G. Hunault *Angers, février 2021*

Licence 2 Informatique

Technologie XML

1. Ecriture de DTD externe

On veut modéliser en XML des fichiers de contacts avec des amis et des adresses géolocalisées. Voici un exemple très court d'un fichier XML possible :

```
<?xml version="1.0" ?>
<amisEtLieux>

  <amis>
     <ami surnom="jojo" ville="Angers">
        <prenom>Georges Henri</prenom>
        <nom>GRANDJEAN</nom>
        <adresse>5 rue Champollion</adresse>
     </ami>
     <ami surnom="kiki" ville="paris">
        <prenom>Christine</prenom>
        <nom>OMEROUE WANEA</nom>
        <adresse>12 boulevard Voltaire</adresse>
     </ami>
     <ami ville="Paris XI">
        <prenom>Eva</prenom>
        <nom>ADAM</nom>
     </ami>
  </amis>

  <lieux>
      <lieu nom="Angers"     latitude="47.4667" longitude="-0.55"   />
      <lieu nom="Nantes"     latitude="47.2173" longitude="-1.5534" />
      <lieu nom="Paris"      latitude="48.8534" longitude="2.3488"  />
      <lieu nom="Strasbourg" latitude="48.5833" longitude="7.75"  />
  </lieux>

</amisEtLieux>
```

Donner pour ce fichier une grammaire DTD « *minimale et raisonnable* ». Vous discuterez soigneusement la cardinalité des éléments, des attributs et le typage des attributs à partir de cet exemple avant de fournir la grammaire.

2. Un peu de XPATH *via* xmllint

Donner les instructions de xmllint exécutées en mode *shell* qui permettent de répondre aux questions suivantes que l'on se pose sur un fichier d'amis et de lieux structuré comme dans la section précédente.

Attention : il ne s'agit évidemment pas du fichier précédent mais d'**un autre fichier** structuré de la même façon. On ne doit donc pas fournir les réponses aux questions mais donner le code xmllint qui fournirait les réponses.

1. quelle est la latitude de la ville "Rouen" ?
2. combien y a-t-il de noms de lieux dans le fichier ?
3. combien y a-t-il d'amis qui ont un surnom ?
4. quelle est la ville pour l'ami dont le surnom est "Gigi" ?
5. quelle est la longitude de la ville pour l'ami surnommé "Freddy" ?
6. quelles sont les villes dont la latitude est supérieure à 47 ?

3. Discussion sur la structuration

Essayer d'expliquer pourquoi la structuration hiérarchique fournie via le fichier XML de la question 1 est sans doute maladroite et peu correcte conceptuellement.

Vous expliquerez ensuite comment mieux structurer et vous fournirez un fichier avec les mêmes informations que celles du fichier XML de la question 1 structurées selon votre solution.

4. Une autre structuration pour les contacts

Donner le contenu XML d'un fichier amis03.xml qui suit le schéma amis.xsd fourni en annexe avec les informations suivantes :

- l'ami de nom VALYET et de prénom Pierre a pour surnom Pierrot et il habite Strasbourg.

- l'amie Paula dont le nom est HANDERSON habite Limoges et n'a pas de surnom.

- la ville Strasbourg dont l'attribut idVille est 67000 a pour latitude 48.5833 et pour longitude 7.75.

- la ville Limoges a pour idVille la valeur 87000.

- la ville Lille a pour idVille la valeur 59350.

5. Transformations XSL (1)

On voudrait produire un fichier qui résume les amis, les villes et les surnoms des amis qui habitent Paris (y compris si l'orthographe est paris). Donner le contenu d'un fichier XSL qui effectue une transformation d'un fichier structuré comme amis01.xml de la question 1 et qui produit le fichier texte suivant où l'absence de surnom correspond à une chaine vide.

On essaiera de respecter le formatage des lignes, mais ce n'est pas grave s'il manque des espaces ou s'il y en a en trop.

```
Fichier des amis et des lieux :

 - 3 amis ;
 - 4 villes.

Voici les gens qui habitent Paris, format nom (surnom) :

   .  OMEROUE (kiki)
   .  ADAM ()
```

Si vous n'arrivez pas à tout reproduire, ce n'est pas grave. Essayez de produire ce que vous pouvez. Il vaut mieux fournir un code XSL plus court mais correct et fonctionnel qu'un code long mais incorrect.

