eXtensible Stylesheet Language

XSL Transformation

Alexis NEDELEC

LISYC EA 3883 UBO-ENIB-ENSIETA
Centre Européen de Réalité Virtuelle
Ecole Nationale d'Ingénieurs de Brest

XML/XSLT

Objectif de XML

- Description, contrôle de données:
  - XML, DTD, XML Schemas...

Objectif de XSLT

- Transformation de document XML

Transformation XSL

- appliquer des règles de transformation
- aux nœuds d'un document XML
- pour publication des données de document XML
- sur différents supports (WEB, WAP, PDF...)

Transformation de document

Processeur XSL

- document XML source (`file.xml`)
- document XSL décrivant les transformations (`file.xsl`)
- produire un document résultat
- sous forme d'arbre sérialisable
- exploitable sur différents supports

Evaluation XSL

- arbre résultat évalué étape par étape
- parcours des nœuds du document source (nœud courant)
- application des règles dans le contexte du nœud courant

Exemple de document XML

message.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="message.xsl"?>
<messages>
 <message id="001">
  <from>Nedelec</from>
  <to>EdT</to>
  <subject>A propos de XSLT</subject>
  <content>Quel jour ?</content>
 </message>
 <message id="002">
  ...
 </message>
</messages>
```

Exemple de document XSL

message.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="/messages">
 <html><body><p>
   <h2>
    Nombre de messages:
    <xsl:value-of select="count(message)"/>
   </h2>
   <xsl:apply-templates/>
 </p></body></html>
</xsl:template>
```

Structure de document XSL

Auto-descriptif : c'est du XML

Le document XSL est lui-même un document XML

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

Début de définition de la feuille de style

- `<xsl:stylesheet...>` ou `<xsl:transform...>`

Premier document XSL

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

   <!-- TO DO  -->

</xsl:stylesheet>
```

Contenu de document XSL

Ensemble de règles XSL

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

Contenu de règles XSL

```
<xsl:template match="expression XPath">
  <!--debut du modele de transformation>
    <!-- TO DO  -->
  <!--fin du modele de transformation>
</xsl:template>
```

Transformation XSL de document XML

Règles de transformations XSLT

- définitions de règles de transformation sur des nœuds
  - `<xsl:template match="...">`
- à appliquer, déclencher, instancier ...
  - `<xsl:apply-templates select="...">`)

Application de règles

- sur un ensemble de noeuds
  - expression XPath (`select="..."`)
- sous conditions (**pattern**) de sélection
  - expression XPath (`match="..."`)

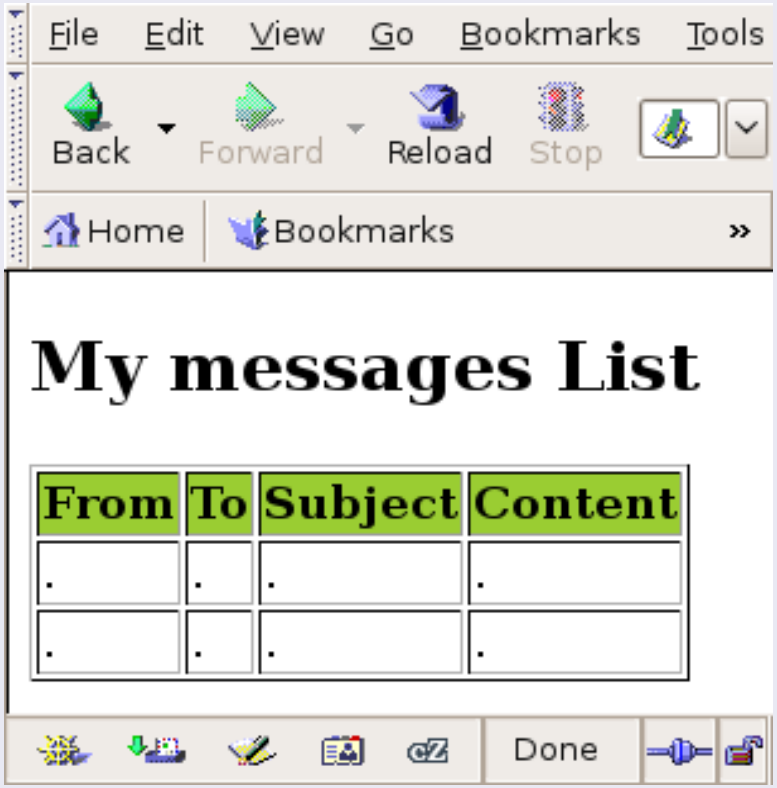
Transformation au format HTML

Sur le nœud racine du document : <messages>

