

# Rapport TP Web Scraping

## Sur le site Wikipédia

*Cécile CESA*

*Lucie GUILLAUD SAUMUR*

*Valentin PACHURKA*

# Sommaire

I – Introduction .....	3
II – Cahier des charges .....	4
III – Architecture des données .....	6
1 - Collection Journal .....	6
2 - Collection Content .....	6
3 - Collection link .....	7
IV - Architecture du code .....	8
1 - Les imports .....	8
2 - La classe Scraper .....	8
3 – Gestion des arguments : argparse .....	9
V - Résultats .....	10
1 - Base de données : scraping_db .....	10
2 - Les collections .....	10
Content .....	11
Journal .....	12
Link .....	13
VI - Améliorations en cours .....	14




## I – Introduction

Nous avons réalisé un projet de web scraping dans le cadre de notre formation Data Analyst chez DIGINAMIC Lyon. Ce projet s'inscrit dans le module "Projet Python avec MongoDB" d'une durée de 4 jours. Le livrable du projet se compose d'un programme python fonctionnel, d'un support de présentation et de ce rapport.

Nous allons revenir sur les différentes étapes du projet avec le cahier des charges du client et les différents éléments qui ont pu être traités à ce jour. De plus, nous verrons l'architecture des données et la manière dont nous avons stockés les données dans MongoDB. Par la suite, nous présenterons l'architecture du code avec les différents modules et classe qui compose le programme. Nous verrons, ensuite, les résultats obtenus par le programme avec une url test et un nombre de documents à récupérer. Pour finir, nous présenterons les différents axes d'améliorations du projet.

## II – Cahier des charges

### Nomenclature des différents points

-  : les éléments traités
-  : les éléments en-cours de traitement
-  : les éléments pas encore traités

### Eléments centraux :

- Modules Python (**pymongo, requests, BeautifulSoup, time, urllib.parse, time, pickle**)

### Scraping Web:

- Récupérer une page web à une URL donnée et récupérer tous les liens de la page web
- Recommencer récursivement sur ces nouveaux liens
- La même page web ne peut pas être récupérée plusieurs fois pour une même session de scraping
- Limiter le scope (le site) se limiter à un certain nom de domaine ou sous domaine ou à un certain préfixe de répertoires dans l'url
- **Plusieurs sessions en parallèle pour deux sites distincts ou en séquence pour le même site**
- Limite (à choisir en liste de commande) du nombre max d'URL distinctes à récupérer, pour une session donnée
- **Réutiliser les cookies envoyés par le serveur pour les requêtes des pages liées**
- Gérer les erreurs réessayer à 60 seconde d'intervalle. Recommencer max 10 fois.
  - Insertion dans le journal

### Fonctionnement distribué :

- **Mutualiser les ressources (machine et réseau) de plusieurs machines**
- **Répartir la charge**
- Utiliser liste d'attente d'URL scapées dans une base de données
- Conserver une trace des évènements

### Tolérance aux pannes :

- **Détecter un scrapeur en panne**
- **Reprendre le traitement**

### Prétraitement du contenu :

- Extraire les titres <title>, <h1>, <h2>
- Extraire les emphases <b>, <strong>, <em>

### Interface :

- Programme en ligne de commande (**argparse**)
- **API flask**

### *Planning du cahier des charges*

Lundi 03/07		Mardi 04/07		Mercredi 05/07		Jeudi 06/07		Vendredi 07/07	
AM	PM	AM	PM	AM	PM	AM	PM	AM	PM
Scraping web	Scraping web	Scraping web	Scraping web	Fonctionnement distribué		Livrable intermédiaire	Entretien avec le client		Livraison finale
Prétraitement du contenu	Prétraitement du contenu	Mise en place de fonction	Fonctionnement distribué	Architecture MongoDB		Documentation sur architecture	Amélioration du programme		Présentation
						Première version fonctionnelle	Préparation ppt et rapport final		

## III – Architecture des données

Pour l'architecture de notre base de données, nous avons choisi de créer 3 collections afin de stocker toutes les infos nécessaires au web scraping.

### 1 - Collection Journal

La collection "journal" est utilisée pour enregistrer des informations sur le processus de scraping. Une partie des documents dans cette collection représente une URL visitée par le scraper. Voici les champs clés utilisés dans les documents de cette collection :

**"\_id"** : L'URL visitée par le scraper.

**"début"** : L'heure de début du scraping.

**"fin"** : L'heure de fin du scraping.

Cette collection comprend également des documents qui référence les sessions. Voici les champs clés utilisés dans ces documents de cette collection :

**"\_id"** : L'URL visitée par le scraper et qui a démarré la session.

**"début\_session"** : L'heure de début de la session de scraping.

**"fin\_session"** : L'heure de fin de la session de scraping.

Le but de cette collection est de garder une trace du début et de la fin de chaque session de scraping et de fournir des informations sur les URLs visitées (et quand elles l'ont été).

