REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERIE DE TIZI OUZOU

Faculté de Génie Electrique et d'Informatique

Département : Informatique

MEMOIRE DE MASTER

DOMAINE : MATHEMATIQUE ET INFORMATIQUE

SPECIALITE : SYSTEME INFORMATIQUE

THÈME

Etude comparative de web scraping :

Etude des outils scrapy et Apache-nutch

Présenté devant le jury :

Président : M A.AIT EL HADJ .

Examinatrice : Mme Z.AIT YAKOUB.

Promotrice :Mme F.AIT EL HADJ.

Réalisé par :

SLIMANI Hayet.

YAICI Souhila.

Promotion: 2018/2019

Résumé

L’objectif de notre travail est d’Acquérir les notions théoriques et pratiques nécessaires à la mise en œuvre d’un meilleur outil d’acquisition automatisé de données sur le web.

Notre projet se concentre particulièrement sur deux outils de web scraping : Scrapy et apach_nutch . Chaque outil fait l’objet d’une présentation théorique et d’exemples pratiques de programmation. Afin de pouvoir les comparer sur tous les aspects et proposer le meilleur outil qui récapitule, regroupe et synthétise les données selon le besoin de chacun de nous.

Abstract

The objective of our work is to acquire the notions of thermology and bratics necessery to the implementation of the better tool of automated acquisition of data on the web.

Our project focuses particularly on two web scraping : Scrapy and apach_nutch ,each tool is the subject of theoretical presentation and practical examples of programming . in order to be able to compare them on all aspects and to propose the best tool which recapitulates, gathers ans synthesizes the data according to the need of each one of us.

Remerciements

je remercie tout d’abord Dieu tout –puissant de nous avoir donné le courage, la santé et la volonté pour réaliser ce modeste travail .

je remercie tout particulièrement mon encadreur, Mme Ait el hadj Fatiha , pour son aide et sa qualité d’encadrement .

j’adresse mes sincères remerciements à mes parents qui m’ont tant encouragée et soutenue dés le début de ce travail.

je remercie aussi les membres de jury pour l’honneur qu’ils m’accordent en acceptant d’évaluer ce mémoire.

Mes remerciements les plus vifs à toutes les personnes qui ont contribué de prés ou de loin à l’élaboration de ce mémoire.

Dédicaces

A nos chers parents,

A nos frères et sœurs,

A nos familles,

A nos amis(es).

Tables des matières

Sommaire

CHAPITRE II: étude et implémentation de scrapy

CHAPITRE III : étude et implémentation de apache-nutch

CHAPITRE IV : Etude comparative de scapy et apache- nutch.

Table des figures

Chapitre1 :

Chapitre2 :

Chapitre3 :

Chapitre4 :

Introduction générale :

Cadre général et objectifs

Etant donné que les données qui circulent sur le web se multiplient rapidement et sont dispersées dans tout l'espace numérique sous forme d'actualités, d'articles, de messages sur les réseaux sociaux, d'images sur instagram, etc., ainsi les entreprises dépendent énormément des données qui peuvent être source de nouvelles opportunités pour la prise de décision, d'amélioration de services et d'augmentation de profits. Le problème est qu'il est rare de trouver des ensembles de données open source qui correspondent parfaitement à ce qu'on recherche, ou des API gratuites qui nous donnent accès aux données.

L'objectif est de mettre en avant un moyen qui nous permet d'extraire des données volumineuses à partir de différentes sources et de garder une meilleure visibilité sur ces dernières, afin de pouvoir les exploiter d'une manière efficace. Dans ce contexte le Web scraping peut s'avérer très utile pour obtenir les données et les rendre accessible aux utilisateurs.

Contribution

Notre travail a pour but de mettre en avant deux outils des plus utilisés du web scraping :Scrapy et apache nutch, et de les étudier sur tous les aspects afin de mettre à notre disposition le bon outil qui offre des techniques basées sur des principes simples pour la récupération de données des pages web, de façon automatique et selon nos propre besoin.

Organisation du mémoire :

Ce mémoire est organisé en quatre chapitres :

Chapitre1 :généralité sur web scraping , dans ce chapitre nous présentons les concepts généraux liés au web scraping, par la suite nous présentons quelques outils utilisés pour faire du web scraping.

Chapitre2 : **étude et implémentation de scrapy**,dans ce chapitre nous présentons l'un des outils du web scraping qui est scrapy, et nous nous intéressons a ses différentes fonctionnalités avancées ainsi son implémentaion.

Chapitre3 : étude et implémentation de apache-nutch, dans ce chapitre nous présentons l'un des outils du web scraping qui est appache-nutch, et nous nous intéressons a ses différentes fonctionnalités avancées ainsi son implémentaion.

Chapitre4:**étude comparative de scrapy et nutch**, dans ce chapitre nous procédons une études comparative entre les deux outils précédents sur les différents aspects.

Chapitre I :

Généralités sur le Web scraping

Introduction

Le **web scrapping** est aussi vieux que le web lui-même, c'est un terme très largement connu dans le monde de la programmation. Dans ce chapitre nous présentons les concepts de base de web scraping ainsi que ses caractéristiques, les différents outils qui nous permettent l'extraction des données.

I. Définitions de web scraping

Le Web Scraping[1] également appelé data scraping est une technique désignée pour l'extraction, la capture et la récolte d'informations provenant de sites web. Cette technique se concentre principalement sur la transformation de données semi-structurées (format HTML) sur le web en données structurées (base de données ou feuille de style) et utilisable.

Figure 1:representation de web scraping

II. Caractéristique du web scraping

1. Grattez plusieurs pages et Parcourir les données récupérées.

[1] https://adrienlachaize.com/blog/quest-ce-que-le-web-scraping/

Web scraping nous permet d'analyser des sites Web et d'extraire des données structurées qui peuvent être utilisées pour une vaste gamme d'applications utiles, telles que l'exploration de données, le traitement de l'information ou l'archivage historique.

2. Exporter les données récupérées au format structuré.

3. Accélérer la recherche.

4. Rassembler des données.

III. Pourquoi faire appel au web scraping ?

L'extraction de données sur internet peut être réalisée de plusieurs manières différentes, notamment par le biais d'APIs.

Les API web sont utilisées pour Obtenir les données du web, telles que les interfaces fournies par les bases de données en ligne et de nombreuses appli cations web modernes (comme Twitter, Facebook et bien d'autres). C'est un moyen fantastique d'accéder à des données gouvernementales ou commerciales et d'extraire des données depuis les réseaux sociaux

Les APIs permettent d'utiliser un service web sans passer par l'interface utilisateur, simplement en codant. Les données sont souvent retournées sous forme d'un JSON – données structurées.

Il existe de nombreux cas d'usages aux APIs. Cependant les APIs trouvent rapidement leurs limites .En effet, les développeurs font souvent face à de fortes limitations en termes d'usage ou même en termes de fonctionnalités.

C'est là que le web scraping rentre en jeu et prend tout son sens. En effet, dans la plupart des cas, le web scraping va permettre à un développeur d'utiliser un service, d'extraire des données sans aucune limitation (sauf site très protégé).

Par exemple, grâce au scrapingPuppeteer, un développeur peut automatiser n'importe quelle tâche réalisable dans un navigateur internet.

Voyons désormais quelques exemples applicables au web scraping.

IV. Cadre d'utilisation du web scraping

Le web scraping est utilisé dans différents activités telles que :

- Utilisation privée : par exemple dans le cas de services online qui permettent d'agréger des éléments provenant de différents sites (exemple : Netvibes)
- Recherche académique : le web est une source de données qui concerne plusieurs domaines et le grand nombre de données disponibles permettent, parfois, de combler

les biais méthodologiques liés aux informations disponibles sur le web (véridicité, authenticité, etc.)

- Utilisation commerciale : comparaison entre informations disponibles entre sites concurrents (exemple : billets d'avion de différentes compagnie pour le même trajet)
- Analyse marketing : les "traces" laissées sur internet sont de plus en plus nombreuses et permettent de fournir une idée de nos préférences, habitudes, etc. - ces données peuvent très bien être exploitées pour des analyses de marketing (dans le cadre notamment de la publicité online)

V. Modalités d'extraction

On peut faire une distinction de base concernant le web scraping selon la modalité d'extraction des données :

- **Extraction manuelle** : une personne navigue le web pour extraire informations pertinentes aux intérêts depuis les pages qu'elle visite. La pratique la plus commune pour ce type d'extraction et le simple copier/coller.
- **Extraction sémi-automatique** : une personne utilise un logiciel ou une application web pour aspirer/nettoyer les éléments d'une ou plusieurs pages web pertinents aux intérêts.
- **Extraction automatique** : l'extraction se fait de manière totalement automatique grâce à l'émulation, par une machine, d'un navigateur web qui visite des pages et qui est capable de suivre les différents liens afin de générer automatiquement un ensemble de pages liées entre elles.

VI. 2.2 Modalités de traitement

Dans les cas d'extraction automatique ou sémi-automatique on peut distinguer ultérieurement par rapport au traitement des données recueilles :

- **Aucun traitement** : les données recueillies ne sont pas traitées et sont tout simplement rendues disponibles - par exemple à travers un fichier de texte ou un spreadsheet -

pour une analyse successive, à travers notamment un autre logiciel spécialisé (e.g. analyse de texte)

- **Visualisation des données brutes** : les données ne sont pas traitées mais seulement organisées selon des critères et les résultats sont affichés à l'aide de représentations graphiques similaires au format original (e.g. news feed, mash-up)
- **Traitement des données** : les données sont traitées à travers des algorithmes qui déterminent un certain résultat sur la base des analyses (e.g. comparaison de prix entre marchants online)
- **Traitement des données et visualisation** : les données sont traitées et, en plus, une représentation graphique adéquate à l'analyse des données et également fournie (e.g. nuages de mots-clés, cartes conceptuelles).

VII. Techniques d'extraction

Dans le cadre d'extractions automatiques ou semi-automatiques, il est nécessaire d'identifier dans les documents analysés les données d'intérêt afin de les séparer de l'ensemble du contenu. Voici une liste non exhaustive de techniques qui peuvent être utilisées :

- Expressions régulières
- XPath
- Traverser le DOM

VII.1 Expressions régulières

Les expressions régulières sont une fonctionnalité disponible pratiquement dans tout langage de programmation et qui permet d'identifier des patterns à l'intérieur de contenu textuel. Grâce à une syntaxe qui permet de combiner plusieurs règles d'analyse en même temps, il est possible d'extraire de manière ponctuelle des éléments qui correspondent aux critères définis dans l'expression régulière. Le mécanisme consiste à rechercher toutes les ressemblances entre la chaîne de caractères à trouver (le pattern) à l'intérieur du contenu cible de l'analyse. Les expressions régulières sont une technique très puissante d'extraction de contenu car elles s'appliquent indépendamment de la structure du document analysé. Cette puissance nécessite cependant une syntaxe assez complexe qui n'est pas très intuitive.

VII.2 XPath

XPath est un standard du W3C (l'organisme qui s'occupe des standards du Web) pour trouver des éléments dans un document XML. Ce langage exploite la structure hiérarchique des nœuds (et attributs) d'un document XML et nécessite par conséquent une structure de document très précise, ce qui n'est pas forcément le cas dans les pages HTML (voir plus bas dans la section HTML et web scraping).

VII.3 Traverser le DOM

Cette technique est similaire à XPath car elle exploite également la structure hiérarchique d'une page web à travers le DOM (Document Object Model). Il s'agit encore une fois d'un standard W3C qui permet d'accéder au contenu des différentes balises d'une page HTML grâce à leur positionnement hiérarchique dans la page. Pour accéder au contenu d'intérêt, on peut utiliser par exemple la notation des CSS (feuilles de style).

VIII. Quelques cas d'usage de web scraping

Le raclage Web possède de nombreux aspects et il existe certainement de nombreuses utilisations :

Scraper les données d'un site e-commerce :

Beaucoup d'entreprises scrapent les sites e-commerce concurrents à la recherche de *toutes modifications de prix, de descriptions de produits et d'images,* afin d'obtenir toutes les données possibles pour stimuler l'analyse et la modélisation prédictive des données.À moins que les tarifs ne soient concurrentiels, les sites e-commerce peuvent fermer leurs portes en un rien de temps.

Même constat avec les sites de voyage qui extraient les prix des sites des compagnies aériennes depuis longtemps.

Des solutions de web scraping personnalisées vous aideront à obtenir toutes les données imaginables dont vous pourriez avoir besoin.

Trouver les données de n'importe qui ou qu'elle entité :

Le webscraping permet de récupérer n'importe quelle donnée sur un individu X ou sur une entreprise Y. (surtout grâce aux réseaux sociaux).Ces données sont ensuite utilisées pour des analyses, des comparaisons, des décisions d'investissement, une embauche et plus encore.De nombreuses entreprises font du webscraping aujourd'hui sur des sites comme Le Bon Coin ou Indeed par exemple.

Analyse complexe et curation de contenu :

Le data scraping va également être très utile avant de lancer un site web par exemple pour comprendre l'intention de recherche des individus (en scrapant les pages de résultats google par exemple).

Le scraper va récupérer tous les résultats et pourra savoir comment les sites dans votre industrie communiquent par exemple. De ce fait vous pourrez vous aligner. A la suite de cette analyse vous pourrez programmer votre robot pour aller chercher du contenu qui match parfaitement avec les besoins découvert dans la première étape.

Le web scraping pour monitorer la réputation d'une marque :

Une fois l'écoute réalisée vous pourrez communiquer de la meilleure façon possible pour répondre parfaitement aux besoins de cette audience. Tout ça, basé sur leurs vrais sentiments.

IX. Le web :

Les Web scraping recueillent les données de sites Web de la même manière que les humains: le racleur accède à une page Web du site Web, obtient les données pertinentes et passe à la page Web suivante.

-Chaque site Web a une structure différente, c'est pourquoi les scrapers Web sont généralement conçus pour explorer un site Web. Les deux problèmes importants qui se posent lors de la mise en œuvre d'un racleur Web sont les suivants:

- **Quelle est la structure des pages Web contenant des données pertinentes?**
- **Comment pouvons-nous accéder à ces pages Web?**

Afin de répondre à ces questions, nous devons comprendre un peu le fonctionnement des sites Web.

IX.1 Les bases de web

IX.1.1 Definition

Le World Wide Web, littéralement la « toile (d'araignée) mondiale », communément appelé le Web, le web parfois la Toile ou le WWW, est un système hypertexte public fonctionnant sur Internet qui permet de consulter, avec un navigateur, des pages accessibles sur des sites.

IX.1.2 Concepts de bases

Une ressource du web est une entité informatique (texte, image, forum Usenet, boîte aux lettres électronique, etc.) accessible indépendamment d'autres ressources. Une ressource en accès public est librement accessible depuis Internet. Une ressource locale est présente sur l'ordinateur utilisé, par opposition à une ressource distante (ou en ligne), accessible à travers un réseau.

On ne peut accéder à une ressource distante qu'en respectant un **protocole de communication**. Les fonctionnalités de chaque protocole varient : réception, envoi, voire échange continu d'informations.

HTTP (pour HyperText Transfer Protocol) est le protocole de communication communément utilisé pour transférer les ressources du Web. HTTPS est la variante sécurisée de ce protocole.

Une URL (pour Uniform Resource Locator) pointe sur une ressource. C'est une chaîne de caractères permettant d'indiquer un protocole de communication et un emplacement pour toute ressource du Web.

Un hyperlien (ou lien) est un élément dans une ressource associé à une URL. Les hyperliens du Web sont orientés : ils permettent d'aller d'une source à une destination.

HTML (pour HyperText Markup Language) et XHTML (Extensible HyperText Markup Language) sont les langages informatiques permettant de décrire le contenu d'un document (titres, paragraphes, disposition des images, etc.) et d'y inclure des hyperliens. Un document HTML est un document décrit avec le langage HTML. Les documents HTML sont les ressources les plus consultées du Web.

HTTP conçu pour accéder aux ressources du Web. Sa fonction de base est de permettre la consultation des documents HTML disponibles sur les serveurs HTTP. Le support d'autres types de ressource et d'autres protocoles de communication dépend du navigateur considéré.

Une page Web (ou page) est un document destiné à être consulté avec un navigateur Web. Une page Web est toujours constituée d'une ressource centrale (généralement un document

HTML) et d'éventuelles ressources liées automatiquement accédées (typiquement des images).

Figure 2:page html de base

Un site Web (ou site) est un ensemble de pages Web et d'éventuelles autres ressources, liées dans une structure cohérente, publiées par un propriétaire (une entreprise, une administration, une association, un particulier, etc.) et hébergées sur un ou plusieurs serveurs Web.

IX.1.3 Langages de balisage : html

XML et HTML sont des *langages de balisage* étroitement liés. En fait, HTML est comme un dialecte XML spécialisé dans la structuration de pages Web. Cela signifie qu'ils utilisent un ensemble de balises ou de règles pour organiser et fournir des informations sur leur contenu. Cette structure permet d'automatiser le traitement, l'édition, le formatage, l'affichage, l'impression, etc. de ces informations.

Les documents XML stockent les données au format texte brut, ce qui rend relativement facile l'utilisation de données XML sans connaissances ni outils très spécialisés. Mais la structure de XML exige des techniques pour localiser son contenu.

HTML et XML ont une structure très similaire, ce qui explique pourquoi les sélecteurs XPath et CSS peuvent être utilisés de manière pratiquement interchangeable pour naviguer dans les documents HTML et XML

IX.1.4 Structure d'un document de balisage

Un document XML respecte les règles de syntaxe de base:

- Un document XML est structuré à l'aide de *nœuds*, qui comprennent des nœuds d'élément, des nœuds d'attribut et des nœuds de texte.
- Les nœuds d'élément XML doivent avoir une balise d'ouverture et de fermeture, par exemple une <catfood>balise d'ouverture et une </catfood>balise de fermeture. Tout ce qui se trouve entre ces balises est contenu dans l'élément.
- Les *noms de balises* XML sont sensibles à la casse, par exemple, ils <catfood>ne sont pas égaux <catFood>.
- Dans un élément, il peut y avoir d'autres *éléments enfants*. Ceux-ci doivent être correctement imbriqués (chaque élément enfant ouvert doit également être fermé):

```
<catfood type="basic">
<manufacturer>Purina</manufacturer>
<contact>
<address class="USA"> 12 Cat Way, Boise, Idaho, 21341</address>
</contact>
<date>2019-10-01</date>
</catfood>
```

- Dans un élément, il peut également y avoir des nœuds de texte. Purinaet 2019-10-01sont les deux nœuds de texte. Un autre nœud de texte contient l'espace blanc entre <catfood>et <manufacturer>.
- Nœuds d'attributs XML (comme typedans <catfood>ci - dessus) ont un nom et une valeur qui doit être cité.

Notez qu'il peut y avoir plusieurs éléments avec un nom de tag particulier:

```
<product>
<catfood type="basic"> ... </catfood>
<catfood type="basic"> ... </catfood>
<catfood type="premium"> ... </catfood>
</product>
```

Certaines de ces règles sont assouplies en HTML:

- les noms de balises et d'attributs sont insensibles à la casse (<catfood type="basic">égaux <catFood Type="basic">)
- certains éléments sont fermés automatiquement (par exemple, <img>ils ne peuvent contenir aucun autre élément ou texte)
- les valeurs d'attribut n'ont pas besoin d'être citées

HTML peut néanmoins être représenté comme une arborescence de nœuds

IX.1.5 Arborescence

Un moyen courant de représenter la structure d'un document XML ou HTML est l'arborescence *des nœuds*, où chaque rectangle est un nœud:

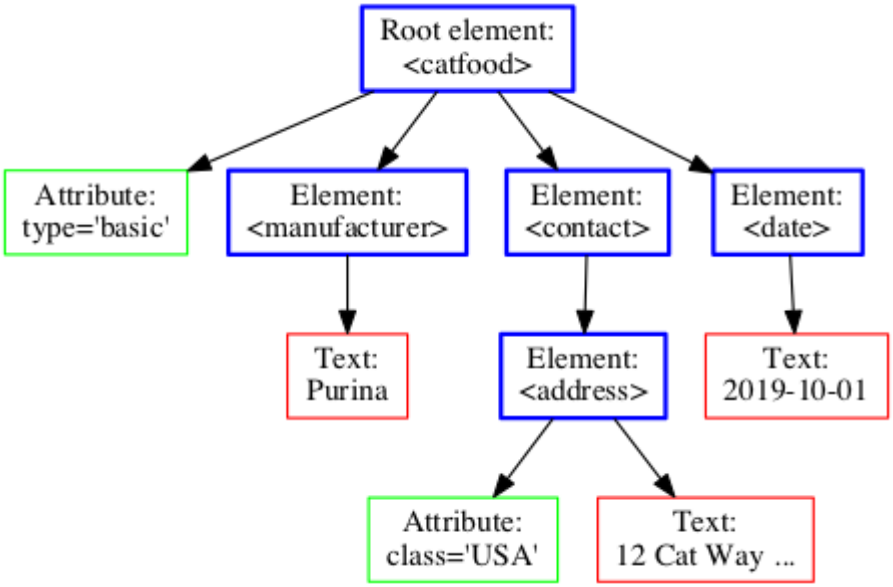


Figure 3:l'arborescence des noeuds

Nous utilisons les termes *parent*, *enfant* et *frère* pour décrire les relations hiérarchiques entre les nœuds:

- Le nœud supérieur est appelé *racine* (ou *nœud racine*). `<catfood>`est la racine ici.
- Chaque nœud a exactement un *parent,* sauf la racine (qui n'a pas de parent). Le `<manufacterer>`parent du `<catfood>`noeud est le noeud.
- Un nœud d'élément (un type de nœud) peut avoir zéro, un ou plusieurs *enfants*. Les nœuds d'attribut et de texte n'ont pas d'enfants. `<address>`a deux nœuds enfants, mais aucun élément enfant.
- *Les frères et sœurs* sont des nœuds avec le même parent.

- Les enfants d'un nœud et les enfants de ses enfants, etc., sont appelés ses *descendants*. De même, le parent d'un nœud et le parent de son parent, etc., sont appelés ses ancêtres. À l'exception du nœud racine, tous les nœuds sont des descendants du nœud racine.

IX.1.6 Eléments HTML communs

En HTML, les noms de balises ne sont généralement pas aussi spécifiques dans leur sémantique que `manufacturer`ou `address`. Voici quelques-uns des éléments HTML les plus courants:

Nom de tag	Signification
P	Un paragraphe de texte
h1	Une rubrique de haut niveau
`h2`, `h3`...	Une rubrique de niveau inférieur
Img	Une image
Tr	Une rangée dans une table
Td	Une cellule dans une table
A	Un lien
Div	Un bloc d'espace sur la page (générique)
Li	Un élément dans une liste
`Meta`	Informations sur la page non affichée
`Span`	Une portion de texte sur la page (générique)

Figure 4:les éléments html

IX.1.7 Expressions XPath

-L'utilisation de XPath est similaire à l'utilisation de la recherche avancée dans un catalogue de bibliothèques, où la nature structurée des informations bibliographiques nous permet de spécifier les champs de métadonnées à interroger.