```
<xsl:template match="messages">
<html><body>
 <h2>My messages List</h2>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>From</th><th>To</th>
   <th>Subject</th><th>Content</th>
  </tr>
  <tr>
   <td>.</td> <td>.</td><td>.</td><td>.</td>
  </tr>
 </table>
</body></html>
</xsl:template>
```

Transformation au format HTML

Application de la règle de transformation



Transformation au format HTML

Récupération de données XML : <xsl:value-of>

```
<xsl:template match="messages">
...
 <tr>
  <td><xsl:value-of select="message/from"/></td>
  <td><xsl:value-of select="message/to"/></td>
  <td><xsl:value-of select="message/subject"/></td>
  <td><xsl:value-of select="message/content"/></td>
 </tr>
...
</xsl:template>
```

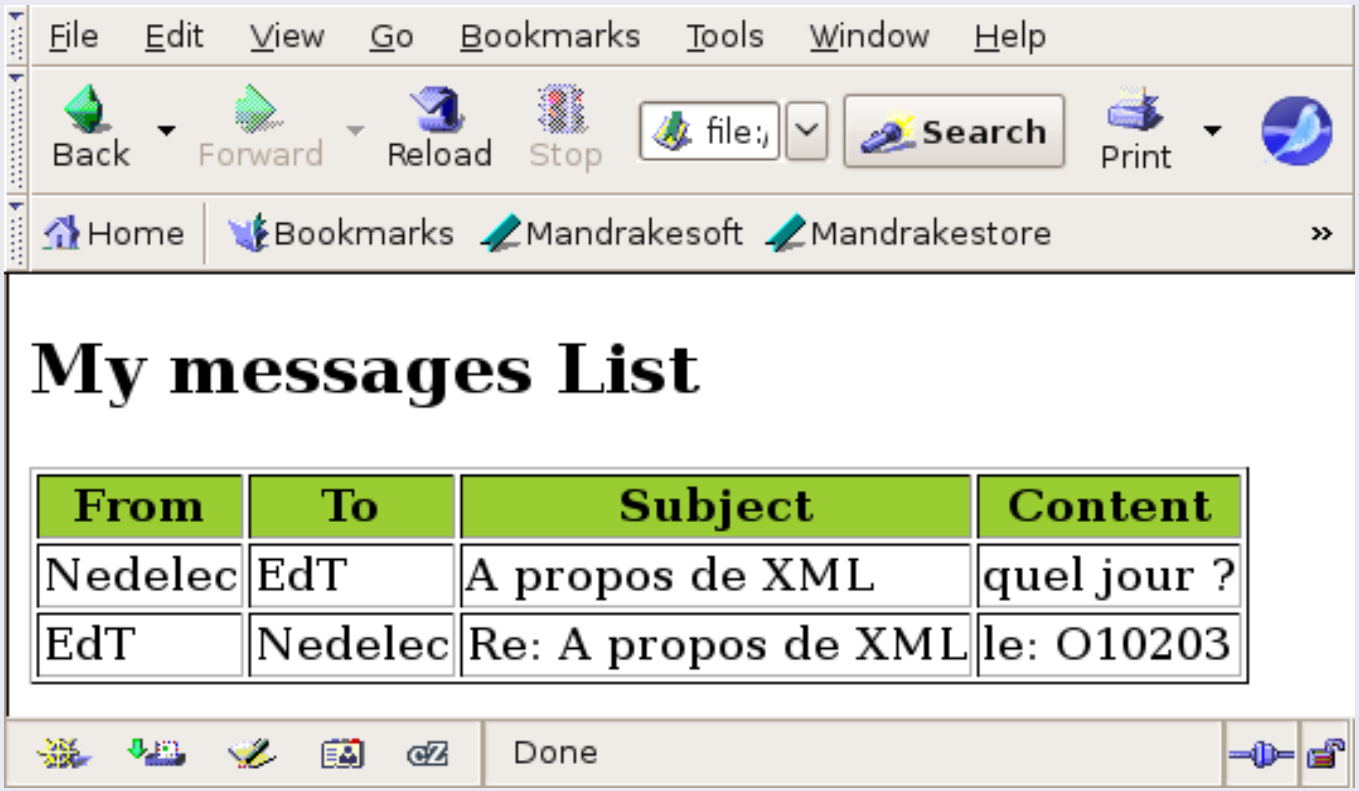
Transformation au format HTML

Parcours de données XML : <xsl:for-each>

