Manipulation de données

Web scraping

Xavier Gendre

Le Web sans API

Les sites web représentent une importante source de données :

- site interne d'entreprise,
- Wikipedia,
- TMDB, …

Tous les sites ne proposent pas une API pour récolter de l'information (outil non pertinent, manque de moyens, modèle économique, …).

Cependant, l'information est assez structurée pour que nous puissions la récupérer de façon **semi-automatisée**.

Principe du webj scraping

L'objet du web scraping est l'**extraction de données** depuis une page web.

La démarche est similaire à une visite de la page avec un navigateur web et des copier-coller pour récupérer l'information pertinente.

Le principe est d'**automatiser** cette démarche :

- récupérer le contenu brut de la page web,
- fouiller ce contenu pour dénicher l'information,
- organiser et stocker les données.

Contenu d'une page web

Le travail d'un navigateur web est de transformer des fichiers textes en pages lisibles et bien présentées. En simplifiant le fonctionnement, cela se décompose en :

- des **informations structurées** dans un fichier HTML,
- la **mise en page** dans des feuilles de style CSS,
- des **scripts** (*JavaScript*) qui rendent la page dynamique.

Les données seront à extraire du fichier HTML. Le style du CSS aident localiser l'information. La gestion des scripts est plus difficile et ne sera pas prise en compte dans un premier temps.

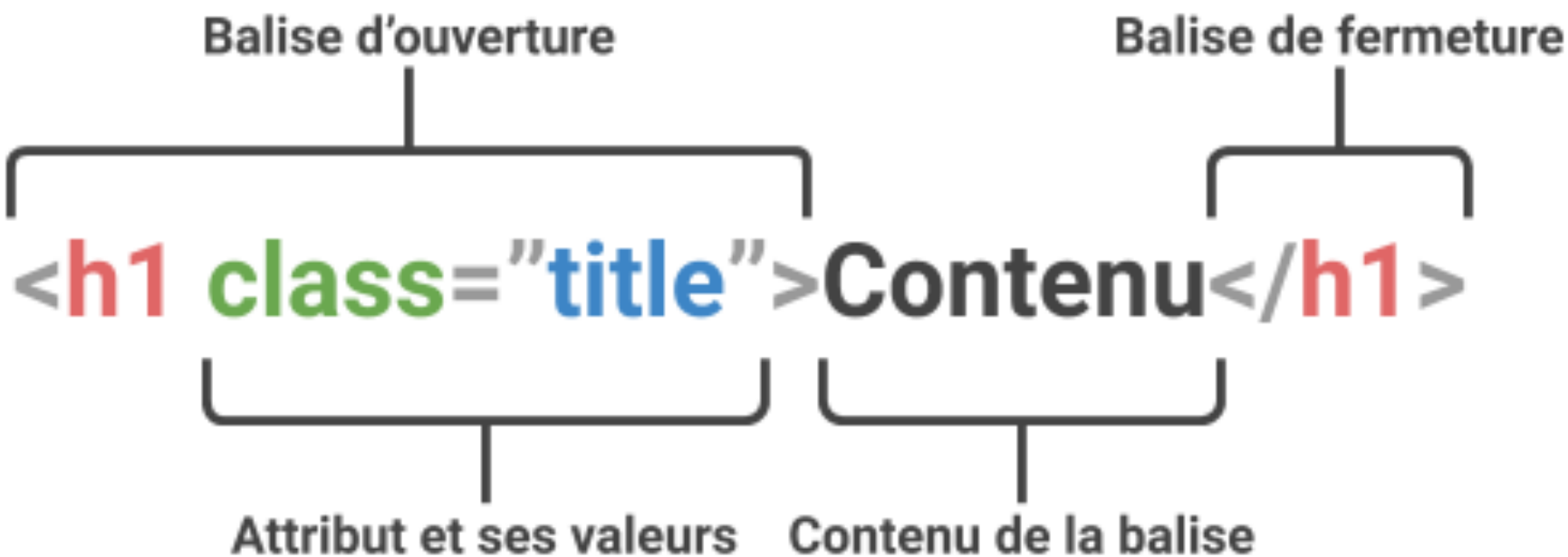
Un peu de HTML

Le HTML (*Hypertext Markup Language*) est le langage utilisé pour représenter les pages web. Il s'agit d'un langage à **balises** qui permettent d'identifier les éléments qui composent une page (lien, image, table, formulaire, …).

```
<html>
    <head>
        <title>Titre de la page</title>
    </head>
    <body>
        <h1>Titre de niveau 1</h1>
        <p>Un paragraphe avec <a href="https://...">un lien</a>.
        <ul>
            <li>Élément de liste</li>
            <li>Élément de liste</li>
        </ul>
    </body>
</html>
```

