SLAM – Structuration des données

1] Notion d'Open Data

- ***Le cadre de la notion***

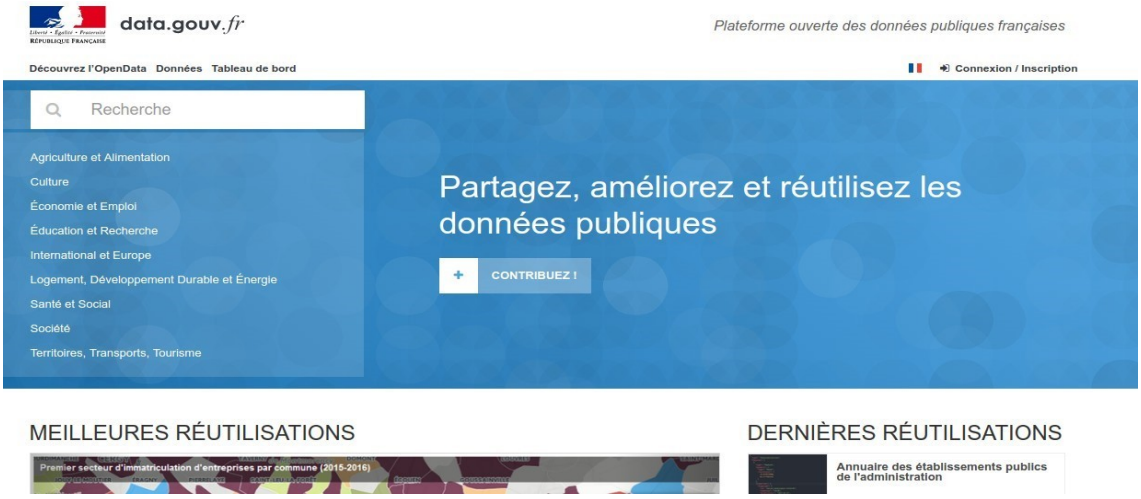
Suivant Wikipédia, une **donnée ouverte** (*Open Data*) est :

"... une donnée numérique dont l'accès et l'usage sont laissés libres aux usagers. Elle peut être d'origine publique ou privée, produite notamment par une collectivité, un service public (éventuellement délégué) ou une entreprise. Elle est diffusée de manière structurée selon une méthode et une licence ouverte garantissant son libre accès et sa réutilisation par tous, sans restriction technique, juridique ou financière."

En clair pour un développeur : c'est la possibilité d'utiliser librement des données fournies par des tiers dans ses applications.

En France nous avons comme sites Internet importants :

- ✗ Le site de la plateforme ouverte des données publiques françaises : https://www.data.gouv.fr/fr/datasets/
- ✗ Les sites spécifiques aux régions (ex : https://data.iledefrance.fr), aux entreprises (ex : https://data.ratp.fr) ou aux villes (ex : https://data.rennesmetropole.fr/explore/?sort=modified)
- ✗ L'avis de la CNIL : https://www.cnil.fr/fr/publication-en-ligne-et-reutilisation-des-donnees-publiques-open-data



- ***Formats de données proposées en Open Data***

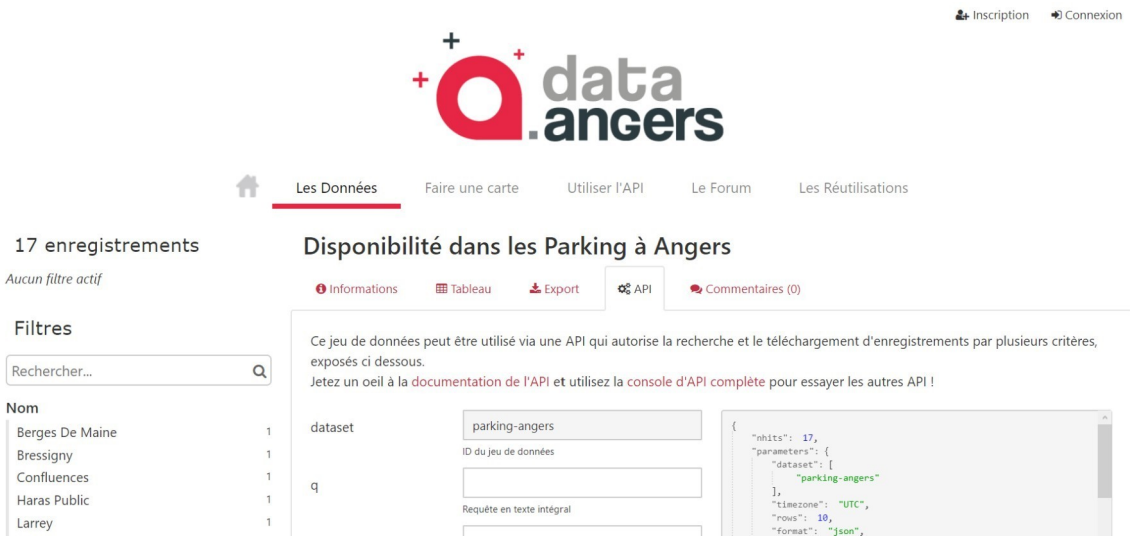
Les formats proposés courants :