```
<xsl:template match="messages">
 ...
 <xsl:for-each select="message">
  <tr>
  <td><xsl:value-of select="message/from"/></td>
  <td><xsl:value-of select="message/to"/></td>
  <td><xsl:value-of select="message/subject"/></td>
  <td><xsl:value-of select="message/content"/></td>
  </tr>
 </xsl:for-each>
 ...
</xsl:template>
```

Transformation au format HTML

Du document XML à sa transformation HTML



Transformation HTML

Règles de transformation HTML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:template match="messages">
 <html><body>
  <h2>
  Nombre de messages:
  <xsl:value-of select="count(message)"/>
  </h2>
  <xsl:apply-templates select="message"/>
</body></html>
</xsl:template>
```

Transformation WML

Règles de transformation WML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:template match="messages">
  <wml>
   <card>
   Nombre de messages:
   <xsl:value-of select="count(message)"/>
   </card>
   <xsl:apply-templates select="message"/>
  </wml>
</xsl:template>
```

Transformation FO

Règles de transformation FO

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 ...
 <xsl:template match="message">
  <fo:block text align="center">
   <fo:block font-size="40pt" font-size="bold">
     <xsl:value-of select="heading"/>,
   </fo:block>
  <fo:block>
   From: <xsl:value-of select="from"/>,
   To: <xsl:value-of select="to"/>,
  </fo:block>
  ...
</xsl:template>
```

Transformations par défaut

Règles de base

```
<!-- sur tous les enfants du noeud courant -->
<xsl:template match="*|/">
     <xsl:apply-templates />
</xsl:template>
<!-- sur un noeud de type texte ou attribut -->
<xsl:template match="text()|@*">
<!-- <xsl:template match="text()|attribute::*"> -->
     <xsl:value-of select="." />
</xsl:template>
<!-- sur un noeud de type commentaire ou traitement  -->
<xsl:template match="comment()|processing-instruction()">
 </xsl:template>
```

Récupération de données

Expression XPath

- pour récupérer un ensemble de nœuds
- par écriture de chemins de localisation
- dans une représentation arborescente XML
- dans laquelle on se déplace (nœud courant)

