Données Semi-Structurées:

Validation XML par rapport à une DTD avec Python

Enseignant: *Dario COLAZZO*
Chargée de TD/TP: *Beatrice NAPOLITANO*

25 Janvier 2021

1 Avant de commencer

Vérifiez que la dernière version de Python est bien installée (3.9.1)
`python -V  or  python3 -V`

Vérifiez que la dernière version de pip est bien installée (21.0). Pip est un gestionnaire de paquets Python qui est utilisé pour télécharger et installer des bibliothèques Python sur votre système local avec facilité, c'est-à-dire qu'il télécharge et installe également toutes les dépendances pour le paquet que vous installez. Pip doit être déjà installé, vérifiez avec :

`pip --version  or  pip3 --version`

— s'il n'est pas installé ou s'il se réfère à une version différente de Python, il faut l'installer :
Linux
`sudo apt install python3.9-pip`
Windows/macOS
https://pip.pypa.io/en/stable/installing/
— si ce n'est pas la dernière version :
Linux/macOS
`python3.9 -m pip install --upgrade pip`
Windows
`python3.9 -m pip install -U pip`

Vérifiez encore une fois si vous avez la bonne version.

2 Bibliothèque lxml

lxml est une bibliothèque Python écrite pour analyser des documents XML et HTML très rapidement, manipulant même des balises invalides au cours du processus. Elle est une extension de l'ancienne libxml2 et de libxsit et elle présente des avantages majeurs :

— API python très simple ;
— Bien documenté ;
— Pas besoin de gérer la mémoire ;

C'est également un moyen très naturel de gérer tout format de données XML. Les données sont automatiquement converties en types de données Python et peuvent être manipulées avec des opérateurs Python normaux.

2.1 Installation

— **Avec pip**
`$ pip install lxml` (or `pip3 install lxml`)

— **Avec apt-get**
Si vous utilisez Linux, vous pouvez installer lxml en tapant cette commande dans votre terminal
`$ sudo apt-get install python-lxml`

2.2 Importation

Pour importer et utiliser la bibliothèque :
`from lxml import etree`

3 Valider un document XML

Il y a plusieurs méthodes pour valider un document XML. Voici une fonction validate dans le document **validator.py** :

```
import sys, os, warnings
from lxml import etree

def validate(xml_path: str, dtd_path: str) -> bool:
        try:
                dtd = etree.DTD(open(dtd_path))
        except etree.DTDParseError as e:
                print("DTD Parse Error: ",e.error_log[0].message,
                " at line ",e.error_log[0].line)
                return False

        try:
                xml_doc = etree.parse(xml_path)
        except etree.XMLSyntaxError as e:
                print("XML Syntax Error: ",e)
                return False

        result = dtd.validate(xml_doc)
        if not result:
        print(dtd.error_log[0])

return result
```

Pour valider un document XML, il faut commencer par le charger. On va donc écrire un main :

```
def main():
        if len(sys.argv) != 3:
        print ("Usage: %s document.xml schema.dtd" % (sys.argv[0]))
        exit(0)

        xml_path = str((sys.argv[1]))
        dtd_path = str((sys.argv[2]))

        if validate(xml_path, dtd_path):
        print('Valid!')
        else:
        print('Not valid!')

if __name__ == "__main__":
        main()
```

3.1 Exemples

3.1.1 Exemple 1

dtddoc.dtd

```
1: <!ELEMENT address (name,company,phone)>
2: <!ELEMENT name (#PCDATA)>
3: <!ELEMENT company (#PCDATA)>
4: <!ELEMENT phone (#PCDATA)>
```

xmldoc.xml

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <!DOCTYPE address SYSTEM "dtddoc.dtd">
3: <address>
4: <name>Beatrice </name>
5: <company>Paris-Dauphine</company>
6: <phone>06 12345678</phone>
7: </address>
```

Si on exécute le programme :

```
$ python3 validator.py xmldoc.xml dtddoc.dtd
  Valid!
```

3.1.2 Exemple 2

dtddoc2.dtd

```
1: <!ELEMENT address EMPTY>
2: <!ATTLIST address name CDATA #REQUIRED>
3: <!ATTLIST address company CDATA #IMPLIED>
4: <!ATTLIST address phone CDATA #REQUIRED>
```

xmldoc2.xml

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <!DOCTYPE address SYSTEM "dtddoc2.dtd">
3: <address name="Beatrice" phone="06 12345678"/>
```

Si on exécute le programme :

```
$ python3 validator.py xmldoc2.xml dtddoc2.dtd
  Valid!
```

3.2 Erreurs

3.2.1 XML mal formé

Si le document XML est mal formé on aura un message d'erreur comme suit

```
XML Syntax Error: Opening and ending tag mismatch: Phone line 6 and phone, line 6,
column 30 (xmldoc.xml, line 6)
```

Cela indique qu'il y a une erreur syntaxique dans la ligne 6 du document xmldoc.xml. S'il y a plusieurs erreurs, le programme indique le premier rencontré ; le document doit donc être corrigé et re-testé.

