G. Hunault
Licence 2 Informatique

Angers, juin 2020

Technologie XML

1. Ecriture de DTD externe

On veut modéliser des sessions d'examens en XML. Voici un exemple d'un fichier XML possible :

```
<Examens>
 <examens an="2019">
     <session1>
        <examen niveau="L1" nature="anglais" date="22 janvier" />
        <examen niveau="L2" nature="xml"   date="12 mars"    />
     </session1>
     <session2>
        <examen niveau="L3" nature="maths" date="14 mai"
         remarque="Groupe 2 seulement" />
        <examen niveau="M1" nature="xml"   date="15 mai"    />
     </session2>
 </examens>
 <examens an="2020">
     <session1>
        <examen niveau="L1" nature="maths" date="20 janvier" />
        <examen niveau="L2" nature="xml"   date="09 mars"    />
     </session1>
     <session2>
        <examen niveau="L3" nature="maths" date=""    />
        <examen niveau="M1" nature="xml"   date=""    />
     </session2>
 </examens>
</Examens>
```

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 sessions d'examens 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. donner toutes les natures (matières) des examens de 2017, peu importe la session

2. combien y a-t-il d'éléments `<examen>` dans le fichier ?

3. combien y a-t-il d'élements `<examen>` en session 2 pour 2019 ?

4. quelle est la date de l'examen en xml pour la session 1 de 2018 ?

5. combien y a-t-il d'examens avec une remarque vide dans le fichier ?

6. quelle est la date et la nature (matière) de tous les examens comportant un attribut remarque avec une seule commande ?

3. Compléments de structuration

- On décide d'ajouter une information *documents autorisés* à un examen. Le contenu de cette information est "oui" ou "non".

  Quel choix feriez-vous pour cette information, élément ou attribut ?

  Justifiez votre choix.

- On décide d'ajouter aussi une information *durée de l'examen.*

  Quel choix feriez-vous pour cette information, élément ou attribut ? Pourquoi ?

  Comment coder le contenu de l'information, sachant que certains examens durent une heure, d'autres une heure trente et enfin d'autres encore plus longtemps ?

4. Une autre structuration pour les examens

Donner le contenu XML d'un fichier examen03.xml qui suit le schéma examens.xsd fourni en annexe avec les informations suivantes :

- l'examen de session 1 dont l'id est L1XML a pour date 06/12/2015
- pour l'examen dont l'id est M1IA, la date est 28/01/2014 et tous les documents sont autorisés ; il s'agit d'un examen en session 1.
- l'examen dont l'id est L3DCRA (session 2) a une date non encore déterminée.

5. Transformations XSL (1)

On voudrait produire un fichier qui résume les examens par cycle. Donner le contenu d'un fichier XSL qui effectue une transformation d'un fichier structuré comme exam01.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.

```
Fichier des examens :

 - 18 examens sur 5 ans

 - 16 examens en Licence dont

     . 8 en L1
     . 6 en L2
     . 2 en L3
```

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 examens d'un fichier structuré comme à la question 1.

```
<h1>Examens, choisissez l'année :</h1>

  <form action="exams.php">
     <div id="an"><select>

       <option>2020</option>
       <option>2019</option>
       <option>2018</option>
       <option>2017</option>

     </select></div>
     <p><input type="submit"/></p>
  </form>
```

7. Discussion

Essayez de répondre à la question suivante :

> *Est-ce que XML réalise un bon compromis entre lisibilité, sémantique et rigueur ?*

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. On utilisera au moins 3 mots de 4 syllabes ou plus pour transmettre « un contenu rédactionnel fort ».

ANNEXE : Schéma examens.xsd

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

  <xsd:element name="examens">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" ref="examen"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="examen">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="session"/>
        <xsd:element ref="date"/>
        <xsd:element minOccurs="0" ref="remarque"/>
      </xsd:sequence>
      <xsd:attribute name="id" use="required" type="xsd:NCName"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="session">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension   base="xsd:integer">
          <xsd:attribute name="documents" type="xsd:NCName"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="date"     type="xsd:string"/>
  <xsd:element name="remarque" type="xsd:string"/>

</xsd:schema>
```

ESQUISSE DE SOLUTION

1. Ecriture de DTD externe

Dans l'exemple fourni, l'élément racine Examens ne comporte que des éléments examens. On peut supposer qu'il y en a toujours au moins 1, donc la cardinalité sera +.

Chaque élément examens comporte exactement deux éléments, nommés session1 et session2. On peut supposer que c'est toujours le cas.

Les éléments session1 et session2 ne comportent que des éléments examen. Il doit certainement toujours y en avoir au moins un, donc la cardinalité sera +.

Tous les éléments examen sont vides et ne comportent que des attributs. Seul l'attribut remarque semble être facultatif, on le mettra en IMPLIED alors que tous les autres attributs seront déclarés REQUIRED.

Pour l'exemple considéré, les attributs niveau et nature correspondent à un seul mot. On peut donc les typer en NMTOKEN.

Par contre date et remarque contiennent plus de texte. Leur type sera CDATA.

Une grammaire possible est donc :

