BE1 XML MOD 8.4

Réalisé par : khaoula abouelfadl

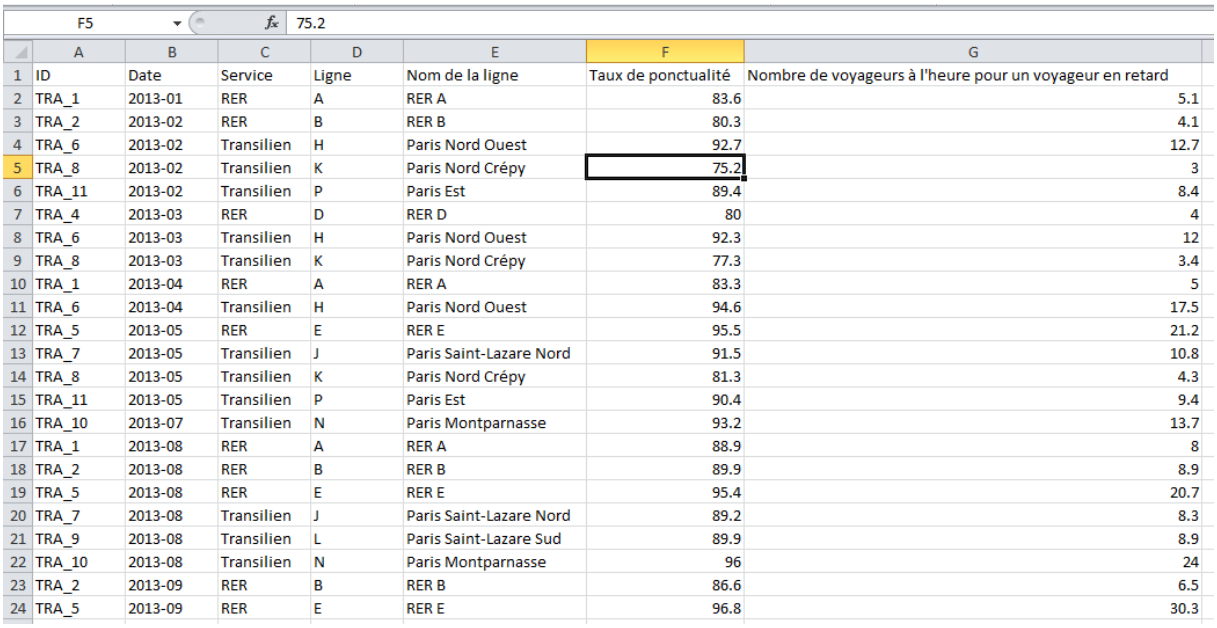
Table des matières

Introduction

Le but de ce Bureau d'étude est de créer une application XML pour ordonner des données de ponctualité du réseau de transports RER et Transilien. Partant de données structurées en CSV, on construit via Python un document XML. Celui-ci sera ensuite muni d'une DTD interne. Le document sera alors validé en ligne. Une dernière tâche consistera en la rédaction d'un script python pour valider directement le document XML obtenu.

1. Visualisation du document CSV

Le document CSV fourni est la matière première de ce bureau d'étude. Il recense plus de 400 mesures de ponctualité mensuelle de plusieurs lignes de RER et de Transilien. Chaque mesure contient des éléments d'identification du train, une date, une mesure de ponctualité ainsi que le nombre de voyageurs arrivant à l'heure pour un voyageur arrivant en retard. Le séparateur utilisé est le point virgule " ; ".

