Mémo : Analyse de données JSON dans Openrefine

Auteur : Mathieu Saby

Licence CC-BY

Historique

|  |  |  |
| --- | --- | --- |
| V1 | 10/11/2018 | Création |

La version d’Openrefine décrite est la 3.1.

Table des matières

[1. Analyse de données JSON en GREL 1](#_Toc529943141)

[2. Analyse de données JSON en Python 1](#_Toc529943142)

# Analyse de données JSON en GREL

Une valeur JSON peut être **atomique** (texte (entre guillemets double), nombre, booléen, null) ou **complexe** (objets, tableaux).

Les **objets** (entre {}) sont composés d’une ou plusieurs paires non ordonnées de couples « "clé" : "valeur" » séparées par des virgules. Les clés sont toujours encadrées de guillemets doubles.

Ex : {"nom" : "Dupont"} ou {"nom" : "Dupont", "prenom" : "Jean", "age" :43}

Les **tableaux** (entre []) sont composés d’une ou plusieurs valeurs ordonnées séparées par des virgules.

Ex : ["Dupont"] ou ["Dupont","Martin",34]

Un tableau ou un objet peuvent être imbriqués dans un autre tableau ou objet.

Ex : {"clients" : [ {"nom" : "Dupont"}, {"nom" : "Martin"} ] }

La fonction GREL parseJson() transforme une chaîne en valeur JSON qui peut ensuite être manipulée en parcourant l’index des tableaux ou en récupérant la valeur d’une clé.

Le Nième élément d’un tableau est récupérable avec la syntaxe [N].

La valeur correspondant à une clé est récupérable avec la syntaxe ["clé"] ou .clé.

* Ex : '["a","b",12]'.parseJson()[0] 🡪 « a »
* Ex : '{"a" : 1,"b" : 2}'.parseJson().a 🡪 1

Les éléments d’un tableau peuvent être listés et manipulés avec forEach (qui parcourt un tableau) et filtrés avec filter (qui renvoie les éléments du tableau correspondant à certains critères)

Ex : soit l’objet JSON suivant (objet -> tableau -> 2 objets)

{"liste":[

{"id":"id001","name":"nom1"},

{"id":"id002","name":"nom2"}

]}

Pour récupérer un tableau contenant la clé "name" pour tous les éléments de la liste :

* forEach(value.parseJson()["liste"], element\_temporaire, element\_temporaire["name"]) 🡪 [ "nom1", "nom2" ]

Pour récupérer le nom de l’élément id001:

* filter(value.parseJson()["liste"], element\_temporaire, element\_temporaire["id"]=="id001")[0]["name"]

# Analyse de données JSON en Python

Si la clé était variable ou inconnue au lieu d’être toujours "liste", le plus simple est de recourir à Jython au lieu de GREL.

Exemple :

**import** json
donnees = json.loads (value)
**for** cle, valeur **in** donnees.items():
 **return** (valeur[**'name'**])

Résultat :

|  |  |
| --- | --- |
| **value** | **import json donnees = json.loa ...** |
| {"inconnu1":{"id":"id001","name":"nom1"}} | nom1 |
| {"inconnu2":{"id":"id002","name":"nom2"}} | nom2 |