A Web Scraping Approach in Node.js

Shikha Mahajan, Nikhit Kumar
Information Science and Engineering
R V College of Engineering
Bangalore, India

Abstract*:*Web scraping is the process of automatically collecting information from the World Wide Web. It is a field with active developments sharing a common goal with the semantic web vision, an initiative that still requires breakthroughs in text processing, semantic understanding, and artificial intelligence and human-computer interactions. Current web scraping solutions range from the ad-hoc, requiring human effort, to fully automated systems that are able to convert entire web sites into structured information, with limitations. This paper describes a method for developing a web scraper in Node.js that locates files on a website and then decompresses and reads the files and stores their contents in a database. It mentions the modules used and the algorithm of automating the navigation of a website via links. It also describes a method of scanning the website at regular time intervals to locate newly added content with the aid of a cron job(scheduled task).

Keywords: web scraping, web mining, locating files in websites, navigating, DOM, cron job, JavaScript, Node.js, cheerio.js, decompressing files.

I. INTRODUCTION

A. DEFINITION

In its most basic form, web scraping enables a way to download web pages and then search for data in them. It often requires converting unstructured data in web pages to structured data and then storing it in a database. Web scraping can be used for indirect content searching on the internet.

B. USES OF WEB SCRAPING

The uses ofweb scraping for business and personal requirements are endless. Each business or individual has his or her own specific need for gathering data. Here are few of the common usage scenarios:

- *Gathering data from multiple sources for analysis*:
  Using a Web Scraper you can extract data from multiple websites to a single spreadsheet (or database) so that it becomes easy for you to analyze (or even visualize) the data.
- *For research*:
  A Web Scraper will help you gather structured data from multiple sources in the Internet with ease.
- *For Marketing*:
  A web scraper can be used to gather contact details of businesses or individuals from websites like yellowpages.com or linkedin.com. Details like email address, phone, website URL etc. can be easily extracted using a web scraper.

C. COMMONLY USED WEB SCRAPING TECHNIQUES

- *Human copy-and-paste*:
  In some cases even the best web-scraping technology cannot replace a human's manual examination and copy-and-paste, and sometimes this may be the only workable solution when the websites for scraping explicitly set up barriers to prevent machine automation.
- *Text grepping and regular expression matching*:
  A simple yet powerful approach to extract information from web pages can be based on the regular expression-matching or UNIX grep command facilities of programming languages.
- *HTTP programming*:
  Static and dynamic web pages can be retrieved by posting HTTP requests to the remote web server using socket programming.
- *DOM parsing*:
  By embedding a web browser, such as the Internet Explorer or the Mozilla browser control, programs can retrieve the dynamic content generated by client-side scripts. These browser controls also parse web pages into a DOM tree, based on which programs can retrieve parts of the pages.
- *Web-scraping software:*
  There are many software tools available that can be used to customize web-scraping solutions. The software may try to automatically recognize the data structure of a page or provide a recording interface that eliminates the necessity to manually write web-scraping code, or some scripting functions that can be used to extract and transform content, and database interfaces that can store the scraped data in local databases.

D. WHY NODE.JS FOR WEB SCRAPING

Node.js is a platform built on Chrome's JavaScript runtime. It uses an event-driven, non-blocking I/O model that makes itlightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. JavaScript was born as a language to be embedded in web browsers, but we

ISSN: 2278 – 7798

can now write stand-alone scripts in JavaScript that can run on a desktop computer or on a web server using Node.js.

Web scraping software till now has been written in Java, Ruby, and most popularly in Python. All modern languages provide functions to download web pages, or have extensions to do it. However, locating and isolating data in HTML pages is a challenging task. An HTML page has content, style and layouts elements all intermixed, so a non-trivial effort is required to parse and identify the interesting parts of the page.

JavaScript and libraries like jQuery can powerfully and easily manipulate the DOM inside a web browser. Therefore writing web scraping scripts in Node.js is advantageous since we can use many techniques that we know from DOM manipulation in the client-side code for the web browser. This paperdescribes simple a method to implement a web scraper in a node.js application and demonstrates its use to locate and download contents of files of a particular format from a website.

The rest of the paper is organized as follows. In Section II, the origin of the requirement to develop a web scraper in Node.js is explained. Section IIIdescribes the steps for developing a web scraper to locate and download files from a website. Section V concludes the paper and future work is mentioned in Section VI. The code snippets and the syntax of methods of various modules are given in the Appendix.

