Project Report: Web Scraper

1. Introduction

This project involves developing a web scraper using Python to extract data from the website Scrape This Site. The extracted data is then stored in a Supabase database. The scraper utilizes multiprocessing to enhance performance by parallelizing the data extraction process.

2. Project Structure

The project is organized into the following main directories and files:

- **database/**: Contains configurations and functions for interacting with Supabase.
    - **database.py**: Configurations and functions to insert and retrieve data from Supabase.
- **web_scraping/**: Contains the web scraping logic.
    - **scrapeadvanced.py**: Core scraping functions to fetch and parse data from the advanced page.
    - **scrapeform.py**: Core scraping functions to fetch and parse data from the form page.
    - **scrapemovies.py**: Core scraping functions to fetch and parse data from the ajax page.
- **multiprocess.py**: Main script to initiate the scraper with multiprocessing.
- **requirements.txt**: Lists all the dependencies required for the project.

3. Prerequisites

To run this project, you need:

- Python 3.x installed on your machine.
- A Supabase account and an appropriate database setup.

4. Installation and Setup

4.1 Clone the Repository

Clone the repository from GitHub:

```
git clone https://github.com/Parth2k3/web-scraper.git
cd web-scraper
```

4.2 Install Dependencies

Install the required Python packages using pip:

```
pip install -r requirements.txt
```

4.3 Configure Supabase

Set up your Supabase project and obtain the URL and API key. Configure these credentials in the **database/database.py** file:

SUPABASE_URL = 'your_supabase_url' SUPABASE_KEY = 'your_supabase_key'

5. Functionality

5.1 Web Scraping

The **web_scraping/** directory contains the files necessary to scrape data from the websites. Key functionalities include:

- Fetching HTML content from the target website.
- Parsing HTML to extract relevant data.
- Handling pagination to scrape data from multiple pages.

5.2 Data Storage

The **database/database.py** file contains functions to interact with Supabase. It includes:

- Inserting scraped data into the database.
- Retrieving data from the database.
- deleting the table from database

5.3 Multiprocessing

The **multiprocess.py** script initiates the scraping process using multiprocessing to speed up data extraction. It divides the workload among multiple processes/threads, enabling concurrent data scraping.

6. Usage

To start the web scraping process with multiprocessing, run the following command:

python multiprocess.py

This command will execute the scraper, which will fetch data from the website and store it in the Supabase database.

7. Error Handling

The project uses Try&Except statements throughout the code to ensure smooth error handling and detailed display of the exception if encountered during the code runtime.

8. Conclusion

This project demonstrates the use of Python for web scraping, multiprocessing for performance enhancement, and Supabase for data storage. The modular structure of the project makes it easy to extend and maintain.

9. References

- Scrape This Site
- Supabase
- BeautifulSoup Documentation
- Requests Documentation