

Transformer un document XML avec XSLT

Stéphane Bortzmeyer

<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 24 Novembre 2007. Dernière mise à jour le 26
Février 2010

<http://www.bortzmeyer.org/transformation-xslt.html>

Un des avantages de XML est qu'on a à sa disposition de nombreux outils tout faits pour éditer les documents, sans avoir besoin de savoir écrire des *"one-liners"* en Perl. Par exemple, pour adapter les liens de mon blog vers Wikipédia, je peux me contenter d'un simple script XSLT.

Le problème était le suivant : les articles de ce blog sont écrits en XML <<http://www.bortzmeyer.org/blog-implementation.html>>. Ils comprennent souvent des liens vers Wikipédia qui ont la forme suivante :

```
<wikipedia>DNS</wikipedia >
```

mais le sigle DNS est trop fréquent et ne pointe pas forcément vers le protocole réseau qui m'intéresse mais vers une page d'homonymie <<http://www.bortzmeyer.org/identifiant-convoye-wikipedia.html>>. Je dois donc modifier tous les liens vers le DNS (et il y en a beaucoup) pour lever l'ambiguïté :

```
<wikipedia name="Domain Name System">DNS</wikipedia >
```

C'est évidemment trop pénible à la main, il faut programmer. Un utilisateur Unix expérimenté penserait toute de suite à `sed` mais cet outil ne connaît pas la syntaxe XML et risquerait de changer trop de texte, par manque de prise en compte du contexte. Un script dans un langage de programmation classique serait déjà meilleur mais XSLT est souvent plus simple.

Voici un script XSLT qui copie tous les éléments XML sauf ceux nommés `<wikipedia>`. Pour ces derniers, il regarde si leur contenu (fonction `text()`) est l'ancien nom et, si oui, il produit un nouvel élément ayant l'attribut `name` avec la nouvelle valeur :

```

<xsl:output method="xml" encoding="ISO-8859-1"
  cdata-section-elements="code" /> <!-- On peut mettre plusieurs noms d'éléments
    ici, séparés par des espaces. -->

<xsl:param name="old">No value for old</xsl:param>
<xsl:param name="new">No value for new</xsl:param>

<!-- Utiliser la contrainte "[text() = $old]" serait plus propre (et
éviterait le <xsl:choose>) mais les contraintes n'acceptent pas les
variables :-( -->
<xsl:template match="wikipedia">
  <xsl:choose>
    <xsl:when test="text()=$old">
      <wikipedia><xsl:attribute name="name"><xsl:value-of select="$new"/></xsl:attribute><xsl:value-of se
    </xsl:when>
    <xsl:otherwise>
      <xsl:copy>
        <xsl:apply-templates select="node()|@*" />
      </xsl:copy>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

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

```

La dernière règle s'occupe de copier tous les autres éléments.

Le script peut se lancer ainsi (avec l'interpréteur XSLT xsltproc) :

```
% xsltproc --stringparam old DNS --stringparam new "Domain Name System" update-wp.xslt document.xml
```

mais j'utilise plutôt un petit script shell qui s'occupe des détails, comme de copier le nouveau fichier :

```

#!/bin/sh

set -e

oldtext=$1
newtext=$2

blog=$HOME/Blog

if [ -z "$3" ]; then
  echo "Usage: $0 old-text new-text xml-files" > /dev/stderr
  exit 1
fi
shift
shift

temp=`mktemp`
for file in $*; do
  xsltproc --stringparam old "$oldtext" --stringparam new "$newtext" \
  ${blog}/schemas/update-wp.xsl \
  $file > $temp
  cp $temp $file
  echo "$file updated, remember to darcs record"
done
rm -f $temp

```

Il reste deux problèmes dans ce script XSLT :

- Ce script change la base des entités numériques (comme œ). J'écris en base 16 et xsltproc me produit de la base 10.
- les entités dans la déclaration ne sont pas gardées. Je m'en sers pour garantir l'homogénéité de certains mots, par exemple : `<!ENTITY any "<foreign>anycast</foreign>"`. Et ainsi, je suis sûr que `&any;` sera toujours l'élément `<foreign>`. Mais xsltproc supprime la définition de l'entité et fait l'expansion partout, ce qui m'interdit de changer la définition par la suite.

Ce sont des problèmes fondamentaux de XSLT, car ce dernier ne voit que l'"infoset", pas le fichier texte original (en tout cas de la version 1 de XSLT, il est possible que la 2 aide sur ce problème). Pour la même raison, on ne peut pas écrire un "pretty printer" XML en XSLT.

Enfin, j'ai donc, pour ce blog, adopté une autre solution <http://www.bortzmeyer.org/transformation-texte.html> qui repose sur un bricolage avec les expressions rationnelles.

Merci à Stéphane Bonhomme pour son aide lors de la mise au point du script XSLT et à Julian Reschke pour ses explications.