1. Le format **CSV** : fichier texte représentant des données issues de tableaux sous la forme de valeurs séparées par des virgules
2. Le format **JSON** : fichier texte représentant des données sous forme structurée et dérivée de la notation objets du langage JavaScript
3. Le format **XLS/ODS** : fichier du tableur Microsoft Excel mais lisible dans d'autres logiciels OpenSource comme LibreOffice
4. Le format **SHAPEFILE** : fichier spécifique contenant des informations géographiques pour les SIG (Systèmes d'information géographiques), logiciels commerciaux ou libres

2] Le principe d'une API

API signifie *Application Programming Interface* et désigne une application dont le but (dans le cadre d'une application Web) est de fournir une interface (en gros le moyen) d'interagir avec un service distant de données.

Les APIs les plus courantes actuellement respectent le standard **REST** (***RE**presentational **S**tate **T**ransfert*) basées sur HTTP (méthodes GET, PUT, POST et DELETE) avec des réponses au format JSon.



- ***Un exemple d'API***

La ville d'Angers propose au travers d'un site une API concernant la disponibilité en temps réel des places de ses parkings : https://data.angers.fr/explore/dataset/parking-angers/api/.

Voici par exemple une demande de type GET sur le parking de Bressigny
https://data.angers.fr/api/records/1.0/search/?dataset=parking-angers&facet=nom&refine.nom=Bressigny
donne en retour :


```
{"nhits": 1, "parameters": {"dataset": ["parking-angers"], "refine": {"nom": "Bressigny"}, "timezone": "UTC", "rows": 10, "format": "json", "facet": ["nom"]}, "records":
[{"datasetid": "parking-angers", "recordid": "a5c896b57c9d99e73de3f713c9b60fa7d817cd29", "fields": {"nom": "Bressigny", "disponible": 79}, "record_timestamp": "2019-08-
19T14:20:00+00:00"}], "facet_groups": [{"name": "nom", "facets": [{"name": "Bressigny", "path": "Bressigny", "count": 1, "state": "refined"}]}]}
```


3] Le format JSon

- ***Descriptif***

Le format JSon décrit des structures de type objet (comme le montre son acronyme ***J**ava**S**cript **O**bject **N**otation*) de façon organisée et facilement lisible lorsqu'on l'ouvre comme un fichier texte. Il s'utilise naturellement avec d'autres techniques comme **AJAX** (***A**synchronous**J**ava**S**cript **A**nd **X**ML*) ou dans le cadre d'**APIs** de type **REST** (vu plus haut). En relation avec la syntaxe du JavaScript, il est par contre indépendant de tout langage de programmation et s'interprète dans n'importe quel autre langage à l'aide d'un "parseur".

Qu'est-ce qu'un parseur ?

Un parseur est une méthode d'objet qui analyse et récupère des informations d'un fichier d'un certain format en vue de les exploiter dans un autre.