### 2 - Collection Content

La collection "content" est utilisée pour stocker les métadonnées des pages web extraites par le scraper. Chaque document dans cette collection représente une page web. Voici les champs clés utilisés dans les documents de cette collection :

**"\_id"** : L'URL de la page web.

**"html"** : Le contenu HTML complet de la page web.

**"titles"** : Une liste des titres trouvés sur la page (balises <title>, <h1>, <h2>, etc.).

**"emphasis"** : Une liste du contenu des balises d'emphase (balises <b>, <strong>, <em>).

Ces informations permettent de capturer les éléments clés des pages visitées, tels que les titres et le contenu mis en emphase.

### 3 - Collection link

La collection "link" est utilisée pour enregistrer les liens découverts lors du scraping. Chaque document dans cette collection représente un lien URL. Voici les champs clés utilisés dans les documents de cette collection :

**"\_id"** : L'URL du lien.

**"status"** : L'état du lien (soit "to do", "in process", "done")

**"cookies"** : tous les cookies de la page url.

Cette collection est utilisée pour garder une trace des liens à traiter, de leur état (ex : est-ce qu'il y a eu une panne ?) et pour éviter de traiter plusieurs fois le même lien.

Pour conclure, le programme utilise la collection "journal" pour enregistrer des informations sur le processus de scraping, la collection "content" pour stocker les métadonnées des pages web extraites, et la collection "link" pour gérer les liens à traiter. Cette architecture de données permet de stocker et d'organiser les informations extraites de manière efficace pour une utilisation ultérieure ou une analyse plus poussée.

## IV - Architecture du code

Nous avons créé un script de scraping web écrit en Python. Il utilise différentes bibliothèques telles que **requests**, **bs4**, **pymongo**, **urllib.parse**, **datetime**, **argparse**, **time** et **pickle** pour extraire des informations à partir de pages web et les enregistrer dans une base de données MongoDB.

### 1 - Les imports

- **import requests** : utilisé pour envoyer des requêtes HTTP et récupérer le contenu des pages web.
- **from bs4 import BeautifulSoup** : fournit des fonctionnalités pour extraire des données HTML et XML.
- **from pymongo import MongoClient** : permet d'interagir avec une base de données MongoDB.
- **from urllib.parse import urljoin, urldefrag, urlparse** : utilisé pour manipuler et normaliser les URL.
- **import time** : fournit des fonctionnalités pour la gestion du temps.
- **import datetime** : pour manipuler les dates
- **Import pickle** : permet de sauvegarder un fichier au format binaire

### 2 - La classe Scraper

Le constructeur **\_\_init\_\_** : initialise les attributs de la classe, notamment les URLs de départ, le nombre de documents à récupérer par url, la limite de domaine et les connexions à la base de données MongoDB.

La méthode **scrape\_website** : c'est la méthode principale du scraping. Elle est utilisée pour démarrer le processus de scraping. Elle parcourt les URLs de départ, appelle la méthode **\_scrape\_link** pour chaque URL, et itère jusqu'à ce qu'un certain nombre de documents (**nb\_doc**) aient été collectés.

La méthode privée **\_scrape\_link** : cette méthode effectue le scraping d'une URL donnée. Elle utilise la méthode **retry\_request** pour effectuer une requête HTTP à l'URL, puis utilise **BeautifulSoup** pour analyser le contenu HTML de la réponse. Ensuite, elle extrait les liens de la page à l'aide de la méthode **\_get\_url\_links** et insère les liens dans une collection MongoDB à l'aide de la méthode **\_insert\_links**. Elle insère également les métadonnées de la



page (titre, contenu d'emphase, etc.) dans une autre collection MongoDB à l'aide de la méthode **\_insert\_metadata**.

La méthode **retry\_request** est utilisée pour effectuer une requête HTTP avec une tentative de réessai en cas d'échec. Elle réessaie la requête un certain nombre de fois (**max\_retries**) avec un intervalle de temps (**retry\_interval**) entre chaque tentative. Si la requête échoue à chaque fois, elle retourne **None**.

La méthode **\_get\_url\_links** extrait les liens des balises **<a>** dans le contenu HTML de la page. Elle vérifie que les liens extraits appartiennent au même domaine limité défini par **self.domain\_limit** (dans cet exemple, 'fr.wikipedia.org').

Les méthodes **\_insert\_links**, **\_insert\_metadata** et **\_insert\_journal** sont utilisées pour insérer les données extraites dans les collections MongoDB correspondantes.

