

Durée: 2^h

Au programme

- Récupération des données depuis le web.
- Construction d'un programme aspirateur ou *web crawler* ou encore *web scraping* en anglais.
- Stocker les données téléchargées.

Avant propos

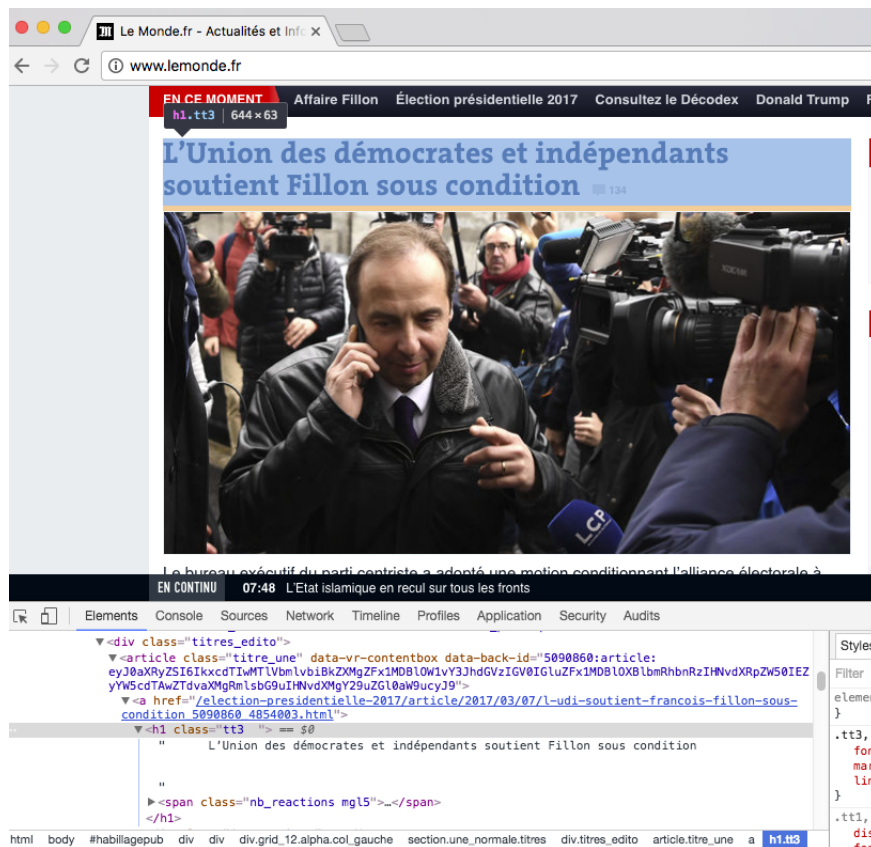


FIGURE 1 – Outils des développeurs de chrome. L'outil permet d'inspecter/accéder aux éléments HTML de la page web courante.

Est-ce-que le *web scraping* est légal ?

Le *web scraping* est l'action de récupérer des données depuis un site web. Quand vous construisez un aspirateur et que vous vous connectez à un site web, vous êtes leurs invités. L'usage des données récupérées doit être strictement personnel. Les données récupérées ne doivent pas être republiées sauf dans des cas exceptionnels. Voir cet article pour plus de détails <http://bit.ly/2k0WWVR>.

Afin de récupérer les données depuis le web, il nous faut d'abord savoir comment naviguer parmi les éléments d'une page web. Pour cela nous allons utiliser la syntaxe XPath qui nous permet de récupérer les données stocker au travers une page web.

Durée: 2^h

Définition 1 (XPath) *XPath pour XML path language fournit une syntaxe pour identifier et localiser des fragments de documents XML ou dans des documents HTML. Il décrit des chemins à prendre pour aller de la racine ou d'un noeud "local" vers le noeud recherché en suivant les filiations.*

Pour copier le XPath d'un élément précis dans une page web, par exemple le titre à la une de la page `lemonde.fr`, il suffit d'ouvrir l'outil des développeurs de votre navigateur web, ensuite faire un clic droit sur l'élément qui vous intéresse, et dans le menu choisir **inspecter**. Cela met en exergue l'élément inspecté par rapport à l'arborescence HTML (voir exemple figure 1).

