Xpath

XML Path

Définition :

XPath est un langage qui permet de parcourir des éléments et des attributs dans un document XML.

XPath est un élément majeur de la norme XSLT, il permet d'extraire des données d'un document XML.

XPath spécifie sept types de nœuds pouvant être générés par l'exécution de l'expression Xpath:

- Root
- Element
- Text
- Attribute
- Comment
- Processing Instruction
- Namespace

La syntaxe de Xpath:

- Catalog/album/track[4]
- Catalog/album[2]/track[4]

```
<?xml version="1.0" encoding="utf-8" ?>
<Catalog>
    <Album artist="The Last Shadow Puppets" title="The Age Of The Understatement">
        <Track rating="4" length="P3M7S">The Age Of The Understatement</Track>
        <Track rating="3" length="P2M18S">Standing Next To Me</Track>
        <Track rating="5" length="P2M26S">Calm Like You</Track>
        <Track rating="3" length="P3M38S">Separate and Ever Deadly</Track>
        <Track rating="2" length="P2M37S">The Chamber</Track>
        <Track rating="3" length="P2M44S">Only The Truth</Track>
    </Album>
    <Album artist="Kings Of Leon" title="Because Of The Times">
        <Track rating="4" length="P7M10S">Knocked Up</Track>
        <Track rating="2" length="P2M57S">Charmer</Track>
        <Track rating="3" length="P3M21S">On Call</Track>
        <Track rating="4" length="P3M09S">McFearless</Track>
        <Track rating="1" length="P3M59S">Black Thumbnail</Track>
    </Album>
</Catalog>
```

Nœuds XPath

Parent

```
<?xml version="1.0"?>
<messages>
  <message numero="4">
    <dest bcc="oui">promo2018</dest>
    <contenu>Salut</contenu>
  </message>
</messages>
```

- l'élément messages est le parent du .
- l'élément message est le parent de et .

Children

C’est un élément enfant d’une élément XML: message

```
<?xml version="1.0"?>
<messages>
  <message numero="4">
    <dest bcc="oui">promo2018</dest>
    <contenu>Salut</contenu>
  </message>
</messages>
```

Siblings

Les nœuds qui ont le même parent.

```
<?xml version="1.0"?>
<messages>
  <message numero="4">
    <dest bcc="oui">promo2018</dest>
    <contenu>Salut</contenu>
  </message>
</messages>
```

Ancestors

Un parent de noeud, parent de parent, etc.

```
<?xml version="1.0"?>
<messages>
  <message numero="4">
    <dest bcc="oui">promo2018</dest>
    <contenu>Salut</contenu>
  </message>
</messages>
```

les ancêtres de l'élément dest sont l'élément message et messages.

Descendants

Les enfants d'un noeud, les enfants des enfants, etc.
Les descendants de messages sont message,dest et contenu.

Les expressions utiles pour sélectionner un nœud ou une liste de nœuds à partir d'un document XML:

Indice	Expression	La description
1)	nom de noeud	Il est utilisé pour sélectionner tous les nœuds avec le nom donné "nodename"
2)	/	Il spécifie que la sélection commence à partir du nœud racine.
3)	//	Il spécifie que la sélection commence à partir du nœud actuel qui correspond à la sélection.
4)	.	Sélectionnez le nœud actuel.
5)	..	Sélectionnez le parent du nœud actuel.
6)	@	Sélectionne les attributs.
7)	étudiant	Exemple - sélectionne tous les nœuds avec le nom "étudiant".
8)	classe / étudiant	Exemple - sélectionne tous les éléments d'élève qui sont des enfants de la classe
9)	//étudiant	Sélectionne tous les éléments de l'élève, où qu'ils se trouvent dans le document.

