Module XML

TD5 : Feuilles de style (correction)

Objectifs :

- Savoir définir des documents XSL
- Utiliser des feuilles de style pour transformer un document XML

Exercice (1) – Eléments template :

Extraire un sous-ensemble d'enregistrements répondant à certaines exigences. Le fichier source à traiter est similaire au fichier suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="storage.xsl"?>
<produits>
  <produit zone="Z23-888" category="outils">
    <description>Marteau panne fendue</description>
    <poids units="g">850</poids>
    <prix>12.99</prix>
  </produit>
  <produit zone="X853-123" category="livres">
    <description>La plomberie pour les débutants</description>
    <ISBN>0-123-456-789</ISBN>
    <prix>10.95</prix>
  </produit>
  <produit zone="X853-S14-5689" category="outils">
    <description>Clé à molette</description>
    <poids units="g">330</poids>
    <prix>5.25</prix>
  </produit>
</produits>
```

L'exigence est d'obtenir un autre fichier de produits, possédant la même structure mais en ne sélectionnant que les produits dont le prix est supérieur à 10 Euros, et en éliminant les éléments prix à la sortie.

Ecrivez une feuille XSL pour générer un fichier résultat.

Pour cela, écrivez trois règles modèles (templates) :

- La première règle traite tous les éléments (*). Son objectif est de recopier le nœud élément de l'arbre source vers l'arbre résultat en recopiant les attributs (utiliser les instructions `<xsl:copy>` et/ou `<xsl:copy-of>`) et d'appeler le traitement des nœuds fils à l'aide des règles modèles appropriées.
- Les deux autres règles sont vides et identifient des éléments sans produire de sortie :
 - Une règle a pour objectif de supprimer les éléments prix de l'arbre résultat.

- L'autre règle modèle a pour objectif de filtrer les produits pour ne conserver que certains éléments produits dans l'arbre résultat.

Comment le document est-il traité ?

La sortie de la feuille de style doit ressembler à :

```
<?xml version="1.0" encoding="UTF-8"?>
<produits>
  <produit zone="Z23-888" category="outils">
    <description>Marteau panne fendue</description>
    <poids units="g">850</poids>
  </produit>
  <produit zone="X853-123" category="livres">
    <description>La plomberie pour les débutants</description>
    <ISBN>0-123-456-789</ISBN>
  </produit>
</produits>
```

Solution :

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

<!--premiere regle : traiter les elements et les fils de ces elements-->
<xsl:template match="*">
  <xsl:copy>

    <xsl:copy-of select="@*" />
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- seconde règle : filtrer les produits pour lequel le prix <10 -->
<xsl:template match="produit[prix &lt; 10]"/>

<!--troisième règle : supprimer les éléments prix-->
<xsl:template match="prix"/>

</xsl:stylesheet>
```

Explications :

- Insister sur la spécificité des règles et leur ordre application (la dernière est beaucoup plus spécifique, s'applique donc en priorité et masque la première règle)
- Parler des règles par défaut
 - Eléments → apply templates (pour exécuter les règles sur les éléments fils directs)
 - Texte → affichage du contenu
- Pensez à utiliser la copie des attributs avec <xsl:copy-of .../>

Que se passe-t-il si on applique la feuille de style suivante au document ?

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

Si aucune règle ne s'applique, ce sont les règles par défaut qui s'appliquent (affichage des textes et recopie des PCDATA contenus dans les balises)

(vu en cours)

Résultat :

```
<?xml version="1.0" encoding="utf-8"?>
```

```
Marteau panne fendue 850 12.99 La plomberie pour les débutants 0-123-456-789 10.95 Clé à molette 330 5.25
```

Exercice 2 : Application récursive des Modèles

Lorsque `<xsl:apply-templates>` est utilisé pour traiter un ensemble de nœuds, il n'est pas obligatoire qu'il y ait exactement une règle modèle correspondant à chaque nœud.

Insister sur les règles par défaut (bis)

Créer une feuille de style sortant toutes les sections d'un chapitre, précédé par une table de matières. Les entrées de la table constituent des liens hypertextes vers les sections pertinentes, les identificateurs des liens étant construits à l'aide de la fonction Xpath `generate-id()`.

N.B. : `generate-id()` (génère une chaîne identifiant de façon unique un nœud) est une fonction XSLT, ne pouvant donc être utilisée que dans des expressions Xpath contenues dans une feuille de style XSLT (à l'instar de `function-available()` (tester si une fonction particulière existe) et `key(arg1, arg2)` (identifier les nœuds possédant une valeur donnée pour une clé particulière).

