
Une sélection d'éléments docbook

Freeduc Sup <freeduc-sup@beaupeyrat.com>

Alix Mascret

Olivier Capuzzo

Relecture: Valérie Emin

O9 juin 2003

Présentation d'une liste restreinte d'éléments DocBook utilisés pour des 'polys' de la distribution freeduc-sup [<http://freeduc-sup.eu.org>].

Objectif visé : faciliter une prise en main rapide de ce système de rédaction de documents.

Table des matières

Présentation	2
Un document se décompose en sections	3
Objectif	3
éléments docbook	3
Exemple	3
Exemple de vue	4
Liste	4
Objectif	4
éléments docbook	4
Exemple	4
Exemple de vue	4
Référence labélisée	5
Objectif	5
éléments docbook	5
Exemple	5
Exemple de vue	5
Tableau	6
Objectif	6
éléments docbook	6
Exemple	6
Exemple de vue	7
Texte en italique ou en gras	7
Objectif	7
éléments docbook	7
Exemple	7
Exemple de vue	7
Listing	7
Objectif	8
élément docbook	8
Exemple	8
Exemple de vue	8
Copie écran console	9
Objectif	9
élément docbook	9
Exemple	9
Exemple de vue	9
Lien vers une ressource	9

Objectif	9
éléments docbook	10
Exemple	10
Exemple de vue	10
Image	10
Objectif	10
éléments docbook	10
Exemple	11
Exemple de vue	11
Remarques	11
Objectif	11
éléments docbook	11
Exemple	11
Exemple de vue	11
Jeu de questions-réponses	12
Objectif	12
éléments docbook	12
Exemple	12
Exemple de vue	13
Entête du document	13
Objectif	13
éléments docbook	13
Exemple	14
Annexe : comment transformer ?	15
sgmltools	15
xslt	15
Prise en main de la DTD DocBook	17
Création de l'entête	17
Le corps du document	18
Modifier le document	18
Transformer le document xml	19
Finaliser tout ça	19
Conclusion	20
Listing complet du tutoriel	20

Présentation

Ce projet est né sur la liste freeduc-sup, lors de l'élaboration des documents accompagnant la distribution. L'objectif initial était de réaliser une liste minimale et exhaustive des éléments de la DTD DocBook, utilisables pour la réalisation des documents. C'est avant tout un document de travail que nous nous sommes produit. Vous ne trouverez pas, dans ce document d'éléments de cours sur xml, xsl ou les processeurs xslt. Il y a pour cela d'autres sources de données. Ce document n'est qu'un outil pour démarrer rapidement un projet d'édition.

Connaissant assez bien DocBook, nous souhaitions ensuite voir dans quelle mesure il était possible de créer un script permettant de porter ces documents de façon simple selon la dtd de Logidée [<http://www.logidee.com/>]. En effet la société Logidée a développé pour ses propres besoins, un ensemble d'applications permettant la création de supports de cours (support étudiant, support enseignant, slides...). Ces applications sont intégrées à la liste des paquets officiels de Debian. Ils sont donc installables sur la distribution freeduc-sup comme n'importe quel paquet Debian.

La première partie du travail consistant à établir la liste des éléments utilisables dans la conception de document a été réalisée. C'est le document que vous lisez.

La seconde partie n'est à ce jour (09/2003) pas faite.

Vous trouverez donc dans ce document, les éléments de la dtd DocBook utilisables, une description du rôle de ces éléments, et la façon de les utiliser.

Vous trouverez également un tutoriel permettant, en quelques minutes, de créer un premier document au format XML, puis générer une sortie au format html ou pdf. Cela devrait vous

permettre de vous familiariser avec l'environnement et les manipulations de bases.

Ce document a été réalisé sous GNU/Linux, avec gvim [<http://www.vim.org>] et Emacs [<http://www.gnu.org/software/emacs/emacs.html>], au format DocBook [<http://www.oasis-open.org/docbook/>], mis en page avec le processeur XSLT saxon [<http://saxon.sourceforge.net>] développé par Michael Kay, les feuilles de styles de Norman Walsh [<http://nwalsh.com/>].

Le processeur XSLT saxon nécessite une machine virtuelle Java. Aucune machine virtuelle n'est installée par défaut sur la version actuelle de la freeduc-sup. Cela sera peut être fait un jour. Vous pouvez toutefois télécharger et installer le jdk de Sun. La procédure est donnée dans la section "Annexe de ce document".

