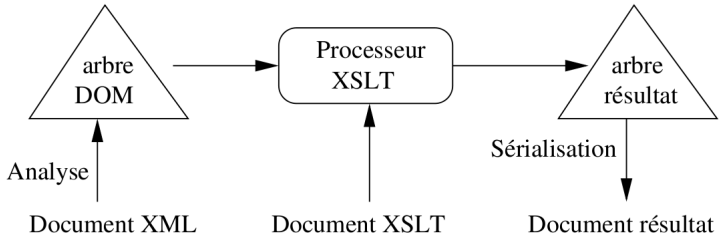
Xsl

Table des matières



XSLT est, comme son nom l'indique, un langage destiné à transformer un fichier XML en quelque chose d'autre.
XSLT offre de nombreuses fonctions dignes d'un langage de haut-niveau : variables, paramètres, tests, boucles, fonctions, inclusion d'une feuille de style XSLT dans une autre, chargement de plusieurs documents XML dans une même feuille de style XSLT, recherche de balises XML selon de nombreux critères, etc.

Structure

xsl:stylesheet

élément racine de la feuille de style

xsl:output

permet de définir le type de sortie qui sera produit et de générer des entêtes.

<xsl:output
method = "xml" | "html" | "text"
version = nmtoken
encoding = string
omit-xml-declaration = "yes" | "no"
standalone = "yes" | "no"
doctype-public = string
doctype-system = string
indent = "yes" | "no"
media-type = string />

method="xml|html|text" *indique le format du document résultant.*

- Pour spécifier **une sortie HTML :**
  <xsl:output method="html" />

```
<xsl:output
   method="html"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd"
   doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
   indent="yes"
   encoding="ISO-8859-1"
/>
```

- Pour spécifier **une sortie TEXT :**
  <xsl:output method="text" />

A mettre au début du fichier (après xsl:stylesheet)

<xsl:template> règle modèle

L'élément <xsl:template> permet de définir une règle de modèle.

Quelle est la syntaxe d'une règle modèle ?

Elle est la suivante :

<xsl:template match="*motif*"> *<!-- Spécifie à quels noeuds la règle est applicable -->*

<!-- Insertions dans le fichier cible de données prélevées/calculées à partir des données du fichier source. Appel éventuel d'autres règles modèles -->

</xsl:template>

Si vous souhaitez que le modèle fasse quelque chose, ne serait-ce que rendre la main à d'autres modèles, eh bien il faut le lui demander explicitement !

Vous pouvez, par exemple lui demander de simplement remplacer les balises <para> et tout ce qu'elles contiennent par le texte "trouvé !". En ce cas il vous suffira d'écrire :

```
<xsl:template match="//para">
trouvé !
</xsl:template>
```

// Indique tous les descendants d'un noeud

Si vous désirez que ce texte "trouvé !" apparaisse à l'intérieur d'une nouvelle balise, par exemple <p>, alors vous écrirez :

```
<xsl:template match="//para">
<p>trouvé !</p>
</xsl:template>
```

Les principaux motifs (patterns) sont :

Pattern	Exemple	Signification
|	**Gauche|Milieu**	Indique une alternative (un noeud ou bien l'autre (ou les deux))
/	**personne/nom**	Chemin d'accès aux éléments (*personne/bras/gauche*) au même titre que l'arborescence utilisée généralement pour les fichiers (*/usr/bin/toto*)
*	*****	Motif "joker" désignant n'importe quel élément
//	**//personne**	Indique tous les descendants d'un noeud
.	**.**	Caractérise le noeud courant
..	**..**	Désigne le noeud parent
@	**@valeur**	Indique un attribut caractéristique

La transformation peut être réalisée

- soit par ajout de texte,
- soit en définissant des *éléments de transformation*, c'est-à-dire des éléments permettant de

définir des règles de transformation à appliquer aux éléments sélectionnés par l'attribut *match*

L'attribut *select* permet de spécifier certains éléments enfants auxquels la transformation doit être appliquée
La première chose à faire quand on commence une feuille XSL devant être appliquée à un fichier XML, c'est écrire **`<xsl:template match="/"> </xsl:template>`** pour se placer à la racine du document.

