Cartouche du document

Année : ING 1
Matière : Struc. Info. : XML
Activité : Travail dirigé

Objectifs

Cette série d'exercices a pour but

- d'approfondir la récurisvité
- d'utiliser des fonctions XPATH

Sommaire des exercices

Corps des exercices

1 - du texte pur vers son équivalent en HTML

Énoncé :

Dans cet exercice, on s'intéresse à un problème classique : convertir du texte pur pour qu'il soit affiché à l'identique en HTML.

Question 1)

Énoncé de la question

On s'intéresse à une liste de produits décrits comme suit :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<produits>
 <produit id="1">
  <libelle>produit1</libelle>
  <descriptif>
  Ce produit est intéressant pour les raisons suivantes :
  1) il n'est pas cher
  2) il est écologique
  3) il n'est pas dangeureux
  </descriptif>
 </produit>
</produits>
```

Ecrire une feuille xsl qui génère une sortie HTML dans laquelle pour chaque produit on a :

- son libellé en gras
- son descriptif dont on aura reproduit fidèlement la présentation dans le fichier XML.

Solution de la question

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- entête commune à tous les fichiers XMl -->
<!-- Début de la feuille XSL -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
<xsl:output method="html" indent="yes" encoding="iso-8859-1" />
```

```
<!-- le patron du noeud produits -->
<xsl:template match="produits">
<html>
 <body>
  <xsl:apply-templates/>
 </body>
</html>
</xsl:template>
<!-- le patron du noeud produit -->
<xsl:template match="produit">
 Produit : <b><xsl:value-of select="libelle"/></b>
 <br/>
 Descriptif :
 <b>
 <xsl:call-template name="nlTobr"><xsl:with-param name="string"
 select="descriptif"/></xsl:call-template>
 </b>
 <br/><br/>
</xsl:template>
<!-- le patron nommé pour remplacer 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 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>
<!-- fin de la feuille xsl-->
</xsl:stylesheet>
```

2 - Affichage d'une commande

Énoncé :

Dans cet exercice, on s'intéresse aux données numériques et aux fonctions associées.

Question 1)

Énoncé de la question

On dispose d'une commande exprimée dans un format xml.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<commande>
 <produit> <libelle>produit1</libelle> <quantite>12</quantite>
 <prix>5</prix> </produit>
 <produit> <libelle>produit2</libelle> <quantite>5</quantite>
 <prix>4</prix> </produit>
 <produit> <libelle>produit3</libelle> <quantite>18</quantite>
 <prix>2</prix> </produit>
</commande>
```

Ecrire un fichier XSL de règles qui permet d'afficher la commande. La présentation est comme suit :

```
 Produit libelleProduit  : quantité : prix du produit en euro :
 quantité * prix du produit en euro
```

. . .
. . .

Solution de la question

Question 2)

Énoncé de la question

On reprend l'exercice précédent et on ajoute à la fin de la commande le total de la commande en euros.

On pourra stocker dans une variable la liste des valeurs quantité * prix du produit en euro séparées par des virgules. On écrit une fonction récursive qui analyse le contenu de cette variable afin de calculer le total.

Solution de la question