Stockage de données: les solutions simples - TP1

Hiba ALQASIR

Automne 2021

Objectif de la séance

L'objectif de cette séance est la prise en main des principaux formats de fichiers de stockage de données, CSV, JSON, XML et YAML. A la fin de la séance, vous déposerez sur le portail une archive au format zip contenant tous vos fichiers, *i.e.* un fichier texte avec vos réponses (pas de code à l'intérieur de ce fichier), et tous les fichiers javascript, csv, json ...

1 Mise en forme des données

A partir du tableau suivant créer quatre fichiers au format CSV, XML, JSON et YAML.

Rank	City	Member State	Official population	Date of census
1	Berlin	Germany	3669495	31 December 2019
2	Madrid	Spain	3348536	1 February 2020
3	Rome	Italy	2856133	31 December 2018
4	Paris	France	2175601	1 January 2019

Source wikipedia.org

1. CSV.
   - Lancer Microsoft Excel ou OpenOffice Calc et saisir les données comme ci-contre.
   - Enregistrer au format csv (cities.csv).
   - Ouvrir le fichier créé avec un éditeur de texte. Reporter et commenter le résultat.
2. XML.
   - Ouvrir un éditeur de texte.
   - Transformer le fichier cities.csv afin de le mettre au format XML, attention il faut ajouter en première ligne le code suivant:
     <?xml version="1.0" encoding="UTF-8"?>

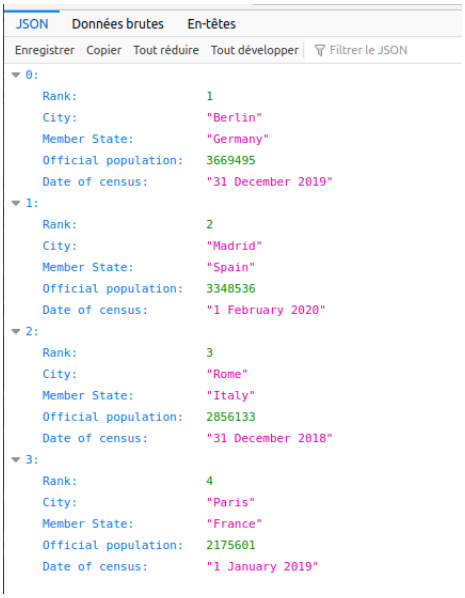
- Enregistrer le fichier avec l'extension XML (cities.xml).

3. JSON.
   Refaire la même chose et transformer le fichier (cities.xml) au format JSON. Enregistrer le fichier avec l'extension JSON (cities.json).

4. YAML.
   Refaire la même chose et transformer le fichier (cities.json) au format YAML. Enregistrer le fichier avec l'extension YAML ou YML (cities.yaml ou cities.yml). Cet outil json2YAML permet de passer de JSON à YAML, et vérifier la syntaxe des deux formats.

Vérifier la syntaxe de vos fichiers XML et JSON en les ouvrant par Firefox, on doit voir les sorties suivantes.

XML



JSON

2 Open data

Les données que nous allons utiliser, sont accessibles sur l'open data de la Plateforme ouverte des données publiques françaises, à l'adresse suivante (Cliquer ici). Il s'agit de données temps réels sur les véliverts (vélo en libre service) de Saint-Étienne Métropole. Ces données permettent de connaître:

- Liste des stations
- État des stations

1. Télécharger les fichiers. Informations des stations vélivert (vls) de Saint-Étienne Métropole et État des stations. Ces fichiers sont de quel format?

2. Combien de lignes y a-t-il? Et combien de colonnes? Nommer toutes les attributs dans chaque fichier.

3 Extraire les données des fichiers CSV

1. Le code JavaScript suivant permet de récupérer quelques détails sur un fichier.

```
const fs = require('fs') // requiring file system module
fs.stat('./cities.csv', (err, stats) => {
    if (err) {
        console.error(err)
        return
    }
    console.log(stats.isFile() )
    console.log(stats.isDirectory() )
    console.log(stats.size)
    console.log(stats.blocks)
    console.log(stats.blksize)
});
```

   (a) Copier le code dans VS Code et enregistrer le fichier sous l'appellation 'file_formats1.js'.

   (b) Remplacer ./cities.csv par le chemin complète ver le fichier, par exemple /home/user/Bureau/cities.csv. Ou placer 'file_formats1.js' et 'cities.csv' dans le même répertoire, sans changer le code.

   (c) Lancer l'exécution de ce programme dans VS Code.

   (d) Reporter et commenter le résultat de l'exécution de ce programme.

   (e) Reprendre le programme précédent en utilisant le fichier 'station_information.json' comme source de données, reporter et commenter le résultat.

2. Le code JavaScript suivant permet d'extraire les données d'un fichier CSV:

```
const fs = require('fs')
fs.readFile('./cities.csv', 'utf8' , (err, data) => {
   if (err) {
       console.error(err)
       return
   }
   var rows = data.split('\n');
   for(var i = 0; i < rows.length; i++){
       var row = rows[i]
       console.log('line ' + i + ': ' + row);
       // a completer avec la fonction split()
   }
});
```

   (a) Copier le code dans un fichier 'file_formats2.js'.