<xsl:apply-templates select= ".." />

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

```
<xsl:apply-templates select="cible"/>
```

L'élément apply-templates utilisé au sein de la balise <xsl:template match="..."> permet d'appliquer la règle de transformation contenu dans la balise template.

L'élément **<xsl:apply-templates/>** indique le traitement de tous les enfants directs de la racine.

<xsl:value-of>

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

Un exemple simple

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="auteurs.xsl"?>
<Cours>
     <Titre>Cours XML</Titre>
     <Auteur>
          <Nom>Poulard</Nom>
          <Prenom>Philippe</Prenom>
     </Auteur>
     <Description>
     Ce cours aborde les concepts de base mis en &#339;uvre dans
XML.
     </Description>
</Cours>
```

Fichier auteurs.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/1999/xhtml">
<xsl:output method="html" encoding="ISO-8859-1"/>
<xsl:template match="/">
<html>
<head>
<title><xsl:value-of select="Cours/Titre"/></title>
</head>
<body>
<h1><xsl:value-of select="Cours/Titre"/></h1>
<xsl:apply-templates select="Cours/Auteur"/>
<xsl:apply-templates select="Cours/Description"/>
</body>
</html>
</xsl:template>
<xsl:template match="Auteur">
<p align="right">par <xsl:value-of select="Nom"/>
 <xsl:value-of select="Prénom"/></p>
</xsl:template>
<xsl:template match="Description">
<p><xsl:value-of select="."/></p>
</xsl:template>
</xsl:stylesheet>
```

dans le contexte de "/"

dans le contexte de "Auteur"

Exemple 2:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="stylesheet.xsl" ?>
<test>
1234567890
</test>
```

Fichier stylesheet.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"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
   doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
   indent="yes"
   encoding="iso-8859-1"
/>
<xsl:template match="/">
 <html><head></head>
 <body style="background-color:#000000; font-family:Algerian; font-size:80px;
color:#33FF33">
  <xsl:value-of select="." />
 </body></html>
</xsl:template>

</xsl:stylesheet>
```

Explication:

Un processeur XSL analyse un fichier XML source et s'efforce de trouver une règle modèle concordante. S'il en trouve une, les instructions qu'elle contient sont évaluées.

Le traitement commence toujours par le modèle sélectionnant le noeud correspondant au motif match="/". Ce noeud est le noeud racine.

Les données XML de l'exemple ne contiennent que l'élément racine avec une suite de chiffres comme contenu. La feuille de style de l'exemple n'en est que plus simple.

Avec **xsl:template** est définie un modèle pour transcrire les données XML en sortie HTML. Ces définitions XSL et d'autres sont incluses dans l'élément racine `xsl:stylesheet`. Dans l'exemple le repère d'ouverture de cet élément contient les mentions typiques sur la version XSLT, sur la source XSLT et sur la source de l'espace de nommage HTML employé (ici: XHTML).

Lorsque ce modèle n'est pas explicitement indiqué, le modèle implicite est utilisé (il comporte une seule instruction). Cette instruction signifie : traiter tous les noeuds fils du noeud courant, y compris les noeuds textuels

Autre exemple

Considérons une liste de livres avec nom du livre, auteur et date

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="bdd.xsl" ?>
<bibliotheque>

      <livre>
            <nom>Le guet des orfèvres</nom>
            <auteur>Terry Pratchett</auteur>
            <date>1993</date>
      </livre>

      <livre>
            <nom>Nobliaux et sorcières</nom>
            <auteur>Terry Pratchett</auteur>
            <date>1992</date>
      </livre>

      <livre>
            <nom>Mécomptes de fées</nom>
            <auteur>Terry Pratchett</auteur>
            <date>1991</date>
      </livre>
      <livre>
            <nom>Tinbin obéit aux serviettes</nom>
            <auteur>Alain Provistt</auteur>
            <date>1999</date>
      </livre>

