Boucle récursive de parcours de branche:

```
<xsl:template name="boucleRecursive">
        <xsl:param name="Node"/>
        ...
        <xsl:if test="$Node/following-sibling::*">
        <xsl:call-template name="boucleRecursive">
        <xsl:with-param name="Node"
                select="$Node/following-sibling::*"/>
        </xsl:call-template>
        </xsl:if>
</xsl:template>
```

Xsl number (*value-of* return like):

```
<xsl:number level="single où multiple"
          count="NodeName [|UnionNodeName]"
           from="StartNodeXpath" format="A.1" />
```

Fonctions XSL 2.0 déclaration:

```
<xsl:function name="NameSpace:functionName">
  <xsl:param name="param1"/>
  <xsl:param name="param2"
                    [select="XpathExpression"]/>
        <!-- Contenu de la fonction-->
    <!--retour de fonction par xsl:value-of -->
<xsl:function>
```

Fonctions 2.0 usage :

```
  <xsl:value-of select="NameSpace:functionName
                        (param1,param2)"/>
```

HTML

Structure minimum Html :

```
<html>
   <head>
      <title>Titre de votre page</title>
   </head>
   <body> <!-- Contenu de page -->        </body>
</html>
```


Intégration fichier css (dans <head>) :

```
<link href="fichier.css" rel="stylesheet" type="text/css">
```

Références balises Html :

Assignation class	`<balise class="class1 [cl2 ...]"/>`
Assignation id	`<balise id="idName"/>`
Elément bloc	`<div>…</div>`
Elément enligne	`<span>…</span>`
Tableau, ligne, cellule	`<table><tr> <td>cellule</td> </tr></table>`
Lien	`<a href='lien'>visuel</a>`
image	`<img src="cheminImg"/>`
Liste de puces et puces	`<ul><li>Une puce</li></ul>`
Formulaire, entrée, soumission	`<form method="POST|GET" action="lienTraitement"> <input type="text|password|checkbox|hidden|radiobutton" name=Nom/> <input type="submit" value="Valider"/> </form>`
Paragraphe	`<p> contenu paragraphe </p>`
Titres niveau 1(fort) à 6 (faible)	`<h1>…</h1>, --> <h6>…</h6>`

CSS

Syntaxe fichier CSS ou enligne html:

sélecteur *{ propriété : valeur ;[propriété : valeur ;] }*

```
<balise style="propriété : valeur ; [ propriété : valeur ; ] "/>
```

Sélecteurs simples CSS :

.NomClass	→Classe réutilisable dans la page
#NomIdentifiant	→Identifiant à usage unique par page
NomDeBalise	→Balise portant NomDeBalise

Sélecteur encapsulé :

SelecteurConteneur SelecteurContenu

Sélecteur multiple :

Selecteur1, Selecteur2, ...

Liste de propriétés : **margin**[-left|-right|-top|-bottom], **padding**[-left|-right|-top|-bottom], **left**, **right**, **top**, **bottom**, **text-align**, **color**, **background**[-color|-image|-attachement|-position], **opacity**, [min-|max-]**width**, [min-|max-]**height**, **display**, **visibility**, **cursor**, **z-index**, **float**, **overflow**, **border**[-width|-color|-style], **...**

Les mémentos Programs Is What You Are, ou piwya, sont une suite d'outils d'aide au développement informatique.

Un concentré XML dans un format poche

Le mémento XML, XPath, XSL, CSS est le rassemblement de tout ce qu'il est nécessaire de savoir sur ces technologies, dans un format de poche transportable partout, et fait pour faciliter la vie du développeur XML et dérivés.

Avant toute chose, ce guide est un outil pédagogique pour progresser en Xsl dès les premiers pas, mais il convient tout a fait aux développeurs plus aguerris.

piwya.net

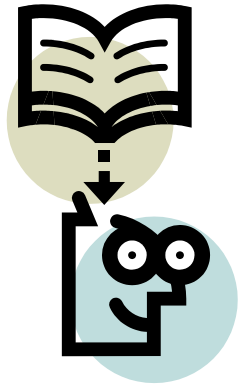
W3schools.com

30000001

30000001

MEMENTO
piwya
XML, XPath, Xsl, CSS

Le mémento des technologies XML



Section : Web XML développement

A.Desorbaix

XML, XPath

Règles XML :

- 1 seule balise Root
- Pas de balise ouverte
- Pas de balises croisées
- nom d'attribut, nom de balises, valeurs d'id commencent par une lettre

XPath

Sélecteurs :

nodeName	→ sélectionne tous les nœuds fils du nœud
/	→ nœud racine
//	→ tous les nœuds depuis le contexte.
.	→ nœud courant
..	→ nœud parent du nœud courant.
@	→ attribut
*	→ n'importe quel nœud élément
@*	→ n'importe quel nœud attribut
text()	→ n'importe quel nœud texte
node()	→ n'importe quel nœud
expr[exprboo]	→ expr : [n] à nième élément [exprBool] vérifiée
| (pipe)	→ UNION
last()	→renvoient le dernier
position()	→renvoient la position courante

