DSSSL, le langage de transformation de SGML

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 novembre 2008

https://www.bortzmeyer.org/dsssl.html

Le langage DSSSL est aujourd'hui bien oublié. Il était le langage favori de transformation de SGML. C'est donc un lointain précurseur de XSL et, jusqu'à très récemment, il faisait des choses que XSL ne pouvait pas faire comme de traduire du Docbook en un beau texte imprimé. Comme j'ai utilisé ce langage pendant des années, il est temps de documenter un peu.

Le langage de balisage SGML, ancêtre de XML avait beaucoup de points communs avec son descendant à succès. La syntaxe concrète est la même (SGML permettait en théorie d'utiliser d'autres syntaxes mais cela a été peu implémenté). Parmi les différences importantes, il y a le fait que SGML n'était pas un format ouvert (la norme n'a jamais été librement distribuée) et le fait qu'un schéma soit obligatoire. Tout document SGML devait être conforme à une DTD, seul langage de schéma disponible.

Comme XML avec XSL, SGML avait son langage de transformation favori, DSSSL (également sans norme distribuée librement). Bien sûr, on pouvait utiliser d'autres langages (c'est ce que faisait le projet LinuxDoc <http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Howtos-with-LinuxDoc.html> à ses débuts où les HOWTO <http://tldp.org/docs.html#howto> étaient en SGML et où Perl était utilisé pour la transformation). Mais DSSSL était le plus répandu.

Une partie du (relatif) succès de DSSSL venait de la disponibilité d'une mise en œuvre en logiciel libre de qualité, Jade <http://www.jclark.com/jade/>, écrit par James Clark, l'auteur de groff et de nxml <http://www.thaiopensource.com/nxml-mode/>. Comme la plupart des logiciels de James Clark, Jade a été écrit en une seule fois et jamais maintenu par la suite (un projet de reprise du code, OpenJade <http://openjade.sourceforge.net/>, a existé, mais n'a pas fait grand'chose).

DSSSL lui-même n'avait pas de norme disponible, il n'a guère existé de documents de qualité sur ce langage, Jade était peu documenté <http://www.jclark.com/jade/>, autant dire que les programmeurs DSSSL n'avaient pas la vie facile ! (Il existe heureusement une excellente liste de diffusion <http://www.mulberrytech.com/dsssl/dssslist/>.) Pourtant, de gros programmes ont été produits en DSSSL, le plus fameux étant les feuilles de style Docbook <http://wiki.docbook.org/topic/DocBookDssslStylesheets> de Norman Walsh.

À quoi ressemble du DSSSL ? Contrairement à XSL, sa syntaxe n'était pas du XML. DSSSL est un langage de Turing qui ressemble étroitement à Scheme. Il dispose donc de toutes les caractéristiques de Scheme, on peut définir des fonctions, et les appliquer sans limites. La connexion avec le fichier SGML se fait par la fonction DSSSL `element` qui prend en paramètre le nom d'un élément SGML. Lorsque jade tourne, il applique la fonction `element` à tous les éléments trouvés dans le fichier (donc la programmation en DSSSL est essentiellement déclarative).

Pour illustrer cela, prenons un fichier SGML et transformons le. Par habitude, je vais le faire en XML, et le document sera donc à la fois du XML et du SGML. Cela impose de créer d'abord une DTD, obligatoire en SGML. Voici donc un schéma DTD qui décrit (de manière très sommaire !) les systèmes de fichiers d'une machine Unix (c'est à peu près l'information qu'on obtient avec `df`) :

```
<!ELEMENT filesystems (filesystem+)>
<!ELEMENT filesystem (mount-point, device, size, used)>
<!ELEMENT mount-point (#PCDATA)>
<!ELEMENT device (#PCDATA)>
<!ELEMENT size (#PCDATA)>
<!ELEMENT used (#PCDATA)>
```

et voici un fichier XML/SGML conforme à cette DTD (on peut vérifier avec `xmllint --noout --valid test.xml`) :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE filesystems SYSTEM "filesystems.dtd">
<filesystems>

<filesystem>
<mount-point>/home</mount-point><device>/dev/sda3</device><size>50000000000</size><used>2341556227</used>
</filesystem>

<filesystem>
<mount-point>/</mount-point>
<device>/dev/sda1</device>
<size/>
<used/>
</filesystem>

</filesystems>
```

Écrivons maintenant un programme DSSSL pour imprimer ce fichier :

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN">

(root (make simple-page-sequence))

(element filesystems (make paragraph))

(element filesystem (make paragraph))

(element mount-point (make sequence
       font-weight: 'bold))
```

Ce programme va créer un paragraphe par système de fichiers (l'appel `(element filesystem ...)`), le point de montage étant en gras.

Si on lance jade en donnant (option `-d`) ce fichier `test.dsl` en paramètre :

```
% jade  -d test.dsl /usr/share/sgml/declaration/xml.dcl test.xml
```

Jade produira par défaut un fichier FOT, un format intermédiaire de Jade. Comme on utilise rarement ce format, produisons plutôt du TeX :

```
% jade -t tex -d test.dsl /usr/share/sgml/declaration/xml.dcl test.xml
```

Notez, dans les deux cas, le paramètre `/usr/share/sgml/declaration/xml.dcl` qui est un fichier contenant les instructions expliquant à Jade comment se débrouiller avec XML (sur une Debian, il est dans le paquetage `sgml-data`).

Et que fait-on du fichier TeX produit ? Ce n'est **pas** du LaTeX, il contient des macros spécifiques à Jade et il faut donc le compiler avec jadetex `<http://jadetex.sourceforge.net/>` :

```
% jadetex test.tex
```

On a désormais un fichier DVI qu'on peut traiter avec les outils TeX habituels.

