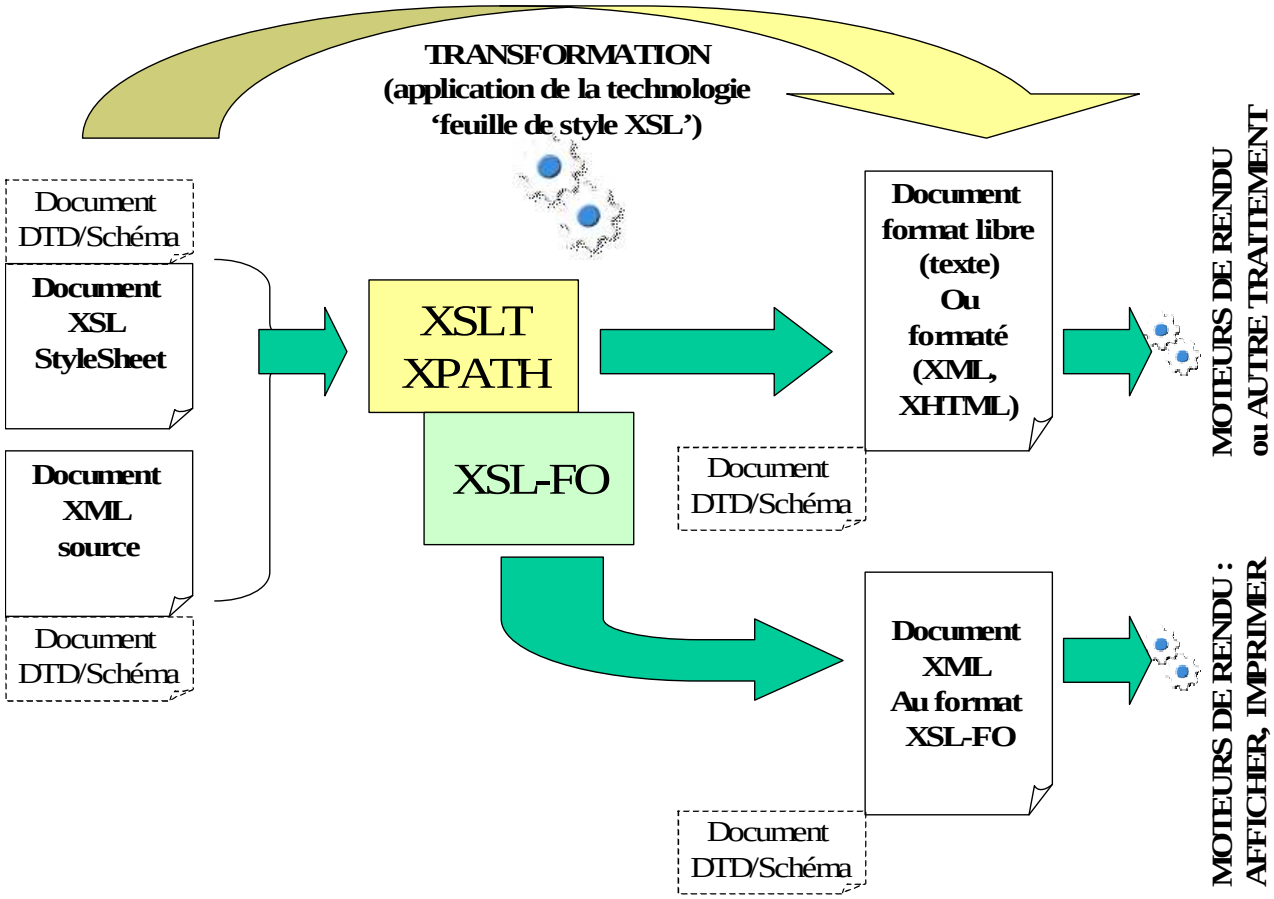
Résumé XSL

XSL, eXtensible Stylesheet Language

1-Définition de XSL

XSL regroupe 3 technologies permettant la **transformation** et la **présentation** des données d'un document XML :