Bien que vous ayez une description des procédures d'installation de Java et d'utilisation de Saxon dans ce document, vous n'aurez pas besoin de ces produits pour réaliser le tutoriel. Les outils de génération html et pdf que vous utiliserez sont fournis sur la distribution freeduc-sup, mais sont également disponibles sur toutes les distributions GNU/Linux. L'exploitation de tout ce qui est dans ce document est donc transposable sur tout environnement GNU/Linux.

Références

- DocBook: The Definitive Guide - copyright © by O'Reilly & Associates - est consultable ici <http://docbook.org/tdg> [<http://docbook.org/tdg/>]
- DocBook home page <http://www.oasis-open.org/docbook/>

Un document se décompose en sections

Objectif

Créer une structure arborescente.

éléments docbook

- `<sect1>`
section de niveau 1
- `<sect2>`
section de niveau 2
- `<sect3>`
section de niveau 3

Exemple

```
<sect3 id="section3" xreflabel="exemple de section 3">
  <title>un titre de section 3</title>
  <para>
    blabla section 3
  </para>
  <sect4 id="section3.1">
```

```
<title>un titre de section 4</title>
<para>
  blabla section 4
</para>
</sect4>
</sect3>
```

Exemple de vue

un titre de section 3

blabla section 3

un titre de section 4

blabla section 4

Liste

Objectif

Créer une structure de liste numérotée ou non.

éléments docbook

- **<itemizedlist>**
Liste à puce non numérotée
- **<orderedlist>**
Liste ordonnée par des numéros
- **<listitem>**
Un élément de liste (à puces ou à numéros)

Exemple

```
<para> Liste à puces</para>
<itemizedlist>
  <listitem><para>item a</para></listitem>
  <listitem><para>item b</para></listitem>
</itemizedlist>

<para>Liste ordonnée</para>
<orderedlist>
  <listitem><para>item a</para></listitem>
  <listitem><para>item b</para></listitem>
</orderedlist>
```

Exemple de vue

Liste à puces

- item a
- item b

Liste ordonnée

1. item a
2. item b

Référence labelisée

Objectif

Marquer une section, une note, afin d'y faire référence à plusieurs endroits dans le document.

éléments docbook

- **id**
un attribut de section ou de note. Sa valeur doit être unique dans le document.
- **xreflabel**
un attribut également. Sa valeur représentera le lien hypertexte.
- **<xref>**
permet d'insérer un lien hypertexte pointant vers une autre zone du document. La valeur textuelle du lien est définie par la valeur de son attribut `xreflabel`.
- **linkend**
Attribut de `xref`. Détermine le texte qui servira de lien hypertexte.

Exemple

```
<para>Ce document contient une référence labelisée :</para>
<programlisting>
  &lt;sect2 id="section3" xreflabel="exemple de section 3"&gt;
</programlisting>
<para>Exemple de lien vers <xref linkend="section3"/> .</para>
```

Exemple de vue

Ce document contient une référence labelisée :

```
<sect2 id="section3" xreflabel="exemple de section 3">
```

Exemple de lien vers exemple de section 3 .

Tableau

Objectif

Fournir une structuration tabulaire.

éléments docbook

- **<table>**
début la structure tableau.
- **<title>**
Le titre du tableau
- **<tgroup>**
Le tableau est structuré en groupes.
- **<thead>**
L'entête du tableau
- **<row>**
Déclare une ligne
- **<entry>**
Défini le contenu d'une cellule
- **<tbody>**
Défini le contenu du corps composé de cellules

Exemple

```
<table frame="all"><title>Un tableau</title>
<tgroup cols="6" align="left" colsep="1" rowsep="1">
<thead>
<row>
  <entry>Nom</entry>
  <entry>Prenom</entry>
  <entry>Telephone</entry>
  <entry>EMail</entry>
  <entry>Num Service</entry>
  <entry>Role</entry>
</row>
</thead>
<tbody>
<row>
  <entry>Kay</entry>
  <entry>Michael</entry>
  <entry>0122334455</entry>
  <entry>mk@saxon.org</entry>
  <entry>1</entry>
  <entry>XSLT and Co</entry>
</row>
</tbody>
</table>
```

```

</row><row>
  <entry>Stallman</entry>
  <entry>Richard</entry>
  <entry>1111111111</entry>
  <entry/>
  <entry>2</entry>
  <entry>Fondateur du projet GNU et de la FSF</entry>
</row>
</tbody>
</tgroup>
</table>

```

Exemple de vue

Tableau 1. Un tableau

Nom	Prenom	Telephone	EMail	Num Service	Role
Kay	Michael	0122334455	mk@saxon.org	1	XSLT and Co
Stallman	Richard	1111111111		2	Fondateur du projet GNU et de la FSF

Texte en italique ou en gras

Objectif

Mettre en avant des mots ou phrases.

