SUP MTI
École Supérieure de Management, de Télécommunication et d'Informatique

W3C®

eXtensible Stylesheet Language Transformation

Ahmed ZELLOU
Ahmed.zellou@um5.ac.ma

Sup-MTI, 2025-2026.

XSL

- XSL (Extensible Stylesheet Language) est un langage de feuille de style pour XML.
- Il décrit comment un document XML sera affiché sur un dispositif.
- Utilisé essentiellement pour transformer des documents XML en d'autres formats, comme XHTML.
- XSL se compose de trois parties:
  - XSLT : permet de transformer un document XML en d'autres formats, comme XHTML.
  - Xpath : permet de naviguer dans un document XML.
  - XSL-FO (formatting objects) : permet de formater un document XML en PDF.

Introduction

- XSLT est un langage pour transformer des documents XML en XHTML ou en d'autres documents XML.
  - Signifie XSL transformation
  - Le standard le plus important de XSL
  - Utilise XPath pour naviguer dans des documents XML
  - Recommandation W3C.
  - Trois versions : 1.0 (1999), 2.0 (2007) est 3.0 depuis 2017.

- XSLT transforms an XML source-tree into an XML result-tree.

Introduction

- Xpath est Utilisé pour naviguer à travers les éléments et les attributs des documents XML.

- XSLT Permet de :
  - Ajouter/supprimer des éléments et des attributs du fichier résultant.
  - Réorganiser les éléments dans le fichier résultant.
  - Effectuer des tests sur le fichier source
  - Boucler sur les éléments et les attributs
  - Trier les éléments dans le fichier résultant.
  - Créer des nouveaux éléments, attributs, variables, commentaires,…

Transformation XSLT

- Espace de noms
  - xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

Prenons un document XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Biblio>
  <Livre num="L1">
                    <titre>Le monde des Web Services</titre>
                    <auteur>Ahmed ZELLOU</auteur>
                    <pays>Maroc</pays>
                    <editeur>Eyrolles</editeur>
                    <prix>120.00</prix>
                    <annee>2011</annee>
  </Livre>
..
</Biblio>
```

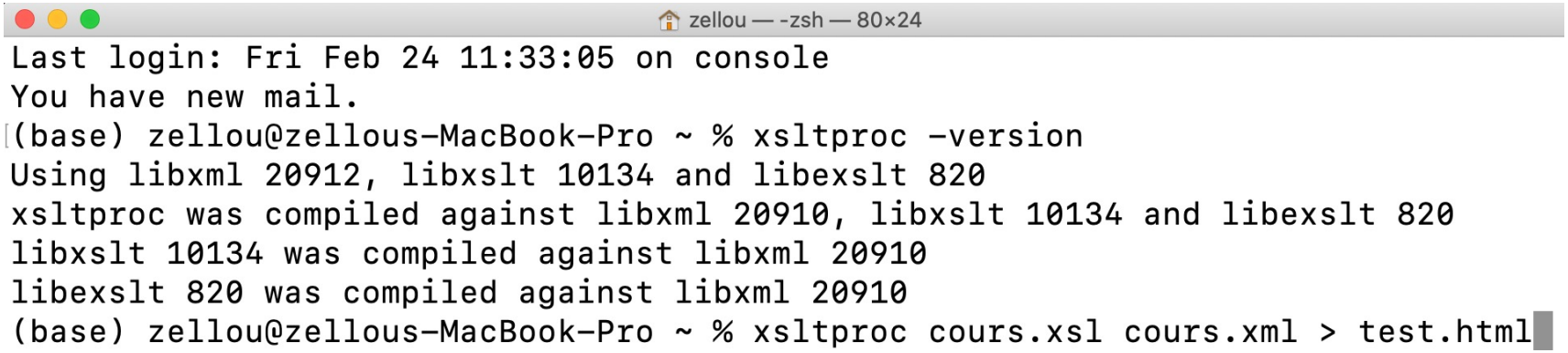
Transformation XSLT

- Liaison avec le fichier XML :
  - Intégrer la référence XSL à votre document XML ("docCatalog.xml") et afficher le résultat via un navigateur

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="docCatalog.xsl"?>
<Biblio>
  <Livre num="L1">
                    <titre>Le monde des Web Services</titre>
                    <auteur>Ahmed ZELLOU</auteur>
                    <pays>Maroc</pays>
                    <editeur>Eyrolles</editeur>
                    <prix>120.00</prix>
                    <annee>2011</annee>
  </Livre>
..
</Biblio>
```