</bibliotheque>
```

et sa feuille xsl: bdd.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"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
   doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
   indent="yes"
   encoding="ISO-8859-1"
/>

      <xsl:template match="/">
             <html>
             <head>
             </head>
             <body>
              <xsl:apply-templates />
             </body>
             </html>
      </xsl:template>

      <xsl:template match="/livre">

              <xsl:apply-templates />
      </xsl:template>

      <xsl:template match="nom">
            <h2>
              <xsl:value-of select="." />
            </h2>
      </xsl:template>

      <xsl:template match="auteur">
            <span style="color:blue">
              <xsl:value-of select="." />
            </span><br/>
      </xsl:template>

</xsl:stylesheet>
```

On remarque que la fin de chaque fiche est affichée sans demande précise;Lorsque le modèle n'est pas explicitement indiqué, le modèle implicite est utilisé (il comporte une seule instruction). Cette instruction signifie : traiter tous les noeuds fils du noeud courant, y compris les noeuds textuels.

Résultat

Le guet des orfèvres
Terry Pratchett
1993
Nobliaux et sorcières
Terry Pratchett
1992
Mécomptes de fées
Terry Pratchett
1991
Tinbin obéit aux serviettes
Alain Provistt
1999

Utiliser un attribut @

select="item/@date" permet de récupérer l'attribut date de item

Les attributs sont accessibles de la même manière que les éléments. Notez la présence du caractère **"@" devant les noms d'attributs**.

Source XML

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl"
href="style.xsl" ?>

<source>

<AAA id="a1" pos="start">
   <BBB id="b1"/>
   <BBB id="b2"/>
</AAA>
<AAA id="a2">
   <BBB id="b3"/>
   <BBB id="b4"/>
   <CCC id="c1">
      <DDD id="d1"/>
   </CCC>
   <BBB id="b5">
      <CCC id="c2"/>
   </BBB>
</AAA>

</source>
```

Feuille de style XSLT

```
<?xml version="1.0" encoding="ISO-8859-
1"?>
<xsl:stylesheet version = '1.0'
   xmlns:xsl='http://www.w3.org/1999/XSL/Tra
nsform'>

<xsl:template match="AAA">
   <div style="color:purple">
      <xsl:value-of select="name()"/>
      <xsl:text> id=</xsl:text>
      <xsl:value-of select="@id"/>
   </div>
</xsl:template>


</xsl:stylesheet>
```

Sortie

```
<div style="color:purple">AAA
id=a1</div>

<div style="color:purple">AAA
id=a2</div>
```

Vue HTML

AAA id=a1
AAA id=a2

xsl:for-each

L'instruction xsl:for-each comporte un modèle qui est appliqué à chaque noeud sélectionné à l'aide de l'attribut select.
xsl:for-each n'est pas vraiment une boucle comme un "for" en langage C. Cette fonction va prendre tous les noeuds d'une requête XPATH, et va leur appliquer un traitement.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="style.xsl" ?>
<source>

<AAA id="a1" pos="start">
     <BBB id="b1"/>
     <BBB id="b2"/>
</AAA>
<AAA id="a2">
     <BBB id="b3"/>
     <BBB id="b4"/>
     <CCC id="c1">
          <DDD id="d1"/>
     </CCC>
     <BBB id="b5">
          <CCC id="c2"/>
     </BBB>
</AAA>

</source>
```

Feuille de style XSLT

```
<?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="/">
     <xsl:for-each select="//BBB">
          <DIV style="color:red">
               <xsl:value-of select="name()"/>
               <xsl:text> id=</xsl:text>
               <xsl:value-of select="@id"/>
          </DIV>
     </xsl:for-each>
     <xsl:for-each select="source/AAA/CCC">
          <DIV style="color:navy">
               <xsl:value-of select="name()"/>
               <xsl:text> id=</xsl:text>
               <xsl:value-of select="@id"/>
          </DIV>
     </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

Feuille de style dans le fichier xsl

Pour inclure la feuille de style directemment dans le fichier xsl, il suffit d'utiliser <![CDATA[...]]>:

