G. Hunault
Licence 2 Informatique

Angers, mai 2019

Technologie XML

1. Ecriture de DTD externe

On dispose de fichiers XML qui décrivent des repas de façon simplifiée. On trouvera ci-dessous un exemple de fichier correspondant, nommé repas.xml.

```
<?xml version="1.0" ?>
<jours>
  <jour>
     <matin heure="8">
       <aliment refid="A50">pain</aliment>
       <aliment refid="A80">chocolat</aliment>
     </matin>
     <midi heure="12 h 30">
       <aliment>viande</aliment>
     </midi>
     <soir><aliment>soupe</aliment></soir>
  </jour>
  <jour numero="52">
     <midi><aliment>poisson</aliment></midi>
     <soir heure="19h30">
       <aliment>viande</aliment>
     </soir>
  </jour>
  <jour numero="72">
     <midi>
     <aliment>pain</aliment><aliment>chocolat</aliment>
     </midi>
     <soir heure="21h"><aliment>soupe</aliment></soir>
  </jour>
</jours>
```

Donner pour ce fichier une grammaire DTD « *minimale et raisonnable* ».

Vous discuterez soigneusement la cardinalité des éléments et 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 de repas structuré comme dans la section précédente.

Attention : il ne s'agit évidemment pas du fichier précédent mais d'un 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. à quel aliment correspond la référence d'identifiant de valeur "A170" ?
2. combien y a-t-il d'élements `<jour>` dans le fichier ?
3. combien y a-t-il d'élements `<jour>` dans le fichier dont le numéro est supérieur à 80 ?
4. combien d'aliments ont été consommés pour le jour numéro 26 ?
5. quel est le dernier aliment consommé pour le soir du jour numéro 253 ?
6. quels sont les numéros de jours pour lesquels on a consommé du pain et du chocolat, que ce soit le matin, le midi ou le soir, mais pas nécessairement pendant le même repas ?

3. D'autres repas

Donner le contenu XML d'un fichier repas03.xml qui suit le schéma repas.xsd fourni en annexe avec les informations suivantes :

- le premier élément jour n'as pas de date, les aliments consommés au petit-déjeuner sont le lait et le miel ;
- le second jour a "19-05-04" comme date, le repas de midi a été pris à 13 h 30, les aliments étaient de la viande (unique) et des fruits (multiples).

4. TLR : tous les repas

On voudrait disposer d'un fichier tlr.xml constitué dynamiquement de deux fichiers de repas nommés repas01.xml et repas02.xml structurés comme à la question numéro 1 à l'aide d'une DTD interne et de deux entités caractères. Donner le texte complet de ce fichier tlr.xml. On nommera `<tousLesRepas>` l'élément racine de ce fichier.

5. Expressions régulières pour les heures de repas

A la question 1, il y avait plusieurs exemples d'heures de repas, à savoir "8", "12 h 30", "19h30", "21h". Donner des expressions régulières qui permettent de modéliser ces exemples. On notera qu'il n'y a jamais d'espace en début ou en fin, qu'il y en a au plus un, facultatif, avant et après le "h" et que le "h" est aussi facultatif. Voici des chaines qui ne sont pas acceptables : "8 ", " 5", "42", "9h72". Si cela vous peut vous aider, on acceptera "09" et "09h30".

En admettant que cette expression régulière est notée XXXX, comment définirait-on dans un schéma XSD le type simple **heureRepas** pour les chaines de caractères qui correspondent à cette expression régulière ? On se limitera ici juste au code entre `<xsd:simpleType...>` et `</xsd:simpleType>`.

6. Transformations XSL (1)

On voudrait produire un fichier texte qui résume les jours, les repas et les aliments. Donner le contenu d'un fichier XSL qui effectue une transformation du fichier repas.xml de la question 1 et qui produit le fichier texte suivant.

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.

```
Dans le fichier des repas, il y a :
  - 3 jours ;
  - 7 repas ;
  - 9 aliments, distincts ou non.
```

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.

7. Transformations XSL (2)

Que faudrait-il changer, dans la tranformation précédente, pour calculer le nombre d'éléments aliments distincts au lieu du nombre d'éléments aliments en tout ? Pour l'exemple proposé, on devrait trouver 5 (les 5 éléments sont chocolat, pain, poisson, soupe et viande).

Indépendamment de XSL, quelle commande Unix qui chaine xmlstarlet et sort permet d'obtenir la liste triée des aliments uniques pour le fichier repas.xml de la question 1 ?

