Remplacer du texte en XSLT

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 mars 2008

https://www.bortzmeyer.org/remplacement-xslt.html

Si on produit un format quelconque à partir de XML, par exemple du CSV <https://www.bortzmeyer.org/xml-to-csv.html> ou bien du LaTeX <https://www.bortzmeyer.org/pdf-version.html>, on est souvent confronté au cas de caractères qui, sans créer de problèmes dans le source XML, sont « spéciaux » pour le format de destination et nécessitent donc d'être **protégés** ("*to be escaped*", dans la langue de J. K. Rowling). Comment le faire en XSLT ?

La technique présentée ici est largement inspirée d'un excellent article de Jenni Tennison <http://www.dpawson.co.uk/xsl/sect2/StringReplace.html>. Les exemples ci-dessous concernent la production de LaTeX pour lequel on peut trouver une liste des caractères « spéciaux » <http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/teTeX/latex/latex2e-html/ltx-164.html>. On définit d'abord un espace de noms XML pour les données sur ces caractères spéciaux :

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:tolatex="http://www.bortzmeyer.org/xmlns/tolatex"
...
```

puis on définit ces caractères et leur remplacement. On note que cette définition est complètement séparée du code qui fait le remplacement, pour faciliter la maintenance :

```
  <tolatex:replacements>
    <tolatex:replacement search="$">DOLLAR-TO-ESCAPE</tolatex:replacement>
    <tolatex:replacement search="\">$\backslash$</tolatex:replacement>
    <!-- Difficult circularity in escaping -->
    <tolatex:replacement search="DOLLAR-TO-ESCAPE">\$</tolatex:replacement>
    <tolatex:replacement search="#">\#</tolatex:replacement>
    <tolatex:replacement search="%">\%</tolatex:replacement>
    <tolatex:replacement search="_">\_</tolatex:replacement>
...
    <tolatex:replacement search=" ">~</tolatex:replacement>
    <tolatex:replacement search="&amp;">\&amp;</tolatex:replacement>
    <tolatex:replacement search="&#x153;">\oe{}</tolatex:replacement>
    <tolatex:replacement search="&#x20ac;">\EUR{}</tolatex:replacement>
  </tolatex:replacements>

  <xsl:variable name="replacements"
    select="document('')/xsl:stylesheet/tolatex:replacements" />
```

Quand on rencontre du texte, le gabarit suivant est appelé. Il appelle alors `replace` en passant un paramètre qui est la position dans l'élément `<tolatex:replacements>` (au début, un) :

```
<xsl:template name="escape_text" match="text()">
  <xsl:variable name="position" select="1"/>
  <xsl:call-template name="replace">
      <xsl:with-param name="position" select="$position"/>
      <xsl:with-param name="content" select="string(.)"/>
  </xsl:call-template>
</xsl:template>
```

En quoi consiste le gabarit `replace` ? Il est récursif, utilisant pour cela le paramètre `position`. Il parcourra ainsi la totalité des `<tolatex:replacement>`.

```
<xsl:template name="replace">
  <xsl:param name="position" select="1"/>
  <xsl:param name="content"/>
  <xsl:variable name="transformed">
    <xsl:call-template name="string_replace">
      <xsl:with-param name="string" select="$content" />
      <xsl:with-param name="search"
        select="$replacements/tolatex:replacement[$position]/@search" />
      <xsl:with-param name="replace"
        select="$replacements/tolatex:replacement[$position]" />
    </xsl:call-template>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test="$position &lt;
                          count($replacements/tolatex:replacement)">
      <!-- There are still replacements to perform. Call myself. -->
      <xsl:call-template name="replace">
        <xsl:with-param name="position" select="$position + 1"/>
        <xsl:with-param name="content" select="$transformed"/>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
       <!-- No more replacements to do -->
       <xsl:value-of select="$transformed"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Pour une position donnée dans l'élément `<tolatex:replacements>`, le gabarit `replace` appelera `string_replace` qui va parcourir une chaîne de caractères en effectuant le remplacement indiqué :

```
<xsl:template name="string_replace">
  <xsl:param name="string"/>
  <xsl:param name="search"/>
  <xsl:param name="replace"/>
  <xsl:choose>
    <xsl:when test="contains($string, $search)">
      <xsl:variable name="rest">
        <xsl:call-template name="string_replace">
          <xsl:with-param name="string" select="substring-after
```

```
                                              ($string, $search)"/>
          <xsl:with-param name="search" select="$search"/>
          <xsl:with-param name="replace" select="$replace"/>
        </xsl:call-template>
      </xsl:variable>
      <xsl:value-of
            select="concat(substring-before($string, $search), $replace, $rest)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$string"/>
    </xsl:otherwise>
  </xsl:choose>
```

Ce gabarit `string_replace` est lui aussi récursif, pour pouvoir effectuer plusieurs remplacements dans la chaîne de caractères, au cas où un caractère spécial apparaitrait plusieurs fois.

Une version complète se trouve dans les sources de ce blog <`https://www.bortzmeyer.org/blog-implementation.html`>, voir notamment le fichier `schema/common-latex.xsl`.