Le document XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="docbook.xsl"?>
<book>
  <part>
    <chapter>
      <sect1>
        <title>Section1 Sujet</title>
        <para>Un bon de commande est émis par une organisation
        identifiée par un nom, une adresse, un numéro de téléphone et éventuellement un numéro de
        télécopie. Il s'adresse à une autre organisation définie de la même manière. Il comporte une
        date, un montant total ainsi que une ou plusieurs lignes de commande. Chaque ligne comporte
        une référence de produit, un intitulé, une quantité, un prix unitaire..</para>
      </sect1>
      <sect1>
        <title>Section2 JustDolt</title>
        <para>
          Rédiger la définition (DTD) d'un document de
          type Bon de commande
        </para>
      </sect1>
    </chapter>
  </part>
</book>
```

```

                </sect1>
            </chapter>
        </part>
    </book>

```

Le résultat (vu dans un navigateur)

Table des matières

[Section1 Sujet](#)

[Section2 JustDoIt](#)

Section1 Sujet

Un bon de commande est émis par une organisation identifiée par un nom, une adresse, un numéro de téléphone et éventuellement un numéro de télécopie. Il s'adresse à une autre organisation définie de la même manière. Il comporte une date, un montant total ainsi que une ou plusieurs lignes de commande. Chaque ligne comporte une référence de produit, un intitulé, une quantité, un prix unitaire..

Section2 JustDoIt

Rédiger la définition (DTD) d'un document de type Bon de commande

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output encoding="ISO-8859-1"/>
    <xsl:template match="chapter">
        <h2>Table des matières</h2>
        <xsl:apply-templates select="sect1" mode="toc"/>
        <xsl:apply-templates select="sect1" mode="body"/>
    </xsl:template>
    <xsl:template match="sect1" mode="toc">
        <p>
            <a href="#{generate-id()}">
                <xsl:value-of select="title"/>
            </a>
        </p>
    </xsl:template>
    <xsl:template match="sect1" mode="body">
        <h2>
            <a name="{generate-id()}">
                <xsl:value-of select="title"/>
            </a>
        </h2>
        <xsl:apply-templates select="para"/>
    </xsl:template>
</xsl:stylesheet>

```

Quelques explications:

- les modes permettent d'écrire plusieurs règles pour un même élément
 - l'appel se fait en spécifiant le mode comme attribut dans apply templates
 - la description se fait en spécifiant le mode comme attribut dans l'élément `<xsl:template >`
- les accolades permettent d'évaluer des expressions dans les chaînes de caractères
- les éléments para sont traités grâce aux règles par défaut (écrire le contenu texte)

rajouter explications pour key et function-available()

Exercice 3 – Règles avec paramètres

Cette exercice copie un nœud et ses descendants, jusqu'à une profondeur spécifiée. La règle s'appelle elle-même de façon récursive.

Chaque fois que la règle s'appelle elle-même, elle réduit le profondeur d'une unité, jusqu'à ce que celle-ci soit égale à zéro, auquel cas la règle ne fait plus rien et s'arrête.

Le document XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="recursive.xsl"?>
<niveau0>
  <niveau1 arg="beta">
    <niveau2>
      <niveau3>texte</niveau3>
    </niveau2>
  </niveau1>
</niveau0>
```

Le résultat (pour une profondeur de 2 : `<xsl:with-param name="profondeur" select="2"/>`)

```
<?xml version="1.0" encoding="UTF-8"?>
<niveau0>
  <niveau1 arg="beta"/>
</niveau0>
```

Solution :

Principe : utiliser dans un template (*) un paramètre précisant la profondeur à laquelle on se trouve dans l'arbre. Chaque appel récursif `<xsl:apply-templates>` réutilisera la même règle avec un paramètre décrémenté.

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

```

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:apply-templates>
      <xsl:with-param name="prof" select="3" />
    </xsl:apply-templates>
  </xsl:template>

  <xsl:template match="*|text()">
    <xsl:param name="prof" />

    <xsl:if test="$prof > 0">
      <xsl:copy>
        <xsl:copy-of select="@*" />
      </xsl:copy>
    </xsl:if>

    <!-- traiter tous les enfants -->
    <xsl:apply-templates>
      <xsl:with-param name="prof" select="$prof - 1" />
    </xsl:apply-templates>
  </xsl:template>
</xsl:stylesheet>

```

Exercice 4 – Forum et fil de discussion

Reprenez le TD2 exercice 7. On souhaite afficher un fil de discussion à partir d'un message donné. Le point d'entrée du fil de discussion sera représenté par la référence du message d'entrée.

DTD correspondante au forum:

```

<!ELEMENT forum ( personnes, messages ) >
<!ELEMENT personnes ( personne* ) >
<!ELEMENT messages ( message* ) >

<!ELEMENT personne ( nom, prenom, adresse ) >
<!ATTLIST personne ref ID #REQUIRED>
<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT adresse (#PCDATA) >

<!ELEMENT message (#PCDATA) >
<!ATTLIST message ref ID #REQUIRED>
<!ATTLIST message expéditeur IDREF #REQUIRED >
<!ATTLIST message antérieur IDREF #IMPLIED >

```

Exemple de fichier XML forum:

```

<?xml version="1.0" ?>
<!DOCTYPE forum SYSTEM "forum.dtd" >
<forum>
  <personnes>
    <personne ref="p1">
      <nom>Ingals</nom>
      <prenom>Charles</prenom>
      <adresse>charles.ingals@lapetitemaison.fr</adresse>
    </personne>
  </personnes>

```

```

</personne>
<personne ref="p2">
  <nom>Ingals</nom>
  <prenom>Marie</prenom>
  <adresse>Marie.ingals@lapetitemaison.fr</adresse>
</personne>
<personne ref="p3">
  <nom>Ingals</nom>
  <prenom>Laura</prenom>
  <adresse>Laura.ingals@lapetitemaison.fr</adresse>
</personne>
</personnes>
<messages>
  <message ref="m1" expéditeur="p1">
    Bonjour
  </message>
  <message ref="m2" expéditeur="p3">
    Hello
  </message>
  <message ref="m4" expéditeur="p1" antérieur="m3">
    Ca va !! je coupe du bois pour l'hiver !!
  </message>
  <message ref="m3" expéditeur="p2" antérieur="m1">
    Bonjour, Ca va bien ?
  </message>
  <message ref="m5" expéditeur="p3" antérieur="m3">
    Ca va, Je vais à l'école
  </message>
  <message ref="m6" expéditeur="p3" antérieur="m4">
    Je m'occupe de la cheminée
  </message>
  <message ref="m7" expéditeur="p2" antérieur="m4">
    Je vais vider la réserve
  </message>
</messages>
</forum>

```

Ecrivez un fichier XSL pour produire un document HTML présentant de manière adaptée le fil de discussion à partir d'une référence de message.

L'application de la feuille de style sur l'exemple doit produire le fichier HTML suivant (fil de discussion à partir du message référencé par « m1 »).

Pensez à indenter correctement les messages en fonction de leurs relations dans le fil de discussion

```

Message : m1
Expéditeur : charles.ingals@lapetitemaison.fr
Texte : Bonjour
  Message : m3
  Expéditeur : Marie.ingals@lapetitemaison.fr
  Texte : Bonjour, Ca va bien ?

```

Idee :

Générale:

- créer une règle modèle fil de discussion
 - cette règle s'applique à des éléments messages
 - Elle affiche le message et lance l'application de la même règle modèle aux messages pour lesquels l'attribut antérieur correspond à la valeur de l'attribut ref du message courant
- Affichage des messages
 - Afficher la chaîne « message » et l'attribut ref (utilisation de `<xsl:value-of >` (cf amphi3 Tr24))
 - Afficher l'e-mail de la personne expeditrice
 - Afficher le contenu textuel
- Affichage de l'adresse e-mail
 - Sélectionner la valeur de la bonne personne (avec un `<xsl:apply-templates>` ou un `<xsl:value-of>`)
- Pour l'indentation
 - Il faut indenter après l'affichage des données d'un message et avant le `apply-templates` (qui génère les fils)
 - L'indentation se fait en créant un tableau `<table>` et en rajoutant une cellule vide, en créant une ligne `<tr>` et en rajoutant une cellule vide `<td>`

Une solution possible :

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

  <xsl:template match="/">
    <table border="1">
      <xsl:apply-templates select="//messages/message[@ref='m1']" mode="fil_discussion">
        </xsl:apply-templates>
      </table>
    </xsl:template>

    <xsl:template match="message" mode="fil_discussion">
      <xsl:variable name="reference" select="@ref" />
      <tr>
        <td width="10">
          </td>
        <td>Message :
          <b><xsl:value-of select="$reference" /></b>
          <br />
          Expéditeur :
          <b><xsl:variable name="exp" select="@expediteur"/>
            <xsl:value-of select="/forum/personnes/personne[@ref=$exp]/adresse/text()" /></b>
          <br />
          Texte :
          <b><xsl:value-of select="text()" /></b>
          <br />
          <table border="1">
            <xsl:apply-templates select="//messages/message[@antérieur=$reference]"
mode="fil_discussion" />
          </table>
        </td>
      </tr>
    </xsl:template>
  </xsl:stylesheet>
```

```
</xsl:template>
</xsl:stylesheet>
```

Note : voici une image avec border='1'

Message	:	m1
Expéditeur	:	charles.ingals@lapetitemaison.fr
Texte : Bonjour		
Message	:	m3
Expéditeur	:	Marie.ingals@lapetitemaison.fr
Texte : Bonjour, Ca va bien ?		

