Cartouche du document

Année : ING 2
Activité : Travail dirigé

Objectifs

L'objectif de cette série d'exercices est :

- d'approfondir les langages XSL et XPath
- de découvrir des régles propres à XSLT 2.0

Quelques liens pour ce travail dirigé : XSLT 2.0, XPath 2.0 les fonctions XPath et SAXON (pour télécharger le moteur XSLT).

Sommaire des exercices

Corps des exercices

1 - Produits et Familles

Énoncé :

Dans cet exercice, on travaille avec le fichier XML qui suit :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<produits>
 <produit nom="p1" famille="f1" prixUnitaire="20" stock="15">
 descriptif de p1
 bla bla
 </produit>
 <produit nom="p2" famille="f2" prixUnitaire="50" stock="2">
 descriptif de p2
 bli bli
 </produit>
 <produit nom="p3" famille="f1" prixUnitaire="1" stock="23">
 descriptif de p3
 blo blo
 </produit>
 <produit nom="p4" famille="f2" prixUnitaire="3" stock="2">
 descriptif de p4
 bli bli
 </produit>
</produits>
```

Question 1)

Énoncé de la question

Ecrire la feuille XSL qui permet d'afficher dans un fichier xhtml les caractéristiques des produits regroupés par famille. On vous demande d'utiliser les objets suivants :

- la règle <xsl:for-each-group group-by="..." ....
- la fonction current-group pour parcourir tous les éléments d'un groupe;

• la fonction current-grouping-key pour récupérer la valeur commune du group-by.

Pour une bonne présentation du fichier résultat, vous devez associer à chaque balise
 un retour à la ligne. Ce traitement devra se faire à l'aide d'un template nommé.

Solution de la question

```
?xml version="1.0" encoding="iso-8859-1"?>
<!-- Début de la feuille XSL -->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
<!-- On génère la sortie en format xhtml -->
<xsl:output method="xhtml" indent="yes" encoding="iso-8859-1"
 doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
 doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
 transitional.dtd"/>
<!-- le patron du noeud produits -->
<xsl:template match="produits">
<html>
 <body>
 <xsl:for-each-group select="produit" group-by="@famille">
  <xsl:text>Famille : </xsl:text><xsl:value-of select="current-
  grouping-key()"/>
  <xsl:call-template name="nlbr"/>
<xsl:for-each select="current-group()">
<xsl:text>Produit : </xsl:text><xsl:value-of select="@nom"/>,
<xsl:text> Prix : </xsl:text><xsl:value-of select="@prixUnitaire"/
>,
<xsl:text> Stock : </xsl:text><xsl:value-of select="@stock"/>
<xsl:call-template name="nlbr"/>
 </xsl:for-each>
 <xsl:call-template name="nlbr"/>
 <xsl:call-template name="nlbr"/>
</xsl:for-each-group>
 </body>
</html>
</xsl:template>
<xsl:template name="nlbr">
<br/>
<xsl:text>
</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

Question 2)

Énoncé de la question

On désire ajouter dans la feuille précédente l'affichage pour chaque famille de la valeur moyenne (en euros) du stock. Les familles doivent être ordonnées. L'ordre d'affichage se fait par rapport aux stocks cumulés des produits du plus grand au plus petit.

On pourra utiliser les fonctions d'agrégation : sum et avg.