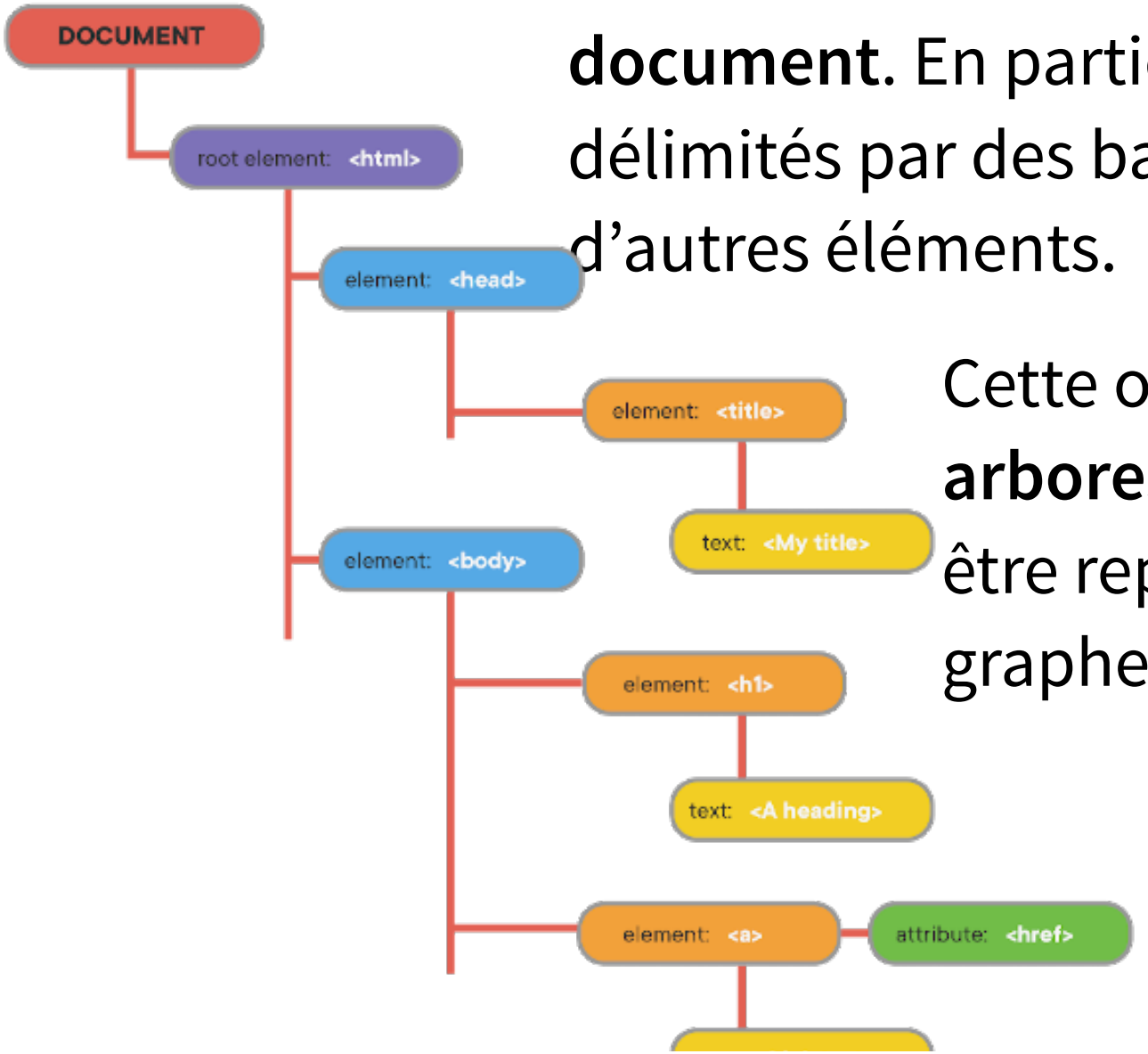
Une balise HTML s'ouvre et se ferme (à quelques exceptions près). Son **contenu** se trouve entre la balise d'ouverture et celle de fermeture. Dans la balise d'ouverture, il est possible de définir des **attributs**.