Interprétation XSLT

- La plupart des navigateurs supportent XSLT.
- Sous Oxygen, créer un scénario d'exécution.
- xsltproc est un outil (en ligne de commande) pour réaliser la transformation
  - xsltproc file.xsl file.xml > test.html



Transformation XSLT

- Pour transformer un fichier XML en XHTML via XSLT :

- Déclaration d'une feuille de style
  - L'élément racine d'un document XSL est <xsl:stylesheet> ou <xsl:transform>.
  - Note: <xsl:stylesheet> et <xsl:transform> sont complètement synonymes.

- Pour déclarer une feuille de style XSL conformément à la recommandation W3C XSLT:
  - <xsl:stylesheet version="3.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  - Ou:
  - <xsl:transform version="3.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

Transformation XSLT

- Une feuille de style XSL se compose d'un ou de plusieurs règles nommées templates (Modèles).
- L'élément <xsl:template> est utilisé pour construire un modèle d'affichage.
- Le template contient des règles qui s'appliquent quand un nœud spécifique est trouvé.
- L'attribut match est utilisé pour associer le modèle à un élément XML.
- Cet attribut peut aussi être utilisé pour définir un modèle pour l'ensemble du document XML (match = "/").

L'élément <xsl:template>

- Prenons une version simplifiée du fichier XSL :

```
<?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>
     <h2>Ma Collection de Livres </h2>
     <table border="1">
      <tr bgcolor="#9acd32">
       <th>Titre</th>
       <th>Auteur</th>
      </tr>
      <tr>
       <td>.</td>
       <td>.</td>
      </tr>
     </table>
     </body>
     </html>
    </xsl:template>
    </xsl:stylesheet>
```

Une feuille de style XSL est un document XML, il commence toujours par le prologue

<xsl:stylesheet> indique que ce document est une feuille de style XSLT.

<xsl:template> définit un modèle, l'attribut match = "/" associe le modèle avec la racine du document XML source.

Le contenu à l'intérieur de l'élément <xsl:template> définit des balises HTML pour le document résultant.

Définissent la fin du modèle et la fin de la feuille de style.

<xsl:value-of >

- Entrée

```
<note>enseigne <mat>XML</mat></note>
```

- Script

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

- Arbre de Sortie

enseigne XML

L'élément text()

- Entrée

```
<note>enseigne <mat>XML</mat></note>
```

- Script

```
<xsl:template match="note">
    <xsl:value-of select="text()"/>
</xsl:template>
```

- Arbre de Sortie

enseigne

- Seul le premier élément sélectionné est produit

L'élément <xsl:value-of>

- Utilisé pour extraire la valeur d'un nœud :

```
<?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>
     <h2>Ma Collection de Livres </h2>
     <table border="1">
      <tr bgcolor="#9acd32">
       <th>Titre</th>
       <th>Auteur</th>
      </tr>
      <tr>
       <td><xsl:value-of select="Biblio/Livre/titre"/></td>
       <td><xsl:value-of select="Biblio/Livre/auteur"/></td>
      </tr>
     </table>
     </body> </html>
    </xsl:template>
    </xsl:stylesheet>
```

La valeur de l'attribut select est une expression XPath.

Une expression XPath fonctionne comme une navigation dans un système de fichiers, où une barre oblique (/) sélectionne les sous-répertoires.

L'élément <xsl:text>

- XSL ne supporte pas les caractères d'espacement:
  - Seulement cinq entités prédéfinies, et non
  - Permet d'injecter des espaces, des tabulations ou des fins de ligne dans le document de sortie
  - Permet aussi d'injecter des textes libres dans une sortie

```
<xsl:text  disable-output-escaping="yes">      </xsl:text>
```

L'élément <xsl:for-each>

- Utilisé pour boucler sur les éléments XML d'un ensemble de nœuds.

```
<?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>
      <h2>Ma Collection de Livres </h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Titre</th> <th>Auteur</th>
        </tr>
  <xsl:for-each select="Biblio/Livre">
        <tr>
            <td><xsl:value-of select="titre"/></td>
            <td><xsl:value-of select="auteur"/></td>
        </tr>
 </xsl:for-each>
      </table> </body> </html>
    </xsl:template> </xsl:stylesheet>
```

