L'Altruiste : Le guide des langages Web

Le langage XSL

Sommaire

1 / Introduction

Le langage XSL permet de présenter visuellement des éléments définis dans un document XML alors que le langage XML (eXtended Markup Language) définit plutôt la sémantique (le sens) des données.

Le langage XSL se divise en trois parties principales :

- **Le formatage** : application de règles de style sur des éléments XML à l'instar du langage CSS.
- **La transformation** : substitution d'un marquage XML en un balisage HTML ou un autre marquage XML.

La partie *formatage* du langage XSL (eXtensible Stylesheet Language) a une fonction semblable à celle du langage CSS (Cascading StyleSheet).

Un ensemble de propriétés de style permet de contrôler la présentation des données à l'écran.

Ainsi, un document XML pourra être mis en forme pour un affichage sur un moniteur informatique, un écran de télévision ou pour l'impression ou encore pour une version auditive.

Le langage XSL, par une série de règles de transformation, remplace les éléments XML et leurs attributs en balisage HTML (HyperText Markup Language) ou en d'autres marqueurs XML. Cette section du langage XSL s'appelle XSLT soit Langage des feuilles de Style de Transformation dont les spécifications sont mises au point par le W3C (Worl Wide Web Consortium).

Par des règles de transformation, les données textuelles contenues dans les éléments XML ou dans leurs attributs sont présentés selon le résultat de la génération d'un balisage HTML.

Une feuille de style XSL peut entièrement réorganiser les éléments XML en sélectionnant des éléments et leurs attributs puis en les transformant en d'autres éléments.

Cette fonctionnalité peut être utilisée lorsqu'un document XML doit être intégré dans un ensemble d'autres documents XML. Si les éléments ne correspondent pas, alors il est nécessaire de les transformer afin de les rendre compatibles.

Le fonctionnement du langage XSL s'effectue selon des règles de style applicables à différent motif d'un document XML.

Ainsi, tous les motifs énoncés par la feuille de style et trouvés dans le document XML par un processeur XSL, sont soit formatés, soit transformés en une combinaison textuelle spécifique.

L'exemple ci-dessous montre un exemple complet regroupant un document XML et sa feuille de style XSL.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="
coursxsl/style.xsl"?>
<!DOCTYPE poesie [
 <!ELEMENT poesie (titre,texte,auteur)>
 <!ELEMENT titre (#PCDATA)>
 <!ELEMENT texte (#PCDATA)>
 <!ELEMENT auteur (#PCDATA)>
]>
<poesie>
 <titre>Locution des pierrots</titre>
 <texte>
 Je ne suis qu'un viveur lunaire
 Qui fait des ronds dans le bassin
 Et cela, sans autre dessein
 Que de devenir légendaire.

 Retroussant d'un air de défin
```

```
Mes manches de Mandarin pâle,
J'arrondis ma bouche et - j'exhale
Des conseils doux de Crucifix

Ah! oui, devenir légendaire,
Au seuil des siècles charlatans !
Mais où sont les Lunes d'antan ?
Et que Dieu n'est-il à refaire ?
</texte>
<auteur>Jules Laforgue</auteur>
</poesie>

<!-- Feuille de style : style.xsl -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:output method="html" encoding="ISO-8859-1"/>
<xsl:template match="/">
<html>
   <xsl:apply-templates/>
</html>
</xsl:template>
<xsl:template match="poesie">
<head>
   <title>
     <xsl:value-of select="titre"/>
   </title>
</head>
<body>
   <h2 align="center">
     <xsl:value-of select="titre"/>
   </h2>
   <div align="center">
     <pre>
     <xsl:value-of select="texte"/>
     </pre>
   </div>
   <h4 align="center">
     <xsl:value-of select="auteur"/>
   </h4>
</body>
</xsl:template>
</xsl:stylesheet>
```

2 / Le processus de transformation

La transformation ou le formatage permet d'afficher sur le navigateur Internet d'un ordinateur client, les données d'un document XML structurées en conformité avec les règles de style d'un document XSL.

Selon la méthode de sortie de la feuille de style, **le document résultant peut être généré en HTML, en texte brut ou en une autre arborescence XML**.

Le processus de transformation ou de formatage d'un document XML peut s'effectuer par trois moyens distincts.

- **Un processeur XSL installé sur l'ordinateur client** (en général dans le navigateur) se charge de transformer localement la source XML en conjonction avec la feuille de style XSL.
- **Un processeur XSL installé sur le serveur** envoie, après traitement, le document résultant au client.
- **Un logiciel spécifique installé sur le serveur** effectue un traitement préalable des documents XML, puis stocke les documents résultants sur le serveur lui-même.

Ces trois moyens nécessitent diverses ressources logiciels, mais tous font appel à un langage XSL identique sans injection d'instructions spécifiques.

La première solution demande à l'ordinateur client de posséder impérativement **un navigateur compatible avec les technologies XML** à l'image d'Internet Explorer 6.0 ou Netscape 6.1.

La seconde solution demande **un aménagement logiciel du serveur web** avec par exemple, un programme XML Enabler d'IBM, afin de le rendre compatible aux technologies XML/XSL.

Enfin, la dernière solution consiste à installer **un moteur de transformation XML nommé XT** (XML Transformer) associé à **un analyseur (ou parser) conforme à SAX (Simple API for XML) ou DOM (Document Object Model)**.

De nombreux programmes permettent de mettre à niveau son serveur :

- JAXP édité par Sun Microsystems.
- Xalan & Xerces édités par l'Oraganisation Apache
- XP & XT édités par l'auteur des spécifications XSL, James Clark.
- MSXML 4.0 édité par Microsoft.
- XML Parser édité par IBM.

L'analyse d'un document XML peut être accomplie selon deux méthodes.

Le DOM (Document Object Model) autorise la gestion de l'ensemble d'un document. Dans ce cas, la source XML est entièrement analysée par rapport au DTD (Document Type Definition) ou au schéma (XSchema) associé. **Si le document est conforme** aux règles structurelles énoncées alors le *parser* le valide, subséquemment peut commencer la manipulation de l'arborescence XML par un programme.

Le **DOM possède un inconvénient** du fait de l'analyse complète d'un document XML avant traitement. En effet, si la source XML est volumineuse, l'analyse risquerait d'affecter les performances d'un serveur.

Le SAX (Simple API for XML) permet de traiter événementiellement un document XML. Les éléments contenus dans le document XML sont analysés au fur et à mesure de leur exploitation par un programme de formatage. Ainsi, seuls les éléments utilisés sont soumis à une validation par le *parser*.

En fait, **lorsque l'analyseur SAX rencontre un nouveau noeud** (racine, élément, attribut, données textuelles analysées, instruction de traitement, espace de noms ou commentaire), est créé un événement que réceptionne le programme de transformation. En fonction de ces prérogatives, le programme déclenchera l'analyse puis traitera le noeud concerné.

Contrairement au DOM, **SAX consomme moins de ressources machines** et en conséquence s'adapte

idéalement au traitement de document XML volumineux. DOM sera préféré pour des sources XML plus légères.

Par ailleurs, **un processeur XML peut être validant ou non-validant ou les deux**.
Ce type doit être pris en considération, en fonction du genre d'activité souhaitée.

Un *parser* non-validant est forcément plus performant.
Cependant, l'arborescence du document XML ne sera pas analysée en conformité avec son DTD ou son schéma. Autrement dit le document XML sera obligatoirement bien formé mais risquera d'être non-valide.

En outre, **les entités analysées ne pourront faire l'objet de substitutions textuelles** dans le document réultant puisque le DTD ne sera pas pris en compte dans le traitement de la source XML.

En raison de ce genre de problèmes, **il est préférable d'utiliser un analyseur XML validant**.

En savoir plus :

W3C Microsoft

3 / Les espaces de noms

Afin de reconnaître les éléments et les attributs des espaces de noms XSL, le processeur XSL a besoin de les identifier précisément.

Les éléments des feuilles de style XSLT utilisent le préfixe *xsl:* pour référencer les éléments de l'espace de noms XSLT (XSL Transformation).

Quant aux feuilles de style de formatage, le préfixe utilisé dans les marqueurs XSL, est *fo:*.

Cependant, **les feuilles de style XSL sont libres d'utiliser n'importe quel préfixe**, à condition que l'espace de noms soit déclaré au préalable.

La déclaration de l'espace de noms s'effectue en associant un préfixe à un URI (Uniform Resource Identifier) dans l'élément racine de feuille de style.

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

- L'URI associé au préfixe *xsl:* est :

  http://www.w3.org/1999/XSL/Transform

- L'URI associé au préfixe *fo:* est :

  http://www.w3.org/1999/XSL/Format/1.0

4 / Les Patterns

Les *patterns* sont des expressions destinées à la sélection de noeuds dans l'arborescence d'un document XML.

En général, les *patterns* se trouvent dans l'attribut *select* ou *match* des éléments XSL-Tranformation suivants :

- xsl:template
- xsl:apply-templates
- xsl:value-of
- xsl:copy-of
- xsl:sort
- xsl:for-each

```
<xsl:template match="pattern">

<xsl:apply-templates select="pattern">
```

Pattern	Description
paragraphe	n'importe quel élément *paragraphe* du noeud courant.
***/paragraphe**	n'importe quel élément *paragraphe* petit-fils du noeud courant.
chapitre|paragraphe	n'impote quel élément *chapître* et n'importe quel élément *paragraphe*.
chapître/paragraphe	tout élément *paragraphe* avec un élément parent *chapître*.
chapître//paragraphe	tout élément *paragraphe* descendant d'un élément *chapître* à quelque niveau de profondeur que ce soit.
/	le noeud racine.
./	tous les éléments fils du noeud courant.
../chapître	tout les éléments *chapître* fils du père du noeud courant.
id("Identificateur")	l'élément avec l'identificateur unique *Identificateur*.
para[1]	n'importe quel élément *para* qui est le premier élément enfant *para* de son parent
***[position()=1 and self::para]**	n'importe quel élément *para* qui est le premier élément enfant *para* de son parent
para[last()=1]	n'importe quel élément *para* qui est seulement l'élément enfant *para* de son parent
items/item[position()>1]	tout élément *item* qui a un *item* parent et qui n'est pas le premier *item* enfant de son parent.
item[position() mod 2 = 1]	n'importe quel élément *item* qui est un élément enfant *item* impair de son parent.
div[@class="appendice"]//para	tout élément *p* avec un élément *div* ancètre qui a un attribut *class* avec la valeur *appendice*.
paragraphe/@class	tout attribut *class* des éléments *paragraphe* mais pas tout élément *paragraphe* qui a un attribut *class*.
@*	tout attribut.

Exemple [voir]

Considérons l'arborescence d'un document XML suivant :

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

```
<!DOCTYPE librairie [
  <!ELEMENT librairie (categorie+)>
  <!ELEMENT categorie (livre+)>
  <!ATTLIST categorie nom CDATA #REQUIRED>
  <!ELEMENT livre (titre, auteur)>
  <!ATTLIST livre ISBN CDATA #REQUIRED>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT auteur (#PCDATA)>
]>
<?xml-stylesheet type="text/xsl" href="pattern_ex.xsl"?>
<librairie>
  <categorie nom="Histoire">
  <livre ISBN="2724267737">
     <titre>La tragédie Cathare</titre>
     <auteur>Georges Bordonove</auteur>
  </livre>
  <livre ISBN="2744113670">
     <titre>Napoléon T3</titre>
     <auteur>Max Gallo</auteur>
  </livre>
  <livre ISBN="9782286042110">
     <titre>Le Second Empire</titre>
     <auteur>Pierre Miquel</auteur>
  </livre>
  </categorie>
  <categorie nom="Fantastique">
  <livre ISBN="2744102792">
     <titre>La ligne verte</titre>
     <auteur>Stephen King</auteur>
  </livre>
  <livre ISBN="85426951124752">
     <titre>Histoires fantastiques</titre>
     <auteur>Edgard A. Poe</auteur>
  </livre>
  </categorie>
  <categorie nom="Informatique">
  <livre ISBN="9782742916474">
     <titre>HTML 4 & XML</titre>
     <auteur>Ralph Steyer</auteur>
  </livre>
  <livre ISBN="9782736115159">
     <titre>3D Studio 4</titre>
     <auteur>Emmanuel Forsans</auteur>
  </livre>
  <livre ISBN="273612796X">
     <titre>Illustrator 7.0</titre>
     <auteur>Mathieu Lavant</auteur>
  </livre>
  </categorie>
  <categorie nom="Education">
  <livre ISBN="9782708005419">
     <titre>La grammaire anglaise</titre>
     <auteur>S. Berland-Délépine</auteur>
  </livre>
  <livre ISBN="9782091855981">
     <titre>Philosophie Tles F G H</titre>
     <auteur>Gérard Durozoi</auteur>
  </livre>
  </categorie>
</librairie>

<!-- La feuille de style associée -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="ISO-8859-1"/>
  <xsl:template match="/">
  <html>
     <body>
       <ul>
       <xsl:apply-templates select="librairie"/>
       </ul>
```

