G. Hunault
Licence 2 Informatique

Angers, mars 2018

Technologie XML

1. Structuration en XHTML

Ecrire le fragment de texte XHTML 1.0 valide au sens de la grammaire *Strict* qui correspond aux indications suivantes. L'élément X1 a pour attributs A2 (de valeur V3) et A4 (de valeur V5). Il contient l'élément X6 d'attribut A7 dont la valeur est V8. X6 contient le texte T9.

Pour mémoire, les noms des éléments et des attributs doivent être écrits en minuscules.

Si on remplace X1 par div, A2 par class, V3 par examen, A4 par id, V5 par l2info, X6 par p, A7 par class, V8 par texteAligne et T9 par Simple, qu'obtient-t-on comme code XHTML ?

2. Compréhension du vocabulaire

On veut écrire un fichier A.xml dont l'élément racine est B. Ce fichier contient uniquement les deux entités caractères nommées C et D, correspondant respectivement aux fichiers C.xml et D.xml, définies via une DTD interne.

Donner toutes les lignes de texte **XML** correspondant à ce fichier. On pourra omettre la ligne 1 correspondant à `<?xml version="1.0" ?>`.

3. Ecriture de DTD externe

Nous admettrons pour ce qui suit qu'un fichier de configuration pour l'exécution d'un exercice d'algorithmique comporte obligatoirement un attribut qui donne le nombre d'exécutions à tester puis pour chaque exécution un élément qui contient toujours les variables d'entrées avec leur valeur et les variables de sortie avec leur valeur.

Voici un exemple de fichier XML comportant deux exécutions à tester, nommé minmax.xml :

```
<?xml version="1.0" ?>
<exercice executions="2" >

  <execution numero="1">
     <entrees>
        <variable nom="valA" valeur="2" />
        <variable nom="valB" valeur="6" />
     </entrees>
     <sorties>
        <variable nom="petit" valeur="2" />
        <variable nom="grand" valeur="6" />
     </sorties>
  </execution>

  <execution numero="2">
     <entrees>
        <variable nom="valA" valeur="8" />
        <variable nom="valB" valeur="5" />
     </entrees>
     <sorties>
        <variable nom="petit" valeur="5" />
        <variable nom="grand" valeur="8" />
     </sorties>
  </execution>

</exercice>
```

Donner pour ce fichier minmax.xml une grammaire DTD « minimale et raisonnable » qui permet de décrire les tests d'exécution d'algorithmes.

4. Un peu de XPATH

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 configuration pour l'exécution d'un exercice d'algorithmique.

Attention : il ne s'agit pas forcément du fichier minmax.xml précédent mais d'un fichier structuré de la même façon.

- combien y a-t-il d'exécutions selon l'attribut executions ?

- combien y a-t-il d'exécutions si on compte le nombre d'éléments nommés execution ?

- quel est le nom de toutes les variables utilisées, aussi bien en entrée qu'en sortie ?

- quelle est la valeur de la variable de sortie nbFichiers pour l'exécution dont le numéro est 5 ?

5. Transformations XSL

Donner le contenu d'un fichier XSL qui effectue une transformation du fichier minmax.xml de la question 2 et qui produit le fichier XML suivant.

On ne tiendra pas compte précisément du formatage, donc ce n'est pas grave s'il manque des espaces ou des sauts de ligne ou s'il y en a en trop.

```
<miniconf>
  <ex>sorties execution 1 : petit=2 ; grand=6 </ex>
  <ex>sorties execution 2 : petit=5 ; grand=8 </ex>
</miniconf>
```

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

6. Discussion

Essayez de répondre à la question suivante :

> *Peut-on considérer qu'en 2018 les outils et méthodes de la pérennisation numérique des données et des métadonnées sont standardisés, au moins en partie, via 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 ».

ESQUISSE DE SOLUTION

1. Structuration en XHTML

Voici la structuration formelle du document. Attention, le nom des éléments et des attributs doit être en minuscule (mais pas leur contenu).

```
<x1 a2="V3" a4="V5">
  <x6 a7="V8">
    T9
  </x6>
</x1>
```

On obtient alors, en XHTML :

```
<div class="examen" id="l2info">
  <p class="texteAligne">
     Simple
  </p>
</div>
```

2. Compréhension du vocabulaire

Le document `A.xml` demandé est :

```
<!DOCTYPE B [
  <!ENTITY C SYSTEM "C.xml" >
  <!ENTITY D SYSTEM "D.xml" >
]>

<B>
  &C;
  &D;
</B>
```

3. Ecriture de DTD externe

Une grammaire DTD, concise, cohérente et minimale pour l'exemple fourni doit ressembler à :

```
<!ELEMENT  exercice   (execution)+                >
<!ATTLIST  exercice   executions  CDATA  "REQUIRED"  >

<!ELEMENT  execution  (entrees,sorties)           >
<!ATTLIST  execution  numero     CDATA  "REQUIRED"  >

<!ELEMENT  entrees    (variable)+                 >
<!ELEMENT  sorties    (variable)+                 >

<!ELEMENT  variable   EMPTY                       >
<!ATTLIST  variable   nom    CDATA  "REQUIRED"      >
<!ATTLIST  variable   valeur CDATA  "REQUIRED"      >
```

4. Un peu de XPATH

Voici les expressions XPATH demandées.

```
ls /exercice/@executions

xpath count(//execution)

ls //variable/@nom

ls //execution[@numero="5"]/sorties/variable[@nom="nbFichiers"]/@valeur
```

5. Transformations XSL

Voici une transformation possible :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" />

<xsl:template match="/">
<miniconf>&#10;
<xsl:apply-templates select="//execution" />
<xsl:text>
</xsl:text>
</miniconf>
</xsl:template>

<xsl:template match="execution">
<xsl:text>
  </xsl:text>
  <ex>
    <xsl:text> execution </xsl:text>
    <xsl:value-of select="./@numero" />
    <xsl:text> :</xsl:text>
    <xsl:apply-templates select=".//sorties/variable" />
  <xsl:text> </xsl:text>
  </ex>&#10;
</xsl:template>

<xsl:template match="variable">
  <xsl:text> </xsl:text>
  <xsl:value-of select="./@nom" />
  <xsl:text>=</xsl:text>
  <xsl:value-of select="./@valeur" />
  <xsl:if test="position()!=last()"> <xsl:text> ;</xsl:text></xsl:if>
</xsl:template>

</xsl:stylesheet>
```