L'élément <xsl:sort>

- Utilisé pour trier le résultat à l'intérieur de l'élément <xsl:for-each>

```
<?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>
      <h2>Ma Collection de Livres </h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Titre</th> <th>Auteur</th>
        </tr>
   <xsl:for-each select="Biblio/Livre">
   <xsl:sort select="auteur"/>
      <tr>
          <td><xsl:value-of select="titre"/></td>
          <td><xsl:value-of select="auteur"/></td>
        </tr>
  </xsl:for-each>
      </table></body> </html>
    </xsl:template>
</xsl:stylesheet>
```

L'élément <xsl:sort>

- Quelques attributs de <xsl:sort> :
  - order : le sens du tri (ascending ou descending)
  - lang : la langue du critère de tri
  - data-type : la nature de la clé de tri (text, number)
  - case-ordre (upper-first, lower-first)

Les tests

- Filtrer le résultat
  - Pour filtrer le résultat, on ajoute un critère pour sélectionner l'attribut dans l'élément <xsl:for-each>.

- Exemple :
  - <xsl:for-each select="Biblio/Livre[auteur='Choukri']">

- Les opérateurs de filtrage autorisés sont:
  - = (égal)
  - ! = (Différent)
  - < inférieur à
  - > supérieur à

Les tests

- Filtrer le résultat : 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>
     <h2>Ma Collection de Livres </h2>
     <table border="1">
      <tr bgcolor="#9acd32">
       <th>Titre</th> <th>Auteur</th>
      </tr>
  <xsl:for-each select="Biblio/Livre[auteur='Choukri']"> <tr>
        <td><xsl:value-of select="titre"/></td>
        <td><xsl:value-of select="auteur"/></td>
      </tr>
 </xsl:for-each>
     </table>
</body>
</html>
    </xsl:template>
</xsl:stylesheet>
```

L'élément <xsl:if>

- Utilisée pour mettre un test conditionnel sur le contenu du fichier XML.

- Syntaxe

```
<xsl:if test="expression">
  ... résultat si le test est vrai ...
</xsl:if>
```

- Où le mettre ?
  - A l'intérieur de l'élément <xsl:for-each> dans le fichier XSL.

L'élément <xsl:if>

- La valeur de l'attribut test contient l'expression à évaluer.

```
<?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>
      <h2>Ma Collection de Livres </h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Titre</th> <th>Auteur</th>
        </tr>
   <xsl:for-each select="Biblio/Livre">
   <xsl:if test="prix &gt; 10">
      <tr>
          <td><xsl:value-of select="titre"/></td>
          <td><xsl:value-of select="auteur"/></td>
        </tr>
    </xsl:if>
  </xsl:for-each>
      </table></body> </html>
    </xsl:template>
</xsl:stylesheet>
```

L'exemple affiche le titre et l'auteur des livres qui ont un prix supérieur à 10.

L'élément <xsl:if>

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>
  <h2>Ma Collection de Livres </h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Titre</th>  <th>Auteur</th>
    </tr>
    <xsl:for-each select="Biblio/Livres">
    <tr>
      <td><xsl:value-of select="titre"/></td>
      <xsl:if test="prix &gt; 10">
        <td bgcolor="#ff00ff"><xsl:value-of select="auteur"/></td>
      </xsl:if>
      <xsl:if test="prix &le; 10">
        <td bgcolor="#00ff00"><xsl:value-of select="auteur"/></td>
      </xsl:if>
    </tr>
    </xsl:for-each>
  </table> </body> </html>
</xsl:template> </xsl:stylesheet>
```

Cet exemple ajoute un fond rose à la colonne "Auteur" lorsque le prix du livre est supérieur à 10.

L'élément <xsl:choose>

- Utilisé en conjonction avec <xsl:when> et <xsl:otherwise> pour exprimer des tests conditionnels multiples.