```
    </body>
  </html>
  </xsl:template>
  <xsl:template match="Pattern1">
  <li><xsl:apply-templates select="Pattern2"/></li>
  </xsl:template>
</xsl:stylesheet>
```

Les *patterns* suivants utilisés dans la feuille de style ci-dessus, permettent d'extraire différentes données XML comme le montre le tableau ci-dessous.

Les deux premiers *patterns* du tableau sont les expressions spécifiées dans le modèle principal de la feuille de style.

Les autres exemples de *patterns* fonctionnent par couple. La première expression *Pattern1* est indiqué sur la première ligne et le second *Pattern2* sur la ligne suivante.

Patterns	Sélection
/	<librairie> ... </librairie>
librairie	<librairie> ... </librairie>
librairie/categorie **sans select="Pattern2"**	<categorie nom="Histoire"> ... </categorie> <categorie nom="Fantastique"> ... </categorie> <categorie nom="Informatique"> ... </categorie> <categorie nom="Education"> ... </categorie>
librairie **categorie[@nom='Informatique']**	<categorie nom="Informatique"> ... </categorie>
librairie/categorie[1] **sans select="Pattern2"**	<categorie nom="Fantastique"> ... </categorie>
librairie/categorie **livre**	<livre ISBN="2724267737"> ... </livre> <livre ISBN="2744113670"> ... </livre> <livre ISBN="..."> ... </livre> <livre ISBN="9782091855981"> ... </livre>
librairie **categorie[3]/*/titre**	<titre>HTML 4 & XML</titre> <titre>3D Studio 4</titre> <titre>Illustrator 7.0</titre>
librairie **categorie/@nom**	nom = "Histoire" nom = "Fantastique" nom = "Informatique" nom = "Education"
librairie/categorie/livre **@ISBN**	ISBN="2724267737" ISBN="2744113670" ISBN="..." ISBN="9782091855981"
librairie/categorie **livre[position()=2]/auteur**	<auteur>Max Gallo</auteur> <auteur>Edgard A. Poe</auteur> <auteur>Emmanuel Forsans</auteur> <auteur>Gérard Durozoi</auteur>

En savoir plus :

W3C Microsoft

5 / Le langage XPath

x

5.1 / Les types de noeuds

Un document XML possède sept types de noeuds différents.

Ces noeuds permettent de **naviguer dans l'arborescence d'un document et surtout de sélectionner** des éléments, attributs, ou tout autre constituant.

Suite à la sélection de ces derniers, **des valeurs, en l'occurrence toujours des chaînes de caractères, peuvent être extraites de ces noeuds**.

Type de noeuds	Description
Racine	représente la valeur de l'instruction de traitement <xml-stylesheet> et l'élément racine.
Elément	représente la valeur de la concaténation de toutes les données caractères analysables trouvées dans l'élément lui-même et tous ces descendants.
Texte	représente le texte contenu dans le noeud courant.
Attribut	représente la valeur de l'attribut du noeud courant.
Espace de noms	représente l'URI (Uniform Resource Identifier) désignant l'espace de noms.
Instruction de traitement	représente la valeur des attributs contenue dans l'instruction.
Commentaire	représente le texte contenu dans le commentaire.

La déclaration XML *<?xml version...>* ainsi que la Définition de Type de Document ne sont pas traitées par le processeur XSL, ainsi elles ne font pas parties de l'arborescence d'un document XML.

Exemple :

```
<xml version="1.0"> <!-- Déclaration ignorée -->
<!-- Déclaration également ignorée -->
<!DOCTYPE element_racine SYSTEM "definition.dtd">
<!-- Le noeud racine comprend les deux premiers noeuds -->
<!-- Noeud instruction de traitement -->
<xml-stylesheet type="text/xsl" href="style.xsl">

 <!-- Premier noeud élément -->
 <element_racine>

 <!-- Second noeud élément -->
 <!-- L'élément contient un noeud attribut -->
 <element_fils attribut="valeur">

 <!-- Troisième noeud élément -->
 <!-- L'élément contient un noeud espace de noms
          ainsi qu'un noeud attribut -->
   <prefixe:element_petit_fils
          xmlns:prefixe="http://www.site.com/namespace/1.0"
          prefixe:attribut="valeur>

    Ce texte contenu dans le troisième noeud élément
    est considéré comme un noeud textuel.

   <!-- Fin du troisième noeud élément -->
   </prefixe:element_petit_fils>

<!-- Fin du second noeud élément -->
</element_fils>

<!-- Fin du premier noeud élément -->
</element_racine>
```

En savoir plus :


W3C


Microsoft

5.2 / Les tests noeuds

Les tests de noeuds permet de préciser le type de noeuds à sélectionner par l'intermédiaire d'un *pattern* d'un élément XSL-T.

Combiné avec des axes nodaux, des prédicats ou encore des fonctions nodales, les tests de noeuds améliorent la puissance de sélection des noeuds d'une arborescence d'un document XML.

Test	Description
comment()	renvoie *true* si un noeud commentaire est trouvé.
text()	renvoie *true* si un noeud textuel est trouvé.
node()	renvoie *true* si un noeud autre qu'un noeud attribut ou un noeud racine est trouvé.
processing-instruction()	renvoie *true* si une instruction de traîtement est trouvé.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
 <xsl:output method="xml" version='1.0'/>
 <xsl:template match="/">
 <logitheque>
    <categorie nom="Editeurs Web">
      <xsl:copy-of
        select="logitheque/categorie[position()=1]/node()[(position()mod 2)=0]"/>
    </categorie>
 </logitheque>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

5.3 / Les axes nodaux

Les axes nodaux permettent de sélectionner des parties de l'arborescence d'un document XML à partir du noeud courant.

En fait, **les axes nodaux ouvrent des directions de recherche** indiquées par des préfixes situé avant le *pattern* et séparé par un double deux-points.

```
préfixe::pattern
```

L'expression suivante sélectionne l'attribut *nom* des noeuds éléments fils directs de l'élément *service*.

```
following-sibling::service/@nom
```

Axe nodal	Description
ancestor::	représente les noeuds parents du noeud courant jusqu'au noeud racine.
ancestor-or-self::	représente le noeud lui-même avec les mêmes caractèristiques que *ancestor*.
descendant::	représente les noeuds fils du noeud courant jusqu'au noeud terminal.
descendant-or-self::	représente le noeud lui-même avec les mêmes caractèristiques que *descendant*.
self::	représente le noeud lui-même.
parent::	représente les noeuds parents directs.
child::	représente les noeuds fils directs.
attribute::	représente les noeuds attributs du noeud courant.
following::	représente tous les noeuds suivant le noeud courant hormis les noeuds attributs et espaces de noms.
following-sibling::	représente la même chose que *following* mais essentiellement les noeuds qui ont le même parent que le noeud courant.
preceding::	représente tous les noeuds précédant le noeud courant hormis les noeuds attributs et espaces de noms.
preceding-sibling::	représente la même chose que *preceding* mais essentiellement les noeuds qui ont le même parent que le noeud courant.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="
coursxsl/axe.xsl"?>
<arriereGrandParent>
  <nom>Arrieres Grands-Parents</nom>
  <GrandOncle>
  <nom>Fils des Arrieres Grands-Parents</nom>
  </GrandOncle>
  <GrandParent>
  <nom>Grand-Parents</nom>
  <Tante>
     <nom>Fille des Grands-Parents</nom>
  </Tante>
  <parent>
     <nom>Parents</nom>
     <Fille>
       <nom>Fille ainée</nom>
     </Fille>
     <Fils>
       <nom>Fils</nom>
       <petitFils>
       <nom>Petit-Fils de Fils</nom>
       </petitFils>
```

```
      <petitFille>
      <nom>Petite-Fille de Fils</nom>
      </petitFille>
    </Fils>
    <Fille>
      <nom>Fille</nom>
      <petitFils>
      <nom>Petit-Fils de Fille</nom>
      </petitFils>
      <petitFille>
      <nom>Petite-Fille de Fille</nom>
      </petitFille>
    </Fille>
  </parent>
  </GrandParent>
</arriereGrandParent>
<!-- Feuille de style : axe.xsl -->
<xsl:template match="Fils">
  <h3>
    <xsl:apply-templates select="nom"/> (<xsl:value-of select="name()"/>)</h3>
  <h4>Tous les ancètres</h4>
  <ul>
    <xsl:for-each select="ancestor-or-self::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
  </ul>
  <h4>Les ancètres</h4>
  <ul>
    <xsl:for-each select="ancestor::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
  </ul>
  <h4>Tous les descendants</h4>
  <ul>
    <xsl:for-each select="descendant-or-self::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
  </ul>
  <h4>Les descendants</h4>
  <ul>
    <xsl:for-each select="descendant::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
  </ul>
  <h4>Les parents</h4>
  <ul>
    <xsl:for-each select="parent::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
  </ul>
  <h4>Le noeud courant</h4>
  <ul>
    <xsl:for-each select="self::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
  </ul>
  <h4>Les enfants</h4>
  <ul>
    <xsl:for-each select="child::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
  </ul>
  <h4>Les suivants directs</h4>
  <ul>
    <xsl:for-each select="following-sibling::*">
```

```
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
</ul>
<h4>Tous les suivants</h4>
<ul>
    <xsl:for-each select="following::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
</ul>
<h4>Les précédents directs</h4>
<ul>
    <xsl:for-each select="preceding-sibling::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
</ul>
<h4>Tous les précédents</h4>
<ul>
    <xsl:for-each select="preceding::*">
      <li>
      <xsl:value-of select="nom"/> (<xsl:value-of select="name()"/>)</li>
    </xsl:for-each>
</ul>
</xsl:template>
```

En savoir plus :

W3C Microsoft

5.3.1 / Les axes nodaux abrégés

Les axes nodaux possèdent des synonymes abrégés permettant de naviguer dans l'arborescence d'un document à l'image des commandes de système de fichiers telles que connaissent UNIX, le DOS ou encore comme les liens relatifs sur Internet.

Par exemple, l'expression suivante est équivalente à la seconde.

parent::personnel

../personnel

Opérateur	Description
element	sélectionne tous les éléments *element* fils du noeud courant (*child::element*).
*	sélectionne tous les éléments fils du noeud courant.
/	représente l'élément racine.
//	représente n'importe quel descendant de l'élément racine, donc tous les éléments (*descendant-or-self::node()*).
.	représente l'élément courant (*self::node()*).
..	permet de remonter d'un niveau dans l'arborescence du document par rapport à l'élément courant (*parent::node()*).
/element	sélectionne tous les éléments *element* sous l'élément racine ().
./element	sélectionne tous les éléments *element* sous l'élément courant (*following::element*).
../element	sélectionne tous les éléments *element* sous l'élément parent du noeud courant (*preceding::element*).
//element	sélectionne tous les éléments *element* descendants du noeud courant à quelque niveau de profondeur que ce soit.
@attribut	sélectionne tous les attributs *attribut* du noeud courant (*attribute::attribut*).
|	correspond à un *ou*.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" media-type="text/xml; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>La logithèque</title>
    </head>
    <body>
      <table border="0" width="60%" class="produit">
      <tr>
        <th>Logiciel</th>
        <th>Lien</th>
      </tr>
      <xsl:apply-templates select="//logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="logiciel">
 <xsl:choose>
    <xsl:when test=".//@lien != "">
```

```
        <xsl:variable name="url" select=".//@lien"/>
        <tr>
        <td class="c1">
          <a href="
coursxsl/{.//@lien}" target="_blank"
             style="font-size:10pt; font-weight:bold">
            <xsl:apply-templates select="./nom"/>
          </a>
        </td>
        <td>
          <xsl:value-of select="$url"/>
        </td>
        </tr>
      </xsl:when>
      <xsl:otherwise>
        <xsl:variable name="url">failed.html</xsl:variable>
        <tr>
        <td class="c1">
          <a href="
coursxsl/failed.html" target="_blank"
             style="font-size:10pt; font-weight:bold">
            <xsl:apply-templates select="./nom"/>
          </a>
        </td>
        <td>
          <xsl:value-of select="$url"/>
        </td>
        </tr>
      </xsl:otherwise>
    </xsl:choose>
    </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

5.4 / Les opérateurs

Les opérateurs sont utilisables dans des expressions XPath permettant de sélectionner des noeuds dans l'arborescence d'un document XML.

Il existe deux sortes d'opérateurs :

- les opérateurs logiques,
- les opérateurs de calcul.

Les premiers renvoient une valeur booléenne *true* ou *false* après avoir effectuer un test entre les deux opérandes.

Les seconds permettent d'accomplir un calcul entre deux nombres et de renvoyer le résultat.

Les opérateurs booléens :

Opérateur	Description
or	représente un OU logique.
and	représente un ET logique.
not()	signifie la négation ou NON logique.
|	permet la sélection de plusieurs motifs.
=	représente l'égalité.
!=	signifie différent de...
<	signifie inférieur à...
<=	signifie inférieur ou égal à...
>	signifie supérieur à...
>=	signifie supérieur ou égal à...

Les opérateurs de calculs :

Opérateur	Description
+	effectue une addition.
-	effectue une soustraction.
div()	effectue une division.
mod()	effectue un modulo.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:output method="xml" version='1.0'/>
  <xsl:template match="/">
  <logitheque>
     <categorie nom="Editeurs Web">
       <xsl:copy-of
         select="logitheque/categorie[position()=1]/node()[(position() mod 2)= 0]"/>
     </categorie>
  </logitheque>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :


W3C


Microsoft

5.5 / Les prédicats

Les prédicats sont des expressions entre crochets permettant de cibler plus précisément une portion de l'arborescence d'un document.

En coopération avec les axes nodaux, les types de noeud ou encore les fonctions nodales, les prédicats assurent un moyen puissant de sélection de noeuds.

L'expression suivante est équivalent à la seconde.

```
noeud[expression]
```

Prédicat	Description
element	sélectionne tous les éléments *element* fils du noeud courant.
element[n]	sélectionne le n^{ième} élément *element* dans le noeud courant.
element[elt]	sélectionne dans le noeud courant, l'élément *element* qui a comme élément fils *elt*.
[elt="valeur"]	sélectionne dans le noeud courant, l'élément ayant pour fils un noeud *elt* qui a une valeur égale à *valeur*.
element[@attribut]	sélectionne dans le noeud courant, l'élément *element* qui possède un attribut *attribut*.
[@attribut='valeur']	sélectionne dans le noeud courant, l'élément dont l'attribut *attribut* a une valeur égale à *valeur*.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
 <html>
    <head>
      <title>
      <xsl:value-of
                  select="/logitheque/categorie[@nom='XML et XSL']/@nom"/>
      </title>
    </head>
    <body>
      <xsl:apply-templates
              select="logitheque/categorie[@nom='XML et XSL']/logiciel"/>
    </body>
 </html>
 </xsl:template>
 <xsl:template
        match="logitheque/categorie[@nom='XML et XSL']/logiciel">
 <h1>
    <xsl:apply-templates select="nom"/>
 </h1>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

5.6 / Les fonctions XPath

Le langage XPath comporte quatre type de fonctions distinctes permettant de travailler sur les noeuds d'une arborescence, sur des chaînes de caractères, sur des valeurs booléennes et enfin sur des nombres.