```
<style type="text/css">
    <![CDATA[
    h1 {....}
    ]]>
</style>
```

Exemple:liste.xsl et liste.xml

```
<?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" doctype-
system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd" doctype-public="-//W3C//DTD XHTML
1.0 Transitional//EN"   indent="yes" encoding="iso-
8859-1" />

<xsl:template match="/">
 <html>
 <head>
  <style type="text/css">
   <![CDATA[
  h1 {color:red}
  p {font-family:Arial,Helvetica,sans-
serif; font-size:12pt}
  b  {color:blue}
  ]]>
   </style>
 </head>
 <body>
  <xsl:apply-templates />
 </body>
 </html>
</xsl:template>

<xsl:template match="titre">
 <h1><xsl:value-of select="." /></h1>
</xsl:template>

<xsl:template match="glossaire/element">
 <p>
  <xsl:apply-templates />
 </p>
</xsl:template>

<xsl:template match="terme">
 <b ><xsl:apply-templates />: </b>
</xsl:template>

<xsl:template match="signification">
  <xsl:value-of select="." />
</xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="ISO-
8859-1"?>
<?xml-stylesheet type="text/xsl"
href="liste.xsl" ?>
<test>
 <titre>Extensions de fichiers</titre>
 <glossaire>
  <element>
   <terme>bak</terme>
   <signification>
    fichier de
    sauvegarde
    </signification>
  </element>
  <element>
   <terme>bmp</terme>
   <signification>
    graphique Bitmap
    </signification>
  </element>
 </glossaire>
</test>
```

Élément <xsl:text>

L'élément <xsl:text> est utlisé pour érire du texte en sortie.

TEST

- **l'égalité** : =
- **supérieur** : &gt ;
- **inférieur** : &lt ;
- **addition** : +
- **soustraction** : - (attention ce symbole est accepté dans les noms de balises, il faut donc impérativement le faire précéder et succéder d'un espace si on veut qu'il soit interprété)
- **division** : div
- **`modulo`** : `mod`

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.

Exemple

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="invites.xsl" ?>
<liste>
   <invite>Moi</invite>
   <invite>Jean</invite>
   <invite>Michel</invite>
</liste>
```

invites.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"
   encoding="ISO-8859-1"
/>
 <xsl:template match="/">
 <ul>
   <xsl:for-each select="//invite">
      <li>
         <xsl:apply-templates select="." />
      <xsl:if test=".='Michel'">
```

```
            <i>
             <xsl:text> bonjour Michel </xsl:text>
            </i>
             </xsl:if>
      </li>
   </xsl:for-each>
</ul>
</xsl:template>
</xsl:stylesheet>
```

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.

xsl:choose - xsl:otherwise

La fonction **xsl:choose** permet d'exécuter différents codes selon différentes conditions. Exemple d'utilisation de la l'instruction choose :

On utilise souvent xsl:choose comme alternative au xsl:if pour disposer de **xsl:otherwise** (qui remplace xsl:else qui n'existe pas).

On utilise souvent xsl:choose comme alternative au xsl:if pour disposer de xsl:otherwise (qui remplace xsl:else qui n'existe pas).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="invites.xsl" ?>
<liste>
   <invite>Moi</invite>
   <invite>Jean</invite>
   <invite>Michel</invite>
</liste>
```

```
<?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="/">
 <ul>
   <xsl:for-each select="//invite">
        <li>
            <xsl:apply-templates select="." />
   <xsl:choose>
            <xsl:when test=".= 'Moi'">
            <xsl:text> Salut</xsl:text>
            </xsl:when>

            <xsl:when test=".= 'Michel'">
            <xsl:text> Fait ciseau</xsl:text>
            </xsl:when>

            <xsl:otherwise>
                <xsl:text> Message impersonnel Bonjour</xsl:text>
            </xsl:otherwise>

   </xsl:choose>
        </li>
```

```
   </xsl:for-each>
</ul>
</xsl:template>
</xsl:stylesheet>
```