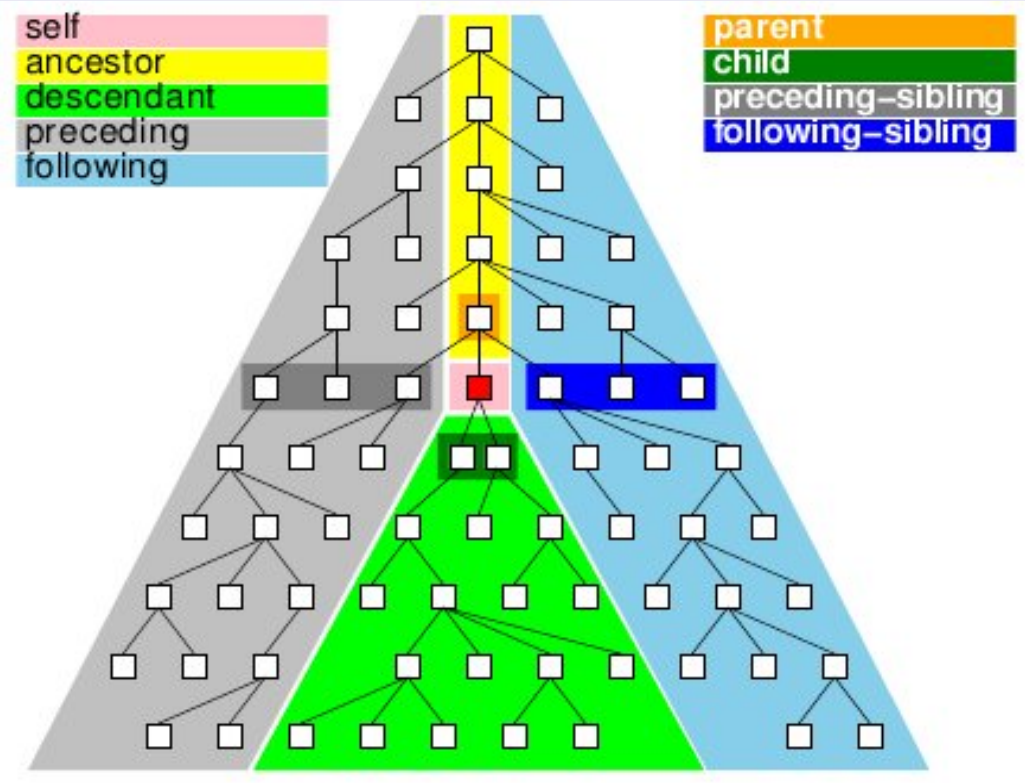
Chemin de localisation

- axes de localisation (parents, frères, cousins...)
- déterminants, filtres (noeuds de test, `TestNode`)
- prédicats (restrictions sur les déterminants)

Langage XPath

Chemin de localisation

- `descendant::node()[position()=1]`



Langage XPath

Axes de localisation

- une première approximation du résultat
- au voisinage du nœud contexte
- 13 axes de localisation en tout
  - 11 pour parcourir l'arbre
  - 2 pour les nœuds de type `attribute, namespace`

Nœuds particuliers

1. `attribute`: les attributs du nœud contexte
2. `namespace`: les espaces de nommage du nœud contexte

Langage XPath

Axes de localisation

1. `child`: les enfants directs
2. `descendant`: toute la descendance
3. `parent`: existe pour tout type de nœud (sauf racine)
4. `ancestor`: tous les ascendants (y compris la racine)
5. `self`: seulement le nœud contexte
6. `following-sibling`: au même niveau après le nœud contexte
7. `preceding-sibling`: au même niveau avant le nœud contexte
8. `following`: nœuds qui précèdent le nœud contexte
9. `preceding`: nœuds qui suivent le nœud contexte
10. `descendant-or-self`: descendants nœud contexte inclus
11. `ancestor-or-self`: ascendants nœud contexte inclus

Langage XPath

Types de nœuds

1. `root`: la racine du document (`/`)
2. `element`: élément de document XML (`<elt>...bla,bla,bla...</elt>`)
3. `text`: texte d'un élément XML (`...bla,bla,bla...`)
4. `attribute`: nœud attribut d'élément XML (`attr="value"`)
5. `namespace`: espace de nommage d'un document XML (`xmlns="http://.../un_schema.xsd"`)
6. `processing-instruction`: directive de traitement sur un document XML (`<?cible arg1 arg2?>`)
7. `comment`: nœud de type commentaire (`<!-- ... -->`)

Evaluation XPath

Collection de nœuds (`node-set`)

- on part d'un ensemble initial
- défini par axe de localisation
- filtré par un déterminant
- auxquels on applique des prédicats