Ces fonctions peuvent effectuer diverses opérations telles que de **la concaténation sur des chaînes de caractères, des calculs sur des nombres, des évaluations d'expressions**, etc..

Les fonctions sont utilisables non seulement dans une expression XPath mais aussi, directement dans un *pattern* d'un élément XSL.

En savoir plus :

W3C Microsoft

5.6.1 / Les fonctions nodales

Les fonctions nodales renvoyent différentes caractèristiques sur des ensemble de noeuds en général passés en argument.

Ces fonctions ont divers rôles dans l'arborescence d'un document XML comme localiser, positionner, identifier, compter, ou encore dénommer des ensembles (ou jeux) de noeuds.

Ces ensembles de noeuds sont des groupes de noeuds sélectionnés par l'intermédiaire des axes nodaux dans l'arborescence d'un document XML.

Fonction	Description
count(ensemble_noeud)	retourne le nombre de noeud dans l'ensemble de noeuds passé en argument.
current()	retourne le noeud courant.
document(objet, ensemble_noeud)	fournit un chemin pour retrouver d'autres ressources XML à l'intérieur d'une feuille de style de transformation au-delà des données fournies par l'entrée courante.
generate-id(ensemble_noeuds)	retourne une chaîne qui identifie individuellement le premier noeud dans un ensemble de noeuds passé en argument.
id("identifiant")	sélectionne l'élément dans le noeud courant par son identifiant (W3C ou Microsoft).
key(nom, valeur)	retrouve les éléments précédemment marqués par une instruction *xsl:key*.
last()	retourne un nombre égal à la dimension contextextuelle de provenant du contexte d'évaluation de l'expression.
local-name(ensemble_noeud)	retourne la partie locale du nom étendu du noeud dans l'ensemble de noeuds passé en argument qui est le premier dans l'ordre du document.
name(noeuds)	retourne une chaîne de caractères contenant un nom qualifié représentant le nom étendu du noeud dans l'ensemble de noeuds passé en argument qui est le premier dans l'ordre du document.
namespace-uri(ensemble_noeud)	retourne l'URI de l'espace de noms du nom étendu du noeud de l'ensemble de noeuds passé en argument qui est le premier dans l'ordre du document.
node-set(chaîne)	convertit une arborescence à l'intérieur d'un ensemble de noeuds. Le noeud résultant contient toujours un unique noeud et le chemin du noeud de l'arbre.
position()	retourne un nombre représentant la position du noeud courant à l'intérieur du noeud parent.

Exemple [voir]

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"
              media-type="text/html; charset=ISO-8859-1"/>
  <xsl:template match="/">
  <h3>Nombre de categories :
      <xsl:value-of select="count(//categorie)"/>
  </h3>
  <h4> Nombre total de logiciels :
              <xsl:value-of select="count(//logiciel)"/>
  </h4>
  <ul>
     <xsl:apply-templates select="//categorie"/>
  </ul>
  </xsl:template>
  <xsl:template match="categorie">
```

```
    <li>
      <xsl:text>La catégorie "</xsl:text>
      <b>
        <xsl:value-of select="./@nom"/>
      </b>
      <xsl:text>" compte </xsl:text>
      <b>
        <xsl:value-of select="count(logiciel)"/>
      </b>
      <xsl:text> logiciels.</xsl:text>
    </li>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

5.6.2 / Les fonctions booléennes

Le langage XPath utilise cinq fonctions booléennes.

Ces fonctions permettent de travailler sur les valeurs logiques *true* ou *false* afin de **créer des conditions dans les expressions XPath**.

Opérateur	Description
boolean(objet)	convertit l'argument en valeur booléenne.
element-available(chaîne)	retourne la valeur booléenne *true* si et seulement si le nom étendu est le nom d'une instruction.
false()	retourne la valeur logique *false*.
function-available()	retourne *true* si la fonction est disponible dans la librairie de fonctions.
lang(chaîne)	retourne la valeur logique *true* si l'attribut *xml:lang* du noeud contextuel est le même que l'argument.
not(valeur)	retourne le contraire de la valeur booléenne passée en argument.
true()	retourne la valeur logique *true*.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
             media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <body>
      <table>
      <tr>
        <th>Logiciel</th>
        <th>Payant</th>
      </tr>
      <xsl:apply-templates select="logitheque/categorie/logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="logitheque/categorie/logiciel">
 <tr>
    <td>
      <xsl:apply-templates select="nom"/>
    </td>
    <td>
      <xsl:apply-templates select="prix"/>
    </td>
 </tr>
 </xsl:template>
 <xsl:template match="prix">
 <xsl:value-of select="boolean(text())"/>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

5.6.3 / Les fonctions chaînes de caractères

Les fonctions sur les chaînes de caractères permettent d'effectuer différentes opérations sur des chaînes de caractères.

Ces fonctions peuvent effectuer des **concaténations**, des **remplacements**, des **comparaisons** sur des chaînes de caractères ainsi que la s**uppression des espaces blancs** superflus ou encore de **renvoyer le nombre de caractères**.

Opérateur	Description
concat(chaîne1,chaîne2,...)	concatène les chaînes de caractères passées en argument.
contains(contenant,contenu)	retourne *true* si la première chaîne de caractères contient la seconde, sinon renvoie *false*.
normalize-space(chaîne)	retourne l'argument chaîne après suppression des espaces superflus.
starts-with(chaîne,chaîne)	retourne *true* si la première chaîne commence avec les mêmes caractères que la seconde, sinon, elle renvoie *false*.
string(ensemble_noeud)	convertit son argument en chaîne de caractères.
string-length(chaîne)	retourne le nombre de caractères dans la chaîne.
substring(chaîne,position,longueur)	retourne la sous-chaîne du premier argument démarrant à la position et à la longueur spécifiées.
substring-after(chaîne,marqueur)	retourne la sous-chaîne du premier argument résultant de la suppression de tous les caractères précédant le marqueur localisé dans la chaîne.
substring-before(chaîne,marqueur)	retourne la sous-chaîne du premier argument résultant de la suppression de tous les caractères suivant le marqueur localisé dans la chaîne.
system-property(chaîne)	retourne un objet représentant la valeur de la propriété système identifié par un nom.
translate(chaîne,chaîne,chaîne)	retourne le premier argument chaîne dans lequel les occurences des caractères de la deuxième chaîne sont remplacées par les caractères correspondant aux mêmes positions de la troisième chaîne.
unparsed-entity-uri(chaîne)	retourne les déclarations d'entités non-analysées dans la Définition de Type de Document (DTD) de la source du document.

Exemple [voir]

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
  <xsl:template match="//logiciel">
  <html>
     <body>
       <h3>
       <xsl:choose>
         <xsl:when test="contains(nom, '(')">
           <xsl:value-of
                 select="substring-before(nom, '(')"/>
           <xsl:value-of
                 select="concat(' (',@code, ') - ', editeur)"/>
         </xsl:when>
         <xsl:otherwise>
           <xsl:value-of
                 select="concat(nom, ' (',@code, ') - ', editeur)"/>
         </xsl:otherwise>
       </xsl:choose>
       </h3>
     </body>
  </html>
```

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

En savoir plus :

W3C Microsoft

5.6.4 / Les fonctions numériques

Les fonctions numériques permettent d'effectuer différentes opérations sur des nombres.

Ces fonctions peuvent calculer une somme, arrondir ou encore convertir des valeurs passées en argument.

Cela permet de construire des expressions XPath complexes et dynamiques.

Fonction	Description
ceiling(nombre)	retourne le plus petit entier qui n'est pas inférieur au nombre passé en argument.
floor(nombre)	retourne le plus grand entier qui n'est pas supérieur au nombre passé en argument.
format-number(nombre,formatage,décimal)	convertit le premier argument en une chaîne de caractères utilisant le masque de formatage sur le nombre et le cas échéant le format décimal spécifié.
number(objet)	convertit l'argument en un nombre.
round(nombre)	retourne la valeur la plus prôche du nombre passé en argument.
sum(ensemble_noeud)	retourne la somme de tous les noeuds composant l'ensemble de noeuds après que chacun des noeuds ait subit en une valeur numérique.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="sum.xsl"?>
<decompte>
 <categorie>
 <nom>Edition Web</nom>
 <compte>39</compte>
 </categorie>
 <categorie>
 <nom>Dreamweaver</nom>
 <compte>28</compte>
 </categorie>
 <categorie>
 <nom>Frontpage</nom>
 <compte>7</compte>
 </categorie>
 <categorie>
 <nom>Director Shockwave</nom>
 <compte>11</compte>
 </categorie>
 <categorie>
 <nom>Feuille de Style</nom>
 <compte>13</compte>
 </categorie>
 <categorie>
 <nom>XML et XSL</nom>
 <compte>32</compte>
 </categorie>
 <categorie>
 <nom>Base de données</nom>
 <compte>33</compte>
 </categorie>
 <categorie>
 <nom>Java</nom>
 <compte>14</compte>
 </categorie>
 <categorie>
 <nom>Visual Basic</nom>
 <compte>16</compte>
 </categorie>
```

```
  <categorie>
  <nom>Delphi</nom>
  <compte>12</compte>
  </categorie>
  <categorie>
  <nom>Connectivité Web</nom>
  <compte>18</compte>
  </categorie>
  <categorie>
  <nom>Navigateur</nom>
  <compte>7</compte>
  </categorie>
  <categorie>
  <nom>Client FTP</nom>
  <compte>18</compte>
  </categorie>
  <categorie>
  <nom>Recherche</nom>
  <compte>15</compte>
  </categorie>
  <categorie>
  <nom>Serveur Web</nom>
  <compte>22</compte>
  </categorie>
  <categorie>
  <nom>Windows 2000</nom>
  <compte>19</compte>
  </categorie>
  <categorie>
  <nom>Back Office</nom>
  <compte>18</compte>
  </categorie>
  <categorie>
  <nom>Exchange</nom>
  <compte>10</compte>
  </categorie>
  <categorie>
  <nom>Graphisme Web</nom>
  <compte>28</compte>
  </categorie>
  <categorie>
  <nom>Produits Adobe</nom>
  <compte>23</compte>
  </categorie>
</decompte>
<!-- Feuille de style : sum.xsl -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"
              media-type="text/html; charset=ISO-8859-1"/>
  <xsl:template match="/">
  <h3>
     Nombre total de logiciels :
     <xsl:value-of select="sum(//compte)"/>
  </h3>
  <h4>
     (Pour <xsl:value-of select="count(//categorie)"/> catégories)
  </h4>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

6 / Les fonctions XSLT

La syntaxe du langage XPath supporte des fonctions qui fournissent des informations relatives à des noeuds dans une collection.

Ces fonctions retournent des chaînes de caractères ou des nombres et peuvent être utilisées avec des opérateurs de comparaisons dans un *pattern*.

Ces fonctions ne sont pas disponibles à partir des méthodes de sélection de noeuds dans le DOM (Document Object Model).

Fonction	Description
current()	retoune un jeu de noeud qui possède le noeud courant comme son seul membre.
element-available()	retourne la valeur *true* si et seulement si le nom étendu eest le nom d'une instruction.
format-number()	convertit le premier argument d'une chaîne de caractères en une chaîne utilisant le *pattern* de formatage spécifié par le second argument.
function-available()	retourne la valeur *true* si la fonction est disponible dans la librairie de fonctions.
generate-id(jeu_noeuds)	retourne une chaîne de caractères unique qui identifie le noeud cible dans un jeu de noeuds passé en argument.
node-set(jeu_noeuds)	convertit une arborescence à l'intérieur d'un jeu de noeuds. Le noeud résultant contient toujours un unique noeud et la racine de l'arborescence.
system-property()	retourne un objet représentant la valeur de la propriété système identifié par un nom.
unparsed-entity-uri()	retourne les déclarations d'entités non-analysées dans la DTD (document type definition) du document XML source.

Exemple [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
             media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <body>
      <xsl:apply-templates
              select="logitheque/categorie[18]/logiciel"/>
    </body>
 </html>
 </xsl:template>

 <xsl:template match="logiciel">
 <h4><xsl:value-of select="nom"/></h4>
 <p><xsl:value-of select="current()"/></p>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7 / Les éléments XSLT

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Elément	Description
xsl:apply-imports	Invoque un template surchargeant celui d'une feuille de style importée.
xsl:apply-templates	Dirige le processeur XSLT à trouver le template approprié à appliquer, fondé sur le type et le contexte de chaque noeud sélectionné.
xsl:attribute	Crée un attribut et l'associe à un élément.
xsl:attribute-set	Définit un nom à une collection d'atributs.
xsl:call-template	Invoque un template par son nom.
xsl:choose	Combiné avec les éléments <xsl:otherwise> et <xsl:when>, exécute des tests conditionnels multiples.
xsl:comment	Génère un commentaire.
xsl:copy	Recopie le noeud courant dans le document.
xsl:copy-of	Recopie les noeuds sélectionnés par le modèle dans l'arborescence du document.
xsl:decimal-format	Déclare un format de nombre, lequel est utilisé pour la fonction *format-number()*.
xsl:element	Crée un élément avec le nom spécifié dans le document.
xsl:fallback	Appelle un template de substitution si la feuille de style comporte un élément non-défini ou fait appel à une fonction inexistante.
xsl:for-each	Applique un template à plusieurs noeuds répondant au modèle invoqué.
xsl:if	Effectue un test conditionnel sur le modèle sélectionné.
xsl:import	Spécifie une feuille de style XSLT à importer dans la feuille de style courante.
xsl:include	Indique une feuille de style XSLT à inclure dans le document.
xsl:key	Déclare un nom de clé pour une utilisation avec la fonction *key()* dans le langage XPath (XML Chemin).
xsl:message	Envoie un message textuel soit dans un message temporaire soit dans une boîte de dialogue.
xsl:namespace-alias	Crée un alias pour un espace de nom spécifié.
xsl:number	Insère un nombre formaté dans l'arborecence du document.
xsl:otherwise	Combiné avec les éléments <xsl:choose> and <xsl:when>, exécute des tests conditionnels multiples.
xsl:output	Spécifie le format du document généré.
xsl:param	Déclare un nom de paramètre pour être utilisé à l'intérieur des éléments xsl:stylesheet ou xsl:template.
xsl:preserve-space	Préserve les espaces-blancs dans un document.
xsl:processing-instruction	Génère une instruction de traîtement dans un document.
msxsl:script *	Définit les variables et les fonctions globales pour des scripts.
xsl:sort	Permet de trier les noeuds sélectionnés par xsl:for-each ou xsl:apply-

	templates selon les critères spécifiés.
xsl:strip-space	Supprime les espaces blancs entre les marqueurs et les données textuelles.
xsl:stylesheet	Représente l'élément racine d'une feuille de style XSL.
xsl:template	Définit les éléments sur lesquels seront appliqués des règles de style.
xsl:text	Crée un texte dans le document.
xsl:transform	Est un synonyme de xsl:stylesheet.
xsl:value-of	Insère la valeur du noeud sélectionné comme du texte.
xsl:variable	Déclare une variable dans une feuille de style.
xsl:when	Combiné avec les éléments <xsl:choose> and <xsl:otherwise>, exécute des tests conditionnels multiples.
xsl:with-param	Passe un paramètre à un template.

7.1 / L'élément <xsl:apply-imports>

L'élément *<xsl:apply-imports>*, suite à l'importation d'une feuille de sttyle externe, permet d'appliquer ses propres règles de style sur le *template* concerné.

Ainsi, il se produit **une surcharge des mises en forme** sur le *pattern* ciblé.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément ne peut contenir aucunes instructions.

Dans l'exemple ci-dessous, la feuille de style *preformatage.xsl* est importée dans la feuille de style *style.xsl*, laquelle est appelée par le document XML.

L'utilisation de l'élément *<xsl:apply-imports>* provoque la mise en forme des données de l'élément *code*.

Ainsi, l'application des règles de styles s'effectue en premier lieu par la feuille de style importée puis par *style.xsl* qui termine la transformation des données XML.

Exemple :

