eXtensible Stylesheet Language

XSL Transformation

Alexis NEDELEC

Centre Européen de Réalité Virtuelle
Ecole Nationale d'Ingénieurs de Brest

XML/XSL

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...)

XSL: Transformation de document XML

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

XSL: c'est du XML

Auto-descriptif : Du XML pour transformer du XML

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

   <!-- TO DO : templates -->

</xsl:stylesheet>
```

XSL: Ensemble de règles

Règle de transformation XSL

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

Contenu de règles XSL

```
<xsl:template match="...">
  ...
  <!-- TO DO  : generate output -->
  ...
</xsl:template>
```

XSL: Ensemble de règles

Règles de transformations XSL

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

Condition d'application de règles XSL

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

Document XML : message.xml

Traitement XSL associé : message.xsl

```
<?xml version="1.0" encoding="utf-8s"?>
<!-- <?xml version="1.0" encoding="ISO-8859-1"?> -->
<?xml-stylesheet type="text/xsl" href="message.xsl"?>
<messages>
 <message id="001">
  <expediteur>Nedelec</expediteur>
  <destinataire>EdT</destinataire>
  <sujet>A propos de XSLT</sujet>
  <contenu>Quel jour ?</contenu>
 </message>
 <message id="002">
  ...
 </message>
</messages>
```

Document XSL : message.xsl

Règle sur le nœud racine du document ("/")

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
 <xsl:template match="/">
  <html>
   <body>
    <xsl:apply-templates/>
   </body>
  </html>
 </xsl:template>
 ...
</xsl:stylesheet>
```

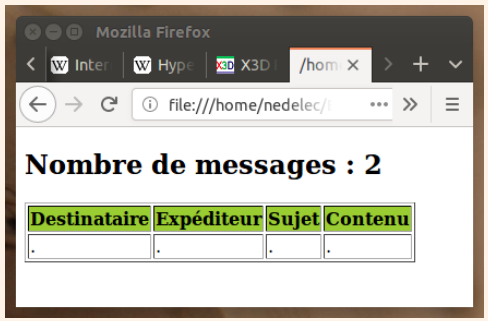
Document XSL : message.xsl

Règle sur l'élément racine du document ("messages")

```
<xsl:template match="messages">
 <h2>Nombre de messages :
  <xsl:value-of select="count(message)"/>
 </h2>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>Destinataire</th><th>Expediteur</th>
   <th>Sujet</th><th>Contenu</th>
  </tr>
  <tr>
   <td>.</td> <td>.</td><td>.</td><td>.</td>
  </tr>
 </table>
</xsl:template>
```

Transformation XSL : du XML au HTML

Chargement sur un navigateur : `messages.xml`



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

```
...
 <tr>
  <td><xsl:value-of select="message/expediteur"/></td>
  <td><xsl:value-of select="message/destinataire"/></td>
  <td><xsl:value-of select="message/sujet"/></td>
  <td><xsl:value-of select="message/contenu"/></td>
 </tr>
...
```

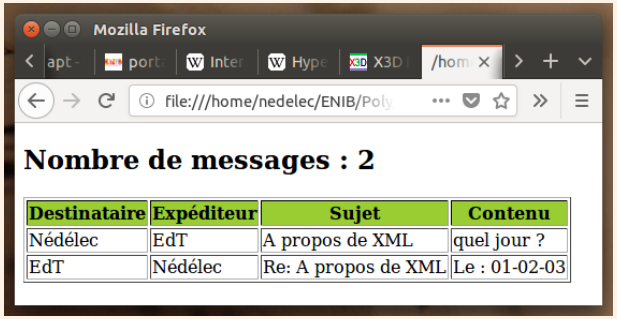
Transformation XSL : du XML au HTML

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

```
<xsl:template match="messages">
 ...
 <xsl:for-each select="message">
  <tr>
   <td><xsl:value-of select="expediteur"/></td>
   <td><xsl:value-of select="destinataire"/></td>
   <td><xsl:value-of select="sujet"/></td>
   <td><xsl:value-of select="contenu"/></td>
  </tr>
 </xsl:for-each>
 ...
</xsl:template>
```

