PHP Web Scraper for Regular Expressions Summary

Monday 6th January, 2020 - 13:44

Flavio De Jesus Matias
University of Luxembourg
Email: flavio.dejesus.001@student.uni.lu

Giacomo Di Tollo
University of Luxembourg
Email: giacomo.ditollo@unive.it

***Abstract*—This document presents the bachelor semester project of Flavio De Jesus Matias under the tutoring of Giacomo di Tollo. The project consists of a web-based scraper which uses an algorithm to identify the topic of a website using regular expressions.**

1. Introduction & Project description

Web scraping plays a big role in the digital business industry nowadays. Web scraping is the use of a crawler, which can be defined as an internet bot, to retrieve information from an external website. This bot consistently browses trough the internet and searches for new specific pieces of information based on a defined topic. In other words, data scraping is the process of extracting data from a different website which is later going to be analysed for commercial or personal use.

The objective of the project is to build a web-based scraping interface which uses an algorithm defined to retrieve the occurrences of regular expressions on a website in order to describe its content.

2. Background

Firstly, the created algorithm expected a full understand of regular expressions and an advanced knowledge in the programming language PHP. This represents the scientific part of the project. To facilitate the creation of the algorithm, a PHP library was used. PHP libraries are collections of code which contain a variety of classes and functions whose goal is to solve common problems.

Lastly, this web scraper needed an user-interface to be simpler and easier to use. This interface was created using the different programming languages HTML, CSS and JavaScript. HTML and CSS give the interface the needed structure and design desired, whereas JavaScript, allows the user to create an infinite long condition, add dynamic content to the website and do AJAX calls to send the information to the server.

3. Requirements

3.1. Input & Output

In order to function correctly, the algorithm needs input given by the user. This input consists of the URL(s) and the condition. Without this input, the algorithm is unable to work since it does not have any data to analyze. To give the user the results back, the algorithm needs to output the occurrences back as a response to the interface.

3.2. User-friendly interface

One important part of the interface was to do it as user-friendly as possible. This means that the user doesn't need any to very little explanations to understand the concept. The interface was kept as simple and attractive as possible to offer a nice user experience while using the website.

4. Design & Production

4.1. Scientific Deliverable

The scientific part of the project consists of developing and using an algorithm to retrieve occurrences of regular expressions in a website. The algorithm was separated and organized into 3 different files.

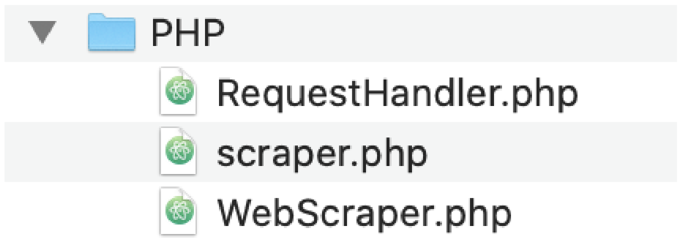
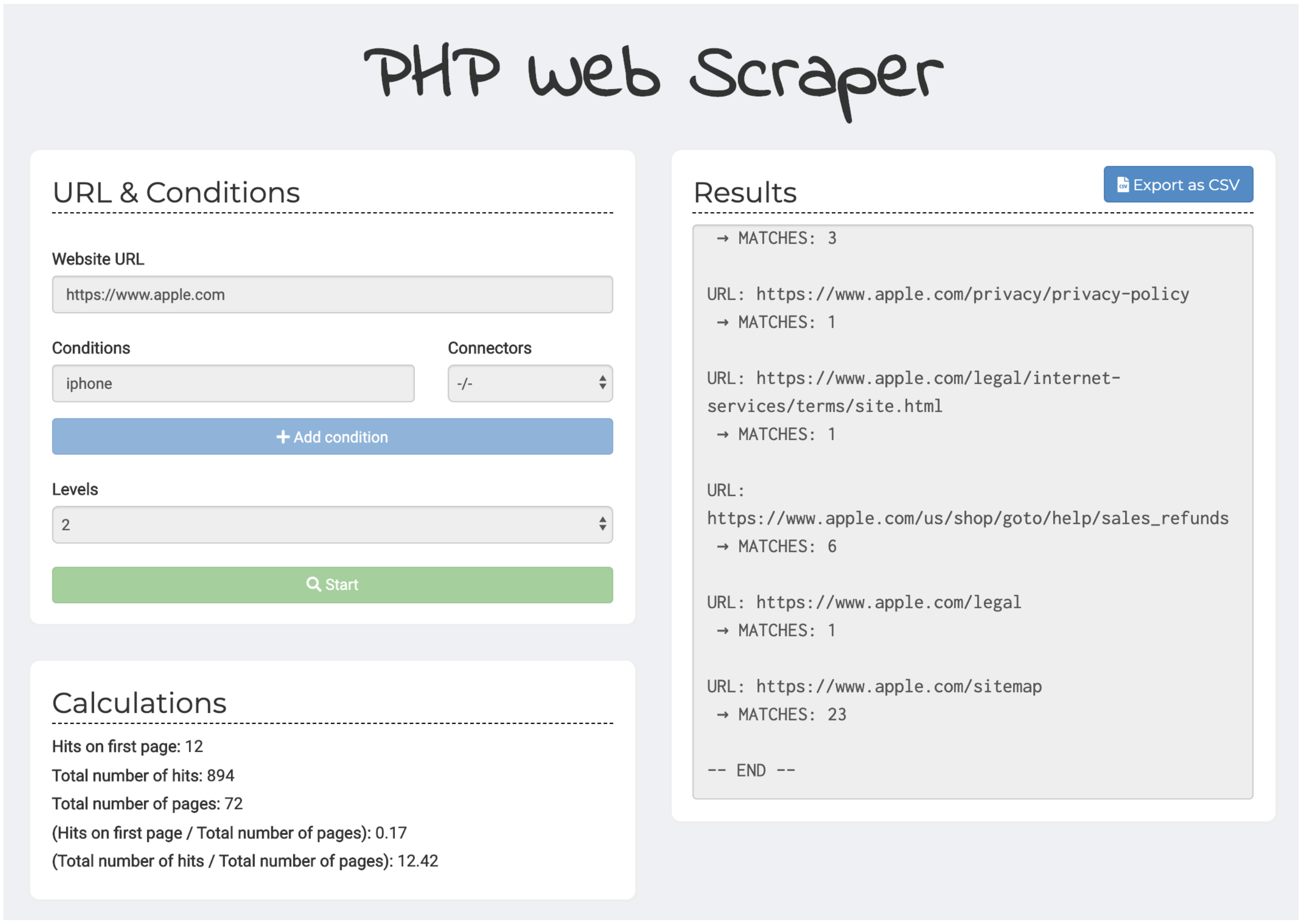