   (b) Remplacer ./cities.csv par le chemin complète ver le fichier, par exemple /home/user/Bureau/cities.csv. Ou placer 'file_formats2.js' et 'cities.csv' dans le même répertoire, sans changer le code.

   (c) Indiquer le fonctionnement de l'instruction split() [1].

   (d) Que représente rows.lenght ?

   (e) Lancer l'exécution de ce programme dans VS Code.

   (f) Reporter et commenter le résultat de l'exécution de ce programme.

3. Compléter le programme ci-dessus en utilisant la fonction split() afin d'obtenir le résultat suivant:

```
line 0: Rank, City, Member State, Official population, Date of census
[0]: Rank
[1]: City
[2]: Member State
[3]: Official population
[4]: Date of census
line 1: 1, Berlin, Germany, 3669495, 31 December 2019
[0]: 1
[1]: Berlin
[2]: Germany
[3]: 3669495
[4]: 31 December 2019
line 2: 2, Madrid, Spain, 3348536, 1 February 2020
[0]: 2
[1]: Madrid
[2]: Spain
[3]: 3348536
[4]: 1 February 2020
line 3: 3, Rome, Italy, 2856133, 31 December 2018
[0]: 3
[1]: Rome
[2]: Italy
[3]: 2856133
[4]: 31 December 2018
line 4: 4, Paris, France, 2175601, 1 January 2019
[0]: 4
[1]: Paris
[2]: France
[3]: 2175601
[4]: 1 January 2019
```

[1] Vous pourrez vous reporter au site internet javascript.info qui est un guide générale sur le JavaScript. Ou au site internet nodejs.dev qui est un tutoriel Node.js.

4 Extraire les données des fichiers JSON

1. Le code JavaScript suivant permet d'extraire les données d'un fichier JSON:

```
const fs = require('fs')
fs.readFile('./cities.json', 'utf8' , (err, data) => {
  if (err) {
    console.error(err)
    return
  }
  var rows = JSON.parse(data);
  for(var i = 0; i < rows.length; i++){
        var row = rows[i]
        console.log('[0] : ' +  row["Rank"]);
        // a completer avec les autres attributes
   }
});
```

   (a) Copier le code dans un fichier 'file_formats3.js'.

   (b) Remplacer ./cities.json par le chemin complète ver le fichier, par exemple /home/user/Bureau/cities.json. Ou placer 'file_formats3.js' et 'cities.json' dans le même répertoire, sans changer le code.

   (c) Indiquer le fonctionnement de l'instruction JSON.parse() .

   (d) Lancer l'exécution de ce programme dans VS Code.

   (e) Reporter et commenter le résultat de l'exécution de ce programme.

2. Compléter le programme ci-dessus afin d'obtenir le même résultat obtenu en question 3.3.

5 Traitement des données

1. Le code JavaScript suivant permet d'écrire des informations dans un fichier CSV:

```
var csv = "";
tmp = 'Rank, City \n';
csv+=tmp
fs.writeFile('./cities2.csv', csv.toString(), 'utf8', (err,
    data) => {
  if (err) {
    console.error(err)
    return
  }
});
```

   (a) Copier le code dans un fichier 'file_formats4.js'.

   (b) Lancer l'exécution de ce programme dans VS Code.

   (c) Un nouveau fichier 'cities2.csv' est créé, qu'est-ce qu'il contient?

(d) Modifier ce code pour écrire un programme qui lit un fichier JSON et le transforme en CSV automatiquement. Utiliser le code écrit en question 4.

(e) Tester le programme sur les données véliverts. Nommer les nouveaux fichiers 'station_information.csv' et 'station_status.csv'.

2. Le code JavaScript suivant permet d'écrire des informations dans un fichier JSON:

```
var json = [];
tmp = {}
tmp['Rank'] = 4;
tmp['City'] = 'Paris';
json.push(tmp);
fs.writeFile('./cities2.json', JSON.stringify(json), 'utf8', (
    err, data) => {
  if (err) {
    console.error(err)
    return
  }
});
```

(a) Copier le code dans un fichier 'file_formats5.js'.

(b) Lancer l'exécution de ce programme dans VS Code.

(c) Un nouveau fichier 'cities2.json' est créé, qu'est-ce qu'il contient?

(d) Modifier ce code pour écrire un programme qui lit un fichier CSV et le transforme en JSON automatiquement. Utiliser le code écrit en question 3.

(e) Tester le programme sur les données véliverts (fichiers reproduits par la question précédente). Nommer les nouveaux fichiers 'station_information2.json' et 'station_status2.json'.

(f) Comparer les fichiers 'station_information.json' et 'station_information2.json'. Ont-ils la même syntaxe? Et pour 'station_status.json' et 'station_status2.json'?