éléments docbook

- `<emphasis>`
mettre en italique
- `<emphasis role="bold">`
mettre en gras en valorisant l'attribut `role` à "bold"

Exemple

```

<para>
  ici en <emphasis>italic</emphasis>,
  ici en <emphasis role="bold">gras</emphasis>
</para>

```

Exemple de vue

ici en *italic*, ici en **gras**

Listing

Objectif

Présenter du code (fonction, module, classe, etc.).

élément docbook

- `<programlisting>`

Permet d'appliquer une police non proportionnelle afin de conserver la mise en page du document. Par commodité, on place parfois le code dans une section DATA (`<![CDATA[du code]]>`), afin d'éviter de redéfinir le symbole `<` en `<`;

Exemple

```

<para><emphasis role="bold">Une fonction en java</emphasis></para>
<programlisting>
Nombre de caractères imprimables sur une ligne (format pdf ou postscript) :
0-----1-----2-----3-----4-----5-----6-----7-----
75 caractères maximum pour ne pas avoir les lignes tronquées par les marges

/*
@param nameFile le nom du fichier initial
@param newExt la nouvelle extension
@return le nom du fichier initial avec une nouvelle extension.
Dans le cas où il n'y a pas d'extension dans le nom initial,
l'extension est ajoutée.
exemple : changeExtension("toto.txt", "xml") rend "toto.xml"
exemple : changeExtension("toto", "xml") rend "toto.xml"
exemple : changeExtension("", "xml") rend ".xml"
*/
public String changeExtension(String nameFile, String newExt) {
    String res;
    int posPt = nameFile.lastIndexOf('.');
    if (posPt > 0)
        res = nameFile.substring(0, posPt) + "." + newExt;
    else
        res = nameFile + "." + newExt;
    return res;
}
</programlisting>

```

Exemple de vue

Une fonction en java

```

Nombre de caractères imprimables sur une ligne (format pdf ou postscript) :
0-----1-----2-----3-----4-----5-----6-----7-----
75 caractères maximum pour ne pas avoir les lignes tronquées par les marges

/*
@param nameFile le nom du fichier initial
@param newExt la nouvelle extension
@return le nom du fichier initial avec une nouvelle extension.
Dans le cas où il n'y a pas d'extension dans le nom initial,
l'extension est ajoutée.
exemple : changeExtension("toto.txt", "xml") rend "toto.xml"
exemple : changeExtension("toto", "xml") rend "toto.xml"
exemple : changeExtension("", "xml") rend ".xml"
*/
public String changeExtension(String nameFile, String newExt) {

```

```
String res;
int posPt = nameFile.lastIndexOf('.');
if (posPt > 0)
    res = nameFile.substring(0, posPt) + "." + newExt;
else
    res = nameFile + "." + newExt;
return res;
}
```

Copie écran console

Objectif

Présenter un état de la console.

élément docbook

- `<screen>`

Présenter le résultat d'un programme sur la sortie standard, ou une séquence de commandes système. Appliquer une police fixe.

Exemple

```
<para>Exemple de sortie console :</para>
<screen>
$ cat bin/makeHtml.sh
java -classpath /home/kpu/xsl/saxon/lib/saxon.jar
  com.icl.saxon.StyleSheet -o $2
  $1 /home/kpu/xsl/stylesheets/docbook-xsl-1.60.1/xhtml/docbook.xsl
$ makeHtml.sh selection-elts.docbook selection-elts.html
$
</screen>
```

Exemple de vue

Exemple de sortie console :

```
$ cat bin/makeHtml.sh
java -classpath /home/kpu/xsl/saxon/lib/saxon.jar
  com.icl.saxon.StyleSheet -o $2
  $1 /home/kpu/xsl/stylesheets/docbook-xsl-1.60.1/xhtml/docbook.xsl
$ makeHtml.sh selection-elts.docbook selection-elts.html
$
```

Lien vers une ressource

Objectif

Proposer un lien vers une ressource.

éléments docbook

- **<ulink>**
nom du lien
- **url**
Attribut de `ulink`, sa valeur localise la ressource.