II. ORIGIN OF THE REQUIREMENT

The need for this application was encountered when a requirement arose tokeep track of logfilescontainingthe sequence of events of a critical process. The logs were uploaded on a website daily andexpired after a certain amount of time. Therefore, a system was necessary to extract useful information from the files and save it in a database for future reference. This application was developed to locate and read from log files hence providing the solution to the requirement. URL of the section of the website where the files were uploaded contained the date and time of uploading. Thelinks to the files could be located anywhere in this section of the website.The files are compressed therefore we have to decompress the files to read them. Infinite loops or links that led back to the previous page were eliminated by the use of simple if conditions and links of interest were found using regular expressions. Using an external web scraper for this application was not appropriate as the requirements were very specific. Therefore a custom web scrapingtechnique was designed using modules available in npm registry. This series of steps can be used by anyone to easily integrate web scraping functionality into their application.

III. PROCEDURE

This section details the steps in making a web scraper and searching all links in a section of a website that contains the path to files of a certain format.The application starts with a base URL, which is the section of the website that we want to search. It then obtains the contents of the webpage pointed to by the base URL and extracts all the links from it. The links are then inspected. If a link contains the URL of a required file as its href attribute then the file is decompressed and its contents are read. Otherwise the link's href value is appended to the base URL and pushed into an array. The process is then repeated by taking each value in the above formed array as the base URL. The above procedure is scheduled to run several times a day with the help of a cron job.

The following steps explain the implementation details to accomplish the above mentioned tasks:

1. Generate the basic URL of a website or the section of a website that you want to search.
2. Make a HTTP request to the URL generated in the previous step.The request is made using the GET function of Node.js' HTTP API. The functionstores the contents of the webpage pointed to by the URL in a variable. The response data does not contain complete content of the webpage as data is sent in the form of chunks. We get the data chunks by listening to the 'data' event on the response. The chunks are appended as they are received to a string variable which ultimately contains the entire content of the webpage.Refer Appendix-[A] for GET function syntax.
3. Convert the string variable containing the HTMLreceived in the previous step into a DOM tree using cheerio's LOAD method. Cheerio is an external package that generates a DOM tree and provides a subset of the jQuery function set to manipulate it. To install Node.js packages we use a package manager called npm that is installed with Node.js. This is equivalent to Ruby's gem or Python's easy_install and pip, it simplifies the download and installation of packages. Refer Appendix-[B] for cheerio's load function syntax.
4. Obtain all the links from the DOM using cheerio's `each` function. Check whether the links href attribute contain the URL of the required file format using regular expressions. Refer Appendix-[C] for cheerio's `each` function's syntax.

ISSN: 2278 – 7798

5. If the link directs to the URL of one such file, then we decompress the file and read its contents. The file is decompressed using gunzip class of node.jszlib APIand the contents of the file are stored in a variable for further processing.Refer Appendix-[D] for zlibmodule's gunzip functionsyntax.
6. If the link does not contain the URL of the required filethen we append value the link's href attribute to the base URL and push the appended value to an array.
7. The process is repeated by taking each of the links in the previously generated array as the base URL until all the links to files are found.
8. The entire process is repeated everyday with the help of npm module `node-schedule`. It uses cron patterns to determine intervals when the module runs. This way we can keep track of newly generated files.Refer Appendix-[E] for function to implement cron job using node-schedule module.