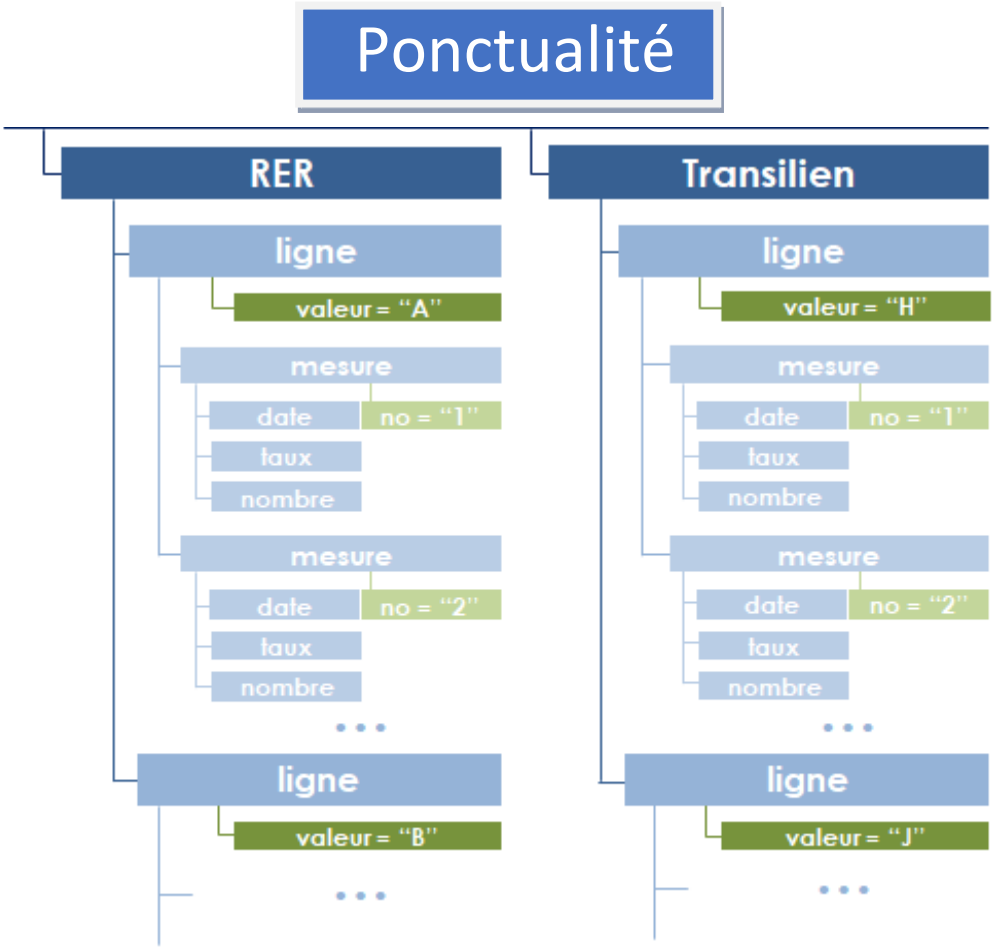


F5 | 75.2

	A	B	C	D	E	F	G
1	ID	Date	Service	Ligne	Nom de la ligne	Taux de ponctualité	Nombre de voyageurs à l'heure pour un voyageur en retard
2	TRA_1	2013-01	RER	A	RER A	83.6	5.1
3	TRA_2	2013-02	RER	B	RER B	80.3	4.1
4	TRA_6	2013-02	Transilien	H	Paris Nord Ouest	92.7	12.7
5	TRA_8	2013-02	Transilien	K	Paris Nord Crépy	75.2	3
6	TRA_11	2013-02	Transilien	P	Paris Est	89.4	8.4
7	TRA_4	2013-03	RER	D	RER D	80	4
8	TRA_6	2013-03	Transilien	H	Paris Nord Ouest	92.3	12
9	TRA_8	2013-03	Transilien	K	Paris Nord Crépy	77.3	3.4
10	TRA_1	2013-04	RER	A	RER A	83.3	5
11	TRA_6	2013-04	Transilien	H	Paris Nord Ouest	94.6	17.5
12	TRA_5	2013-05	RER	E	RER E	95.5	21.2
13	TRA_7	2013-05	Transilien	J	Paris Saint-Lazare Nord	91.5	10.8
14	TRA_8	2013-05	Transilien	K	Paris Nord Crépy	81.3	4.3
15	TRA_11	2013-05	Transilien	P	Paris Est	90.4	9.4
16	TRA_10	2013-07	Transilien	N	Paris Montparnasse	93.2	13.7
17	TRA_1	2013-08	RER	A	RER A	88.9	8
18	TRA_2	2013-08	RER	B	RER B	89.9	8.9
19	TRA_5	2013-08	RER	E	RER E	95.4	20.7
20	TRA_7	2013-08	Transilien	J	Paris Saint-Lazare Nord	89.2	8.3
21	TRA_9	2013-08	Transilien	L	Paris Saint-Lazare Sud	89.9	8.9
22	TRA_10	2013-08	Transilien	N	Paris Montparnasse	96	24
23	TRA_2	2013-09	RER	B	RER B	86.6	6.5
24	TRA_5	2013-09	RER	E	RER E	96.8	30.3

2. Structure envisagée du document XML

Avant d'utiliser Python comme levier pour structurer un document XML à partir du document CSV fourni, il faut définir l'arborescence XML envisagée dans le document qui sera produit.