Exemple : `descendant::message[attribute::id='001']`

- axe de localisation : `descendant`
- déterminant : `message`
- prédicat : `[attribute::id='001']`

Evaluation XPath

Déterminants

- représente une fonction booléenne
- qui, appliquée à un nœud de l'ensemble
- détermine s'il doit rester dans l'ensemble
- un déterminant peut être
  - un nom : `descendant::message`
  - n'importe quoi : `descendant::*`
  - un type : `descendant::text()`

Evaluation XPath

4 types de déterminants

1. `text()`
2. `comment()`
3. `processing-instruction()`
4. `node()`

Exemples : avec la racine (/) pour nœud contexte

- `processing-instruction()`:
  - 1 nœud (`xml-stylesheet`)
- `descendant::node()[text()="Nedelec"]`:
  - 2 nœuds (`to`, `from`)
- `child::messages/child::node()[@id="001"]`:
  - 1 nœud (`message`)

Langage XSL

Eléments top-level

Type d'élément	Description
xsl:attribute-set	définir un groupe d'attributs
xsl:decimal-format	définir un format d'affichage numérique
xsl:import	importer un programme XSLT
xsl:include	inclure un programme XSLT
xsl:key	clé pour un groupe de nœuds
xsl:namespace-alias	alias pour certains espaces de noms
xsl:output	format de sortie (HTML,XML...)
xsl:param	définir un paramètre
xsl:preserve-space	conserver les nœuds blancs
xsl:strip-space	supprimer les nœuds blancs
xsl:template	définir une règle XSLT
xsl:variable	définir une variable XSLT

Langage XSL

Eléments du langage

Type d'élément	Description
`xsl:apply-imports`	appliquer une règle importée
`xsl:apply-templates`	déclencher une règle
`xsl:attribute`	insérer un attribut
`xsl:call-template`	appeler une règle par son nom
`xsl:choose`	équivalent d'un `switch`
`xsl:comment`	insérer un commentaire dans le résultat
`xsl:copy`	copier un nœud source dans le résultat
`xsl:copy-of`	même chose avec la descendance
`xsl:element`	insérer un nœud dans le résultat
`xsl:fallback`	déclencher une règle en cas d'instruction non-reconnue
`xsl:for-each`	boucle d'itérations

Langage XSL

Eléments du langage

Type d'élément	Description
`xsl:if`	branchement conditionnel
`xsl:message`	produire un message
`xsl:number`	numérotation des nœuds du document résultat
`xsl:otherwise`	action par défaut (`xsl:choose`)
`xsl:processing-instruction`	instruction XML
`xsl:result-document`	documents résultats
`xsl:text`	insérer un nœud `Text`
`xsl:value-of`	évaluer une expression XPath et l'insérer dans le résultat
`xsl:when`	action (`xsl:choose`)
`xsl:with-param`	paramètres de règle

Structures de contrôle

<xsl:if>

```
<xsl:if test="...&gt;...">
  ... some code ...
</xsl:if>
```

<xsl:for-each>

```
<xsl:for-each select="message">
 <xsl:sort select="expediteur"/>
  ... some code ...
</xsl:for-each>
```

Opérateurs de comparaison

- = (égalité) , != (différence)
- > (supérieur), < (inférieur)

Structures de contrôle

<xsl:choose>, <xsl:when>,</xsl:otherwise>

```
<xsl:choose>
 <xsl:when test="...&gt....">
  ... some code ...
 </xsl:when>
 <xsl:when test="...&lt....">
  ... some code ...
 </xsl:when>
 <xsl:otherwise>
  ... some code ....
 </xsl:otherwise>
</xsl:choose>
```

<xsl:import>, <xsl:include>

message.xsl: inclusion de règle (msg.xsl)

```
<xsl:import href="msg.xsl"/>
<!--     <xsl:include href="msg.xsl"/>  -->

<xsl:template match="messages">
<html><body><p>
    <h2> Nombre de messages (message.xsl):
      <xsl:value-of select="count(message)"/>
    </h2>
    <xsl:apply-templates select="message"/>
</p></body></html>
</xsl:template>
```

