G. Hunault
Licence 2 Informatique

Angers, mars 2020

Technologie XML

1. Question de cours

Si vous êtes dispensé(e) d'assiduité et autorisé(e) à ne pas suivre les cours, veuillez l'indiquer.

Sinon, indiquez les quatre termes importants cités dans le dernier cours en amphi.

2. Ecriture de DTD externe

On admet pour ce qui suit qu'un document nommé **trombinoscopes**, désigné par un élément `<trombis>` contient au moins un trombinoscope simplifié (élément `<trombi>`). Un trombinoscope simplifié contient une référence, des noms, des prénoms et des photos. Voici un exemple d'un fichier XML qui contient un tel trombinoscope simplifié :

```
<?xml version="1.0" encoding="ISO-8859-15" ?>
<trombis>
  <trombi>
    <référence>
      Groupe de travail 28-RZ
    </référence>
    <nom>
      MARTIN
    </nom>
    <prénom> Jean </prénom>
    <photo lieu="../media/" id="P24021.JPG" />
    <nom> ZOLA </nom>
    <prénom>
      Violette
    </prénom>
    <photo lieu="../media/" id="P24022.JPG" />
  </trombi>
</trombis>
```

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.

3. 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 trombinoscopes simplifiés 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. combien y a-t-il d'éléments `<trombi>` dans le fichier ?

2. combien y a-t-il d'élements `<nom>` dans le troisième trombinoscope ?

3. quel est l'id de la photo pour la personne de nom "DURAND" et de prénom "Gisèle" ?

4. combien y a-t-il de trombinoscopes dont la référence contient le mot "Groupe" ?

4. Mauvaise structuration

Essayer d'expliquer pourquoi la structuration hiérarchique fournie via le fichier XML de la question 2 n'est sans doute pas correcte conceptuellement et proposer une meilleure structuration.

5. D'autres structurations pour les trombinoscopes

Donner le contenu XML d'un fichier trombis03.xml qui suit le schéma trombis.xsd fourni en annexe avec les informations suivantes :

- il y a un seul trombinoscope simplifié dans le document, dont le numéro est "G203" ;

- la référence de ce trombinoscope simplifié est "Ensemble vocal 2020" ;

- il y a une seule personne actuellement dans ce trombinoscope ; son id est "P35", son nom est "DUPOND", son prénom "Jeanne" et sa photo est "P2020-19.JPG".

On notera ici, et c'est volontaire, que le schéma XSD ne contient aucune information sur l'*id* d'une personne.

Indiquer comment vous arrivez à inclure quand même l'id dans le document. Si vous n'y arrivez pas, n'incluez pas l'id.

6. Transformations XSL (1)

On voudrait produire un fichier qui résume les trombinoscopes avec affichage des noms par ordre décroissant. Donner le contenu d'un fichier XSL qui effectue une transformation du fichier trombi01.xml de la question 2 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. On remarquera que les espaces et les sauts de ligne ne sont pas normalisés dans les éléments <nom>.

```
Dans le fichier des trombinoscopes, il y a 2 noms en tout.
Les voici par ordre alphabétique décroissant :
-- début de liste
    Nom : ZOLA
    Nom : MARTIN
-- fin de liste
```

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)

Donner le code d'une transformation XSL qui produit un fichier au format CSV2 comme suit pour tous les trombinoscopes simplifiés d'un fichier structuré comme à la question 2.

```
RefTrombi    ; Nom ; Prénom ; Photo
Groupe de travail 28-RZ ; MARTIN ; Jean ; P24021.JPG ;
Groupe de travail 28-RZ ; ZOLA ; Violette ; P24022.JPG ;
```

8. Discussion

Essayez de répondre à la question suivante :

> *A-t-on vraiment besoin de sérialisation pour 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.

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 trombi.xsd

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="lesTrombis">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" ref="unTrombi"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="unTrombi">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="personnes"/>
      </xsd:sequence>
      <xsd:attribute name="num" use="required" type="xsd:NCName"/>
      <xsd:attribute name="ref" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="personnes">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" ref="personne"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="personne">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="nom"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="nom">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute name="photo" use="required" type="xsd:NCName"/>
          <xsd:attribute name="prénom" use="required" type="xsd:NCName"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

ESQUISSE DE SOLUTION

1. Question de cours

Les quatre termes sont :

- SimpleXML ;
- XSL 2.0 ;
- XAML ;
- Web sémantique.

