping library, to extract data from the website Books to Scrape. It collects data on books, including their title, price, rating, and availability. The scraped data is then processed and stored in a CSV file for further analysis.

1. Spider Implementation

File: `books_spider.py`

The spider defines how the data is scraped from the website.

Key Components

- **Spider Name**: `"books"`
  This is used to run the spider using the Scrapy command line.

- **Allowed Domains**: Ensures the spider scrapes only the specified domain (`books.toscrape.com`).

- **Start URL**: The initial URL to begin scraping.

Core Methods

1. `parse(self, response)`

   - Extracts book details such as title, price, rating, and availability.
   - Each book is represented as an `<article>` HTML element with a class `product_pod`.

   Code Snippet:

```
for book in response.css("article.product_pod"):
    title = book.css("h3 a::attr(title)").get()
    price = book.css("p.price_color::text").get()
    rating = book.css("p.star-rating::attr(class)").get().replace('star-rating', '').strip()
    availability = book.css("p.availability::text").getall()
    availability = ''.join([text.strip() for text in availability]).strip()

    yield {
        "title": title,
        "price": price,
        "rating": rating,
        "availability": availability,
    }
```

2. **Pagination**:

   - Detects and follows the “next page” link to scrape data from additional pages.
   - Utilizes the `response.follow()` method.

2. Data Pipeline

File: `pipelines.py`

The pipeline processes and writes the scraped data into a CSV file.

Key Components

1. `open_spider(self, spider)`
    - Initializes the CSV file for writing.
    - Defines the field names: `title`, `price`, `availability`, and `rating`.
2. `process_item(self, item, spider)`
    - Cleans and formats the data:
        - Removes non-numeric characters from the `price` field.
        - Converts `rating` from text (e.g., "One") to numeric values using a mapping dictionary.
    - Writes the processed data to the CSV file.

    Code Snippet:

    ```
    rating_mapping = {
        "One": 1,
        "Two": 2,
        "Three": 3,
        "Four": 4,
        "Five": 5
    }
    item["rating"] = rating_mapping.get(item["rating"], 0)
    ```

3. `close_spider(self, spider)`
    - Closes the CSV file after the spider completes its execution.

3. Usage Instructions

Setup

1. Install Scrapy:

    ```
    pip install scrapy
    ```

2. Ensure the following files are in the same directory:
    - `books_spider.py`
    - `pipelines.py`

Run the Spider

Use the following command to start the spider and save the data:

```
scrapy runspider books_spider.py
```

Output

- A file named `books.csv` will be created in the project directory, containing the scraped data in a structured format.

4. Data Fields

Scraped Fields

1. **Title**: Name of the book.
2. **Price**: Price of the book (e.g., `£51.77` is converted to `51.77`).
3. **Rating**: Numeric rating (e.g., "Three" → `3`).
4. **Availability**: Stock status (e.g., `"In stock"`).

5. Enhancements

Here are some potential improvements:

1. **Error Handling**:
    - Add mechanisms to handle missing or malformed data.
2. **Database Storage**:
    - Store scraped data in a database (e.g., SQLite or PostgreSQL) for better querying.
3. **Additional Data**:
    - Scrape more fields like book description, genre, and publisher.
4. **Concurrency**:
    - Utilize Scrapy's concurrency settings for faster scraping of large datasets.