Transformation XSL : du XML au HTML

Chargement sur un navigateur : `messages.xml`



Du `<xsl:for-each ..>` à la règle `<xsl:template ..>`

```
<xsl:template match="message">
   <tr>
    <td><xsl:value-of select="expediteur"/></td>
    <td><xsl:value-of select="destinataire"/></td>
    <td><xsl:value-of select="sujet"/></td>
    <td><xsl:value-of select="contenu"/></td>
   </tr>
</xsl:template>
```

Transformation XSL : du XML au HTML

Règles de transformation XSL

```
<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 XSL : du XML au WML

Règles de transformation XSL

```
<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 XSL : du XML au XSL-FO

Règles de transformation XSL

```
<xsl:template match="messages">
 <fo:block text align="center">
  Nombre de messages:
  <xsl:value-of select="count(message)"/>
   <fo:block font-size="40pt" font-size="bold>
    <xsl:apply-templates select="message"/>
   </fo:block>
  </fo:block>
</xsl:template>
```

Transformation XSL : Recherche d'information

XPath : Langage de requête

- recherche d'ensemble de nœuds (`node-set`) dans un document XML
- désignation sous forme de chemins dans un arbre
- utilisable par d'autres spécifications (`XSLT, XQuery ...`)
- syntaxe particulière :

```
/descendant-or-self::node()/attribute::*[starts-with(.,'0')]
//@*[starts-with(.,'0')]
```

XPath : Recherche d'information

Règles XSLT par défaut : requêtes XPath

```
<!-- 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>
```

XPath : Parcours d'arbre

Parcours d'arbre pour récupérer un ensemble de nœuds

- 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)
- prédicats (restriction sur les nœuds filtrés)

```
/descendant-or-self::node()/attribute::*[starts-with(.,'0')]
```

XPath : Parcours d'arbre

Evaluation d'expression

- Chemin : désignation de nœuds
- Evaluation : en fonction d'un contexte général
- Résultat : valeur alphanumérique, sous-ensemble de nœuds

Sept types de nœuds XML

- `Document`,nœud racine : `/`
- `Element` ,élément XML : `/Bibliotheque`
- `Attribute`, attribut XML : `/Bibliotheque/Livre/@number`
- `Text`, valeurs textuelles :
  `<xsl:value-of select="Titre"/>`
- `Comment, ProcessingInstruction, NameSpace`

XPath : Parcours d'arbre

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`

Axes de localisation

- `/A/B/C/text()` : texte des nœuds `C` sur le chemin `A-B-C`
- `/child::A/child::B/child::C/text()` :
  - `child::node()` : axe par défaut
- `//C/text()` : les textes des nœuds `C` du document
  - `/descendant-or-self::node()` : descendants à partir de la racine du document