For-each: autre exemple -catalogue cd-

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<catalog>
	<cd>
		<title>Empire Burlesque</title>
		<artist>Bob Dylan</artist>
		<country>USA</country>
		<company>Columbia</company>
		<price>10.90</price>
		<year>1985</year>
	</cd>
	<cd>
		<title>Hide your heart</title>
		<artist>Bonnie Tyler</artist>
		<country>UK</country>
		<company>CBS Records</company>
		<price>9.90</price>
		<year>1988</year>
	</cd>
	<cd>
		<title>Greatest Hits</title>
		<artist>Dolly Parton</artist>
		<country>USA</country>
		<company>RCA</company>
		<price>9.90</price>
		<year>1982</year>
	</cd>
	<cd>
		<title>Still got the blues</title>
		<artist>Gary Moore</artist>
		<country>UK</country>
		<company>Virgin records</company>
		<price>10.20</price>
		<year>1990</year>
	</cd>
	<cd>
		<title>Eros</title>
		<artist>Eros Ramazzotti</artist>
		<country>EU</country>
		<company>BMG</company>
		<price>9.90</price>
		<year>1997</year>
	</cd>
	<cd>
		<title>One night only</title>
		<artist>Bee Gees</artist>
		<country>UK</country>
		<company>Polydor</company>
		<price>10.90</price>
		<year>1998</year>
	</cd>
	<cd>
		<title>Sylvias Mother</title>
		<artist>Dr.Hook</artist>
		<country>UK</country>
		<company>CBS</company>
```

```
      <price>8.10</price>
      <year>1973</year>
</cd>
<cd>
      <title>Maggie May</title>
      <artist>Rod Stewart</artist>
      <country>UK</country>
      <company>Pickwick</company>
      <price>8.50</price>
      <year>1990</year>
</cd>
<cd>
      <title>Romanza</title>
      <artist>Andrea Bocelli</artist>
      <country>EU</country>
      <company>Polydor</company>
      <price>10.80</price>
      <year>1996</year>
</cd>
<cd>
      <title>When a man loves a woman</title>
      <artist>Percy Sledge</artist>
      <country>USA</country>
      <company>Atlantic</company>
      <price>8.70</price>
      <year>1987</year>
</cd>
<cd>
      <title>Black angel</title>
      <artist>Savage Rose</artist>
      <country>EU</country>
      <company>Mega</company>
      <price>10.90</price>
      <year>1995</year>
</cd>
<cd>
      <title>1999 Grammy Nominees</title>
      <artist>Many</artist>
      <country>USA</country>
      <company>Grammy</company>
      <price>10.20</price>
      <year>1999</year>
</cd>
<cd>
      <title>For the good times</title>
      <artist>Kenny Rogers</artist>
      <country>UK</country>
      <company>Mucik Master</company>
      <price>8.70</price>
      <year>1995</year>
</cd>
<cd>
      <title>Big Willie style</title>
      <artist>Will Smith</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>9.90</price>
      <year>1997</year>
</cd>
<cd>
```

```
      <title>Tupelo Honey</title>
      <artist>Van Morrison</artist>
      <country>UK</country>
      <company>Polydor</company>
      <price>8.20</price>
      <year>1971</year>
</cd>
<cd>
      <title>Soulsville</title>
      <artist>Jorn Hoel</artist>
      <country>Norway</country>
      <company>WEA</company>
      <price>7.90</price>
      <year>1996</year>
</cd>
<cd>
      <title>The very best of</title>
      <artist>Cat Stevens</artist>
      <country>UK</country>
      <company>Island</company>
      <price>8.90</price>
      <year>1990</year>
</cd>
<cd>
      <title>Stop</title>
      <artist>Sam Brown</artist>
      <country>UK</country>
      <company>A and M</company>
      <price>8.90</price>
      <year>1988</year>
</cd>
<cd>
      <title>Bridge of Spies</title>
      <artist>T`Pau</artist>
      <country>UK</country>
      <company>Siren</company>
      <price>7.90</price>
      <year>1987</year>
</cd>
<cd>
      <title>Private Dancer</title>
      <artist>Tina Turner</artist>
      <country>UK</country>
      <company>Capitol</company>
      <price>8.90</price>
      <year>1983</year>
</cd>
<cd>
      <title>Midt om natten</title>
      <artist>Kim Larsen</artist>
      <country>EU</country>
      <company>Medley</company>
      <price>7.80</price>
      <year>1983</year>
</cd>
<cd>
      <title>Pavarotti Gala Concert</title>
      <artist>Luciano Pavarotti</artist>
      <country>UK</country>
      <company>DECCA</company>
```