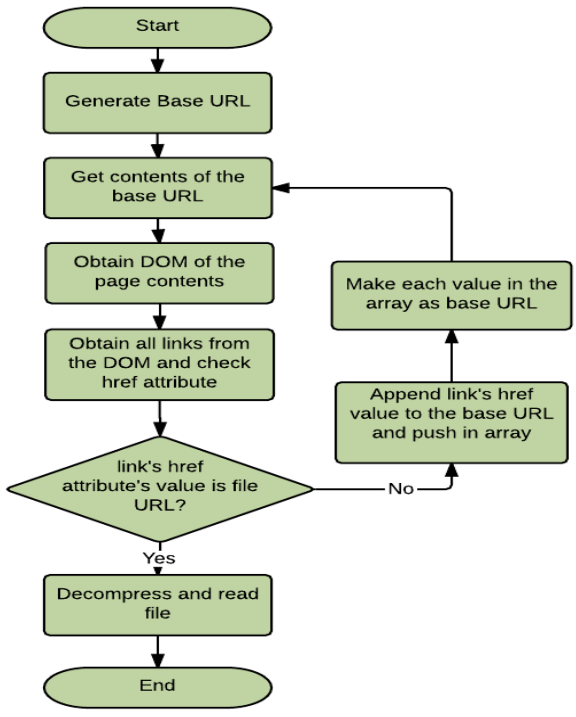

Fig 1. Flow chart to locate files in a website

IV. CONCLUSION

This paper begins with definingweb scraping and highlights its uses. It presents a method of automating navigation of websites via links, obtaining desired information from the webpage and designing a cron job to extract newly uploaded content from the website. It then explains a method of automating the process of locating files of a given format on a website with the help of web scraping. The paper hence provides an example of custom web scraping functionality in node applications with the help of npm modules

V. FUTURE WORK

This method of locating files can be applied to extract images, tables and other information from websites. The method can be enhanced to handles infinite loops while using links to traverse websites. Additionallytechniques to handle pagination in web pages can be incorporated.

APPENDIX

[A] HTTP get function to get the content of a webpage pointed to by BASEURL in a variable called 'body'.

```
http.get(BASEURL, function(res) {
res.on('data', function(chunk) {
body += chunk;
});
res.on('end', function() {
        /*define function to be done after getting the page body*/
});
})
.on('error', function(e) {
        /*define function to handle error while retrieving the page*/
});
}
```

[B] Cheerio load method to convert the HTML string stored in a variable called `body` to DOM.

```
$ = cheerio.load(body);
```

[C]cheerio's each function to get and check all the links in a webpage's DOM.

```
/*if the text of the link does not say 'Parent Directory' then append the link's text to the base URL and push into an array*/
$('a')
.each(function() {
if (($(this).text() != 'Parent Directory'){
URL1 = BASEURL + $(this).text();
url_list1.push(URL1);
}
});
```


ISSN: 2278 – 7798

[D] Decompressing the contents of a compressed file whose URL is stored in a variable called 'url'using GUNZIP method.

```
functiongetGzipped(url, callback) {
var buffer = [];
http.get(url, function(res) {
        // pipe the response into the gunzip to decompress
vargunzip = zlib.createGunzip();
res.pipe(gunzip);
gunzip.on('data', function(data) {
            // decompression chunk ready, add it to the buffer
buffer.push(data.toString())
        })
        .on("end", function() {
            // response and decompression complete, join the
    buffer and return
callback(null, buffer.join(""));
        })
 .on("error", function(e) {
callback(e);
        })
    })
    .on('error', function(e) {
callback(e)
    });}
```

[E] Function to implement cron job

```
/*cron pattern read to func the function every hour*/
    varcron_pattern = '00 * * *';
    var j = schedule.scheduleJob(cron_pattern, function(){
    // function body
    });
```

REFERENCES

[1] Sanjay Kumar Malik1 , SAM Rizvi2 "Information Extraction using Web Usage Mining, Web Scrapping and Semantic Annotation" *2011 International Conference on Computational Intelligence and Communication Systems.*

[2] Richard Baron Penman, Timothy Baldwin, David Martinez "Web Scraping Made Simple with SiteScraper"

[3] GuChengjian, Huang Lucheng, "Web Mining in Technology Management, 2008 *International Seminar on Business and Information Management*978-0-7695-3560- 9/08, 2008 IEEEDOI 10.1109/ISBIM.2008.127.

[4] Clara Sacramento, Ana C. R. Paiva "Web Application Model Generation through Reverse Engineering and UI Pattern Inferring" 2014 *9th International Conference on the Quality of Information and Communications Technology.*

[5]PranamKolari And Anupam Joshi, "Web mining: Research And Practice", *Web Engineering*, 1521-9615/04, 2004 IEEE.

[6] Giovanni Grasso, Tim Furche, and Christian Schallhart "Effective Web Scraping with OXPath" *WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.*

[7] Kosala, R. and Blockeel, H. (2000). Web mining research: *A survey. ACM SIGKDD Explorations Newsletter, 2(1):1–15.*

[8] "Web Scraping", (Wikipedia), [online] 2015, http://en.wikipedia.org/wiki/Web_scraping (Accessed: 27 March 2015).

[9] VasaniKrunal A. "*Content Evocation Using Web Scraping Content Evocation Using Web Scraping*",*IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN*: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 3, Ver. IX (May-Jun. 2014), PP 54-60

ISSN: 2278 – 7798