Taurre

Zeste de savoir

JSON, YAML, TOML, XML,... Ou comment se compliquer la vie

22 mars 2019

Table des matières

% JSON, YAML, TOML, XML,... OU COMMENT SE COMPLIQUER LA VIE % Taurre % 10 juin 2018

1. Introduction

Bonjour à tous,

Ces dernières années, j'ai pu constater une net augmentation de l'emploi de formats de ficher textuels complexes. Alors, l'existence et l'emploi de tels formats n'est certainement pas chose nouvelle, il suffit de penser au XML et a son dérivé le plus usité, le XHTML, pour s'en convaincre.

En revanche, l'emploi de ce type de fichier semble devenue à présent quasi systématique, alors pourtant que ces formats sont loin d'être les plus simples aussi bien à lire que *parser*.

Afin d'illustrer mon propos, je vous propose de prendre trois exemples d'utilisation qui peuvent parfaitement être remplacé par l'emploi du format INI (j'ai choisi ce format parce qu'il est connu et un minimum structuré, cela étant ce n'est bien entendu pas le seul possible) :

1. L'utilisation de YAML par *Ansible* ;
2. L'utilisation du XML par *Openbox* ;
3. L'utilisation du JSON par Zeste de Savoir.

2. Ansible

Ansible est un logiciel qui permet d'effectuer des actions à distances sur plusieurs machines à la fois. Cela est réalisé via des connexions en SSH et l'exécution d'instructions en Python. Chaque action à effectuer est décrite à l'aide d'une entrée rédigée en YAML.

L'exemple ci-dessous spécifie que les fichiers `etc/apt/sources.list` et `etc/apt/preferences` présents sur la machine source (donc celle qui exécute *Ansible*) doivent être déplacées vers la ou les machines de destinations aux emplacements `/etc/apt/sources.list` et `/etc/apt/preferences` avec les permissions indiquées.

```
- name: Copy configuration files
  copy:
    src: '{{ item }}'
    dest: '/{{ item }}'
    owner: root
    group: root
    mode: 0644
  with_items:
    - etc/apt/sources.list
    - etc/apt/preferences
```

Or, dans ce cas ci, le format YAML fait-il sens ? Est-il nécessaire de se farcir une syntaxe exigeante en termes d'éléments (chaînes, dictionnaires, listes, etc.), d'indentation (si vous ne mettez pas le bon nombre d'espaces, c'est mort) et de *parsing* ? *Eh* bien, pas vraiment, non.

Techniquement, la même chose est possible avec le format INI, sans s'ennuyer avec toutes cette lourdeur syntaxique.

```
[Copy configuration files]
MODULE=copy
SRC={{ item }}
DEST=/{{ item }}
OWNER=root
GROUP=root
MODE=0644
WITH_ITEMS=etc/apt/sources.list
WITH_ITEMS=etc/apt/preferences
```

Avait-on besoin du YAML ? Visiblement, non.

3. Openbox

Openbox est un gestionnaire de fenêtre minimaliste disponible sous GNU/Linux et *BSD, offrant de nombreuses possibilités. Il est souvent utilisé pour concevoir des environnements graphiques peu gourmand en ressources. Parmi ses fichiers de configuration, on retrouve un certain `rc.xml` qui permet de régler différents aspects, des raccourcis claviers au positionnement par défaut de certaines fenêtres.

Voici un court exemple de ce fichier en rapport avec configuration des raccourcis claviers.

```
  <keyboard>
    <chainQuitKey>C-g</chainQuitKey>
    <keybind key="W-d">
```

```
      <action name="ToggleShowDesktop"/>
    </keybind>
    <keybind key="A-F4">
      <action name="Close"/>
    </keybind>
    <keybind key="W-m">
      <action name="ShowMenu">
        <menu>root-menu</menu>
      </action>
    </keybind>
    <keybind key="A-Tab">
      <action name="NextWindow">
        <finalactions>
          <action name="Focus"/>
          <action name="Raise"/>
          <action name="Unshade"/>
        </finalactions>
      </action>
    </keybind>
    <keybind key="W-e">
      <action name="Execute">
        <command>pcmanfm</command>
      </action>
    </keybind>
    <keybind key="W-t">
      <action name="Execute">
        <command>urxvt -bg black -fg grey +sb -fn
            'xft:Monospace:pixelsize=16'</command>
      </action>
    </keybind>
    <keybind key="XF86AudioRaiseVolume">
      <action name="Execute">
        <command>amixer set Master 1%+</command>
      </action>
    </keybind>
    <keybind key="XF86AudioLowerVolume">
      <action name="Execute">
        <command>amixer set Master 1%-</command>
      </action>
    </keybind>
  </keyboard>
```