- **XSLT** (XSL Transformation) : langage XML pour transformer le XML
- **XPath** (XML Path language) : langage d'expression permettant de rechercher un ensemble de nœuds (*anglais : nodeset*) dans la structure arborescente d'un document XML
- **XSL-FO** (XSL Formatting Objects) : langage XML permettant de décrire tous les aspects visuels des documents imprimés et affichés. XSL-FO utilise CSS

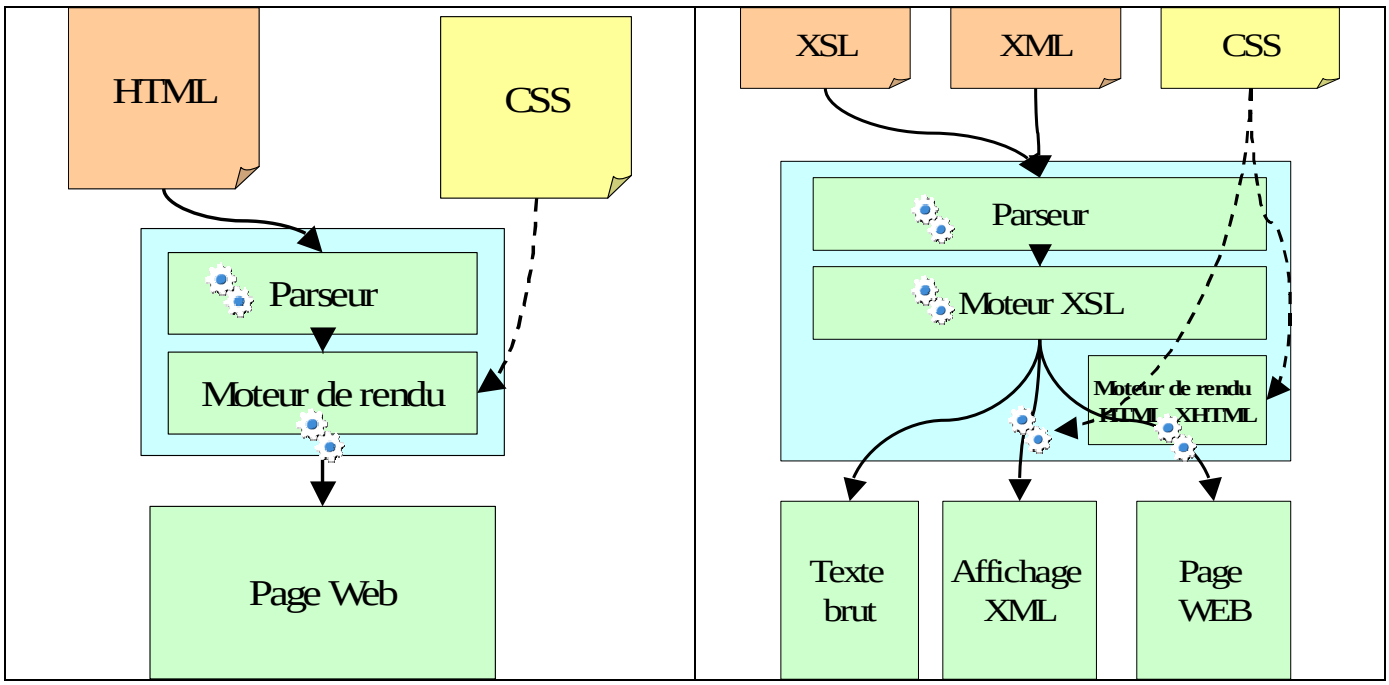


Une fois le document transformé construit (incluant éventuellement les attributs de style nécessaires), c'est un moteur de rendu spécifique (navigateur web ou application de création de documents PDF) qui va rendre, produire, l'aspect final à destination de l'utilisateur.

2-XSL et CSS

Un peu comme le ferait la technologie des CSS, le moteur XSL définit des règles/motifs de mise en forme associées à des éléments XML. Il va cependant beaucoup plus loin et on peut le considérer comme un langage de programmation (interprété) : il inclut en effet des structures algorithmiques (boucles, conditions, etc.), des fonctions (tri, fonctions numériques, etc.).