XPath : Parcours d'arbre

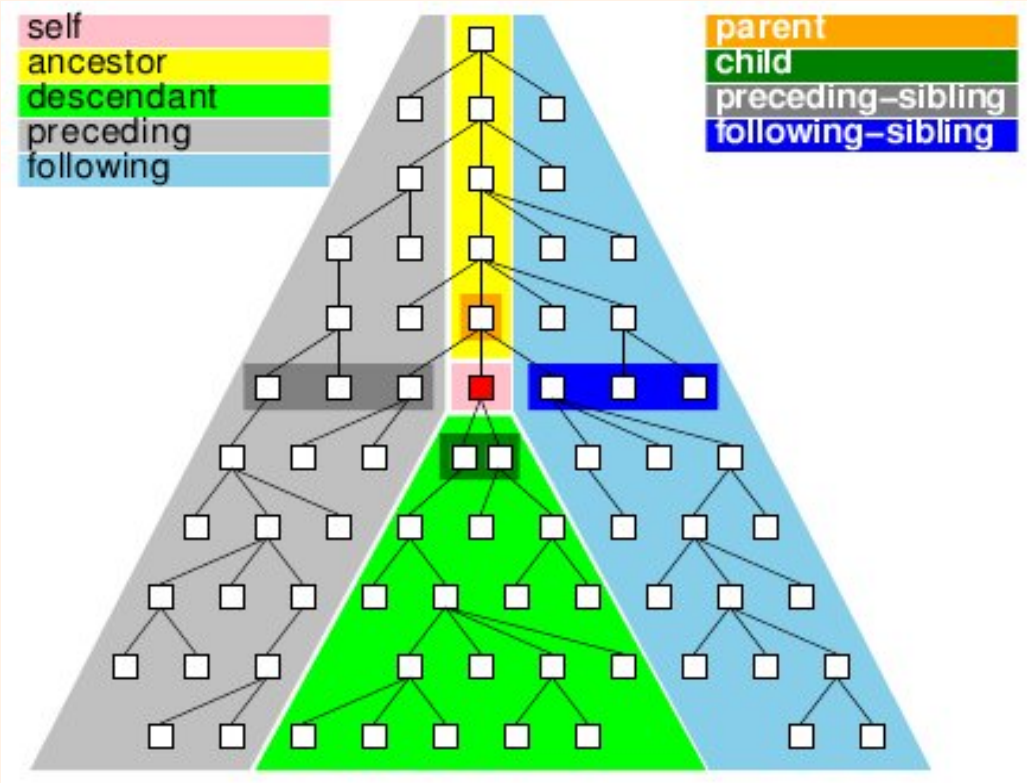
Axes de localisation

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

XPath : Parcours d'arbre

Axes de localisation

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



(Thanks to : Vincent Quint)

XPath : Expression

Exemple d'arbre XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<A>
 <B attr="1">
  <C>bla</C>
  <C>bla bla</C>
 </B>
 <B attr="2">
  <C attr="OK">
    bla bla bla
    <D attr1="1" attr2="OK"/>
  </C>
 </B>
 <C attr="OK">bla bla bla bla</C>
</A>
```

XPath : Expression

Chemins absolus, relatifs

- `/A/B/@attr` : recherche à partir du nœud racine
- `A/B/@attr` : recherche à partir d'un nœud contexte

Recherche en absolu

- A partir du nœud racine (`/`)
- Si l'élément racine a pour nom de balise `A`
- On parcourt les élements fils de nom de balise `B`
- Pour récupérer leurs attributs de nom `attr`

Recherche en relatif vs absolu

- la même chose mais à partir d'un nœud contexte (`A`)

XPath : Expression

Syntaxe XPath

Séquence d'étapes : `[/]step1/step2/.../stepN`
Composition d'une étape :

- Un axe de recherche à partir d'un nœud contexte
  - fils, parent, descendants, frères ...
- Un filtre pour retenir certains nœuds sur l'axe
  - type de nœuds (texte, commentaires, instructions ...)
  - noms d'éléments ou d'attributs
- Un ou des prédicats pour décrire les propriétés que devront satisfaire les nœuds à retenir dans le résultat.

Forme générale d'une étape

```
axe::filtre[predicat1][predicat2]...
```

XPath : Evaluation

Résultat d'évaluation d'une étape

- une valeur (`string, number, boolean`)
- un ensemble de (références sur des) nœuds (`node-set`)

Evaluation d'une suite d'étapes

- à partir d'un nœud contexte
- on évalue la première étape
- on obtient un `node-set`
- on parcourt un à un chaque nœud du `node-set`
- qui devient le nœud contexte d'évaluation de l'étape suivante

XPath : Evaluation

Evaluation d'une suite d'étapes

A chaque étape, on prend successivement comme nœud contexte d'évaluation chaque nœud faisant partie du `node-set` de l'étape précédente.

Exercice

Répresenter, sous forme d'arbres, les étapes successives d'évaluation de l'expression :