```
            <price>9.90</price>
            <year>1991</year>
      </cd>
      <cd>
            <title>The dock of the bay</title>
            <artist>Otis Redding</artist>
            <country>USA</country>
            <company>Atlantic</company>
            <price>7.90</price>
            <year>1987</year>
      </cd>
      <cd>
            <title>Picture book</title>
            <artist>Simply Red</artist>
            <country>EU</country>
            <company>Elektra</company>
            <price>7.20</price>
            <year>1985</year>
      </cd>
      <cd>
            <title>Red</title>
            <artist>The Communards</artist>
            <country>UK</country>
            <company>London</company>
            <price>7.80</price>
            <year>1987</year>
      </cd>
      <cd>
            <title>Unchain my heart</title>
            <artist>Joe Cocker</artist>
            <country>USA</country>
            <company>EMI</company>
            <price>8.20</price>
            <year>1987</year>
      </cd>
</catalog>
```

Feuille 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"
   encoding="ISO-8859-1"
/>
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each
      select="catalog/cd">
      <xsl:if test="price>10">
            <tr>
            <td><xsl:value-of
            select="title"/></td>
            <td><xsl:value-of
            select="artist"/></td>
            </tr>
      </xsl:if>
    </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Pour chaque cd du catalogue

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Still got the blues	Gary Moore
One night only	Bee Gees
Romanza	Andrea Bocelli
Black angel	Savage Rose
1999 Grammy Nominees	Many

xsl:sort

Tri alphabétique **"ascending"** par défaut

<xsl:sort select="..." **order**="descending"/>

Tri numérique

<xsl:sort select="..." data-type="number" order="descending"/>

Trier les éléments d'après leur contenu.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="sort.xsl" ?>
<test>
<joueur><nom>Antoine</nom><points>12</points></joueur>
<joueur><nom>Patrick</nom><points>19</points></joueur>
<joueur><nom>Richard</nom><points>27</points></joueur>
<joueur><nom>William</nom><points>10</points></joueur>
</test>
```

fichier sort.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"
   encoding="ISO-8859-1"
/>
<xsl:template match="/">
 <html>
 <head>
 </head>
 <body>
 <table border="1">
  <xsl:for-each select="test/joueur">
   <xsl:sort select="points" order="descending" data-type="number" />
   <tr>
   <td><xsl:value-of select="nom" /></td>
   <td><xsl:value-of select="points" /></td>
   </tr>
  </xsl:for-each>
 </table>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

Élément <xsl:number>

http://msdn.microsoft.com/fr-fr/library/ms256084%28v=vs.80%29.aspx

Insère un nombre formaté dans l'arborescence résultat.

```
<xsl:number
  level = "single" | "multiple" | "any"
  count = Pattern
  from = Pattern
  value = number-expression
  format = { string }
  lang = { nmtoken }
  letter-value = { "alphabetic" | "traditional" }
  grouping-separator = { char }
  grouping-size = { number } />
```

Attributs

---

level

Spécifie les niveaux de l'arborescence source à prendre en considération ; les valeurs possibles sont `"single"`, `"multiple"` et `"any"`. La valeur par défaut est `"single"`.

count

Modèles qui spécifie les nœuds à compter à ces niveaux. Si l'attribut `count` n'est pas spécifié, la valeur par défaut est le modèle correspondant à tout nœud ayant le même type de nœud que le nœud actuel et, si le nœud actuel a un nom développé, ayant le même nom développé que le nœud actuel.

from

Modèles qui spécifie où commence le comptage.