Lorsque nous utilisons XPath, nous n'avons pas besoin de savoir à l'avance à quoi ressemble le contenu souhaité (comme avec les expressions régulières, où nous devons

connaître le modèle des données). Puisque les documents XML sont structurés comme un réseau de nœuds, XPath utilise cette structure pour parcourir les nœuds et sélectionner les données souhaitées. Nous avons juste besoin de savoir dans quels nœuds d'un fichier XML se trouve le contenu que nous voulons trouver.

Une *expression* XPath est, un peu comme une requête de recherche, un court texte décrivant les nœuds recherchés. Une expression XPath peut être *évaluée* sur un document (ou sur chacun des nombreux documents): l'évaluateur XPath suit les instructions impliquées par l'expression XPath, trouve les nœuds recherchés dans le document et les renvoie.

IX.1.8 Sélecteurs CSS

Un sélecteur CSS (comme avec un sélecteur XPath) est, un peu comme une requête de recherche, un court morceau de texte décrivant les nœuds recherchés. Un sélecteur CSS peut être *évalué* sur un document (ou sur chacun des nombreux documents): l'évaluateur suit les instructions impliquées par le sélecteur, trouve les nœuds recherchés dans le document et les renvoie au programme qui les a demandés.

Voici quelques exemples de ce que l'on peut exprimer avec les sélecteurs CSS (et XPath à des fins de comparaison) en fonction des fragments de document ci-dessus:

Sélecteur CSS	Expression Xpath	La description
`Address`	`//address`	Récupère chaque `address`élément (et son contenu) dans le document
`Catfoodaddress`	`//catfood//address`	Obtenez tous les `address`éléments quelque part dans un `catfood`élément
`catfood[type=basic]`	`//catfood[@type='basic']`	Obtenez tous les `catfood`éléments qui ont un `type`attribut avec la valeur «basic»

Figure 5:les selecteurs css

Les sélecteurs CSS peuvent uniquement récupérer les nœuds d'élément. Le texte et les attributs doivent être extraits en dehors de l'expression du sélecteur CSS.

X. Fonctionnement du grattage Web

Le raclage Web ressemble à tout autre processus Extract-Transform-Load (ETL). Web Scrapers explore des sites Web, en extrait des données, les transforme en un format structuré utilisable et les charge dans un fichier ou une base de données pour une utilisation ultérieure.

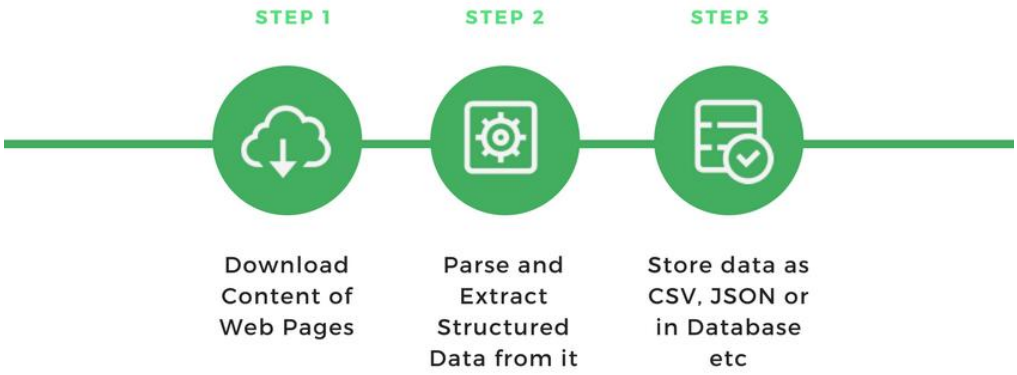


Figure 6:processus de grattage

Un grattoir à bande typique comprend les composants suivants :

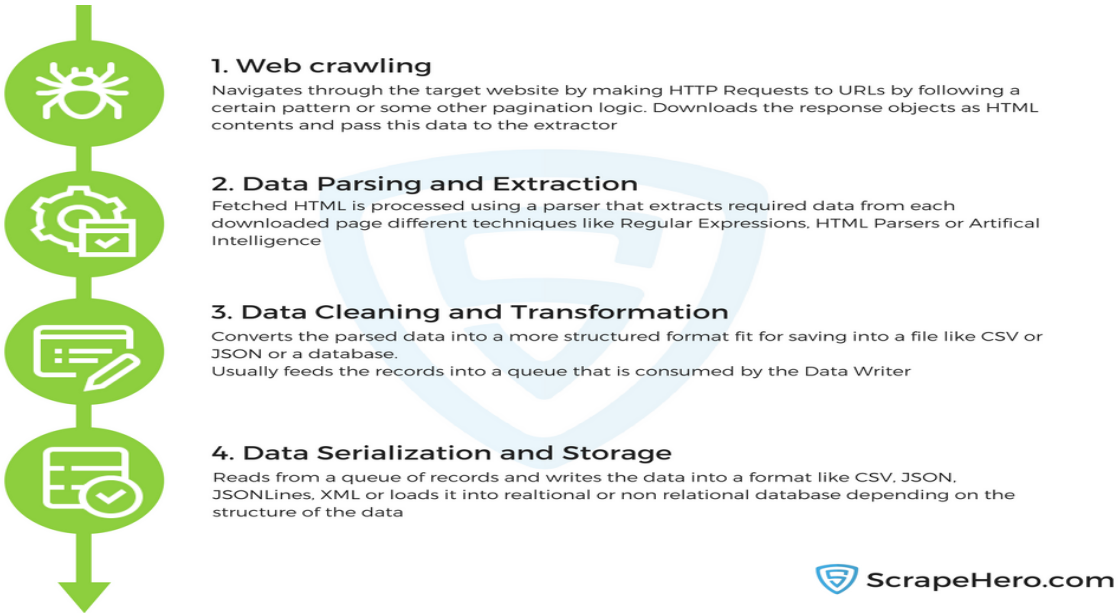


Figure 7:composant d'un grattoire

1. Un module de robot d'indexation web

Un module de robot d'indexation Web navigue sur le site Web cible en effectuant des demandes HTTP ou HTTPS vers des URL suivant un modèle spécifique ou une logique de pagination. Le robot télécharge les objets de la réponse sous forme de contenu HTML et transmet ces données à l'extracteur.
Par exemple, le robot d'exploration démarrera à l'adresse https://scrapehero.com et analysera le site en suivant les liens de la page d'accueil.

2. Un extracteur ou un module d'analyse

Le code HTML récupéré est traité à l'aide d'un analyseur qui extrait les données requises à partir du code HTML sous une forme semi-structurée.

3. Un module de transformation et de nettoyage des données

Les données extraites à l'aide d'un analyseur ne seront pas toujours au format qui convient à une utilisation immédiate. La plupart de données extraites nécessitent une forme de «nettoyage» ou de «transformation ». Des expressions régulières, et des méthodes de recherche sont utilisées pour effectuer ce nettoyage et cette transformation.

4. Module de stockage et de sérialisation des données

Une fois les données nettoyées récupérées, elles doivent être sérialisées en fonction des modèles de données requis. Il s'agit du dernier module qui produira les données dans un format standard pouvant être stocké dans des bases de données (Oracle, SQL Server, MongoDB, etc.), des fichiers JSON / CSV ou transmis à des entrepôts de données.

XI. Limites du web scraping

Il existe deux types de limites en ce qui concerne la pratique du web scraping :

- Limites techniques
- Limites déontologiques/légales

XI.1 Limites techniques

Selon la modalité d'extraction choisie , différentes restrictions techniques peuvent s'appliquer.

XI.1.1 Limites de l'extraction manuelle

Les limites techniques de ce type d'extraction concernent principalement le temps et l'effort nécessaires pour extraire une grande quantité de données. Ce type d'extraction est adéquat pour des informations ciblées et avec des critères de choix qui seraient difficiles à traduire de manière formelle, mais ne s'adapte pas à une recherche systématique.

XI.1.2 Limites des extractions automatique et sémi-automatique

Les extractions automatiques et sémi-automatiques sont similaires en ce qui concerne les limites techniques, mais se différencient notamment sur deux plans :

- **La taille et le temps d'extraction** : la recherche automatique permet d'extraire beaucoup plus d'information et de plusieurs sources en moins de temps
- **Le degré d' "envahissement" sur la source de données extraites** : la recherche automatique est généralement beaucoup plus chargée en termes d'impact sur les serveurs hébergeant les sites analysés

Dans les deux cas, les limites techniques suivantes peuvent s'appliquer :

- **Organisation de l'information inconsistente ou chaotique** : l'extraction sémi-automatique ou automatique nécessite l'utilisation de critères d'extraction qui impliques la possibilité d'identifier les informations à retenir par rapport aux informations inutiles aux finalités de l'analyse. L'information d'intérêt pourrait dans ce sens être présentée de manière différente selon le site d'appartenance, ou même au sein du même site (voir à ce propos la section Web scraping et HTML).
- **Changement de la structure de l'information** : même si des critères adéquats pour l'extraction des informations pertinents sont établis, dans le cadre d'une analyse qui s'étale dans le temps, il y a la possibilité que les sites analysés changent de structure et par conséquent les critères utilisés auparavant ne soient plus valides
    - Ce phénomène s'applique surtout aux sites qui utilisent des systèmes de gestion de contenu (e.g. WordPress, mais également Moodle) qui permettent de changer de thème graphique : le changement du thème implique très souvent aussi un changement de la structure de la page

- **Accès restreint à la page** : certains contenus d'un site sont disponibles exclusivement à travers un système d'authentification (exemple : login). Même s'il existe des outils qui permettent d'émuler un mécanisme de login, il faudrait en tout cas disposer d'un accès et, surtout, fournir ses propres informations d'accès (username et password) à l'outil d'extraction, c'est-à-dire exposer ses propres droits d'accès

- **Contenu généré dynamiquement** : certaines pages ne téléchargent pas tout le contenu à la première requête, mais intègrent du contenu en fonction des actions du navigateur. Même si certains outils avancés permettent d'émuler des interactions de ce type, dans la plupart des cas, l'analyse est faite sur le contenu HTML disponible lors de la première requête, sans interactions successives.

- Pour analyser le contenu d'une page web il faut la "visiter", c'est-à-dire générer du traffic pour le server qui l'héberge. Ce type de traffic "non humain" n'est souvent pas très bien accepté par les administrateurs des sites qui disposent de quelques instruments techniques pour le limiter :

    - **Limitation des requêtes dans un délais de temps** : les administrateurs peuvent limiter le nombre de "hit" (i.e. les fois que l'on accède à une page spécifique ou à n'importe quelle page sur un même domain) provenant de la même source, justement parce que un navigateur "humain" nécessite normalement d'un certain temps pour passer d'une page à l'autre

    - **Bloque des adresses IP** : si un site reçoit trop de requêtes en même temps, il y a le risque d'enchainer un Denial of Service (DoS), c'est-à-dire que le server ne peut satisfaire toutes les requêtes qu'il reçoit et pour éviter des problèmes il préfère couper les services (ce phénomène est souvent utilisé dans les attaques informatiques). Les administrateur peuvent identifier des adresse IP qui sont la source de nombreuses requêtes et leur empêcher l'accès.

    - **Limitation de la bande passante** : visiter un site signifie télécharger son contenu, ce qui implique la "consommation" de bande passante d'un server. Dans le cas de petit server et hébergement mutualisés, cette bande peut être limitée sur le mois et une fois dépassée, le site n'est plus disponible jusqu'au mois prochain.

    - **Obligation de saisir des données pour acceder à un contenu** : surtout dans le cadre d'ouverture de compte online, les sites demandent de saisir des champs qu'une machine ne peut pas remplir (e.g. saisir le texte tiré d'une image, CAPTCHA, etc.)

En conclusion, il faut être très attentif à l'utilisation de systèmes d'extraction, surtout dans le cas d'une extraction massive et répétée dans le temps.

XI.2 Limites déontologiques/légales

C'est des limites liées au droit d'utilisation des sites web , par exemple le respect des droits d'auteur dans le cas des sites web de presses .

XII. Outils de Web scraping

Pour réaliser du web scraping , de nombreux outils sont disponibles. Nous allons passer en revue quelques-uns .Ceux-ci diffèrent par leur capacité ou leur expressivité, et par leur facilité d'utilisation :

XII.1 Scrapy :

Scrapy[2] est un framework open source collaboratif qui permet d'extraire les données d'un site web de manière simple et rapide. Développé sous Python, Scrapy dispose d'une grande communauté qui n'hésite pas à créer des modules supplémentaires pour améliorer l'outil.

[2] https://codecondo.com/web-scraping-tools-extracting-data/

XII.2 Apache Nutch ™ :

Apach_Nutch est open-source et offre des interfaces modulaires et enfichables pour résoudre les problèmes d'analyse. Vous pouvez facilement créer votre propre moteur de recherche si vous le souhaitez.

XII.3 Import.IO :

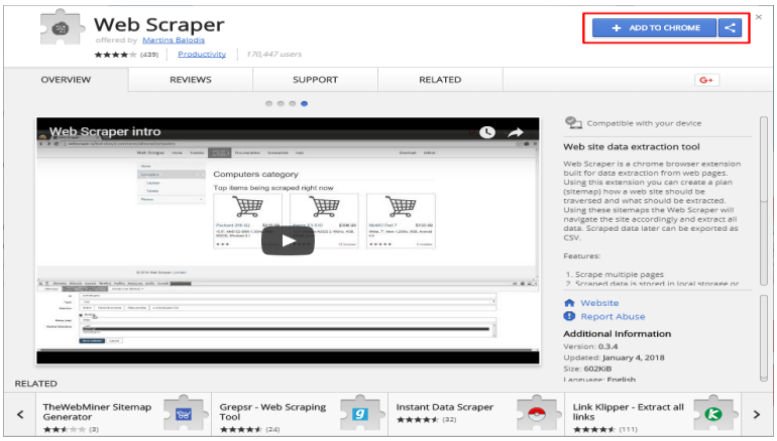
Import.io [3] Cet outil scrape les données de n'importe qu'elle page web. Import.io permet également d'exporter les données au format CSV. C'est idéal pour scraper un grand nombre de pages rapidement sans coder.

XII.4 Scrapinghub :

[3] https://codecondo.com/web-scraping-tools-extracting-data/

Scrapinghub est un outil basé sur le Cloud qui convertit les pages web en contenu structuré. Il permet de centraliser des données au même endroit afin de les analyser correctement. Scrapinghub utilise un outil qui permet d'éviter les robots et ainsi pouvoir explorer facilement des sites importants ou protégés contre les robots

XII.5 Web scraper :



Webscraper[4] est une extension disponible sous Google Chrome qui permet d'extraire les données d'un site internet très rapidement. Web Scraper naviguera sur les sites choisis afin d'en extraire toutes les données. Les données collectées peuvent être exportées sous forme de CSV. L'extension vous permet également de scrapper plusieurs sites à la fois ou même les programmer.

[4] https://codecondo.com/web-scraping-tools-extracting-data/

Conclusion :

Dans ce chapitre, nous avons présenté les différents concepts du web scraping ainsi que les complexités typiques associées au grattage des sites Web, et pour finir nous avons présenté quelques outils qui pourrons nous aider a faire du web scraping .

Dans le chapitre suivant nous allons nous intéresser à l'un des outils du web scrapping, permettant d'analyser des sites Web et d'extraire des données structurées qui peuvent être utilisées pour une vaste gamme d'applications utiles, telles que l'exploration de données, le traitement de l'information ou l'archivage historique qui est scrapy

Chapitre II :

Etude et implémentation de scrapy

Introduction

Extraire des données de pages Web peut être un travail fastidieux. Le web scraping nous met en avant certain des ses outils pour nous faciliter cette tache.
Dans ce chapitre, nous présentons Scrapy l'un des outils de web scraping, et nous nous intéressons a ses avantages ainsi que sa performance et ses différentes fonctionnalités avancées.

I. Présentation de scrapy

Plusieurs définitions de Scrapy ont vu le jour ces dernieres années, nous citons dans ce contexte les deux définition ssuivantes :

I.1 Définition1 :

Scrapy[1] est un framework écrit en python. Il permet de scrapy des sites web, d'en extraire d'énormes quantités de données de manière robuste et efficace. Il est simple et puissant, avec de nombreuses fonctionnalités et extensions possibles.
Scrapy prend en charge l'extraction de données à partir de sources HTML à l'aide des expressions XPath et CSS.

I.2 Definition 2 :

Scrapy[2] est un framework d'application permettant d'analyser des sites Web et d'extraire des données structurées qui peuvent être utilisées pour une vaste gamme d'applications utiles, telles que l'exploration de données, le traitement de l'information ou l'archivage historique.

II. Pourquoi utiliser Scrapy?

- Simple : aucune notion avancée en Python n'est nécessaire pour utiliser Scrapy

- Productif : l'empreinte de code à générer est très courte, la plupart des opérations sont gérées par Scrapy

- Rapide : le framework est rapide, avec une gestion d'actions en parallèle notamment

- Extensible : chaque robot peut être personnalisé via des extensions, modifiant son comportement

[1] https://docs.scrapy.org/
[2] https://docs.scrapy.org/en/0.10.3/intro/overview.html

- Portable : les robots Scrapy sont compatibles Linux, Windows, Mac et BSD

- Open Source

- Robuste : grâce à une batterie de tests effectuées aussi bien par les développeurs que la communauté

- Il est plus facile de construire et d'étendre de grands projets d'exploration.

- Il a un mécanisme intégré appelé sélecteurs, pour extraire les données des sites Web.

III. Caractéristiques de scrapy

- Rapide et puissant :écrivez les règles pour extraire les données et laissez Scrapy faire le reste.

- Facilement extensible : extensible de par sa conception, branchez facilement de nouvelles fonctionnalités sans toucher au cœur.

- Portabilité :écrit en Python et fonctionne sous Linux, Windows, Mac et BSD.

- Prise en charge intégrée de la sélection et de l'extraction de données à partir de sources H²TML / XML à l'aide de sélecteurs CSS étendus et d'expressions XPath, avec des méthodes d'assistance permettant une extraction à l'aide d'expressions régulières.

- Une console de shell interactive (compatible IPython) pour essayer les expressions CSS et XPath afin d'extraire des données, ce qui est très utile lors de l'écriture ou du débogage de vos araignées.

-Prise en charge intégrée permettant de générer des exportations de flux dans plusieurs formats (JSON, CSV, XML) et de les stocker dans plusieurs backends (FTP, S3, système de fichiers local)

-Prise en charge étendue de l'extensibilité, vous permettant de brancher votre propre fonctionnalité à l'aide de signaux et d'une API bien définie (middlewares, extensions et pipelines).

- Large gamme d'extensions et de middlewares intégrés pour la gestion: cookies et gestion de session. Fonctionnalités HTTP telles que compression, authentification, mise en cache, usurpation d'agent utilisateur, robots.txt, restriction de profondeur d'analyse

-Une console Telnet permettant de se connecter à une console Python s'exécutant à l'intérieur votre processus Scrapy, pour analyser et déboguer votre robot d'exploration. D'autres objets, tels que des robots réutilisables, pour explorer des sites à partir de sitemaps et de flux XML / CSV, un pipeline de médias pour le téléchargement automatique d'images (ou de tout autre média) associés aux éléments récupérés, une mise en cache

IV. L'architecture de Scrapy

Le diagramme[3] suivant présente une vue d'ensemble de l'architecture Scrapy et comment ses composants interagissent, ainsi qu'un aperçu du flux de données à l'intérieur du système (indiqué par les flèches rouges).

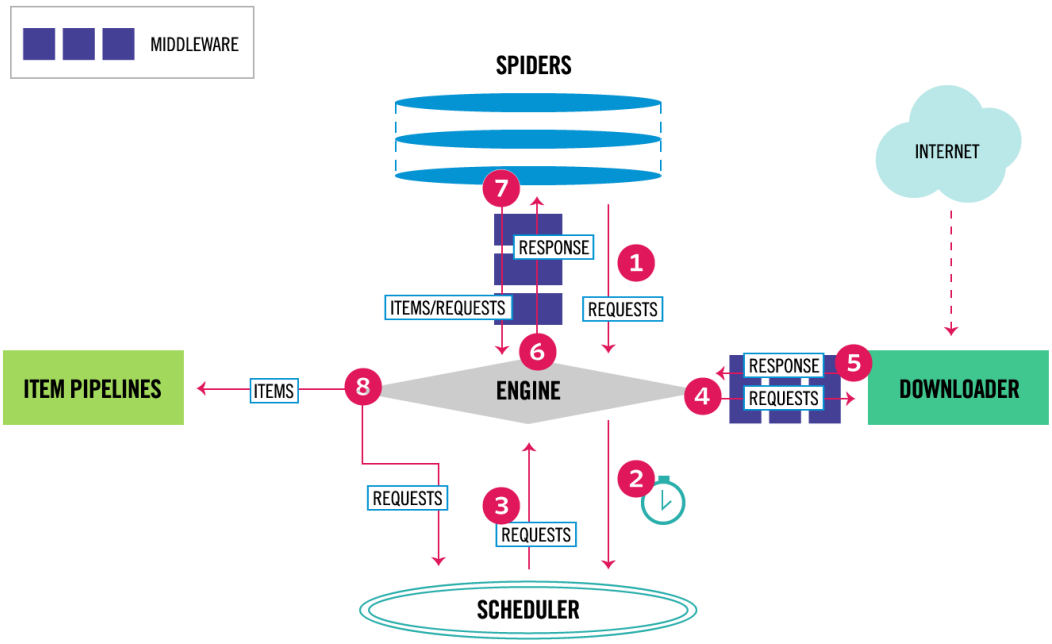


Figure 1:L'architecture de Scrapy

IV.1 Description des composants :

Scrapy Engine

Le moteur est chargé de contrôler le flux de données entre tous les composants du système et de déclencher des événements lorsque certaines actions se produisent.

Planificateur

Le planificateur (schediler) reçoit les demandes du moteur et les met en file d'attente pour les envoyer plus tard (également au moteur) lorsque le moteur les demande.

[3] https://docs.scrapy.org/en/latest/topics/architecture.html

Téléchargeur

Le téléchargeur(downloader) est chargé d'extraire les pages Web et de les envoyer vers le moteur, qui les transmet ensuite aux araignées.

Araignées

Les araignées sont des classes personnalisées écrites par les utilisateurs de Scrapy pour analyser les réponses et en extraire des éléments (c'est-à-dire des éléments grattés) .

Item Pipeline

Le pipeline d'Item est responsable du traitement des items une fois qu'ils ont été extraits (ou grattés) par les araignées. De plus que le grattage des donees il inclue la validation le stocker l'élément dans une base de données.

