Документація для проєкту Book Search Automation

Огляд

Book Search Automation — це Python-проєкт, який парсить дані про книги на Amazon із спеціально заданими вхідними даними (в даному випадку книги із назвою “Java”). Отриманні данні про книги зберігються у CSV-файл.

Залежності

- Python 3.12
- Selenium 4.24.0
- ChromeDriver 128.0.6613.119
- Модуль CSV
- time

Структура файлів

1. main.py -> Головний скрипт
2. searchPage.py -> Клас SearchPage
3. bookListPage.py -> Клас BookPage
4. specificBookPage.py -> Клас SpecificBookPage
5. csvDataCleaner.py -> Клас CSVDataCleaner
6. Archive/
    6.1. books.csv -> Вихідний файл з даними про книги
    6.2. books_cleaned.csv -> Очищений від дублікатів файл з даними про книги
7. chromedriver/
    7.1. chromedriver.exe -> Виконуваний файл ChromeDriver

Flow виконання програми

1. Скрипт ініціалізує Selenium, інші модулі та бібліотеки. Точка входу -> main(). Відбувається вивід консольного інтерфейсу. У користувача є можливість вибрати скільки сторінок він хоче спарсити (для цього треба ввести число) або вибрати опцію для парсингу усіх можливих сторінок з даною книгою (треба ввести “all”).
2. Переходимо до логіки парсингу. Ініціалізуємо WebDriver, та передаємо посилання для переходу (в данному випадку https://www.amazon.com/). Визиваємо методи

`go_to_books_section()` та `search_for_books("Java")` із класу **SearchPage**. Виконується вибір фільтру - “Books” та пошук книг по нашому запросу.

3. Вхід в цикл. Визов методу `get_book_elements()` із класу `BookPage` для отримання усіх елементів сторінки з потрібним ключем (в данному випадку усі позиції товару на одній сторінці)
4. Вхід в цикл для оббігу всіх елементів `book_elements`. Оновлення списку усіма елементами, що на разі є на сторінці з потрібним ключем. (зроблено для того, щоб уникнути видалення елементу із DOM і як наслідок, унеможливлення його знаходження). Визов методу `open_specific_book(book_element)` із класу `BookPage` для відкриття сторінки із поточною книгою. Визов методу `get_book_data(book_element)` із класу `SpecificBookPage` для парсингу усієї потрібної інформації про книгу. При відсутності або некоректності інформації, підставляється значення -> `"Unknown"` або `"0.00"`

   * Для парсингу ціни прийшлось використовувати декілька XPATH-ів, бо html розмітка на різних сторінках товарів, може суттєво відрізнятися, тому довелося використовувати обробку різних варіанів, а також обробку ціни у форматі: “$10 - 15”. В такому випадку знаходиться середнє арифметичне цих двох цін.

   Значення записуються у масив та відбувається вихід із поточної сторінки на минулу.
5. Коли сторінка закінчується (відбувається перевірка, на вибір який був дан користувачем на початку роботи застосунка), визвиається метод `get_next_page()` із класу `BookPage` для відкриття наступної сторінки.
6. Відбувається сортування усіх данних записаних у масиві, за ключем `"price"` в порядку спадання
7. Запис результутаів парсинга в файл `books.csv`
8. Визов методу `clean_csv()` із класу `CSVDataCleaner` для видалення дублікатів за певною колонкою у масиві (в нашому випадку `"title"`) . Отриманні очищенні данні записуються в файл `books_cleaned.csv.` Запис у двох різних файлах реалізований для порівнянна та аналізу коректності зібранної інформації.

   *Пілся парсингу (особливо це бачно після парсингу великого обсягу сторінок), деякі позицію дублюються, хоча посилання в них різне. Тому для чистоти данних, використовується цей метод.

Обробка помилок

Проект включає базову обробку помилок для застарілих елементів та інших поширених винятків. Якщо виникає будь-яка несподівана помилка, вона обробляється, виводиться на консоль, і браузер закривається коректно.

Деталі

- Для коректної роботи проєкта на інших машинах, треба замніти шлях до chromedriver.exe на свій у змінній `service` (13 рядок, файл - main)