Deux attributs ont un rôle particulier : `id` (unique dans la page) et `class`.



- `h1`, `h2`, `h3`, … Titres de différentes niveaux
- `p` Paragraphe
- `a` Lien (attribut `href` pour la cible)
- `i` Morceau de texte spécial (souvent en italique)
- `img` Image (**autofermante**, attribut `src` pour la source)
- `div` et `span` Conteneurs génériques
- `ul` Liste à puces
- `li` Élément d'une liste à puces
- `table` Table (avec noms de lignes et de colonnes)
- …

Arbre DOM



Le format d'un fichier HTML est un **format document**. En particulier, les éléments délimités par des balises peuvent contenir d'autres éléments.

Cette organisation est dite **arborescente** et peut donc être représenté comme un graphe appelé **arbre DOM**.

L’interface de programmation DOM (*Document Object Model*) permet d’examiner et de modifier cet arbre et donc le contenu d’une page web.

Les scripts manipulent l’arbre DOM d’une **page dynamique**.

Le chargement d’un site se fait en deux temps :

- construction de l’arbre DOM de base,
- exécution des scripts modifiant l’arbre DOM.

> **Page web sans script**
>
> Cette considération est importante pour le web scraping lorsque seul l’arbre DOM de base est fouillé.

Objectif

Cate Blanchett interprète le rôle de *Galadriel* dans les films de Peter Jackson. Nous allons chercher avec quels acteurs Cate Blanchett a le plus joué dans les années 2000.

Nous pouvons commencer par visiter la section dédiée à la filmographie de Cate Blanchett sur sa page Wikipedia.

https://fr.wikipedia.org/wiki/Cate_Blanchett#Filmographie

Nous constatons que :

- les films sont classés par décennie,
- chaque titre de film (sauf un) possède un lien vers une page,
- ces pages contiennent des listes d'acteurs.

Web scraping avec Python

La première étape consiste à récupérer le HTML de la page ciblée. Il s'agit d'une requête HTTP `GET`.

```python
import requests

url_wikipedia = "https://fr.wikipedia.org"
url_blanchett = url_wikipedia + "/wiki/Cate_Blanchett"

r = requests.get(url_blanchett)

if r.status_code == 200:
    print("Page web récupérée")
else:
    print(f"Erreur {r.status_code}")
```

```
Page web récupérée
```

Le contenu de la réponse est le code HTML brut de la page web.

```
1 print(r.text[:800] + "...")
```

```
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-
feature-language-in-main-page-header-disabled vector-feature-page-tools-
pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feature-main-
menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-
limited-width-content-enabled vector-feature-custom-font-size-clientpref-1
vector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-
enabled skin-theme-clientpref-day vector-sticky-header-enabled vector-toc-
available" lang="fr" dir="ltr">
<head>
<meta charset="UTF-8">
<title>Cate Blanchett — Wikipédia</title>
<script>(function(){var className="client-js vector-feature-language-in-
header-enabled vector-feature-language-in-main-page-header-disabled vector-
feature-page-...
```

Pour se déplacer dans l'arbre DOM et extraire de l'information, il faut analyser sa structure. Cette étape s'appelle le *parsing*.

- **BeautifulSoup** Largement utilisé et bien documenté, ce module permet d'explorer facilement l'arbre DOM de base.
- **Scrapy** Ce module réputé pour sa rapidité permet aussi d'explorer facilement l'arbre DOM de base.
- **Selenium** Module avancé qui émule un navigateur afin de simuler des actions et d'exécuter le JavaScript.
- **Urllib3** Module efficace mais offrant moins de fonctionnalités et plus difficile à prendre en main.
- **Lxml** Module de parsing XML (donc HTML), le contenu doit être parfaitement formaté et tout est à construire.