8. Transformations XSL (3)

Comment produire en XHTML strict un tableau qui affiche, jour par jour, le nombre d'aliments consommés à chaque repas ? Voici ce qu'on devrait obtenir pour l'exemple du fichier repas.xml de la question 1 :

Nombre d'aliments par repas

Jour	Nombre d'Aliments	Repas		
		Matin	Midi	Soir
???	4	2	1	1
52	2	0	1	1
72	3	0	2	1

La partie entêtes du tableau (les deux éléments `<tr>` avec des `<th>`) sera définie dans un sous-programme XSL nommé `headerTableau` et les deux premières colonnes de chaque ligne (parties jour et nombre d'aliments) seront générées via un sous-programme XSL nommé `nbAliment`. On viendra inclure le fichier stdWeb2.xsl et on utilisera obligatoirement ses sous-programmes à chaque fois que c'est possible.

9. Discussion

Essayez de répondre à la question suivante :

> *Serait-il intéressant d'implémenter une option*
>
> `Edition/Rechercher selon une expression XPATH`
>
> *dans les menus des logiciels GEDIT et/ou GEANY ou équivalent pour aider à l'édition de fichiers XML ?*

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. Votre réponse devra faire 10 lignes au minimum, sans limite de maximum. On utilisera au moins 3 mots de 4 syllabes ou plus pour transmettre ≪ un contenu rédactionnel fort ≫.

ANNEXE : Schéma repas.xsd

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

  <xs:element name="desRepas">
    <xs:complexType><xs:sequence>
        <xs:element maxOccurs="unbounded" ref="unRepas" />
    </xs:sequence></xs:complexType>
  </xs:element>

  <xs:element name="unRepas">
    <xs:complexType><xs:sequence>
        <xs:element ref="jour" />
        <xs:element ref="repas" />
        <xs:element ref="aliments" />
    </xs:sequence></xs:complexType>
  </xs:element>

  <xs:element name="jour">
    <xs:complexType>
      <xs:attribute name="date" type="xs:string" />
    </xs:complexType>
  </xs:element>

  <xs:element name="repas">
    <xs:complexType>
      <xs:attribute name="heure"  type="xs:integer" />
      <xs:attribute name="min"    type="xs:integer" />
      <xs:attribute name="moment" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="aliments">
    <xs:complexType><xs:sequence>
        <xs:element maxOccurs="unbounded" ref="aliment" />
    </xs:sequence></xs:complexType>
  </xs:element>

  <xs:element name="aliment">
    <xs:complexType>
      <xs:attribute name="multiple" type="xs:string" />
      <xs:attribute name="nom"      type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

</xs:schema>
```

ESQUISSE DE SOLUTION

1. Ecriture de DTD externe

Dans l'exemple fourni, l'élément `<matin>` n'est pas toujours présent et il y a toujours un seul aliment le soir. L'attribut heure n'est pas écrit pareil pour le matin et les autres éléments. La grammaire qui "colle" le plus à l'exemple est donc :

```
<!ELEMENT jours (jour)+            >
<!ELEMENT jour  (matin?,midi,soir) >  <!-- au vu des exemples  -->
<!ELEMENT matin (aliment)+         >
<!ELEMENT midi  (aliment)+         >
<!ELEMENT soir  (aliment)          >  <!-- au vu des exemples  -->
<!ELEMENT aliment (#PCDATA)>

<!ATTLIST jour    numero CDATA   #IMPLIED  >
<!ATTLIST matin   heure  CDATA   #REQUIRED >  <!-- attention ! -->
<!ATTLIST midi    heure  NMTOKEN #IMPLIED  >
<!ATTLIST soir    heure  NMTOKEN #IMPLIED  >
<!ATTLIST aliment refid  NMTOKEN #IMPLIED  >
```

Une grammaire plus générale utiliserait moins de subtilités.

2. Un peu de XPATH *via* xmllint

1. `ls //aliment[@refid="A170"]`
2. `xpath count(//jour)`
3. `xpath count(//jour[@numero>80])`
4. `xpath count(//jour[@numero=26]//aliment)`
5. `ls //jour[@numero=253]//soir/aliment[last()]`
6. `ls //jour[.//aliment="pain" and .//aliment="chocolat"]/@numero`

3. D'autres repas

Voici une solution possible :

```
<desRepas>

  <unRepas>
     <jour />
     <repas moment="pdj" />
     <aliments>
       <aliment nom="lait" />
       <aliment nom="miel" />
     </aliments>
  </unRepas>

  <unRepas>
     <jour date="19-05-04"/>
     <repas moment="midi" heure="13" min="30" />
     <aliments>
       <aliment nom="viande" />
       <aliment nom="fruit"  multiple="oui" />
     </aliments>
  </unRepas>