Exemple

```
<para> <ulink url="./selection-dbk.xml">un lien vers une ressource ici</ulink>  
<para> <ulink url="http://www.linux-france.org/prj/edu/archinet/"><citetitle>u  
autre lien ici</citetitle></ulink> </para>
```

Exemple de vue

un lien vers une ressource ici [`./selection-dbk.xml`]

un autre lien ici [`http://www.linux-france.org/prj/edu/archinet/`]

Image

Objectif

Insérer une image.

éléments docbook

- **<figure>**
Permet le plus souvent une numérotation des figures dans le document. Pour l'impression papier (pdf ou postscript), il faut faire attention à la taille des images afin qu'elles ne soient pas tronquées sur les marges. Une largeur de 15 cm est généralement correcte. Pour le format, préférer le format "png".
- **<title>**
Élément de l'élément `figure`. Le nom de la figure.
- **<graphic>**
Concerne l'image à insérer.
- **srccredit**
Attribut de `graphic`. Information sur la source du document image.
- **fileref**
Attribut de `graphic`. Nom du fichier image.

Exemple

```
<figure><title>Classe VoyageTibet</title>
<graphic srccredit="kpu" fileref="voyageTibet.png"/>
</figure>
```

Exemple de vue

Figure 1. Classe VoyageTibet

Remarques

Objectif

Attirer l'attention par une note de quelques lignes.

éléments docbook

- `<note>`
Information en relation avec le texte courant, mais de portée plus générale.
- `<important>`
Une note importante.

Exemple

```
<para>Un exemple de note.</para>
<note><title>Envoi de message</title>
<para>L'instruction suivante :</para>
<programlisting>
hello.exprimeToi();
</programlisting>
<para>Se lit de plusieurs façons, en voici deux équivalentes :</para>
<orderedlist>
  <listitem><para>
    <emphasis>J'invoque la méthode</emphasis> <literal>exprimeToi</literal> de l'
  </para></listitem>
  <listitem><para>
    <emphasis>J'envoie le message </emphasis> <literal>exprimeToi</literal> à l'
  </para></listitem>
</orderedlist>
</note>
<para>Un exemple de remarque importante.</para>
<important><title>Brevet logiciel</title>
<para>La créativité en danger. S'informer <ulink url="http://swpat.ffii.org/in
</important>
```

Exemple de vue

Un exemple de note.

Envoi de message

L'instruction suivante :

```
hello.exprimeToi();
```

Se lit de plusieurs façons, en voici deux équivalentes :

1. *J'invoque la méthode* `exprimeToi` de l'objet référencé par la variable `hello`.
2. *J'envoie le message* `exprimeToi` à l'objet référencé par la variable `hello`.

Un exemple de remarque importante.

Brevet logiciel

La créativité en danger. S'informer ici [<http://swpat.ffii.org/index.fr.html#intro>].

Jeu de questions-réponses

Objectif

Idéal pour la construction de FAQ.

éléments docbook

- `<qandaset>`
Élément racine d'un jeu de Q/R
- `<qandaentry>`
Structure contenant une question et ses éventuelles reponses (0 à n)
- `<question>`
Une question. Élément enfant de `qandaentry`
- `<answer>`
Une des réponses à la question. Élément enfant de `qandaentry`

Exemple

```
<qandaset>
<qandaentry>
<question>
<para>
To be, or not to be?
</para>
</question>
```

```
<answer>
<para>
That is the question.
</para>
</answer>
</qandaentry>
<qandaentry>
<question>
<para>Comment changer le format de papier par défaut ?</para>
</question>
<answer>
<para>Allouer la valeur <literal>A4</literal> à 'paper.type'</para>
</answer>
<answer>
<para>Dans votre fichier de redéfinition des règles : </para>
<para>&lt;xsl:param name="paper.type" select="A4" /&gt;</para>
</answer>
</qandaentry>
</qandaset>
```

Exemple de vue

1. To be, or not to be?

That is the question.
2. Comment changer le format de papier par défaut ?

Allouer la valeur A4 à 'paper.type'

Dans votre fichier de redéfinition des règles :

<xsl:param name="paper.type" select="A4" />

Entête du document

Objectif

Informations générales sur le document. L'exemple ci-dessous est celui utilisé comme source docbook de ce document. Voici quelques uns de ces éléments, pour les autres, référez vous à l'exemple ci-après.

éléments docbook

- **<article>**
Élément racine d'un article
- **<groupauthor>**
Pour rassembler les auteurs (<author>)
- **<author>**

Un des auteurs du document (voir l'exemple pour connaître les éléments retenus décrivant un auteur)

- **<othercredit>**

Une personne ou entité, autre qu'un auteur, ayant participé au document.

