Google spreadsheet template: Scraping_Loop

Author:
Juan Elosua (@jjelosua)

Access:
Once logged in your google account

1. From the google drive templates search: here
2. Directly creating a new copy on your drive using this link.

Context:
The tools that exist to perform web scraping without having coding knowledge are really scarce, slow and not user-friendly right now (import.io is doing a good job and it is worth keeping an eye on them). So every time I am facing trainees with no coding knowledge I find the options they have available are very limited.

The option out there that I like the most is using the import functions in google spreadsheets (ImportXML or ImportHTML), they are easy to use and really straight to the point….but you need to go one URL at a time, that means, page by page.

Sometimes you need to get all the results of one search but the site uses pagination so it does not show all the results on the first page, is shows only the first 10, 20 or 50 items. This means you need to go to the next page use the importXML function with that URL to get the next results, and so on until you reach the end of your dataset.

The other day a journalist friend asked me to help her scrape data from a really simple site that had the results spread through 170 pages. She knew how to use ImportXML and ImportHTML but she would need to manually enter 170 importXML functions to get all the desired dataset (by the way Google limits to 50 the number of importXMLs per spreadsheet)

Here is when Scraping_Loop started to develop in my head...I realised that she was so close to being able to scrape the data on her own….but she needed just to make a loop...so I said, Let's give her a loop.

Sometimes (some of you may be familiar with this) when you develop things trying to cover as many use cases as possible, things start to get complicated so I've tried to keep the functionality limited so that the complexity did not shoot up…

Tool:

The Scraping_Loop tool is basically a google spreadsheet template that is linked to a google apps script that performs the loop based on the parameters of the google spreadsheet.

The loop is only thought for changing a **numeric parameter** such as a page number or similar, after retrieving the data is appends the new data to the contents of a new sheet called "*Data*" and proceeds to the next number.

As I have already told you, I have tried to balance between trying to be flexible on the one hand and simple on the other. I have exposed some of the parameters to tweak the execution of the script but I also gave them default values so that you only need to worry about them when you are already familiar with the tool and want to fine tune it. For example, all the parameters inside the "*Advanced*" sheet belong to this category..

Permissions
The script needs some authorizations in order to be able to be executed:

- View and Manage Spreadsheets
    - To create new sheets with the scraped data.
- Execute while you are not present
    - To allow time-based triggers to continue scraping and thus leverage the maximum execution time of 6 minutes imposed by google per run.
- Send emails as you
    - Only to notify the result of the script (optional)
- Connect to an external service
    - This authorization is needed to verify that the URL you are scraping returns a valid HTTP response code.

The template has 3 sheets initially:

- README: Description of the tool and use case examples.
- Advanced: Advanced execution parameters that you can tweak.
- **Parameters**: Here is where all the magic happens. We need to enter the input parameters and the script will inform us of the status of the scraping progress. Let's focus on this sheet with some more details.

Parameters Sheet (MAIN):

- Contains the input parameters:
  - Parameters to form the URL from which we are going to scrape data with two possible options:
    - SITE+PREFIX+PARAM_NAME+"="+VALUE+SUFFIX
    - SITE+PREFIX+VALUE+SUFFIX (if param_name empty)
  - A parameter that defines what information you want for each page (XPATH syntax)
  - Email address to notify the script result.

- Contains the output parameters that inform you about the script execution status:
  - Number of runs of the script
  - Number of data elements scraped
  - Time and date of the last execution
  - Last value of the loop that was correctly executed
    - To allow the script to continue in the next execution.
  - Status: Running, Waiting for next execution, Error, Finished.

- To access to the script functionality you can use the "*Ad-hoc*" menu called "*Scraping*" or the buttons inside the "*Parameters*" sheet:
  1. **Test Scraping parameters**: You will always start with this script to test if the input parameters (URL + XPATH) are correct and that the execution of a single importXML function works.

  2. **Scrape Data**: Once the Test has succeeded. You can launch the complete scraping process using this function. The script scrapes the data block by block not to overpass the maximum execution time allowed by google (6 minutes), when it reaches the end of a block it creates a time-based trigger to continue where it stopped. The script repeats this workflow until the end of the desired dataset.

  3. **Reset Project**: You will use this function when you want to restart from zero…"something" has gone wrong and you want to start fresh again.
     a. **Warning!!** This function will delete the "*Data*" sheet.

  4. **Reset Project keeping data**: You will use this function is you want to use the same spreadsheet to launch a new scraping process keeping the data sheet. You will need to rename the "Data" sheet so that the next script execution does not append the results to the existing ones.