Middlewares téléchargeurs

Les middlewares Downloader sont des crochets spécifiques qui se situent entre le moteur et le téléchargeur et traitent les demandes quand elles passent du moteur au téléchargeur, ainsi que les réponses qui passent de Downloader au moteur.

un middleware Downloade est utilisé pour effectuer l'une des opérations suivantes:

-traiter une demande juste avant qu'elle soit envoyée au téléchargeur (c'est-à-dire juste avant que Scrapy n'envoie la demande au site Web);

-changer la réponse reçue avant de la transmettre à une araignée;

-envoyer une nouvelle demande au lieu de transmettre la réponse reçue à une araignée;

-transmettre une réponse à une araignée sans récupérer une page Web;

-déposer silencieusement certaines demandes.

Middlewares Spider

Les middlewares Spider sont des crochets spécifiques situés entre le moteur et les araignées et capables de traiter les entrées (réponses) et les sorties (éléments et demandes) de l'araignée.

un middleware Spider est utilisé pour :

- sortie post-traitement des rappels d'araignées - modification / ajout / suppression de demandes ou d'éléments;
- post-traitement start_requests;
- gérer les exceptions d'araignées;
- appel errback au lieu de callback pour certaines des demandes en fonction du contenu de la réponse.

IV.2 Présentations de flux de données dans Scrapy

Le flux de données dans Scrapy est contrôlé par le moteur d'exécution et se présente comme suit:

1. Le moteur (engine) obtient les requêtes initiales à analyser de l' araignée .
2. Le moteur planifie les demandes dans le planificateur et demande que les prochaines demandes soient analysées.
3. Le planificateur renvoie les prochaines demandes au moteur .
4. Le moteur envoie les demandes au téléchargeur en passant par les middlewares du téléchargeur par le méthode (process_request()).
5. Une fois la page téléchargée, le téléchargeur génère une réponse (télécharge la page) et l'envoie au moteur, en passant par les middlewares du téléchargeur utilisant la méthode (process_response()).
6. Le moteur reçoit la réponse du téléchargeur et l'envoie à l' araignée pour le traitement, en passant par le spider Middleware , utilisant la methode (process_spider_input()).
7. L' araignée traite la réponse et retourne les éléments grattées (les articles) ainsi les nouvelles demandes au moteur , en passant par le Middleware Araignée par la méthode (process_spider_output()).
8. Le moteur envoie les éléments traités aux pipelines d'éléments puis envoie les demandes a traitées au planificateur et demande les prochaines demandes éventuelles à analyser.
9. Le processus se répète (à partir de l'étape 1) jusqu'à ce qu'il n'y ait plus de demandes du planificateur .

V. Concepts de base

V.1 Outil de ligne de commande

Scrapy est contrôlé via « scrapy outil de ligne de commande », appelé ici «outil_Scrapy» pour le différencier des sous-commandes, que nous appelons simplement «commandes» ou «commandes Scrapy».
L'outil Scrapy fournit plusieurs commandes, à des fins multiples, et chacune accepte un ensemble différent d'arguments et d'options.

V.1.1 Structure par défaut d'un projet Scrapy

Avant de plonger dans l'outil de ligne de commande et ses sous-commandes, commençons par comprendre la structure de répertoires d'un projet Scrapy.
Bien qu'il puisse être modifié, tous les projets Scrapy ont la même structure de fichier par défaut, semblable à ceci:

```
├── scrapy.cfg         # déploie le fichier de configuration
└── scrapy_spider      # module Python de project, vous allez importer votre code
                              A partir d'ici
   ├── __init__.py
   ├── items.py       # fichier de définition des éléments de projet
   ├── middlewares.py # fichier middlewares du projet
   ├── pipelines.py   # fichier de pipeline de projet
   ├── setting.py     # fichier de paramètres du projet
   └── spiders        # un répertoire où se trouvent les araignées
      ├── __init__.py
      └── example.py # l'Araignée qu'on  vient de créer
```

les deux fichiers les plus importants sont:

- settings.py : Ce fichier contient les paramètres que vous avez définis pour votre projet.
- spiders / :Ce dossier est l'endroit où tous les spiders personnalisés seront stockés. Chaque fois que vous demandez à Scrapy de faire fonctionner une araignée, il la recherchera dans ce dossier.

Un répertoire racine du projet et celui qui contiens le fichier scrapy.cfg, ce répertoire peut être partagé par plusieurs projet scrapy chacun avec son propre module de paramètre. Voici un example :

```
[settings]
default = myproject1.settings
project1 = myproject1.settings
project2 = myproject2.settings
```

V.1.2 Commandes d'outil disponibles

Il existe deux types de commandes, celles qui ne fonctionnent que dans un projet Scrapy (commandes spécifiques au projet) et celles qui fonctionnent également sans projet Scrapy actif (commandes globales), bien qu'elles puissent se comporter différemment lors de l'exécution depuis un projet.

Pour obtenir plus d'information sur chaque commande en exécute la commande suivante :

```
scrapy -h
```

- Commandes globales:

startproject

Syntaxe: scrapy startproject <project_name> [project_dir]

Crée un nouveau projet Scrapy nommé project_name, sous le répertoire project_dir . Si project_dir n'a pas été spécifié, project_dir sera le même que project_name.

genspider

Syntaxe: scrapy genspider [-t template] <name> <domain>

Créez une nouvelle araignée dans le dossier actuel ou dans le spiders dossier du projet actuel , si elle est appelée depuis un projet. Le paramètre <name> est défini comme étant de l'araignée name, tandis que <domain>est utilisé pour générer les attributs de l'araignée allowed_domains et start_urls

shell

Syntaxe: scrapy shell [url]

Démarre le shell Scrapy pour l'URL donnée, le cas échéant ou vide si aucune URL n'est donnée.

fetch

Syntaxe: scrapy fetch <url>

Télécharge l'URL donnée à l'aide du téléchargeur Scrapy et écrit le contenu sur la sortie standard.

runspider

Syntax: scrapy runspider <spider_file.py>

Exécutez une araignée autonome dans un fichier Python, sans avoir à créer de projet.

settings - spécifie la valeur du paramètre du projet.
version - Il affiche la version Scrapy.
view - Il récupère l'URL à l'aide de Scrapy downloader et affiche le contenu dans un navigateur.

- commandes spécifiques au projet

crawl - Il est utilisé pour analyser des données à l'aide de l'araignée.
check - Vérifie les éléments renvoyés par la commande analysée.
liste - Affiche la liste des araignées disponibles présentes dans le projet.
edit - Vous pouvez éditer les araignées en utilisant l'éditeur.
parse - Il analyse l'URL donnée avec l'araignée.
bench - Il est utilisé pour exécuter un test de référence rapide (Benchmark indique le nombre de pages pouvant être explorées par minute avec Scrapy)

V.2 Araignées : (spiders)

Les araignées[4] sont des classes qui définissent comment un site donné (ou un groupe de sites) seront scapé, y compris comment effectuer l'analyse et comment extraire des données structurées de leurs pages . En d'autres termes, les araignées sont l'endroit où vous définissez le comportement personnalisé pour l'exploration et l'analyse des pages d'un site particulier (ou, dans certains cas, d'un groupe de sites).

V.2.1 Cycle de grattage

Pour les araignées, le cycle de grattage passe les etapes suivantes:

1. Vous commencez par générer les demandes initiales pour analyser les premières URL et spécifiez une fonction de rappel à appeler avec la réponse téléchargée à partir de ces demandes.

Les premières requêtes à exécuter sont obtenues en appelant la méthode start_requests() générée par Request (par défaut) pour les URL spécifiées dans la start_urls, et méthode la parse en tant que fonction de rappel pour les requêtes.

2. Dans la fonction de rappel callback, vous analysez la réponse (page Web) et renvoyez les données extraites, des Itemobjets, des Requestobjets ou un objet itérable de ces objets. Ces demandes contiendront également un rappel (peut-être le même) et seront ensuite téléchargées par Scrapy, puis leur réponse sera traitée par le rappel spécifié.

[4] https://docs.scrapy.org/

3. Dans les fonctions de rappel callback, vous analysez le contenu de la page, généralement à l'aide de sélecteurs (mais vous pouvez également utiliser BeautifulSoup, lxml ou le mécanisme de votre choix) et générez des éléments avec les données analysées.

4. Enfin, les éléments renvoyés par l'araignée seront généralement conservés dans une base de données (dans un pipeline d'éléments) ou écrits dans un fichier à l'aide des exportations par flux

V.2.2 Type d'araignées

-il existe différents types d'araignées par défaut regroupées dans Scrapy à des fins différentes :

scrapy.Spider :

***classe* scrapy.spiders.Spider**

C'est l'araignée la plus simple, et celle dont toutes les autres araignées doivent hériter (y compris les araignées fournies avec Scrapy, ainsi que les araignées que vous écrivez vous-même). Il ne fournit aucune fonctionnalité spéciale. Il fournit simplement une methode **start_requests()** implémentation par défaut qui envoie des requêtes à partir de l' attribut **start_urls** et appelle la méthode de spider **parse** pour chacune des réponses obtenues.

Name

Une chaîne qui définit le nom de cette araignée. Le nom de l'araignée indique comment l'araignée est localisée (et instanciée) par Scrapy; elle doit donc être unique.

allowed_domains

Liste facultative de chaînes contenant des domaines que cette araignée est autorisée à analyser. Les demandes d'URL n'appartenant pas aux noms de domaine spécifiés dans cette liste (ou leurs sous-domaines) ne seront pas suivies.

start_urls

Une liste d'URL à partir desquelles l'araignée va commencer à l'analyse,

from_crawler(*crawler , * args , ** kwargs*)

C'est la méthode utilisée par Scrapy pour créer vos araignées.

Paramètre:

- **crawler** (**Crawler** instance) - crawler auquel l'araignée sera liée
- **args** (*list*) - arguments passés à la méthode **init()**
- **kwargs** (*dict*) - arguments de mot clé transmis à la méthode **init()**

start_requests( )

Si aucune URL particulière n'est spécifiée et que l'araignée est ouverte pour le l'analyse, *Scrapy* appelle la méthode ***start_requests ()*** .

parse(*réponse*)

Il s'agit du rappel par défaut utilisé par Scrapy pour traiter les réponses téléchargées, lorsque leurs demandes ne spécifient pas de rappel.
Le **parse** procédé est en charge du traitement de la réponse et du renvoi des données extraites et / ou de plusieurs URL à suivre. Les rappels d'autres demandes ont les mêmes exigences que la class **Spider** .
Cette méthode, ainsi que tout autre rappel de demande, doit renvoyer un objet intégrable **Request** et / ou **dict** or **Item** objects.

Paramètres: response (Response) - la réponse à analyser

closed(*raison*)

Appelé quand l'araignée se ferme. Cette méthode fournit un raccourci vers signals.connect() pour le **spider_closed** signal

-Araignées Génériques

Scrapy est livré avec des araignées génériques utiles que vous pouvez utiliser pour sous-classer vos araignées. Leur objectif est de fournir des fonctionnalités pratiques pour quelques cas de grattage courants, tels que le suivi de tous les liens d'un site en fonction de certaines règles.

-CrawlSpider

***classe* scrapy.spiders.CrawlSpider**

Il s'agit de l'araignée la plus utilisée pour explorer les sites Web classiques, car elle offre un mécanisme pratique pour suivre les liens en définissant un ensemble de règles.

Règles :

Il s'agit d'une liste d'objets de règle qui définit la manière dont le robot suit le lien.

- **LinkExtractor** :Il spécifie comment l'araignée suit les liens et extrait les données.

- **callback :**Il doit être appelé après chaque page grattée.

- **follow :**Il spécifie s'il faut continuer à suivre les liens ou non

V.2.3 Arguments d'araignée

Les araignées peuvent recevoir des arguments qui modifient leur comportement. Certaines des utilisations courantes des arguments de l'araignée sont de définir les URL de début ou de limiter l'analyse à certaines sections du site, mais elles peuvent être utilisées pour configurer les fonctionnalités de l'araignée.
Les arguments de l'araignée sont passés à travers la commande **crawl** en utilisant l'option **-a** .
Par exemple:

```
scrapy crawl myspider -a category=electronics
```

Les araignées peuvent accéder aux arguments dans leurs méthodes ***__init__*** .
Example :

```
import scrapy

class MySpider(scrapy.Spider):
    name = 'myspider'
    def __init__(self, category=None, *args, **kwargs):
        super(MySpider, self).__init__(*args, **kwargs)
        self.start_urls = ['http://www.example.com/categories/%s' % category]
        # ..
```

V.3 Extraction des données

Pour scraper des pages Web, la tâche la plus courante consiste à extraire des données de la source HTML.

-Scrapy est livré avec son propre mécanisme d'extraction de données. Ils sont appelés des sélecteurs car ils «sélectionnent» certaines parties du document HTML spécifiées par **des expressions XPath** ou **CSS** .
Les sélecteurs ont quatre méthodes de base qui sont :

re() :Il renvoie une liste de chaînes unicode, extraites lorsque l'expression régulière a été donnée en argument.

XPath () : Il retourne une liste de sélecteurs, qui représente les noeuds sélectionnés par l'expression xpath donnée en argument

CSS() : Il retourne une liste de sélecteurs, qui représente les nœuds sélectionnés par l'expression CSS donnée sous forme d'argument.

extract() : Il renvoie une chaîne unicode avec les données sélectionnées.

Pour extraire des données d'un site HTML , nous devons inspecter le code source du site pour obtenir XPaths.

V.3.1 Utilisation de sélecteurs dans le shell

Le shell Scrapy est un shell interactif qui vous permet d'extraire des données des pages web, il est utilisé pour tester les expressions XPath ou CSS et voir comment elles fonctionnent et quelles données extraites des pages Web que vous essayez de scrapy. Il vous permet de tester vos expressions de manière interactive lorsque vous écrivez votre araignée, sans avoir à exécuter l'araignée pour tester chaque modification.

Pour lancer le shell Scrapy, vous pouvez utiliser la commande shell comme ceci :

```
Scrapy shell url
```

Où <url>est l'URL de la page que vous voulez gratter.

V.3.2 Raccourcis disponibles

Le shell Scrapy est une console Python standard, qui fournit des fonctions de raccourci supplémentaires pour plus de commodité :

shelp() - imprimer une aide avec la liste des objets et des raccourcis disponibles
fetch(request) - extraire une nouvelle réponse de la requête donnée et mettre à jour tous les objets liés en conséquence.
view(response)- ouvre la réponse donnée dans votre navigateur Web local, pour inspection.

V.4 items

Le principal objectif de cette opération consiste à extraire des données structurées à partir de sources non structurées, généralement des pages Web. Utilisant les araignées, Scrapy fournit la classe **Item**. Les objects **Item** sont de simples conteneurs utilisés pour collecter les données récupérées. Ils fournissent une API de type dictionnaire avec une syntaxe pratique pour déclarer leurs champs disponibles.

Déclaration des items
Vous pouvez déclarer les éléments à l'aide de la syntaxe de définition de classe et des objets de champ suivants:

```
import scrapy


class QuoteItem(scrapy.Item):
    auther = scrapy.Field()
    quote = scrapy.Field()
```

V.5 Item Pipeline

Une fois qu'un item a été gratté par une araignée, il est envoyé au pipeline d'éléments qui le traite via plusieurs composants exécutés de manière séquentielle.
Chaque composant de pipeline d'éléments (parfois appelé simplement «Item Pipeline») est une classe Python qui implémente une méthode simple. Ils reçoivent un élément et effectuent une action dessus.
Les utilisations d'item pepeline sont les suivantes:

- nettoyage des données HTML
- valider les données récupérées (vérifier que les éléments contiennent certains champs)
- vérifier les doublons (et les supprimer)
- stocker l'article gratté dans une base de données

-Chaque composant de pipeline d'éléments est une classe Python qui doit implémenter la méthode suivante:

process_item(*self, item, spider*)

Cette méthode est appelée pour chaque composant de pipeline d'éléments. process_item() doit soit: renvoyer un dict avec des données, renvoyer un objet Item (ou une classe descendante),ou bien renvoyer une exception.

Paramètres:

item (Itemobjet ou un dict) - l'élément gratté
araignée (Spiderobjet) - l'araignée qui a gratté l'objet

V.6 Stockage de données

Le meilleur moyen de stocker les données récupérées consiste à utiliser les exportations de flux, qui garantissent que les données sont correctement stockées à l'aide de plusieurs formats de sérialisation. JSON, lignes JSON, CSV, XML sont les formats pris en charge facilement dans les formats de sérialisation.

JSON (**JavaScript Object Notation**) : est un format de données utiliser par scrapy a fin de stocker les donneés récupérées a l'aide de la commande suivante :

```
scrapy crawl quotes -o quotes.json
```

Cela générera un fichier quotes.json contenant tous les éléments récupérés, sérialisés en JSON .
Pour des raisons historiques, Scrapy ajoute un fichier au lieu d'écraser son ancien contenu. Si vous exécutez cette commande deux fois sans supprimer le fichier avant la deuxième fois, vous vous retrouverez avec un fichier JSON endommagé.

les lignes JSON : est un format pratique pour stocker des données structurées pouvant être traitées enregistrement par enregistrement. Utiliser par scrapy a l'aide de la ligne de commande suivante :

```
scrapy crawl quotes -o quotes.jl
```

Le format des lignes JSON est utile car il ressemble à un flux, vous pouvez facilement y ajouter de nouveaux enregistrements. Il n'ya pas le même problème de JSON lorsque vous exécutez deux fois. De plus, comme chaque enregistrement est une ligne distincte, vous pouvez traiter de gros fichiers sans avoir à tout stocker en mémoire

-Ces deux techniques sont valables pour une petite quantité de données. Si une grande quantité de données doit être traitée, nous pouvons utiliser le pipeline d'éléments pour le stockage dans une base de données, Si les données sont structurées (les champs sont connus à l'avance), des systèmes SQL tels que MySQL ou PostgreSQL sont utilisé.si elles sont non structuré, il serait préférable de le stocker sur des systèmes NoSQL tels que Mongo db .

VI. Implémention et évaluation

VI.1 Outils de développement

Pour implémenter notre approche, nous avons été amenées a utilisé l'environnements python (anaconda)ainsi que le serveur wampserveur pour le stockage des données grâce a PHPMyAdmin . aussi le langage de programmation python. Et enfin un éditeur de texte (sublimtext).

VI.1.1 I'environnement Python :

Pour installer un environnement de développement Python on as opté a l'utilisation d' Anaconda. A fin d avoir un environnement de développement fonctionnel avec les packages nécessaires pour faire du web scraping on utilisant Une des librairies les plus populaires du python qui est Scrapy.

VI.1.1.1 anaconda :

-**Anaconda Distribution**[5] Open Source est le moyen le plus simple d'exercer la science des données Python / R et l'apprentissage automatique sur Linux, Windows et Mac

A son installation, Anaconda installera Python ainsi qu'une multitude de packages. Cela nous évitera les problèmes d'incompatibilités entre les différents packages.

Finalement, Anaconda propose un outil de gestion de packages appelé **Conda**. Ce dernier permettra de mettre à jour et installer facilement les librairies dont on aura besoin pour nos développements

-Presentation de Conda :

Conda[6] est un système de gestion de paquets open source et un système de gestion de l'environnement fonctionnant sous Windows, macOS et Linux. Conda installe, exécute et met

[5] https://anaconda/distridution/
[6] https://docs.conda.io/

à jour rapidement les packages et leurs dépendances. D'autre part il enregistre, charge et commute facilement entre les environnements de votre ordinateur local. Il a été créé pour les programmes Python, mais il peut empaqueter et distribuer des logiciels pour n'importe quelle langue.

Conda est également inclus dans Anaconda , qui fournit une gestion sur site des packages et de l'environnement pour Python, R, Node.js, Java et d'autres piles d'applications. Conda est également disponible sur **Conda-Forge** , un canal communautaire.

- Instalation de python :

Pour installer le package python qui se situe dans l'environnement anaconda, on a utilisé conda :

En exécutant cette commande :

```
(base) C:\Users\User>conda install -c anaconda python
```

-Instalation de scrapy :

Scrapy s'exécute sur un environnement python, dans notre cas nous avons utilisé anaconda comme environnement de développement. Avec le quel on as installer le package scrapy a partir du canal conda-forge qui contient des packages a jour pour windows , en utilisant conda

En exécutant cette commande :

```
(base) C:\Users\User\Anaconda22>conda install -c conda-forge scrapy
```

-Scrapy est écrit en python pur et dépend de quelque package python clés suivant :

- **lxml** : un analyseur XML et HTML efficace

- **Parsel** :une bibliothèque d'extraction de données HTML / XML écrite au dessus de lxml,

- **w3lib** : une aide polyvalente pour traiter les URL et les encodages de pages Web

- **twisted** : un framework de réseau asynchrone

- **cryptographie** et **pyOpenSSL** : pour répondre à divers besoins de sécurité au niveau du réseau

VI.1.2 wampserveur

WampServer[7] est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données.

VI.1.3 Langage de programmation python

Python[8] est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet.

VI.1.4 Sublime Text

Sublime text[9] est un éditeur de texte générique codé en C++ et Python, disponiblsur Windows, Mac et Linux.

VI.2 Implémentation

Examinons maintenant une étude d'un cas pour acquérir plus d'expérience de Scrapy en tant qu'outil et de ses diverses fonctionnalités.

Nous allons scraper elwatan.com[10] , un site qui répertorie des articles de presse important. Et dans le but de rester au courant des actualités et des dernières informations de notre quotidiens, et de donner une vue qui rassemble tout les articles de presse disponible sur ce site en suivant chaque lien de chaque article de la page elwatan.com.

[7] http://www.wampserver.com/
[8] https://docs.python.org/
[9] www.sublimetext.com
[10] https://www.elwatan.com/

VI.2.1 Présentation du site

Figure 2:aperçu sur le site d'el watan

La page web de site elwatan.com est composer d'un ensemble de titres qui représentent des liens, et chaque lien fais référence a un article de presse et ces détails.