3.2.2 DTD mal formé

Si la DTD est mal formé, par exemple on ajoute
`<!ATTLIST id (IDREF) #REQUIRED>`
on aura un message d'erreur comme suit

```
DTD Parse Error: ATTLIST: no name for Attribute at line 4
```

Cela indique que dans la définition de l'attribut id, il manque la référence à l'élément dont il dépend.

3.2.3 XML non valide par rapport à la DTD

Considérez la DTD "dtddoc2.dtd" et le document XML suivant
xmldoc3.xml

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <!DOCTYPE address SYSTEM "dtddoc2.dtd">
3: <address name="Beatrice" company="Paris-Dauphine"/>
```

Si on exécute le programme :

```
$ python3 validator.py xmldoc3.xml dtddoc2.dtd
   xmldoc3.xml:3:0:ERROR:VALID:DTD_MISSING_ATTRIBUTE: Element address does not carry
   attribute phone
   Not Valid!
```

on aura un message d'erreur expliquant pourquoi le document n'est pas valide.

4 Exercices

Écrivez un document XML et une DTD pour les applications suivantes et assurez-vous que vos documents sont valides par rapport à les DTD à l'aide du script python ci-dessus.

Exercice 1 (Exercice 3 du TD du 18/01) *:*
Envisager une application dans laquelle les résultats des matchs de football doivent être représentés en XML. Pour chaque ***jeu****, nous voulons être en mesure de représenter* ***les deux équipes*** *impliquées, l'équipe qui jouait* ***chez eux****, quels* ***joueurs*** *ont marqué des buts (dont certains peuvent avoir été* ***pénalités****) et le* ***moment*** *où chacun a été marqué, et quels joueurs ont reçu des* ***cartons jaunes*** *et* ***rouges****.*

Exercice 2 (Curriculum Vitae) *:*
Ecrire un document XML modélisant votre Curriculum Vitae (réfléchir à ce qui doit être mémorisé et comment structurer l'information). Ecrire la DTD associée et vérifier que votre document XML est valide par rapport à cette DTD.

Exercice 3 (Les brèves) *:*
Un site d'actualités veut présenter des nouvelles brèves, regroupées par thème. Quatre ***thèmes*** *sont possibles : actualités, sport, bourse et média. Chaque* ***brève*** *correspond à un unique thème. Les brèves peuvent être rédigées en* ***français*** *ou* ***anglais****, chacune est* ***datée*** *et possède un* ***titre****. Il est également possible d'illustrer une brève par une* ***photo*** *et de fournir une ou plusieurs* ***urls*** *vers des sites détaillant l'information : chaque url sera agrémentée d'une* ***courte phrase*** *résumant le contenu de la page pointée. Écrire une DTD et un document respectant cette DTD.*

Exercice 4 (Salle de concert) *:*
Envisager une application permettant de décrire la programmation d'une salle de concert. Chaque salle possède un ***nom****, un ensemble d'informations (****adresse****,* ***téléphone*** *etc.) et un texte présentant le* ***lieu*** *(son historique par exemple). Ensuite, nous voulons décliner une liste de concerts pour chaque salle. Un concert est caractérisé par un* ***titre****, une* ***date****, une* ***description****, un* ***tour de chant****, c'est-à-dire une liste de chansons/interprètes (en général, il n'y aura qu'un interprète par concert mais nous devons pouvoir traiter le cas de duos ou de chanteurs se succédant dans le cas d'une soirée caritative par exemple). Un artiste sera décrit par ses* ***nom****,* ***prénom****,* ***date de naissance****, éventuellement une* ***photo****, une* ***adresse de site web*** *si existant, et une* ***biographie*** *de l'artiste. Pour chaque chanson, on voudra disposer d'un* ***titre****, des* ***auteurs*** *et* ***compositeurs*** *et d'un bref* ***texte*** *(l'historique de la chanson et ses différents interprètes par exemple).*

Exercice 5 (Entreprise) *:*
Une entreprise fonctionne avec un certain nombre de ***personnes*** *et gère un certain nombre de* ***projets****. Une personne possède un* ***nom*** *et un ensemble d'informations personnelles qu'on ne détaillera pas. Un projet est caractérisé par un* ***nom*** *et un* ***descriptif****. Une personne peut participer à plusieurs projets et un projet est, en général, conduit par plusieurs personnes. Écrire une DTD, et après un document XML valide, qui décrit des documents destinés à publier les projets d'une entreprise ainsi que les personnes qui y travaillent. Connaissant une personne on doit pouvoir connaître immédiatement les projets auxquels elle participe. Connaissant un projet on doit pouvoir connaître immédiatement quels sont ses participants. Utiliser à bon escient les attributs de type* ***ID*** *et* ***IDREF****.*