```
<!-- Feuille de style : preformatage.xsl -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="code">
  <pre><xsl:apply-templates/></pre>
  </xsl:template>
</xsl:stylesheet>
<!-- Feuille de style : style.xsl -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="preformatage.xsl"/>
  <xsl:template match="code">
  <html>
      <body>
        <table align="center">
        <tr>
          <td>
            <xsl:apply-imports/>
          </td>
        </tr>
        </table>
      </body>
  </html>
  </xsl:template>
</xsl:stylesheet>
<!-- Document XML -->
<?xml version="1.0"?>
<!DOCTYPE code [
  <!ELEMENT code (#PCDATA)>
]>
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
<code>
```

```
  ...
</code>
<!-- Résultat -->
<html>
 <body>
 <table align="center">
    <tr>
      <td>
      <pre>...</pre>
      </td>
    </tr>
 </table>
 </body>
</html>
```

En savoir plus :

W3C Microsoft

7.2 / L'élément <apply-templates>

L'élément *<apply-templates>* permet d'appliquer les templates d'une feuille de style sur les fils du noeud courant et les noeuds textuels.

```
<xsl:apply-templates>
  ...
</xsl:apply-templates>
<!-- Elément vide -->
<xsl:apply-templates/>
```

Cependant cet élément possède deux attributs, en l'occurrence *select* et *mode*. S'ils sont utilisés alors seuls les noeuds correspondant à leurs valeurs seront traités.

```
<xsl:apply-templates select="pattern" mode="mode">
```

Attributs

Attribut	Description
select="pattern"	traite les noeuds sélectionnés par le pattern spécifié, au lieu de traiter tous les éléments.
mode="mode"	applique les templates avec un mode donné.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

L'élément *<apply-templates>* ne peut contenir que deux autres éléments. Il s'agît de *<xsl:sort>* permettant de trier des données et *<xsl:with-param>* déclarant un paramètre à passer à un *template*.

```
<xsl:apply-templates select="pattern" mode="mode">
    <xsl:sort.../>
    <xsl:with-param...>...</xsl:with-param>
</xsl:apply-templates>
```

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
             media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
   <body>
    <table border="0" width="60%" class="produit">
    <tr>
      <th colspan="3">Logiciel</th>
      <td/>
    </tr>
    <tr>
```

```
        <th style="text-align:left">Editeur</th>
        <th>Langue</th>
        <th>OS</th>
      </tr>
      <xsl:apply-templates
            select="logitheque/categorie[position()=1]/logiciel"/>
      </table>
    </body>
  </html>
  </xsl:template>
  <xsl:template match="logitheque/categorie[position()=1]/logiciel">
  <tr>
    <td colspan="3" class="c1">
      <a href="{editeur/@lien}" target="_blank"
          style="font-size:10pt; font-weight:bold">
    <xsl:apply-templates select="nom"/>
      </a>
       - <xsl:apply-templates select="commentaire"/>
    </td>
  </tr>
  <tr>
    <td>
      <a href="{editeur/@lien}" target="_blank">
      <xsl:apply-templates select="editeur"/>
      </a>
    </td>
    <td>
      <xsl:apply-templates select="langue"/>
    </td>
    <td>
      <xsl:apply-templates select="plateforme"/>
    </td>
  </tr>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.3 / L'élément <xsl:attribute>

L'élément *<xsl:attribute>* insère un attribut avec le nom indiqué dans l'arborescence d'un document résultant.

```
<xsl:attribute name="nom_attribut">
  Valeur de l'attribut
</xsl:attribute>
```

L'attribut ***name*** **permet d'affecter un nom** à l'attribut créé. Evidemment cette commande est obligatoire.

Un autre attribut, ***namespace*** **permet de désigner un espace de noms**.

```
<xsl:attribute
        name="nom_attribut"
        namespace="URI-référence">
  Valeur de l'attribut
</xsl:attribute>
```

Attributs

Attribut	Description
name="nom"	spécifie un nom pour l'attribut créé.
namespace="URI-référence"	indique un espace de noms.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param

L'élément *<xsl:attribute>* peut contenir les types d'éléments suivants :

- xsl:apply-templates
- xsl:call-template
- xsl:choose
- xsl:copy
- xsl:copy-of
- xsl:for-each
- xsl:if
- xsl:text
- xsl:variable

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" media-type="text/xml; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <xsl:element name="recueil">
    <xsl:attribute name="auteur">Achille Chavée</xsl:attribute>
    <xsl:element name="poeme">
     <xsl:attribute name="titre">Blason d'amour</xsl:attribute>
     <xsl:text>
```

```
        Es-tu plus belle es-tu moins belle
        qu'auparavant
        et qu'importe
        et qu'en sais-je
        je ne ferai jamais du temps
        un attendu secret
        dans le serment de vie qui nous délivre

        Quelques images pures ont brulé ta mémoire
        ô femme
        Dans l'alcôve secrète de la contagion
        j'ai bu très doucement le philtre

        Droite comme un couteau de rose mémorables
        cloué en nous
        en nos deuc coeurs
        en grave fidélité
        affirme chaque jour son évidence blanche
      </xsl:text>
      </xsl:element>
  </xsl:element>
  </xsl:template>
</xsl:stylesheet>


<xsl:template name="illustration">
  <img src="../coursxsl/image.gif">
  <xsl:if test="@align">
     <xsl:attribute name="align">
       <xsl:value-of select="@align"/>
     </xsl:attribute>
  </xsl:if>
  </img>
</xsl:template>
```

Dans ce second exemple, la valeur de l'attribut *align* dépend de la validité du test exécuté par l'élément *<xsl:if>*. Si un attribut *align* est trouvé dans le *template* alors la valeur de l'attribut *align* sera celle récupérée par l'élément *<xsl:value-of>* dans l'élément *illustration*.

En savoir plus :

W3C Microsoft

7.4 / L'élément <xsl:attribute-set>

L'élément *<xsl:attribute-set>* permet de nommer une collection d'attributs crés par l'élément *<xsl:attribute>*.

```
<xsl:attribute-set name="nom_jeu_attributs">
```

Attributs

Attribut	Description
name="nom"	permet de donner un nom au jeu d'attributs créé.

L'utilisation de cette collection d'attributs nécessite l'emploi de **l'attribut *xsl:use-attribute-set*** dans l'élément concerné auquel sera associé les attributs créés.

```
<xsl:attribute-set name="nom">
  Attributs
</xsl:attribute-set>
...
<element xsl:use-attribute-set="nom">
...
```

L'attribut *xsl:use-attribute-set* accepte également une liste de jeux d'attributs séparés par un espace blanc.

```
<element xsl:use-attribute-set="jeu1 jeu2 jeuN">
```

L'élément *<xsl:attribute-set>* ne peut contenir que l'élément *<xsl:attribute>* permettant de créer les attributs composant la collection.

Cet élément peut être contenu que dans les instructions <xsl:stylesheet> ou <xsl:transform>.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" encoding="ISO-8859-1"
             doctype-system="attributeset.dtd"/>
 <xsl:attribute-set name="caracteristique">
 <xsl:attribute name="code">13404148</xsl:attribute>
 <xsl:attribute name="devise">FRF</xsl:attribute>
 <xsl:attribute name="prix">429.00</xsl:attribute>
 <xsl:attribute name="langue">FR</xsl:attribute>
 </xsl:attribute-set>
 <xsl:template match="/">
 <xsl:element name="logitheque">
    <xsl:element name="categorie">
      <xsl:attribute name="nom">Editeurs Web</xsl:attribute>
      <xsl:element name="logiciel" use-attribute-sets="caracteristique">
      <xsl:element name="nom">WebExpert 2000</xsl:element>
      <xsl:element name="editeur">
        <xsl:attribute name="lien">http://www.visic.com/</xsl:attribute>
        Visicom
      </xsl:element>
      </xsl:element>
    </xsl:element>
 </xsl:element>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.5 / L'élément <xsl:call-template>

L'élément *<xsl:call-template>* permet d'appeler un *template* par son nom.

Le nom du *template* sollicité est déclaré par l'intermédiaire d'un attribut *name* d'ailleurs obligatoire dans cet élément.

```
<xsl:call-template name="nom_template">
  ...
</xsl:call-template>
```

Attributs

Attribut	Description
name="nom"	permet d'appeler une règle de modèle par son nom

Cet élément peut être contenu par les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

L'élément *<xsl:call-template>* ne peut contenir que l'élément *<xsl:with-param>* permettant de faire passer des paramètres à un *template*.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
              media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>La logithèque</title>
    </head>
    <body>
      <table border="0" width="60%" class="produit">
      <tr>
        <th>Logiciel</th>
        <th>Lien</th>
      </tr>
      <xsl:apply-templates select="logitheque/categorie/logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template name="cellule">
 <td>
    <xsl:value-of select="commentaire"/>
 </td>
 </xsl:template>
 <xsl:template match="logitheque/categorie/logiciel">
 <xsl:choose>
```

```
    <xsl:when test="editeur/@lien != ''">
      <tr>
      <td class="c1">
        <xsl:variable name="url" select="editeur/@lien"/>
        <a href="{editeur/@lien}" target="_blank"
              style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
        </a>
      </td>
      <xsl:call-template name="cellule"/>
      </tr>
    </xsl:when>
    <xsl:otherwise>
      <tr>
      <td class="c1">
        <xsl:variable name="url">failed.html</xsl:variable>
        <a href="failed.html" target="_blank"
              style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
        </a>
      </td>
      <xsl:call-template name="cellule"/>
      </tr>
    </xsl:otherwise>
  </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.6 / L'élément <xsl:choose>

L'élément *<xsl:choose>* combiné avec *<xsl:when>* et *<xsl:otherwise>*, permet de construire des tests conditionnels à l'instar des commandes *switch* de Java ou Javascript.

```
<xsl:choose>
  <xsl:when test="condition">
    instructions...
  </xsl:when>
  ...
  <xsl:otherwise>
    instructions...
  </xsl:otherwise>
</xsl:choose>
```

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

L'élément *<xsl:choose>* ne peut donc contenir que les élément *<xsl:when>* et *<xsl:otherwise>*.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
             media-type="text/html; charset=ISO-8859-1" version="4.0"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>La logithèque :
               <xsl:value-of select="logitheque/categorie[position()=7]/@nom"/>
      </title>
    </head>
    <body>
      <table border="0" width="60%" class="produit">
      <tr>
        <th>Logiciel</th>
        <th>Lien</th>
      </tr>
      <xsl:apply-templates
                 select="logitheque/categorie[position()=7]/logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="logitheque/categorie[position()=7]/logiciel">
 <xsl:choose>
    <xsl:when test="editeur/@lien != ''">
      <xsl:variable name="url" select="editeur/@lien"/>
      <tr>
      <td class="c1">
        <a href="{editeur/@lien}" target="_blank"
             style="font-size:10pt; font-weight:bold">
```

```
        <xsl:apply-templates select="nom"/>
       </a>
      </td>
      <td>
       <xsl:value-of select="$url"/>
      </td>
      </tr>
     </xsl:when>
     <xsl:otherwise>
      <xsl:variable name="url">failed.html</xsl:variable>
      <tr>
      <td class="c1">
       <a href="failed.html" target="_blank"
            style="font-size:10pt; font-weight:bold">
        <xsl:apply-templates select="nom"/>
       </a>
      </td>
      <td>
       <xsl:value-of select="$url"/>
      </td>
      </tr>
     </xsl:otherwise>
    </xsl:choose>
   </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.7 / L'élément <xsl:comment>

L'élément *<xsl:comment>* permet d'insérer un commentaire dans l'arborescence d'un document XML.

```
<xsl:comment>
  commentaire...
</xsl:comment>
```

Cet élément peut être contenu dans les instructions suivantes :

- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:call-template
- xsl:choose
- xsl:copy
- xsl:copy-of
- xsl:for-each
- xsl:if
- xsl:text
- xsl:value-of
- xsl:variable