Voila un aperçu sur la page principale inspectée:

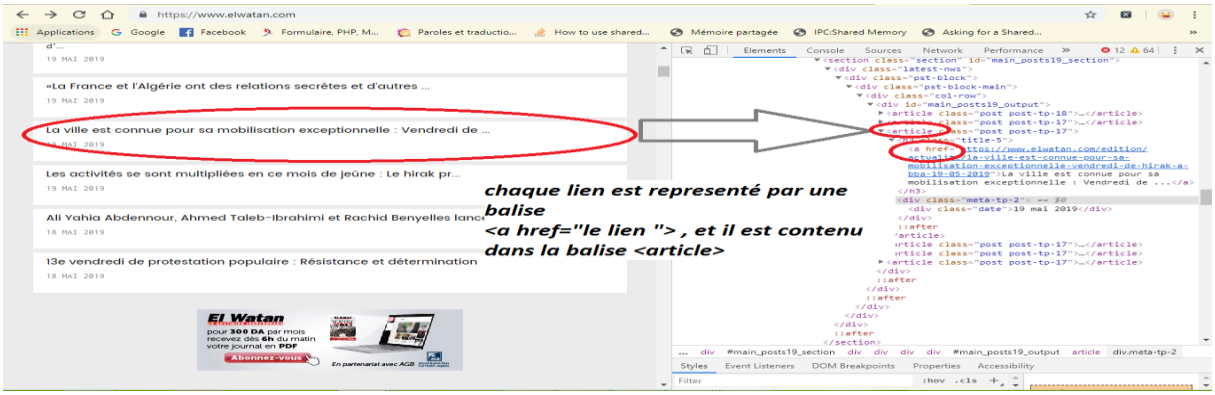


Figure 3:représentation de la page principale

Chaque lien représenté par un titre d'article fais appelle a une nouvelle page web qui contiens plus de détails sur cet article :le titre, l'image, date, auteur, et l'article.

Voila un aperçu sur la page secondaire:

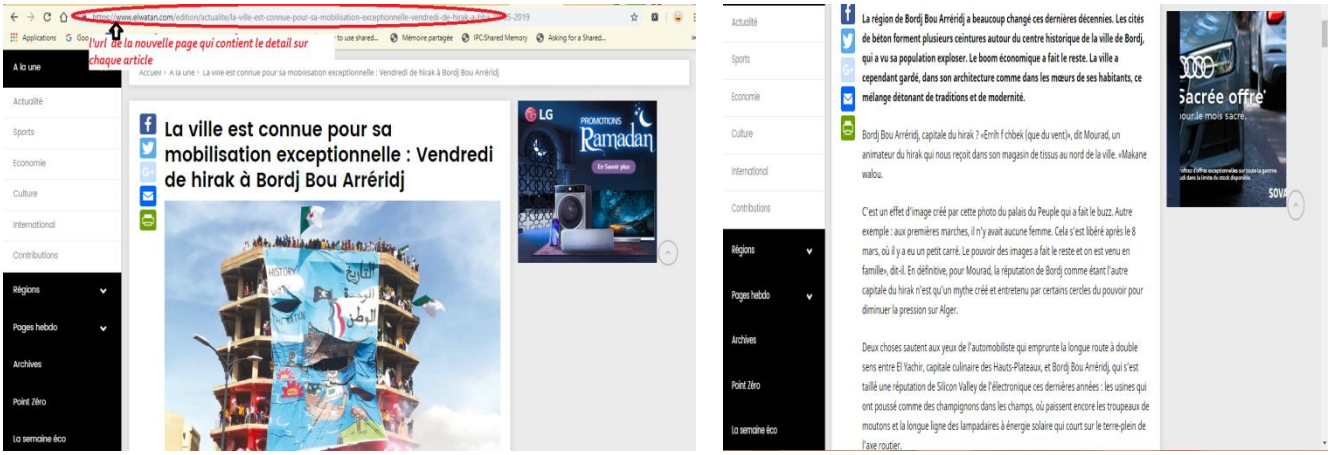


Figure 4:aperçu sur la page secondaire

la page web secondaire comporte les détails de chaque article : elle est composé d un titre contenu dans une balise qui elle-même contiens la balise <h1 classe=’’title-21’’> qui contiens le titre de l’article. Et une image dans une balise qui contiens la balise <img > avec un lien vers l’image de l’article. L’auteur de l’article dans la balise le nom de l auteur, la date dans la balise et la date ainsi l’heur de l’article. Ainsi le contenu de l’article représenté par l’ensemble de balises <p> et le paragraphe de l’article. Et toute ces balises sont regroupés dans une balise mère appelée <article classe= ‘’article’’>.

Voila un aperçu sur la page secondaire inspectée:

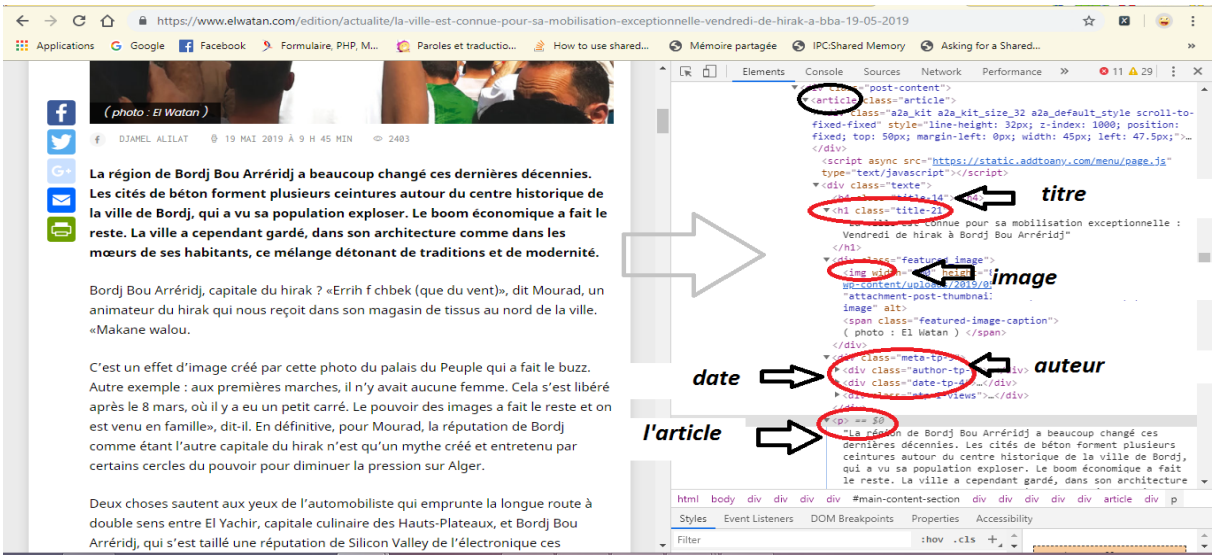


Figure 5:inspection de la page secondaire

Dans notre exemple qui se porte sur le site web elwatan.com, nous allons nous intéressé a l’extraction de quelques composant d’un article de presse tel que :

-le titre de l’article.

-l'auteur de l'article.

-le contenu de l'article.

VI.2.2 Extraction des données avec scrapy shell

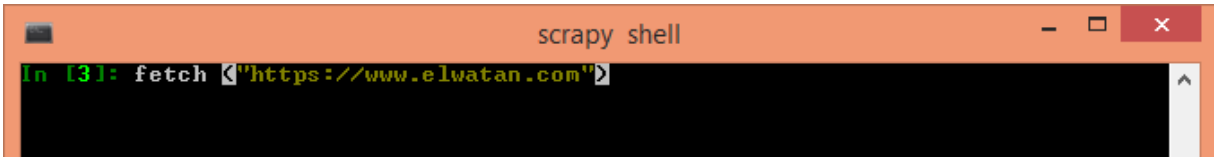
Dans cette partie, nous étudierons comment extraire les données des pages de notre site. apres l'inspection des pages du site on a constaté que les composant qui nous intéressent se situent dans les pages secondaire, et pour y accéder il faut suivre les liens de la page principale.

Les deux étapes principales de fonctionnement de processus d'extraction de données de notre site sont :

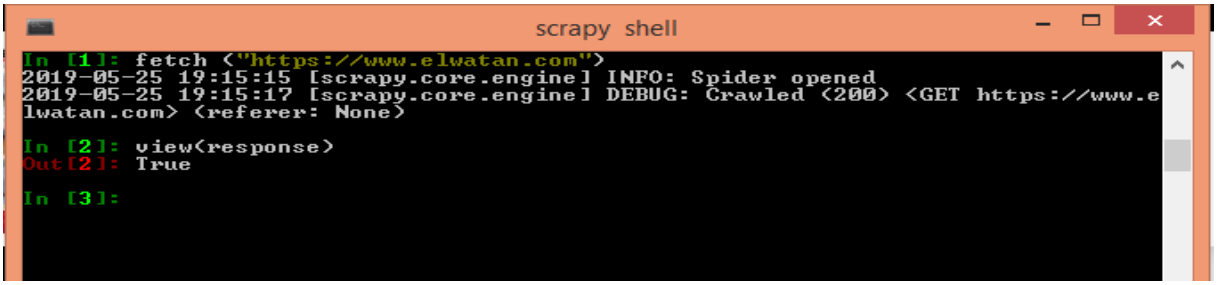
1. Extraction des liens de la page principale et poursuivre chaque lien.
2. Extraction des composants (titre, auteur, contenu) de chaque article.

VI.2.2.1 Extraction des liens de la page principale

Scrapy permet d'extraire des informations à partir de HTML à l'aide de sélecteurs CSS. Pour extraire le lien de chaque article nous devrons trouver le sélecteur CSS correspondant, et cela en utilisant scrapy shell.



L'adresse https://elwatan.com seras l'url de départ de rebot d'exploitation, suite à l'analyse scrapy retourne un objet «réponse» contenant les informations téléchargées. Voyons ce que le robot a téléchargé:



La commande view(response) ouvrira la page téléchargée dans votre navigateur par défaut comme suite :

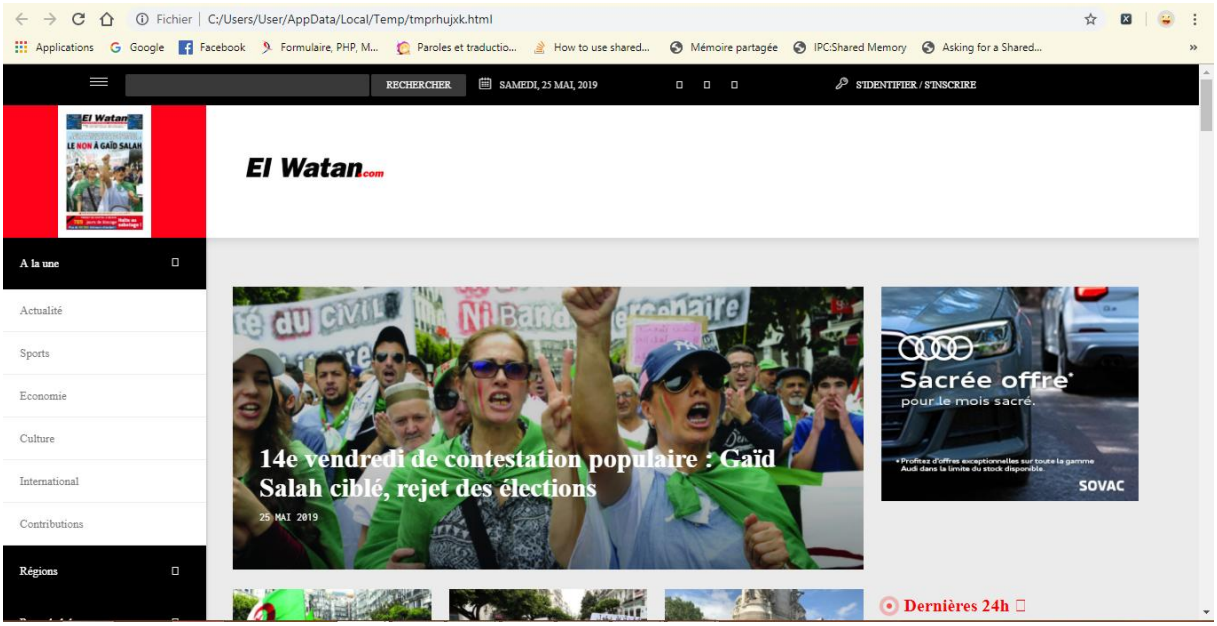


Figure 6:téléchargement de la page à partir du robot d'exploration

le robot d'exploration a téléchargé avec succès la page Web entière.

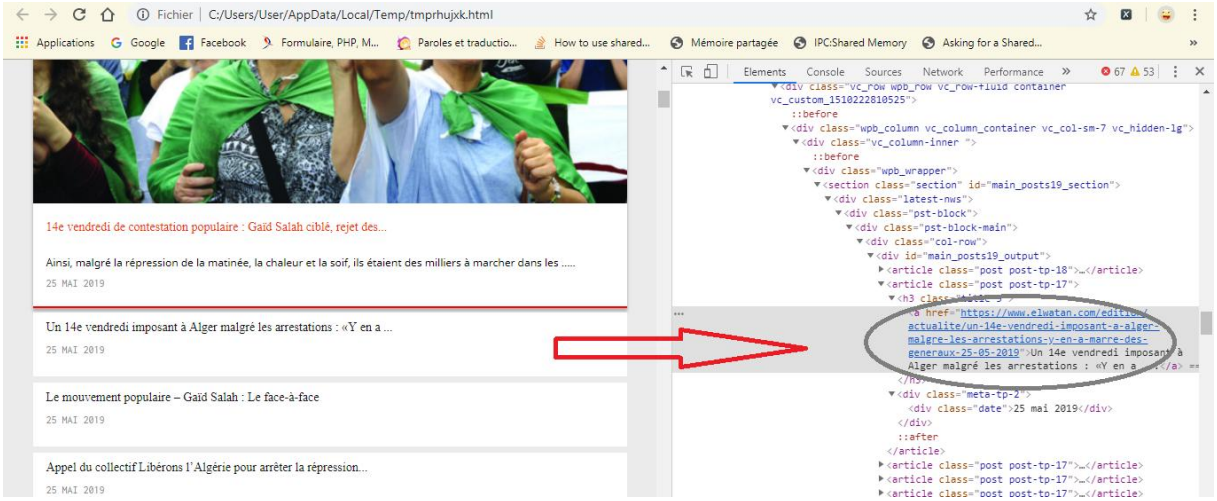
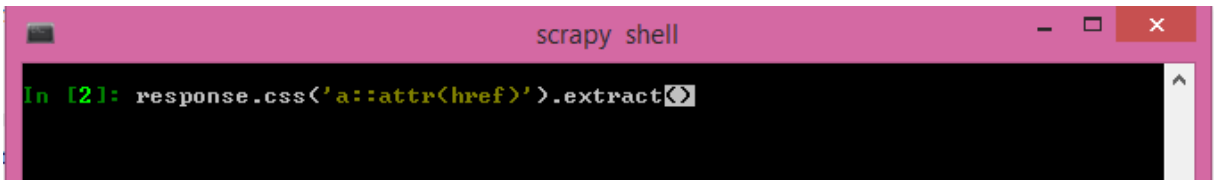
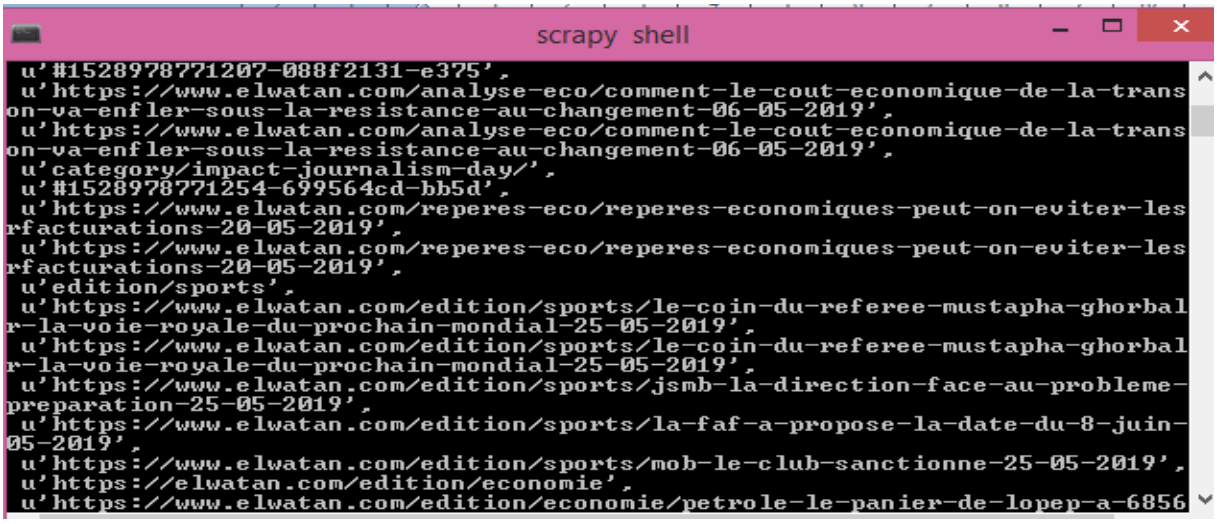


Figure 7: inspection des éléments

On inspectant la page téléchargé on constate que chaque titre est représenté par un lien cliquable, qui conduit vers une page secondaire dans le lien est indiquer dans la balise <a href=’’le lien ‘’> . Cela sera utile pour filtrer ce dernier du reste du contenu dans l'objet de réponse:



response.css (..) est une fonction qui aide à extraire le contenu basé sur le sélecteur css qui lui est transmis. :: attr(href) est utilisé pour extraire uniquement l'url des éléments correspondants. Cela est fait car scrapy renvoie directement l'élément correspondant avec le code HTML.

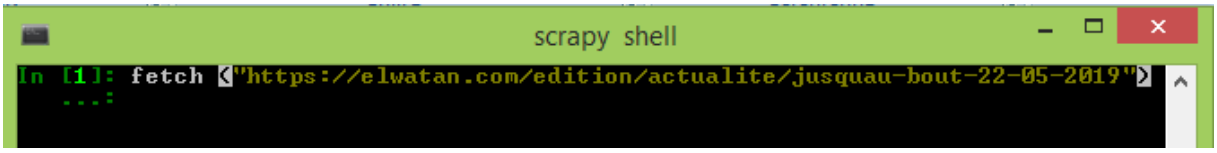


Apres avoir récupéré tout les url de nos articles de la page principale, on suit chaque url a fin d'en extraire le contenue nécessaire de chaque article.

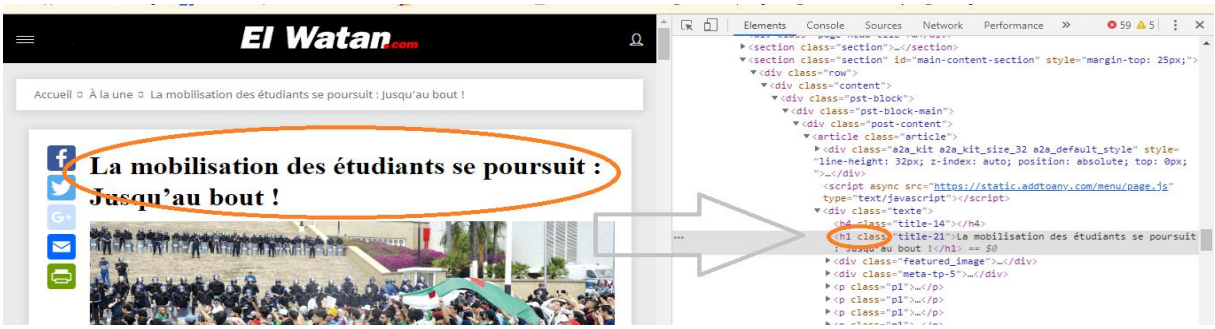
VI.2.2.2 Extraction des composants de chaque article

Extraction de titre de l'article :

Pour extraire le titre de chaque article nous devrons trouver le sélecteur CSS correspondant, et cela en utilisant scrapy shell.



L'adresse https://elwatan.com/edition/actualite/jusquau-bout-22-05-2019 seras l'url de départ de rebot d'exploitation qui es url de la page secondaire ou se trouve nos article détaillé.



On examinant la page téléchargé on constate que le titre se trouve dans la balise <h1> . Cela sera utile pour filtrer ce dernier du reste du contenu dans l'objet de réponse:

```
In [18]: fetch ("https://elwatan.com/edition/actualite/jusquau-bout-22-05-2019/
    ...: ")
2019-05-23 00:00:10 [scrapy.downloadermiddlewares.redirect] DEBUG: Redirecting (
301) to <GET https://www.elwatan.com/edition/actualite/jusquau-bout-22-05-2019>
from <GET https://elwatan.com/edition/actualite/jusquau-bout-22-05-2019/>
2019-05-23 00:00:10 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.e
lwatan.com/edition/actualite/jusquau-bout-22-05-2019> (referer: None)

In [19]: response.css('h1::text')[0].extract()
Out[19]: u'La mobilisation des \xe9tudiants se poursuit : Jusqu\u2019au bout !'

In [20]:
```

La fonction **response.css (..)**. nous permet d'extraire le titre de chaque article, :: text est utilisé pour extraire uniquement le contenu textuel des éléments correspondants. Cela est fait car scrapy renvoie directement l'élément correspondant avec le code HTML.

extraction de l auteur de l article :

On inspectant la page on constate que l'auteur se trouve entre la balise <a> l'auteur , cette dernier est contenu dans le qui est définit par une classe css <div class= ''author-tp-2''>

On utilise l'objet response() pour filtrer l'auteur du reste du contenu de la page . comme ceci :

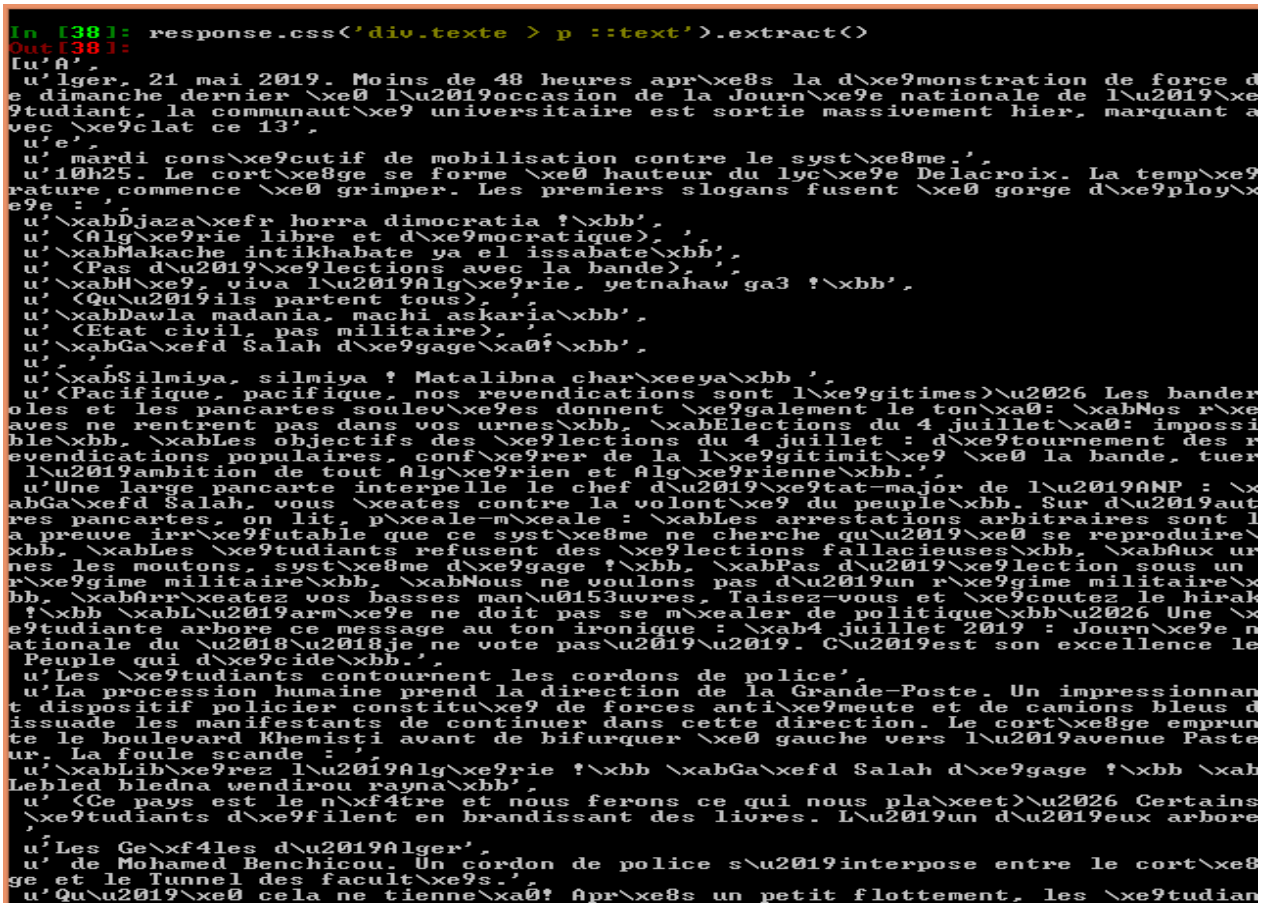
```
In [20]: response.css('div.author-tp-2 > a::text')[0].extract()
Out[20]: u'\nMustapha Benfodil '

In [21]:
```

Dans cette fonction on utilise la classe .author-tp-2 pour obtenir la valeur de la classe spécifié.

extraction de contenu de l'article :

Le contenu de l'article est dans une balise <p> , qui est le contenu de la balise mère définit par la classe css <div class=''texte''>

La fonction response() pour filtre le contenue de reste des composants de la page et comme suite :



- Apres avoir réussi à extraire tout nous données de notre site avec l'outil scrapy shell, en passe a la construction de notre araignée.