- **<date>**

Date de création du document

- **<pubdate>**

Dernière date de publication

- **<abstract>**

Un résumé du document, contenant également des infos légales.

- **<keywordset>**

Un ensemble de mots clés pouvant être utilisé pour un référencement.

- **<keyword>**

Un mot clé.

Exemple

```
<articleinfo>
<title>Une sélection d'éléments docbook</title>
<authorgroup>
<author>
  <firstname>Freeduc</firstname>
  <surname>Sup</surname>
  <affiliation>
    <address><email>freeduc-sup@beaupeyrat.com</email></address>
  </affiliation>
</author>
<author>
  <firstname>Alix</firstname>
  <surname>Mascret</surname>
</author>
<author>
  <firstname>Olivier</firstname>
  <surname>Capuzzo</surname>
</author>
</authorgroup>

<othercredit>
  <firstname>Valérie</firstname>
  <surname>Emin</surname>
  <contrib>Relecture</contrib>
</othercredit>

<date>01 juin 2003</date>
<pubdate>09 juin 2003</pubdate>
<abstract>
  <para>
    Présentation d'une liste restreinte d'éléments DocBook
    utilisés pour des 'polys' de la distribution
    <ulink url="http://freeduc-sup.eu.org">freeduc-sup</ulink>.</para>
```

```
<para> Objectif visé : faciliter une prise en main rapide de
ce système de rédaction de documents.</para>
</abstract>
<keywordset>
  <keyword>Open source</keyword>
  <keyword>DocBook</keyword>
</keywordset>
</articleinfo>
```

Annexe : comment transformer ?

Il faut, avant toute transformation, valider le document source. Cela est réalisé grâce à un parseur (parser en Anglais). On peut comparer cette opération à la validation grammaticale et syntaxique que réalise un compilateur sur un source. Des exemples d'outils et la façon de les utiliser sont donnés dans la partie tutoriel. Aucune transformation ne sera possible si le document n'est pas "valide", ou au mieux "bien formé".

Il existe 2 techniques : une basée sur les sgmltools (outils de transformation basés sur DSSSL et le système jade) et l'autre sur xml et xslt.

sgmltools

sgmltools : un ensemble d'outils de transformation basés sur jade et disponible sur les principales distributions GNU/Linux.

Exemple d'utilisation :

```
$ db2html test.xml
output is test
Using catalogs: /etc/sgml/xml-docbook-4.1.2-1.0-14.cat
Using stylesheet: /usr/share/sgml/docbook/utils-0.6.11/docbook-utils.dsl#html
Working on: /home/kpu/docbook/test.xml
$
```

Suite à cette commande, le répertoire ./test est créé et contient les documents HTML.

Autre exemple, conversion vers du PDF

```
$ db2pdf test.xml
Using catalogs: /etc/sgml/xml-docbook-4.1.2-1.0-14.cat
Using stylesheet: /usr/share/sgml/docbook/utils-0.6.11/docbook-utils.dsl#print
Working on: /home/kpu/docbook/test.xml
$
```

Le document test.pdf est créé dans le répertoire courant.

xslt

Un document DocBook est un document XML valide (qui respecte la DTD DocBook).

Pour transformer une document XML, nous utilisons une à plusieurs feuilles de style XSLT et un processeur XSLT.

Il nous faut donc :

- **Un processeur XSLT**

Nous présenterons quelques exemples qui utilisent :

- saxon : un processeur xslt écrit en java par Michael Kay [<http://saxon.sourceforge.net>].
- xsltproc : un processeur xslt écrit en C (voir ici [<http://xmlsoft.org/XSLT/>]).
- Un document source en DocBook
- Une ou plusieurs feuilles de styles basées sur la DTD DocBook.

Vous trouverez une collection de feuilles des styles xslt <http://docbook.sourceforge.net/projects/xsl/> (choisir *XSL stylesheet distribution*) permettant de traiter des documents DocBook en différents formats.

Il existe d'autres processeurs XSLT écrits en C ou en C++ comme xalan [<http://xml.apache.org>], sablotron [<http://http://www.gingerall.com/>] et bien d'autres encore. Une liste de produits et outils autour d'xml est accessible ici [<http://www.garshol.priv.no/download/xmltools/>].