Résumé XSL



(Les notions de 'parseur', 'moteur...' sont implémentées concrètement sous forme de .DLL)

3-Structure générale d'un document XSL

Un document XSL est une sorte de document XML. Il utilise un jeu de balises spécifiques (les instructions du langage) définies dans l'espace de nom XSLT (XSL Transformation).

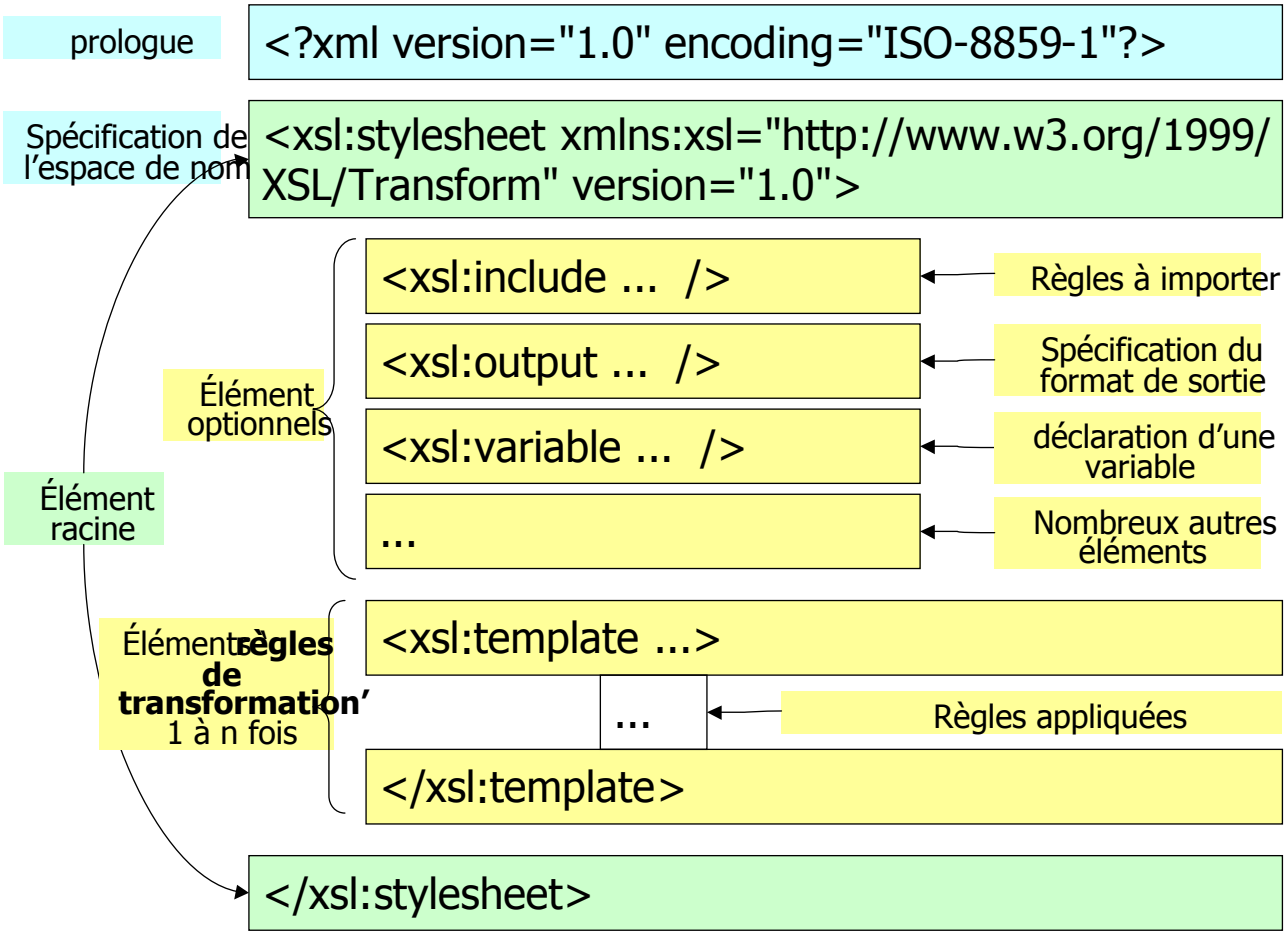


Figure 1 : syntaxe générale d'un feuille de style XSL

4- XSLT, langage pour exprimer des règles de transformations

La **transformation XSLT** (appelée feuille de style XSL) **décrit les règles** (templates) qui vont permettre de **transformer un document XML source en un document résultat**.
Les noms des éléments XSL sont les instructions du langage et seront interprétées par le moteur XSL.
Les **éléments** du document XML source auxquels on applique ces règles sont **sélectionnés** grâce à l'utilisation d'expressions **XPath** (ces expressions permettent de choisir un ensemble d'éléments ou de données à traiter ; on les appelle des **nœuds** dans l'arborescence du document XML)
Le **parcours du document et son traitement** passe par une succession de

- **sélection de nœuds** (expressions XPATH) par rapport à un nœud courant
- **application de règles/motifs** de transformation à ces nœuds (XSLT).

Le point d'entrée du parcours est souvent l'élément racine, nœud de départ. Des nœuds fils sont ensuite sélectionnées à partir de ce nœud courant, afin de leur appliquer des règles de transformations, et ainsi de suite.

A- stylesheet : DEFINIR L'ESPACE DE NOM

C'est **l'élément racine d'une feuille de style XSL**.
On y définit l'**espace de nom** (*anglais : namespace*) : cela permet d'associer un préfixe (en général 'xsl') à un 'dictionnaire', ou référentiel, des noms d'éléments du 'dialecte' XSLT (sa syntaxe).
On peut être amené à utiliser plusieurs dialectes au sein d'un même document : ce préfixe permet alors de définir l'origine de chaque nom d'élément et de supprimer les risques d'ambiguïté de nom d'éléments.

