Mémo : Analyse de données XML et HTML dans Openrefine

Auteur : Mathieu Saby

Licence CC-BY

Historique

|  |  |  |  |
| --- | --- | --- | --- |
| V1.3 |  | 6/11/2020 | Correction de l’exemple Jython |
| V1.2 |  | 8/05/2019 | Traduction complétée |
| V1.1 |  | 15/03/2019 | Avancement de traduction, exemples revus |
| V1 |  | 24/10/2018 | Création, traduction à finir |

La version d’Openrefine décrite est la 3.1.

Table des matières

[1. Analyse de données XML et HTML 2](#_Toc3954260)

[1.1. Les sélecteurs jsoup (traduction en cours) 2](#_Toc3954261)

[2. Utilisation de Jython 5](#_Toc3954262)

# Analyse de données XML et HTML

La fonction **select()** permet de sélectionner des éléments issus de **parseHtml()** ou de **parseXml()** en utilisant la syntaxe du parseur **jsoup** (<https://jsoup.org>). Jsoup utilise des sélecteurs insensibles à la casse, identiques aux sélecteurs CSS3 et Jquery.

Voir la documentation complète : <https://jsoup.org/apidocs/org/jsoup/select/Selector.html>

Interface de démonstration (pour tester une expression jsoup) : <https://try.jsoup.org/>

## Les sélecteurs jsoup (traduction en cours)

Un sélecteur est une chaîne de sélecteurs simples séparés par des combinateurs. Les sélecteurs sont **insensibles à la casse**. Le **sélecteur universel** (\*) est implicite lorsqu’aucun sélecteur d’élément n’est fourni (ex : \*.header et .header sont équivalents)

**Sélecteurs d’éléments**

Peuvent être combinés dans n’importe quel ordre. Ex : div.header[title] = div[title].header

|  |  |  |
| --- | --- | --- |
| **Motif** | **Trouve** | **Exemple** |
| \* | N’importe quel élément | \* |
| tag | Les éléments de type tag | div |
| \*|E | Les éléments de type E dans le namespace *ns* | \*|name trouve les éléments <fb:name> |
| ns|E | Les éléments de type E dans le namespace *ns* | fb|name trouve les éléments <fb:name> |
| #id | Les éléments avec un attribut "id" | div#x trouve les éléments <div id="x"> |
| .class | Les éléments avec une classe "class" | div.x trouve les éléments <div class="x"> |
| [attr] | Les éléments avec un attribut "attr" | a[href] trouve les éléments <a href=…> |
| [^attrPrefix] | Les éléments avec un attribu commençant par "attrPrefix" | div[^data-] trouve les éléments <div data-…=…>  |
| [attr=val] | Les éléments avec un attribut "attr" de valeur égale à "val" | img[width=500], a[rel=nofollow] |
| [attr="val"] | Les éléments avec un attribut "attr" de valeur égale à "val" | span[nom="Martin"], a[rel="nofollow"] |
| [attr^=valPrefix] | Les éléments avec un attribut "attr" de valeur commençant par "valPrefix" | a[href^=http:] |
| [attr$=valSuffix] | Les éléments avec un attribut "attr" de valeur se terminant par "valSuffix" | img[src$=.png] |
| [attr\*=valContaining] | Les éléments avec un attribut "attr" de valeur contenant "valContaining" | a[href\*=/search/] |
| [attr~=regex] | Les éléments avec un attribut "attr" de valeur correspondant à une expression régulière. | img[src~=(?i)\\.(png|jpe?g)] |

**Sélecteurs de combinaison**

|  |  |  |
| --- | --- | --- |
| **Motif** | **Trouve** | **Exemple** |
| E F | Les éléments F descendant d’un élément E | div a, .logo h1 |  |
| E > F | Les éléments F fils d’un élément E | ol > li |  |
| E + F | Les éléments F immédiatement précédés par un élément frère E | li + li, div.head + div |  |
| E ~ F | Les éléments F précédés par un élément frère E | h1 ~ p |  |
| E, F, G | Les éléments E ou F ou G | a[href], div, h3 |  |

**Pseudo-sélecteurs**

Peuvent être combinés entre eux dans n’importe quel ordre, et avec les autres sélecteurs

