Licence 3 d'Informatique — XML — TP 3

Début : mardi 14 mars 2023

Cette dernière séance de travaux pratiques vise à vous familiariser avec les rudiments du langage XML Schema, ce langage offrant une alternative aux DTD [1], alternative beaucoup plus satisfaisante pour la spécification des types attachés aux contenus des éléments et des attributs d'un document en XML [2]. En outre, les schémas sont eux-mêmes des documents XML, alors que les fichiers DTD utilisent la syntaxe du langage EBNF [3]. L'exemple que nous allons utiliser reprend la spécification de bouteilles d'eau minérale, sur laquelle vous avez travaillé précédemment en travaux dirigés. Comme de coutume, les fichiers mentionnés ci-après ont été réunis en un fichier d'archive .tar [4] :

for-lc-3.tar

à récupérer sur le cours « Programmation fonctionnelle, scripts --- XML » de moodle, dans le répertoire « XML - semestre 6 > Travaux pratiques 2023 ». Il contient les trois fichiers suivants :

water.xml water-bottles.dtd water-bootles.xsd

1 Corriger et compléter un schéma

Les figures 1 & 2 donnent une première version du schéma water-bottles.xsd, décrivant notre organisation pour spécifier des bouteilles d'eau minérale. Il s'agit en réalité d'une version *inachevée* : certaines définitions sont clairement incomplètes, aucune définition ne tient compte du fait que quelques éléments sont facultatifs ou répétés. Nous vous rappelons que de telles possibilités sont spécifiées par les attributs `minOccurs` et `maxOccurs` de l'élément `xsd:element` du langage XML Schema. Votre premier exercice consiste donc à compléter le fichier représenté dans les figures 1 & 2, afin d'obtenir un schéma permettant la validation du texte water.xml, qui est une version légèrement retouchée par rapport à celle que vous avez utilisée durant les premières séances de travaux dirigés. Nous vous avons aussi fourni une version légèrement retouchée du fichier water-bottles.dtd, vu en travaux dirigés, version de laquelle vous pouvez vous inspirer pour compléter le schéma, et qui nous servira au § 3.

2 Améliorer un schéma

Alors que le langage des fichiers DTD est très pauve quant à la spécification des types simples pour les valeurs associées aux attributs ou aux contenus textuels des éléments, les améliorations

1. ***D**ocument **T**ype **D**efinition.*
2. *e**X**tensible **M**arkup **L**anguage.*
3. ***E**xtended **B**ackus-**N**aur **F**orm.*
4. ***T**ape **AR**chive.*

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="water-bottles">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="water-bottle" ...>
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="name" .../>
              <xsd:element name="capacity" type="measurement-type"/>
              <xsd:element name="analysis">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="contains" ...>
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="body" type="..."/>
                          <xsd:element name="mass-concentration" type="measurement-type" .../>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="comes-from">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="town" type="..."/>
                    <xsd:choice>
                      <xsd:element name="french-department">
                        <xsd:complexType>
                          <xsd:simpleContent>
                            <xsd:extension base="xsd:token">
                              <xsd:attribute name="no" use="..." type="xsd:string"/>
                            </xsd:extension>
                          </xsd:simpleContent>
                        </xsd:complexType>
                      </xsd:element>
                      <xsd:element name="division">...</xsd:element>
                    </xsd:choice>
                  </xsd:sequence>
                  <xsd:attribute name="country" type="xsd:token" use="..."/>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="barcode" type="xsd:token"/>
              <xsd:element name="pH">
                <xsd:complexType>
                  <xsd:attribute name="value" use="..." type="xsd:decimal"/>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
```

Figure 1 : Schéma water-bottles.xsd, version incomplète (début).

```
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="measurement-type">
    <xsd:attribute name="unit" use="..." type="xsd:string"/>
    <xsd:attribute name="value" type="..." use="..."/>
  </xsd:complexType>

</xsd:schema>
```

Figure 2 : Schéma water-bottles.xsd, version incomplète (fin).

---

suivantes que vous allez apporter au fichier water-bottles.xsd vont vous montrer toute la puissance du langage XML Schema. Quelques-unes des améliorations suggérées ci-après utilisent les *expressions régulières* de langages modernes de programmation tels que C#, Java, Perl [5], Python ou Ruby. Rappelons que par rapport aux conventions utilisées dans les expressions régulières de ce dernier langage :

— le caractère « / » n'est pas un caractère spécial des expressions régulières de XML Schema comme il l'est dans celles de Ruby où il joue le rôle de délimiteur ; en conséquence, il ne doit pas y être précédé de « \ » ; par contre, ce sont des *entités simples* (« `<` », « `>` », « `&apos;` », « `&quot;` », « `&` ») qui doivent être utilisées pour les caractères spéciaux de XML (« < », « > », « ' », « " », « & ») ;

— c'est *toute la chaîne* qui doit être mise en correspondance avec l'expression régulière spécifiée par l'attribut `value` de l'élément `xsd:pattern` de XML Schema, aussi les marqueurs « `\A` » et « `\z` » (ou « `^` » et « `$` ») de début et fin de chaîne sont implicites et n'ont pas à être mentionnés.

Et nous livrons à présent le programme des réjouissances à la foule en délire.

1. Nous avons défini le contenu de l'attribut `value` de l'élément `pH` [6] comme un nombre décimal quelconque. Ce qui est beaucoup trop large car en pratique, une mesure de pH est toujours comprise entre 0 (exclus) et 14 (inclus). Modifiez la spécification afin qu'elle incorpore le test entre ces deux bornes :