Exemples :

```
<?xml version="1.0" encoding="utf-8" ?>
<Catalog>
    <Album artist="The Last Shadow Puppets" title="The Age Of The Understatement">
        <Track rating="4" length="P3M7S">The Age Of The Understatement</Track>
        <Track rating="3" length="P2M18S">Standing Next To Me</Track>
        <Track rating="5" length="P2M26S">Calm Like You</Track>
        <Track rating="3" length="P3M38S">Separate and Ever Deadly</Track>
        <Track rating="2" length="P2M37S">The Chamber</Track>
        <Track rating="3" length="P2M44S">Only The Truth</Track>
    </Album>
    <Album artist="Kings Of Leon" title="Because Of The Times">
        <Track rating="4" length="P7M10S">Knocked Up</Track>
        <Track rating="2" length="P2M57S">Charmer</Track>
        <Track rating="3" length="P3M21S">On Call</Track>
        <Track rating="4" length="P3M09S">McFearless</Track>
        <Track rating="1" length="P3M59S">Black Thumbnail</Track>
    </Album>
</Catalog>
```

Parcours des éléments XML:

Le parcours d'un éléments XML peut être absolu ou relatif:
Absolu: se fait à partir de la racine en utilisant seulement :/
Exemple:

/ Catalogue / Album

Relatif : se fait directement à partir d'un élément courant en utilisant ://

//Album/track

Pour parcourir tous les éléments d'u autre élément on utilise : //*

Parcours des attributs des éléments XML:

Pour parcourir les attributs d'un élément XML en utilise @.
Exemple:
/ Catalogue / Album/@artist

Parcours des éléments XML par prédicat:

```
< ?xml  version  = '1.0' ?>
<Catalogue>
        <Plante>
                <NomCommun>bloodroot</NomCommun>
                <NomBotanique>sanguinaria canadensis</NomBotanique >
                <zone>4</zone>
                <Eclairage> ombragé </Eclairage >
                <Prix>25</Prix>
                <QuantiteStock>0</QuantiteStock>
        </Plante>
        <Plante>
                <NomCommun>columbine</NomCommun>
                <NomBotanique>aquilegia canadensis</NomBotanique >
                <zone>3</zone>
                <Eclairage> ombragé </Eclairage >
                <Prix>100</Prix>
                <QuantiteStock>7</QuantiteStock>
        </Plante>

        <Plante>
                <NomCommun>marsh marigold</NomCommun>
                <NomBotanique>caltha palustris</NomBotanique >
                <zone>4</zone>
                <Eclairage>Ensoleillé</Eclairage >
                <Prix>70</Prix>
                <QuantiteStock>51</QuantiteStock>
        </Plante>
        <Plante>
                <NomCommun>cowslip</NomCommun>
                <NomBotanique>caltha palustris</NomBotanique >
                <zone>4</zone>
                <Eclairage>Ombrage</Eclairage >
                <Prix>100</Prix>
                <QuantiteStock>30</QuantiteStock>
        </Plante>
```

Sélectionnez tous les albums:

/ Catalogue / Album

Sélectionnez l'attribut artiste pour tous les albums:

/ Catalogue / Album / @ artiste

Sélectionnez toutes les pistes de tous les albums:

/ Catalogue / Album / Piste

Sélectionner tous les albums de Kings Of Leon:

/ Catalogue / Album [@ artist = "Kings Of Leon"]

Sélectionnez toutes les pistes de tous les albums de Kings Of Leon:

/ Catalogue / Album [@ artist = "Kings Of Leon"] / Piste

Sélectionnez le deuxième album:

/ Catalogue / Album [2]

Sélectionne toutes les pistes dont le classement est supérieur à '2':

/ Catalogue / Album / Piste [@rating> 2]

Sélectionne le texte pour toutes les pistes:

/ Catalogue / Album / Piste / texte ()

Examen 2018

Quelles sont les plantes qui ont l'élément 'nom'

```
//plante[nom]
```

//Catlogue/Plante [zone= 3]. **(1 pt)**

//Catlogue/Plante [QuantiteStock>0]. **(1 pt)**

On peut utiliser des fonctions filtrant les nœuds:

Prédicats	Notes
`last()`	renvoie le dernier nœud de la sélection
`position()`	renvoie le nœud situé à la position précisée
`count(contexte)`	renvoie le nombre de nœuds en paramètre
`starts-with(chaine1, sous-chaine2)`	renvoie *true* si le premier argument commence avec le se
`contains(botte_de_foin, aiguille)`	renvoie *true* si le premier argument contient le second
`sum(contexte)`	renvoie la somme des valeurs numériques des nœuds en
`floor(nombre)`	renvoie le nombre arrondi à l'entier inférieur
`ceiling(nombre)`	renvoie le nombre arrondi à l'entier supérieur
`round(nombre)`	renvoie le nombre arrondi à l'entier le plus proche

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tronc nom="tronc1">
    <!-- commentaire 1 -->
    <branche nom="branche1" epaisseur="gros">
        <brindille nom="brindille1">
            <!-- commentaire 2 -->
            <feuille nom="feuille1" couleur="marron" />
            <feuille nom="feuille2" poids="50" />
            <feuille nom="feuille3" />
        </brindille>
        <brindille nom="brindille2">
            <feuille nom="feuille4" poids="90" />
            <feuille nom="feuille5" couleur="violet" />
        </brindille>
    </branche>
    <branche nom="branche2">
        <brindille nom="brindille3">
            <feuille nom="feuille6" />
        </brindille>
        <brindille nom="brindille4">
            <feuille nom="feuille7" />
            <feuille nom="feuille8" />
            <feuille nom="feuille9" couleur="noir" />
            <feuille nom="feuille10" poids="100" />
        </brindille>
    </branche>
    <branche nom="branche3">
        <brindille nom="brindille5">
        </brindille>
    </branche>
</tronc>
```

Toutes les <feuille> de <brindille>:

```
/tronc/branche/brindille/feuille
```

la <branche> dont l'attribut "nom" est "branche3 ".

```
/tronc/branche [@nom= " branche3 "]
```

toutes les brindilles ont au moins une feuille

```
//brindille [feuille]
```

dernière branche du tronc.

```
/tronc/branche[last()]
```

tous les noms des brindilles qui n'ont pas de feuille.

```
//brindille[not(feuille)]/@nom
```

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/css" href="article.css" ?>
<?test cd ghgh ghghh ghgh?>
<article >
<titre > A propos des DTD </titre>
<corps >
<p>
Les DTD, <important>c'est facile</important>, il suffit de lire le
cours de Marc Tommasi <cite ref="siteMarc" />.
</p>
<p >
Il y a aussi de bons livres <cite ref="AmannRigaux" />.
</p>
</corps>
<références >
<référence code="siteMarc">
<nom>Site de Marc Tommasi</nom>
<url>
http://www.grappa.univ-lille3.fr/~tommasi/Homepage/SurDtd.html
</url>
</référence>
<référence code="AmannRigaux">
<intitulé>Comprendre XSLT</intitulé>
<auteur>Bernd Amann</auteur>
<auteur>Philippe Rigaux</auteur>
<date>2002</date>
<editor>O'Reilly</editor>
</référence>
<référence code="AmannRigaux">
<intitulé>Comprendre XSLT</intitulé>
<auteur>Bernd Amann 2</auteur>
<auteur>Philippe Rigaux 2</auteur>
<date>2002</date>
<editor>O'Reilly</editor>
</référence>
</références>
</article>
```

le nœud racine ;

/

tous les descendants de la racine ;

//node()

tous les attributs ;

//@*

tous les nœuds de type texte ;

//text()

les instructions de traitement présentes ;

//processing-instruction()

les contenus de tous les paragraphes;

//p/text()

la référence portant le code *siteMarc*;

//référence[@code='siteMarc']

tous les deuxièmes auteurs.