<xsl:import>, <xsl:include>

msg.xsl: règle incluse (message.xsl)

```
<xsl:template match="messages">
<html><body><p>
    <h2> Nombre de messages (msg.xsl):
      <xsl:value-of select="count(message)"/>
    </h2>
    <xsl:apply-templates select="message"/>
</p></body></html>
</xsl:template>
```

Attribut de règle : priority

Elimination de nœuds

```
<xsl:output method="xml" encoding="ISO-8859-1" />
<xsl:template match="/">
 <xsl:apply-templates />
</xsl:template>

<xsl:template match="from" />

<xsl:template match="@*|node()" priority="-1">
 <xsl:copy>
  <xsl:apply-template select="@*|node()" />
 </xsl:copy>
</xsl:template>
```

Variables : <xsl:variable>

Variables globales (top-level element), locales aux règles

```
<xsl:variable name="name" select="expression">
 <!-- Content:template -->
</xsl:variable>
```

Exemples : avec et sans attribut select

```
<xsl:variable name="color" select="red" />
<xsl:variable name="color">
 red
<xsl:variable/>
```

Variables : <xsl:variable>

Variables globales (top-level element), locales aux règles

```
<xsl:variable name="header">
 <tr bgcolor="#9acd32"><th>From</th><th>To</th>
 <th>Subject</th><th>Content</th></tr>
</xsl:variable>
<xsl:template match="messages">
<html><body>
 <h2>My messages List</h2>
 <table border="1">
  <xsl:copy-of select="$header"/>
   <xsl:for-each select="messages">
   ...
   </xsl:for-each>
  </table></body></html>
</xsl:template>
```

Paramètres de règles : <xsl:param>

Règle de transformation avec paramètres

```
<xsl:template name="templateName">
 <xsl:param name="paramName" />
 ...
</xsl:template>
```

Apppel de règle de transformation avec paramètres

```
<xsl:call-template name="templateName">
 <xsl:with-param name="paramName" select="aValue" />
...
</xsl:call-template>
</xsl:template>
```

Paramètres de règles : <xsl:param>

Appel de règle : calcul de factorielle

```
<xsl:template match="/">
 <html>
  <body bgcolor="#FFFFFF">
   <xsl:call-template name="factorielle">
    <xsl:with-param name="n" select="6" />
   </xsl:call-template>
  </body>
 </html>
</xsl:template>
```

Paramètres de règles : <xsl:param>

Définition de la règle : calcul de factorielle

```
<xsl:template name="factorielle">
 <xsl:param name="n" />
 <xsl:choose>
  <xsl:when test="$n&lt;=1">1</xsl:when>
  <xsl:otherwise>
   <xsl:variable name="fact">
    <xsl:call-template name="factorielle">
     <xsl:with-param name="n" select="$n-1" />
    </xsl:call-template>
   </xsl:variable>
   <xsl:value-of select="$fact * $n" />
  </xsl:otherwise>
 </xsl:choose>
</xsl:template>
```

Attributs : <xsl:attribute>

Insertion d'attributs dans un élément

```
<xsl:template match="messages">
 <xsl:for-each  select="message">
   <message>
    <xsl:attribute name="Exp">
     <xsl:value-of select="from">
    </xsl:attribute>
    <xsl:attribute name="Dest">
     <xsl:value-of select="to">
    </xsl:attribute>
   </message>
 </xsl:for-each>
</xsl:template>
```

Attributs : <xsl:attribute-set>

Définition de groupe d'attributs

```
<xsl:attribute-set name="MonStyle">
 <xsl:attribute name="bgcolor">
   white
 </xsl:attribute>
 <xsl:attribute name="font-name">
  Helvetica
 </xsl:attribute>
 <xsl:attribute name="font-size">
  20pt
 </xsl:attribute>
</xsl:attribute-set>
```

