Tirème SARL

XSLT, XSL Transformations

Rédaction : **Pierre Attar**

Recommandation(s) liée(s) : DOM - SAX - XSL

Cette spécification définit la syntaxe et la sémantique d'un langage de transformation de documents XML en autres documents, qu'ils soient XML ou purement textuels.

XSLT fait partie de la recommandation XSL, un langage généralisé d'expression de feuilles de styles. En ce sens, XSLT *n'est pas un langage de programmation générique* d'accès aux documents XML comme DOM ou SAX peuvent l'être. C'est un *langage spécialisé pour toutes les questions de transformations* nécessaires à la présentation d'un document, sur papier, et surtout sur le Web...

Cependant, la spécification permet d'implémenter XSLT de façon indépendante de tout processus de formatage, basé sur les *formatting objects* de XSL ou sur HTML. Du coup, il existe une zone de recouvrement entre XSLT et les API DOM ou SAX.

Objectifs

La norme DSSSL de l'*ISO* définit les concepts nécessaires à la présentation d'un document : il est nécessaire de transformer l'arbre, avant de pouvoir le formater. Les transformations nécessaires sont, par exemple, le fait de prendre l'auteur d'une lettre, défini en attribut de lettre, pour générer une signature, en fin de lettre.

De façon plus générique, les transformations sont de l'ordre de :

- la réorganisation d'informations ;
- la duplication d'informations ;
- la génération d'informations, fixe ou calculée ;
- la suppression d'informations.

L'objectif d'XSLT, dans le cadre du Web, est d'être la recommandation de transformation liée à des processus de formatage. Cependant, puisqu'il transforme un document XML en un autre document XML (un arbre d'objets de formatage), XSLT peut être utilisé pour beaucoup d'autres besoins de transformation et, cela, même si, dans la volonté de ses concepteurs, il est spécialisé dans la présentation.

Principes

L'objectif de XSLT est de construire, à partir d'un document XML, un ou plusieurs documents textuels XML.

Plus précisément, les documents issus d'une transformation sont des arbres HTML ou des arbres d'objets de formatage, selon la recommandation XSL. Par extension du langage, ce peut être n'importe quel arbre ... voir même des documents purement textuels.

A noter que depuis la version 2.0, il est possible de lire en entrée des documents textuels non XML ... ceci est fort pratique pour structurer des documents non XML en utilisant le langage d'expressions régulières intégré à XSLT.

Le langage est basé sur des sélections d'objets typés (`xsl:apply-templates`) auxquels on applique une transformation (`xsl:template`) ; la transformation par défaut copie la source vers la cible.

Plus précisément, c'est à partir de la racine (`/`) du document d'entrée que sont effectuées des sélections. Par exemple, si le document d'entrée est un ensemble de lettres commerciales, la transformation suivante fournira une liste de tous les titres et auteurs des lettres contenues :

```
<xsl:stylesheet ...>
    <xsl:template match="/">
        <!-- faire un traitement sur les objets typés lettre -->
        <xsl:apply-templates select="lettre"/>
    </xsl:template>
    <!-- définition du traitement des lettres -->
    <xsl:template match="lettre">
        <!-- générer un paragraphe -->
        <fo:block>
            <!-- prendre le contenu caractères de l'objet titre
            contenu dans la lettre -->
            <xsl:value-of select="titre"/>
            <!-- ajouter du texte de liaisons -->
            <xsl:text> ; auteur : <xsl:text>
            <!-- prendre le contenu caractères de l'objet auteur
            contenu dans la lettre -->
            <xsl:value-of select="auteur"/>
        </fo:block>
    </xsl:template>
</xsl:stylesheet>
```

La sélection d'objets typés est réalisée en utilisant le langage XPath, initialement conçu pour ce seul besoin.