VI.2.3 Construction de l'araignée

Spider est une classe qui définit l'URL initiale à partir de laquelle extraire les données de notre site elwatan.com, éventuellement comment suivre les liens dans les pages. Et comment analyser le contenu de la page téléchargée pour extraire des données.

Dans notre cas nous étudierons comment extraire les liens de page principale, les suivre et extraire les données (titre , auteur ainsi contenu de chaque article) de cette page.

Et cela est illustre par l'araignée suivante :

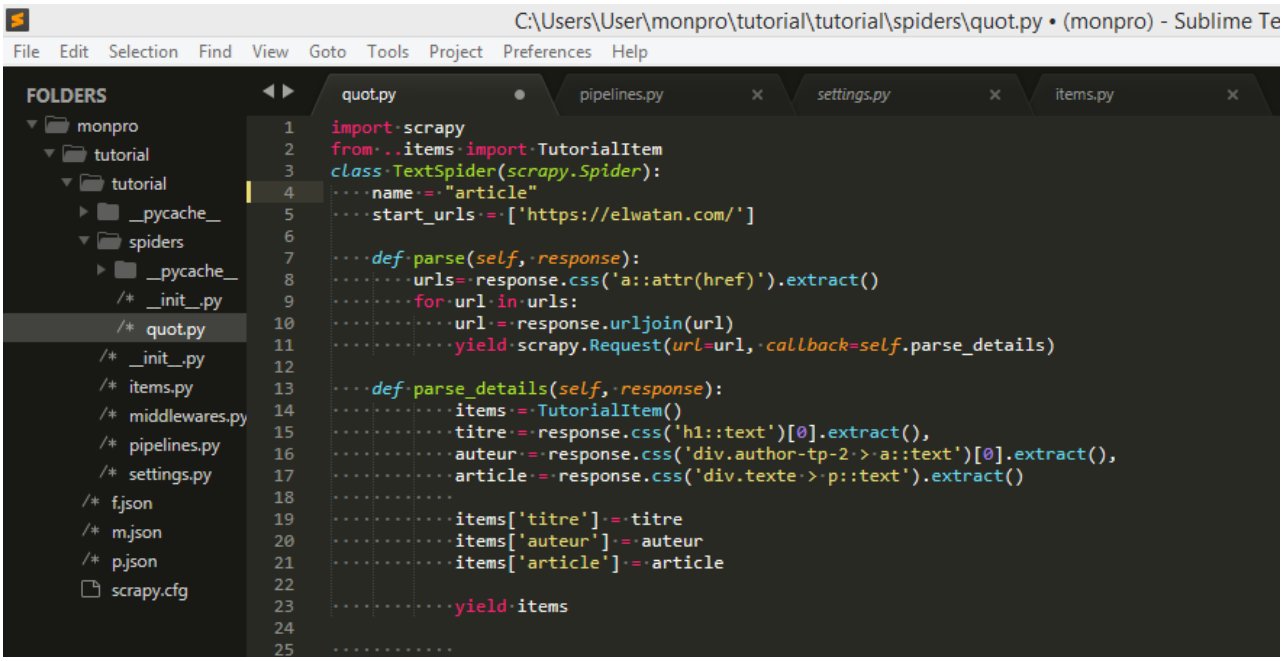


Figure 8:construction de l'araignée

name -: identifie l'araignée. Il doit être unique dans un projet, autrement dit, vous ne pouvez pas définir le même nom pour différentes araignées.

start-urls - Une liste d'URL à partir desquelles l'araignée commence à explorer, dans notre cas notre araignée va scraper l'url suivant https://elwatan.com .

parse () - C'est une méthode qui extrait l'ensemble des url de nous article.

response.urljoin - La méthode parse() utilisera cette méthode pour créer une nouvelle URL et fournir une nouvelle demande, qui sera envoyée plus tard au callback .

parse_details () - Ceci est un callback qui va réellement extraire les données dans notre site pour chaque article.

VI.2.3.1 Suivre les liens

Cette araignée commencera à partir de la page principale, elle suivra tous les liens vers les pages des article appelant le parse_details pour chacun d'eux, ainsi que les liens de pagination avec le callback de la methode parse .

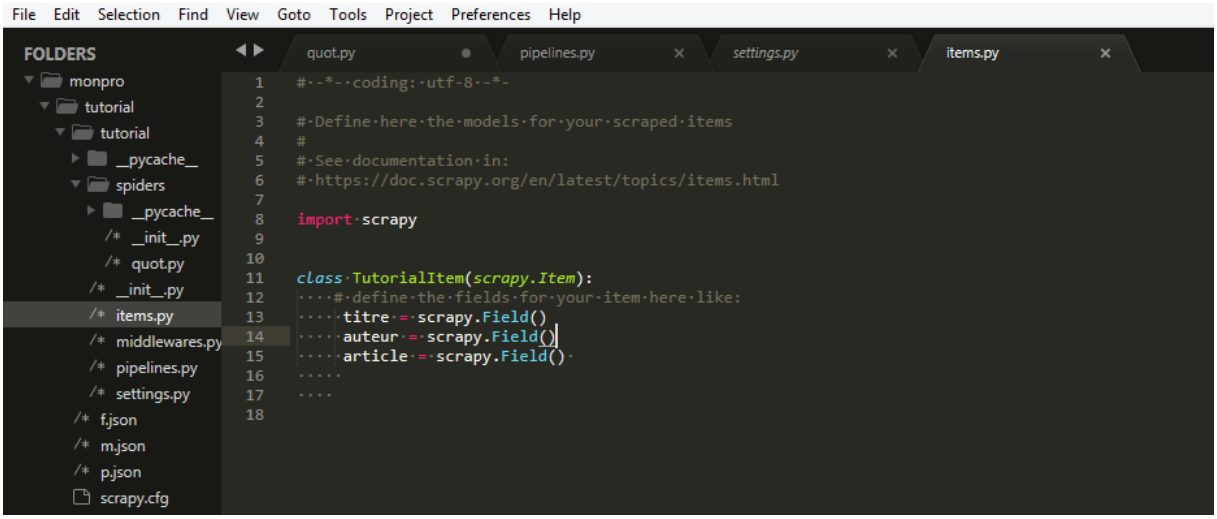
Scrapy utilise un mécanisme de callback pour suivre les liens. En utilisant ce mécanisme, on peut suivre des liens de notre site pour extraire les données souhaitées de différentes pages. Dans notre cas après avoir extrait les données, la méthode parse() recherche le lien vers la page suivante, crée une URL absolue à l'aide de la méthode urljoin() et génère une nouvelle demande à la page suivante, s'enregistrant comme callback à gérer. pour l'extraction des données pour la page suivante et pour continuer l'exploration à travers toutes les pages.

Ce que vous voyez ici est le mécanisme de suivi de liens de Scrapy: lorsque vous soumettez une demande dans une méthode de callback, Scrapy planifiera l'envoi de cette demande et enregistrera une méthode de callback à exécuter à la fin de cette demande.

Le callback parse_details définit une fonction d'assistance pour extraire les données d'une requête CSS et génère le dict Python avec les données de l'article.

VI.2.3.2 Déclaration d'items

Scrapy utilise la classe **Item** pour collecter les données récupérées. Les item sont déclaré a l'aide de la syntaxe suivante dans le fichier de déclaration d'item item.py :

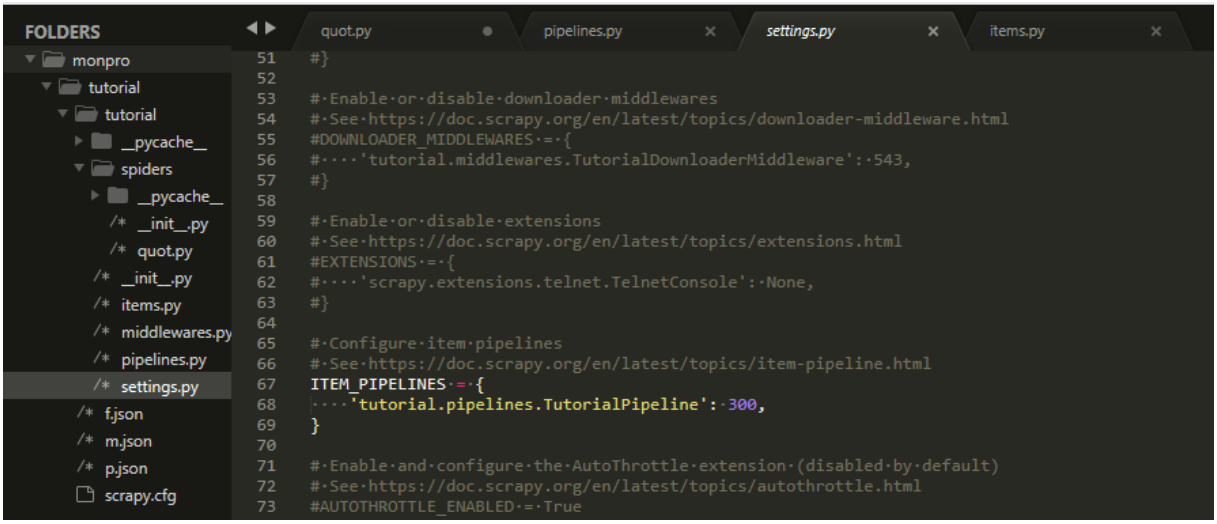


VI.2.3.3 Pipeline d'item

Dans notre exemple on va utiliser le pipeline d'item a fin de stocker les données récupéré dans une base de donnée MySQL .

Au premier lieu on va activer le pipeline d'item dans le fichier de paramètre de projet setting.py

Comme suite :



nous devons importer la classe TutorialItem. Qui réside dans le fichier item.py dans notre spider quot.py avec le code suivant :

```
from ..items import TutorialItem
```

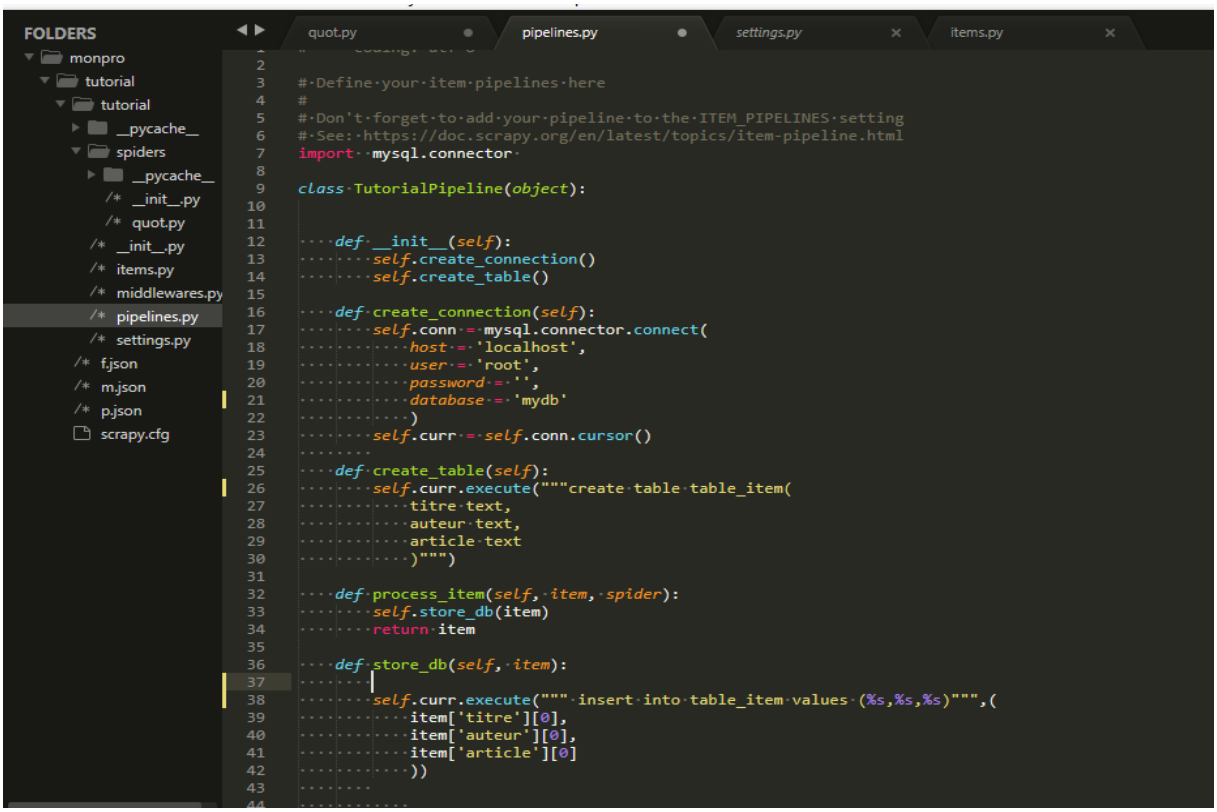
Ensuite, nous devons instancier l'objet item avec l'instruction suivante :

```
items = TutorialItem()
```

Qui crée un nouvel item, et alors nous pouvons assigner des expressions à ses champs comme suit:

```
titre = response.css('h1::text')[0].extract(),
auteur = response.css('div.author-tp-2 > a::text')[0].extract(),
article = response.css('div.texte > p::text').extract()

items['titre'] = titre
items['auteur'] = auteur
items['article'] = article

yield items
```

Enfin, nous retournons l'item avec yield items.
A fin d'ecrire nous items récupère dans une base de donne Mysql on utilise mysql.connector, Est l'ensemble des instructions pour effectue cette opération de stockage des item est définie dans le fichier pipelines.py comme suit :



Avec le fichier piplines.py on va réaliser le pipeline d'item a fin d'écrire nous donnée récupéré dans notre site grâce a l'araignée.

On va établir la connexion avec la base de donne mysql a l'aide de mysql.connector avec une methode creat_connection(self) defini comme suit :

```
def create_connection(self):
    self.conn = mysql.connector.connect(
        host = 'localhost',
        user = 'root',
        password = '',
        database = 'mydb'
    )
    self.curr = self.conn.cursor()
```

D'où le nom de la base de donne ou les données vont être enregistre est «mydb »
Maintenant on peut crée une table avec les donnée scrapy , est définir l'ensemble des champs de notre table et les remplir avec les donnée scrapy .

Création de la table avec la methode create_table (self) :

```
def create_table(self):
    self.curr.execute("""create table table_item(
        titre text,
        auteur text,
        article text
        )""")
```

On va crée une table « table_item » qui contiendra trois champ titre, auteur, et article qui vont représenter les item qu'on va récupérer dans notre site.

Chaque composant de pipeline d'éléments est une classe Python qui doit implémenter la méthode suivante:

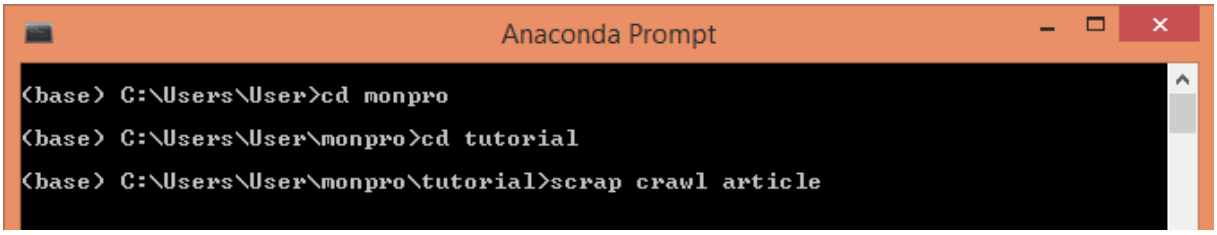
```
    def process_item(self, item, spider):
        self.store_db(item)
        return item
```

Cette méthode est appelée pour chaque composant d'item pipeline. Renvoie un objet de type item. Pour qu'il seras stocker dans notre table avec l'appelle a la méthode store_db(self ,item).

VI.2.4 L'exécution de l'araignée

Apres avoir défini notre araignée et identifier les items a extraire dans notre site, ainsi utiliser le pipeline d'items pour stocker nous item dans une base de donne. Voila comment exécuter notre araignée ainsi le résultat de notre scrapy :

On accède dans le répertoire « /tutorial » ou se trouve notre araignée, puis on exécute la commande suite :



« article» est le name de notre araignée, on peux aussi utiliser le nom de l'araignée « quot.py » pour l'exécuter.

Lorsque on exécute la commande scrapy run article, Scrapy recherche une définition de cette araignée.

L'exploration a commencé en envoyant des demandes aux URL définies dans l'attribut start_urls (dans ce cas, seul l'URL) et appelé l'analyse de la méthode perse par défaut, en transmettant l'objet de réponse comme attribut . Dans la méthode perse, nous parcourons les URL's de nous articles, a fin de planifier les prochaine demande qui sera envoyé a la méthode parse_details , qui parcoure les éléments de l'article à l'aide d'un sélecteur CSS, nous produisons un dict Python avec le titre de citation et l'auteur ainsi le contenu extraits à l'aide d'un sélecteur CSS, nous produisons un dict Python

l'un des principaux avantages de Scrapy: les demandes sont planifiées et traitées de manière asynchrone. Ce signifie que Scrapy n'a pas besoin d'attendre qu'une requête soit terminée et traitée, il peut envoyer une autre requête ou faire d'autres taches dans l'intervalle. Cela signifie également que d'autres demandes peuvent continuer, même si une demande échoue ou en cas d'erreur

Cela nous permet d'effectuer des analyses très rapides (envoi simultané de plusieurs demandes simultanées, dans un environnement à tolérance de pannes).

Conclusion

Dans ce chapitre, nous venons de cerner le potentiel de Scrapy en tant qu'outil de raclage. Ainsi nous avons mis en pratique un exemple qui illustre l'utilisation de cette outils, en termes d'efficacité et d'application pratique.

Chapitre III :

Etude et configuration d'Apach-nutch

Introduction

Apache Nutch est l'un des outils de web scraping, il est très robuste et évolutif pour l'exploration Web. Apache Nutch est présenté comme étant un outil utilisé pour extraire des données dans des applications quicontiennent d'énormes données afin de les exploiter.

Ce chapitre couvre l'introduction à Apache Nutch, ainsi que le fonctionnement de cet outil dans le raclage web et nous nous intéressonsàses avantages ainsi qu'à sa performance et ses différentes fonctionnalités et conclure avec un exemple applicatif.

I. Presentationde ApacheNutch

Apache Nutch[1] est un outil open source flexible et puissant pour l'analyse du Web, développé par Apache Software Foundation et sa communauté. Il s'appuie sur Apache Solr et intègre le très populaire Apache Hadoop , qui a commencé comme un sous-projet de Nutch. De nos jours, Nutch est un outil largement utilisé et le plus populaire de son créneau.

Son architecture hautement modulaire permet aux développeurs de créer des plug-ins pour l'analyse de type de média, la récupération de données, les requêtes et la mise en cluster. C'est là qu'Apache Solr entre en jeu. Solr est un cadre de recherche en texte intégral open source. Avec Solr, nous pouvons rechercher les pages visitées à partir de Nutch.

Le projet s'est diversifié et comprend désormais deux bases de code, à savoir:

- Nutch 1.x: Une chenille bien mûrie et prête pour la production. 1.x permet une configuration fine, en s'appuyant sur les structures de données Apache Hadoop , idéales pour le traitement par lots.Nutch fournit des interfaces extensibles telles que Parse pour Indexé et ScoringFilter pour les implémentations personnalisées, Apache Tika pour l'analyse. De plus, il existe une indexation pour Apache Solr, ElasticSearch, SolrCloud .
- Nutch 2.x: Une alternative émergente directement inspirée de 1.x, mais qui diffère d'un domaine clé à l'autre. le stockage est extrait de tout magasin de données sous-jacent spécifique en utilisant Apache Gora pour gérer les mappages persistants d'objet. Cela signifie que nous pouvons implémenter un modèle / pile extrêmement flexible pour tout stocker (temps de récupération, statut, contenu, texte analysé, liens sortants, liens entrants, etc.) dans un certain nombre de solutions de stockage NoSQL.

II. Notions de base sur Nutch :

[1]https://cwiki.apach.org/ - https://nutch.apach.org

Pour comprendre la documentation de nutch, il est nécessaire de connaitre les types de données qu'il manipule :

- La base de données d'analyse ou **crawldb** . Il contient des informations sur toutes les URL connues de Nutch
- La base de données de liens, ou **linkdb** . Cela contient la liste des liens connus vers chaque URL, y compris l'URL source et le texte d'ancrage du lien.
- Un ensemble de **segments** . un segment est une unité qui est un ensemble de page récupérées et indexées.Un segment est divisé en trois parties :
- **fetchlist :** une liste de page à récupérer
- **fetcher output :** la sortie des pages récupérées (ie. la réponse de chaque page avec les headers et le body)
- **index :** les indexes (au sens large non lucene) du fetcher output.