//*/auteur[2]

Operator	Description
|	Computes two node-sets
+	Addition
-	Subtraction
*	Multiplication
div	Division
=	Equal
!=	Not equal
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
or	or
and	and
mod	Modulus (division remainder)

```
<ROOT>
    <AA>
        <BB/>
    </AA>
    <AA>
        <BB/>
        <BB/>
        <BB/>
    </AA>
    <AA>
        <BB/>
        <BB/>
    </AA>
</ROOT>
```

Exemples de fonctions

last() : retourne ***nombre***

/ROOT/AA/BB[position()=last()]

Récupérer les noeuds qui ont moins de N fils :

/ROOT/AA[BB[last()<2]]

count(*node-set*) : *retourne* ***nombre***

Elle retourne le nombre de noeuds de l'ensemble passé en argument.

count(/ROOT/AA) retounre 3

count(/ROOT/AA/BB) reoutne 6

count(/ROOT/*) retoune 3

```
<ROOT>
    <AA>
        <BB/>
    </AA>
    <AA>
        <BB/>
        <BB/>
        <BB/>
    </AA>
    <AA>
        <BB/>
        <BB/>
    </AA>
</ROOT>
```

name(*node-set* ?) : *retourne* ***string***

Elle retourne une chaîne contenant un nom

/ROOT/AA/*[name()='BB']

contains(*string* , *string*) : *retourne* ***boolean***

Elle retourne true si la première chaîne de caractères passée en argument contient la chaîne de caractères passée en deuxième argument sinon retourne la valeur false.

contains("ABCDE", "BC") retourne true.

contains("ABCDE", "Z") retourne false.

sum(*node-set*) : *retourne* ***nombre***

La fonction **sum** retourne la somme, pour tous les noeuds de l'ensemble passé en argument, du résultat de la conversion en numérique de leur valeur textuelle. Si un ou plusieurs noeuds sélectionnés ne sont pas convertibles la fonction renverra **NaN**. Attention de ne pas oublier que la conversion d'un noeud vide n'est pas zéro mais **NaN**.

```
<ROOT>
     <AA>
     1
     </AA>
     <AA>
     2
     </AA>
</ROOT>
```

sum(/ROOT/AA) retourne 3

not(*boolean*) : *retourne* ***boolean***

La fonction **not** retourne l'inverse de la valeur du booléen passé en argument : vrai (true) si l'argument est faux et vice-versa.

https://erwy.developpez.com/tutoriels/xml/xpath-liste-fonctions/

XSLT

Extensible Style Language Transformation

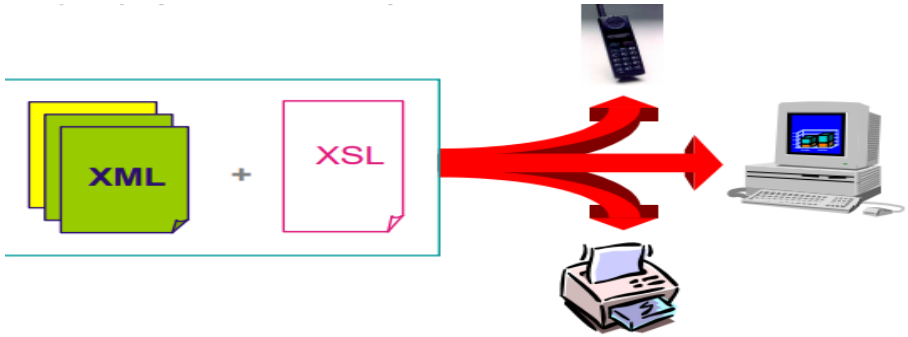
Les feuilles de style du XML

C’est quoi XSLT?

Décrit la manière dont les documents XML seront affichés, imprimés ou ... prononcés