2. Ecriture de DTD externe

Dans l'exemple fourni, à part l'élément racine et l'élément trombi, tous les éléments et tous les attributs ont rigoureusement la même cardinalité à savoir la valeur 1 exactement. Une grammaire possible est donc :

```
<!ELEMENT trombis     (trombi)+                          >
<!ELEMENT trombi      (référence,(nom,photo,prénom)+)    >
<!-- autre solution : (référence,(nom|photo|prénom)+)  -->
<!ELEMENT référence   (#PCDATA)                          >
<!ELEMENT nom         (#PCDATA)                          >
<!ELEMENT prénom      (#PCDATA)                          >
<!ELEMENT photo        EMPTY                             >

<!ATTLIST photo id   NMTOKEN #REQUIRED >
<!ATTLIST photo lieu CDATA   #REQUIRED >
```

Remarque : on a considéré ici que les retours à la ligne n'étaient pas importants. Ainsi

```
<nom>
 ZOLA
</nom>
```

et

```
<nom> ZOLA  </nom>
```

ont été considérés comme équivalents.

3. Un peu de XPATH *via* xmllint

Voici les commandes xmllint demandées :

1. `xpath count(//trombi)`
2. `xpath count(//trombi[3]//nom)`
3. `ls //trombi[.//nom="DURAND" and .//prénom="Gisèle"]//photo/@id`
4. `xpath count(//référence[contains(.,"Groupe")])`

4. Mauvaise structuration

La structuration n'est sans doute pas correcte puisque les noms, prénoms et photos ne sont pas associés au sein d'un même élément. Il manque vraisemblablement un élément `<personne>` pour regrouper `<nom>`, `<prenom>` et `<photo>`.

Il serait donc logique et intéressant d'avoir aussi un élément `<personnes>` pour regrouper toutes les personnes.

Voici un exemple de structuration plus correcte :

```
<?xml version="1.0" encoding="ISO-8859-15" ?>
<trombis>

  <trombi>

    <référence> Groupe de travail 28-RZ </référence>

    <personnes>

       <personne>
          <nom> MARTIN </nom>
          <prénom> Jean </prénom>
          <photo lieu="../media/" id="P24021.JPG" />
       </personne>

       <personne>
          <nom> ZOLA </nom>
          <prénom> Violette </prénom>
          <photo lieu="../media/" id="P24022.JPG" />
       </personne>
    <personnes>
  </trombi>
</trombis>
```

5. D'autres structurations pour les trombinoscopes

Le fichier demandé ressemble certainement au texte qui suit :

```
<?xml version="1.0" encoding="ISO-8859-15" ?>
<lesTrombis>
  <unTrombi num="G203" ref="Ensemble vocal 2020">
    <personnes>
      <personne>
        <nom prénom="Jeanne" photo="P2020-19.JPG">
             DUPOND
<![CDATA[
        <id>P35</id>
]]>
        </nom>
      </personne>
    </personnes>
  </unTrombi>
</lesTrombis>
```

Puisque l'élément nom est défini comme une extension du type standard string, on peut utiliser une section CDATA qui sera ignorée lors de la validation.

6. 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:template match="/">Dans le fichier des trombinoscopes,
   il y a <xsl:value-of select="count(//nom)" /> noms en tout.
   Les voici par ordre alphabétique décroissant :
   <xsl:text>-- début de liste &#10;</xsl:text>

   <xsl:apply-templates select="//nom">
     <xsl:sort select="." order="descending"/>
   </xsl:apply-templates>
   <xsl:text> &#10;</xsl:text>

   <xsl:text>-- fin de liste</xsl:text>
</xsl:template>

<xsl:template match="nom">
   Nom : <xsl:value-of select="normalize-space(.)" />
   <xsl:if test="not( position()=last())" ><xsl:text>&#10;</xsl:text></xsl:if>
</xsl:template>

</xsl:stylesheet>
```

7. Transformations XSL (2)

Il y a une difficulté majeure : les noms, prénoms et photos sont à la suite les uns des autres directement dans l'élément `<trombi>`.

Pour trouver le prénom associé au nom, il faut donc utiliser un indice de position : puisqu'il y a toujours le nom suivi du prénom, alors prenom[i] correspond à nom[i]. On utilise ici une variable nommé pos qui permet de garder cet indice de position.

La même remarque s'applique à l'élément `<photo>` et à son attribut id, mais attention : il faut utiliser l'indice de position sur photo, pas sur id, au cas où d'autres éléments auraient aussi des id.

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

<xsl:template match="/">
<xsl:text>RefTrombi    ; Nom ; Prénom ; Photo  &#10;</xsl:text>
<xsl:apply-templates select="//trombi" />
</xsl:template>

<xsl:template match="trombi">
<xsl:for-each select="./nom" >

   <xsl:variable name="pos" select="position()" /> <!-- important -->

   <xsl:value-of select="normalize-space(../référence)" /> ;
   <xsl:value-of select="normalize-space(.)" /> ;
   <xsl:value-of select="normalize-space(..//prénom[$pos])" /> ;
   <xsl:value-of select="normalize-space(../photo[$pos]/@id)" /> ;

</xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```