```
/A/B/@attr
```

XPath : Evaluation

Axes de recherches : écritures de chemins

- `/A/B/C` ou explicitement :
  `/child::A/child::B/child::C`
- `//C` ou explicitement :
  `/descendant-or-self::C`
- `//D/@*` ou explicitement :
  `/descendant-or-self::C/attribute::*`
- `.//C` ou explicitement :
  `self::node()/descendant-or-self::C`
- `./..//C` ou explicitement :
  `self::node()/parent::node()/descendant-or-self::C`

XPath : Evaluation

Filtres : sélection de nœuds sur un axe

- sélection par type de nœuds
- sélection par noms, seulement pour les nœuds :
  - Element, Attribute et ProcessingInstruction

Filtrage par nom

- d'élément : `/A/B/C`
- d'attribut : `//@attr1` (nom d'attribut en dernière étape)
- de PI : `/processing-instruction('python')`

XPath : Evaluation

Filtrage de tous les attributs, éléments, instructions

- `/descendant::node()/@*`
- `/descendant::node()/*`
- `/processing-instruction()`

Filtrage par type de nœuds

- commentaires : `//comment()`
- textes : `/A//C/text()`

XPath : Evaluation

Prédicats de sélections de nœuds

- expression booléenne
- constitué d'un ou plusieurs tests
- utilisant les connecteurs logiques (`and, or`)
- et la négation (`not`)

Exemples

- Avec ou sans attributs : `/A/B[@attr], /A/B[not(@attr)]`
- Certains nœuds :
  - `/A/B[position()=1],/A/B[last()]`
  - `/A/B[@attr="1"], /A/B[/A/D/@attr="OK"]`

XPath : Evaluation

Composition de prédicats

- `/A/B[@attr="1" and position()=last()]`
- `/A/B[@attr="1"][position()=last()]`
- `/A/B[position()=last()][@attr="1"]`

XSL : Langage de programmation

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

XSL : Langage de programmation

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

XSL : Langage de programmation

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

Alternatives : <xsl:if>

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

Boucles : <xsl:for-each>

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

Expression booléennes

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

Structures de contrôle

switch : <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>
```

Priorité de règles

Importation de fichiers de règles XSL

```
<!-- message.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>
```

Priorité de règles

Importation de fichiers de règles XSL

```
<!-- msg.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>
```

XSL : Priorité de règles : intra-document

Attribut de règle : priority

```
<xsl:output method="xml" encoding=utf-8" />
<xsl:template match="/">
 <xsl:apply-templates />
</xsl:template>

<xsl:template match="expediteur" />

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

XSL : Variables

globales (top-level element) ou locales : <xsl:variable>

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

Exemples : avec et sans attribut select

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

XSL : Variables

Variables globales : top-level element

```
<xsl:variable name="header">
 <tr bgcolor="#9acd32">
  <th>Expediteur</th>
  <th>Destinataire</th>
  <th>Sujet</th>
  <th>Contenu</th>
 </tr>
</xsl:variable>
```

XSL : Variables

Variables globales : top-level element

```
<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>
```

XSL : Paramètres de règles

Règle avec paramètres : <xsl:param>

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

Appel de règle avec paramètres: <xsl:with-param>

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

XSL : Paramètres de règles

Exemple : 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>
```

XSL : Paramètres de règles

Définition de 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>
```

XSL : 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="expediteur">
    </xsl:attribute>
    <xsl:attribute name="Dest">
     <xsl:value-of select="destinataire">
    </xsl:attribute>
   </message>
 </xsl:for-each>
</xsl:template>
```

XSL : 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>
```

XSL : 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>
```

XSL : 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')">
    Id: <xsl:value-of select="@id"/><br />
    Exp : <xsl:value-of select="expediteur"/><br />
    Dest : <xsl:value-of select="destinataire"/><br />
    Sujet : <xsl:value-of select="sujet"/><br />
    Contenu : <xsl:value-of select="contenu"/><br />
  </xsl:for-each>
 </body></html>
</xsl:template>
```

XSL : Tests : <xsl:choose>

Tests Conditionnels Multiples

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

XSL : Tests : <xsl:choose>

Tests Conditionnels Multiples

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

Présentation dans une table HTML

```
<xsl:template match="messages">
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>Expediteur</th><th>Destinataire</th>
   <th>Sujet</th><th>Contenu</th></tr>
  <!-- code du xsl:for-each -->
 </table>
```

XSL : Fonctions XSL et XPath

Fonctions XSL

Nom	Description
`current()`	nœud courant du source
`document()`	accès à un document externe
`element-available()`	supporté par le processeur XSLT
`format-number()`	conversion d'un nombre en chaîne
`function-available()`	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

XSL : 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

XSL : 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

XSL : 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

XSL : 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 recu de ',expediteur)"/>
  <p>
   <xsl:value-of select="concat('Sujet: ',sujet)"/>
  </p>
  <p>
   <xsl:value-of select="concat('Contenu: ',contenu)"/>
  </p>
 </xsl:for-each>
</xsl:template>
```

XSL : 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>
```

XSL : 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`
- `http://lxml.de`

Chargement de données : etree.parse()

Requêtes XPath

