

XML & XSLT

Quelques fonctions classiques

XSL : Utilisation des fonctions et opérateurs

- XSL propose une grande variété de fonctions et les opérateurs classiques.
- XSL n'est pas un langage impératif. L'appel d'une fonction ou d'un opérateur ne se fait que dans une clause **select** ou une clause de **test**.

<!-- On fait la somme des variables a et b -->

<xsl:value-of select="\$a + \$b"/>

<!-- On test si le nœud courant (contexte) est le dernier -->

<xsl:if test="position() = last()>

XSL : Les fonctions de chaînes

- La fonction **concat** permet de concaténer des chaînes
Le nombre de chaînes à concaténer est variable.
La fonction renvoie une nouvelle chaîne qui est donc la concaténation des chaînes passées en paramètres.

```
<!-- On construit un nom de fichier avec l'extension .gif -->  
<xsl value-of select="concat($nom, '.gif')"/>
```
- La fonction **substring** permet d'extraire une sous chaîne dans une chaîne. Elle a trois paramètres : la chaîne, la position de début et le nombre de caractères à extraire.

```
<!-- Dans un numéro de fixe passé en attribut on extrait les chiffres  
pour la région -->  
<xsl:value-of select="substring(@noTel,1,2)"/>
```
- Si on omet le 3^{ème} paramètre dans un appel de substring, alors la fonction renvoie toute la chaîne à partir de la position de début indiqué par le deuxième paramètre.

XSL : Les fonctions de chaînes

- La fonction `substring-before(txt,token)` permet d'extraire de la chaîne `txt`, la sous chaîne située avant la chaîne `token`.
- La fonction `substring-after(txt,token)` permet d'extraire de la chaîne `txt`, la sous chaîne située après la chaîne `token`.
- La fonction `string-length(txt)` renvoie la longueur de la chaîne `txt`.
- La fonction `translate(txt,avant,apres)` renvoie une copie de la chaîne `txt` dans laquelle tous les caractères `avant` ont été remplacés par les caractères `apres`. Si la chaîne `apres` est plus courte que la chaîne `avant` alors des caractères ne seront pas traduits.

XSL : La fonction document (1)

- Cette fonction permet en plus du document XML d'entrée de charger en mémoire vive un autre document.
- Le document doit être un arbre XML valide.
- On extrait tout le fichier

<!-- On renvoie le contenu du fichier monFichier.xml -->

<xsl:value-of select="document('monFichier.xml')" />

- On extrait une partie du fichier

<!-- On renvoie une partie du contenu du fichier
monFichier.xml -->

<xsl:value-of
select="document('monFichier.xml')/pere/fils[@att =
'???']" />

XSL : La fonction document (2)

- Pour récupérer l'arbre dans une variable

<!-- On renvoie le contenu du fichier monFichier.xml -->

<xsl:variable name="monArbre">

 <xsl:copy-of select="document('monFichier.xml')" />

</xsl:variable>

- Pour manipuler cet arbre, dans la clause select

<xsl:??? select="\$monArbre/pere/..." >

...

</xsl:???>

XSL : Formats text et number

- XSL est un langage non typé.
- Toute information est par défaut une chaîne de caractères.
- Cela pose un certain nombre de problèmes quand le contenu de l'information est du numérique.
- Dans la clause `xsl:sort` il faut préciser si ce que l'on veut trier est du texte ou du numérique avec l'attribut `data-type`.

Si en entrée, on a la série 5, 13 et 12 :

- avec `<xsl:sort data-type="text"/>` on aura en sortie 12, 13 et 5
- avec `<xsl:sort data-type="number"/>` on aura en sortie 5, 12 et 13

XSL : Formats text et number

- Dans une clause `test` si les valeurs à tester sont des numériques il faut utiliser la fonction `format-number`.
- Cette fonction reçoit deux paramètres : la valeur et le format pour transformer cette valeur.
- Dans le format on utilise les caractères `0` et `#`. `0` signifie qu'un chiffre non significatif sera remplacé par `0`. `#` signifie qu'un chiffre non significatif ne sera pas remplacé.

`format-number(53.51, "000.###")` retourne 053.51

`format-number(53.51, "000.000")` retourne 053.510

`format-number(53.51, "#.000")` retourne 53.510

- `<xsl:if test="format-number($prix,"000.00") < '012.25'">`
- Par défaut la virgule est le séparateur de groupe et le point le séparateur de décimal
- `format-number(5351, "#,###")` retourne 5,351