ПРИКЛАДНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК,
БЕЗПЕКИ ТА МАТЕМАТИКИ, 2023, № 1

УДК 004.4'2

ОТРИМАННЯ ДАНИХ З ВЕБСАЙТІВ НА ПРИКЛАДІ РОЗРОБКИ ПЛАГІНУ ДЛЯ ВЕБСКРАПІНГУ

Павленко Ю.С. (ORCID: 0000-0002-4065-045X),
Пелех Г.В.
Волинський національний університет імені Лесі Українки

EXTRACTING DATA FROM WEBSITES ON THE EXAMPLE OF DEVELOPING A PLUGIN FOR WEB SCRAPING

Pavlenko Y., Pelekch H.V.
Lesya Ukrainka Volyn National University

Abstract. Internet search involves not only getting acquainted with this or that information in order to gain new knowledge, it also involves comparing, analyzing and summarizing data found on different websites and pages. This problem can be solved by monotonous manual copying of the required information into your own files and formatting them according to your needs. Such a process is inefficient because it takes a lot of time.

Web scraping is an automated process of extracting huge amounts of data from websites and converting it into structured data. Programs that carry out such processes are called web scrapers and are able to retrieve required HTML content, work with JavaScript, filter received data and output it as a ready-to-use database, Excel spreadsheet, CSV file or separate API etc.

Presented is an implementation option of web scraping plugin for Chromium-based browsers. The plugin has client and server sides. Implementing client part involves designing and creating interface of plugin pop-up window, web page elements highlight on hover. Server side of the plugin analyzes requests, generates links and performs element selection.

Keywords: web scraping, extracting data, web site.

Вступ. Обсяг даних, які постійно розміщуються та передаються з допомогою Інтернет, збільшується з великою швидкістю. Глобальність та доступність Інтернет дозволяє отримувати та використовувати результати праці людей, що знаходяться у вільному доступі, і ділитися власними.

Часто пошук в Інтернет передбачає не лише ознайомлення з тією чи іншою інформацією з метою отримання нових знань, а й порівняння, аналіз та узагальнення даних, які знаходяться на різних вебсайтах та сторінках. Наприклад, для порівняння та аналізу цін, переліку та порівняння характеристик товарів, збору контактної інформації для залучення потенційних клієнтів тощо. Поряд з цим багато вебсайтів не дозволяють користувачам напряму зберігати дані для особистого використання. Виходом із ситуації може бути монотонне ручне копіювання потрібної інформації у власні файли та форматування їх відповідно до потреб. Такий процес, безумовно, є неефективним, оскільки вимагає багато часу. Багато великих вебсайтів, наприклад, Google, Twitter, Facebook, StackOverflow тощо, мають API, які дозволяють отримати доступ до їхніх даних у структурованому форматі. Це найкращий варіант, але є інші сайти, які не дозволяють користувачам отримувати доступ до великих обсягів даних у структурованій формі.

Вебскрапінг – це автоматизований процес отримання великого обсягу даних із вебсайтів та перетворення їх в структуровані дані [1, 2].

Програми, які здійснюють такий процес, називаються вебскраперами і здатні отримувати потрібний вміст HTML, працювати з JavaScript, фільтрувати отримані дані та виводити їх у формі готових баз даних, таблиць Excel, файлів CSV або окремих API тощо, а також оминати встановлені сайтами обмеження.

Методологія дослідження. Збирання даних у відкритих джерелах Інтернет використовується для різних бізнес-цілей, зокрема, для дослідження ринку та аналізу конкурентів, для заповнення стрічки новин, для створення навчальних наборів даних для моделей машинного навчання, для відстеження змін цін на вебсайтах електронної комерції, з метою залучення потенційних клієнтів для формування списків адрес електронної пошти та номерів телефонів тощо. [3]

Ручне видобування таких даних передбачає безпосереднє виділення та копіювання інформації з вебсторінок або ж використання інструментів розробника вебоглядача та роботу з вихідним кодом вебсторінки. В такому випадку з допомогою буфера обміну потрібні дані вставляють у файл, зберігають та використовують далі відповідно до поставленого завдання для досягнення певної мети.

Автоматизований процес вибірки потрібних даних з вебсторінок передбачає використання готових інструментів – вебскраперів (рис. 1), що, безумовно, є ефективнішим. Як правило, такі програмні засоби працюють наступним чином [3]:

- інструмент збирання програмно надсилає HTTP-запити на сервери, на яких розміщені цільові вебсторінки;
- сервери повертають вихідний код HTML для цільових сторінок;
- інструмент збирання аналізує HTML і витягує потрібні дані;
- витягнуті дані зберігаються для подальшого аналізу або обробки.