Enfin, le code utilise **argparse** pour analyser les arguments de ligne de commande. Il attend deux arguments : les URLs de départ (**url**) et le nombre de documents à collecter (**count**). Il crée une instance de la classe **Scraper** avec ces arguments et appelle la méthode **scrape\_website**.

### 3 – Gestion des arguments : argparse

En résumé, ce code représente un scraper web qui récupère les liens et les métadonnées des pages web à partir d'une liste d'URLs de départ spécifiée, puis les enregistre dans une base de données MongoDB. Après une première session de scraping, si une nouvelle session est lancée alors le programme reprendra là où il s'est arrêté (s'il reste des liens à traiter et que l'on souhaite récupérer d'autres documents dans la base de données. L'utilisateur peut choisir un nombre de documents par url en l'indiquant en ligne de commande.

Exemple : python web\_scraping.py <https://fr.wikipedia.org/wiki/France> 1337

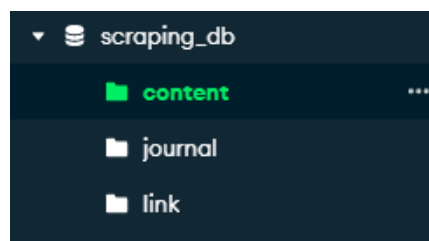
Dans cet exemple, on souhaite récupérer 1337 documents provenant de la page Wikipédia "France".

## V - Résultats

Voici un visuel dans MongoDB du rendu de notre code :

### 1 - Base de données : `scraping_db`

Cette base de données sous MongoDB, comprend les 3 collections citées précédemment (**content**, **journal**, **link**). Voici un visuel de cette base de données :



### 2 - Les collections

Voici un visuel des 3 collections dans la base de données :

+ Create collection

Refresh

View

☰

⌵

Sort by

Collection Name

▼

🗑

content

Storage size:

Documents:

Avg. document size:

Indexes:

Total index size:

3.08 MB

30

327.62 kB

1

20.48 kB

journal

Storage size:

Documents:

Avg. document size:

Indexes:

Total index size:

20.48 kB

31

131.00 B

1

20.48 kB

link

Storage size:

Documents:

Avg. document size:

Indexes:

Total index size:

663.55 kB

4.8 K

1.28 kB

1


159.74 kB







## Content

Nous avons le champ **\_id** qui correspond à l'URL de la page traitée, le champ **html**, le champ **titles** (balises titles) et le champ **emphasis** (balises emphasis). Voici un visuel de la collection :

**scraping\_db.content** 30 DOCUMENTS 1 INDEXES

[Documents](#) [Aggregations](#) [Schema](#) [Indexes](#) [Validation](#)

[Filter](#)  Type a query: { field: 'value' } [Explain](#) [Reset](#) [Find](#) [Options](#)

[ADD DATA](#) [EXPORT DATA](#) 1 - 20 of 30      

<pre>_id: "https://fr.wikipedia.org/wiki/France" html: "&lt;!DOCTYPE html&gt;  &lt;html class=\"client-nojs vector-feature-language-in-h_\"   titles: Array (82)   emphasis: Array (9)</pre>
<pre>_id: "https://fr.wikipedia.org/w/index.php?title=Atlantic_Treaty_Association_\" html: "&lt;!DOCTYPE html&gt;  &lt;html class=\"client-nojs vector-feature-language-in-h_\"   titles: Array (2)   emphasis: Array (empty)</pre>
<pre>_id: "https://fr.wikipedia.org/w/index.php?title=Fichier:Flag_of_France_(179_\" html: "&lt;!DOCTYPE html&gt;  &lt;html class=\"client-nojs vector-feature-language-in-h_\"   titles: Array (21)   emphasis: Array (65)</pre>
<pre>_id: "https://fr.wikipedia.org/w/index.php?title=Fichier:Mapadefrancia.svg&amp;L_\" html: "&lt;!DOCTYPE html&gt;  &lt;html class=\"client-nojs vector-feature-language-in-h_\"   titles: Array (28)   emphasis: Array (6)</pre>

## Journal

Dans cette collection, nous avons enregistré dans un champ **\_id** l'url de la page, un champ **début\_session** et **fin\_session** pour le document correspondant à la session. Pour les documents issus de la session (urls issus de l'url de départ), il y a également un champ **\_id** pour l'url de la page, un champ **début** et un champ **fin** qui correspondent au début et fin de scraping du document. Voici un visuel de la collection :

## scraping\_db.journal

30 1  
DOCUMENTS INDEXES

[Documents](#) [Aggregations](#) [Schema](#) [Indexes](#) [Validation](#)