- Syntaxe

```
<xsl:choose>
  <xsl:when test="expression">
    ... Éléments de sortie ...
  </xsl:when>
  <xsl:otherwise>
    ... Éléments de sortie ....
  </xsl:otherwise>
</xsl:choose>
```

L'élément <xsl:choose>

- Exemple avec when et otherwise :

```
<?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>
     <h2>Ma Collection de Livres </h2>
     <table border="1">
      <tr bgcolor="#9acd32">
       <th>Titre</th> <th>Artiste</th> </tr>
      <xsl:for-each select="Biblio/Livres">
      <tr>
       <td><xsl:value-of select="titre"/></td>
       <xsl:choose>
        <xsl:when test="prix &gt; 10">
         <td bgcolor="#ff00ff"> <xsl:value-of select="auteur"/></td>
        </xsl:when>
        <xsl:when test="prix &gt; 9">
         <td bgcolor="#0000ff"> <xsl:value-of select="auteur"/></td>
        </xsl:when>
        <xsl:otherwise>
         <td><xsl:value-of select="artiste"/></td>
        </xsl:otherwise>
       </xsl:choose>
      </tr>
      </xsl:for-each>
     </table> </body> </html>
    </xsl:template></xsl:stylesheet>
```

L'exemple ajoute un fond rose à la colonne "Auteur" lorsque le prix du livre est supérieure à 10, et un fond gris lorsque le prix du livre est supérieur à 9 et inférieur ou égal à 10.

Types de Sortie

- Trois types de document en sortie :
  - xml : sortie par défaut
    <xsl:output method="xml">
  - html :
    <xsl:output method="html">
  - text : du code natif, RTF, LaTex
    <xsl:output method="text">
- Par défaut, xml, sauf si la première balise de l'arbre résultat est html.

L'élément <xsl:element>

- Permet créer un élément (Transformation)

```
<xsl:output method="xml" indent="yes"/>

<xsl:template match="chemin">
   <xsl:element name="nom">
      <xsl:value-of select="valeur"/>
   </xsl:element>
</xsl:template>
```

L'élément <xsl:attribute>

- Permet de créer un attribut

```
<xsl:output method="xml" indent="yes"/>

<xsl:template match="chemin">
      <xsl:element name="nom">
            <xsl:value-of select="valeur"/>
            <xsl:attribute name="nom">
                  <xsl:value-of select="valeur"/>
            </xsl:attribute>
      </xsl:element>
</xsl:template>
```

L'élément { }

- Entrée

```
<a href="fic.txt"/>
```

- Script

```
<xsl:template match="a">
  <b id="{@href}"/>
</xsl:template>
```

- Arbre de Sortie

```
<b id="fic.txt"/>
```

- Autre script

```
<xsl:template match="a">
    <b><xsl:attribut name="id">
      <xsl:value-of select="@href"/>
    </xsl:attribut></b>
</xsl:template>
```

L'élément name()

- Entrée

```
<note>enseigne
    <mat>XML</mat>
 </note>
```

- Script

```
<xsl:template match=".">
    <xsl:value-of select="name()"/>
</xsl:template>
```

- Arbre de Sortie

note

L'élément <xsl:variable>

- Déclaration de variable 1

```
<xsl:variable name="backcolor" select="'#FFFFCC'" />
```

- Déclaration de variable 2

```
<xsl:variable
name="backcolor">#FFFFCC</xsl:variable>
```

- Référence à une variable

```
<BODY BGCOLOR='{$backcolor}'>
```

- N.B. Une variable ne peut être réaffectée (des constantes)

Passage de paramètres

- Déclaration (paramètre formel)

```
<xsl:template match=".">
    <xsl:param name="p" select="0"/>
    ... utilisation de p ...
</xsl:template>
```

- Obtenir la valeur d'un paramètre

```
select="$p"
```

- Appel (paramètre effectif)

```
<xsl:apply-templates>
    <xsl:with-param name="p" select="$p+1"/>
</xsl:apply-templates>
```

Passage de paramètres

- Les paramètres descendants directs d'un élément <xsl:stylesheet> sont autorisés

```
<xsl:stylesheet>
    <xsl:param name="dir" select="'mondir'"/>
    ...
</xsl:stylesheet>
```

- Equivalent à

dir='monDir'

L'élément <xsl:number>

- Pour numéroter des nœuds d'un document en fonction de leur position dans l'arbre source :
  - level : pour préciser le niveau dans l'arbre
  - from : point de départ de la numérotation
  - format : format d'affichage de la numérotation