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" media-type="text/xml; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>La logithèque</title>
    </head>
    <body>
      <table border="0" width="60%" class="produit">
      <tr>
        <th>Logiciel</th>
        <th>Lien</th>
      </tr>
      <xsl:apply-templates select="/logitheque/categorie/logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="/logitheque/categorie/logiciel">
 <xsl:choose>
    <xsl:when test="editeur/@lien != ''">
      <xsl:variable name="url" select="editeur/@lien"/>
      <tr>
      <td class="c1">
        <a href="{editeur/@lien}" target="_blank"
          style="font-size:10pt; font-weight:bold">
```

```
        <xsl:apply-templates select="nom"/>
       </a>
      </td>
      <comment>
       Ce site est accessible à cette adresse URL !
      </comment>
      <td>
       <xsl:value-of select="$url"/>
      </td>
      </tr>
     </xsl:when>
     <xsl:otherwise>
      <xsl:variable name="url">failed.html</xsl:variable>
      <tr>
      <td class="c1">
       <a href="failed.html" target="_blank"
         style="font-size:10pt; font-weight:bold">
        <xsl:apply-templates select="nom"/>
       </a>
      </td>
      <comment>
       L'adresse URL de ce site est actuellement introuvable !
      </comment>
      <td>
       <xsl:value-of select="$url"/>
      </td>
      </tr>
     </xsl:otherwise>
   </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.8 / L'élément <xsl:copy>

L'élément *<xsl:copy>* est utilisé afin de recopier le noeud courant dans l'arborescence du document XML résultant.

```
<xsl:copy use-attribute-sets="liste">
  template...
</xsl:copy>
```

Cet élément peut être utilisé comme ci-dessus, mais également en tant que balise vide.

```
<xsl:copy/>
```

Les attributs :

Elément	Description
use-attribute-sets="liste"	spécifie une liste d'attributs à ajouter au noeud recopier.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

L'élément *<xsl:copy>* peut contenir les élément suivants :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="* | @* | comment() | text()">
  <xsl:copy>
     <xsl:apply-templates select="* | text()"/>
  </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.9 / L'élément <xsl:copy-of>

L'élément *<xsl:copy-of>* permet de recopier les noeuds sélectionnés par le pattern dans l'arbre du document XML résultant.

```
<xsl:copy-of select="pattern"/>
```

Les attributs :

Elément	Description
select="pattern"	permet de sélectionner par une expression des noeuds.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:variable name="entete">
 <tr>
    <th style="text-align:left">Editeur</th>
    <th>Langue</th>
    <th>OS</th>
 </tr>
 </xsl:variable>
 <xsl:template match="/">
 <html>
    <body>
     <xsl:for-each select="logitheque/categorie/logiciel">
     <h1>
       <xsl:value-of select="nom"/>
     </h1>
     <table border="1">
      <xsl:copy-of select="$entete"/>
      <tr>
        <td>
          <a href="{editeur/@lien}" target="_blank">
            <xsl:value-of select="editeur"/>
          </a>
        </td>
        <td>
          <xsl:value-of select="langue"/>
        </td>
        <td>
          <xsl:value-of select="plateforme"/>
        </td>
      </tr>
```

```
        </table>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.10 / L'élément <xsl:decimal-format>

L'élément *<xsl:decimal-format>* permet de déclarer un format de nombre pour la fonction format-number().

```
<xsl:decimal-format.../>
```

Les attributs :

Elément	Description ("Défaut")
digit="caractère"	indique un caractère représentant un chiffre dans un pattern de format ("#").
zero-digit="caractère"	indique un caractère utilisé pour le zéro ("0")
per-mille="caractère"	indique un signe des pour-mille.
percent="caractère"	spécifie un signe pour les pourcentages ("%").
NaN="chaîne"	spécifie une chaîne de caractère représentant une valeur NaN (Not a Number).
name="nom"	exprime le nom à ce format.
decimal-separator="caractère"	indique le caractère à utiliser entre la partie entière et la partie décimale d'un nombre ("," : FR ou "." : US).
grouping-separator="caractère"	Indique le caractère de séparation entre les groupes de chiffres ("." en France).
infinity="chaîne"	spécifie une chaîne représentant l'infini ("Infinity").
minus-sign="caractère"	indique un caractère représentant le signe moins("-").
pattern-separator="caractère"	indique un caractère représentant le séparateur entre sous-patterns positifs et négatifs (";").

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" encoding="ISO-8859-1"/>
 <xsl:decimal-format name="francais"
                     decimal-separator=","
                     grouping-separator=" "/>
 <xsl:decimal-format name="europeen"
                     decimal-separator=","
                     grouping-separator="."/>
 <xsl:decimal-format name="americain"
                     decimal-separator="."
                     grouping-separator=","/>
 <xsl:template match="/">
 <html><body>
 <table>
   <tr>
    <td>Notation française</td>
    <td>
    <xsl:value-of
        select="format-number(100110429.5, '# ###,00', 'francais')"/>
    </td>
   </tr>
   <tr>
    <td>Notation européenne</td>
    <td>
    <xsl:value-of
```

```
            select="format-number(100110429.5, '#.###,00', 'europeen')"/>
        </td>
      </tr>
      <tr>
        <td>Notation américaine</td>
        <td>
        <xsl:value-of
            select="format-number(100110429.5, '#,###.00', 'americain')"/>
        </td>
      </tr>
    </table>
    </body></html>
    </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.11 / L'élément <xsl:element>

L'élément *<xsl:element>* permet d'insérer d'un marqueur dans l'arborescence d'un document XML résultant.

```
<xsl:element
        name="nom_élément"
        namespace="adresse_URI"
        use-attribute-sets="nom_jeu_attributs">
  ...
</xsl:element>
```

Les attributs :

Elément	Description
name="nom"	permet de donner un nom à l'élément XML créé.
namespace="adresse_URI"	permet de spécifier l'adresse URI d'un espace de noms.
use-attribute-sets="nom"	permet d'utiliser un jeu d'attributs pour l'élément créé.

Cet élément peut être contenu par les instructions suivantes :

- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" encoding="ISO-8859-1"
             doctype-system="logitheque.dtd"/>
 <xsl:attribute-set name="caracteristique">
 <xsl:attribute name="code">13404148</xsl:attribute>
 <xsl:attribute name="devise">FRF</xsl:attribute>
 <xsl:attribute name="prix">429.00</xsl:attribute>
```

```
<xsl:attribute name="langue">FR</xsl:attribute>
</xsl:attribute-set>
<xsl:template match="/">
<xsl:processing-instruction name="xml-stylesheet">
   type="text/xsl" href="style.xsl"
</xsl:processing-instruction>
<xsl:element name="logitheque">
   <xsl:element name="categorie">
     <xsl:attribute name="nom">Editeurs Web</xsl:attribute>
     <xsl:element name="logiciel" use-attribute-sets="caracteristique">
     <xsl:element name="nom">WebExpert 2000</xsl:element>
     <xsl:element name="editeur">
       <xsl:attribute name="lien">http://www.visic.com/</xsl:attribute>
       Visicom
     </xsl:element>
     </xsl:element>
   </xsl:element>
</xsl:element>
</xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.12 / L'élément <xsl:fallback>

L'élément *<xsl:fallback>* permet, en cas de reprise sur un élément d'instructions par le processeur XSLT, d'instancier séquentiellement son contenu, sinon une erreur se produit.

```
<xsl:fallback>
  Instructions...
</xsl:fallback>
```

Cet élément peut être contenu par les instructions suivantes :

- xsl:apply-imports
- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:message
- xsl:number
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable

Cet élément peut contenir les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:message
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
 <html>
    <body>
     <table border="1" width="100%">
    <xsl:include href="entete.xsl">
     <fallback>
     <tr>
       <th colspan="3">Logiciel</th>
     <tr>
     </tr>
       <th>Editeur</th>
       <th>Langue</th>
       <th>Prix</th>
     </tr>
     </fallback>
```

```
    </xsl:include>
      <xsl:apply-templates select="logitheque/categorie/logiciel"/>
      </table>
    <xsl:include href="pieddepage.xsl">
      <fallback>
      <table>
        <tr>
          <td width="100%" align="left">
            Mise à jour du site : 1 septembre 2001
          </td>
        </tr>
      <table>
      </fallback>
    </xsl:include>
    </body>
  </html>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.13 / L'élément <xsl:for-each>

L'élément *<xsl:for-each>* permet d'appliquer des règles de style sur chaque noeud identique d'un *template*.

Les noeuds sont identifiés par un *pattern* spécifié par un attribut *select* d'ailleurs obligatoire.

```
<xsl:for-each select="pattern">
  instructions...
</xsl:for-each>
```

Les attributs :

Attribut	Description
select="pattern"	permet de sélectionner une série de noeuds dans un *template*.

Evidemment, **cet élément ne peut être employé que sur une arborescence dont la structure soit uniforme et connue** tel que par exemple un document XML formé à partir de titres et de paragraphes.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:sort
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
```

```
    <head>
      <title>
      La logithèque : les <xsl:value-of
                  select="/logitheque/categorie[position()=6]/@nom"/>
      </title>
    </head>
    <body>
      <table border="1">
      <tr>
        <th>Logiciel</th>
        <th>Editeur</th>
        <th>Lien</th>
      </tr>
      <xsl:for-each
                  select="/logitheque/categorie[position()=6]/logiciel">
        <tr>
          <td>
            <xsl:value-of select="nom"/>
          </td>
          <td>
            <xsl:value-of select="editeur"/>
          </td>
          <td>
            <xsl:choose>
              <xsl:when test="editeur/@lien != ''">
                <xsl:variable name="url" select="editeur/@lien"/>
                <xsl:value-of select="$url"/>
              </xsl:when>
              <xsl:otherwise>
                <xsl:variable name="url">URL indisponible</xsl:variable>
                <xsl:value-of select="$url"/>
              </xsl:otherwise>
            </xsl:choose>
          </td>
        </tr>
      </xsl:for-each>
      </table>
    </body>
  </html>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.14 / L'élément <xsl:if>

L'élément *<xsl:if>* permet d'appliquer un test conditionnel dans la structure d'une feuille de style XSL.

```
<xsl:if test="condition">
  Instructions...
</xsl:if>
```

Si le test conditionnel est vérifié alors le processeur XSL exécutera les instructions contenues à l'intérieur des marqueurs, sinon il les ignorera et passera aux instructions suivantes.

Les attributs :

Attribut	Description
test="condition"	permet de poser une condtion d'éxécution.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:transform version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"
          media-type="text/html; charset=ISO-8859-1"/>
  <xsl:template match="/">
  <html>
     <head>
       <title>La logithèque</title>
     </head>
```

```
    <body>
      <table border="1">
      <tr>
        <th>N°</th>
        <th>Logiciel</th>
        <th>Langue</th>
      </tr>
      <xsl:apply-templates select="logitheque/categorie/logiciel"/>
      </table>
    </body>
  </html>
  </xsl:template>
  <xsl:template match="logitheque/categorie/logiciel">
  <tr>
    <td>
      <xsl:variable name="num">
      <xsl:number level="any" from="logitheque"/>
      </xsl:variable>
      <xsl:value-of select="$num"/>
    </td>
    <td>
      <a href="{editeur/@lien}" target="_blank"
         style="font-size:10pt; font-weight:bold">
      <xsl:apply-templates select="nom"/>
      </a>
        - <xsl:apply-templates select="commentaire"/>
    </td>
    <td>
      <xsl:if test="langue != 'FR'">
      <img src="../images/interface/drapeau_us.gif"
            alt="Drapeau américain !"/>
      </xsl:if>
      <xsl:if test="langue != 'EN'">
      <img src="../images/interface/drapeau_fr.gif"
            alt="Drapeau français !"/>
      </xsl:if>
    </td>
  </tr>
  </xsl:template>
</xsl:transform>
```

En savoir plus :

W3C Microsoft

7.15 / L'élément <xsl:import>

L'élément *<xsl:import>* permet d'importer les règles de formatage d'une feuille de style externe.

L**a feuille de style externe est appelé au moyen de l'attribut *href*** qui pointe son adresse URI (Uniform Resource Identifier).

```
<xsl:import href="
coursxsl/adresse_URI"/>
```

Cette élément doit impérativement se placer immédiatement après *<xsl:stylesheet>*.

Les attributs :

Attribut	Description
href=*"adresse_URI"*	permet de localiser la feuille de style à importer.

Les règles de correspondance de la feuille de style prévalent sur celles de la feuille importée.

Si plusieurs feuilles de style ont été importées, alors celle l'ayant été le plus récemment prévalera sur toutes les autres.

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple :

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="logitheque.xsl"/>
  <xsl:import href="regle_de_style.xsl"/>
  <xsl:import href="formatage.xsl"/>
  <xsl:template match="/">
  <xsl:apply-templates select="//*"/>
  </xsltemplate>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.16 / L'élément <xsl:include>

L'élément *<xsl:include>* permet d'inclure une feuille de style dans un document.

L'instruction d'inclusion produit l'**insertion des règles de style** dans la feuille de style hôte.

En conséquence, **les règles de style incluses sont d'importances équivalentes** à celles existantes dans le document de référence.

```
<xsl:include href="
coursxsl/Adresse"/>
```

Les attributs :

Elément	Description
href=" coursxsl/Adresse"	indique l'adresse de la feuille de style à inclure.

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple :

```
<!-- Feuille de style hôte -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <body>
      <table border="0" width="60%" class="produit">
      <tr>
        <th>Logiciel</th>
        <th>Lien</th>
      </tr>
      <xsl:apply-templates select="/logitheque/logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template name="cellule">
 <td>
    <xsl:value-of select="commentaire"/>
 </td>
 </xsl:template>
 <xsl:include href="logiciel.xsl"/>
</xsl:stylesheet>
<!-- Feuille de style incluse : logiciel.xsl -->
 <xsl:template match="/logitheque/logiciel">
 <xsl:choose>
    <xsl:when test="editeur/@lien != "">
      <tr>
      <td class="c1">
        <xsl:variable name="url" select="editeur/@lien"/>
        <a href="{editeur/@lien}" target="_blank"
            style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
        </a>
      </td>
      <xsl:call-template name="cellule"/>
      </tr>
    </xsl:when>
    <xsl:otherwise>
      <tr>
      <td class="c1">
        <xsl:variable name="url">failed.html</xsl:variable>
```

```
        <a href="failed.html" target="_blank"
            style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
        </a>
      </td>
      <xsl:call-template name="cellule"/>
      </tr>
    </xsl:otherwise>
</xsl:choose>
</xsl:template>
```

En savoir plus :

7.17 / L'élément <xsl:key>

L'élément *<xsl:key>* permet de créer un identifiant pour des éléments sélectionnés par un *pattern* et désigné par l'expression de l'attribut *use*.