Attributs : <xsl:use-attribute-sets>

Utilisation de groupe d'attributs

```
<xsl:template match="/">
 <html>
  <head>
   <title>
    Ceci est Mon Style
   </title>
  </head>
  <body xsl:use-attribute-sets="MonStyle">
   Exemple de style
  </body>
 </html>
</xsl:template>
```

Clés : <xsl:key>

Utilisation de Clés

```
<xsl:key name="idx" match="message" use="@id"/>
<xsl:template match="/">
 <html><body>
  <xsl:for-each  select="key('idx','001')">
   <p>
    Id: <xsl:value-of select="@id"/><br />
    From : <xsl:value-of select="from"/><br />
    To : <xsl:value-of select="to"/><br />
    Heading : <xsl:value-of select="heading"/><br />
    Body : <xsl:value-of select="body"/><br />
   </p>
  </xsl:for-each>
 </body></html>
</xsl:template>
```

Tests : <xsl:choose>

Tests Conditionnels Multiples

```
<xsl:for-each select="message">
<tr>
 <xsl:choose>
  <xsl:when test="from='Nedelec'">
   <td bgcolor="#ff00ff">
    <xsl:value-of select="from"/>
   </td>
  </xsl:when>
  <xsl:otherwise>
   <td><xsl:value-of select="from"/></td>
  </xsl:otherwise>
 </xsl:choose>
 <!-- ctnd : xsl:for-each -->
```

Tests : <xsl:choose>

Tests Conditionnels Multiples

```
  <td><xsl:value-of select="to"/></td>
  <td><xsl:value-of select="heading"/></td>
  <td><xsl:value-of select="body"/></td> </tr>
</xsl:for-each>
```

Présentation dans une table HTML

```
<xsl:template match="messages">
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>From</th><th>To</th>
   <th>Heading</th><th>Body</th></tr>
  <!-- code du xsl:for-each -->
 </table>
</xsl:template>
```

Fonctions XSL et XPath

Fonctions XSL

Nom	Description
`current()`	nœud courant du source
`document()`	accès à un document externe
`element-avaliable()`	supporté par le processeur XSLT
`format-number()`	conversion d'un nombre en chaîne
`function-avaliable()`	supporté par le processeur XSL
`generate-id()`	chaîne associée au nœud
`key()`	nœud associé à un index `<xsl:key>`
`system-property()`	propriétés du système
`unparsed-entity-uri()`	l'entité associé à une URI

Fonctions XSL et XPath

Manipulation de nœuds

Nom	Description
`count()`	nombre de nœuds d'un sous-arbre
`id()`	récupérer un sous-arbre par son identifiant
`last()`	dernier nœud de la liste en traitement
`local-name()`	nom de l'élément non-préfixé
`name()`	nom de l'élément
`namespace-uri()`	nom de l'espace de nommage de l'URI
`position()`	position du nœud de la liste en traitement

Fonctions XSL et XPath

Manipulation de chaînes de caractères

Nom	Description
`concat()`	concaténation des arguments
`contains()`	vrai si 2ème argument dans le 1er
`normalize-space()`	homogénéisation des espaces
`starts-with()`	vrai si 1er arg. commence par le 2ème
`string()`	conversion de l'argument en chaîne
`string-length()`	nombre de caractères dans une chaîne
`substring()`	retourne une partie de chaîne
`substring-after()`	retourne le reste d'une chaîne
`substring-before ()`	retourne le début d'une chaîne
`translate()`	traduire les occurences d'une chaîne par une autre

Fonctions XSL et XPath

Manipulation de nombres

Nom	Description
`ceiling()`	retourne le plus petit entier non-inférieur
`floor()`	retourne le plus grand entier non-supérieur
`number()`	retourne l'argument en nombre
`round()`	retourne l'entier le plus proche
`sum()`	somme des valeurs d'un ensemble de nœuds

Fonctions booléennes

Nom	Description
`boolean()`	retourne la valeur booléenne après conversion
`false()`	retourne la valeur fausse
`lang()`	vrai si associé à l'élément `xsl:lang`
`not()`	retourne la négation de l'argument
`true()`	retourne la valeur vrai

