

TP n° 1

Le but de ce TP est de découvrir, par la pratique, certains concepts, en particulier du DOM. Ils seront revus en Web 2 avec le langage Javascript. Tous les exercices ne sont pas exigibles au niveau Lycée. Les exercices sont à faire dans un premier temps en Python (avec la bibliothèque standard et le package `xml.dom.minidom`). Il est possible de faire les exercices en OCaml (en utilisant les bibliothèques `ocaml-expat` et `PXP`), en C (en utilisant `libxml2`) et en Java (en utilisant la bibliothèque standard).

1 DOM

1.1 Chargement de fichier avec DOM

1. Écrire un programme Python `dom.py` qui charge un fichier XML en utilisant l'api DOM de Python. Faire en sorte qu'une fois le fichier chargé, le programme attende une entrée de lecture au clavier, puis se termine.
2. Compiler le programme `xmark_gen.c` fourni, puis générer des fichiers XML de taille croissante (en appelant le programme avec un paramètre `-f x` avec $0.0 \leq x \leq 1.0$).
3. Comparer la taille des fichiers générés avec l'occupation en mémoire et proposer une explication de la différence.

1.2 Parcours d'arbre DOM

1. Écrire un programme Python `dump_xml.py` qui charge un fichier XML dont le chemin est passé en argument au programme en utilisant l'api DOM de Python.
2. Écrire une fonction `toJSON` qui transforme l'arbre DOM en une structure de donnée Python de la forme suivante :
 - un élément `<tag att1="val1" att2="val2" ...>` est transformé en un dictionnaire `{"name" : "tag", "attributes" : {"att1": "val1", "att2": "val2", ...}, "children" : l}` où `l` est la liste des enfants de l'élément, récursivement transformés.
 - les nœuds texte sont des chaînes de caractères.On s'intéressera en particulier à la classe `xml.dom.Node` de l'API DOM de Python.
3. On rappelle qu'en Python, la pile d'appel est limitée à 1000 appels. Proposer un document qui provoque un *stack overflow* (si votre fonction `toJSON` est récursive). Comment régler ce problème ?
4. Ajouter une fonction `dump` qui prend en argument une valeur telle que renvoyée par `toJSON` et qui affiche cette valeur dans la console.

1.3 Création d'arbre DOM

1. Écrire un programme Python `xml_upper.py` qui prend en argument un fichier XML et qui crée une copie de ce fichier pour lesquels les noms des éléments et les nœuds texte sont en majuscules. On s'intéressera aux fonctions `xml.dom.getDOMImplementation()` et aux classes `xml.dom.DOMImplementation` et `xml.dom.Document` pour créer des nœuds.
2. A t'on vraiment besoin de charger tout le document initial en mémoire pour effectuer cette transformation ?

1.4 Parser SAX

Certaines opérations sur des fichiers XML sont exécutable en *streaming*. Dans ce contexte, cela signifie qu'on peut garder en mémoire uniquement une pile de taille proportionnelle à la profondeur de l'arbre correspondant au fichier XML (cette pile est nécessaire pour vérifier la bonne imbrication des balises).

Une API générique de chargement de fichiers XML nommée SAX (*simple API for XML*) est définie pour permettre de charger un fichier XML en *streaming*.

Fournir une implémentation du programme `xml_upper.py` en utilisant l'API `xml.sax` de Python. On s'intéressera à la fonction `parse` ainsi qu'à la classe `ContentHandler` de l'API.

Comparer l'empreinte mémoire de cette version et de la version utilisant le DOM sur des documents générés par `xmark_gen.c`.

2 DTD

Lire le fichier `movies.dtd` fourni dans l'archive.

1. Pour l'élément `name` comment changer la définition pour autoriser le nom et le prénom à apparaître dans n'importe quel ordre (mais exactement un nom et un prénom) ?
2. Même question pour `movie` (on veut autoriser les éléments à apparaître dans n'importe quel ordre, mais avec les même cardinalités que dans la DTD initiale)
3. Créer un fichier minimal valide. Ce dernier commencera par :

```
<!DOCTYPE movies SYSTEM "movies.dtd">
```

Vérifier sa validité au moyen de la commande `xmlstarlet` :

```
xmlstarlet val -d movies.dtd fichier.xml
```

Si la commande n'est pas installable sur votre système (*package* `xmlstarlet` sous Debian/Ubuntu) vous pouvez utiliser un validateur en ligne. Dans ce cas, il faut ajouter au début de votre fichier XML :

```
<?xml version="1.0"?>
<!DOCTYPE movie [
    ... contenu de votre DTD
]>
<movies>
    ... suite du fichier XML.
</movies>
```