Attribut :

- **xmlns** (XML NameSpace) : définit un alias pour l'espace de nom dont l'adresse de définition suit comme valeur de cet attribut. Ici on dit que toutes les balises relatives à XSL seront préfixées par 'xsl'.

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

Sans motif de transformation, des règles implicitement définies.

B- template : DEFINIR UNE REGLE/UN MOTIF DE TRANSFORMATION

Il est utilisé pour **définir une transformation** pour les nœuds sélectionnés..

Attribut :

- **match** : permet de **définir le filtre de recherche des nœuds** concernés par cette règle ; le filtre est défini grâce à une expression XPATH (/ est associé à la racine du document)

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

C- apply-templates : APPLIQUER UNE REGLE/UN MOTIF DE TRANSFORMATION

Il est utilisé pour **demander l'application d'une règle de transformation** pour les nœuds qui seront sélectionnés.

Attribut :

- **select** : permet de **définir le filtre de recherche des nœuds** concernés par cette règle template' à travers une expression XPATH (/ est associé à la racine du document).

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

- Si l'attribut select n'est pas défini, c'est l'ensemble des nœuds fils du nœud courant qui seront sélectionnés et pour lesquels une règle sera recherchée
- Un second attribut peut être spécifié, il s'agit de l'attribut mode : il permet la définition de plusieurs templates associés à un nom d'élément, mais permet ainsi des traitements différents sur ceux-ci (l'attribut mode est alors également ajouté à l'élément template)

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

D- value-of : PLACER LE CONTENU D'UN NŒUD DANS LE FLUX DE SORTIE

Il est utilisé pour **récupérer le contenu d'un noeud** et le **placer dans le flux de sortie**.