Beautiful Soup

Le module `BeautifulSoup` sera utilisé dans la suite.

```
1 from bs4 import BeautifulSoup
2
3 soup = BeautifulSoup(r.text, "html.parser") # HTML parser de Python
4 print(str(soup)[:700] + "...")
```

```
<!DOCTYPE html>

<html class="client-nojs vector-feature-language-in-header-enabled vector-
feature-language-in-main-page-header-disabled vector-feature-page-tools-
pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feature-main-
menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-
limited-width-content-enabled vector-feature-custom-font-size-clientpref-1
vector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-
enabled skin-theme-clientpref-day vector-sticky-header-enabled vector-toc-
available" dir="ltr" lang="fr">
<head>
<meta charset="utf-8"/>
<title>Cate Blanchett — Wikipédia</title>
<script>(function(){var className="client-js vector-featur...
```

Plusieurs méthodes sont disponibles pour rechercher dans l'arbre DOM. En particulier, la méthode `find_all` permet de récupérer tous les éléments d'un type donné.

```
1 soup.find_all("h2")
```

```
[<h2 class="vector-pinnable-header-label">Sommaire</h2>,
 <h2 id="Biographie">Biographie</h2>,
 <h2 id="Théâtre"><span id="Th.C3.A9.C3.A2tre"></span>Théâtre</h2>,
 <h2 id="Filmographie">Filmographie</h2>,
 <h2 id="Distinctions">Distinctions</h2>,
 <h2 id="Voix_francophones">Voix francophones</h2>,
 <h2 id="Notes_et_références"><span id="Notes_et_r.C3.A9f.C3.A9rences"></
span>Notes et références</h2>,
 <h2 id="Liens_externes">Liens externes</h2>]
```

La méthode `find_all` admet plusieurs arguments dont `id` qui permet de récupérer l'unique élément avec l'attribut `id` donné.

```
1 soup.find_all(id="Biographie")
```

```
[<h2 id="Biographie">Biographie</h2>]
```

L'argument `attrs` permet de filtrer sur les attributs et leur contenu.

```
1 soup.find_all("a", attrs={"title": "Festival de Cannes 1982"})
```

```
[<a href="/wiki/Festival_de_Cannes_1982" title="Festival de Cannes
1982">1982</a>]
```

Il est aussi possible de filtrer simplement sur l'attribut `class`.

```
1 soup.find_all("div", "mw-heading")[:2]
```

```
[<div class="mw-heading mw-heading2"><h2 id="Biographie">Biographie</h2><span
class="mw-editsection"><span class="mw-editsection-bracket">[</span><a
class="mw-editsection-visualeditor" href="/w/index.php?
title=Cate_Blanchett&amp;veaction=edit&amp;section=1" title="Modifier la
section : Biographie"><span>modifier</span></a><span class="mw-editsection-
divider"> | </span><a href="/w/index.php?
title=Cate_Blanchett&amp;action=edit&amp;section=1" title="Modifier le code
source de la section : Biographie"><span>modifier le code</span></a><span
class="mw-editsection-bracket">]</span></span></div>,
 <div class="mw-heading mw-heading3"><h3 id="Jeunesse_et_formation">Jeunesse
et formation</h3><span class="mw-editsection"><span class="mw-editsection-
bracket">[</span><a class="mw-editsection-visualeditor" href="/w/index.php?
title=Cate_Blanchett&amp;veaction=edit&amp;section=2" title="Modifier la
section : Jeunesse et formation"><span>modifier</span></a><span class="mw-
editsection-divider"> | </span><a href="/w/index.php?
title=Cate_Blanchett&amp;action=edit&amp;section=2" title="Modifier le code
```

Sélecteur CSS

Explorer l'ensemble des possibilités du module `BeautifulSoup` serait fastidieux et `find_all` n'est pas la méthode la plus pratique. Dans la suite, une autre approche largement utilisée en pratique sera présentée : l'utilisation d'un **sélecteurs CSS**.

Un sélecteur CSS est une expression utilisée dans les feuilles de style CSS pour identifier à quels éléments HTML s'applique une règle (texte en gras, couleur du fond, …).

Un sélecteur CSS se construit à partir des **noms des balises**, de **combinateurs** et de **pseudo-classes**.