</desRepas>
```

4. TLR : tous les repas

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE va [
  <!ENTITY repas01 SYSTEM "repas01.xml"   >
  <!ENTITY repas02 SYSTEM "repas02.xml" >
]>

<tousLesRepas>
   &repas01;
   &repas02;
</tousLesRepas>
```

5. Expressions régulières pour les heures de repas

Juste un entier de 00 à 24 :

```
^[0-1]?[0-9]$|^2[0-4]$
```

Un entier correspondant à une heure avec un h entouré éventuellement d'espaces et suivi éventuellement d'un entier correspondant à des minutes :

```
^([0-1]?[1-9]|^2[0-4]) ?h ?[0-5]?[0-9]?$
```

Dans un schéma XSD, pour utiliser ces expressions régulières, il suffirait d'écrire :

```
<xs:simpleType name="heureRepas">
  <xs:restriction base="xs:string">
     <xs:pattern value="XXXX" />
  </xs:restriction>
</xs:simpleType>
```

6. Transformations XSL (1)

Comme XSL autorise à mettre dans des expressions select des opérations comme des additions, on peut écrire :

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

<xsl:template match="/">Dans le fichier des repas, il y a :<xsl:text>&#10;</xsl:text>
 - <xsl:value-of select="count(//jour)" /> jours ;
 - <xsl:value-of select="count(//matin)+count(//midi)+count(//soir)" /> repas ;
 - <xsl:value-of select="count(//aliment)" /> aliments, distincts ou non.
</xsl:template>

</xsl:stylesheet>
```

Une autre solution consisterait à utiliser des variables :

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

<xsl:variable name="nbMatin" select="count(//matin)" />
<xsl:variable name="nbMidi"  select="count(//midi)" />
<xsl:variable name="nbSoir"  select="count(//soir)" />

<xsl:template match="/">Dans le fichier des repas, il y a :<xsl:text>&#10;</xsl:text>
[...]
 - <xsl:value-of select="$nbMatin+$nbMidi+$nbSoir" /> repas ;
[...]
```

Enfin, `<xsl:value-of select="count(//matin | //midi | //soir)"` est OK aussi.

7. Transformations XSL (2)

Pour compter le nombre d'aliments distincts, il faudrait mettre comme dernier count une expression comme `count(//aliment[not(./text()=following::aliment/text())])` qui ne retient pas les valeurs-textes des éléments égaux aux aliments déjà rencontrés.

Toutefois, compte-tenu des notations en raccourci de XSL, on peut se contenter de `count(//aliment[not(.=following::aliment)])`. Pour que xmlstarlet affiche des contenus, il faut écrire : `xmlstarlet sel -t -v "//aliment" repas01.xml | sort -u`.

8. Transformations XSL (3)

```
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version="1.0">
<xsl:import href="stdWeb2.xsl" />
<xsl:output method="xml" encoding="UTF-8" />

<!-- ********************************************************************** -->

<xsl:template name="nbAliment">
  <td>
       <xsl:choose>
          <xsl:when test="./@numero"><xsl:value-of select="./@numero" /></xsl:when>
          <xsl:otherwise>???</xsl:otherwise>
       </xsl:choose>
  </td>
  <td align="right"><xsl:value-of select="count(.//aliment)" /></td>
</xsl:template>

<!-- ********************************************************************** -->

<xsl:template name="headerTableau">
<tr>
  <th rowspan="2"><p>Jour</p></th>
  <th rowspan="2"><p>Nombre<br />d'Aliments</p></th>
  <th colspan="3"><p>Repas</p></th>
</tr>
<tr>
  <th>Matin</th>
  <th>Midi</th>
  <th>Soir</th>
</tr>

</xsl:template>

<!-- ********************************************************************** -->

<xsl:template match="/">
```

```
<xsl:call-template name="debutPage">
  <xsl:with-param name="leTitre">Aliments par repas</xsl:with-param>
</xsl:call-template>
<xsl:call-template name="debutSection" />

<blockquote>
<h1>Nombre d'aliments par repas</h1>
<blockquote>

<xsl:call-template name="debutTableau" />
<xsl:call-template name="headerTableau" />
<xsl:for-each select="//jour">
  <tr>
     <xsl:call-template name="nbAliment" />
     <td><xsl:value-of select="count(./matin/aliment)" /></td>
     <td><xsl:value-of select="count(./midi/aliment)"  /></td>
     <td><xsl:value-of select="count(./soir/aliment)"  /></td>
  </tr>
</xsl:for-each>
<xsl:call-template name="finTableau" />

</blockquote>
</blockquote>
<xsl:call-template name="debutSection" />
<xsl:call-template name="finPage" />
</xsl:template>

</xsl:stylesheet>
```