L'élément racine est nommée ponctualité.

Celui-ci contient deux sous éléments *RER* et *Transilien* faisant référence aux deux types de train en circulation. Ces deux éléments ont la même organisation interne.

Plusieurs éléments *ligne* recensent les multiples lignes de chaque catégorie en circulation via un attribut qui prend en valeur la lettre d'identification de la ligne. Notons qu'en termes d'identification des lignes en circulation, les trois critères "ID", "Ligne" et "Nom de la ligne" sont redondants dans le document CSV. Nous avons choisi de ne conserver que "Ligne".

Chaque ligne contient ensuite l'ensemble des mesures la concernant. Chaque mesure est modélisée par un sous-élément *mesure* numéroté avec un attribut. Les informations de date,

de taux de ponctualité et du nombre de voyageurs à l'heure pour un voyageur en retard font l'objet d'un sous-élément chacun.

3. Import en listes structurées du document .csv via Python

Le script python employé pour réaliser cet import est le suivant :

```
### Importation des données CSV ###
#On ouvre le fichier CSV
with open('ponctualite-mensuelle-transilien.csv', newline='') as csvfile:
    spamreader=csv.reader(csvfile,delimiter=';', quotechar="|")
    for row in spamreader:

        for t in range(len(ListeTrain)):
            #On remplit la liste ListeRERTransit
            if row[2]==ListeTrain[t] and row[3] not in ListeRERTransit[t]:
                ListeRERTransit[t]+=[row[3]]
                L[t]+=[[]]
            ListeRERTransit[t].sort()
            #On remplit la liste de donnée
            for k in range (len(ListeRERTransi[t])):
                if row[2]==ListeTrain[t] and row[3]==ListeRERTransit[t][k]:
                    L[t][k]+=[[row[1],row[5],row[6]]]
```

L'ensemble du code python commenté employé pour l'import structuré du csv et son export en xml est disponible dans le rendu et sur le notebook.

4. Ecriture du document XML à partir des listes structures

Pour l'écriture du document XML de ponctualité des différentes lignes de transport, on utilise le module python lxml.etree. Comme vu précédemment, la stucture en liste emboitées des données importées reflète la structure prévisionnelle du document XML à créer.Ainsi, le script python fourni ci dessous parcourt les différents étages des listes emboitées pour créer les éléments et redistribuer les données et attributs sous forme d'arborescence XML

5. Ecriture d'une DTD interne au document XML créé

Le fichier XML obtenu en sortie du script Python n'est composé que d'une arborescence XML d'éléments et d'attributs qui structure les données de ponctualités des trains. Si la syntaxe est correcte, ce fichier nécessite néanmoins une DTD avant d'être valide. Une DTD interne est ajoutée en tête de document ainsi qu'un prologue pour vérifier la version d'XML utilisée, l'encodage utilisé et le fait que le fichier se suffit à lui-même.

La DTD interne ajoutée est la suivante :

```
<!DOCTYPE ponctualite [
    <!ELEMENT ponctualite (RER,Transilien)>
      <!ELEMENT RER (ligne+)>
      <!ELEMENT Transilien (ligne+)>
        <!ELEMENT ligne (mesure+)>
        <!ATTLIST ligne valeur (A|B|C|D|E|H|J|K|L|N|P|R|U) #REQUIRED>
          <!ELEMENT mesure (date, taux, nombre)>
          <!ATTLIST mesure no NMTOKEN #REQUIRED>
            <!ELEMENT date (#PCDATA)>
            <!ELEMENT taux (#PCDATA)>
            <!ELEMENT nombre (#PCDATA)>
    ]>
```

Le fichier XML assorti d'une DTD interne est alors été bien validé en ligne avec le validateur du w3c

6. Validation directe via Python

On construit avec le module lxml.etree un valideur direct via Python. Dans un premier temps on importe le fichier XML pourvu de sa DTD interne, puis on isole d'une part la DTD et le XML dans deux chaines séparées. On peut alors utiliser les fonctions de lxml.etree pour vérifier la conformité du xml à sa DTD. Enfin, le script affiche si le document est ou non conforme, et, en cas de non-conformité, une explication des erreurs repérées. Ainsi, le script python a été bien fourni dans le notebook ci-joint dans ce fichier.