6. Transformations XSL (2)

Donner le code d'une transformation XSL qui produit l'extrait de fichier HTML suivant pour les contacts d'un fichier structuré comme à la question 1.

```
<h2>Choisis une ville parmi les 4 villes de tes 3 amis :</h2>
<div>

  <form action="localiseAmis.php" method="get">
    <select id="villes" name="villes">
      <option value="Angers">Angers</option>
      <option value="Nantes">Nantes</option>
      <option value="Paris">Paris</option>
      <option value="Strasbourg">Strasbourg</option>
    </select>
    <input type="submit"/>
  </form>

</div>
```

Il s'agit donc de produire un élément <h2> puis un formulaire dans une division. L'élément <h2> doit afficher le nombre d'amis et de lieux présents dans le fichier XML. Le formulaire doit contenir une liste de sélection générée automatiquement à partir des lieux du fichier XML. La liste doit être triée par ordre alphabétique.

7. Discussion

Essayez de répondre à la question suivante :

> *Est-ce que XML risque de disparaitre dans les prochaines années au profit d'un JSON amélioré ?*

Votre réponse devra mettre en évidence votre culture naissante ainsi que votre recul et votre esprit de synthèse en matière de modélisation et de traitement d'informations structurées.

Le texte de votre réponse devra faire 10 lignes au minimum, sans limite de maximum. Il devra contenir au moins 3 mots de 4 syllabes ou plus pour transmettre ≪ un contenu rédactionnel fort ≫.

Vous utiliserez des **exemples concrets** pour essayer de justifier votre avis.

ANNEXE : Schéma amis.xsd

```
<?xml version="1.0" encoding="UTF-8"?><!-- examen L2 XML 02/2021 -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="amisEtVilles">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="amis"/>
        <xs:element ref="villes"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="amis">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element ref="ami"/>
        <xs:element ref="amie"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>

  <xs:element name="ami">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="prenom"/>
        <xs:element ref="nom"/>
      </xs:sequence>
      <xs:attribute name="refVille" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="amie">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="prenom"/>
        <xs:element ref="nom"/>
      </xs:sequence>
      <xs:attribute name="refVille" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="villes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="ville"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="ville">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:NCName">
          <xs:attribute name="idVille" use="required" type="xs:integer"/>
          <xs:attribute name="latitude" use="required"/>
          <xs:attribute name="longitude" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="prenom" type="xs:NCName"/>

  <xs:element name="nom">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:NCName">
          <xs:attribute name="surnom" type="xs:NCName"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

ESQUISSE DE SOLUTION

1. Ecriture de DTD externe

Dans l'exemple fourni, l'élément racine amisEtLieux ne comporte que deux éléments, nommés respectivement amis et lieux. L'élément amis (avec un "s") ne contient que des éléments ami (sans "s") et on peut supposer qu'il y en a toujours au moins 1, donc la cardinalité sera +. De même pour l'élément lieux (avec un "x") qui ne contient que des éléments lieu (sans "s").

Dans l'exemple, les éléments ami contiennent toujours un attribut ville qui sera donc REQUIRED alors que l'attribut surnom est parfois absent donc sa cardinalité sera IMPLIED. Sans plus de renseignements, on typera ces attributs en CDATA. Tous les éléments ami contiennent systématiquement un élément prenom, un élément nom et éventuellement un élément adresse. Leur type sera #PCDATA.

Tous les éléments lieu sont vides et ne comportent que des attributs. Les trois attributs nom, latitude et longitude sont toujours présents et seront déclarés REQUIRED. Ils contiennent du texte que nous typerons en CDATA car il peut y avoir plus d'un mot.

Une grammaire possible est :

```
<!-- grammaire pour des amis et des lieux
                              (gH) 02/2021
                                          -->

<!ELEMENT amisEtLieux (amis,lieux)          >
<!ELEMENT amis        (ami)+                >
<!ELEMENT ami         (prenom,nom,adresse?) >
<!ELEMENT lieux       (lieu)+               >
<!ELEMENT lieu        EMPTY                 >
<!ELEMENT prenom      (#PCDATA)             >
<!ELEMENT nom         (#PCDATA)             >
<!ELEMENT adresse     (#PCDATA)             >

<!ATTLIST ami    surnom     CDATA  #IMPLIED   >
<!ATTLIST ami    ville      CDATA  #REQUIRED  >
<!ATTLIST lieu   nom        CDATA  #REQUIRED  >
<!ATTLIST lieu   latitude   CDATA  #REQUIRED  >
<!ATTLIST lieu   longitude  CDATA  #REQUIRED  >
```

2. Un peu de XPATH *via* xmllint

Voici les commandes xmllint demandées :

```
1.  ls //lieu[@nom="Rouen"]/@latitude

2.  xpath count(//lieu[@nom]) # ou  xpath count(//@nom)

3.  xpath count(//ami[@surnom])

4.  ls //ami[@surnom="Gigi"]/@ville

5.  ls //lieu[@nom=//ami[@surnom="Freddy"]/@ville]/@longitude

6.  ls //lieu[number(@latitude)>47]/@nom
```

3. Discussion sur la structuration

La structuration fournie à la question est certainement maladroite dans le traitement des villes. Par exemple paris est parfois écrit avec l'initiale en majuscule et d'autres fois en minuscule. On trouve aussi "Paris" et "Paris XI" comme contenu texte. Comme pour une base de données, il serait plus correct de définir un **identifiant** (id) de ville pour les lieux et d'utiliser une **référence** à cet identifiant pour les amis. Voici un exemple de fichier mieux structuré :