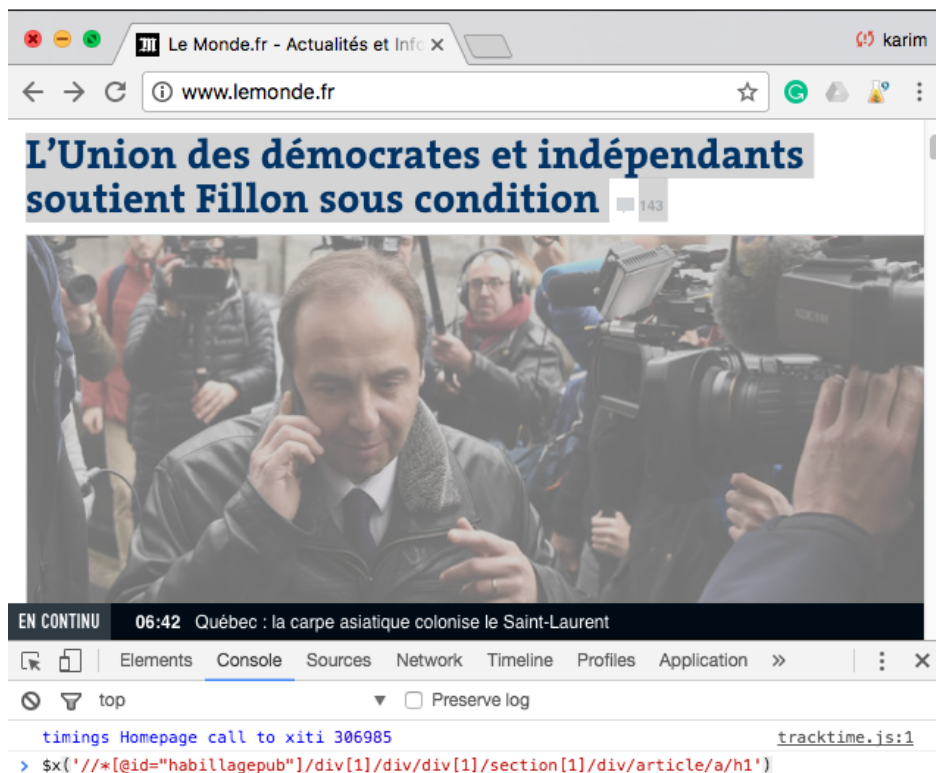


FIGURE 2 – Vérification de l'intégrité du XPath dans la console JavaScript fournit par google chrome.

Afin de vérifier si le XPath que vous avez copié est intègre, vous êtes invités à le tester dans une console JavaScript comme c'est décrit dans le code ci-dessous et la figure 2

```
1 $x( ' //*[@id="habillagepub "]/div [1]/ div /div [1]/ section [1]/ div / article /a/h1 ' )
```

Définition 2 (Parser) *Un parser est un analyseur syntaxique d'un langage. Communément codé dans une fonction, il prend en entrée une structure définie avec une syntaxe par exemple XML et retourne un dictionnaire où les clés de ce dernier sont les attributs des balises et les valeurs sont les données se trouvant entre les balises.*

Durée: 2^h

Nous utiliserons la programmation orienté objets pour programmer notre aspirateur de données. Nous faisons dans ce qui suit un rappel des concepts fondamentaux de la programmation orientés objets.