Il est possible d'optimiser en rajoutant un test pour éviter la création des tables vides quand on arrive à la fin d'un fil de discussion. Pour cela, rajouter un test (avec l'élément <xsl:if> (cf amphi 3 Tr54)

Le test peut être une expression Xpath

- Elle retourne true si le résultat de l'expression Xpath est non vide (auquel cas, il est utile de créer les tableaux)
- Pb : on fait deux fois le calcul Xpath ...

Soit :

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

  <xsl:template match="/">
    <table border='1'>
      <xsl:apply-templates select="//messages/message[@ref='m1']" mode="fil_discussion">
        </xsl:apply-templates>
      </table>
    </xsl:template>

    <xsl:template match="message" mode="fil_discussion">
      <xsl:variable name="reference" select='@ref' />
      <tr>
        <td width='10'>
          </td>
          <td>Message :
          <b><xsl:value-of select="$reference" /></b>
          <br />
          Expéditeur :
          <b><xsl:variable name="exp" select="@expéditeur"/>
            <xsl:value-of select="/forum/personnes/personne[@ref=$exp]/adresse/text()" /></b>
          <br />
          Texte :
          <b><xsl:value-of select="text()" /></b>
          <br />
          <xsl:if test="//messages/message[@antérieur=$reference]">
            <table border='1'>
              <xsl:apply-templates
                mode="fil_discussion" />
              </table>
            </xsl:if>
          </td>
        </tr>
      </xsl:template>
```

```
</xsl:stylesheet>
```

Et on obtient

```
Message : m1
Expéditeur : charles.ingals@lapetitemaison.fr
Texte : Bonjour
Message : m3
Expéditeur : Marie.ingals@lapetitemaison.fr
Texte : Bonjour, Ca va bien ?
```

Remarques :

- parcours systématique de tous les messages : c'est long (préférer localiser le plus précisément)
- J'ai pas trouvé mieux pour le moment qu'une variable pour faire des requêtes sur les IDs

Exercice 5 – Numérotation d'items

Soit le fichier XML suivant

```
<?xml version="1.0"?>
<liste>
  <item>item 1
  <item>ss-item 1.1
  <item>ss-ss-item 1.1.1</item>
  </item>

  <item>ss-item 1.2</item>
  </item>

  <item>item 2</item>

  <item>item 3
  <item>ss-item 1</item>
  </item>
</liste>
```

Ecrire un fichier XSL pour produire des listes indentées numérotées comme :

+.**1**item 1

+1.1ss-item 1.1

+1.2ss-item 1.2

+2item 2

+3item 3

+3.1ss-item 1

Solution

Idée générale

- Utiliser une règle pour afficher un item
- Cette règle utilise un paramètre qui est la position dans l'arborescence (représenté par la chaîne de caractères correspondante)
- Ce paramètre est mis à jour et transmis pour les appels de règle modèles suivants (avec la variable \$index)

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" />
  <xsl:template match="liste">
    <table border="1">
      <xsl:apply-templates>
        <xsl:with-param name="avant" select="" />
      </xsl:apply-templates>
    </table>
  </xsl:template>

  <xsl:template match="item">
    <xsl:param name="avant" />
    <xsl:variable name="index" select='concat( concat($avant, ".") , position())' />
    <tr>
      <td width='10'></td>
      <td><b><xsl:value-of select="$index" /></b>
        <xsl:value-of select="text()" />
        <br />
        <xsl:if test="item">
          <table border="1">
            <xsl:apply-templates select="item">
              <xsl:with-param name="avant" select="$index" />
            </xsl:apply-templates>
          </table>
        </xsl:if>
      </td>
    </tr>
  </xsl:template>
</xsl:stylesheet>
```

Remarques:

- Ne pas oublier les "" pour le point du concat dans la définition de index sinon, cela est interprété en . → tous les PCDATA du sous arbre
- Utiliser position
- De la même manière, il faudrait un if dans la création des tables