```
<amisEtLieux>
  <amis>
     <ami surnom="jojo" refVille="1">
        <prenom>Georges</prenom>
        <nom>GRANDJEAN</nom>
        <adresse>5 rue Champollion</adresse>
     </ami>
[...]
  </amis>
  <lieux>
      <lieu idVille="1" nom="Angers"  latitude="47.4667" longitude="-0.55"   />
      <lieu idVille="2" nom="Nantes"  latitude="47.2173" longitude="-1.5534" />
[...]

  </lieux>
</amisEtLieux>
```

4. Une autre structuration pour les contacts

Le fichier demandé ressemble certainement au texte suivant :

```
<?xml version="1.0" ?>
<amisEtVilles>

  <amis>
     <ami refVille="67000">
        <prenom>Pierre</prenom>
        <nom surnom="Pierrot">VALEYET</nom>
     </ami>
     <amie refVille="87000">
        <prenom>Paula</prenom>
        <nom>HANDERSON</nom>
     </amie>
  </amis>

  <villes>
      <ville idVille="59350" latitude="" longitude="">Lille</ville>
      <ville idVille="87000" latitude="" longitude="">Limoges</ville>
      <ville idVille="67000" latitude="48.5833" longitude="7.75">
             Strasbourg</ville>
  </villes>

</amisEtVilles>
```

En effet la grammaire XSD déclare l'élément racine `amisEtVilles` comme composé de deux éléments seulement, `amis` et `villes` dans cet ordre (`sequence`).

Un élément `amis` comporte, dans un ordre quelconque (`choice`) des éléments `ami` ou `amie` qui contiennent dans cet ordre un élément `prenom` et un élément `nom` avec un attribut `refVille` obligatoire (`use="required"`) qui doit être un entier.

L'élément `nom` peut avoir de façon facultative, un attribut `surnom`.

L'ément `villes` ne comporte que des éléments `ville`.

Un élément `ville` comporte un contenu texte simple et trois attributs **obligatoires**, à savoir `idVille`, `latitude` et `longitude`.

5. Transformations XSL (1)

```
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version="1.0">
<xsl:output method="text" encoding="UTF-8" />

<xsl:variable name="nba"><xsl:value-of select="count(//ami)"  /></xsl:variable>
<xsl:variable name="nbv"><xsl:value-of select="count(//lieu[@nom])"     /></xsl:variab

<xsl:template match="/">Fichier des amis et des lieux :

 - <xsl:value-of select="$nba" /> amis ;
 - <xsl:value-of select="$nbv" /> villes.
 <xsl:text>&#10;</xsl:text>

 Voici les gens qui habitent Paris, format nom (surnom) :
<xsl:apply-templates select="//ami[@ville='Paris' or @ville='paris']" />

</xsl:template>

<xsl:template match="ami">
   <xsl:text>   .  </xsl:text>
   <xsl:value-of select="nom" />
   <xsl:text> (</xsl:text>
   <xsl:value-of select="@surnom" />
   <xsl:text>) &#10;</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

6. Transformations XSL (2)

```
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version="1.0">
<xsl:output method="xml" encoding="UTF-8" indent="yes" omit-xml-declaration="yes" />

<xsl:variable name="nba"><xsl:value-of select="count(//ami)"  /></xsl:variable>
<xsl:variable name="nbv"><xsl:value-of select="count(//lieu[@nom])"     /></xsl:variab

<xsl:template match="/">
<h2>Choisis une ville parmi les <xsl:value-of select="$nbv" /> villes de tes <xsl:valu
<xsl:text>&#10;</xsl:text>
<div>
```

```
  <form action="localiseAmis.php" method="get">
      <select id="villes" name="villes">
          <xsl:apply-templates select="//lieu">
               <xsl:sort />
          </xsl:apply-templates>
      </select>
      <input type="submit" />
  </form>
</div>
</xsl:template>

<xsl:template match="lieu">
   <xsl:element name="option">
    <xsl:attribute name="option">
        <xsl:value-of select="@nom" />
    </xsl:attribute>
    <xsl:value-of select="@nom" />
   </xsl:element>
</xsl:template>

</xsl:stylesheet>
```

On remarquera que cette deuxième transformation se base uniquement sur les villes des éléments <lieu> même si aucun ami n'y habite.

Une transformation plus correcte ne devrait afficher que les villes où on a des amis, c'est-à-dire n'afficher que Angers et Paris pour l'exemple présenté à la question 1.