Définition 3 (Classe) *Une classe est une représentation conceptuelle d'un ensemble d'objets observés. Chaque classe possède un ensemble d'attributs et un ensemble de fonctions.*

Définition 4 (Attributs) *Les attributs représentent les caractéristiques d'une classe. C'est des variables qui sont initialisées avec des valeurs qui varient selon l'objet.*

Définition 5 (Objet et instance) *Un objet est la représentation concrète d'une classe. Une instance est un objet dont les attributs sont initialiser avec un **constructeur**. Le constructeur est une fonction qui a pour objectif d'initialiser les attribut de l'objet de la classe.*

Définition 6 (Héritage) *L'héritage est une relation filiale entre deux classes. Une classe générale ou classe mère et une classe particulière ou classe fille qui hérite toutes les différentes caractéristiques et les fonctions de sa classe mère.*

Définition 7 (Polymorphisme) *Le principe du polymorphisme et la redéfinition de la suite d'instructions dans le corps de la fonction. C'est-à-dire que la fonction garde la même signature, mais avec une suite différente d'instructions.*

Pour construire notre aspirateur de données, nous utiliserons la librairie **Scrapy**¹. Cette librairie écrite en python, fournit un ensemble de fonctionnalités orientée objets qui nous permettront de spécifier le type de données que nous souhaitons télécharger, de créer un aspirateur en fournissant un lien en entrée, d'extraire les données et analyser la syntaxe de la structure retournée avec un parser, et enfin de se connecter à une base de données pour y stocker les données aspirées. L'installation de scrapy passe par le gestionnaire de librairie python **pip**.

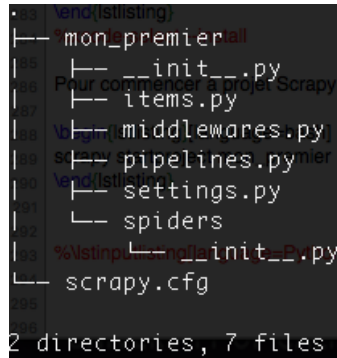
¹ `pip install Scrapy`

1. <https://doc.scrapy.org/en/latest/>

Durée: 2^h

Pour commencer à projet Scrapy :

```
1 scrapy startproject mon_premier
```



```
├── mon_premier
│   ├── __init__.py
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
└── scrapy.cfg
2 directories, 7 files
```

FIGURE 3 – Les fichiers et le dossier créé par la commande scrapy startproject

Cela créera une arborescence avec un dossier et plusieurs fichiers (voir figure 3). Le dossier `spiders` contiendra les aspirateurs que nous allons créer. Dans les fichiers créer nous utiliserons les suivants : le fichier `settings.py` contient tous les paramètres nécessaires au fonctionnement de notre aspirateur, comme les identifiant et mot de passe pour se connecter à une base de données. Le fichier `pipelines.py` contiendra les instructions nécessaires pour rediriger le flux de données téléchargées vers une base de données. Le fichier `item.py` contiendra les types de données que nous souhaitons télécharger, par exemple le titre d'un article, les images dans un article, le lien de l'article, etc. Ces types de données définies dans le fichier `item.py` serviront après comme nom d'attributs pour créer la table dans la base qui stockera les données téléchargées par l'aspirateur. Chaque fichier python est un module qui contiendra une classe et un ensemble de fonctions. Nous nous référons, dans la suite, à un aspirateur comme un *spider*.

Exercice

Dans cet exercice, nous allons récupérer le titre des articles recherchés sur le catalogue de la bnf, et l'heure où nous avons récupéré ces depuis le site <http://catalogue.bnf.fr/>. Nous stockerons ensuite les données dans un fichier json.

- 1) Créer un projet scrapy et l'appeler `bnf`.
- 2) Modifier dans le module `items` la classe `BnfItems` qui contiendra les champs : titre, temps_du_scrap.
- 3) Écrire un nouveau module dans le dossier `spiders` et l'appeler `bnf_catalogue`.
- 4) Dans le module `bnf_catalogue`, créer une classe et l'appeler `BnfSpider`. Cette classe hérite de `Spider` et extrait les données depuis un lien issu du site <http://catalogue.bnf.fr/>.
- 5) Créer au sein de la classe `BnfSpider`, la fonction `parse`.
- 6) Stocker les données récupérées dans un fichier json.