|  |  |  |
| --- | --- | --- |
| **Motif** | **Trouve** | **Exemple** |
| :lt(n) | Les éléments dont l’index parmi les éléments frères est inférieur à *n* (0 pour le 1er) | td:lt(3) trouve les 3 premières colonnes de chaque ligne |  |
| :gt(n) | Les éléments dont l’index parmi les éléments frères est supérieur à *n* (0 pour le 1er) | td:gt(1) trouve toutes les colonnes de chaque ligne sauf la 1re |  |
| :eq(n) | Les éléments dont l’index parmi les éléments frères est égal à *n* (0 pour le 1er) | td:eq(0) trouve la 1re colonne de chaque ligne |  |
| :has(selector) | Les éléments contenant au moins un des éléments correspondant à *selector* | div:has(p) trouve les div contenant des p |  |
| :not(selector) | Les éléments ne correspondant pas à *selector* | div:not(.logo) trouve les div n’ayant pas la classe « logo »div:not(:has(p)) trouve les div ne contenant pas des p. |  |
| :contains(text) | Les éléments contenant le texte *text* (recherche insensible à la casse), directement ou dans un de leurs descendants | p:contains(hell) trouve les p contenant le texte Hello |  |
| :matches(regex) | Les éléments correspondant à l’expression régulière *regex*, directement ou dans un de leurs descendants le texte *text.* Les classes de caractères doivent être échappés par \\. Les modificateurs sont indiqués dans un groupe (?) précédant le motif | td:matches(\\d+) trouve les colonnes contenant des chiffresdiv:matches((?i)hell)  trouve les div contenant "hell" mais aussi "HELLO". |  |
| :containsOwn(text) | Les éléments contenant directement le texte *text* (recherche insensible à la casse) | p:containsOwn(hello) trouve les éléments p contenant directement "hello" |  |
| :matchesOwn(regex) | Les éléments correspondant directement à l’expression régulière *regex* | td:matchesOwn(\\d+) trouve les colonnes contenant directement des chiffres.div:matchesOwn((?i)hell) trouve les div contenant directement "hell" mais aussi "HELLO". |  |
| :containsData(data) | Les éléments contenant les données data (inclut les nœuds de type script, style, comment mais exclut les nœuds de type texte), directement ou dans un de leur descendant.. | script:contains(hello) trouve des éléments script contenant "hello" |  |
| :matchText | Traite les nœuds texte comme des éléments auxquels appliquer d’autres sélecteurs | p:matchText:first-child appliqué à <p>oui <br/> non</p> trouve la partie de <p> précédant <br/> : "oui" :matchText appliqué à <p>oui</p><div><p>non</p></div> trouve [oui, non] |  |
|  |  |  |  |

**Pseudo-sélecteurs structuraux**

|  |  |  |
| --- | --- | --- |
| **Motif** | **Trouve** | **Exemple** |
| :root | L’élément racine du document. En HTML l’élément html | :root |  |
| :nth-child(an+b) | Les éléments de position (*a*n+*b*) dans la liste de tous les enfants (quelque soit leur type) de leur parent. a et b sont des entiers positifs, négatifs ou nuls, n est un compteur démarrant à 0. La position du 1er enfant de la liste est 1. *a*n peut être omis pour trouver les éléments de position (*b*) dans la liste de tous les enfants de leur parent | tr:nth-child(2n+1) trouve chaque ligne impaire d’un tableau.:nth-child(10n-1) trouve le 9e, le 19e, le 29e élément… li:nth-child(5) trouve le 5e élément li |  |
| :nth-last-child(an+b) | Même principe que :nth-child en comptant à rebours à partir du dernier élément de la liste de tous les enfants | tr:nth-last-child(-n+2) trouve les 2 dernières lignes d’un tableau |  |
| :nth-of-type(an+b) | Même principe que :nth-child en en prenant en compte que la liste des enfants du même type que l’élément cible | img:nth-of-type(2n+1) |  |
| :nth-last-of-type(an+b) | Même principe que :nth-last-child en ne prenant en compte que la liste des enfants du même type que l’élément cible | img:nth-last-of-type(2n+1) |  |
| :first-child | Les éléments qui sont le 1er enfant d’un autre élément | div > p:first-child |  |
| :last-child | Les éléments qui sont le dernier enfant d’un autre élément | ol > li:last-child |  |
| :first-of-type | Les éléments qui sont le 1er enfant du même type dans la liste des enfants de leur parent | dl dt:first-of-type |  |
| :last-of-type | Les éléments qui sont le dernier enfant du même type dans la liste des enfants de leur parent | tr > td:last-of-type |  |
| :only-child | Les éléments qui sont le seul enfant de leur parent |  |  |
| :only-of-type | Les éléments qui sont le seul enfant de leur type de leur parent |  |  |
| :empty | Les éléments sans enfant |  |  |

# Utilisation de Jython

Il est également possible d’écrire du code en Jython en utilisant la librarie xml.etree.

Ex : récupérer le titre et l’auteur de livres écrits par des auteurs anglais

**from** xml.etree **import** ElementTreeelement = ElementTree.fromstring(value.encode(**"utf-8"**))
listeResultats = element.findall(**".//author[@country='GB']/.."**)
titreauteur = []

**for** resultat **in** listeResultats:

 titreauteur.append ( resultat.find(**'title'**).text + " / " + resultat.find(**'author'**).text )

**return** "|". **return** (titreauteur)

Résultat :

|  |
| --- |
|  |
|  | **value** | **from xml.etree import ElementT ...** |
|  | <firstlist><book><title>Le Seigneur des anneaux</title><author country="GB">Tolkien</author></book><book><title>La peste</title><author country="FR">Camus</author></book></firstlist> | Le Seigneur des anneaux / Tolkien |
|  | <secondlist><book><title>Hamlet</title><author country="GB">Shakespeare</author></book><book><title>Portnoy</title><author country="US">Roth</author></book></secondlist> | Hamlet / Shakespeare |