Et voici le même, au format INI.

```
[keybind]
KEY=W-d
ACTION=ToggleShowDesktop

```

```
[keybind]
KEY=A-F4
ACTION=Close

[keybind]
KEY=W-m
ACTION=ShowMenu
MENU=root-menu

[keybind]
KEY=A-Tab
ACTION=NextWindow
FINALACTION=Focus
FINALACTION=Raise
FINALACTION=Unshade

[keybind]
KEY=W-e
ACTION=Execute
COMMAND=pcmanfm

[keybind]
KEY=W-t
ACTION=Execute
COMMAND=urxvt -bg black -fg grey +sb -fn
    'xft:Monospace:pixelsize=16'

[keybind]
KEY=XF86AudioRaiseVolume
ACTION=Execute
COMMAND=amixer set Master 1%+

[keybind]
KEY=XF86AudioLowerVolume
ACTION=Execute
COMMAND=amixer set Master 1%-
```

À nouveau, le XML était-il indispensable ou particulièrement nécessaire pour un tel fichier de configuration ? J'en doute.

4. Zeste de Savoir

Zeste de Savoir emploie le format JSON afin de représenter la structure d'un contenu via le fichier `manifest.json` que vous retrouvez à la racine des archives ZIP de vos contenus. Par exemple, voici le fichier `manifest.json` produit pour ce billet.

```
{
    "licence":"CC 0",
    "slug":"json-yaml-toml-xml-ou-comment-se-compliquer-la-vie",
    "introduction":"introduction.md",

        "title":"JSON, YAML, TOML, XML,... Ou comment se compliquer la vie",
    "description":"",
    "version":2,
    "children":[
        {
            "slug":"ansible",
            "text":"ansible.md",
            "title":"Ansible",
            "object":"extract"
        },
        {
            "slug":"openbox",
            "text":"openbox.md",
            "title":"Openbox",
            "object":"extract"
        },
        {
            "slug":"zeste-de-savoir",
            "text":"zeste-de-savoir.md",
            "title":"Zeste de Savoir",
            "object":"extract"
        }
    ],
    "type":"OPINION",
    "conclusion":"conclusion.md",
    "object":"container"
}
```

Ce format si agréable à lire était-il indispenable ? Probablement pas.

```
[json-yaml-toml-xml-ou-comment-se-compliquer-la-vie]
LICENCE=CC-0
INTRODUCTION=introduction.md
TITLE=JSON, YAML, TOML, XML,... Ou comment se compliquer la vie
DESCRIPTION=
VERSION=2
TYPE=OPINION
CONCLUSION=conclusion.md

[json-yaml-toml-xml-ou-comment-se-compliquer-la-vie:ansible]
TEXT=ansible.md
TITLE=Ansible
```

```
13
14 [json-yaml-toml-xml-ou-comment-se-compliquer-la-vie:openbox]
15 TEXT=openbox.md
16 TITLE=Openbox
17
18 [json-yaml-toml-xml-ou-comment-se-compliquer-la-vie:zeste-de-savoir]
19 TEXT=zeste-de-savoir.md
20 TITLE=Zeste de Savoir
```

5. Conclusion

Alors, mon propos ici n'est pas de réaliser une ode au format INI, ni de prétendre que tous les choix techniques présentés ici sont foncièrement mauvais et encore moins d'affirmer connaître la complexité inhérente de chacun de ces trois projets.

Toutefois, trop souvent, le choix du format semble réalisé « par défaut », c'est-à-dire qu'on prend ce qui semble être le plus souvent utilisé ou dans le vent et pour le reste, « osef ». Or, foncièrement, les formats complexes que sont le JSON, le YAML, le TOML ou le XML ne sont finalement véritablement nécessaires que dans de rares cas où il est assez difficile de faire autrement (par exemple, le choix du XHTML pour les pages web n'est a priori pas déconnant, sans que cela soit parfait évidemment).

Donc, s'il vous plaît amis développeurs : quand vous devez stocker des données, choisissez un format textuel simple, le *parsing* se fera sans difficultés et votre vie et celle de vos utilisateurs n'en sera que meilleure.

Liste des abréviations

osef on s'en fout. 6