VYGA- WEB SCRAPING AND ADMIN PANEL

REPORT

Overview

This web scraping is a process employed to extract data from Superrar.com, Hashaxis.com, and Rareable.com. We have leveraged a combination of Selenium for web automation, Celery Workers for task distribution within a Django Rest application, and the parsing capabilities of Beautiful Soup and SelectorLib to effectively scrape, filter, and store data from these websites.

Tools and Technologies

- **Selenium**: A versatile web automation tool used for browser control and data retrieval.
- **Celery Workers**: A distributed task queue system integrated with Django Rest, designed to efficiently distribute scraping tasks.
- **Django Rest Framework**: A Django extension tailored for creating RESTful APIs.
- **SelectorLib**: A web scraping library for creating and running web scraping robots.
- **Beautiful Soup**: A Python library designed for parsing HTML and XML documents.
- **Rayobyte Residential Proxies**: Rayobyte Proxies, specifically residential proxies, play a pivotal role in ensuring the anonymity and reliability of your web scraping efforts. These proxies effectively mask your IP addresses, making it challenging for websites to trace your scraping activities back to your original IP. Residential proxies, sourced from real IP addresses of internet users, further enhance your web scraping capabilities.

Workflow

Environment Setup

Ensure that your development environment is equipped with the necessary tools and libraries, including Django Rest, Selenium, Celery, SelectorLib, and Beautiful Soup.

Defining Scraping Objectives

Clearly define the objectives of the scraping tasks to be performed on Superrar.com, Hashaxis.com, and Rareable.com. These objectives may encompass extracting data such as NFT details, pricing information, and source links.

Implementing Selenium Automation

Selenium is used to automate the interaction with the target websites. Here's a high-level overview of the process for each task:

- Launch a web browser (e.g., Chrome or Firefox) programmatically using Selenium.
- Navigate to the target website(s).
- Employ Selenium to interact with the website, including actions like clicking, scrolling, and loading additional data.
- Retrieve the HTML source code of the web pages.

Task Distribution with Celery Workers

Celery Workers play a pivotal role in efficiently distributing scraping tasks. Define individual Celery tasks for each scraping objective, allowing for asynchronous execution to scrape multiple pages concurrently.

HTML Parsing with Beautiful Soup

Once you have secured the HTML source code of the web pages, Beautiful Soup is employed for parsing and data extraction. This encompasses:

- Locating and extracting data elements by parsing the HTML using HTML tags and attributes.

Creating SelectorLib Robots

Utilize SelectorLib to create scraping robots that define the structure of the data you intend to extract. This is achieved by defining CSS selectors for the relevant data elements and storing these configurations in JSON files for each target website.

Executing SelectorLib Robots

Execute the SelectorLib robots that have been configured, enabling the extraction of specific data elements defined in your configurations. The extracted data is then structured for storage.

Data Storage in the Database

Store the scraped data within your Django database. Define appropriate database models and schemas to accommodate the extracted data structure.

Best Practices

- Respect website terms of service and employ rate limiting to prevent overloading target websites.
- Utilize user agents and proxies to mitigate IP blocking risks.
- Implement robust error handling to gracefully manage unexpected website changes or connectivity issues.
- Periodically update scraping robots and configurations to adapt to any alterations on the target websites.

Customized Admin panel

Dashboard

The Dashboard is a central hub within the web scraping system, providing users with a comprehensive view of the scraped NFT data. It caters to both SuperAdmin and User roles, offering different levels of access and functionality.

Features:

1. NFT Ratio Bar Chart (Based on Source)

- **Objective:** Visual representation of the distribution of NFTs based on their source.
- **Visualization:** A bar chart illustrating the ratio of NFTs from different sources.
- **Interaction:**Users can hover over bars to view specific ratios.

2. Most Viewed NFTs Table

- **Objective:** Display a table showcasing the most viewed NFTs.
- **Sorting:**Users can sort the table based on various columns.
- **Interaction:**Clicking on a row provides detailed information about the selected NFT.

3. New Account Details

- **Objective:** Provide information about new user accounts.

4. Number of Sales (In Last 30 Days)

- **Objective:** Display the number of NFT sales within the last 30 days.

5. Average Basket Value (In Last 30 Days)

- **Objective:** Present the average value of baskets (collections) in the last 30 days.

6. Chains in Collection

- **Objective:** Display the number of unique chains represented in the NFT collection

7. Dashboard Interface

- **Layout:** Organized and visually appealing dashboard layout with sections for each feature.
- **Interactivity:** Users can navigate between different sections seamlessly.

Data Products

Features:

User Categories:

- **SuperAdmin:** Full access and the ability to create additional SuperAdmins using roles and permissions.
- **User:** Limited access based on assigned roles, including create, update, delete, and view permissions.

Sidebar Options:

a) Dashboard:

- Accessible by any user.
- Provides an overview of key system metrics.

b) All:

- Displays all scraped products.
- Accessible to SuperAdmin and users with specific permissions.

c) Chains:

- Displays products categorized by chains or currency modes (e.g., Hbar, Eth, Immutable X, Polygon).
- Accessible to SuperAdmin and users with specific permissions.

d) Source:

- Displays products filtered by the source of scraping (e.g., Hash Axis, Superrare, Rareable).
- Accessible to SuperAdmin and users with specific permissions.

e) Category:

- Displays products categorized by user-defined categories.
- Accessible to SuperAdmin and users with specific permissions.

f) Status:

- Filters products based on status (live or draft).
- Accessible to SuperAdmin and users with specific permissions.

g) Update/Scrapping:

- Initiates scraping of NFTs from different sources or updates current product details.
- Accessible to SuperAdmin and users with specific permissions.

h) Upload/Add Category:

- Adds products via CSV to a specific category or creates a new category.
- Accessible to SuperAdmin and users with specific permissions.

i) Roles:

- **Permission:** Manages user roles and permissions and can update details of users.
- **Create User:** Creates new users, assigns roles, and sends login credentials via email.
- Accessible to SuperAdmin only.

Conclusion

Web scraping serves as a potent means of acquiring valuable data from websites for a multitude of purposes. This documentation has provided an insight into the process of utilizing Selenium, Celery Workers, SelectorLib, and Beautiful Soup to scrape data from Superrar.com, Hashaxis.com, and Rareable.com, with the end goal of storing it in a Django database. When executed effectively, web scraping can yield invaluable data for analysis, reporting, and further utilization.

Furthermore, the introduction of the Customized Admin Panel, featuring an enhanced Dashboard and Data Products sections, provides a user-friendly interface for efficiently handling web scraping activities and delving into the scraped NFT data. The comprehensive documentation ensures a clear understanding of user roles, features, sidebar options, and detailed explanations of each block. Regular updates to this documentation are emphasized to maintain clarity and accuracy in the utilization of the system for ongoing and future endeavors.