Exemple de génération d'un document HTML

```
CLASSPATH=saxon.jar:$CLASSPATH
export CLASSPATH
java com.icl.saxon.StyleSheet filename.xml \
    docbook-xsl-1.60.1/html/docbook.xsl > output.html
```

On suppose ici que `saxon.jar` est situé dans le répertoire courant, et que ce même répertoire contient un répertoire DocBook correspondant au résultat de la décompression du fichier `docbook-xsl-1.60.1.tar.gz` - voir ici [<http://nwalsh.com/>].

Avec `saxon` vous devez préalablement avoir installé un JRE (*java runtime engine*) ou un JDK (*java development kit*) - qui inclut un JRE.

Vous avez un document qui décrit la procédure de téléchargement et d'installation de Java sur une machine GNU/Linux ici [<http://www.linux-france.org/prj/edu/archinet/DA/install-java/>].

Sous GNU/Linux, vous pouvez plus simplement utiliser le processeur `xsltproc` :

```
xsltproc docbook-xsl-1.60.1/html/docbook.xsl filename.xml > output.html
```

Attention, l'ordre des arguments est inversé par rapport aux outils sous java.

Exemple de génération d'un document PDF

Pour produire un document PDF, on s'appuie sur FOP, un projet libre en open source de la fondation Apache <http://xml.apache.org/fop/> [<http://xml.apache.org/fop/>]. Pour cela on réalise une opération en 2 passes : une pour générer un fichier FO (formatting object) et une deuxième pour transformer en PDF.

```
CLASSPATH=saxon.jar:fop.jar:$CLASSPATH
export CLASSPATH
java com.icl.saxon.StyleSheet filename.xml \
    docbook-xsl-60.1/fo/docbook.xsl > output.fo
java org.apache.fop.apps.CommandLine output.fo output.pdf
```

Prise en main de la DTD DocBook

Cette partie doit vous permettre de prendre en main rapidement l'environnement de développement de documentation disponible sur la distribution freeduc-sup. Quelques adaptations mineures seront sûrement nécessaires sur d'autres distributions.

Nous allons utiliser Emacs comme éditeur car il est configuré pour supporter le mode psgml [<http://sourceforge.net/projects/psgml/>] qui est un ensemble de macros simplifiant l'édition de documents xml.

Voir aussi la documentation [http://www.lysator.liu.se/~lenst/about_psgml/psgml.html] des macros.

La DTD DocBook est supposée installée sur votre système ici : `/usr/share/sgml/docbook/dtd/xml/4.2/docbookx.dtd`. Si ce n'est pas le cas, vous pouvez vous la procurer ici : <http://www.oasis-open.org/docbook/xml/>, choisir DocBook XML V4.2 en archive ZIP que vous décompresserez.

Mettez vous dans un répertoire de travail et copiez l'image "voyageTibet.png" qui fait partie de ce document dans le répertoire.

Ouvrez, dans ce répertoire un nouveau document avec emacs :

```
$>emacs index.xml&
```

Création de l'entête

Tout document commence par une déclaration. Vous pouvez saisir ou copier l'extrait ci-dessous :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD docbook XML V4.2//EN"
"/usr/share/sgml/docbook/dtd/xml/4.2/docbookx.dtd">

<article lang="fr">
<articleinfo>
  <title>Titre de l'article</title>
  <author>
    <firstname>Votre Nom</firstname>
    <surname>Votre prénom</surname>
    <affiliation>
      <address><email>Votre adresse de mël</email></address>
    </affiliation>
  </author>
  <date>Ici la date de conception</date>
  <pubdate>Ici la date de publication ou de dernière modification</pubdate>
  <abstract>
    <para>Résumé ou présentation succincte de l'article</para>
  </abstract>
</articleinfo>
<sect1><title></title>
<para>
</para>
</sect1>
</article>
```

Vous pourrez ensuite l'adapter et le garder comme squelette pour tous les documents que vous aurez à réaliser.

A ce stade vous avez déjà un document complet, valide, qui contient déjà une section bien que celle-ci soit vide.

Le corps du document

Mettez un titre à la section "sect1", entre les éléments "title" et "/title". Saisissez ensuite ou copier le morceau de texte ci-dessous qui permet d'insérer un morceau de texte et une image.