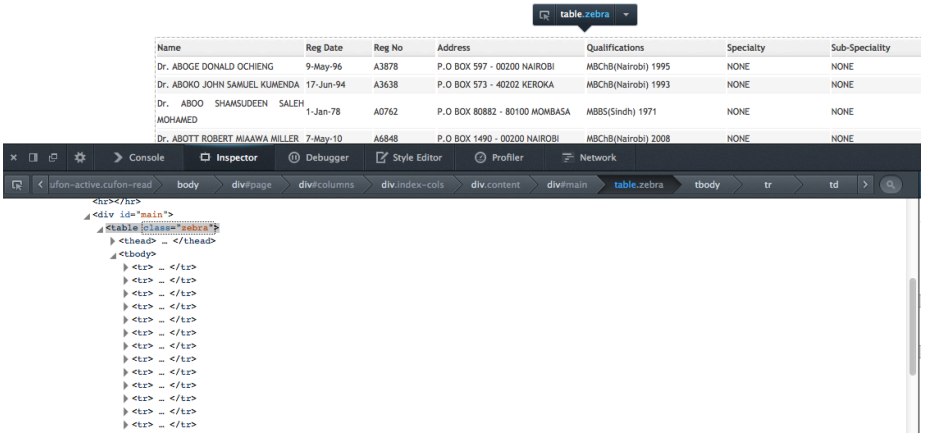
Ok, sounds good but...how do I use it?:

1. Once you see a web site with data that you want to scrape and that has many similar pages with information and you want all of them. Scraping_Loop may help you automate your scraping process.
    a. Example: http://medicalboard.co.ke/online-services/retention/

2. You need to analyze the URL of the pages to check how they vary from one page of results to the next
    a. You can copy two or three URLs inside a text editor and compare them to see the differences between them.
        i. **Hint**: Normally the first page does not have the parameter by default you need to go to the second page and come back to the first one to see it...test it in the example web.
    b. If they differ only by a numeric value then **Scraping_Loop** can help you.
        i. http://medicalboard.co.ke/online-services/retention/?currpage=**1**
        ii. http://medicalboard.co.ke/online-services/retention/?currpage=**2**
        iii. http://medicalboard.co.ke/online-services/retention/?currpage=**3**

3. Next you need to decompose the URL in parts and enter them in the input parameters:
    a. SITE: http://medicalboard.co.ke/
    b. PREFIX URL: online-services/retention/?
    c. VALUE PARAM NAME: currpage
    d. SUFFIX URL: *(empty)*
    e. START VALUE: 1
        i. What number to start the loop (included)
    f. STEP VALUE: 1
        i. How much do I want to increase the value on each call of the importXML function
    g. STOP VALUE: 170
        i. What number is the last one I want (included)

4. You have solved the navigation loop but you need to tell the ImportXML function where is the data you need to extract from each HTML page, you will use XPATH to do that.
    a. XPATH tutorial
    b. You can use the browser Web Inspector to determine the appropriate XPATH to our desired data.
        i. XPATH: //table[@class='zebra']//tr



5. Once you have entered the URL input parameters and the XPATH parameter you are ready to use the "**Test Scraping Parameters**" script function to check if a single execution returns the results you are expecting. This function will create a "*Test*" sheet with the description of the test and the result.

6. If you see that the test is not successful you need to verify a couple of things and keep trying until you succeed.
    a. If you see in the test that the URL does not bring any results and when you copy it and paste it on a browser you see that it works maybe the site needs **cookies**. You can not scrape those sites with this tool.
        i. Google spreadsheet function ImportXML does not work with cookies.
    b. If the error is that the XPATH is not returning anything you can use importHTML in a new sheet to verify that the URL is not the problem, if you get results you need to keep working on the XPATH syntax until you get the results...it will work!!
        i. **Hint**: Begin with a simple XPATH and continue until we get the desired results.

7. Once the test works we can launch the complete scraping through the "Scrape Data" script function.
    a. The execution can take a long time to complete depending on the number of pages to scrape.
    b. By default the script will scrape 10 pages and then create a time-based trigger to execute again in 5 minute time.
        i. In the "*Advanced*" sheet we can fine-tune this.
    c. We can close the spreadsheet once we have launched the script….we can check from time to time the progress of the execution through the Output parameters inside the "*Parameters*" sheet.
    d. We can enter an email address in the input parameters, If we do so an email will be sent with the final result of the script (Successful or Error)

8. Nothing more to say, just hope you find it useful.

Known Limitations:

- The tool is in beta test so if you find a bug please contact me in order to tackle it.

- This script largely depends on network requests/responses sometimes if the site is offline the script will get errors and stop.

- Since the script resides inside a google spreadsheet all the limits of a google spreadsheet apply.