Solution de la question

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- Début de la feuille XSL -->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
<!-- On génère la sortie en format xhtml -->
<xsl:output method="xhtml" indent="yes" encoding="iso-8859-1"
 doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
 doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
 transitional.dtd"/>
<!-- le patron du noeud produits -->
<xsl:template match="produits">
<html>
 <body>
 <xsl:for-each-group select="produit" group-by="@famille">
 <xsl:sort select="sum(current-group()/@stock)" data-type="number"
 order="descending"/>
  <xsl:text>Famille : </xsl:text><xsl:value-of select="current-
  grouping-key()"/>
  <xsl:text> ---- Valeur moyenne du stock : </xsl:text><xsl:value-
  of select="avg(current-group()/(@stock * @prixUnitaire))"/>
  <xsl:call-template name="nlbr"/>
<xsl:for-each select="current-group()">
<xsl:text>Produit : </xsl:text><xsl:value-of select="@nom"/>,
<xsl:text> Prix : </xsl:text><xsl:value-of select="@prixUnitaire"/
>,
<xsl:text> Stock : </xsl:text><xsl:value-of select="@stock"/>
<xsl:call-template name="nlbr"/>
 </xsl:for-each>
 <xsl:call-template name="nlbr"/>
 <xsl:call-template name="nlbr"/>
</xsl:for-each-group>
 </body>
</html>
</xsl:template>
<xsl:template name="nlbr">
<br/>
<xsl:text>
```

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

Question 3)

Énoncé de la question

On désire inclure la notion de langue. On passera en paramètre la langue de l'utilisateur et la sortie sera présentée dans la langue choisie.

On devra créer autant de fichiers XML qu'il y a de langues. Tous ces fichiers auront le même schéma. Ils contiendront chacun les expressions utilisées dans la sortie traduite dans la langue. Dans le fichier XSL on utilisera la fonction document qui permettra de charger dans une variable globale l'arbre des expressions et donc d'accéder aux différentes expressions à n'importe quel endroit de la feuille XSL.

Rappel : Pour passer un paramètre dans la ligne de commande, on écrit **nomParam=valParam**.

Solution de la question

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- Début de la feuille XSL -->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
<!-- On génère la sortie en format xhtml -->
<xsl:output method="xhtml" indent="yes" encoding="iso-8859-1"
 doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
 doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
 transitional.dtd"/>
<!-- récupération du paramètre de la langue. fr est la langue par
défaut -->
<xsl:param name="langue">fr</xsl:param>
<!-- le patron du noeud produits -->
<xsl:template match="produits">
<html>
 <head>
 </head>
 <body>
  <!-- chargement du fichier des expressions dans la langue choisie
  par l'utilisateur -->
  <xsl:variable name="expressions"
  select="document(concat('produits-',$langue,'.xml'))/
  expressions"/>
 <xsl:for-each-group select="produit" group-by="@famille">
 <xsl:sort select="sum(current-group()/@stock)" data-type="number"
 order="descending"/>
  <xsl:value-of select="$expressions/expression[@id = 'FAMILLE']"/
  ><xsl:text> : </xsl:text><xsl:value-of select="current-grouping-
  key()"/>
```

```
  <xsl:text> </xsl:text>
  <xsl:value-of select="$expressions/expression[@id =
  'VALEUR_MOYENNE_STOCK']"/>
  <xsl:text> : </xsl:text><xsl:value-of select="avg(current-
  group()/(@stock * @prixUnitaire))"/>
  <xsl:call-template name="nlbr"/>
<xsl:for-each select="current-group()">
<xsl:value-of select="$expressions/expression[@id = 'PRODUIT']"/
><xsl:text> : </xsl:text><xsl:value-of select="@nom"/>,
<xsl:value-of select="$expressions/expression[@id =
'PRIX_UNITAIRE']"/><xsl:text> : </xsl:text><xsl:value-of
select="@prixUnitaire"/>,
<xsl:value-of select="$expressions/expression[@id = 'STOCK']"/
><xsl:text> : </xsl:text><xsl:value-of select="@stock"/>
<xsl:call-template name="nlbr"/>
 </xsl:for-each>
 <xsl:call-template name="nlbr"/>
 <xsl:call-template name="nlbr"/>
</xsl:for-each-group>
 </body>
</html>
</xsl:template>
<!-- le patron nommé pour placer dans la sortie la balise <br/> et
un retour à la ligne -->
<xsl:template name="nlbr">
<br/>
<xsl:text>
</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

2 - Recherche de mots dans un texte

Énoncé :

Dans cet exercice, on travaille avec le fichier XML de la forme suivante :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<texte>
 <motCle>classe</motCle>
 <motCle>UML</motCle>
 <motCle>attribut</motCle>
 <contenu>On appelle classe la structure d'un objet, c'est-à-
 dire la déclaration de l'ensemble des entités qui composeront un
 objet. Un objet est donc "issu" d'une classe, c'est le produit
 qui sort d'un moule. En réalité on dit qu'un objet est une
 instanciation d'une classe, c'est la raison pour laquelle on
```

```
 pourra parler indifféremment d'objet ou d'instance (éventuellement
 d'occurrence).
Une classe est composée:
* d'attributs: il s'agit des données, dont les valeurs
représentent l'état de l'objet
* Les méthodes : il s'agit des opérations applicables aux objets
Si on définit la classe voiture, les objets Peugeot 406,
Volkswagen Golf seront des instanciations de cette classe. Il
pourra éventuellement exister plusieurs objets Peugeot 406,
différenciés par leur numéro de série.
Mieux, deux instanciations de classes pourront avoir tous leurs
attributs égaux sans pour autant être un seul et même objet (c'est
la différence entre état et identité). C'est le cas dans le monde
réel, deux T-shirts peuvent être strictement identique (avoir
le même état) et pourtant ils sont distincts (ils ont chacun
leur identité propre). D'ailleurs en les mélangeant il serait
impossible de les distinguer...
Une classe se représente avec UML sous forme d'un rectangle divisé
en trois sections. Le premier contient le nom donné à la classe
(non souligné). Les attributs d'une classe sont définis par un
nom, un type (éventuellement une valeur par défaut, c'est-à-dire
une valeur affectée à la propriété lors de l'instanciation) dans
le second compartiment. Les opérations sont répertoriées dans le
troisième volet du rectangle.
 </contenu>
</texte>
```