Рис. 1. Схема роботи вебскрапера

Деякі автоматизовані інструменти також надають розширені функції, наприклад, здатність обробляти файли cookie або обходити Умови використання сайту, які забороняють або обмежують копіювання вмісту.

Взагалі кажучи, для сканування вебсторінок потрібні дві частини, а саме сканер і вебскрапер. Сканер (краулер) – це алгоритм штучного інтелекту, який переглядає вебсторінки для пошуку конкретних необхідних даних, переходячи за посиланнями в Інтернеті. Вебскрапер призначений для отримання даних із вебсайту. Конструкції вебскраперів можуть дуже відрізнятися залежно від складності та обсягу проєкту, щоб вони могли швидко й точно отримувати дані. [4]

Готовими програмними рішеннями, що призначені для вебскраперу, є, наприклад, ScrapingBot [5], Octoparse [6], Xtract.io тощо.

Також можна написати власний скрипт з використанням Python, JavaScript або інших мов програмування.

Вебскрапінг – це автоматичний метод отримання великої кількості даних із вебсайтів [4]. Більшість даних, що містяться в Інтернет, є неструктурованими даними у форматі HTML. Якщо це однотипні дані, що потребують подальшого аналізу або обробки певним програмним забезпеченням, їх доречно перетворити на структуровані дані в таблицях, базах даних тощо.

Результати дослідження та їхнє обговорення. Наведемо один із варіантів реалізації плагіну для вебскрапінгу для браузерів типу Chromium.

Типовий плагін для Chrome складається із файлу manifest.json, де повністю описана структура плагіну, файлів .html, що містять розмітку спливаючого вікна (як правило, popup.html), файлів .js з головним кодом програми та, за необхідності, – файлів .css зі стилями оформлення. Формати файлів можуть відрізнятися залежно від використовуваних бібліотек [7].

Вебскрабери можуть видобувати всі дані з вебсайтів, а можуть лише конкретні дані, які потрібні користувачу. В такому випадку доречно організувати можливість потенційному користувачеві вказати, що саме його цікавить.

Плагін для вебскраперу складається з двох частин: клієнтської та серверної. Клієнтська частина забезпечує:

- відображення межі елементів розмітки при наведенні на них вказівником миші;
- реєстрацію натискання на елементи;
- збереження селекторів вибраних елементів та їх найменування;
- можливість обрати користувачем необхідні елементи серед збережених;
- виконує запит до сервера на отримання покликання доступу;
- відображає отримане посилання;
- очищає сховище збережених елементів.

Логічна складова реєструє натискання, зберігання елементів вебсторінки та генерує запити. При натисканні по елементу вебсторінки скрипт плагіну повинен записати його повний шлях з врахуванням повного селектору кожного батьківського елементу, разом з ідентифікатором та усіма класами, а не лише тег елементу, виділити його візуально та відправити повідомлення для збереження повного селектора у сховищі. Код функції, що реалізовує вибір елементу наведено на рис. 2.

```
function selectElement(path, length)
{
    let selector = getSelector(path[0])
    for(let i=1;i<path.length-2;i++)
    {
        selector = getSelector(path[i]).includes('#')?`${getSelector(path[i])} ${selector}`:selector
    }
    document.querySelectorAll(selector).forEach(el=>{
        el.classList.add('fqselected')
        fqSelected.push(el)
    })
    chrome.runtime.sendMessage({update: true, data: selector, length:length}, function(response) {
        if(chrome.runtime.lastError) console.log(chrome.runtime.lastError)
        console.log(response)
    })
}
```

Рис. 2. Код функції вибору елемента

Фрагмент коду обробника вхідних повідомлень від плагіну для ввімкнення та вимкнення режиму вибору елементів, очищення збережених представлений на рис. 3.

Серверна частина плагіну обробляє запити від плагіну; генерує покликання на вибраний контент; добуває контент вибраних елементів за покликанням. Для програмування серверної частини обрано фреймворк Express на Node.js.

```
chrome.runtime.onMessage.addListener(
    function(request, sender, sendResponse) {
      console.log(sender.tab ?
                  "from a content script:" + sender.tab.url :
                  "from the extension");
      if (request.working === true)
      {
        sendResponse({changedState: "on", length: fqLength});
        _fq_working = true;
        removeSelection()
      }
      else if(request.working === false)
      {
        sendResponse({changedState: "off"});
        _fq_working = false;
      }
      else if(request.clear === true)
      {
        sendResponse({cleared:true})
        fqLength = 0
      }
    }
  );
```

Рис. 3. Фрагмент коду обробника вхідних повідомлень

Робота сервера організована наступним чином: при отриманні запиту на генерацію покликання сервер створює пару ключ-об'єкт, де ключем виступає випадкове значення (у нашому випадку – поточний час у мілісекундах), а об'єктом – дані запиту з іменами та селекторами елементів. Створену пару сервер зберігає у локальному сховищі. У відповідь на запит відправляється покликання, що містить у собі згенерований ключ.