III.Architecture d'ApacheNutch

Le diagramme [2]suivant illustre l'architecture du plug-in Apache Nutch, qui vous montre comment fonctionne Apache Nutch. Ainsi,il développe les fonctions de recherche, d'indexation, de base de données Web et d'extraction. Il nous indique comment les données circulent d'une étape à l'autre. C'est un composant essentiel d'Apache Nutch. Par conséquent, la compréhension de cette architecture est indispensable pour comprendre le fonctionnement d'Apache Nutch.

L'image ci-dessous donne aperçu de l'architecture du plugin Apache Nutch.

[2] https://sites.google.com/site/nutch1936/home/introduction

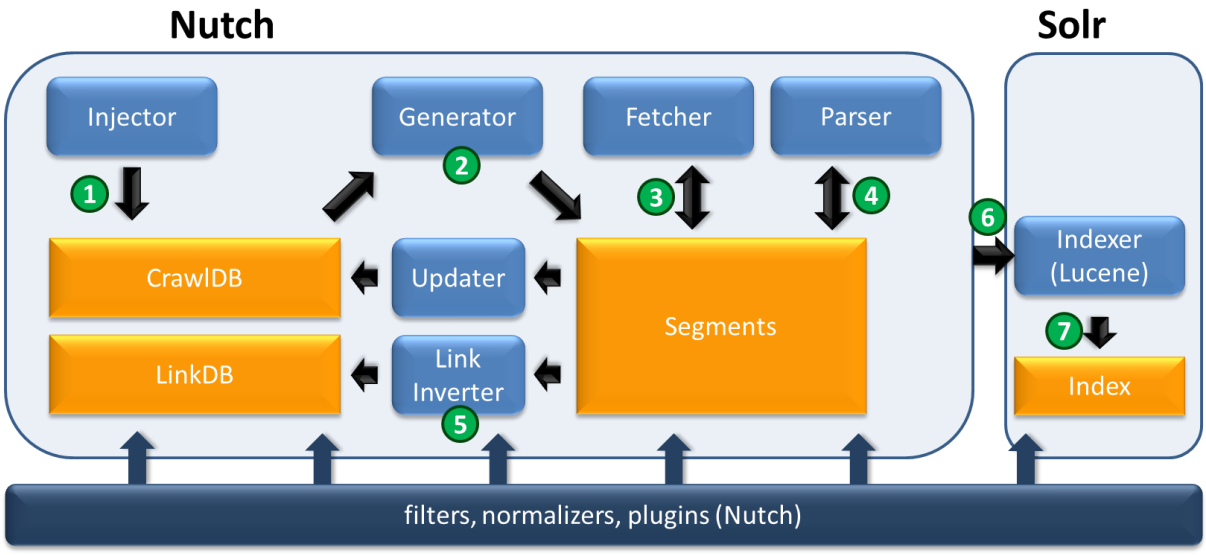


Figure 1 : architecture d'Apache-nutch

Les différentes étapes sont présentés par les flux suivants (numéroté de 1 à 7 comme indiqué dans le schéma précédant) :

1. L'injecteur prend toutes les URL du fichier nutch.txt et les ajoute à crawldb. En tant que partie centrale de Nutch, crawldb conserve des informations sur toutes les URL connues (calendrier de récupération, statut de récupération, métadonnées,…).

2. En fonction des données de crawldb, le générateur crée une liste d'extraction et la place dans un répertoire de segment nouvellement créé.

3. Ensuite, l'extracteur récupère le contenu des URL de la liste d'extraction et le réécrit dans le répertoire du segment. Cette étape est généralement la plus longue.

4. Maintenant, l'analyseur traite le contenu de chaque page Web et omet par exemple toutes les balises HTML. Si l'analyse fonctionne comme une mise à jour ou une extension d'une déjà existante (par exemple, une profondeur de 3), le programme de mise à jour ajoutera les nouvelles données à la crawldb à l'étape suivante.

5. Avant l'indexation, tous les liens doivent être inversés, ce qui tient compte du fait que ce n'est pas le nombre de liens sortants d'une page Web qui présente un intérêt, mais plutôt le nombre de liens entrants. Ceci est assez similaire au fonctionnement de Google PageRank et est important pour la fonction de scoring. Les liens inversés sont enregistrés dans la linkdb.

   6, 7. À l'aide de données provenant de toutes les sources possibles (crawldb, linkdb et segments), l'indexeur crée un index et l'enregistre dans le répertoire Solr. Pour l'indexation, la bibliothèque Lucene est utilisée. Désormais, l'utilisateur peut rechercher des informations concernant les pages Web analysées via Solr.

De plus, les filtres, les normalisateurs et les plugins permettent à Nutch d'être hautement modulaire, flexible et très personnalisable tout au long du processus. Cet aspect est également souligné dans l'image ci-dessus.

IV. Le cycle de vie de nutch

Le processus de nutch est divisé en plusieurs tâches, dont voici le cycle de vie standard :

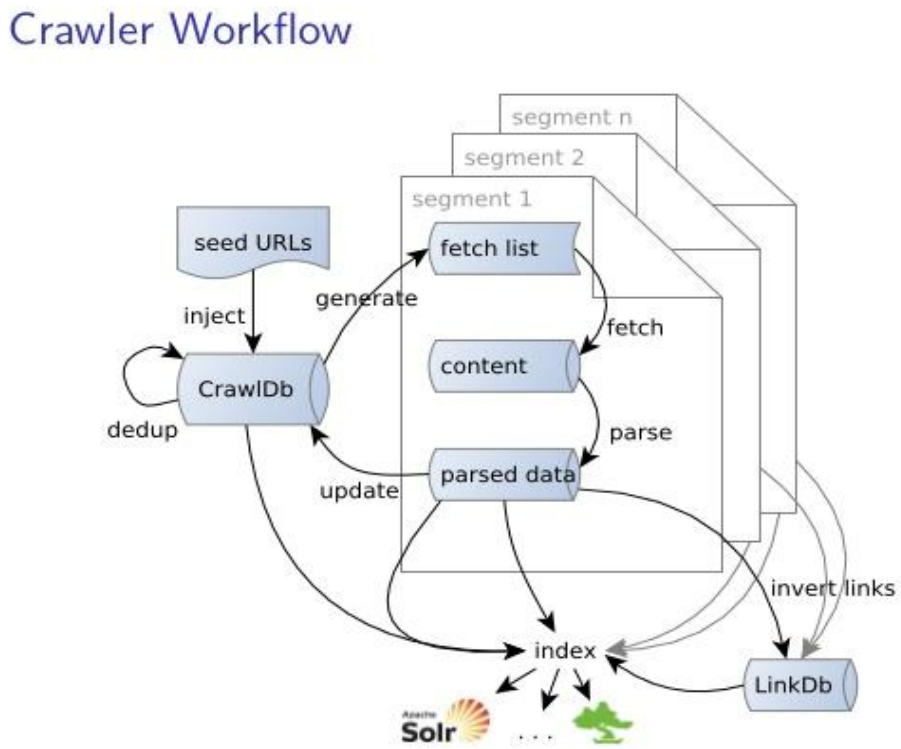


Figure 2: cycle de vie d'apach-nutch

- **Inject :** cette action permet d'injecter manuellement des urls dans la base crawldb de nutch. Lorsqu'on installe nutch, c'est la première étape a effectuer pour lui donner les urls des sites qu'on souhaite indexer.
- **Generate :** à partir de la liste des pages qu'il connait, nutch va créer une liste de page à récupérer (la fetchlist)

- **Fetch :** nutch va interroger chaque page de la fetchlist et enregistrer la réponse (avec le code http, les headers et le body).
- **Parse :** c'est l'opération d'indexation des pages, mais pas au sens lucene. A partir des données récoltées, nutch va venir les parser pour créer un objet/document structuré. Il extrait le titre, le type, les metadatas, l'url, … et les enregistre
- **UpdateDB :** c'est l'opération qui consitea mettre à jour la base de données des pages (la crawlDB) à partir des données déjà présente. Nutch va regarder le code html et en extraire tous les liens pour les enregistrer
- **InvertLinks :** permet à nutch de maintenir sa base de données de connaissances du web (la LinkDB), qui est un point essentiel de nutch.

V. Quelques limitations

Comme tout outil, il faut savoir quand ne pas l'utiliser. Voici une liste des limitations de nutch :

- **Page vs objet de contenu :** Nutch indexe les pages web, il n'indexe pas des objets de contenu. Vous ne pourrez donc pas faire avec nutch un moteur de recherche spécialisé sur un type de contenu. (A moins de faire du web sémantique).
- **Contenu sécurisé :** Nutch est un robot, il parse votre site comme un internaute lambda, il n'est pas authentifié, et donc il ne peut pas avoir accès à du contenu sécurisé. Même s'il est possible d'authentifié nutch pour votre site, il ne va pas gérer(nativement) les droits d'accès lors de son indexation.
- **Asycnhrone :** Comme tout crawler, l'indexation de votre site n'est pas immédiate. Pour voir un nouveau contenu (ou sa suppression) pris en compte, il faut attendre le prochain passage du robot
- **Système d'hyperlien :** pour qu'une page soit indexée, il faut qu'un lien pointe vers elle depuis une autre page. Si une page, ou une sous-arborescence, est indépendante du point d'indexation de votre site, celle-ci ne sera jamais indexée.

VI. Caractéristique de nutch :

- L'extraction et l'analyse sont effectuées séparément par défaut, ce qui réduit le risque d'erreur avec Nutch.
- Le nombre de plug-ins permettant de traiter différents types de documents expédiés avec Nutch a été précisé :

  -Texte brut (plugin: tika),

  -HTML / XHTML + XML (parse-html / tika),

  -javaScript (pour extraire des liens uniquement?) (Parse-js)

  -Microsoft Power Point, le fichier .ppt (parse-tika)

  -Microsoft Word, le fichier .doc (parse-tika)

  -Adobe PDF (parse-tika)

  -RSS (parse-feed / tika)

  -RTF (parse-tika)

  -MP3 (parse-tika) Le mp3 contient les balises ID3v1 ou ID3v2 qui contiennent des informations sur les chansons de métadonnées, telles que (titre, artiste, album, commentaires, etc. Les informations utiles pour la recherche de mp3)

  -ZIP (parse-zip) Cela semble élargir le zip des fichiers texte brut et renvoyer le texte concaténé.

- Utilise la méthodeMapReduce : il permet de traiter des ensembles de données volumineux de manière distribuée en décomposant le traitement en plusieurs petits calculs
- Système de fichiers distribué (via Hadoop)
- Base de donne Link-graph
- Authentification NTLM

VII. Quelques Composants de nutch

Nutch suit les structures des plugins et fournit des interfaces pour de nombreux composants populaires qui peuvent être utilisés selon les besoins. Par exemple :

1. **hadoop :**

Hadoop est unframeworkopen source qui repose surJava. Hadoop prend en charge le traitement des données volumineuses (Big Data) au sein d'environnements informatiques distribués. Hadoop fait partie intégrante du projet Apache parrainé par l'Apache Software Foundation.

2. **Apache Tika -** une boîte à outils d'analyse de contenu

La boîte à outils Apache Tika ™ détecte et extrait les métadonnées et le texte de plus de mille types de fichiers différents (tels que PPT, XLS et PDF). Tous ces types de fichiers peuvent être analysés via une interface unique, ce qui rend Tika utile pour l'indexation des moteurs de recherche, l'analyse de contenu, la traduction, etc

3. **Apache Solr**

Est la plate-forme de recherche d'entreprise open source populaire et ultra-rapide, basée sur Apache Lucene, qui permetl'indexation de fichiers ou de bases de données ainsi que des sites web.

4. **Elasticsearch** :

Elasticsearch est un moteur de recherche et d'analyse distribué, open source, RESTful, basé sur Apache Lucene.Il est rapidement devenu le moteur de recherche le plus populaire. Il est couramment utilisé pour l'analyse de journaux, la recherche en texte intégral, les informations de sécurité, les analyses commerciales et les cas d'utilisation des informations opérationnelles.

VIII. Implémentation et évaluation :

VIII.1 Outil de développement

Nous présentant les différents outils utilisés pour l'implémentation de notre approche :

VIII.1.1 Apach nutch 1.x :

Apach-nuch[3] est un outil bien prêt pour la production, 1.x permet une configuration fine, en s'appuyant sue les structures de données Apach hadoop, idéales pour le traitement par lots.

VIII.1.2 Le Java Development Kit :

Le Java Development Kit désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java. Il existe plusieurs éditions de JDK, selon la plate-forme Java considérée.

VIII.1.3 Cygwin :

Cygwin[4] est une collection de logiciels libres à l'origine développés par Cygnus Solutions permettant à différentes versions de Windows de Microsoft d'émuler un système Unix. Il vise principalement l'adaptation à Windows de logiciels qui fonctionnent sur des systèmes POSIX.

VIII.1.4 ApacheTomcat

Apache Tomcat[5] est un outil de serveur Web open source développé par Apache Software Foundation (ASF). C'est l'un des nombreux produits open source liés à Apache utilisés par les professionnels de l'informatique pour diverses tâches et objectifs**.** Apache Tomcat permet l'implémentation de Java Servlets et de JSP (JavaServer Pages) afin de promouvoir un environnement de serveur Java efficace. Les utilisateurs peuvent également accéder aux ressources pour la configuration et utiliser un langage de balisage extensible (XML) pour configurer des projets.

VIII.1.5 Apache SOLR

Solr[6] est la plate-forme de recherche d'entreprise open source populaire et flambante de ApacheLuceneproject. Ses principales fonctionnalités comprennent la recherche en texte intégral, la mise en évidence des occurrences, la recherche par facettes, l'indexation en temps quasi réel, la mise en cluster dynamique, l'intégration de la base de données, la gestion de documents enrichis (Word, PDF, par exemple) et la recherche géo spatiale. Solr est extrêmement fiable, évolutif et tolérant aux pannes. Il fournit une indexation, une réplication et une interrogation équilibrées de la charge. Solr est le moteur des fonctions de recherche et de navigation de plusieursdes plus grands sites Internet du monde.

[3] https://cwiki.apach.org/
[4] http://cygwin.com/instal.html
[5] http://tomcat.apach.org
[6] http://idodevjobs.wordpress.com

Dans la section suivante, nous avons procédé à la configuration et l'installation locale d'Apache Nutch.

VIII.2 L'installation et la configuration des outils

Dans notre études nous avons opté a utilisé nutch 1.x une version qui existe depuis beaucoup plus longtemps, de plus il a des fonctionnalités plus avancés et de nombreuses corrections de bug par rapport à nutch 2.x.

nutch 2.x et nutch 1.x sont assez différents en termes de configuration, d'exécution et d'architecture. nutch 2.x utilise Apache Gora en tant que couche d'abstraction de stockage , ce qui permet d'utiliser différentes versions de base de données Nosql telles que Hbase, Cassandra. En revanche nutch 1.x est considéré pour les recherches qui sont plus avancé , et nutch 2.x est utilisé si la flexibilité des magasins de base de données est importante.

Téléchargements

- JDK 7 - jdk-7u55-windows-x64.exe
- Cygwin - setup-x86_64.exe
- Apache Tomcat - apache-tomcat-7.0.94-windows-x64.zip
- Apache SOLR 4.8 - solr-4.8.0.zip
- Apache Nutch 1.4 - apache-nutch-1.4 -bin.zip

Installation JDK 7:

- Exécuter l'exécutable téléchargé pour installer Java à l'emplacement souhaité.
- Définir la variable d'environnement JAVA_HOME.

Installation de Cygwin :

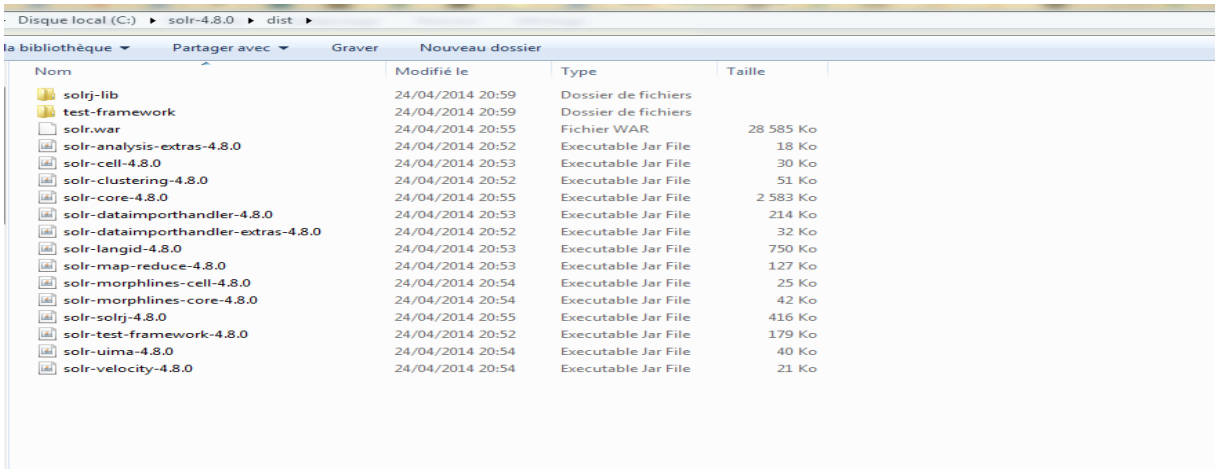
- Télécharger et installer cygwin dans n'importe quel répertoire.

Installation de ApacheTomcat 7 :

- Télécharger et extraire le fichier .zip à nimporte quel emplacement .

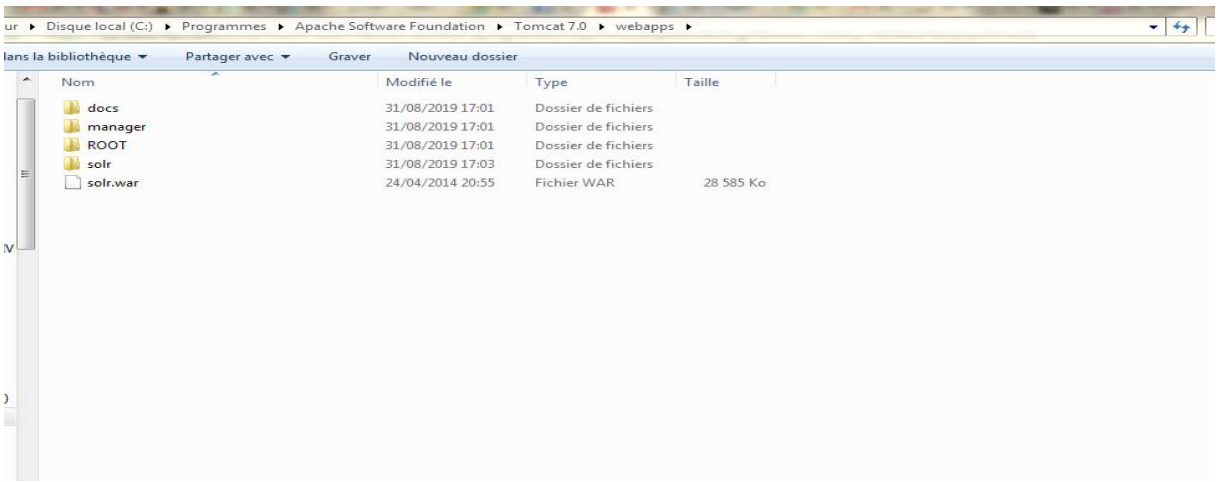
VIII.2.1 Configuration d'Apache SOLR :

- Télécharger et extraire le fichier zip SOLR à n'importe quel emplacement.
- Déployer le fichier solor.war dans le dossier TomcatWebapps :
  localiser le dossier « \ **dist** », Dans le dossier dist, rechercher le **fichier** « **solr-4.2.0.war** ». Il s'agit essentiellement du fichier compressé contenant tout le binaire nécessaire au conteneur de servlets pour exécuter l'application Web.



Renommer le fichier en **solr.war** , car ce sera le nom de l'URL permettant d'accéder à Solr.

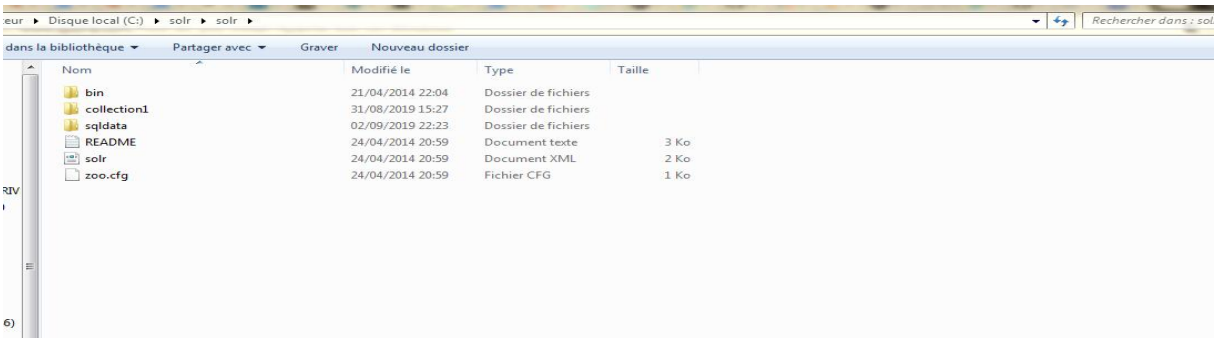
Déplacer ce fichier dans le dossier « **webapps** » de Tomcat. Ce dossier se trouve dans le dossier d'installation de Tomcat pour que solr s'execute dans tomcat.