Cela, c'était pour l'impression. Et pour produire du texte seul ? Ou bien du HTML pour publication sur le Web ? Dans le premier cas, Jade n'a malheureusement jamais offert de solution. Pour le second, Jade disposait de fonction de transformation SGML -¿ SGML (puisque HTML est du SGML) qui avaient été utilisées par les "*Docbook Stylesheets*" `<http://wiki.docbook.org/topic/DocBookDssslStylesheets>` mais vite remplacées par du XSL (alors que, pour l'impression, les outils de la chaîne XSL ont été longtemps médiocres par rapport à ceux de la famille DSSSL).

Je l'ai dit, un des grands succès de DSSSL avait été le développement des "*Docbook Stylesheets*" `<http://wiki.docbook.org/topic/DocBookDssslStylesheets>`, un ensemble très riche de fonctions de production de TeX ou de HTML à partir de fichiers Docbook. La lecture de leur code donne plein d'idées et d'information sur DSSSL. On peut bien sûr les utiliser telles quelles, sans les modifier mais on peut aussi utiliser le caractère dynamique de DSSSL pour enrichir et/ou modifier leur comportement. Commençons par un petit article Docbook simple :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN"
     "http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd">
<article>
<title>Simple essai avec Docbook</title>
<para>La seconde sous-plate-forme est composée d'une
machine physique principale et d'une machine de secours.</para>
<para>Vue la
nécessité d'une certaine variété des machines, vue les progrès de la
virtualisation, vue la nécessité de lutter contre le <ulink url="http://www.jres.org/planning/paper.php?pid=39">
global</ulink>, la machine
physique principale portera plusieurs machines virtuelles.</para>
</article>
```

Jade peut le transformer :

```
% jade -t tex \
    -d /usr/share/sgml/docbook/stylesheet/dsssl/modular/print/docbook.dsl    \
    /usr/share/sgml/declaration/xml.dcl test.db
```

(Le fichier `docbook.xsl`, sur une Debian, est dans le paquetage `docbook-dsssl`.) (Oui, la ligne de commande est longue. Une occasion pour les alias <https://www.bortzmeyer.org/shell-alias.html>.)

Jade ayant produit le TeX, on peut lancer jadetex :

```
% jadetext test.tex
```

et regarder le beau DVI produit.

Méthode similaire pour HTML, d'ailleurs :

```
% jade -t sgml \
       -d /usr/share/sgml/docbook/stylesheet/dsssl/modular/html/docbook.dsl \
       /usr/share/sgml/declaration/xml.dcl test.db
```

En faisant cela, on a utilisé du DSSSL sans le voir. Et si on veut modifier la sortie de Jade ? Par exemple, on veut utiliser la police Palatino et augmenter la marge gauche. Les *"Docbook stylesheets"* étant en logiciel libre, on peut toujours modifier le source et même le distribuer à ses amis. Mais cela rendrait difficile la synchronisation avec les développements ultérieurs de ces programmes. Il vaut mieux faire une simple couche d'adaptation <http://docbook.sourceforge.net/release/dsssl/current/doc/custom.html>, par exemple avec le programme DSSSL suivant :

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" [
<!ENTITY docbook.dsl PUBLIC "-//Norman Walsh//DOCUMENT DocBook Print Stylesheet//EN" CDATA dsssl>
]>
<style-sheet>

<style-specification id="print" use="docbook">
<style-specification-body>

(define %body-font-family%
  ;; The font family used in body text
  "Palatino")

(define %left-margin%
  ;; Width of left margin
  3.5cm)

</style-specification-body>
</style-specification>

<external-specification id="docbook" document="docbook.dsl">

</style-sheet>
```

On a dû « emballer » le code DSSSL dans des instructions (en XML) qui permettent de dire à Jade où il doit trouver le programme (`&docbook.dsl;`) après avoir chargé nos instructions. L'utilisation est la même qu'avant, seul le nom du programme DSSSL a changé :

```
%  jade -t tex  -d print.dsl /usr/share/sgml/declaration/xml.dcl test.db
```

et on a désormais la grande marge demandée et la bonne police. En effet, les "*Docbook Stylesheets*" `<http://wiki.docbook.org/topic/DocBookDssslStylesheets>` définissent un certain nombre de variables `<http://docbook.sourceforge.net/release/dsssl/current/doc/print/index.html>` comme `%left-margin%` et on peut les redéfinir avec la fonction `define` de Scheme.

Comme DSSSL est un langage de programmation complet, on peut bien sûr. définir des fonctions plus riches. Par exemple, si on a localement étendu Docbook `<https://www.bortzmeyer.org/docbook-custom-schema.html>` pour y mettre un élément `<rfc>` avec un attribut optionnel `num` qui est le numéro du RFC, on peut inclure un code comme :

```
(element rfc
  (let (
(num (attribute-string "num"))
)
    (if num
  (literal (string-append
    "RFC "
    num
    ))
  (literal "RFC UNKNOWN")
    )))
```

Pour un exemple réel, on peut regarder le programme DSSSL (en ligne sur `https://www.bortzmeyer.org/files/print-jres-2003.dsl`) que j'avais développé pour la réunion JRES de 2003. Comme beaucoup de conférence, celle-ci exigeait une présentation particulière des articles. Pour aider les auteurs, des feuilles de style étaient fournies pour certains environnements tels LaTeX mais rien pour Docbook. Avec DSSSL, il a été assez facile d'en développer une.