```
from lxml import etree

tree = etree.parse("messages.xml")
for msg in tree.xpath("/messages/message"):
    print(msg.tag,msg.get("id"))

msg_attr = tree.xpath("//message/@id")
for attr in msg_attr :
    print attr

for exp in tree.xpath("/messages/message/expediteur"):
    print(exp.text)
```

Création de document : etree.Element()

Créations de nœuds: etree.SubElement()

```
from lxml import etree
root=etree.Element("messages")
msg=etree.SubElement(root,"message", id="001")
exp=etree.SubElement(msg,"expediteur")
exp.text="Nedelec"
dest=etree.SubElement(msg,"destinataire")
dest.text="EdT"
sujet=etree.SubElement(msg,"sujet")
sujet.text="A propos de XML"
contenu=etree.SubElement(msg,"content")
contenu.text="quel jour ?"
```

Création de document : etree.Element()

Créations de nœuds: etree.SubElement()

```
msg=etree.SubElement(root,"message", id="002")
exp=etree.SubElement(msg,"expediteur")
exp.text="EdT"
dest=etree.SubElement(msg,"destinataire")
dest.text="Nedelec"
sujet=etree.SubElement(msg,"sujet")
sujet.text="Re: A propos de XML"
contenu=etree.SubElement(msg,"contenu")
contenu.text="Le : 2001-02-03"
file = open("messages.xml","w")
file.write(etree.tostring(root, pretty_print=True))
file.close()
```

Transformation de document

etree.parse(), etree.XSLT()

```
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("messages.xml")
result_tree = transform(tree)
#print result_tree

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

Transformation de document

etree.parse(), etree.XSLT()

```
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("messages.xml")
result_tree = transform(tree)
#print result_tree

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

Fonctions `lxml`

Fonctions principales

Fonction	Description
`addnext(element)`	Ajouter un frère juste après `element`
`addprevious(element)`	Ajouter un frère juste avant `element`
`getnext(element)`	Retourner le prochain `element` frère
`getprevious(element)`	Retourner l'`element` frère précédent
`getparent(element)`	Retourner le parent de l'`element`
`getchildren(element)`	Retourner les enfants de l'`element`
`remove((elt1,elt2)`	Remplacer un nœud `elt1` par `elt2`
`replace(element)`	Supprimer l'`element` en paramètre
`clear()`	Supprimer tous les sous-éléments
`index(child)`	Trouver la position d'un sous-élément
`insert(index, elt)`	Insérer un sous-élément `elt` à la position `index`

Fonctions `lxml`

Fonctions principales

Fonction	Description
`find(xpath)`	Rechercher le premier sous-élément de l'ensemble retourné par l'expression (`xpath`)
`findtext(xpath)`	Rechercher le texte du premier sous-élément de l'ensemble retourné par l'expression
`findall(xpath)`	Rechercher tous les éléments de l'ensemble retourné par l'expression (`xpath`)
`items()`	Retourner les attributs d' élément
`keys()`	Retourner les noms des attributs d'élément
`values()`	Retourner les valeurs des attributs
`set(key,value)`	Créer un attribut avec sa valeur
`append(element)`	Ajouter un sous-élément
`extends(elements)`	Ajouter les sous-éléments

Bibliographie

Ouvrages

- **D. Hunter** “Initiation à XML”
  ed. Eyrolles Wrox Press 2000
- **S. Lecomte** “XML par la pratique”
  ed. ENI 2005
- **A. Brillant** “XML Cours et exercices”
  ed. Eyrolles Wrox Press 2007
- **G. Chagnon, F. Nolot** “XML”
  ed. Pearson Education, coll. Synthex 2007
- **Bernd Amann et Philippe Rigaux**
  “Comprendre XSLT”ed. O’Reilly 2002
- **Philippe Drix**
  “XSLT Fondamental” ed. Eyrolles 2002

Bibliographie

Liens Au Net

- le consortium W3C : `www.w3.org`
- l'organisme : `www.xml.org`
- les tutoriaux : `www.w3schools.com`
- Python et XML : `http://lxml.de`
- cours en ligne de Gilles Chagnon : `www.licence.elec.upmc.fr/S_tec/coursEnLigne/xml`
- `http://cortes.cnam.fr:8080/XSLT` (Philippe Rigaux)
- Le club d'entraide des développeurs : `www.developpez.com`