Attribut :

- **select** : permet de **définir le noeud à récupérer** à travers une expression XPATH.
  - Exemple de nœuds (XPATH) :
    - '**name()**' = nom du noeud courant
    - '**position()**' = numéro du noeud courant dans l'arbre traité
    - '**node()** = contenu du nœud courant
    - '**text()**' = contenu du nœud courant
    - '**.**' = contenu du nœud courant

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

ou récupérer la valeur d'un attribut :

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

On peut également récupérer la valeur l'un élément ou attribut en utilisant les accolades :

```
<a href={lien/@url}>texte</a>
```

E- for-each : EFFECTUER UN TRAITEMENT REPETITIF

Il est utilisé pour définir une boucle de traitement

Attribut :

- **select** : permet de **définir les noeuds à traiter dans la boucle**, par rapport au noeud courant

```
. . .
      <xsl:for-each select="entendu_en_concert">
```

```
      . . .
    </xsl:for-each>
. . .
```

F- sort : TRIER LES NOEUDS

A l'intérieur d'un élément **for-each**, après que les noeuds aient été récupérés, cette instruction permet le tri de ces éléments selon un critère.

Attribut :

- **Select** : permet de définir le nom de l'attribut sur lequel le tri va opérer

```
    <xsl:for-each select="entendu_en_concert">
      <xsl:sort select="@date" />
        . . .
    </xsl:for-each>
```

G- if : EFFECTUER UN TRAITEMENT CONDITIONNEL

Ce traitement évalue la validité de contenu de l'attribut 'test' : si cette condition est VRAIE le contenu de l'élément est appliqué. Il n'y a pas de clause SINON.

```
  <xsl:if test="@prix &gt; 10">
    TROP CHER
  </xsl:if>
```

Attribut :

- **test** : expression condition XPATH dont on va tester la validité : VRAI ou FAUX

H- choose, when, otherwise : CHOISIR UN TRAITEMENT A EFFECTUER SELON DES CONDITIONS MULTIPLES

Ce traitement permet d'effectuer un traitement selon une sélection de choix multiple.

```
  . . .
  <xsl:choose>
    <xsl:when test="@prix &gt; 99">
      . . .
    </xsl:when>
    <xsl:when test="@prix &gt; 10">
      . . .
    </xsl:when>
    <xsl:when test="@prix &lt; 5">
      . . .
    </xsl:when>
    <xsl:otherwise>
      . . .
```

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

I- comment : CONSTRUIRE UN COMMENTAIRE XML DANS LE FLUX DE SORTIE

Il est utilisé pour **définir un commentaire** et le **placer dans le flux de sortie**.

```
. . .
      <xsl:comment>
        Ceci est un commentaire qui sera inséré dans le flux de sortie
      </xsl:comment>
. . .
```

G- element : CONSTRUIRE UN ELEMENT XML DANS LE FLUX DE SORTIE

Il est utilisé pour **construire un élément au sens XML** et le **placer dans le flux de sortie**.

```
. . .
  <xsl:element name="img">
    . . .
  </xsl:element>. . .
```

Attribut :

- **name** : permet de **définir le nom de l'élément à créer**.

H- attribute : CONSTRUIRE UN ATTRIBUT AU SEIN D'UN ELEMENT XML DANS LE FLUX DE SORTIE

Il est utilisé pour **construire un attribut associé à un élément** et le **placer dans le flux de sortie**.

```
. . .
  <xsl:element name="img">
    <xsl:attribute name="src">
      compil.jpg
    </xsl:attribute>
    . . .
  </xsl:element>. . .
```

Attribut :

- **name** : permet de **définir le nom de l'élément à créer**.

E- copy : RECOPIER UN NOEUD DANS LE FLUX DE SORTIE

Il est utilisé pour **récupérer le contenu d'un noeud** et le **placer dans le flux de sortie**.

Attribut :

- **select** : permet de **définir le noeud à récupérer** à travers une expression XPATH.

F- copy-of : RECOPIER UN NOEUD ET SES FILS DANS LE FLUX DE SORTIE

Il est utilisé pour **récupérer le contenu d'un nœud et de ses descendants** et le **placer dans le flux de sortie**.

Attribut :

- **select** : permet de **définir le noeud à récupérer** à travers une expression XPATH.

G- call-template et param : APPELER UN MOTIF NOMME ET PASSER DES PARAMETRES

Il est utilisé pour **récupérer le contenu d'un nœud et de ses descendants** et le **placer dans le flux de sortie**.