Axes :

self	→nœud courant
ancestor[-or-self]	→Ancêtre [ou courant]
descendant[-or-self]	→Descendant [ou courant]
child, parent	→enfant, parents
following[-sibling]	→suivant [même fratrie]
preceding[-sibling]	→suivant [même fratrie]
namespace	→espace de nom

Axes usage :

cheminXpath/axeName::nodeName/fils

Xpath, fonctions préfixe : fn:*

Evaluation XPath dans un attribut de sortie :

<BaliseHtml attribut="{*ExpressionXpath*}" />

Fonctions Boolean:

boolean(arg)→ cast, *true*() à vrai, *false*() → faux

not(boolean) → renvoie l'inverse de boolean

Fonctions Math:

number(arg) → cast *abs*(num) → valeur absolue

floor(num) → Entier inf. *ceiling*(num) → Entier sup.

round(num) → Arrondi entier plus proche

Fonctions String:

concat(arg1,arg2) → concaténation de arg1 et arg2

compare(str1,str2) → Compare 2 chaines str1 et str2

substring(str ,s[,L])→Coupe str de s[sur L caractères]

string-length(str) → renvoie la taille de str

contains(str,str2),*match*(str,str2)→Vrai si str2 dans str

end-with(str,str2),*start-with*(str,str2)→Vrai si str débute ou finie par str2

Fonctions agrégat 1.0 & 2.0:

min(Node) →min *max*(Node) →faux

count(Node) →Nombre d'éléments Node

sum(Node) →Somme des éléments Node

avg(Node) →Moyenne des éléments Node (2.0)

Fonctions de nœuds :

position()→Renvoie la position du contexte

last()→position du dernière éléments du contexte

Opérateurs :

arithmétiques → + - div * mod

logique → = != < > <= >= or and |

Xsl

Liaison XML/XSL(text/Xsl) ou Css(text/css) :

```
<?xml-stylesheet type="text/ext" href="fichier.ext"?>
```

Commentaires :

```
<xsl:comment> Commentaires </xsl:comment>
```

Balise racine :

```
<xsl:stylesheet version="1.0|2.0"
   xmlns:xsl="http://www. w3.org/1999/XSL/Transform" >
```

Balise type de sortie :

```
<xsl:output
      method="xml|html|text" [indent="yes|no"]
            [omit-xml-declaration="yes|no"] />
```

Balise d'affichage de valeur :

```
<xsl:value-of select="Xpath expression ou Node"/>
```

Expression Xpath dans un attribut de balise finale:

```
<balise attr="{XpathExpr ou XpathNode}"/>
```

Balise de test simple :

```
<xsl:if test="Bool Xpath exp">…</xsl:if>
```

Test complexe :

```
<xsl:choose>
   <xsl:when test="Bool Xpath exp">…</xsl:when>
   <xsl:otherwise>…</xsl:otherwise>
</xsl:choose>
```

Boucles:

```
<xsl:for-each select="XpathExpression">
            <…/>
      </xsl:for-each>
```

Balise génération élément & attribut :

```
<xsl:element name="ElementName">
   <xsl:attribute name="AttributName">
      ValAttribut
   </xsl:attribute>
   ValElement
</xsl:element>
```


Xsl

Include prioritaire de bibliothèques Xsl :

```
<xsl:import href="fichier.xsl"/>
```

Import de bibliothèques Xsl :

```
<xsl:import href="fichier.xsl"/>
```

Templates match :

```
<xsl:template match="CheminNodeXpath" > …
</xsl:template>
```

Variables const et usage :

```
<xsl:variable name="varName"
                  select="XpathExpression"/>
<xsl:value-of select="$varName"/>
```

Template match & apply:

```
<xsl:template match="XpathNodeName">
      …
</xsl:template>
<xsl:apply-template match="XpathNode"/>
```

Template nommé , param [valeur par défaut] :

```
<xsl:template name="tmplName">
<xsl:param name="prmName"
                  [select="XpathExpression"] />
   <xsl:value-of select="$prmName"/> etc. …
</xsl:template>
```

Template nommé sans [avec] param appels :

```
<xsl:call-template name="tmplName">
   [<xsl:with-param name="prmName"
                  [select="XpathExpression"/>]]
</xsl:call-template>
```

Multi-XML 2.0:

```
<xsl:variable name="xmlDoc"
            select="Document('fichier.xml')" />
<xsl:value-of select="$xmlDoc/nodeName"/>
```

Trie par nœud :

```
<xsl:sort select="NodeNameTrie"/>
```