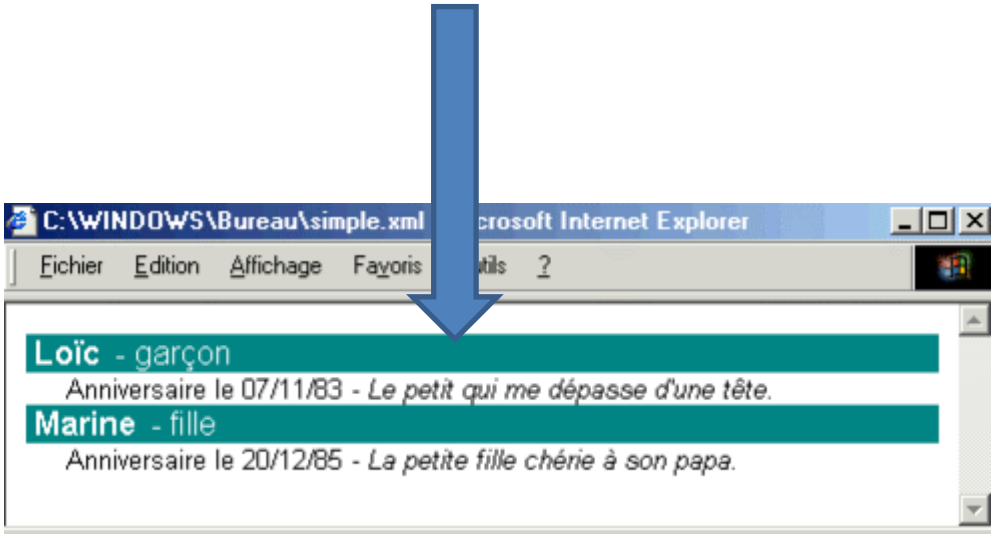
Le XSL est en quelque sorte le langage de feuille de style du XML. Un fichier de feuilles de style reprend des données XML et produit la présentation ou l'affichage de ce contenu XML selon les souhaits du créateur de la page.

Le XSL comporte en fait 3 langages :
° Le XSLT qui est un langage qui Transforme un document XML en un format, généralement en Html, reconnu par un navigateur.
° Le Xpath qui permet de définir et d'adresser des parties de document XML.
° Le XML Formatter pour "formater" du XML (transformé) de façon qu'il puisse être rendu sur des PCpockets ou des unités de reconnaissance vocale.



Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<body style="font-family:Arial; font-size:12pt;">
    <xsl:for-each select="racine/enfant">
    <div style="background-color:teal; color:white;">
    <span style="font-weight:bold; color:white; padding:4px">
    <xsl:value-of select="nom"/></SPAN>
     - <xsl:value-of select="lien"/>
    </div>
    <div style="margin-left:20px; font-size:10pt">
    <span> Anniversaire le <xsl:value-of select="date"/>
    </span>
    <span style="font-style:italic"> - <xsl:value-of select="data"/>
    </span>
    </div>
    </xsl:for-each>
</body>
</html>
```



Pourquoi le XSLT:

Le XSL ne permet pas uniquement l'affichage de XML. Il permet aussi :

- De sélectionner une partie des éléments XML.
- De trier des éléments XML.
- De filtrer des éléments XML en fonction de certains critères.
- De choisir des éléments.
- De retenir des éléments par des tests conditionnels.

Création d'un document XSL:

Liaison du document XSL avec XML:

```
<?xml-stylesheet href="fichierxsl.xsl"?>
```

Le prologue et la racine:

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



< -- Le corps du Document -- >




</xsl:stylesheet>
```

<xsl:template match="/">

L'élément <xsl: template> permet de créer un modèle de document .

L' attribut **match** est utilisé pour associer un modèle à un élément XML.

La valeur de l'attribut match est une expression Xpath.

```
<?xml version="1.0"?>
  <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:template match="/">


< -- le corps du document - - >



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

le corps du document est constitué par des balises html, CSS et des instructions XSL.

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

 <xsl:template match="/">

 <html>

 <body>

                    <les instrcutions  xsl et les balises html -- >



</body>

 </html>

 </xsl:template>

 </xsl:stylesheet>
```

Exemple 1:

```
<?xml version="1.0"?>
 <?xml-stylesheet href="fichierxsl.xsl"?>
 <demoXML>
 <message>Voici du XML</message>
 </demoXML>
```

```
<?xml version="1.0"?>
 <xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
 <xsl:template match="/">
 <html>
 <body>
 <xsl:value-of select="demoXML/message"/>
 </body>
 </html>
 </xsl:template>
 </xsl:stylesheet>
```