Définition d'un motif nommé (non lié à un nom d'élément) :

```
<xsl:template name="faireunlien">
    <xsl:param name="url" />
    <xsl:param name="texte" />
    <a href="{$url}">
        <xsl:value-of select="$texte" />
    </a>
</xsl:template>
```

Appel du motif, avec passage des paramètres par nom :

```
. . .
    <xsl:call-template name="faireunlien">
        <xsl:with-param      name="texte"      select="concat(prénom,'
',nommarital,' ',nom)" />
        <xsl:with-param name="url"   select="@adresse" />
    </xsl:call-template>
. . .
```

5- XPATH

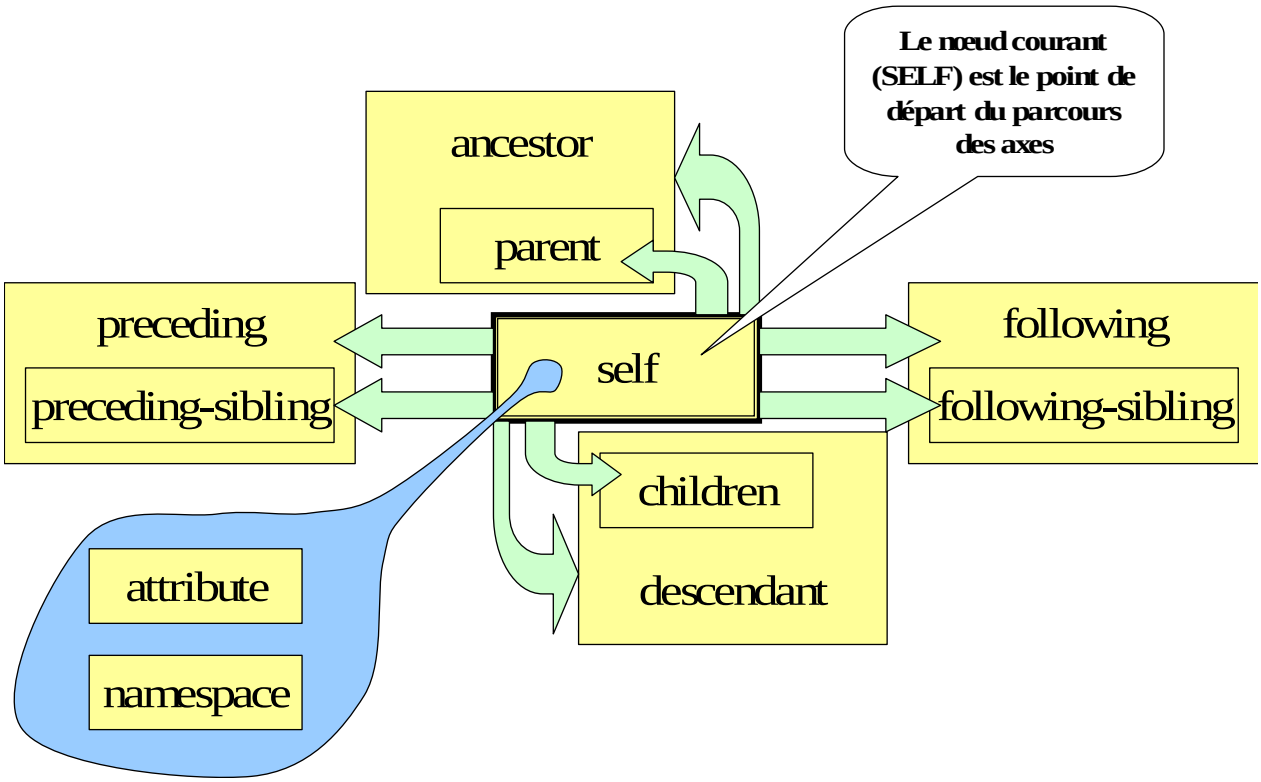
XPath est un langage basé sur des expressions qui vont permettre, appliquées à un document XML, de retourner des valeurs de type :