Figure 1: File structure of the server-side code showing the 3 files



https://www.apple.com

iphone -/-

2

```
 → MATCHES: 3

URL: https://www.apple.com/privacy/privacy-policy
 → MATCHES: 1

URL: https://www.apple.com/legal/internet-
services/terms/site.html
 → MATCHES: 1

URL:
https://www.apple.com/us/shop/goto/help/sales_refunds
 → MATCHES: 6

URL: https://www.apple.com/legal
 → MATCHES: 1

URL: https://www.apple.com/sitemap
 → MATCHES: 23

-- END --
```

Hits on first page: 12
Total number of hits: 894
Total number of pages: 72
(Hits on first page / Total number of pages): 0.17
(Total number of hits / Total number of pages): 12.42

Figure 2: Screenshot of the interface after a scraping example

4.1.1. WebScraper.php. This file contains the algorithm itself which was created using the free PHP library called '*Html2Text*'. This library transforms the source code of a website into formatted data which is easier to manipulate for the purpose of the project. Using a sentence recognition function, the sentences on the website are identified and stored inside a list. The identified URLs are considered sub-URLs and used to scrape the different levels. With the help of another function, the given condition by the user is modified into a condition which can be evaluated by PHP. Checking this condition on each of the identified sentences outputs the number of occurrences of this particular regular expression.

4.1.2. RequestHandler.php. This file prepares all the requests to the WebScraper-class. It will compute the output and provide the needed information for the next scrape. It is the middleware between the user and the scraping algorithm.

4.1.3. Scraper.php. This file accepts all the requests from the interface and sends the important information to the RequestHandler-class. Finally, it returns the response data provided back to the interface.

4.2. Technical Deliverable

The technical deliverable of the project consists of building a web interface to facilitate the interaction between the user and the algorithm. This interface was kept minimal and attractive to provide the best user experience possible. The page created using Bootstrap 4, which is a free framework that allows one to develop clean and responsive websites, and then split into 3 different sections.

4.2.1. URL & Conditions. This box contains all the settings the user can modify. In the beginning, the user has to choose between two different scraping types which are 'URL' and 'file'. According to the chosen type, the rest of the settings appear and the user can insert the desired URL or upload the CSV file. The user may then define the desired condition and levels and start scraping.

4.2.2. Results. The 'Results'-box is the 'console' of the page. After each request, the results are written inside this box. This allows the user to see the results in real-time and know which URL has just been scraped.

4.2.3. Calculations. When the chosen type is 'URL', this box appears at the very end of the scraping process. It gives the user multiple calculations about the complete scrape. When the chosen type is 'file', this data is included in the export file since there are multiple URLs.

5. Assessment & Conclusion

The multiple requirements set at the beginning were successfully achieved since the user can give the algorithm the desired input through the interface and also obtains the final output as a response from the interface back. The interface was also completed with success and kept very user-friendly since it is minimal and attractive at the same time.

In the future, the project could be improved in several ways such as uploading and running the code on a free web hoster. Furthermore, the project could use an user authentication system which allows users to save their scraping progress and consult it later. Finally, using a machine learning procedure the algorithm could slowly improve its sentence recognition mechanism with the objective of augmenting the scraping quality of the final product.