```
<xsl:key
        name="nom"
        match="pattern"
        use="expression"/>
```

L'attribut *use* permet de **faire une différenciation entre les éléments sélectionnés** par l'attribut *match*.

La valeur extraite par l'attribut *use* crée **un identifiant à deux niveaux**.

```
key("nom_identifiant", "valeur_use")
```

La référence à ce type d'identifiant s'effectue par la fonction *key*.

Les attributs :

Elément	Description
name="nom"	affecte un nom à l'identifiant créé.
match="pattern"	sélectionne un type de noeud dans l'arborescence d'un document XML.
use="expression"	désigne une valeur de différenciation.

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" version="1.0" encoding="ISO-8859-1"/>
 <xsl:key name="logiciel" match="logiciel" use="editeur"/>
 <xsl:param name="editeur" select="'Microsoft'"/>
 <xsl:template match="/">
 <xsl:element name="logitheque">
    <xsl:copy-of select="key('logiciel', $editeur)"/>
 </xsl:element>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.18 / L'élément <xsl:message>

L'élément *<xsl:message>* permet d'afficher un message contenu en son sein.

```
<xsl:message terminate="yes|no">
  Message...
</xsl:message>
```

En fonction du comportement du processeur XSL, le message pourra être affiché dans une boîte de dialogue notamment.

Si la valeur de l'attribut *terminate* est *yes* alors le traîtement du fichier XML sera interrompu en affichant le message.

Les attributs :

Elément	Description
terminate="yes|no"	interrompt le traîtement du fichier si la valeur est *yes* sinon le traîtement continue.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:apply-imports
- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:fallback
- xsl:for-each
- xsl:if
- xsl:message
- xsl:number
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable

Cet élément peut contenir les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:fallback
- xsl:for-each
- xsl:if
- xsl:message
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" encoding="ISO-8859-1"/>
```

```
<xsl:variable name="num" select="7"/>
<xsl:template match="/">
<html>
   <body>
    <xsl:value-of
          select="logitheque/categorie/logiciel[$num]/nom"/>
    <br/>
    <xsl:value-of
          select="logitheque/categorie/logiciel[$num]/editeur"/>
    <br/>
    <xsl:choose>
    <xsl:when
         test="logitheque/categorie/logiciel[$num]/editeur/@lien != ''">
     <xsl:value-of
           select="logitheque/categorie/logiciel[$num]/editeur/@lien"/>
    </xsl:when>
    <xsl:otherwise>
     <xsl:message terminate="yes">
       <xsl:text>
        L'élément appelé ne contient pas d'URL !
       </xsl:text>
     </xsl:message>
    </xsl:otherwise>
    </xsl:choose>
   </body>
</html>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.19 / L'élément <xsl:namespace-alias>

L'élément *<xsl:namespace-alias>* permet la création d'un alias pour un espace de noms.

```
<xsl:namespace-alias
        stylesheet-prefix="préfixe|#default"
        result-prefix="préfixe|#default"/>
```

En fait, **le préfixe spécifié par le premier attribut *stylesheet-prefix* sera remplacé par le préfixe du second attribut *result-prefix*** dans le document résultant.

La valeur *#default* désigne l'espace de noms par défaut.

Les attributs :

Elément	Description
stylesheet-prefix="préfixe|#default"	désigne le préfixe d'espace de noms à remplacer.
result-prefix="préfixe|#default"	désigne le préfixe d'espace de noms de substitution.

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format/1.0"
                version="1.0">
<xsl:namespace-alias stylesheet-prefix="fo" result-prefix="xsl"/>
<xsl:param name="service" select="'logitheque'"/>
  <xsl:template match="/">
   <fo:stylesheet>
   <xsl:choose>
       <xsl:when test="$service='logitheque'">
       <fo:import href="logitheque.xsl"/>
       <fo:template match="/">
          <fo:include href="entete.xsl"/>
          <fo:call-template name="affichage"/>
          <fo:include href="pieddepage.xsl"/>
       </fo:template>
       </xsl:when>
       <xsl:when test="$service='librairie'">
       <fo:import href="librairie.xsl"/>
       <fo:template match="/">
          <fo:include href="
coursxsl/entete.xsl"/>
          <fo:call-template name="affichage"/>
          <fo:include href="pieddepage.xsl"/>
       </fo:template>
       </xsl:when>
       <xsl:otherwise>
       <fo:import href="autre.xsl"/>
       <fo:template match="/">
          <fo:include href="entete.xsl"/>
          <fo:call-template name="affichage"/>
          <fo:include href="pieddepage.xsl"/>
       </fo:template>
       </xsl:otherwise>
   </xsl:choose>
   </fo:stylesheet>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :


W3C


Microsoft

7.20 / L'élément <xsl:number>

L'élément *<xsl:number>* permet l'insertion d'un nombre dans l'arborescence d'un document résultant.

```
<xsl:number
        level="single|multiple|any"
        count="pattern"
        from="pattern"
        value="expression"
        format="chaîne_de_caractères"
        lang="langue"
        letter-value="alphabetic|traditional"
        grouping-separator="caractère"
        grouping-size="nombre"/>
```

Les attributs :

Elément	Description
level="single|multiple|any"	indique les niveaux de numérotations : unique (1 (1, n), 2 (1, n), n (1, n)), multiple (1.1.1, 1.1.2, n.n.n), ou (1, 2, n).
count="pattern"	sélectionne les éléments à numéroter.
from="pattern"	indique quand commence la numérotation.
value="expression"	spécifie une valeur permettant de commencer la numérotation.
format="chaîne_de_caractères"	indique une chaîne de caractères à utiliser pour la numérotation comme I.1 (I : chiffres romain, 1 chiffres arabes).
lang="langue"	indique la langue à utiliser dans les numérotations alphabétiques.
letter-value="alphabetic|traditional"	désigne une numérotation alphabétique ou traditionnelle.
grouping-separator="caractère"	indique le caractère de séparation entre les groupes de chiffres.
grouping-size="nombre"	spécifie la taille des groupes de chiffres

Cet élément peut être contenu dans les instructions suivantes :

- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
  <xsl:template match="/">
  <html>
     <head>
       <title>La logithèque</title>
     </head>
     <body>
```

```
      <table border="1">
      <tr>
        <th>N°</th>
        <th>Logiciel</th>
      </tr>
      <xsl:apply-templates select="logitheque/categorie/logiciel"/>
      </table>
    </body>
  </html>
  </xsl:template>
  <xsl:template match="logitheque/categorie/logiciel">
  <tr>
    <td>
      <xsl:variable name="num">
      <xsl:number level="any" from="logitheque"/>
      </xsl:variable>
      <xsl:value-of select="$num"/>
    </td>
    <td>
      <xsl:apply-templates select="nom"/>
        - <xsl:apply-templates select="commentaire"/>
    </td>
  </tr>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.21 / L'élément <xsl:otherwise>

L'élément *<xsl:otherwise>* désigne une expression conditionnelle par défaut au sein d'un élément *<xsl:choose>*.

```
<xsl:otherwise>
  Instructions...
</xsl:otherwise>
```

Cet élément, utilisé conjointement avec les éléments *<xsl:choose>* et *<xsl:when>*, constitue donc **la dernière alternative possible dans une structure conditionnelle**.

Cet élément ne peut être contenu que par l'instruction <xsl:choose>.

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
             media-type="text/html; charset=ISO-8859-1" version="4.0"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>La logithèque :
               <xsl:value-of select="logitheque/categorie[position()=7]/@nom"/>
      </title>
    </head>
    <body>
      <table border="0" width="60%" class="produit">
      <tr>
        <th>Logiciel</th>
        <th>Lien</th>
      </tr>
      <xsl:apply-templates
                 select="logitheque/categorie[position()=7]/logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="logitheque/categorie[position()=7]/logiciel">
 <xsl:choose>
    <xsl:when test="editeur/@lien != ''">
      <xsl:variable name="url" select="editeur/@lien"/>
      <tr>
      <td class="c1">
        <a href="{editeur/@lien}" target="_blank"
             style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
```

```
      </a>
     </td>
     <td>
      <xsl:value-of select="$url"/>
     </td>
     </tr>
    </xsl:when>
    <xsl:otherwise>
     <xsl:variable name="url">failed.html</xsl:variable>
     <tr>
     <td class="c1">
      <a href="failed.html" target="_blank"
          style="font-size:10pt; font-weight:bold">
       <xsl:apply-templates select="nom"/>
      </a>
     </td>
     <td>
      <xsl:value-of select="$url"/>
     </td>
     </tr>
    </xsl:otherwise>
   </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.22 / L'élément <xsl:output>

L'élément *<xsl:output>* indique le format de sortie du document résultant.

```
<xsl:output
        method="xml|html|text|nom"
        version="version"
        encoding="type_encodage"
        omit-xml-declaration="yes|no"
        standalone="yes|no"
        doctype-public="identifiant"
        doctype-system="identifiant"
        cdata-section-elements="nom"
        indent="yes|no"
        media-type="type"/>
```

Cet élément est à placer immédiatement après l'élément *<xsl:stylesheet>*.

Les attributs :

Elément	Description
method="xml|html|text"	indique le format du document résultant.
version="version"	spécifie la version du format utilisé.
encoding="type_encodage"	désigne le type d'encodage des caractères.
omit-xml-declaration="yes|no"	précise si le document doit écrire une déclaration XML dans le document résultant.
standalone="yes|no"	contrôle la valeur de l'attribut du même nom dans la déclaration XML générée.
doctype-public="identifiant"	spécifie un identifiant public du type du document résultant.
doctype-system="identifiant"	spécifie un identifiant système du type du document résultant.
cdata-section-elements="nom"	spécifie la liste des éléments dont
indent="yes|no"	précise si le document résultant doit être indenté.
media-type="type"	spécifie le type MIME du document résultant.

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml"
             encoding="ISO-8859-1"
             doctype-public="oeuvre.dtd"/>
 <xsl:template match="/">
 <oeuvre>
    <xsl:apply-templates/>
 </oeuvre>
 </xsl:template>
 <xsl:template match="poesie">
 <nom>
     <xsl:value-of select="titre"/>
 </nom>
     <poeme>
     <xsl:value-of select="texte"/>
     </poeme>
   <poete>
     <xsl:value-of select="auteur"/>
```

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

En savoir plus :

W3C Microsoft

7.23 / L'élément <xsl:param>

L'élément *<xsl:param>* permet de déclarer un paramètre utilisable par une règle de modèle *<xsl:template>*.

```
<xsl:param name="nom">
  Valeur...
</xsl:param>
```

La valeur du paramètre se trouve au sein des marqueurs.

Si l'attribut *select* est employé, alors l'élément *<xsl:param>* devra être un élément vide.

```
<xsl:param name="nom" select="expression"/>
```

Les attributs :

Elément	Description
name="nom"	affecte un nom au paramètre.
select="expression"	sélectionne la valeur du paramètre dans l'arborescence d'un document XML.

Dans le cas où le paramètre est déclaré dans l'élément *<xsl:stylesheet>*, alors il serait valable dans l'ensemble de la feuille.

Déclarée dans une règle de modèle *<xsl:template>*, le paramètre a une portée locale.

Un paramètre est appelé par l'intermédiaire de son nom qui doit être précédé du signe dollar (*$*).

Deux éléments *<xsl:param>* ne peuvent se faire mutuellement référence.

Les paramètres doivent se trouver au sommet de la règle de modèle *<xsl:template>*.

Les paramètres sont souvent utilisés comme récepteur d'une valeur provenant d'un programme externe.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:stylesheet
- xsl:template
- xsl:transform

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" encoding="ISO-8859-1" indent="no"/>
 <xsl:template match="/">
 <html>
    <body>
     <table border="1">
     <tr>
       <th>Nom</th>
       <th>Position</th>
       <th>Profondeur</th>
       <th>Données contenues</th>
     </tr>
     <xsl:for-each select="//*">
       <tr>
         <td>
           <xsl:value-of select="name()"/>
         </td>
         <td>
           <xsl:value-of select="position()"/>
         </td>
         <td>
           <xsl:call-template name="profondeur"/>
         </td>
         <td><pre>
           <xsl:value-of select="current()"/>
         </pre></td>
       </tr>
     </xsl:for-each>
     </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template name="profondeur">
 <xsl:param name="noeud" select="."/>
 <xsl:value-of select="count($noeud/ancestor::node())"/>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.24 / L'élément <xsl:preserve-space>

L'élément *<xsl:preserve-space>* permet de préserver les espaces blancs tels quels pour des éléments énumérés dans un liste.

```
<xsl:preserve-space elements="liste_éléments"/>
```

Les attributs :

Elément	Description
elements="liste_éléments"	déclare une liste d'éléments séparés par un espace blancs.