- **une collection de noeuds** (*anglais : nodeset*) pour application des motifs de transformation, avec la possibilité de **définir des prédicats de filtrage de certains noeuds** selon une valeur d'attribut ou d'élément, par exemple
- **une valeur unique** :
    - chaînes de caractères : soit valeurs d'éléments ou d'attributs, ou bien résultant d'application de fonctions sur ces valeurs
    - nombres : soit valeurs d'éléments ou d'attributs, ou bien résultant d'application de fonctions sur ces valeurs
    - booléens : test d'existence d'un nœud, d'un attribut, de comparaison de valeurs

A- XPATH, expression de sélection de noeuds selon un axe de recherche

Le langage XPATH permet tout d'abord de sélectionner des noeuds par rapport à un noeud courant en spécifiant un chemin par rapport à ce nœud (un peu comme on pourrait le faire pour le parcours d'une arborescence de répertoires).

Résumé XSL



Ce parcours est celui d'un arbre constitué par des noeuds selon des **axes** et les relations entre ces nœuds :

- **self** : le nœud courant, l'endroit d'où l'on part
- **parent** : le père du noeud courant
- **child** : les enfants du noeud courant
- **descendant** : L'axe des descendants (enfants et leurs descendants)
    - **descendant-or-self** : inclut le noeud courant
- **ancestor** : L'axe des ancêtres (parents et leurs ancêtres)
    - **ancestor-or-self** : inclut le noeud courant
- sibling : axe des frères (ceux qui ont le même parent) : **preceding-sibling** et **following-sibling**
- **preceding** : les noeuds qui précèdent
- **following** : les noeuds qui suivent
- **attribute** : les attributs du noeud courant
- **namespace**: les espaces de nom

Exemples de notations :

*****	Tous les noeuds
.	Nœud courant
..	Nœud parent
./*	Tous les nœuds fils du nœud courant
.//*	Tous les nœuds descendants du nœud courant
@nom	Attribut nom
cd	Tous les nœuds 'cd' fils du nœud courant
/compil/cd	Tous les noeuds 'cd' à partir du noeud 'compil' à la racine

//titre	Tous les nœuds 'titre' où qu'ils soient
//@nom	Tous les nœuds attribut 'nom' où qu'ils soient
child::cd	Tous les noeuds fils du nœud courant possédant l'attribut prix
attribute::prix	Tous les noeuds fils possédant l'attribut prix

A partir du nœud courant, on considère :
- Un **axe** : la direction vers laquelle la recherche va s'effectuer
- Un **filtre** : le type de nœud à sélectionner dans l'axe
- Un **prédicat** : condition supplémentaire pour sélectionner les noeuds

Exemple :
➔ axe ::filtre[predicat]
➔ axe1 ::filtre1[predicat1]/axe2 ::filtre2[predicat2]
Un chemin commençant par / indique un chemin absolu (à partir de la racine)

Il est possible d'effectuer l'union de 2 ensembles de nœuds avec l'opérateur | :
➔ parcours1 | parcours2
où parcours est une expression Xpath

B- XPATH, expressions de filtres des noeuds

Une fois l'axe de parcours choisi, XPATH propose une syntaxe permettant d'appliquer un filtre sur ces nœuds en utilisant une expression du type :

```
Axe ::noeud
```

où 'expression' est, par exemple :
- **node()** : tous les noeuds
- **text()**-1 : les nœuds textuels
- ***** : tous les éléments
- **comment()** : les nœuds commentaires
- **processing-instruction()** : les nœuds instruction
- nom : les éléments portant ce nom

C- XPATH, prédicats

Une fois l'axe de parcours choisi, XPATH propose une syntaxe permettant d'appliquer un filtre sur ces nœuds en utilisant une expression du type :

```
Axe ::noeud[expression]
```

où 'expression' est, par exemple :
- **last()** : le dernier
- **last()**-1 : l'avant-dernier
- **first()** : de premier
- **position()**<=3 : les 3 premiers
- prix : qui possèdent un élément prix
- prix<100 : qui possèdent un élément prix dont la valeur est inférieur à 100
- @attribut : qui possèdent l'attribut de nom attribut
- @attribut=3 : qui possèdent l'attribut de nom attribut de valeur 3