```
<!-- GRAMMAIRE DTD DU FICHIER examen01.xml -->

<!-- éléments par ordre d'inclusion -->

     <!ELEMENT Examens  (examens)+            >
     <!ELEMENT examens  (session1,session2)   >
     <!ELEMENT session1 (examen)+             >
     <!ELEMENT session2 (examen)+             >
     <!ELEMENT examen   EMPTY                 >

<!-- attributs par ordre alphabétique -->

     <!ATTLIST examens an       CDATA     #REQUIRED  >
     <!ATTLIST examen  date     CDATA     #REQUIRED  >
     <!ATTLIST examen  nature   NMTOKEN   #REQUIRED  >
     <!ATTLIST examen  niveau   NMTOKEN   #REQUIRED  >
     <!ATTLIST examen  remarque CDATA     #IMPLIED   >
```

2. Un peu de XPATH *via* xmllint

Voici les commandes xmllint demandées :

1. `ls //examens[@an="2017"]//examen/@nature`

2. `xpath count(//examen)`

3. `xpath count(//examens[@an="2019"]/session2/examen)`

4. `ls //examens[@an="2018"]/session1/examen[@nature="xml"]/@date`

5. `xpath count(//examen[@remarque=""])`

6. on peut trouver séparément les dates et les matières par

   ```
   ls //examen[@remarque]/@date
   ls //examen[@remarque]/@nature
   ```

   pour les avoir ensemble comme demandé :

   ```
   ls //examen[@remarque]/@date | //examen[@remarque]/@nature
   ```

3. Compléments de structuration

Il est possible de mettre les deux informations complémentaires soit sous forme d'élément soit sous forme d'attribut. Toutefois, comme toutes les autres informations sont codées comme des attributs, le plus cohérent est de mettre les nouvelles informations dans des attributs.

La durée peut être donnée en minutes ou en heure et minutes, dans un format texte comme duree="1h", duree="1h30" etc. C'est le plus lisible même si ce n'est pas le plus simple à traiter. Une solution plus utilisable est de séparer les heures et les minutes, par exemple avec les attributs dureeHeure="1" dureeMinute="30".

4. Une autre structuration pour les examens

Le fichier demandé ressemble certainement au texte de la page suivante.

```
<examens>
   <examen id="L1XML">
     <session>1</session>
     <date>06/12/2015</date>
     <remarque>tous les groupes</remarque>
   </examen>
   <examen id="M1IA">
     <session documents="oui">1</session>
     <date>28/01/2014</date>
   </examen>
   <examen id="L3DCRA">
     <session documents="oui">2</session>
     <date></date>
   </examen>
</examens>
```

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="nbe"><xsl:value-of select="count(//examen)"  /></xsl:variable>
<xsl:variable name="nba"><xsl:value-of select="count(//@an)"     /></xsl:variable>
<xsl:variable name="nl1"><xsl:value-of select="count(//examen[@niveau='L1'])" />
</xsl:variable>
<xsl:variable name="nl2"><xsl:value-of select="count(//examen[@niveau='L2'])" />
</xsl:variable>
<xsl:variable name="nl3"><xsl:value-of select="count(//examen[@niveau='L3'])" />
</xsl:variable>

<xsl:template match="/">Fichier des examens :

 - <xsl:value-of select="$nbe" /> examens sur <xsl:value-of select="$nba" /> ans
 <xsl:text>&#10;</xsl:text>
 - <xsl:value-of select="$nl1 + $nl2 + $nl3" /> examens en Licence dont
 <xsl:text>&#10;</xsl:text>
     . <xsl:value-of select="$nl1"  /> en L1
     . <xsl:value-of select="$nl2"  /> en L2
     . <xsl:value-of select="$nl3"  /> en L3
</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" />

<xsl:template match="/">
<h1>Examens, choisissez l'ann&#233;e :</h1>
<xsl:text>&#10;  </xsl:text>
<form action="exams.php">
  <xsl:text>&#10;     </xsl:text>
  <div id="an"><select><xsl:text>&#10;</xsl:text>
  <xsl:apply-templates select="//@an">
     <xsl:sort select="." order="descending"/>
   </xsl:apply-templates>
  <xsl:text>&#10;     </xsl:text>
  </select></div>
  <xsl:text>&#10;     </xsl:text>
  <p> <input type="submit" /> </p>
<xsl:text>&#10;  </xsl:text>
</form>
</xsl:template>

<xsl:template match="@an">
  <xsl:text>&#10;</xsl:text>
  <xsl:variable name="annee"><xsl:value-of select="." /></xsl:variable>
  <xsl:text>       </xsl:text><option>
  <xsl:value-of select="$annee" />
  </option><xsl:text>&#10;</xsl:text>
</xsl:template>

</xsl:stylesheet>
```