```
<xsd:attribute name="value" use="required">
  <xsd:simpleType>
    <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="..."/><xsd:maxInclusive value="..."/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

5. ***P**ractical **E**xtraction and **R**eport **L**anguage.*
6. **P**otentiel d'**H**ydrogène.

2. De même, la valeur associée à l'attribut `country` est le code ISO [7] d'un pays, toujours formé de deux lettres capitales [8]. Incorporez cette contrainte dans le schéma :

```
<xsd:attribute name="country">
  <xsd:simpleType>
    <xsd:restriction base="xsd:token">
      <xsd:pattern value="..."/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

3. Puis vérifiez que si dans la déclaration précedente, « `base="xsd:token"` » est remplacé par « `base="xsd:string"` » ou « `base=xsd:normalizedString"` », alors la validation du fragment suivant :

```
<comes-from country=" FR ">...</comes-from>
```

— remarquez les caractères d'espacement dans la valeur associée à l'attribut `country` — n'a pas lieu alors qu'elle est tout à fait possible avec « `base="xsd:token"` ».

4. Spécifiez à présent qu'un code barre — le contenu de l'élément `barcode` — est formé de 13 chiffres.

5. Nous allons maintenant nous concentrer sur le type `measurement-type` et convenir d'une taxonomie pour les valeurs associées à l'attribut `unit`, spécifiant des unités en Physique. Nous n'allons faire aucun test sur les noms des unités « simples » — le mètre, le millimètre, *etc.* — que nous considérerons tout simplement comme des *mots*. Par contre, nous allons convenir qu'une unité utilisée par l'attribut `unit` de l'élément `mass-concentration` est un produit d'unités simples, figuré par un point (« `.` »). Les exposants différents de 1 seront donnés entre parenthèses. De plus, nous n'utiliserons pas la barre de division et indiquerons les unités au dénominateur au moyen d'exposants négatifs. À titre d'exemples, voici comment exprimer des unités de moment de force, vitesse, viscosité cinématique, accélération, masse volumique :

Newton-mètre	`N.m`
mètre par seconde	`m.s(-1)`
mètre carré par seconde	`m(2).s(-1)`
mètre par seconde par seconde	`m.s(-2)`
milligramme par litre	`mg.l(-1)`

Mettez à jour dans ce sens la spécification du type complexe `measurement-type` du schéma water-bottles.xsd.

6. Pour finir, nous allons tenter de donner une définition aussi complète et précise des numéros des départements français. Nous rappelons que selon les pays d'origine d'une bouteille, nous pouvons utiliser les éléments `department` ou `french-departement`, à contenu identique.

7. ***I**nternational **S**tandardisation **O**rganisation.*

8. Par exemple, « `FR` » pour la France, « `UK` » pour le Royaume-Uni, « `DE` » pour l'Allemagne, « `CN` » pour la République de Chine Populaire.

L'élément `french-department` possède en outre un attribut obligatoire donnant le numéro. Une première spécification du contenu de cet attribut `no` a été donnée dans le fichier water-bottles.dtd : il s'agit de l'adapter au fichier water-bottles.xsd en lieu et place de la spécification simplifiée qui vous a été donnée. En ce qui concerne la taxonomie de ces numéros, nous rappelons que :

— ils doivent être considérés comme des chaînes de caractères : par exemple, le numéro du premier département (*Ain*) est `01`, pas « `1` » ;

— le département de numéro `20` n'existe plus [9] : il a été subdivisé en deux départements `2A` (*Corse du Sud*) et `2B` (*Haute-Corse*) ;

— après le dernier département métropolitain (`95` pour le *Val d'Oise*) viennent des départements d'Outre-Mer, avec une numérotation parfois interrompue car certains numéros ont été supprimés ou correspondent à des *territoires d'Outre-Mer* qui ne sont pas des départements. Voici la liste des *départements d'Outre-Mer* :

`971`	Guadeloupe	`973`	Guyane	`976`	Mayotte
`972`	Martinique	`974`	La Réunion		

3 Et pour finir...

Plusieurs artifices permettent de valider le document XML water.xml aussi bien avec le fichier DTD water-bottles.dtd qu'avec le schéma water-bottles.xsd. Quels sont-ils et pourquoi ne sont-ils pas complètement satisfaisants ?

9. Anciennement, c'était la *Corse*.