/compil/cd[@prix<100]	Tous les noeuds 'cd' à partir du noeud 'compil' à la racine dont la valeur de l'attribut prix est inférieure à 100

/compil/cd[@prix<100]/titre	Tous les nœuds 'titre' des noeuds 'cd' à partir du noeud 'compil' à la racine dont la valeur de l'attribut prix est inférieure à 100
/compil/cd/piste[last()]	Le dernier élément piste de chaque cd

D- XPATH, expression de calculs, fonctions et comparaisons

Le langage XPATH offre également la possibilité d'évaluer l'application d'opérateurs et de fonctions sur des valeurs de nœuds (élément, attributs, etc.) .

Exemple de fonctions XPATH

Fonction	Description
number(@prix)	Evaluer une expression arithmétique
sum(@montant)	Effectuer une somme
count(/descendant::piste)	Nombre de nœuds dans l'ensemble sélectionné
position()	Position du nœud dans l'ensemble de noeuds
last()	Le dernier nœud dans l'ensemble de noeuds
substring(@lib,1,2)	Extraire une sous-chaîne d'une chaîne
upper-case(@lib)	La valeur de l'attribut lib en majuscule
concat	Concaténation des chaînes
String-length	Longueur d'une chaîne
Contains, starts-with, ends-with	Test si contient, débute par ou se termine par une chaîne donnée
Name()	Nom de l'élément courant

Opérateurs arithmétiques : retourne le résultat de l'opération

Opérateur	Description	Exemple	Valeur retournée
+	Addition	6 + 4	10
-	Soustraction	6 - 4	2
*****	Multiplication	6 * 4	24
div	Division	8 div 4	2
mod	Modulo (reste de la division)	5 mod 2	1

Opérateurs de comparaison : retourne true ou false en fonction de la comparaison

Opérateur	Description	Exemple
=	égal	prix=1.5
!=	différent	prix!= 1.5
<	Plus petit que	prix<1.5
<=	Plus petit ou égal	prix<=1.5
>	Plus grand que	prix>1.5
>=	Plus grand ou égal	prix>=1.5

Opérateurs logiques : combinaison d'opérateurs de comparaison

Opérateur	Description	Exemple
or	ou	Prix<2.5 or prix>10
and	et	prix>=2.5 and prix<=10
Not	non	

E- XPATH, accès à un document XML

Un document XML peut être accédé à partir d'une feuille de style XSL :

```
...
    <xsl:variable name="extNode"
select="document('../album.xml')/album"/>
    <xsl:variable name="numPhotos" select="$extNode/@numPhotos"/>
    . . .
    <xsl:for-each  select="$extNode/photo[position()  mod  $numColumns  =
1]">
    . . .
    </xsl:for-each>
```

6-Appel des transformations XSLT, implémentation moteur XSLT

La plupart des navigateurs Web implémentent un moteur XSLT pour produire dynamiquement un résultat affiché.
Il est cependant souvent intéressant de produire un document mémorisé en sortie. Cela fait appel à un moteur XSLT externe.

Il est possible de passer des paramètres à partir d'un fichier de commandes pour modifier le comportement d'application des règles :

Définition des paramètres dans le document XSL (en début de programme XSLT, après la balise de début xsl-stylesheet :

```
<xsl:param name="par1" />
<xsl:param name="par2" />
    . . .règles de transfiormation utilisant la valeur du parameter .
. .
```

Fichier de commande (où xsltproc est la commande à lancer pour invoquer le moteur XSLT) :

```
xsltproc --stringparam par1 valeur1 --stringparam par2 valeur2 doc.xsl
doc.xml > doc.html
```

7-XSLT comme langage

On place XSLT dans la famille des langages fonctionnels (les variables ne sont pas modifiables dans le cours du programme, pas d'effet de bord)

8-Références sur Internet

http://www.w3c.org : les normes relatives à XML et ses dialectes, des liens vers des traductions françaises et vers les outils qui exploitent les dialectes XML.

Résumé XSL