Filter ⓘ ⓘ Type a query: { field: 'value' }

Explain Reset Find ↻ Options ▶

ADD DATA EXPORT DATA

1 - 20 of 30 ↺ < > ⋮ ⚙ | ⌂

```
_id: "https://fr.wikipedia.org/wiki/France"
end_session: 2023-07-07T15:47:45.227+00:00
start_session: 2023-07-07T15:47:32.306+00:00
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=Atlantic_Treaty_Association-"
end: 2023-07-07T15:47:32.669+00:00
start: 2023-07-07T15:47:32.306+00:00
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=Fichier:Flag_of_France_(179-"
end: 2023-07-07T15:47:32.850+00:00
start: 2023-07-07T15:47:32.306+00:00
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=Fichier:Mapadefrancia.svg&L="
end: 2023-07-07T15:47:33.000+00:00
start: 2023-07-07T15:47:32.306+00:00
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=France&action=edit"
end: 2023-07-07T15:47:34.342+00:00
start: 2023-07-07T15:47:32.306+00:00
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=France&action=edit&section=_"
end: 2023-07-07T15:47:35.084+00:00
start: 2023-07-07T15:47:32.306+00:00
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=France&action=history"
end: 2023-07-07T15:47:35.324+00:00
start: 2023-07-07T15:47:32.306+00:00
```

## Link

Dans la collection **link**, nous avons un champ **\_id** de l'url de la page, **status** ("to do", "in process" ou "done") et la liste de **cookies** récupérés. Voici un visuel de la collection :

### scraping\_db.link

5.5k 1  
DOCUMENTS INDEXES

Documents Aggregations Schema Indexes Validation

Filter  Type a query: { field: 'value' }

Explain Reset Find  Options 

 ADD DATA 

 EXPORT DATA 

1 - 20 of 5518      

```
_id: "https://fr.wikipedia.org/w/index.php?title=Atlantic_Treaty_Association..."
status: "done"
cookies: BinData(0, 'gASV0wQAAAAAACMEHJlcXVlc3RzLmNvb2tpZXOUjBFSZXF1ZXN0c0Nvb2tpZUphcpSTlCmBlH2UKIwHX3BvbG1jeZSMDmh0dHAU...')
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=Fichier:Flag_of_France_(179..."
status: "done"
cookies: BinData(0, 'gASV0wQAAAAAACMEHJlcXVlc3RzLmNvb2tpZXOUjBFSZXF1ZXN0c0Nvb2tpZUphcpSTlCmBlH2UKIwHX3BvbG1jeZSMDmh0dHAU...')
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=Fichier:Mapadefrancia.svg&l..."
status: "done"
cookies: BinData(0, 'gASV0wQAAAAAACMEHJlcXVlc3RzLmNvb2tpZXOUjBFSZXF1ZXN0c0Nvb2tpZUphcpSTlCmBlH2UKIwHX3BvbG1jeZSMDmh0dHAU...')
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=France&action=edit"
status: "done"
cookies: BinData(0, 'gASV0wQAAAAAACMEHJlcXVlc3RzLmNvb2tpZXOUjBFSZXF1ZXN0c0Nvb2tpZUphcpSTlCmBlH2UKIwHX3BvbG1jeZSMDmh0dHAU...')
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=France&action=edit&section=..."
status: "done"
cookies: BinData(0, 'gASV0wQAAAAAACMEHJlcXVlc3RzLmNvb2tpZXOUjBFSZXF1ZXN0c0Nvb2tpZUphcpSTlCmBlH2UKIwHX3BvbG1jeZSMDmh0dHAU...')
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=France&action=history"
status: "done"
cookies: BinData(0, 'gASV0wQAAAAAACMEHJlcXVlc3RzLmNvb2tpZXOUjBFSZXF1ZXN0c0Nvb2tpZUphcpSTlCmBlH2UKIwHX3BvbG1jeZSMDmh0dHAU...')
```

```
_id: "https://fr.wikipedia.org/w/index.php?title=France&action=info"
status: "done"
cookies: BinData(0, 'gASV0wQAAAAAACMEHJlcXVlc3RzLmNvb2tpZXOUjBFSZXF1ZXN0c0Nvb2tpZUphcpSTlCmBlH2UKIwHX3BvbG1jeZSMDmh0dHAU...')
```

## VI - Améliorations en cours

Nous avons pu répondre à une partie des attentes client dans le temps qui nous été imparti. Les prochains axes à développer concernent :

- L'interface API **flask** : pour le moment le programme fonction en ligne de commande avec la gestion des arguments (**argparse**).
- **Distribution** des ressources entre des machines/serveurs.
- **Parallélisation** des sessions : pour le moment cela fonctionne en séquence. Une session après l'autre.
- Réutilisation des **cookies (en cours)**.
- Utilisable en dehors de Wikipédia.