Python Project: Build a web crawler

Start Date: Sunday, 16-08-2020 07:00 PM
End Date: Sunday, 23-08-2020 07:00 PM

Project Head: Ananth SNC (https://t.me/a1ananth)
Organised by: Flinkhub (https://flinkhub.com)

Telegram Group link: https://t.me/pythonwebcrawler

Problem statement

Build a spiderbot (web scraper) that continuously runs in the background and recursively scrapes all links it can find

---

Problem Details

The Root URL for this project will be https://flinkhub.com . The Root URL must be manually entered into the database before the process starts.

As soon as the process starts, it should check the "links" table for pending links to scrape. It should scrape all these links, extract all **valid links** (through <a> tags) from each of these pages, and save them in the database. It should also save the response HTML on the disk as a file with a random file name. This is considered as one scraping cycle. The process will then start the next cycle of scraping.

The process should sleep for 5 seconds between each cycle of scraping.

The process must be written inside an infinite loop i.e. it should never end and should start a new cycle 5 seconds after last cycle is complete.

The process should not scrape links that are scraped already in the last 24 hours.

The process should implement multithreading with 5 threads running parallelly i.e. it should be crawling up to 5 links in parallel.

The process should never crash and kill itself due to run-time errors. All run-time errors must be handled properly to keep the process running.

If all links have been scraped in last 24 hours and there are no new links, the process should just print "All links crawled" and consider that cycle as completed.

For the confines of this project, we will limit maximum number of links to 5000. i.e. Once the database has got 5000 links, the process should just print "Maximum limit reached" and consider that cycle as completed.

Database

You are free to choose any database you like. We suggest using MongoDB as the integration with Python is pretty straight forward (using pymongo).

Data Structure

There is only one table (or collection) in this entire application.

1. Links – This table will store all web page links
   a. Link – The link itself
   b. Source Link – The source page from which this link was first extracted
   c. Is Crawled – True/False – indicates whether this links has been crawled
   d. Last Crawl Dt – Last date when this link was crawled
   e. Response Status – HTTP Status code for the link
   f. Content type – HTTP Response Content type header for the link
   g. Content length – Size of HTTP Response (usually sent in headers)
   h. File path – Path to the file stored on disk containing the HTML response
   i. Created at – Date at which this link was first inserted

---

Tools you will need

- Python 3 Interpreter – You can do this project using Python 3.7 or 3.8
- IDE / Code Editor – As per your choice
- Robo 3T – If you are using Mongo database
- MySQL Workbench – if you are using MySQL database

Open source modules

You are allowed to use the following open source modules:

1. requests
2. beautifulsoup4
3. Any Database ORM / Connectors (like pymongo / sqlalchemy)

Submission Details

You must submit your code as a public git repository on github.com . The name of the repository should be “python-web-crawler”.

We will share a link to a form in the group in which you can submit the link to your repository.

Notes

- All variables related to application configuration must be put in a separate file “cfg.py”
- HTML Response files must be saved with the correct extension based on content type
- Certificate will be provided to students who submit working code that satisfies ALL problem details without fail
- Do not check in any sensitive information like usernames and passwords into your Github repositories