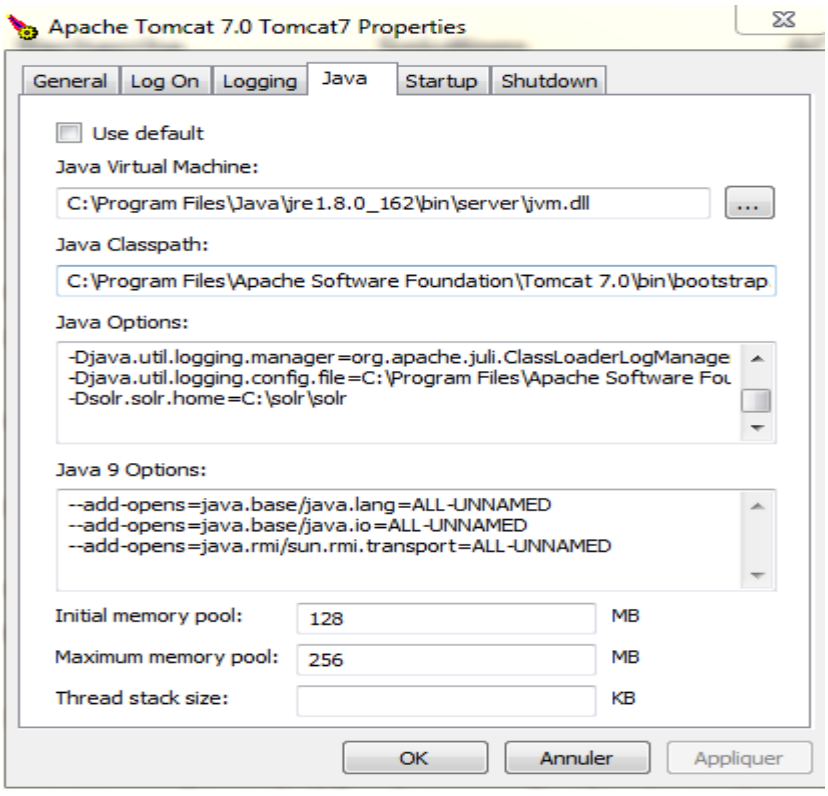
Si le service tomcat est en cours d'exécution, vous constaterez qu'un répertoire solr est automatiquement créé. Si non il sera créé au prochain démarrage de tomcat.

- Créer une maison Solr :

Créer le répertoire / solr / solr à n'importe quel emplacementoù vous souhaitez stocker les fichiers de configuration Solr, le fichier d'index Solr et Solrcore .par exemple c: / solr / solr.
Copier le contenu de /solr-4.8.0.zip/solr-4.8.0/example/solr/ dans le répertoire créé.

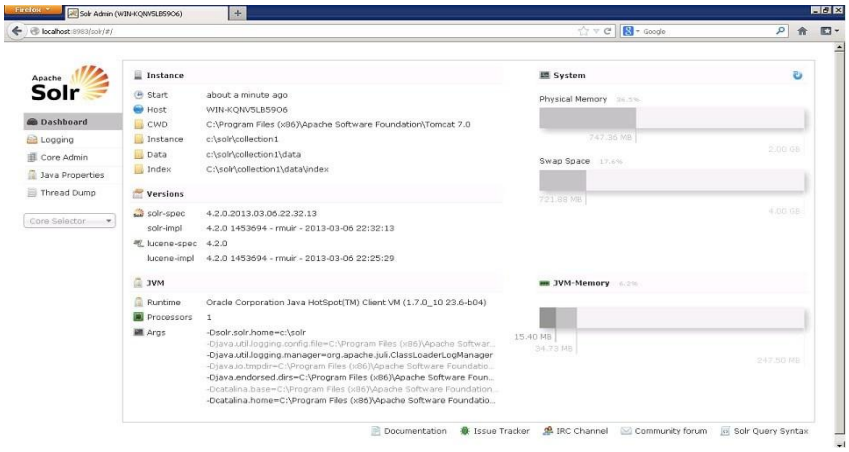


- Copier /solr-4.8.0.zip/solr-4.8.0/example/solr/collection1/conf/lang/stopwords_en.txt dans le répertoire c: / solr / solr / collection1 / conf /.
- Copier tous les fichiers JAR du répertoire /solr-4.8.0.zip/solr-4.8.0/example/lib/ext/ dans le répertoire /apache-tomcat-7.0.53/lib/
- Configuration de Tomcat pour détecter Solr :
  prochaine étape consiste à indiquer à Tomcat où le dossier Solr Home est placé. Dans la fenêtre de configuration de Tomcat et on accède à l'onglet Java, localiser la zone de texte Options Java et entrer cette ligne à la fin de tous les éléments déjà présents:
  Dsolr.solr.home = c: \ solr



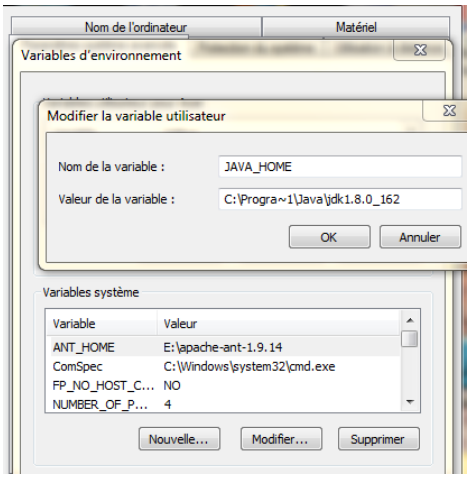
- Test :
  On lance un navigateur et on appuie sur « http: // localhost: 8080 / solr » (le numéro de port et le nom de l'application peuvent différer en fonction de l'installation de Tomcat et du nom du dossier que nous avons configuré) pour voir si la page d'administration de adminsolr apparaît.

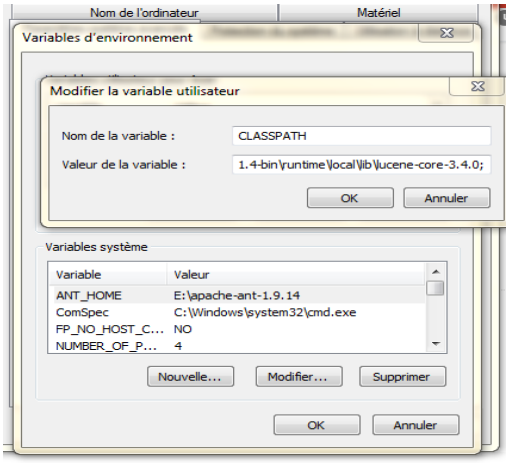
La page sera semblable à ceci:



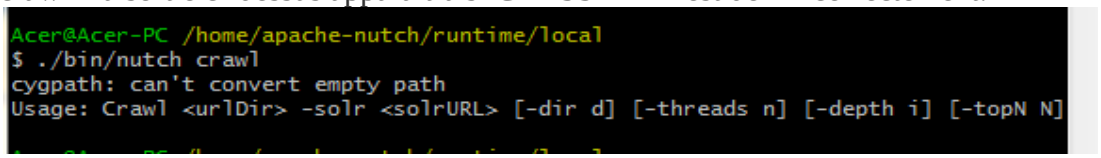
VIII.2.2 Configuration d'Apache Nutch

- Télécharger et extraire le fichier zip Nutch à n'importe quel endroit.
- Définir les variables d'environnements JAVA_HOME , CLASS_PATH.





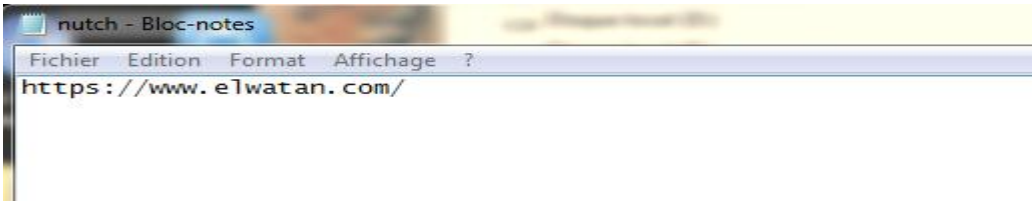
- pour vérifier que les chemins sont correctement définis, on tape ce qui suit: './bin/nutch crawl' La sortie ci-dessus apparaîtra si CLASSPATH est défini correctement.



Nutch est maintenant prêt à explorer et à indexer

- Création d'un répertoire avec un nom, par exemple "urls" dans le répertoire /apache-nutch-1.4-bin/runtime/local/.

- Création d'un fichier texte avec n'importe quel nom. On ajoute la liste des sites Web à explorer.



Remarque : pour démarrer l'analyse à partir de plusieurs URL, on peut ajouter plusieurs fichiers contenant les URL à analyser.

- Edition du fichier /apache-nutch-1.4-bin/runtime/local/conf/regex-urlfilter.txt – on spécifiant le nom de domaine que nous souhaitons analyser.

```
# skip file: ftp: and mailto: urls
-^(file|ftp|mailto):

# skip image and other suffixes we can't yet parse
# for a more extensive coverage use the urlfilter-suffix plugin
-\.(gif|GIF|jpg|JPG|png|PNG|ico|ICO|css|CSS|sit|SIT|eps|EPS|wmf|WMF|zip|ZIP|ppt|PPT|mpg|MP

# skip URLs containing certain characters as probable queries, etc.
-[?*!@=]

# skip URLs with slash-delimited segment that repeats 3+ times, to break loops
-.*(/[^/]+)/[^/]+\1/[^/]+\1/

# accept anything else
+^https://([a-z0-9\-A-Z]*\.)*www.elwatan.com/([a-z0-9\-A-Z]*\/)*
```

- Editions du fichier /nutch-site.xml et ajout des deux propriétés suivantes comme indiqué ci-dessous.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
         <name>http.agent.name</name>
         <value>my nutch spider</value>
         <description>nutch-crawler</description>


<name>plugin.includes</name>
  <value>protocol-http|protocol-httpclient|urlfilter-
regex|parse-(html)|index-(basic|anchor)|indexer-solr|query-
(basic|site|url)|response-(json|xml)|summary-basic|scoring-
opic|urlnormalizer-(pass|regex|basic)</value>

</property>


</configuration>
```

Pour permettre àNutch d'explorer les sites web sécurisés

- Lancement decygwin et pointer sur le répertoire apache-nutch-1.4-bin. Exécuter la commande suivante pour vérifier si Nutch fonctionne correctement.

```
1        ./bin/nutch crawl urls -dirmyCrawl -depth 3 -topN 4
```

```
Acer@Acer-PC /home/apache-nutch/runtime/local
$ ./bin/nutch crawl urls -dir myCrawl -depth 3 -topN 4
cygpath: can't convert empty path
solrUrl is not set, indexing will be skipped...
crawl started in: myCrawl
rootUrlDir = urls
threads = 10
depth = 3
solrUrl=null
topN = 4
Injector: starting at 2019-09-07 16:43:36
Injector: crawlDb: myCrawl/crawldb
Injector: urlDir: urls
Injector: Converting injected urls to crawl db entries.
Injector: Merging injected urls into crawl db.
Injector: finished at 2019-09-07 16:43:41, elapsed: 00:00:05
Generator: starting at 2019-09-07 16:43:42
Generator: Selecting best-scoring urls due for fetch.
Generator: filtering: true
Generator: normalizing: true
Generator: topN: 4
Generator: jobtracker is 'local', generating exactly one partition.
Generator: Partitioning selected urls for politeness.
Generator: segment: myCrawl/segments/20190907164344
Generator: finished at 2019-09-07 16:43:46, elapsed: 00:00:04
Fetcher: Your 'http.agent.name' value should be listed first in 'http.robots.ag
Fetcher: starting at 2019-09-07 16:43:46
Fetcher: segment: myCrawl/segments/20190907164344
Using queue mode : byHost
Fetcher: threads: 10
Fetcher: time-out divisor: 2
QueueFeeder finished: total 1 records + hit by time limit :0
Using queue mode : byHost
Using queue mode : byHost
fetching https://www.elwatan.com/
Using queue mode : byHost
-finishing thread FetcherThread, activeThreads=1
Using queue mode : byHost
Using queue mode : byHost
Using queue mode : byHost
```

VIII.2.3 Intégration d'Apache Nutch - Apache SOLR

-Copier le schéma Nutch fourni du répertoire
apache-nutch-1.0 / conf dans le répertoire apache-solr-1.3.0 / example / solr / conf (remplacez le fichier existant).
Nous souhaitons autoriser Solr à créer les extraits de code pour les résultats de recherche.
Nous devons donc stocker le contenu en plus de l'indexation:
-Modifier le fichier schema.xml afin que l'attribut stocké du champ «contenu» soit vrai.

- Renommer le fichier schema.xml dans le répertoire / solr / collection1 / conf / en un nom quelconque. Copier le fichier /apache-nutch-1.4-bin/runtime/local/conf/schema-solr4.xml dans le répertoire / solr / solr / collection1 / conf, renommer le fichier en tant que schema.xml. Editez le fichier schema.xml copié pour qu'il contienne

```
<fieldname="_version_"type="long"indexed="true"stored="true"/>
```

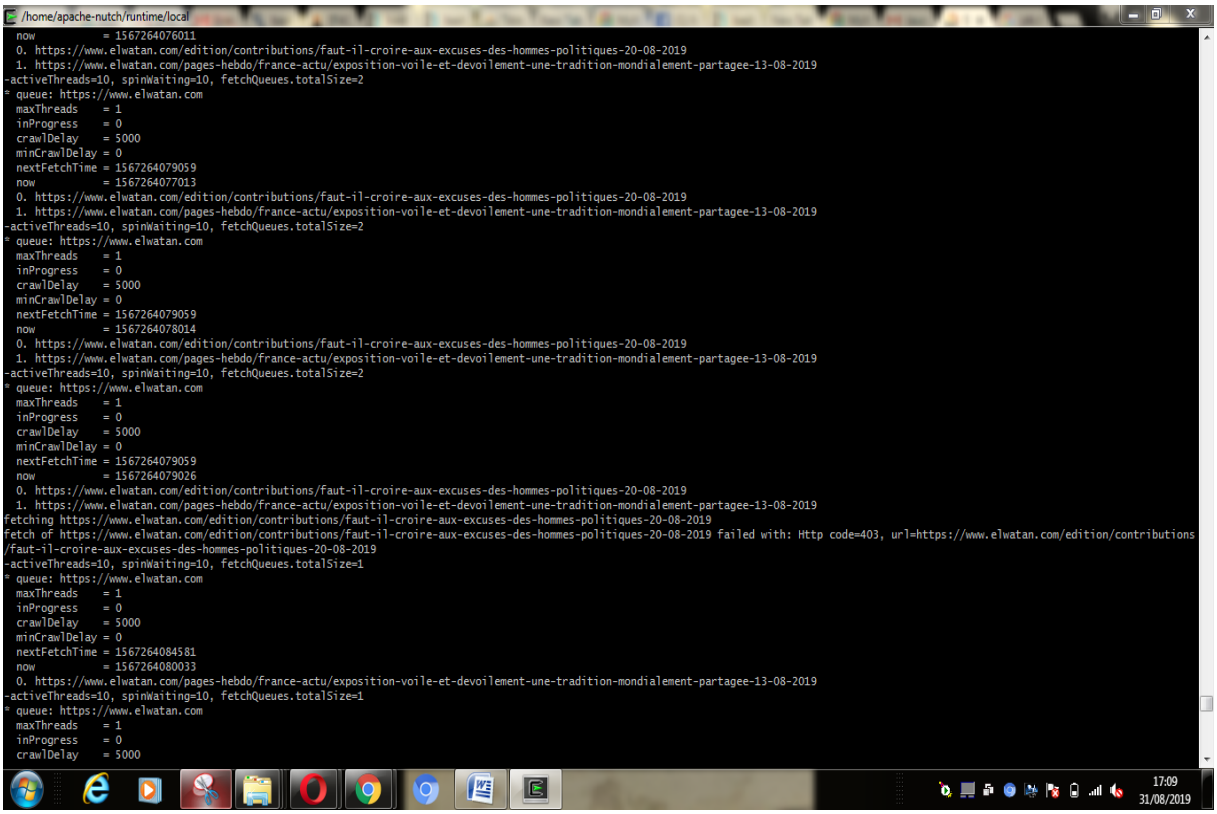
- Redémarrer le serveur tomcat.

- Lancer cygwin et pointer sur le répertoire apache-nutch-1.4-bin. Exécuter la commande suivante pour que l'analyse soit effectuée à l'aide de Nutch et pour vider les données dans solr.

```
./bin/nutch crawl urls -dirmyCrawl -solrhttp://localhost:8080/solr/ -depth 3 -topN 4
```

Cette commande comprend les options suivantes :

-dir *dir* nomme le répertoire dans lequel l'exploration doit être effectuée.
-threads *threads* détermine le nombre de threads à récupérer en parallèle.
-depth *profondeur* indique la profondeur du lien à partir de la page racine à explorer.
-topN *N* détermine le nombre maximum de pages qui seront récupérées à chaque niveau jusqu'à la profondeur.



- Lancer un navigateur et appuyer sur http: // localhost: 8080 / solr pour voir si la page d'administration de adminsolr apparaît. Tester comme indiqué ci-dessous.

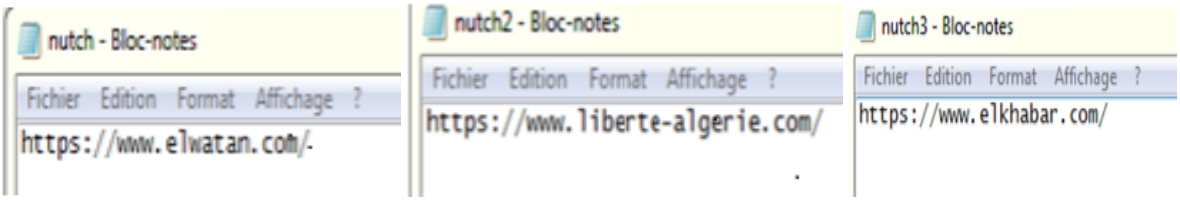
Figure 3: l'interface d'Apach-solr

VIII.3 Implémentation

Examinons maintenant une étude d'un cas qui porte sur l'utilisation de l'outil Nutch ainsi que ses composants afin d'acquérir plus d'expérience et mettre en œuvre ses divers fonctionnalités.

Dans notre cas, on procède à l'exploration du site '' elwatan.com '' , ''liberté.com'' et ''elkhaber.com'' afin d'extraire des articles de presse à partir de ces derniers et les afficher dans un format structuré.

On commence par créer le fichier urls dans le répertoire nutch contenant des fichiers texte qui portent l'URL de la page d'accueil des trois sites .



Pour démarrer l'analyse à partir de plusieurs URL, on peut ajouter plusieurs fichiers contenant les urlsà analyser.

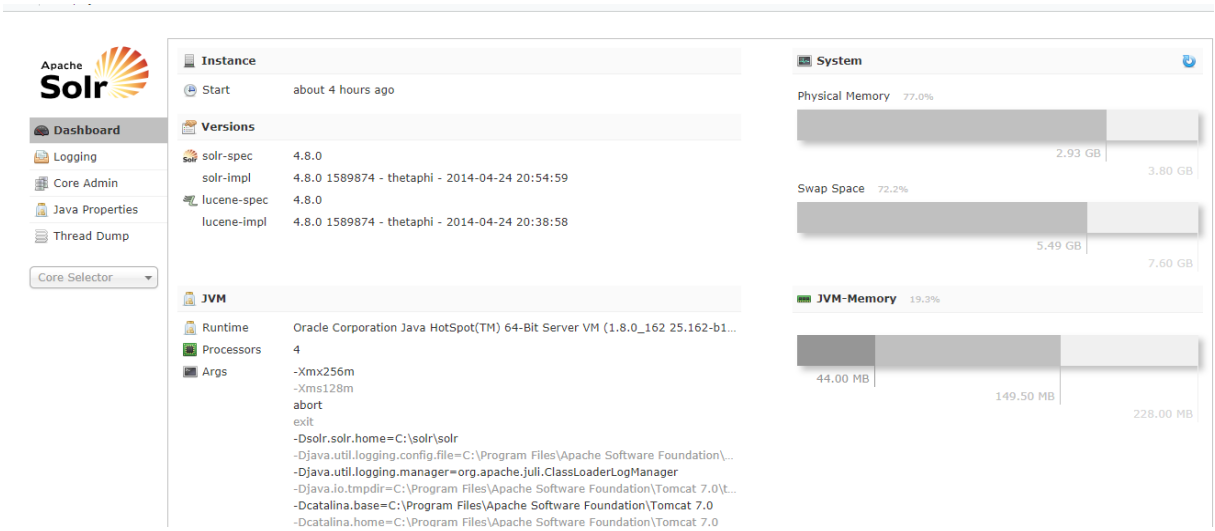
Une fois l'url du site à analyser est indiqué, on modifie le fichier conf/crawl-urlfilter.txt et remplacer MY.DOMAIN.NAME par le nom du domaine qu'on souhaite analyser comme suit :

```
# determines whether a URL is included or ignored.  If no
pattern
# matches, the URL is ignored.