При переході за покликанням до серверу надходить запит із ключем та здійснюється його пошук у сховищі. У випадку знаходження створюється об'єкт класу Scrape із використанням відповідного ключа об'єкту. Клас Scrape приймає у конструкторі адресу вебсторінки та необхідні селектори, містить метод, що буде повертати об'єкт зі значеннями вибраних елементів сторінки. Результат роботи методу getData(), що видобуває потрібні дані згідно переданих селекторів та формує html-код вибраного елементу, а також його текст та атрибути «href», «src» у разі їхнього існування, відправляється у вигляді відповіді на запит. У випадку відсутності запитуваного ключа, повертається об'єкт із текстом помилки – ключ не знайдено (рис. 4).

```
app.post('/generate', async (request, response)=>{
    console.log(request.body)
    let key = Date.now().toString()
    vault.set(key, request.body);
    response.status(200).send(`${process.env.URL}:${process.env.PORT}/scrape/${key}`)
})

app.get('/scrape/:key', async (req, res)=>{
    if(vault.has(req.params.key))
    {
        let scraper = new Scrape(vault.get(req.params.key))
        res.status(200).json(await scraper.getData())
    }
    else
    {
        res.status(200).json({error:"key not found"})
    }
})
```

Рис. 4. Фрагмент коду обробників запитів серверу

Висновки. Розглянуто поняття вебскрапінгу та ефективність його використання для отримання великих об'ємів даних з вебсторінок та їх структурування.

Представлений варіант реалізації плагіну для вебскрапінгу для браузерів типу Chromium. Плагін складається з клієнтської та серверної частини.

Клієнтська частина передбачає проєктування та створення графічної складової, а саме: спливаючого вікна плагіну та можливості візуального виділення елементів вебсторінки при наведенні на них курсора миші. Серверна частина плагіну виконує аналіз запитів, генерацію покликань та здійснює вибірку елементів. Правильна робота плагіну гарантована при використанні останньої версії браузера.

Бібліографія

1. Introduction to Web Scraping - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/introduction-to-web-scraping/?ref=rp (date of access: 29.03.2023).
2. Що таке веб-скрейпінг і як він пов'язаний з проксі. Enterprise data gathering infrastructure | ASTROPROXY. URL: https://astroproxy.com/ua/blog/shho-take-veb-skreiping-i-yak-vin-povyazanii-z-proksi (дата звернення: 29.03.2023).
3. Web Scraping. Techopedia. URL: https://www.techopedia.com/definition/5212/web-scraping (date of access: 29.03.2023).
4. What is Web Scraping and How to Use It? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/ (date of access: 29.03.2023).
5. ScrapingBot • Web Scraping API - Extract HTML content. Scraping-bot.io. URL: https://www.scraping-bot.io/ (date of access: 29.03.2023).
6. Web Scraping Tool & Free Web Crawlers | Octoparse. Web Scraping Tool & Free Web Crawlers | Octoparse. URL: https://www.octoparse.com/ (date of access: 29.03.2023).
7. Chrome Extensions architecture overview - Chrome Developers. Chrome Developers. URL: https://developer.chrome.com/docs/extensions/mv3/architecture-overview/ (date of access: 29.03.2023).

References

1. Introduction to Web Scraping - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/introduction-to-web-scraping/?ref=rp (date of access: 29.03.2023).
2. Shcho take veb-skreipinh i yak vin poviazanyi z proksi. Enterprise data gathering infrastructure | ASTROPROXY. URL: https://astroproxy.com/ua/blog/shho-take-veb-skreiping-i-yak-vin-povyazanii-z-proksi (data zvernennia: 29.03.2023).
3. Web Scraping. Techopedia. URL: https://www.techopedia.com/definition/5212/web-scraping (date of access: 29.03.2023).
4. What is Web Scraping and How to Use It? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/ (date of access: 29.03.2023).
5. ScrapingBot • Web Scraping API - Extract HTML content. Scraping-bot.io. URL: https://www.scraping-bot.io/ (date of access: 29.03.2023).
6. Web Scraping Tool & Free Web Crawlers | Octoparse. Web Scraping Tool & Free Web Crawlers | Octoparse. URL: https://www.octoparse.com/ (date of access: 29.03.2023).
7. Chrome Extensions architecture overview - Chrome Developers. Chrome Developers. URL: https://developer.chrome.com/docs/extensions/mv3/architecture-overview/ (date of access: 29.03.2023).