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
          media-type="text/html; charset=ISO-8859-1"/>
 <xsl:preserve-space elements="titre texte"/>
 <xsl:template match="/">
 <html>
    <body>
    <xsl:apply-templates/>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="poesie">
 <h2>
    <xsl:value-of select="titre"/>
 </h2>
 <pre>
    <xsl:value-of select="texte"/>
 </pre>
 <h3>
    <xsl:value-of select="auteur"/>
 </h3>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.25 / L'élément <xsl:processing-instruction>

L'élément *<xsl:processing-instruction>* permet de créer une instruction de traîtement dans un document XML résultant.

```
<xsl:processing-instruction name="nom">
  Instructions...
</xsl:processing-instruction>
```

Les instructions sont en fait des attributs avec leurs valeurs qui apparaîtront dans l'instruction de traîtement.

```
<?nom_instruction attributs="valeur"?>
```

Les attributs :

Elément	Description
name="nom"	affecte un nom à l'instruction de traîtement.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

L'élément *<xsl:processing-instruction>* peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:call-template
- xsl:choose
- xsl:copy
- xsl:copy-of
- xsl:for-each
- xsl:if
- xsl:text
- xsl:value-of
- xsl:variable

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" encoding="ISO-8859-1"
             doctype-system="logitheque.dtd"/>
 <xsl:attribute-set name="caracteristique">
 <xsl:attribute name="code">13404148</xsl:attribute>
 <xsl:attribute name="devise">FRF</xsl:attribute>
 <xsl:attribute name="prix">429.00</xsl:attribute>
 <xsl:attribute name="langue">FR</xsl:attribute>
 </xsl:attribute-set>
 <xsl:template match="/">
```

```
<xsl:processing-instruction name="xml-stylesheet">
   type="text/xsl" href="style.xsl"
</xsl:processing-instruction>
<xsl:element name="logitheque">
   <xsl:element name="categorie">
     <xsl:attribute name="nom">Editeurs Web</xsl:attribute>
     <xsl:element name="logiciel" use-attribute-sets="caracteristique">
     <xsl:element name="nom">WebExpert 2000</xsl:element>
     <xsl:element name="editeur">
       <xsl:attribute name="lien">http://www.visic.com/</xsl:attribute>
         Visicom
       </xsl:element>
     </xsl:element>
   </xsl:element>
</xsl:element>
</xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.26 / L'élément <msxsl:script>

L'élément *<msxsl:script>* permet d'insérer un script dans une feuille de style.

```
<msxsl:script
              language="nom_langage"
              implements-prefix="préfixe">
  Script...
</msxsl:script>
```

Cette commande doit être placée aussitôt après l'élément *<xsl:stylesheet>*.

Les attributs :

Attribut	Description
language="nom_langage"	spécifie le nom du langage utilisé dans le script, à défaut la valeur est *JScript*.
implements-prefix="préfixe"	déclare un préfixe d'espace de noms et l'associe au script.

A l'intérieur de l'élément *<msxsl:script>* peuvent être déclarées des variables et des fonctions qui seront accessibles par l'intermédiaire du préfixe d'espace de noms dans l'ensemble de la feuile de style.

L'élément *<xsl:value-of>*, à l'intérieur d'une règle de modèle *<xsl:template>* peut appeler une fonction dans un script.

```
<xsl:template match="/">
  <xsl:value-of select="préfixe:nom_fonction(arguments)"/>
</xsl:template>
```

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions hormis évidemment un script.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                xmlns:log="
namespace">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <msxsl:script language="VBScript" implements-prefix="log">
 Function UpperCase(noeud)
    UpperCase = UCase(noeud)
 End Function
 </msxsl:script>
 <xsl:template match="/">
 <html>
   <head>
    <title>La logithèque</title>
   </head>
   <body>
    <table border="1">
    <tr>
     <th>Logiciel</th>
     <th>Editeur</th>
    </tr>
    <xsl:for-each select="/logitheque/categorie/logiciel">
     <tr>
      <td>
       <xsl:value-of select="log:UpperCase(string(nom))"/>
      </td>
      <td>
       <xsl:value-of select="log:UpperCase(string(editeur))"/>
      </td>
```

```
        </tr>
      </xsl:for-each>
      </table>
    </body>
  </html>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

Microsoft

7.27 / L'élément <xsl:sort>

L'élément *<xsl:sort>* permet d'effectuer un tri selon des critères spécifiés.

```
<xsl:sort
        select="pattern"
        lang="langue"
        data-type="text|number|nom"
        order="ascending|descending"
        case-order="upper-first|lower-first"/>
```

Cet élément ne peut apparaître que dans les éléments *<xsl:for-each>* ou *<xsl:apply-templates>*.

L'élément *<xsl:sort>* doit être contenu dans une règle de modèle *<xsl:template>* qui pointe un des éléments parents de la clé de tri sélectionné par *<xsl:sort select="clé_de_tri">*.

Le noeud pointé par cette clé sera traîté dans un autre *template* qui appliquera le tri.

```
<xsl:template match="élément_parent">
  <xsl:apply-templates>
    <xsl:sort select="clé_de_tri"/>
      </xsl:apply-templates>
</xsl:template>
<xsl:template match="élément_fils">
  <xsl:apply-templates select="clé_de_tri"/>
</xsl:template>
```

Les attributs :

Attribut	Description
select="pattern"	sélectionne des noeuds dans une arborescence source.
lang="langue"	spécifie un bigramme désignant une langue.
data-type="text|number|texte"	indique le type de tri à effectuer
order="ascending|descending"	spécifie un ordre de tri, soit descendant, soit ascendant.
case-order="upper-first|lower-first"	indique si les majuscules doivent apparaître en premier.

Cet élément peut être contenu dans les instructions <xsl:apply-templates> et <xsl:for-each>.

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>
      La logithèque :
            <xsl:value-of select="logitheque/categorie[position()=8]/@nom"/>
      </title>
    </head>
    <body>
      <h2>
      <xsl:value-of
              select="logitheque/categorie[position()=8]/@nom"/>
      </h2>
      <table border="0">
      <tr>
        <th colspan="2">Logiciel</th>
```

```
      </tr>
      <tr>
        <th style="text-align:left">Editeur</th>
        <th>Langue</th>
      </tr>
      <xsl:apply-templates
                  select="logitheque/categorie[position()=8]/logiciel">
        <xsl:sort select="nom"/>
      </xsl:apply-templates>
      </table>
    </body>
</html>
</xsl:template>
<xsl:template match="logitheque/categorie[position()=8]/logiciel">
<tr>
    <td colspan="2" class="c1">
        <a href="{editeur/@lien}" target="_blank"
             style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
        </a>
        - <xsl:apply-templates select="commentaire"/>
    </td>
</tr>
<tr>
    <td>
      <a href="{editeur/@lien}" target="_blank">
      <xsl:apply-templates select="editeur"/>
      </a>
    </td>
    <td>
      <xsl:apply-templates select="langue"/>
    </td>
</tr>
</xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.28 / L'élément <xsl:strip-space>

L'élément *<xsl:strip-space>* permet la suppression des espaces blancs superflus à l'intérieur d'éléments.

```
<xsl:strip-space elements="liste_éléments"/>
```

Dans une liste sont énumérés des noms d'éléments, séparés par des espaces blancs.

Les attributs :

Attribut	Description
elements="liste_éléments"	spécifie une liste de noms d'éléments.

Cet élément peut être contenu dans les instructions <xsl:stylesheet> et <xsl:transform>.

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
          media-type="text/html; charset=ISO-8859-1"/>
 <xsl:strip-space elements="titre texte auteur"/>
 <xsl:template match="/">
 <html>
    <body>
      <xsl:apply-templates/>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="poesie">
 <h2>
    <xsl:value-of select="titre"/>
 </h2>
 <pre>
    <xsl:value-of select="texte"/>
 </pre>
 <h3>
    <xsl:value-of select="auteur"/>
 </h3>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.29 / L'élément <xsl:stylesheet> et <xsl:transform>

L'instruction *<xsl:stylesheet>* est l'élément racine des feuilles de style.

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

Un synonyme existe pour cet élément racine, il s'agit de *<xsl:transform>*. Ce dernier possède la même fonction et les mêmes attributs.

```
<xsl:transform...>
  ...
</xsl:transform>
```

Par convention, *<xsl:transform>* peut être utilisé pour les feuilles de style de transformation et *<xsl:stylesheet>* pour celles de formatage. Mais cela n'a aucun caractère obligatoire.

Cet élément définit **la version du langage XSL** utilisé.

```
<xsl:stylesheet version="1.0">
```

L'espace de noms est également déclaré dans cet élément.
D'ailleurs, plusieurs espaces de noms peuvent être indiqués dans l'élément *<xsl:stylesheet>*.

```
<xsl:stylesheet
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:html="http://www.w3.org/TR/REC-html40">
  ...
</xsl:stylesheet>
```

Cet élément, étant placé au plus haut niveau hiérarchique, ne peut être contenu dans une autre instruction.

En conséquence, l'élément racine peut contenir des éléments dont l'espace de noms est différent.

L'élément *<xsl:stylesheet>* peut contenir les types d'éléments suivants :

- xsl:import
- xsl:include
- xsl:strip-space
- xsl:preserve-space
- xsl:output
- xsl:key
- xsl:decimal-format
- xsl:namespace-alias
- xsl:attribute-set
- xsl:variable
- xsl:param
- xsl:template
- msxsl:script

L'ordre d'apparition des éléments est indifférent ,hormis pour *<xsl:import>* qui doit impérativement se trouver immédiatement après l'élément *<xs:stylesheet>*.

Les attributs

Attribut	Description
id=*identificateur*	permet de repérer la feuille de style par un identificateur.
extension-element-prefixes="*liste*"*	représente une liste de préfixes d'espace de noms.

exclude-result-prefixes="*liste*"*	exclut une liste de préfixes d'espaces de noms.
version="*number*"	indique la version du langage XSL utilisé.
xmlns:*prefixe*="*URI*"	spécifie le ou les préfixes d'espace de noms à utiliser dans la feuille de style.

* Une liste de préfixes comme *html:* ou *xhtml:* séparés par des espaces blancs.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:param name="valeur">XML et XSL</xsl:param>
 <xsl:template match="/">
 <html>
    <head>
     <title>
     La logithèque :
        la catégorie <xsl:value-of select="$valeur"/>
     </title>
    </head>
    <body>
     <h2>
     La logithèque :
        la catégorie <xsl:value-of select="$valeur"/>
     </h2>
     <xsl:for-each
              select="logitheque/categorie[@nom=$valeur]/logiciel">
      <xsl:variable name="url" select="editeur/@lien"/>
      <h3>
        <xsl:value-of select="nom"/>
          (<xsl:value-of select="langue"/>)
      </h3>
      <p><xsl:value-of select="commentaire"/></p>
      <h4>
        <a href="{$url}"><xsl:value-of select="editeur"/></a>
      </h4>
      <u>Prix : </u><br/>
      <p>
        <xsl:value-of select="prix"/>
          <xsl:value-of select="prix/@monnaie"/>
      </p>
     </xsl:for-each>
    </body>
 </html>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.30 / L'élément <xsl:template>

L'élément *<xsl:template>* permet de définir une règle de modèle, c'est-à-dire, par l'intermédiaire de la valeur de l'attribut *match*, des éléments seront prêts pour l'application de règles de style.

```
<xsl:template match="pattern">
  ...
</xsl:template>
```

Il suffira ensuite d'**utiliser l'élément *<xsl:apply-templates>*** en son sein, afin d'effectuer la mise en forme des éléments ciblés.

```
<xsl:template match="pattern">
  ...
  <xsl:apply-templates select="cible"/>
  ...
</xsl:template>
```

Par ailleurs, **si un mode est indiqué dans *<xsl:apply-template>***, alors seuls les éléments correspondants à la fois au *pattern* du *template* et à ce mode, subiront une mise en forme.

```
<xsl:template match="pattern">
  ...
  <xsl:apply-templates mode="nom"/>
  ...
</xsl:template>
```

Enfin, **l'élément *<xsl:value-of>*** utilisé dans une règle de modèle permet de récupérer les valeurs des éléments ciblés ou de leurs attributs.

```
<xsl:template match="pattern">
  ...
  <xsl:value-of select="expression"/>
  ...
</xsl:template>
```

Si l'attribut *name* est utilisé dans le *template*, ce-dernier se comportera comme une fonction.

```
<xsl:template name="nom">
  ...
</xsl:template>
```

La fonction générée est appelée explicitement par son nom avec l'élément *<xsl:call-template>*.

```
<xsl:template name="nom_template">
  ...
</xsl:template>
<xsl:template match="pattern">
  ...
  <xsl:call-template name="nom_template"/>
  ...
</xsl:template>
```

Outre les règles de style contenues dans le *template* appelé, **des paramètres peuvent également être passés** par l'intermédiaire des éléments *<xsl:param>* dans la fonction template et *<xsl:with-param>* dans *<xsl:call-template>*.

```
<xsl:template name="nom_template">
  <xsl:param name="nom_paramètre">
  Valeur
  </xsl:param>
  ...
</xsl:template>
<xsl:template match="pattern">
```

```
...
<xsl:call-template name="nom_template">
<xsl:with-param name="nom_paramètre">
  Nouvelle valeur
<xsl:/with-param>
</xsl:call-template>
...
</xsl:template>
```

Les attributs :

Attribut	Description
match="pattern"	traite les noeuds sélectionnés par le pattern spécifié, au lieu de traiter tous les éléments.
name="nom"	traite les noeuds sélectionnés par le pattern spécifié, au lieu de traiter tous les éléments.
priority="niveau"	applique les templates avec un mode donné.
mode="mode"	applique les templates avec un mode donné.

Cet élément ne peut être contenu que dans les instructions <xsl:stylesheet> et <xsl:transform>.