Films de Cate Blanchett

Voici comment récupérer les noms des films de Cate Blanchett de la liste à puces des années 2000 à partir d'un sélecteur CSS et de la méthode `select`.

```
1 css_selector = "#mw-content-text div ul:nth-of-type(3) li i a"
2 films = soup.select(css_selector)
3
4 for film in films[:4]:
5     print(f"-> {film.text}") # Contenu textuel
6     print(film.attrs) # Attributs de l'élément
```

```
-> Les Larmes d'un homme
{'href': '/wiki/Les_Larmes_d%27un_homme', 'title': "Les Larmes d'un homme"}
-> Intuitions
{'href': '/wiki/Intuitions', 'title': 'Intuitions'}
-> Bandits : Gentlemen braqueurs
{'href': '/wiki/Bandits_(film,_2001)', 'title': 'Bandits (film, 2001)'}
-> Le Seigneur des anneaux : La Communauté de l'anneau
{'href': '/wiki/Le_Seigneur_des_anneaux_:_La_Communaut%C3%A9_de_l%27anneau',
'title': "Le Seigneur des anneaux : La Communauté de l'anneau"}
```

```
1 css_selector = "#mw-content-text div ul:nth-of-type(3) li i a"
```

Quelques explications : 🤯

- l'espace sert de **combinateur d'enfant** et permet d'utiliser la filiation pour désigner un élément contenu dans un autre,
- `#mw-content-text` est un **sélecteur d'identifiant** (attribut `id`, élément unique),
- `ul:nth-of-type(3)` est une **pseudo-classe** indiquant que seul le 3ème élément `ul` doit être considéré.

👉 Utiliser l'**inspecteur** dans les outils de développement du navigateur (`F12`) pour construire un sélecteur CSS.

Sélecteur CSS (Suite)

- le chevron > est un combinateur d'enfant **direct**,
- le point . est un sélecteur de classe,

```
".toto" capture <p class="toto">...</p>, <li class="toto">...</li>, ...
"p.toto" capture <p class="toto">...</p> uniquement
```

- les crochets [ ] filtrent sur les attributs,

```
"a[href]" capture <a href="truc">...</p> et <a href="bidule">...</p>
"a[href='truc']" ne capture que <a href="truc">...</p>
```

- les pseudo-classes permettent beaucoup de choses,

```
"p:first-child" capture le premier enfant d'un élément p
```

- ...

Distribution d'un film

Le même procédé peut être utilisé pour extraire les acteurs d'un film à partir des liens collectés sur la page de C. Blanchett.

```
# Noter le découpage utile de l'URL
url_film = url_wikipedia + "/wiki/Heaven_(film,_2002)"
r = requests.get(url_film)
soup = BeautifulSoup(r.text, "html.parser")
# Le combinateur + donne l'élément suivant (adjacent sibling)
# La pseudo-classe :has filtre sur les éléments enfants
css_selector = "div:has(h2#Distribution) + ul li > a:nth-of-type(1)"
[acteur.text for acteur in soup.select(css_selector)]
```

```
['Cate Blanchett',
 'Giovanni Ribisi',
 'Remo Girone',
 'Stefania Rocca',
 'Mattia Sbragia',
 'Stefano Santospago',
 'Alberto Di Stasio',
 'Giovanni Vettorazzo',
 'Gianfranco Barra',
 'Vincenzo Ricotta',
 'Mauro Marino']
```

Application

Contenu de la page

```
url_wikipedia = "https://fr.wikipedia.org"
url_blanchett = url_wikipedia + "/wiki/Cate_Blanchett"

r_blanchett = requests.get(url_blanchett)

if r_blanchett.status_code == 200:
    print("Page 'Cate Blanchett' récupérée")
else:
    print(f"Erreur {r_blanchett.status_code}")

soup_blanchett = BeautifulSoup(r_blanchett.text, "html.parser")
```

```
Page 'Cate Blanchett' récupérée
```

Liste des films

```
selector_films = "#mw-content-text div ul:nth-of-type(3) li i a"
films = soup_blanchett.select(selector_films)

print(f"{len(films)} films")
```

```
26 films
```

Chaque film a un titre (*title*) et un lien vers sa page (*href*) …

```
films[0].attrs
```

```
{'href': '/wiki/Les_Larmes_d%27un_homme', 'title': "Les Larmes d'un homme"}
```

… sauf un !

```
films[15].attrs
```

```
{'href': '/w/index.php?title=Stories_of_Lost_Souls&action=edit&redlink=1',
 'class': ['new'],
 'title': 'Stories of Lost Souls (page inexistante)'}
```

Liste des acteurs

```
selector_acteurs = (
    "div:has(h2#Distribution) + ul li > a:nth-of-type(1)"
)

acteurs = []
for film in films:
    if (
        film.attrs.get("class") == ["new"] # Film sans page
        or film.attrs["title"] == "Galadriel" # Mauvais lien
    ):
        continue # Saute le film
    url_film = url_wikipedia + film.attrs["href"]
    r_film = requests.get(url_film)
    soup_film = BeautifulSoup(r_film.text, "html.parser")
    acteurs.extend(
        [acteur.text for acteur in soup_film.select(selector_acteur
    )
```