Le vocabulaire d'XSLT permet de :

- créer de nouveaux éléments dans la cible (`xsl:element`, `xsl:attribute`, `xsl:attribute-set`) ;
- générer du texte (`xsl:text`) ou des sous-arbres issus de la source (`xsl:copy`) ;
- effectuer des tests (`xsl:if`) et, plus généralement, des traitements conditionnés par des choix (`xsl:choose`) ;
- déclarer des variables (`xsl:variable` et `xsl:param`). À noter que les notions de variables globales n'existent pas, du fait de la volonté de définir un langage "*context-free*" où l'on n'a pas besoin de connaître l'état de toute la spécification pour appliquer un changement local ;
- se déclarer des listes d'objets identifiés (`xsl:key`) qui seront ensuite utilisables dans les transformations (`key`) ;
- réaliser des boucles (`xsl:for-each`), triées si nécessaires (`xsl:sort`). XSLT 2.0 introduit (`xsl:for-each-group`) qui permet de répéter une action sur des sous-ensembles possédant une caractéristique commune (par exemple la première lettre du nom des personnes).
- modulariser les feuilles de styles en différents fichiers (`xsl:import` et `xsl:include`) ou en modules de traitement factorisés (`xsl:call-template`, `xsl:with-param`) ;
- contrôler le type d'écriture de la sortie (`xsl:output`, `xsl:strip-space`, `xsl:preserve-space`) et le fichier (ou les fichiers) à écrire (`xsl:result`);

...

À ce vocabulaire sont ajoutés, outres les fonctions de XPath, un certain nombre de nouvelles fonctions permettant, d'une part, de mieux sélectionner et, d'autre part, de mieux contrôler le contenu de l'arbre cible.

Depuis la version 2.0, les NameSpace sont bien gérés dans la recommandation.

Dans la pratique

XSLT est un langage *a priori* difficile d'accès : son écriture sous forme XML le rend "verbeux" et les méthodes d'écriture par sélection d'objets typés nécessitent de revoir beaucoup de méthodes de programmation.

En revanche, une fois cet apprentissage réalisé, ce langage est extrêmement puissant... si puissant que beaucoup l'utilisent même uniquement comme langage de transformation pur, en dehors de toute problématique de présentation.

Par exemple, écrire un programme de validation du fait que toute les références bibliographiques d'un livre existent n'est pas, avec un langage de programmation usuel, quelque chose de trivial. Avec XSLT, l'opération est aisée à écrire :

```
<xsl:stylesheet ...>
    <!-- les items de bibliographie (bib-item) ont un attribut nom
    d'identification -->
    <xsl:key name="itemDeBiblio" match="bib-item" use="./@nom"/>
    <xsl:template match="/">
            <!-- BIBREF réalise la référence, au travers d'un attribut
            lien -->
            <xsl:for-each select="//BIBREF">
                <xsl:choose>
                    <!-- le test permet de savoir si le résultat de
                    recherche dans la clé existe et correspond à la
                    valeur de l'attribut lien -->
                    <xsl:when test="key('itemDeBiblio',@lien)/
                    @nom=@lien">
                        <xsl:text>Tout va bien</xsl:text>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:text>Tout va mal</xsl:text>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

Par ailleurs, le langage XSLT n'est pas fermé et un mécanisme permet à chaque outil de se rajouter ses propres extensions.

Quoi qu'il en soit et du fait des extensions, il est nécessaire de choisir l'environnement applicatif dans lequel on se situe, avant d'utiliser des extensions. En effet, si, par exemple, on utilise des extensions dans une feuille de styles chargée dynamiquement par un logiciel de consultation du Web, on se contraint très rapidement à rendre dépendantes les applications du type de logiciel de consultation. Il est donc nécessaire de différencier les transformations génériques, utilisables par n'importe quel outil de traitement XSLT de celles que l'on décide, liées à un outil particulier, et ses extensions.

Recommandations(s)

- *Transformations XSL (XSLT)*
  Recommandation, version 1.0, du 16-11-1999
  Document sur http://xmlfr.org/w3c/TR/xslt/

XSL Transformations
Recommandation, version 2.0, du 23-01-2007
Document sur http://www.w3.org/TR/xslt20/

XSLT Requirements
Projet en cours, version 2.0, du 14-02-2001
Document sur http://www.w3.org/TR/xslt20req

XSLT 2.0 and XQuery 1.0 Serialization
Recommandation, version 20072301, du 23-01-2007
Document sur http://www.w3.org/TR/xslt-xquery-serialization/