- ***Règles syntaxiques***

1. Il ne doit exister qu'un seul élément père par document contenant tous les autres : un élément racine.
2. Tout fichier JSon bien formé doit être soit un objet (commençant par "{" et se terminant par "}"), soit un tableau (commençant par "[" et terminant par "]"), éventuellement vides
3. Les séparateurs utilisés entre deux paires/valeurs sont des virgules

4. Un objet JSon peut contenir une chaîne de caractères, un nombre, un booléen, un tableau : […] ou d'autres objets Json
5. Il ne peut y avoir d'éléments croisés comme en HTML

Exemple 1 :

```
[{
   "nom": "Kiwis",
   "quantité": 3
 },
 {
   "nom": "Mangues",
   "quantité": 6
 },
 {
   "nom": "Bananes",
   "quantité": 9
 }
]
```

Exemple 2 un peu plus conforme :

```
{
   "fruits": [{
         "nom": "Kiwis",
         "quantité": 3
     },
     {
         "nom": "Mangues",
         "quantité": 6
     },
     {
         "nom": "Bananes",
         "quantité": 9
     }
   ]
}
```

Exemple 3 un peu plus compliqué :

```
{
   "fruits": [{
      "kiwis": 3,
      "mangues": 4,
      "pommes": 0
      },{
      "panier": true
   }],
   "legumes": {
      "patates": "amandine",
      "poireaux": false
   },
   "viandes": ["poisson", "poulet", "bœuf"]
}
```

- ***Liens Web pour des validateurs JSon***

La syntaxe particulière d'un fichier JSon implique de vérifier si il respecte les règles. On peut utiliser pour cela des validateurs en lignes : http://jsonlint.com/ ou http:// jsonviewer.stack.hu/**.**

4] JSon et les langages de programmation

- ***JSon et JavaScript***

En reprenant l'exemple JSon n°3 et en le mettant dans une variable, on peut tester le code suivant dans un fichier nommé `parsing_json.html` :

```
<!DOCTYPE HTML>
```

```
<html lang="fr">
 <head>

  <title>Parsing JSon/JavaScript</title>
  <meta charset="utf-8" />

 </head>
 <body>

  <script language="JavaScript">
   var data = '{"fruits":[{"kiwis":3,"mangues":4,"pommes":0},{"panier":true}],"legumes":
{"patates":"amandine","poireaux":false},"viandes":["poisson","poulet","boeuf"]}';

   json = JSON.parse(data);

   alert(json.fruits[0].kiwis);
   alert(json.legumes.patates);
   alert(json.viandes);

   lesFruits = json.fruits[0];

   alert(Object.entries(lesFruits));
   alert(Object.values(lesFruits));
  </script>

 </body>
</html>
```

A vous de voir la documentation JavaScript comme pour `Object.values` :
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Object/values

- ***JSon et PHP***

Prenons cette fois le fichier JSon venant de la région Île de France, recensant les gares et que vous trouverez ici : https://data.iledefrance.fr/explore/dataset/gares-du-reseau-ferre-dile-de-france0/export/

Le code du fichier nommé `parsing_json.php` (le fichier JSon a été renommé pour être plus court) :

```
<!DOCTYPE HTML>
<html lang="fr">
 <head>

  <title>Parsing JSon/PHP</title>
  <meta charset="utf-8" />

 </head>
 <body>

 <?php
  $json = file_get_contents("idf_sncf.json");

  // Sortie du premier élément sous forme d'objet, paramètre false
  $parsed_json = json_decode($json,false);

  print("<p>\n");
  print_r($parsed_json[0]);
  print("</p>\n");

  // Sortie du premier élément sous forme de tableau associatif, paramètre true
  // Les traitements suivants seront sous cette forme
  $parsed_json = json_decode($json,true);

  print("<p>\n");
  print_r($parsed_json[0]);
  print("</p>\n");


  // On enlève un objet
  foreach ($parsed_json as &$value) {
```

```
   unset($value['datasetid']);
   }

  // Affichage seuleument des trois premiers
  print("<p>\n");
  print_r($parsed_json[0]);
  print_r($parsed_json[1]);
  print_r($parsed_json[2]);
  print("</p>\n");

  // On enlève un élément
  foreach ($parsed_json as $i => &$value) {
   foreach ($parsed_json[$i] as $key => $valfields) {
    if ($key=="fields") {
     if (isset($valfields['desserte'])) {
      $code = $valfields['desserte'];

     if ($code=="C") {
      array_splice($parsed_json,$i,1);
     }
    }
   }
  }
  }

  // Affichage seulement des trois premiers
  print("<p>\n");
  print_r($parsed_json[0]);
  print_r($parsed_json[1]);
  print_r($parsed_json[2]);
  print("</p>\n");

  // Pour voir la structure (très long)
  // var_dump(json_decode($json));
  ?>

 </body>
</html>
```