Réponse avec Pandas

```
1 import pandas as pd
2
3 print(
4     pd.DataFrame({"Acteur": acteurs})
5     .Acteur.value_counts()
6     .head(3)
7 )
```

```
Acteur
Cate Blanchett     11
Judi Dench          2
Giovanni Ribisi     2
Name: count, dtype: int64
```

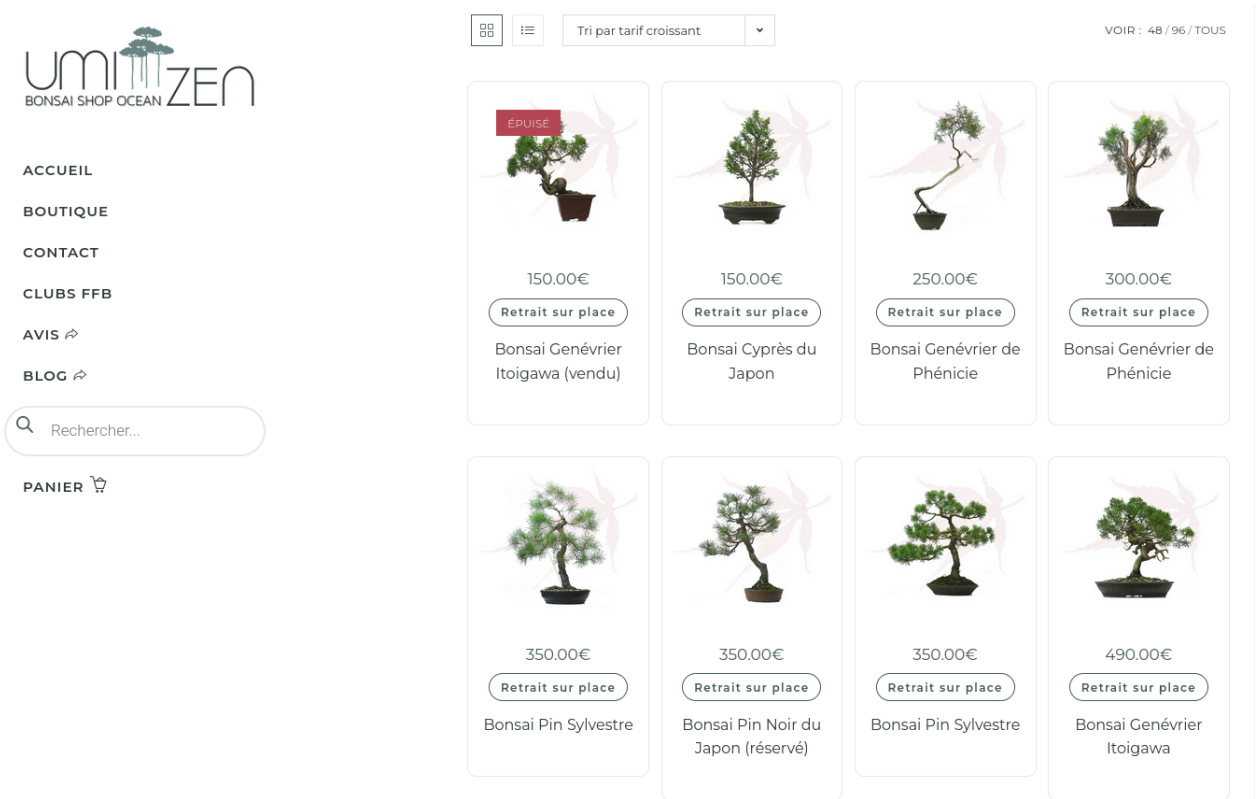
Où sont les acteurs du *Seigneur des anneaux* ? 😢

Le sélecteur CSS de la liste des acteurs ne capture pas assez de choses. Ce sera amélioré dans les exercices 😉

Dernière remarque

Les méthodes utilisées telles que `find_all`, `select`, … fonctionnent aussi sur les éléments capturés dans la page.

Lorsque une même organisation se répète, le processus consiste souvent à capturer ces blocs et à extraire l'information ensuite dans chacun d'entre eux.



À vous de jouer !