**CityGrossScraper Documentation**

Please run the script from scrapy-selenium-project\selenium\_scraper\selenium\_scraper\citygrossScrapper.py

Inside the folder there is a product\_data.csv already created containing all the products found on the citygross.se website. The code trive in efficiency of obtaining all the available products on the website but it lacks on the time it takes to obtain all of them, estimated wait time is around 8 hours.

**Introduction**

The CityGrossScraper is a sophisticated Python-based tool designed for efficiently extracting and compiling product information from the City Gross website into a structured CSV file. This document serves as a comprehensive guide to utilizing the scraper, understanding its functionalities, and customizing it to suit specific data extraction needs.

**Installation Requirements**

Before deploying the CityGrossScraper, ensure your system meets the following prerequisites:

Python 3.x installed.

External libraries: requests, re, csv, datetime, pytz, and python-dateutil. These can be installed via pip.

Example: pip install requests python-dateutil pytz

**Class Overview: CityGrossScraper**

The CityGrossScraper class is designed with encapsulation of all necessary functionalities to navigate through categories, fetch product details, and output the data into a CSV file. Below is a detailed overview of its components.

**Attributes**

* headers: Mimics browser requests to ensure successful HTTP requests.
* base\_url: Endpoint for fetching product data.
* delay\_seconds: Controls request rate to avoid server overload.
* responses\_with\_products: Accumulates fetched product data.
* unique\_product\_signatures: Ensures data uniqueness.
* current\_date: Checks promotion validity.
* csv\_headers: Defines the structure of the output file.
* Key Methods
* fetch\_navigation(): Retrieves navigation data to find category IDs.
* extract\_category\_ids(data): Parses category IDs from navigation data.
* fetch\_product\_data(category\_ids): Collects data for specified categories.
* save\_to\_csv(filename='products\_data.csv'): Outputs data to CSV.
* prepare\_product\_info(product\_data, product\_signature): Formats product information.
* run(): Executes the scraping process.

**Customizing Your Scraper**

**Adjusting Category Scraping**

By default, the scraper processes the first four categories due to time and resource efficiency. However, City Gross hosts over 1600 categories. To modify the number of categories:

1. Navigate to the extract\_category\_ids method.
2. Adjust the slice in the return statement to include more or fewer IDs.

**Modifying Request Delay**

To prevent overwhelming the City Gross servers, a delay is introduced between requests. To adjust this delay:

In the \_\_init\_\_ method, change the delay\_seconds attribute to your preferred wait time.

**Configuring CSV Output**

The default CSV file includes predefined headers. To alter these modify the csv\_headers attribute in the \_\_init\_\_ method to match your desired output structure.

**Execution Guide**

To run the scraper:

1. Ensure all requirements are installed.
2. Instantiate the CityGrossScraper.
3. Call the run method.

This action initiates the scraping process, with progress logged to the console. Upon completion, the products\_data.csv file will be available in your working directory.