Les instructions XSL

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

Permet de récupérer la valeur d'une balise ou plusieurs

Exemple1 : TP0(diapo précedent)

Exemple2 : TP1

```
<td><xsl:value-of select="compilation/mp3/@id"/></td>
```

select="count(compilation/mp3/titre)"

select="." retourne la valeur textuelle de l'élément courant

select="@attrib" retourne la valeur d'un attribut de l'élément courant

xsl:for-each

La fonction xsl:for-each va prendre tous les noeuds d'une requête XPATH, et va leur appliquer un traitement.

```
 </tr>
  <xsl:for-each  select="compilation/mp3">
 <tr>
<td>
<xsl:value-of select="titre"/>
</td>
 </tr>

 </xsl:for-each>
```

TP2

Trier avec le XSL

Le langage XSL permet en quelques mots de trier des données du fichier XML associé en ordre croissant ou décroissant. Ainsi, il suffit d'ajouter l'attribut order="descending" pour trier en ordre croissant et **order="ascending"/>** pour trier en ordre décroissant.

<xsl:sort select=« titre" order="descending"/>
<xsl:sort select="titre" order="ascending"/>

```
 </tr>
  <xsl:for-each  select="compilation/mp3"  >
<xsl:sort select="titre" order="ascending"/>
 <tr>
<td>
<xsl:value-of select="titre"/>
</td>
 </tr>

 </xsl:for-each>
```

Filtrage en xsl

Il est possible de n'extraire qu'une partie du document XML. Dans le select il faut utiliser [*expression*]. L'expression peut comporter des opérateurs, des fonctions, des chemins d'éléments.

Exemple:

```
<?xml version="1.0" ?>
- <bibliotheque>
  - <livre>
      <titre>1984</titre>
    - <auteur>
        <nom>Orwell</nom>
        <prenom>George</prenom>
      </auteur>
      <ref>Fiction-O-1</ref>
    </livre>
  - <livre>
      <titre>N ou M</titre>
    - <auteur>
        <nom>Christie</nom>
        <prenom>Agatha</prenom>
      </auteur>
      <ref>Policier-C-15</ref>
    </livre>
  + <livre>
  + <livre>
  + <livre>
  </bibliotheque>
```

Extraire les livres dont le nom de l'auteur est Christie :

```
<xsl:for-each
select="bibliotheque/livre/auteur[nom='Christi
e']">
<xsl:value-of select="prenom"/><br/>
</xsl:for-each>
```

TP2

Extraire les livres dont le nom de l'auteur est différent de Christie :

```
<xsl:for-each
select="bibliotheque/livre/auteur[nom!='Chris
tie']">
<xsl:value-of select="prenom"/><br/>
</xsl:for-each>
```

Afficher les titres des livres de christie:

```
<xsl:for-each select="bibliotheque/livre[auteur/nom='Christie']">
<xsl:value-of select="titre"/><br/>
 </xsl:for-each>
```

Exercices

```xml
<?xml version="1.0"?>
<compilation>
    <mp3>
        <titre>Foule sentimentale</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Solaar pleure</titre>
        <artiste>MC Solaar</artiste>
    </mp3>
    <mp3>
        <titre>Le baiser</titre>
        <artiste>Alain Souchon</artiste>
    </mp3>
    <mp3>
        <titre>Pourtant</titre>
        <artiste>Vanessa Paradis</artiste>
    </mp3>
    <mp3>
        <titre>Chambre avec vue</titre>
        <artiste>Henri Salvador</artiste>
    </mp3>
</compilation>
```

Afficher la liste des MP3 de la compilation

```
<xsl:for-each  select="compilation/mp3"  >
<b><xsl:value-of select="titre"/>
<xsl:text>----</xsl:text>
<xsl:value-of select="artiste"/> </b> <br/>
</xsl:for-each>
```

Foule sentimentale----Alain Souchon
Solaar pleure----MC Solaar
Le baiser----Alain Souchon
Pourtant----Vanessa Paradis
Chambre avec vue----Henri Salvador