Fonctions XSL et XPath

Exemple d'utilisation

```
<xsl:template name="mes_messages">
 <h2> Messages pour Nedelec: </h2>
 <xsl:for-each select="message">
  <xsl:value-of select="
    concat(position(),'/',last(),' : ',
           'message reçu de ',from) "/>
  <p>
   <xsl:value-of select="concat('Sujet: ', subject)"/>
  </p>
  <p>
   <xsl:value-of select="concat('Contenu: ', content)"/>
  </p>
 </xsl:for-each>
</xsl:template>
```

Fonctions XSL et XPath

Exemple d'utilisation

```
<xsl:template match="messages">
<html><body><p>
 <h1>
  Nombre Total de messages:
  <xsl:value-of select="count(message)"/></h1>
 <xsl:call-template name="mes_messages"/>
</p></body></html>
</xsl:template>
```

Fonctions XSL et XPath

Exemple d'utilisation

```
<xsl:choose>
  <xsl:when test="function-available('sum')">
    <p>sum() is supported.</p> </xsl:when>
  <xsl:otherwise>
   <p>sum() is not supported.</p>
  </xsl:otherwise>
</xsl:choose>
<xsl:choose>
  <xsl:when test="element-available('xsl:comment')">
    <p>xsl:comment is supported.</p></xsl:when>
  <xsl:otherwise>
    <p>xsl:comment is not supported.</p>
  </xsl:otherwise>
</xsl:choose>
```

Python et XML

Stephan Richter : "lxml takes all the pain out of XML"

- basé sur `libxml2 et libxslt`
- mieux que l'API `ElementTree`

Création de document

lxml.etree

```
from lxml import etree

root = etree.Element("messages")
msg1=etree.SubElement(root,"message", id="001")
fNed=etree.SubElement(msg1,"from")
fNed.text="Nedelec"
toEdT=etree.SubElement(msg1,"to")
toEdT.text="EdT"
subj1=etree.SubElement(msg1,"subject")
subj1.text="A propos de XML"
cont1=etree.SubElement(msg1,"content")
cont1.text="quel jour ?"
```

Sauvegarde de document

lxml.etree

```
msg2=etree.SubElement(root,"message", id="002")
fEdT=etree.SubElement(msg2,"from")
fEdT.text="EdT"
toNed=etree.SubElement(msg2,"to")
toNed.text="Nedelec"
subj2=etree.SubElement(msg2,"subject")
subj2.text="Re: A propos de XML"
cont2=etree.SubElement(msg2,"content")
cont2.text="le ... "
file = open("messages.xml",'w')
file.write(etree.tostring(root, pretty_print=True))
file.close()
```

Transforamtion XSL

lxml.etree

```
from lxml import etree
import StringIO

xslt_tree = etree.parse("messages.xsl")
#print etree.tostring(xslt_tree)
transform= etree.XSLT(xslt_tree)
tree = etree.parse("services.xml")
result_tree = transform(tree)
#print result_tree

file = open("services.html",'w')
file.write(etree.tostring(tree))
file.close()
```

Recherche d'éléments

Requètes XPath

```
from lxml import etree

tree = etree.parse("messages.xml")
msgNodes = tree.xpath('/messages/message')

for node in msgNodes :
    print node.tag, node.get("id")

msgAttributes = tree.xpath('//message/@id')
for attribute in msgAttributes :
    print attribute
```

Bibliographie

Ouvrages

- Bernd Amann et Philippe Rigaux
  “Comprendre XSLT”ed. O’Reilly 2002
- Philippe Drix
  “XSLT Fondamental” ed. Eyrolles 2002

Bibliographie

Liens Au Net

- l'organisme (`www.xml.org`)
- le consortium W3C (`www.w3.org`)
- les tutoriaux (`www.w3schools.com`)
- `http://cortes.cnam.fr:8080/XSLT` (Philippe Rigaux)