L'élément *<xsl:template>* accepte en son sein les éléments suivants :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:param
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
          media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
   <head>
    <title>La logithèque</title>
   </head>
   <body>
    <table border="0" width="60%" class="produit">
    <tr>
      <th>Logiciel</th>
      <th>Editeur</th>
    </tr>
    <xsl:apply-templates select="logitheque/categorie/logiciel"/>
    </table>
   </body>
 </html>
 </xsl:template>
 <xsl:template match="logitheque/categorie/logiciel">
 <tr>
```

```
        <td>
          <a href="{editeur/@lien}" target="_blank"
              style="font-size:10pt; font-weight:bold">
            <xsl:apply-templates select="nom"/>
          </a>
          - <xsl:apply-templates select="commentaire"/>
        </td>
        <td>
          <a href="{editeur/@lien}" target="_blank">
          <xsl:apply-templates select="editeur"/>
          </a>
        </td>
      </tr>
      </xsl:template>
    </xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.31 / L'élément <xsl:text>

L'élément *<xsl:text>* permet d'insérer des données textuelles dans un document XML résultant.

```
<xsl:text disable-output-escaping="yes|no">
  Données textuelles...
</xsl:text>
```

Les attributs :

Attribut	Description
disable-output-escaping="yes|no"	active ou désactive le remplacement des caractères spéciaux par leur entité XML.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément ne peut contenir que des données textuelles.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
             media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>La logithèque</title>
    </head>
    <body>
      <table border="1">
      <tr>
        <th>N°</th>
        <th>Logiciel</th>
        <th>Version</th>
      </tr>
      <xsl:apply-templates
                    select="logitheque/categorie/logiciel"/>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="logitheque/categorie/logiciel">
 <tr>
    <td>
      <xsl:variable name="num">
      <xsl:number level="any" from="logitheque"/>
      </xsl:variable>
      <xsl:value-of select="$num"/>
    </td>
    <td>
```

```
        <a href="{editeur/@lien}" target="_blank"
           style="font-size:10pt; font-weight:bold">
        <xsl:apply-templates select="nom"/>
        </a>
          - <xsl:apply-templates select="commentaire"/>
      </td>
    <td>
    <xsl:if test="langue != 'FR'">
       <xsl:text>Langue anglaise</xsl:text>
    </xsl:if>
    <xsl:if test="langue != 'EN'">
       <xsl:text>Langue française</xsl:text>
    </xsl:if>
    </td>
    </tr>
    </xsl:template>
  </xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.32 / L'élément <xsl:value-of>

L'élément *<xsl:value-of>* permet d'extraire la valeur d'un noeud sélectionné dans l'arborescence d'un document XML.

```
<xsl:value-of
        select="expression"
        disable-output-escaping="yes|no"/>
```

Cet élément ne peut traiter qu'un seul noeud à la fois contrairement à *<xsl:apply-tempates>* qui gére un ensemble de noeuds sélectionné par son *pattern*.

Utilisé en conjonction avec l'instruction *<xsl:for-each>*, l'élément *<xsl:value-of>* peut alors traiter une série de noeuds correspondant à l'expression de l'attribut *select*.

Les attributs :

Attribut	Description
select="pattern"	sélectionne des noeuds dans une arborescence source.
disable-output-escaping="yes|no"	active ou désactive le remplacement des caractères spéciaux par leur entité XML.

L'élément *<xsl:value-of>* peut non-seulement **sélectionner des éléments XML**, mais aussi **des attributs** par l'intermédiaire du signe *@* ainsi que **des paramètres ou des variables** avec *$*.

L'élément précitée ne peut être contenu que dans une règle de modèle *<xsl:template>* ou un appel à cette dernière *<xsl:call-template>*.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément ne peut contenir aucunes instructions.

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
    <head>
      <title>
      La logithèque :
        <xsl:value-of
                select="/logitheque/categorie[position()=10]/@nom"/>
      </title>
    </head>
    <body>
```

```
      <h2>
      <xsl:value-of
              select="/logitheque/categorie[position()=10]/@nom"/>
      </h2>
      <table border="0" width="60%">
      <tr>
        <th colspan="3">Logiciel</th>
      </tr>
      <xsl:apply-templates
              select="/logitheque/categorie[position()=10]/logiciel"/>
      </table>
    </body>
  </html>
  </xsl:template>
  <xsl:template
          match="/logitheque/categorie[position()=10]/logiciel">
  <tr>
     <td class="c1">
      <a href="{editeur/@lien}" target="_blank"
        style="font-size:10pt; font-weight:bold">
      <xsl:apply-templates select="nom"/>
      </a>
     </td>
     <td> - </td>
     <td><xsl:apply-templates select="commentaire"/></td>
  </tr>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.33 / L'élément <xsl:variable>

L'élément *<xsl:variable>* permet de déclarer une variable dans une feuille de style.

```
<xsl:variable name="nom">
  Valeur...
</xsl:variable>
```

La valeur du paramètre se trouve au sein des marqueurs.

Si l'attribut *select* est utilisé, alors *<xsl:variable>* doit être un élément vide et le contenu de l'attribut sera la valeur du paramètre.

```
<xsl:variable name="nom" select="expression"/>
```

Les attributs :

Attribut	Description
name="nom"	affecte un nom au paramètre.
select="pattern"	sélectionne un type de noeuds dans l'arborescence du document XML.

Dans le cas où la variable est déclarée dans l'élément *<xsl:stylesheet>*, alors elle serait valable dans l'ensemble de la feuille.

Déclarée dans une règle de modèle *<xsl:template>*, la variable a une portée locale.

Une variable est appelée par l'intermédiaire de son nom qui doit être précédé du signe dollar (*$*).

Deux éléments *<xsl:variable>* ne peuvent se faire mutuellement référence.

Cet élément peut être contenu dans les instructions suivantes :

- xsl:attribute
- xsl:comment
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:otherwise
- xsl:param
- xsl:processing-instruction
- xsl:stylesheet
- xsl:template
- xsl:variable
- xsl:when
- xsl:with-param
- éléments HTML

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if

- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
             media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <xsl:variable name="num" select="1"/>
 <html>
    <head>
      <title>La logithèque</title>
    </head>
    <body>
      <h2>
      Le premier logiciel de chaque catégorie
      </h2>
      <table border="0" width="60%" class="produit">
      <tr>
        <th colspan="3">Logiciel</th>
      </tr>
      <tr>
        <th style="text-align:left">Editeur</th>
        <th>Langue</th>
        <th>OS</th>
      </tr>
      <xsl:apply-templates
                  select="logitheque/categorie/logiciel[$num]">
        <xsl:sort select="nom"/>
      </xsl:apply-templates>
      </table>
    </body>
 </html>
 </xsl:template>
 <xsl:template match="logitheque/categorie/logiciel">
      <tr>
        <th colspan="3">
          <xsl:value-of select="../@nom"/>
        </th>
      </tr>
 <tr>
    <td colspan="3" class="c1">
       <a href="{editeur/@lien}" target="_blank"
            style="font-size:10pt; font-weight:bold"
            title="{prix} {prix/@monnaie}">
         <xsl:apply-templates select="nom"/>
       </a>
       - <xsl:apply-templates select="commentaire"/>
    </td>
 </tr>
 <tr>
    <td>
      <a href="{editeur/@lien}" target="_blank">
      <xsl:apply-templates select="editeur"/>
      </a>
    </td>
    <td>
      <xsl:apply-templates select="langue"/>
    </td>
    <td>
      <xsl:apply-templates select="plateforme"/>
    </td>
 </tr>
 </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.34 / L'élément <xsl:when>

L'élément *<xsl:when>* permet de créer une condition dans une structure conditionnelle *<xsl:choose>*.

```
<xsl:when test="expression>
  Instructions...
</xsl:when>
```

Les attributs :

Attribut	Description
test="pattern"	effectue un test conditionnel sur l'expression *pattern* spécifiée.

Cet élément ne peut être contenu que par l'instruction <xsl:choose>.

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html"
          media-type="text/html; charset=ISO-8859-1" version="4.0"/>
 <xsl:template match="/">
 <html>
   <head>
    <title>La logithèque :
          <xsl:value-of select="logitheque/categorie[position()=7]/@nom"/>
    </title>
   </head>
   <body>
    <table border="0" width="60%" class="produit">
    <tr>
     <th>Logiciel</th>
     <th>Lien</th>
    </tr>
    <xsl:apply-templates
          select="logitheque/categorie[position()=7]/logiciel"/>
    </table>
   </body>
 </html>
 </xsl:template>
 <xsl:template match="logitheque/categorie[position()=7]/logiciel">
 <xsl:choose>
   <xsl:when test="editeur/@lien != ''">
    <xsl:variable name="url" select="editeur/@lien"/>
    <tr>
    <td class="c1">
     <a href="{editeur/@lien}" target="_blank"
```

```
                style="font-size:10pt; font-weight:bold">
              <xsl:apply-templates select="nom"/>
             </a>
            </td>
            <td>
             <xsl:value-of select="$url"/>
            </td>
            </tr>
           </xsl:when>
           <xsl:otherwise>
            <xsl:variable name="url">failed.html</xsl:variable>
            <tr>
            <td class="c1">
             <a href="failed.html" target="_blank"
                style="font-size:10pt; font-weight:bold">
              <xsl:apply-templates select="nom"/>
             </a>
            </td>
            <td>
             <xsl:value-of select="$url"/>
            </td>
            </tr>
           </xsl:otherwise>
         </xsl:choose>
         </xsl:template>
        </xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

7.35 / L'élément <xsl:with-param>

L'élément *<xsl:with-param>* permet de déclarer un paramètre à passer à une règle de modèle *<xsl:template>*.

```
<xsl:with-param name="nom">
  Valeur...
</xsl:with-param>
```

Si l'attribut *select* est utilisé, alors *<xsl:with-param>* doit être un élément vide et le contenu de l'attribut sera la valeur du paramètre.

```
<xsl:with-param select="expression"/>
```

Cet élément ne peut apparaître que dans les instructions *<xsl:call-template>* ou *<xsl:apply-templates>*.

```
<xsl:call-template select="expression">
  <xsl:with-param name="nom">
    Valeur
  </xsl:with-param>
</xsl:call-template>
```

Les attributs :

Attribut	Description
name="nom"	affecte un nom au paramètre.
select="pattern"	sélectionne un type de noeuds dans l'arborescence du document XML.

Un paramètre est appelé par l'intermédiaire de son nom, qui doit être précédé du signe dollar (*$*).

Cet élément ne peut être contenu que par les instructions <xsl:call-template> et <xsl:apply-templates>.

Cet élément peut contenir les instructions suivantes :

- xsl:apply-templates
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:text
- xsl:value-of
- xsl:variable
- éléments HTML

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:variable name="incrementations" select="100"/>

 <xsl:template match="/">
 <html>
   <body>
     <table>
```

```
      <xsl:call-template name="annee"/>
      </table>
    </body>
  </html>
  </xsl:template>
  <xsl:template name="annee">
  <xsl:param name="i">0</xsl:param>
  <xsl:param name="bissextile">2000</xsl:param>

  <xsl:if test="$bissextile mod 4 = 0
                and not($bissextile mod 100 = 0)
                or $bissextile mod 400 = 0">
    <tr>
      <td>L'année "</td>
      <td>
      <xsl:value-of select="$bissextile"/>
      </td>
      <td>" est une année bissextile</td>
    </tr>
    <xsl:text>   </xsl:text>
  </xsl:if>
  <xsl:if test="$i &lt;= $incrementations">
    <xsl:call-template name="annee">
      <xsl:with-param name="i" select="$i + 1"/>
      <xsl:with-param name="bissextile" select="$bissextile + 4"/>
    </xsl:call-template>
  </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft

8 / Les éléments de sortie HTML

Les éléments de sortie HTML permettent de présenter clairement des données XML dans un navigateur web.

Ce type de transformation est non seulement très utile, mais fait également preuve d'une **quasi-parfaite portabilité** sur des systèmes d'exploitation ou sur des navigateurs.

Toutes les balises HTML peuvent être employées au sein des éléments de tranformation XSL.

Néanmoins, **le balisage HTML doit correspondre à celui de la version 4.0** du langage et ainsi être parfaitement structuré.

Toutes les balises doivent être fermées, y compris les éléments vides tels que *<img>* ou
.

Les marqueurs doivent s'imbriquer correctement.

Malgré l'absence d'une méthode de transformation annoncée par l'élément <xsl:output>,
le processeur XSL active automatiquement une sortie sous forme HTML, si la feuille de style de transformation contient le marqueur *<html>* avec n'importe quelle combinaison de casse.

Le balisage HTML doit se trouver au sein des éléments XSL suivants :

- xsl:template
- xsl:element
- xsl:for-each
- xsl:if
- xsl:number
- xsl:otherwise
- xsl:param
- xsl:text
- xsl:variable
- xsl:when
- xsl:with-param

Ces éléments peuvent contenir les instructions suivantes :

- xsl:apply-imports
- xsl:apply-templates
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy-of
- xsl:copy
- xsl:element
- xsl:for-each
- xsl:if
- xsl:processing-instruction
- xsl:value-of
- xsl:variable

Exemple : [voir]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:template match="/">
 <html>
   <head>
    <title>
    La logithèque : les <xsl:value-of select="/logitheque/@nom"/>
    </title>
```

```
    </head>
    <body>
      <h2>
      Les <xsl:value-of select="/logitheque/@nom"/>
      </h2>
      <table border="0" width="60%" class="produit">
      <tr>
        <th colspan="3">Logiciel</th>
      </tr>
      <tr>
        <th style="text-align:left">Editeur</th>
        <th>Langue</th>
        <th>OS</th>
      </tr>
      <xsl:apply-templates select="/logitheque/logiciel">
        <xsl:sort select="nom"/>
      </xsl:apply-templates>
      </table>
    </body>
</html>
</xsl:template>
<xsl:template match="/logitheque/logiciel">
<tr>
    <td colspan="3" class="c1">
      <xsl:choose>
      <xsl:when test="editeur/@lien != ''">
        <a href="{editeur/@lien}" target="_blank"
            style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
        </a>
      </xsl:when>
      <xsl:otherwise>
        <a href="failed.html" target="_blank"
            style="font-size:10pt; font-weight:bold">
          <xsl:apply-templates select="nom"/>
        </a>
      </xsl:otherwise>
      </xsl:choose>
       - <xsl:apply-templates select="./commentaire"/>
    </td>
</tr>
<tr>
    <td style="white-space:nowrap">
      <a href="{editeur/@lien}" target="_blank">
      <xsl:apply-templates select="./editeur"/>
      </a>
    </td>
    <td>
      <xsl:apply-templates select="./langue"/>
    </td>
    <td>
      <xsl:apply-templates select="./plateforme"/>
    </td>
</tr>
</xsl:template>
</xsl:stylesheet>
```

En savoir plus :

W3C Microsoft