# skip file: ftp: and mailto: urls
-^(file|ftp|mailto):

# skip image and other suffixes we can't yet parse
# for a more extensive coverage use the urlfilter-suffix
plugin
-\.
(gif|GIF|jpg|JPG|png|PNG|ico|ICO|css|CSS|sit|SIT|eps|EPS|wmf|W
MF|zip|ZIP|ppt|PPT|mpg|MPG|xls|XLS|gz|GZ|rpm|RPM|tgz|TGZ|mov|M
OV|exe|EXE|jpeg|JPEG|bmp|BMP|js|JS)$

# skip URLs containing certain characters as probable queries,
etc.
-[?*!@=]

# skip URLs with slash-delimited segment that repeats 3+
times, to break loops
-.*(/[^/]+)/[^/]+\1/[^/]+\1/

# accept anything else
+^https://([a-z0-9\-A-Z]*\.)*www.elwatan.com/([a-z0-9\-A-Z]*
\/)*
+^https://([a-z0-9\-A-Z]*\.)*www.liberte-algerie.com/([a-z0-9
\-A-Z]*\/)*
+^https://([a-z0-9\-A-Z]*\.)*www.elkhabar.com/([a-z0-9\-A-Z]*
\/)*
```

Une fois la configuration est terminée, on lance l'exploration du site avec SIGWIN en utilisant la commande suivante pour que l'analyse soit effectuée à l'aide de nutch et pour vider les données dans solr

```
Acer@Acer-PC /home/apache-nutch/runtime/local
$ ./bin/nutch crawl urls -dir myCrawl -solr http://localhost:8181/solr/ -depth 3 -topN 4
```

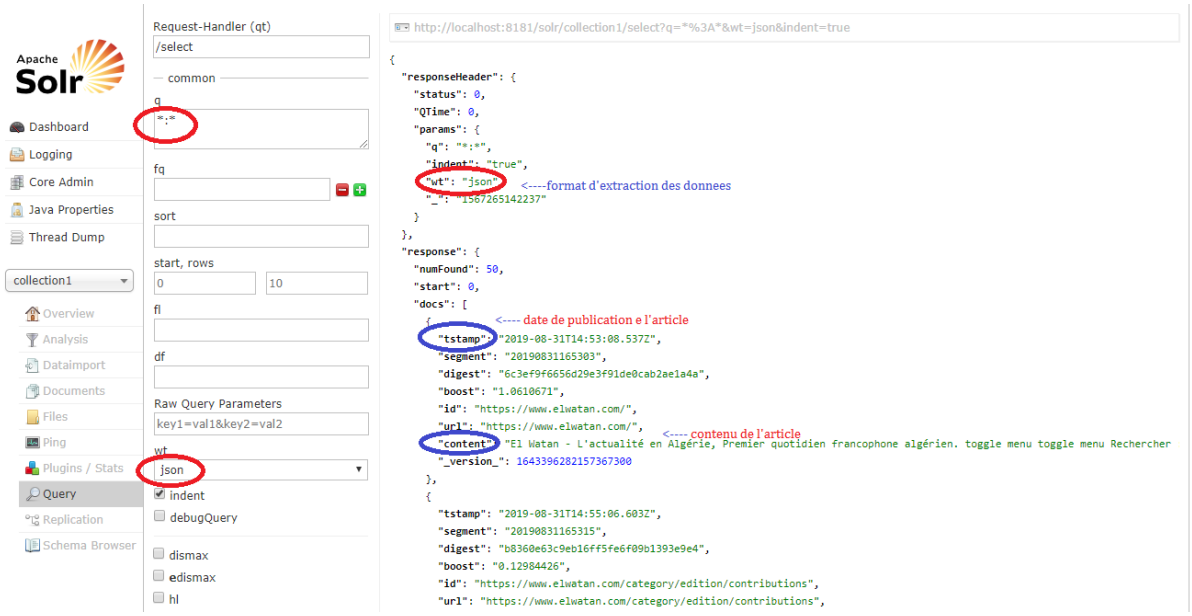
Dans tout ce qui suit, nous allons utiliser l'interface query de l'application Web d'administration de Solr, accessible démarré à partir de Tomcatpar le lien http://localhost:8983/solr. C'est un moyen simple de tester l'interrogation.Vous obtiendrez l'image de l'exécution d'Apache Solr sur le navigateur, comme illustré dans la capture d'écran suivante:

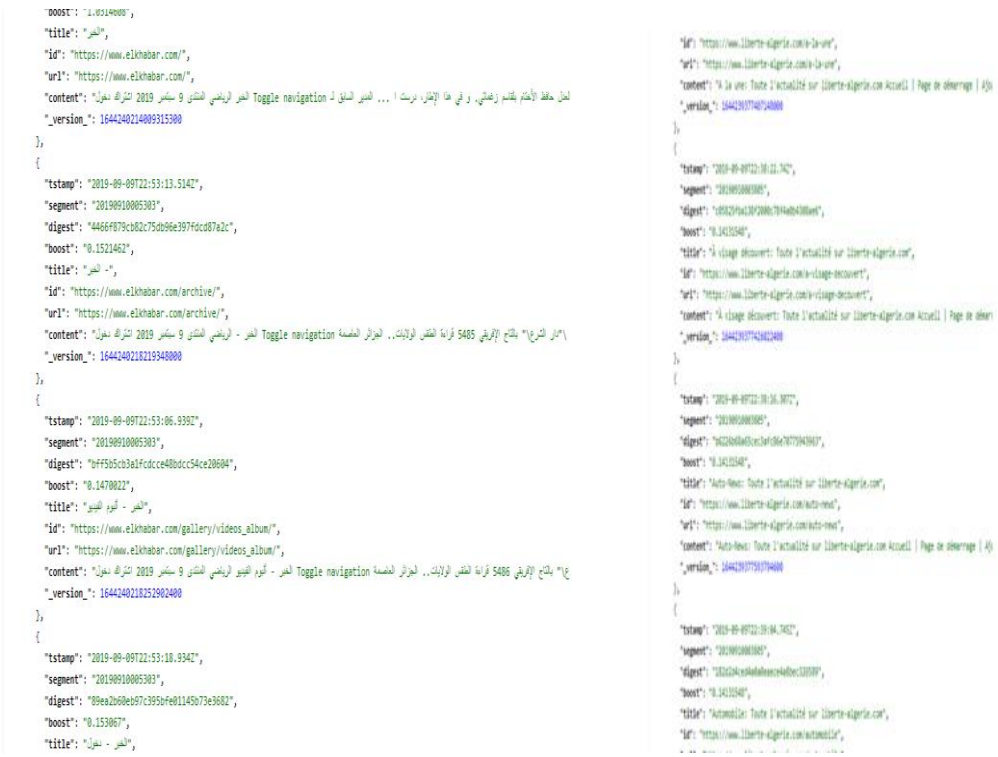


Le principal paramètre que nous allons utiliser est simplement nommé **q** pour *query*. Par défaut, il est proposé dans l'interface avec la valeur *.*ce qui permet de ramener *tous* les documents de l'index.

Les autres paramètres sont, en bref:

- fq: pour *filterquery*, un moyen d'interroger non pas l'index entier mais un résultat pré-calculé et stocké en *cache*;
- sort, pour trier le résultat;
- start et row, les paramètres classiques de pagination du résultat;
- fl pour *fieldlist*, indique la liste des champs (stockés) à inclure dans le résultat;
- df, le champ à interroger si non spécifié dans la requête (la valeur par défaut est indiqué dans la configuration et vaut en principe text, le champ dans lequel nous avons concaténé toutes nos chaînes de caractères);
- enfin, on trouve la liste des *queryparsers* disponibles; un *queryparser* correspond à une syntaxe d'interrogation particulière(json, xml, php, csv).





Analysons la réponse transmise par Solr. Elle comprend un en-tête donnant quelques propriétés sur l'exécution (notamment le temps de réponse).

Dans notre étude on s'intéresse uniquement àl'extraction du contenu de l'article ainsi que la date de la publication de ce dernier.

Et cela se fait grâce au paramètre ''fl'' pour indiquer les deux champs « content » et « tstamp»

Et les paramètres de l'interface query s'écrivant comme suit :

- associé nutchà Solr, est une base très puissantepour explorer et extraire le contenu des sites webs.Nutchest un robot très évolutif et relativement riche en fonctionnalités généralement utilisé pour leraclage web.L'analyse est effectuée à l'aide de l'outil d'analyse Apache Nutchde plus il est extensible ce qui nous permet de connecter ànutch l’applicationsolr qui sera utilisée comme seule source pour la diffusion des résultats de recherche (y compris les extraits).

Conclusion

Dans ce chapitre, nous avons commencé avec Apache NUTCH en couvrant son introduction et d’autres composants associés: tel que solr. Nous avons vu en détails comment configurer Apache NUTCH à l’aide de tous les composants mentionnés. Nous avons également effectué l'exploration à l’aide d’Apache nutch. Après cela, nous avons commencé à intégrer Apache Nutch à Solr, Ainsi nous avons mis en pratique un exemple qui illustre l’utilisation de cet outil, en termes d’efficacité et d’application pratique.

Chapitre IV :

Etude comparative de scrapy et apach-nutch

Introduction

Dans les chapitres précédents nous avons réalisé une étude sur le web scraping afin de recueillir les données sur les sites web de façon structurées. Il existe plusieurs programmes et outils permettant de faire du web scraping. Ils se différencient par leur utilisation.

Dans notre quête d'un bon outil de raclage web, et après quelques recherches initiales, on a limité le choix aux deux systèmes qui semblaient être les plus avancés et les plus utilisés: Scrapy (Python), et Apache Nutch (Java).

Comme point de départ, on doit vérifier que les services d'exploration choisis répondent aux propriétés décrites dans l'analyse web.

I. Caractéristiques d'un bon outil de web scraping

Nous répertorions ci-après les caractéristiques et les fonctionnalités principales que les outils de web scraping doivent fournir, afin de définir les bons outils pour le raclage web.

- **Robustesse** : le Web contient des serveurs qui créent des interruptions d'araignées, générateurs de pages Web qui induisent les robots d'exploration en erreur en les obligeant à récupérer un nombre infini de pages dans un domaine particulier. Les robots d'exploration doivent être conçus pour résister à de tels pièges. Tous ces pièges ne sont pas malveillants; certains sont l'effet involontaire d'un développement de site Web défectueux.
- **Politesse** : les serveurs Web ont à la fois des règles implicites et explicites régissant le débit auquel un robot d'exploration peut les visiter. Ces politiques de politesse doivent être respectées.
- **Distribué** : le robot devrait pouvoir s'exécuter de manière distribuée sur plusieurs machines.
- **Évolutif** : l'architecture du moteur de balayage devrait permettre d'augmenter la vitesse d'exploration en ajoutant des machines et de la bande passante supplémentaires.

- **Performances et efficacité** : le système d'analyse doit utiliser efficacement différentes ressources système, notamment le processeur, le stockage et la bande passante du réseau.
- **Fraîcheur** : dans de nombreuses applications, le robot devrait fonctionner en mode continu: il devrait obtenir de nouvelles copies des pages précédemment récupérées. Un robot de moteur de recherche, par exemple, peut ainsi s'assurer que l'index du moteur de recherche contient une représentation assez récente de chaque page Web indexée. Pour une telle exploration continue, un robot devrait pouvoir explorer une page avec une fréquence proche du taux de changement de cette page.
- **Qualité** : étant donné qu'une fraction importante de toutes les pages Web est peu utile pour répondre aux besoins des utilisateurs en matière de requêtes, le robot devrait privilégier la recherche de pages «utiles».
- **Extensible** : les robots d'exploration doivent être conçus de manière à être extensibles à bien des égards - pour faire face aux nouveaux formats de données, aux nouveaux protocoles de récupération, etc. Cela exige que l'architecture du robot soit modulaire.

- En plus de ces caractéristiques on ajoute une liste de fonctionnalités supplémentaires qu'il serait bien d'avoir. Au lieu d'être simplement évolutif, on souhaite que le robot soit *évolutif de manière dynamique*, afin de pouvoir ajouter et supprimer des machines lors d'analyses Web continues. On souhaite également que le robot d'exploration puisse *exporter des données* dans divers systèmes de stockage ou pipelines de données tels qu'Amazon S3, HDFS ou Kafka.

II. Approches d'analyse

Avant d'entrer dans le vif du sujet de la comparaison, prenons un peu de recul et examinons deux cas d'utilisation différents des robots d'exploration de sites Web *:* les analyses ciblées et les analyses larges.

Dans une analyse ciblée, nous sommes intéressés par un ensemble spécifique de pages (généralement un domaine spécifique). Par exemple, nous pouvons explorer toutes les pages de produits sur Amazon. Com.

Dans une analyse large, l'ensemble des pages qui nous intéressent est très volumineux ou illimité et s'étend sur plusieurs domaines. C'est généralement ce que font les moteurs de recherche.

Une analyse ciblée comportant un grand nombre de domaines (ou plusieurs analyses ciblées effectuées simultanément) s'approchera essentiellement des propriétés d'une analyse large.

Lorsque nous analysons un domaine (tel que amazon.com), nous sommes essentiellement limité par notre politique de politesse. Nous ne voulons pas surcharger le serveur avec des milliers de demandes par seconde, sinon nous serons bloqués. Ainsi, nous devons imposer une limite de demandes par seconde. Cette limite est généralement basée sur le temps de réponse du serveur. En raison de cette limite, la plupart des ressources de la CPU ou du réseau de notre serveur seront inactives. Avoir un robot d'exploration distribué utilisant des milliers de machines ne fera pas avancer une analyse concentrée plus rapidement que de l'exécuter sur votre ordinateur.

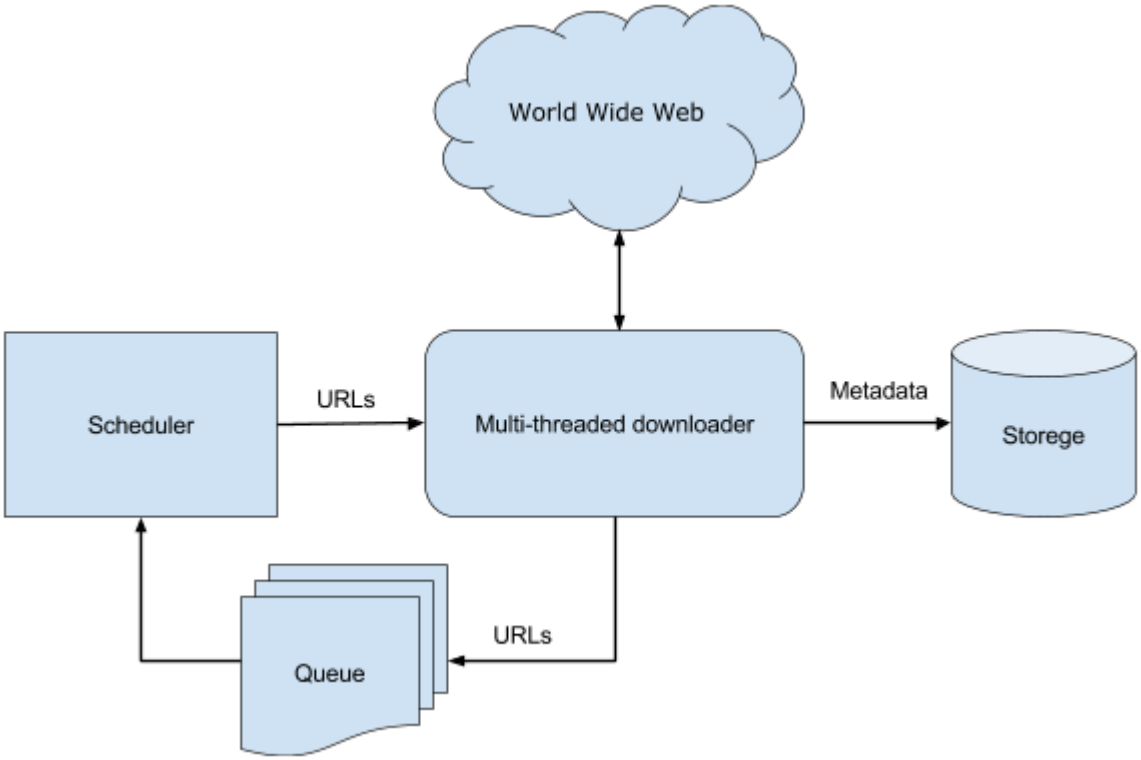


Figure 1:Architecture d'un robot Web.

En cas d'analyse large, le goulot d'étranglement est constitué par les performances et l'évolutivité du robot. Comme vous devez demander des pages à différents domaines, vous pouvez potentiellement effectuer des millions de demandes par seconde sans surcharger un serveur spécifique. Vous êtes limité par le nombre de machines dont vous disposez, leur processeur, la bande passante du réseau et la capacité de votre robot d'exploration à utiliser ces ressources.

Si tout ce que vous voulez, c'est récupérez les données de plusieurs domaines, recherchez un robot d'exploration à l'échelle Web peut s'avérer excessif. Dans ce cas Scrapy est un excellent choix pour les analyses ciblées.

III. Comparaison des robots Web

Au premier lieu passons à la présentation de nos deux outils open source : **Scrapy** et **apache-nutch.**

Open source	Langages	OS type	Operating
Scrapy	Python	Linux / mac os x /Windows /others	BSD license
Apache Nutch	Java	Linux/ mac os x /windows	Apach license

Figure 2: comparaison des 2 principeaux robots open source

III .1.scrapy

Performance et efficacité

-Scrapy est un Framework d'analyse Web permettant au développeur d'écrire du code à créer spider, qui définit la manière dont un site donné (ou un groupe de sites) sera supprimé. Scrapy est donc implémenté à l'aide d'un code non bloquant (ou asynchrone) pour la simultanéité, ce qui rend les performances de l'araignée extrêmement performantes.

-Ce qui ressort de Scrapy, c'est sa facilité d'utilisation et son excellente documentation . Si vous connaissez Python, vous serez opérationnel en quelques minutes.

Evolutivité

-Scrapy fonctionne également très bien sur Python 2 et Python 3, la compatibilité ne sera donc pas un problème. Il prend en charge de manière intégrée l'extraction de données à partir de sources HTML à l'aide d'une expression XPath et d'une expression CSS.

Distribution

-Il ne possède pas de fonctionnalité intégrée pour s'exécuter dans un environnement distribué, de sorte que ses analyses de cas d'utilisation principales sont ciblées. Cela ne veut pas dire que Scrapy ne peut pas être utilisé pour l'exploration à grande échelle, mais d'autres outils pourraient mieux convenir à cette fin, en particulier à très grande échelle. Selon la documentation, la meilleure pratique pour distribuer les analyses est de partitionner manuellement les URL en fonction du domaine.

Extensibilité

-L'architecture de Scrapy est bien conçue, vous pouvez facilement développer un middleware ou un pipeline personnalisé pour ajouter des fonctionnalités personnalisées. Notre Scrapy projet peut être à la fois robuste et flexible. Après avoir développé plusieurs projets Scrapy, vous profiterez de l'architecture et de son design car il est facile de migrer d'un projet Scrapy existant vers un autre.

Qualité

-Scrapy a quelques formats d'exportation intégrés très pratiques, tels que JSON, lignes JSON, XML et CSV. Scrapy a été conçu pour extraire des informations spécifiques de sites Web, sans pour autant obtenir un vidage complet du code HTML et l'indexer. Ce dernier nécessite un travail manuel pour éviter d'écrire l'intégralité du contenu HTML de toutes les pages dans un fichier de sortie gigantesque. Vous devrez découper les fichiers manuellement.

-Sans la possibilité d'exécuter dans un environnement distribué, de faire évoluer de manière dynamique ou d'exécuter des analyses en continu, Scrapy manque de certaines fonctionnalités clés. Cependant, si vous avez besoin d'un outil facile à utiliser pour extraire des informations spécifiques d'un ou plusieurs domaines, alors le scrapy est presque parfait.

Les avantages:

- Facile à installer et à utiliser si vous connaissez Python
- Excellente documentation pour les développeurs
- Formats d'exportation JSON, lignes JSON, XML et CSV intégrés

Les inconvénients:

- Pas de support pour l'exécution dans un environnement distribué
- Pas de support pour les explorations continues
- Exportation de grandes quantités de données est difficile

III.2. Apache-nutch

Apache Nutch est un robot Web bien établi basé sur Apache Hadoop. En tant que tel, il fonctionne par lots, d'où les différents aspects de l'analyse Web sont définis séparément (générer une liste d'URL à récupérer, analyser les pages Web et mettre à jour ses structures de données)ce qui réduit le risque d'erreur.

Distribution

Avec nutch au lieu de créer son propre système distribué, il utilise l'écosystème Hadoop et utilise MapReduce pour son traitement. Si vous avez déjà un cluster Hadoop existant, vous pouvez simplement le diriger vers Nutch. Si vous ne possédez pas de cluster Hadoop, vous devrez en configurer un.

Performance

Nutch hérite des avantages (tels que la tolérance aux pannes et l'évolutivité), mais également des inconvénients (accès au disque lent entre les tâches en raison de la nature du lot) de l'architecture HadoopMapReduce.

Il est intéressant de noter que Nutch n'a pas commencé comme un robot entièrement Web. Il a commencé comme un moteur de recherche open source qui gère à la fois l'exploration et l'indexation du contenu Web. Même si, depuis, Nutch est devenu davantage un robot d'exploration de Web, il est toujours livré avec une intégration profonde pour des systèmes d'indexation tels que Solr (par défaut) et ElasticSearch (via des plugins). La nouvelle branche 2.x de Nutch tente de séparer le serveur de stockage du composant d'analyse à l'aide d' Apache Gora., mais est encore à un stade relativement précoce. Dans notre propres expériences, on l as trouvé plutôt immature. Cela signifie que si vous envisagez d'utiliser Nutch, vous serez probablement limité à le combiner avec Solr comme dans notre case ou bien ElasticSearch Web Crawler, ou à écrire votre propre plug-in pour prendre en charge un backend ou un format d'exportation différent.

Efficacité

Malgré de nombreuses expériences antérieures, on a trouvé que Nutch est assez difficile à installer et à configurer, principalement en raison d'un manque de bonne documentation ou d'exemples concrets.

Extensibilité

Nutch (1.x) semble être une plate-forme stable utilisée par diverses organisations, notamment CommonCrawl.Il possède un système de plug-in flexible, vous permettant de l'étendre avec des fonctionnalités personnalisées. En effet, cela semble être nécessaire pour la plupart des cas d'utilisation. Lorsque vous utilisez Nutch, vous pouvez vous attendre à passer un peu de temps à écrire vos propres plugins ou à parcourir le code source pour l'adapter à votre cas d'utilisation. Si vous avez le temps et l'expertise pour le faire, alors Nutch semble être une excellente plate-forme sur laquelle s'appuyer.

Qualité

Nutch ne prend actuellement pas en charge les analyses continues, mais vous pouvez écrire quelques scripts pour émuler une telle fonctionnalité.

Les avantages:

- Dynamiquement évolutif (et tolérant aux pannes) via Hadoop
- Système de plugin flexible :il fournit un cadre de plug-in, qu'il peut prendre en charge toutes sortes d'analyses de contenu Web, une variété de collecte de données, de requêtes, de filtres et d'autres fonctions. C'est grâce à ce cadre, le développement de plug-in Nutch est très facile.
- Branche stable 1.x

Les inconvénients:

- Mauvaise documentation et gestion des versions déroutante. Aucun exemple.
- Hérite des inconvénients de Hadoop (lecture de disque, configuration difficile)
- Pas de support intégré pour les analyses continues
- Export limité à Solr / ElasticSearch (sur une branche 1.x)

IV. Conclusion

Dans ce chapitre, nous avons présenté les divers fonctionnalités et caractéristiques de chacun des outils Scrapy et apache-nutch. Nous avons réussi à les comparer sur plusieurs aspects tels que l' évolutivité, la robustesse, ainsi la performance et l'efficacité d'utilisation afin de nous aider à choisir le meilleur outil qui convient le mieux à nos futurs projets, et cela selon nos besoins.

Sans surprise, il n'existe pas de robot «parfait» sur le Web. Il s'agit de choisir le bon pour votre cas d'utilisation et selon vos besoins. Voulez-vous faire une analyse ciblée ou large? Avez-vous un cluster Hadoop existant ? Où voulez-vous stocker vos données ?

Parmi les deux outils qu'on a examinés dans les chapitres précédents, Scrapy est probablement notre préféré, mais il est loin d'être parfait.

C'est probablement pourquoi certaines personnes et organisations ont choisi de construire leur propre robot d'exploration. Cela peut être une alternative fiable si rien de ce qui précède ne correspond à votre cas d'utilisation exact.

V. **Conclusion général** :

Alors que l'Internet a connu une croissance astronomique et que les différents dommaines de nos jours sont devenues de plus en plus dépendants des données, il est désormais impérieux d'avoir accès aux dernières données disponibles sur chaque sujet.

Les données sont devenues la base de tous les processus décisionnels, qu'il s'agisse d'une entreprise ou d'une organisation à but non lucratif. Par conséquent, le Web scraping a trouvé ses applications dans tous les efforts notables de l'époque contemporaine.

Il devient également de plus en plus clair que ceux qui feront une utilisation créative et avancée des outils de raclage Web devanceront les autres et gagneront un avantage concurrentiel.

Alors, exploitons le raclage Web et augmentons nos perspectives dans le domaine de notre choix!