Afficher les infos du premier MP3

```
<xsl:for-each  select="compilation/mp3[1]"  >
<b><xsl:value-of select="titre"/>
<xsl:text>----</xsl:text>
<xsl:value-of select="artiste"/> </b> <br/>
</xsl:for-each>
```

Foule sentimentale----Alain Souchon

Afficher les infos du dernier MP3

```
<xsl:for-each  select="compilation/mp3[last()]"  >
<b><xsl:value-of select="titre"/>
<xsl:text>----</xsl:text>
<xsl:value-of select="artiste"/> </b> <br/>
</xsl:for-each>
```

Afficher le nombre de MP3 dans la compilation

```
<xsl:value-of select="count(compilation/mp3)"/>
```

Pour ajouter une chaine de caractères dans xsl en utilise :

```
<xsl:text> </xsl:text>
```

xsl:if

permet d'exécuter ou non certaines parties du code.

```
<xsl:if test="nom == 'Paul'">
 <xsl:text>Le spécialiste XML</xsl:text>
 </xsl:if>
```

Exemple:

```
<xsl:for-each select="compilation/mp3">
    <xsl:if test="titre='Le baiser' ">
      <tr>
        <td><xsl:value-of select="titre"/></td>
        <td><xsl:value-of select="artiste"/></td>

      </tr>
    </xsl:if>
  </xsl:for-each>
```

xsl:choose

La fonction xsl:choose permet d'exécuter différents codes selon différentes conditions.

```
<xsl:choose>
      <xsl:when test="nom = 'Paul'">
            <xsl:text>Le spécialiste XML</xsl:text>
            </xsl:when>
            <xsl:when test="nom = 'Pierre'">
            <xsl:text>L'infographiste</xsl:text>
      </xsl:when>
            <xsl:when test="nom = 'Pierre'">
                  <xsl:text>L'ergonome</xsl:text>
      </xsl:when>
</xsl:choose>
```

Les fonctions en xsl

Fonction sans paramètre

Pour déclarer une fonction, on utilise la fonction xsl:template.
Exemple :

```
<xsl:template name="hello_world">
  <xsl:text>Hello World !</xsl:text>
</xsl:template>
```

Pour appeler la fonction, on utilise xsl:call-template :

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

Fonction avec paramètres

Pour déclarer des paramètres, on utilise xsl:param

```
<xsl:template name="affiche_somme">
  <xsl:param name="a" select="0" />
  <xsl:param name="b" select="0" />

  <xsl:text>a = </xsl:text>
  <xsl:value-of select="$a" />
  <xsl:text>, b = </xsl:text>
  <xsl:value-of select="$b" />
  <xsl:text>, et a+b = </xsl:text>
  <xsl:value-of select="$a + $b" />
  <xsl:text>.</xsl:text>
</xsl:template>
```

Appel de la fonction avec xsl:call-template :

```
<xsl:call-template name="affiche_somme">
  <xsl:with-param name="a" select="173" />
  <xsl:with-param name="b">9001</xsl:with-
param>
</xsl:call-template>
```

➔ *Affichage obtenu :*

a = 173, b = 9001, et a+b = 9174.

Exercice:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<liste_nombres>
  <nombre valeur="10">dix</nombre>
  <nombre valeur="0">zéro</nombre>
  <nombre valeur="33">trente trois</nombre>
  <nombre valeur="6">le premier nombre
parfait</nombre>
</liste_nombres>
```

```
<?xml version=« 2.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="yes"/>
<xsl:template match="liste_nombres">
  <html>
    <body>
      <p>Liste de nombres :</p>
        <ul>
            <xsl:apply-templates select="nombre" />
        </ul>
    </body>
  </html>
</xsl:template>
<xsl:template match="nombre">
  <li>
    <xsl:value-of select="@valeur"/>
    <xsl:text> : </xsl:text>
    <xsl:value-of select="."/>
  </li>
</xsl:template>
</xsl:stylesheet>
```

Liste de nombres :

- 10 : dix
- 0 : zéro
- 33 : trente trois
- 6 : le premier nombre parfait