Question 1)

Énoncé de la question

Ecrire la feuille XSL qui reçoit en paramètre un mot et qui affiche dans un fichier texte le nombre d'occurences de ce mot dans le texte contenu dans le noeud contenu.

On pourra s'inspirer du code xsl qui suit :

```
<!-- le patron nommé pour remplacer dans une chaîne les retours à
la ligne par la balise br -->
<xsl:template name="nlTobr">
<xsl:param name="string"/>
 <xsl:choose>
 <xsl:when test="contains($string,'&#10;')">
  <xsl:call-template name="tabToli">
   <xsl:with-param name="string" select="substring-
   before($string,'&#10;')"/>
  </xsl:call-template>
  <br/>
  <xsl:call-template name="nlTobr">
  <xsl:with-param name="string" select="substring-
  after($string,'&#10;')"/>
```

```
 </xsl:call-template>
 </xsl:when>
 <xsl:otherwise>
  <xsl:call-template name="tabToli">
   <xsl:with-param name="string" select="$string"/>
  </xsl:call-template>
 </xsl:otherwise>
 </xsl:choose>
</xsl:template>
<!-- le patron nommé pour remplacer dans une chaîne les
tabulations par une série d'espaces insécables -->
<xsl:template name="tabToli">
<xsl:param name="string"/>
 <xsl:choose>
 <xsl:when test="contains($string,'&#9;')">
  <xsl:value-of select="substring-before($string,'&#9;')" disable-
  output-escaping="yes"/>
  <!--   est le code ascii de l'espace insécable -->
     
  <xsl:call-template name="tabToli">
  <xsl:with-param name="string" select="substring-
  after($string,'&#9;')"/>
  </xsl:call-template>
 </xsl:when>
 <xsl:otherwise>
  <xsl:value-of select="$string" disable-output-escaping="yes"/>
 </xsl:otherwise>
 </xsl:choose>
</xsl:template>
```

Solution de la question

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- Début de la feuille XSL -->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
<xsl:output method="text" indent="yes" encoding="iso-8859-1"/>
<xsl:param name="mot"/>
<xsl:template match="texte">
 <xsl:choose>
  <xsl:when test="$mot = ''"><xsl:text>Vous devez préciser le mot
  recherché dans la ligne de commande mot=???</xsl:text></xsl:when>
  <xsl:otherwise>
   <xsl:variable name="nbOcc">
```