value

Spécifie l'expression à convertir en un nombre et à insérer dans l'arborescence résultat. Si aucun attribut `value` n'est spécifié, l'élément `<xsl:number>` insère un nombre d'après la position du nœud actuel dans l'arborescence source.

format

Séquence de jetons spécifiant le format à utiliser pour chaque nombre figurant dans la liste. S'il n'y a pas de jetons de format, la valeur par défaut utilisée est 1, qui génère une séquence 1 2 ... 10 11 12.... Chaque nombre hormis le premier est séparé du précédent par le jeton séparateur, lequel précède le jeton de format utilisé pour formater ce nombre. S'il n'y a pas de jetons séparateurs, c'est un point qui est utilisé (« . »).

.

Jeton de format	Séquence générée
1	1 2 3 4 5 ... 10 11 12 ...
01	01 02 03 ... 19 10 11 ... 99 100 101...
A	A B C ... Z AA AB AC...
i	i ii iii iv v vi vii viii ix x...
I	I II III IV V VI VII VIII IX X...

lang

Spécifie l'alphabet utilisé. Si aucune langue n'est spécifiée, elle est déterminée par l'environnement système.

letter-value

Lève l'ambiguïté entre les séquences de numérotation qui utilisent des lettres. Une séquence de numérotation affecte des valeurs numériques aux lettres dans l'ordre alphabétique, tandis que l'autre affecte des valeurs numériques à chaque lettre d'une autre manière traditionnelle pour cette langue. En anglais, ces séquences de numérotation sont spécifiées par les jetons de format « a » et « i ». Dans certaines langues, le premier membre de chaque séquence est identique, si bien que le jeton de format seul serait ambigu. La valeur `"alphabetic"` spécifie l'ordre alphabétique ; la valeur `"traditional"` spécifie l'autre possibilité. La valeur par défaut est `"alphabetic"`.

grouping-separator

Définit le séparateur utilisé pour le regroupement (p. ex. des milliers) dans les séquences de numérotation décimales. Par exemple, `grouping-separator=","` et `grouping-size="3"` produiraient des nombres du genre 1,000,000. Si un seul des attributs `grouping-separator` et `grouping-size` est spécifié, il est ignoré.

grouping-size

Spécifie la taille (normalement 3) des groupes. Par exemple, `grouping-separator=","` et `grouping-size="3"` produiraient des nombres du genre 1,000,000. Si un seul des attributs `grouping-separator` et `grouping-size` est spécifié, il est ignoré.

Les langues/schémas de numérotation suivants sont pris en charge. « Jeton de format » correspond à l'attribut `format`, « Langue » correspond à l'attribut `lang` et « Valeur lettre » correspond à l'attribut `letter-value`.

Description	Jeton de format	Langue	Valeur lettre
Occidental	0x0031 (1)	n/a	n/a
Lettre majuscule	0x0041 (A)	n/a	n/a
Lettre minuscule	0x0061 (a)	n/a	n/a
Chiffre romain majuscule	0x0049 (I)	n/a	n/a
Chiffre romain minuscule	0x0069 (i)	n/a	n/a
Russe (cyrillique) majuscule	0x0410	n/a	n/a
Russe (cyrillique) minuscule	0x0430	n/a	n/a
Hébreu alphabétique	0x05d0	n/a	Alphabétique
Hébreu traditionnel	0x05d0	n/a	Arabe
traditionnel	0x0623	n/a	n/a
Hindi, consonnes	0x0905	n/a	n/a
Hindi, voyelles	0x0915	n/a	n/a
Hindi, chiffres	0x0967	n/a	n/a
Thaï, lettres	0x0e01	n/a	n/a
Thaï, chiffres	0x0e51	n/a	n/a
Japonais, Aiueo (double octet)	0x30a2	n/a	n/a
Japonais, Iroha (double octet)	0x30a4	n/a	n/a