```
<sect1><title>Creation d'un paragraphe</title>
<para>Pour insérer un élément a partir d'Emacs avec le mode psgml,
    il suffit de taper "CTRL c i", puis entrer l'élément a inserer,
    par exemple "para". </para>
    <para>Pour fermer un élément qui est ouvert ,
    il faut positionner le curseur à l'endroit où doit venir la balise
    de fermeture et taper "CTRL /". </para>

<para>Pour contrôler la validité du document directement à partir d'Émacs
    il faut utiliser "CTRL c v".</para>
<para>Une autre option consiste à utiliser par exemple
un parseur externe comme "xmllint" en ligne de commande.
</para>
<programlisting>
$ xmllint --noout --loaddtd --valid index.xml
</programlisting>

<para>Ajoutons une image :
<figure>
    <title>Voyage au Tibet</title>
    <graphic fileref="voyageTibet.png"/>
</figure>
</para>
</sect1>
```

La première partie du document est terminée. Il faut vérifier qu'il est conforme a la dtd DocBook. Pour cela il faut utiliser un "parser".

Emacs permet de faire cela assez simplement avec la commande "CTRL c v".

Le résultat du contrôle apparaît dans une fenêtre du bas. Si des erreurs apparaissent il faut les corriger. En général ce sont des balises de fermetures qui manquent.

Voici un exemple d'erreur :

```
nsgmls:index.xml:35:9:E: end tag for "sect1"
    omitted, but OMITTAG NO was specified
nsgmls:index.xml:21:0: start tag was here
```

Ici il manque une balise "sect1" ligne 35 pour fermer la balise qui ouvre ligne 21.

Deux warnings peuvent apparaître que vous pouvez laisser, ils n'influeront en rien sur le document final.

```
nsgmls:/usr/share/sgml/declaration/xml.dcl:1:W:
    SGML declaration was not implied
nsgmls:/usr/share/sgml/declaration/xml.dcl:31:27:W: characters in the document
    character set with numbers exceeding 65535 not supported
```

Vérifier la bonne structure du document et corrigez les éventuelles erreurs.

Modifier le document

Nous allons rajouter un deuxième niveau de section contenant une liste. Nous allons le mettre juste avant la balise de fermeture "/sect1".

```
<sect2><title>Compilation du source xml</title>
<orderedlist>
  <listitem><para>
    db2html index.xml pour générer un document html sur "n" pages
  </listitem>
  <listitem><para>
    db2html -u index.xml pour générer un document html sur une seule page
  </listitem>
  <listitem><para>db2pdf index.xml pour produire une sortie pdf
  </listitem>
</orderedlist>
<para>Pensez, pour les documents html, à copier l'image dans
le répertoire de destination. Dans notre exemple il s'agit du
répertoire "index".</para>
</sect2>
```

Enfin nous allons compléter le document avec une autre section "sect1" faisant référence à la licence qui couvre ce document :

```
<sect1>    <title>Licence</title>
<para>Ce document est couvert par la licence
  <ulink url="http://www.gnu.org/copyleft/gpl.html">GPL</ulink>.</para>
<para>Le site officiel est <ulink url="http://ici mettre l'url">ici</ulink>.</para>
<para>Vous pouvez utiliser, copier, modifier, distribuer librement ce
document. Son utilisation n'engage en aucune façon la responsabilité de son
auteur.</para>
</sect1>
```

Transformer le document xml

Il ne reste plus qu'à transformer ce document xml au format html ou pdf. Pour réaliser la génération, mettez vous dans le répertoire où se trouve le document et utilisez :

```
db2html index.xml pour générer un document html sur "n" pages
db2html -u index.xml pour générer un document html sur une seule page
db2pdf index.xml pour produire une sortie pdf
```

La génération html se fait par défaut dans le répertoire "index" (nom du document). Si vous utilisez deux fois de suite la commande "db2html", la version qui est dans le répertoire index sera détruite.

Si l'image n'apparaît pas dans le document html, vérifier qu'elle existe bien dans le répertoire index.

Finaliser tout ça

Il ne reste plus qu'à automatiser un peu tout ça. Pour cela il est possible de créer un "Makefile" qui permettra la génération automatique de tous les formats de sortie.

Nous allons en faire un tout simple qui va faire l'essentiel pour nous.