```
   <xsl:call-template name="compterOccurences">
    <xsl:with-param name="string" select="upper-case(contenu/
    text())"/>
    <xsl:with-param name="motCle" select="upper-case($mot)"/>
    <xsl:with-param name="nbOcc" select="0"/>
   </xsl:call-template>
  </xsl:variable>
  <xsl:text>Dans le texte ci-dessous, il y a </
  xsl:text><xsl:value-of select="$nbOcc"/> occurences du mot
  <xsl:value-of select="$mot"/>
  <xsl:text>
  </xsl:text>
  <xsl:value-of select="contenu/text()"/>
 </xsl:otherwise>
</xsl:choose>
</xsl:template>
<!-- le patron nommé pour compter le nombre d'occurences de motCle
dans string -->
<xsl:template name="compterOccurences">
<xsl:param name="string"/>
<xsl:param name="motCle"/>
<xsl:param name="nbOcc"/>
 <xsl:choose>
 <xsl:when test="contains($string,$motCle)">
  <xsl:call-template name="compterOccurences">
   <xsl:with-param name="string" select="substring-after($string,
   $motCle)"/>
   <xsl:with-param name="motCle" select="$motCle"/>
   <xsl:with-param name="nbOcc" select="$nbOcc + 1"/>
  </xsl:call-template>
 </xsl:when>
 <xsl:otherwise>
  <xsl:value-of select="$nbOcc"/>
 </xsl:otherwise>
 </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

Question 2)

Énoncé de la question

Ecrire la feuille XSL qui affiche pour chaque mot-clé, le mot-clé et son nombre d'occurences dans le texte. L'affichage se fera par ordre décroissant d'occurences.

Solution de la question

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

```
<!-- Début de la feuille XSL -->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
<xsl:output method="text" indent="yes" encoding="iso-8859-1"/>
<xsl:template match="texte">
 <xsl:variable name="texte" select="contenu/text()"/>
 <xsl:variable name="occurences">
  <xsl:for-each select="motCle">
   <xsl:element name="mot">
    <xsl:attribute name="libelle" select="text()"/>
    <xsl:call-template name="compterOccurences">
     <xsl:with-param name="string" select="upper-case($texte)"/>
     <xsl:with-param name="motCle" select="upper-case(text())"/>
     <xsl:with-param name="nbOcc" select="0"/>
    </xsl:call-template>
   </xsl:element>
  </xsl:for-each>
 </xsl:variable>
 <xsl:text>Fréquences (par ordre décroissant) des différents mots-
 clés dans le texte qui suit &#10;</xsl:text>
 <xsl:for-each select="$occurences/mot">
  <xsl:sort select="text()" data-type="number" order="descending"/>
  <xsl:value-of select="@libelle"/><xsl:text> : </
  xsl:text><xsl:value-of select="text()"/><xsl:text>&#10;</
  xsl:text>
 </xsl:for-each>
 <xsl:text>&#10;</xsl:text>
 <xsl:value-of select="contenu/text()"/>
</xsl:template>
<!-- le patron nommé pour compter le nombre d'occurences de motCle
dans string -->
<xsl:template name="compterOccurences">
<xsl:param name="string"/>
<xsl:param name="motCle"/>
<xsl:param name="nbOcc"/>
 <xsl:choose>
 <xsl:when test="contains($string,$motCle)">
  <xsl:call-template name="compterOccurences">
   <xsl:with-param name="string" select="substring-after($string,
   $motCle)"/>
   <xsl:with-param name="motCle" select="$motCle"/>
   <xsl:with-param name="nbOcc" select="$nbOcc + 1"/>
```

```
 </xsl:call-template>
 </xsl:when>
 <xsl:otherwise>
  <xsl:value-of select="$nbOcc"/>
 </xsl:otherwise>
 </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```