Description	Jeton de format	Langue	Valeur lettre
Coréen, Chosung	0x3131	n/a	n/a
Taïwanais décimal	0x4e01	« zh-tw »	n/a
Coréen décimal	0x4e01	« ko »	n/a
Asiatique décimal	0x4e01	toute autre langue	n/a
Asiatique, Kanji	0x58f1	n/a	n/a
Taïwanais traditionnel	0x58f9	« zh-tw »	n/a
Chinois traditionnel	0x58f9	toute autre langue	n/a
Chinois « Zodiaque » 12	0x5b50	n/a	n/a
Chinois « Zodiaque » 10	0x7532	n/a	n/a
Chinois « Zodiaque » 60	0x7532, 0x5b50	n/a	n/a
Coréen, Ganada	0xac00	n/a	n/a
Coréen décimal	0xc77c	n/a	n/a
Coréen 99	0xd558	n/a	n/a
Occidental (double octet)	0xff11	n/a	n/a
Japonais, Aiueo (simple octet)	0xff71	n/a	n/a
Japonais, Iroha (simple octet)	0xff72	n/a	n/a

Si le jeton de format à lui seul suffit pour lever l'ambiguïté pour un schéma de numérotation

particulier, il n'est pas nécessaire de spécifier la langue ni la valeur lettre.

Exemple :

```
<?xml version='1.0'?>
<?xml-stylesheet type="text/xsl" href="numelem.xsl" ?>
<items>
   <item>voiture</item>
   <item>Plume</item>
   <item>LP 33 tours</item>
   <item>Sagesse</item>
   <item>GSM</item>
   <item>Projecteur</item>
   <item>trou</item>
   <item>Verrière</item>
   <item>Widget</item>
   <item>Concepte</item>
   <item>Null character</item>
</items>
```

et son fichier xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:output method="html"/>

 <xsl:template match="items">
    <xsl:text>nombre d'items :</xsl:text>
    <xsl:value-of select="count(item)"/> <br/>
   <xsl:for-each select="item">
      <xsl:sort select="."/>
      <xsl:number value="position()" format="1. "/>
      <xsl:value-of select="."/>
      <blockquote>
      <xsl:number value="position()" format="&#x0069;-> "/>
      <xsl:value-of select="."/>
      </blockquote>
    <br/>
   </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

template et call-template

Il existe deux versions de l'élément template qui se différencient par leurs attributs :

<xsl:template match="XPath" > ... </xsl:template>	<xsl:template name="nom_modele"> ... </xsl:template>

La version avec l'attribut **match** définit un modèle par défaut, applicable à tel type d'élément. Ce modèle est instancié par le processeur XSLT à chaque fois qu'un élément correspondant est rencontré.
La version avec l'attribut **name** correspond aux fonctions des langages fonctionnels. Ils sont destinés à êtres "appelés" par l'instruction call-template.
Exemples
XSL

```
<xsl:template match="/" >
    <xsl:call-template name="dis_bonjour"/>
</xsl:template>
<xsl:template name="dis_bonjour">
    Bonjour !
</xsl:template>
```

Quelque soit le document XML en entrée, le processeur commence à chercher un modele pouvant s'appliquer à la racine, trouve ici le premier modele dans la feuille de style et l'applique. Ce modele spécifie la génération de texte et l'instanciation du modele dont le nom est dis_bonjour.
Il est possible, comme pour une fonction, de passer à un tel modele des paramètres et de réaliser des appels récursifs :

```
<xsl:template match="/" >
    énumération :
    <xsl:call-template name="enumer">
        <xsl:with-param name="start">0</xsl:with-param>
        <xsl:with-param name="end">9</xsl:with-param>
    </xsl:call-template>
</xsl:template>

<xsl:template name="enumer">
    <xsl:param name="start"/>
    <xsl:param name="end"/>

    <xsl:value-of select="$start"/>

    <xsl:if test="$start &lt; $end">
        <xsl:call-template name="enumer">
            <xsl:with-param name="start"><xsl:value-of select="$start +
1"/></xsl:with-param>
            <xsl:with-param name="end"><xsl:value-of select="$end"/></xsl:with-
param>
        </xsl:call-template>
    </xsl:if>
</xsl:template>
```

Réponse

énumération :

0123456789