```
# Makefile, utilise dsssl
# À adapter pour xsl/xslt fo

# On considère que le fichier se nomme index.xml
```

```
FILE=index

all: txt html rtf pdf

clean:
    rm -rf index_multi_page index_mono_page *.pdf *.ps *.txt *.rtf

parse:
    nsgmls -sv -wxml /usr/share/sgml/declaration/xml.dcl index.xml

# Sortie txt
# Pour consulter avec more ou less, pas avec un éditeur
# ou alors virer la table des matières.
txt: htm
    html2text index_mono_page/${FILE}.html > ${FILE}.txt

# Sortie html sur un seul fichier html
htm:
    db2html -o index_mono_page -u index.xml
    cp -Rf images index_mono_page

# Sortie html sur plusieurs pages
html:
    db2html -o index_multi_page index.xml
    cp -Rf images index_multi_page

# Sortie pdf
# On passe par une génération postscript
# On édite les document 1 feuille par page et 2 feuilles par page
pdf:
    db2ps ${FILE}.xml
    psnup -pletter -Pa4 -b.05 -m.01 -2 ${FILE}.ps ${FILE}_2p.ps
    ps2pdf ${FILE}.ps
    ps2pdf ${FILE}_2p.ps

# Sortie RTF
rtf:
    db2rtf ${FILE}.xml
```

Vous pouvez utiliser tout simplement ce Makefile avec les commandes "make parse", "make clean", "mke rtf", "make all" ...

La génération d'un document de plus de 200 pages est réalisée comme ça automatiquement dans tous les formats de sortie en moins de 2 minutes sur ma machine.

Conclusion

Si vous êtes arrivé jusque là, votre environnement fonctionne correctement. Il ne reste plus qu'à se familiariser un peu plus avec la liste des éléments DocBook, le mode psgml, et tester d'autres processeurs XSLT comme Saxon.

Vous découvrirez avec le temps la joie de n'avoir à maintenir qu'un seul source de votre document, indépendamment des différents formats de sortie.

Le guide complet sur xsl DocBook [<http://www.sagehill.net/docbookxsl/index.html>] vous donnera toutes les indications détaillées sur l'utilisation de saxon, xalan, xsltproc, fop.

DocBook: The Definitive Guide [<http://www.docbook.org/tdg/en/>] de Norman WALSH, vous donnera tous les autres éléments sur la dtd de DocBook dont vous pourriez avoir besoin.

Listing complet du tutoriel

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"/usr/share/sgml/docbook/dtd/xml/4.2/docbookx.dtd">

<article lang="fr" id="index.html">
<articleinfo>
  <title>Titre de l'article</title>
  <author>
    <firstname>Votre Nom</firstname>
    <surname>Votre prénom</surname>
    <affiliation>
      <address><email>Votre adresse de mèl</email></address>
    </affiliation>
  </author>
  <date>Ici la date de conception</date>
  <pubdate>
    Ici la date de publication ou de dernière modification</pubdate>
  <abstract>
    <para>Résumé ou présentation succincte de l'article</para>
  </abstract>
</articleinfo>
<sect1><title>Création d'un paragraphe</title>
  <para>Pour insérer un élément a partir d'emacs avec le mode psgml,
  il suffit de taper "CTRL c i", puis entrer l'élément à insérer,
  par exemple "para". </para>
  <para>Pour fermer un élément qui est ouvert ,
  il faut positionner le curseur à l'endroit où doit venir la balise
  de fermeture et taper "CTRL /. </para>
  <para>Pour contrôler la validité du document directement à partir d'Émacs
  il faut utiliser "CTRL c v".</para>
  <para>Une autre option consiste à utiliser par exemple
  un parseur externe comme "xmllint" en ligne de commande.
  </para>
  <programlisting>
  $ xmllint --noout --loadtd --valid index.xml
  </programlisting>

  <para>Ajoutons une image :

  <figure>
    <title>Voyage au Tibet</title>
    <graphic srccredit="vt" fileref="voyageTibet.png"/>
  </figure>
  </para>

  <sect2><title>Compilation du source xml</title>
  <orderedlist>
    <listitem><para>
      db2html index.xml pour générer un document html sur "n" pages
    </para></listitem>
    <listitem><para>
      db2html -u index.xml pour générer un document html sur une seule pag
    </para></listitem>
    <listitem><para>
      db2pdf index.xml pour produire une sortie pdf
    </para></listitem>
  </orderedlist>
  <para>Pensez, pour les documents html, à copier l'image dans le répertoire
  de destination. Dans notre exemple il s'agit du répertoire "index".</para>
  </sect2>
</sect1>

  <sect1>
    <title>Licence</title>
    <para>Ce document est couvert par la licence
      <ulink url="http://www.gnu.org/copyleft/gpl.html">GPL</ulink>.
    </para>
  </sect1>

```

```
Le site officiel est <ulink url="http://ici mettre l'url">ici</ulink>.
</para>
<para>
Vous pouvez utiliser, copier, modifier, distribuer librement ce document.
Son utilisation n'engage en aucune façon la responsabilité de son auteur.
</para>
</sect1>

</article>
```