L'élément <xsl:message>

- Déclaration

```
<xsl:message>
     code = <xsl:value-of select="$code"/>
</xsl:message>
```

- La Sortie

code = 25

L'élément <xsl:copy-of>

- Crée une copie d'un noeud

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:variable name="header">
                <tr bgcolor="#9acd32">
                        <th align="left">Title</th>
                        <th align="left">Artist</th>
                </tr>
        </xsl:variable>
        <xsl:template match="/">
                <html>
                        <body>
                        <h2>Ma Collection de livres</h2>
                        <table border="1">
                                <xsl:copy-of select="$header"/>
                                <xsl:for-each select="Biblio/Livre/titre">
```

<xsl:comment> & <xsl:import>

- Pour sortir les commentaires

```
<xsl:template match="comment()">
  <xsl:comment>
    <xsl:value-of select="."/>
  </xsl:comment>
</xsl:template>
```

- Modularité des feuilles de style

```
<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" indent="yes"/>
    <xsl:import href="feuille1.xsl"/>
...
</xsl:stylesheet>
```

Divers

- Modulariser sa stylesheet
  - `<xsl:include>`
    - Inclut textuellement la stylesheet référencée.
  - `<xsl:include href = "uri-reference" />`
    - Même niveau de précédence : les templates se comportent comme ceux de la stylesheet.

- Gérer les espaces
  - `<xsl:strip-space>`
    - ne préserve pas les espaces dans les éléments
  - `<xsl:preserve-space>`
    - préserve les espaces dans les éléments

Divers

- `<xsl:key>`
  - permet d'associer un identifiant à un nœud
- `<xsl:decimal-format>`
  - permet de définir un modèle de représentation des nombres décimaux (utile si internationalisation)
- `<xsl:attribute-set>`
  - déclare un ensemble d'attributs réutilisables

Les Templates

- **Deux styles de programmation**
  - Itérative via l'attribut select
    <xsl:for-each>
  - Récursive
    <xsl:apply-templates>
- Déclaration

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

- Appel

<xsl:call-template> ou <xsl:apply-templates>

L'élément <xsl:apply-templates>

- L'élément <xsl:apply-templates> applique un modèle.
- L'attribut select <xsl:apply-templates>, précise à qui s'applique le modèle.
- Nous pouvons utiliser l'attribut **select** pour spécifier l'ordre dans lequel les nœuds fils seront traités.

Les Templates

- **Exemple**

```
<xsl:template name="navbar">
      <a href="index.htm">Home</a> |
      <a href="help.htm">Help</a> |
      <a href="toc.htm">Contents</a>
</xsl:template>

<xsl:template match="mainblock">
      <body>
            <xsl:call-template name="navbar"/>
            <xsl:apply-templates/>
            <xsl:call-template name="navbar"/>
      </body>
</xsl:template>
```

L'élément <xsl:apply-templates>

- Prenons l'exemple suivant:

```
<?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>
 <h2>Ma Collection de Livres </h2>
 <xsl:apply-templates/>
 </body>
 </html>
</xsl:template>
<xsl:template match="catalogue/cd">
 <p>
 <xsl:apply-templates select="titre"/>
 <xsl:apply-templates select="artiste"/>
 </p>
</xsl:template>
<xsl:template match="titre">
 Title: <span style="color:#ff0000">
 <xsl:value-of select="."/></span>
 <br />
</xsl:template>
<xsl:template match="artiste">
 Artist: <span style="color:#00ff00">
 <xsl:value-of select="."/></span>
 <br />
</xsl:template></xsl:stylesheet>
```

Ce qui donne comme résultat

Ma Collection de CD

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Title: Still got the blues
Artist: Gary Moore

Title: Eros
Artist: Eros Ramazzotti

Title: One night only
Artist: Bee Gees

Title: Sylvias Mother
Artist: Dr.Hook

Title: Maggie May
Artist: Rod Stewart

XSLT - référence

Élément	Description	IE	FF
apply-imports	Applique une règle modèle à partir d'une feuille de style importé	6.0	1.0
apply-templates	Applique une règle modèle à l'élément courant ou au nœud fils de l'élément courant	5.0	1.0
attribute	Ajoute un attribut	5.0	1.0
attribute-set	Définit un ensemble nommé d'attributs	6.0	1.0
call-template	Appelle un modèle nommé	6.0	1.0
choose	Utilisé en conjonction avec <when> et <otherwise> pour exprimer des tests conditionnels multiples	5.0	1.0
comment	Crée un nœud de commentaire dans l'arbre résultat	5.0	1.0
copy	Crée une copie du noeud courant (sans nœud fils et sans attributs)	5.0	1.0
copy-of	Crée une copie du noeud courant (avec nœud fils et attributs)	6.0	1.0
decimal-format	Définit les caractères et les symboles à utiliser pour convertir les nombres en chaînes de caractères avec la fonction format-number()	6.0	1.0
element	Crée un nœud d'élément dans le document résultant	5.0	1.0
fallback	Spécifie un code alternatif à exécuter si le processeur ne supporte pas un élément XSLT	6.0	

XSLT - référence

Élément	Description	IE	FF
for-each	Boucle sur chaque noeud dans un ensemble de noeud spécifié	5.0	1.0
if	Contient un modèle qui sera appliqué si une condition spécifiée est vraie	5.0	1.0
import	Importe le contenu d'une feuille de style dans une autre. Remarque: Une feuille de style importée est moins prioritaire que la feuille de style principale	6.0	1.0
include	Inclut le contenu d'une feuille de style dans une autre. Remarque: Une feuille de style incluse à la même priorité que la feuille de style principale	6.0	1.0
key	Déclare une clef nommée qui peut être utilisé dans la feuille de style avec la fonction key ()	6.0	1.0
message	Écrit un message à la sortie (utilisé pour signaler les erreurs)	6.0	1.0
namespace-alias	Remplace un espace de noms dans la feuille de style par un autre espace de noms dans le résultat	6.0	
number	Détermine la position du noeud courant (retourne un nombre)	6.0	1.0
otherwise	Indique une action par défaut pour l'élément <choose>	5.0	1.0
output	Définit le format du document de résultat	6.0	1.0

XSLT - référence

Élément	Description	IE	FF
param	Déclare un paramètre	6.0	1.0
preserve-space	Définit les éléments pour lesquels les espaces doivent être préservées	6.0	1.0
processing-instruction	Écrit une instruction de traitement	5.0	1.0
sort	Trie	6.0	1.0
strip-space	Définit les éléments pour lesquels les espaces blancs doivent être supprimés	6.0	1.0
stylesheet	Définit l'élément racine de la feuille de style	5.0	1.0
template	Règles à appliquer quand un noeud spécifié est trouvé	5.0	1.0
text	Ecrit un texte littéral	5.0	1.0
transform	Définit l'élément racine de la feuille de style	6.0	1.0
value-of	Extrait la valeur d'un nœud sélectionné	5.0	1.0
variable	Déclare une variable	6.0	1.0
when	Spécifie une action pour l'élément <choose>	5.0	1.0
with-param	Définit la valeur d'un paramètre à passer à un modèle	6.0	1.0

Quelques fonctions XSLT

Nom	Description
current()	Retourne le noeud courant
document()	Utilisé pour accéder aux nœuds dans un document XML externe
element-available()	Teste si un élément est supporté par le processeur XSLT
format-number()	Convertit un nombre en chaîne de caractères
function-available()	Teste si une fonction est supportée par le processeur XSLT
generate-id()	Retourne une chaîne de caractères qui identifie de manière unique un noeud spécifié
key()	Retourne un ensemble de nœuds on utilisant l'index spécifié par un élément <xsl:key>
system-property()	Retourne les propriétés du système
unparsed-entity-uri()	Retourne l'URI d'une entité non analysée

Conclusion

- XSLT est un langage pour transformer des documents XML en XHTML ou en d'autres documents XML.
- Le standard le plus important de XSL
- Utilise XPath pour naviguer dans des documents XML
- Permet d'afficher, masquer, réorganiser, trier, effectuer des tests sur les éléments et des attributs du fichier résultant.
- XSLT est un vrai langage de programmation
- XSLT n'a pas son équivalent pour la transformation d'arbre
- Mais, la mise au point de programmes XSLT peut s'avérer « délicate »

Merci