

# De Shoebox à XML

Shoebox est l'outil idéal pour la gestion de données lexicales. Il permet de gérer simultanément plusieurs bases de données au sein d'un même projet.

## Fichier texte

Une de ses particularités en tant que gestionnaire de bases de données est qu'il génère un fichier *pur texte*, dont les données sont délimités par des balises appelées *marqueurs de champ* (de type \xxx). Un marqueur particulier délimite chaque fiche lexicale, c'est le *marqueur d'enregistrement* défini dans le *type de base de données* choisi au moment de la création de celle-ci.

Lors de la saisie, il n'y a pas d'interclassement de la nouvelle fiche parmi celles déjà existantes : les données d'une nouvelle fiche s'ajoute simplement en fin de fichier.

Le logiciel peut présenter les enregistrements sous la forme :

- d'un *Index* ordonné alphabétiquement suivant le *champ de tri*, et dont chaque ligne représente le contenu d'une fiche, chaque champ dans une colonne. Le choix des champs affichés dans l'Index est facilement accessible. Un double-clic sur une ligne ouvre la fiche en saisie.
- d'une fenêtre de saisie contenant la fiche en cours, avec des boutons permettant de passer à la fiche suivante ou précédente...

marq. enr.....	\lx bää
cat. gram. ....	\ps n
définition .....	\gn Dieu
	\lx bääbūr
	\ps n
	\gn avion
	...

## Encodage de langue

Une deuxième particularité de ce logiciel est sa capacité à gérer des jeux de caractères ad hoc qui pourront être associés sélectivement aux différents champs d'une base de données. Un *encodage de langue* définit une police pour l'affichage des données, ainsi que l'*ordre de tri* des caractères qui le composeront.

De plus un *clavier virtuel* peut être associé à chaque jeu de caractères de façon à faciliter l'accès aux caractères spéciaux de la police choisie.

Chaque champ d'une base de données se voit alors affecter un encodage de langue afin que ses données s'affichent dans la police voulue.

Les encodages de langue constituent des fichiers de configuration autonomes (d'extension *.lng*) qui peuvent ainsi être réutilisés dans différents projets et pour différentes bases de données.

## Ordre de tri

Deux niveaux d'ordre de tri sont paramétrables dans un encodage de langue :

- un *ordre de tri primaire* qui ne différencie pas des caractères pouvant être considérés comme ayant la même valeur, comme par exemple en français : e, é, è, ê, ë qui ont tous la valeur du "e" par rapport aux autres lettres de l'alphabet
- et un *ordre de tri secondaire* qui, lui, tiendra compte de l'ordre de ces caractères pour ranger entre eux des quasi-homophones.

On aura ainsi en français (sur différentes lignes l'ordre principal, sur une même ligne l'ordre secondaire) :

A a à â  
B b  
C c Ç ç

...

L'encodage permet également de prendre en compte les *multigraphes*, par exemple en wolof :

Mm  
Mb mb  
N n  
Ng ng

Les digraphes "mb" et "ng" seront considérés chacun comme un caractère unique lors du processus de tri.

De même en spécifiant les caractères *diacritiques* (tons, nasalité...) qui pourront s'ajouter à des caractères de bases, ceux-ci pourront être ignorés comme caractères dans un ordre de tri primaire et servir pour l'ordre de tri secondaire.

Des caractères peuvent enfin être considérés comme nul ou à ignorer, par exemple le tiret "-" dans les mots composés ou d'autres caractères de découpage.

Plusieurs ordres de tri (orthographique, phonologique...) peuvent être définis pour un même encodage permettant ainsi d'afficher les entrées dans un ordre différent suivant le type de travail effectué.

## Définition des champs et Hiérarchie

La définition des champs d'une base de données consiste à choisir les marqueurs qui serviront à délimiter les différents champs, à établir une hiérarchie entre eux et à leur affecter un encodage de langue.

La hiérarchisation des champs n'est pas obligatoire, ils peuvent tous être au même niveau, sous le lexème, mais définir une hiérarchie permet de relier les données d'un champ à celles du champ parent. Ainsi si une entrée présente plusieurs catégories grammaticales (nom et adjectif par exemple) chacune d'elle pourra comporter un champ définition qui lui sera relative. Sans hiérarchie et dans une présentation en *Index* par exemple, les différentes définitions paraîtraient relatives à la première catégorie grammaticale trouvée dans la fiche.

La définition des champs est relative à un *type de bases de données* et constitue un fichier autonome (d'extension *.typ*) qui pourra ainsi être réutilisé pour créer différentes bases de données de même structure sans avoir à refaire le paramétrage des champs.

## Champ de Tri

Le *champ de tri* par défaut est bien entendu celui de l'entrée lexicale, mais tout autre champ peut être choisi. (Cela ne change en rien l'ordre des données dans le fichier, seul l'affichage est affecté par un changement de champ de tri. )

Lorsque le champ de tri est changé, chaque valeur du champ de tri fait l'objet d'une fiche. Par exemple en choisissant le champ "définition" comme nouveau champ de tri, une entrée lexicale comportant deux définitions (deux sens) sera dupliquée, une pour chacune des valeurs du champ "définition". Il s'agit des mêmes données mais elles apparaîtront à deux endroits dans la base. Le nombre de fiches de la base apparaîtra de ce fait plus important qu'il n'est réellement.

**Remarque :** Il est possible d'avoir simultanément plusieurs fenêtres présentant chacune la même base de données mais ordonnée suivant des champs de tri différents.

## Exportation en XML

La structure d'un fichier Shoebox présente une des caractéristiques essentielles d'un document XML à savoir un *fichier texte dont les données sont structurés par des balises*.

Un document XML est toutefois plus exigeant quand à sa syntaxe. En effet les différents champs du document doivent être encadrés par des paires de balises **ouvrantes et fermantes**, alors que dans Shoebox les balises fermantes sont implicites. Ainsi, par exemple, la balise fermante correspondant au marqueur d'entrée lexicale "`\lx`", c'est le prochain marqueur "`\lx`".

De même il n'y a pas de balise fermante correspondant au marqueur de catégorie grammaticale "\ps "; ce sera implicitement le prochain marqueur "\ps " ou bien le prochain marqueur de champ du même niveau hiérarchique que "\ps ". (Lorsqu'il n'y a pas de hiérarchie déclarée entre marqueurs, c'est implicitement le champ suivant qui ferme le précédent.)

L'exportation en XML d'une base de données Shoebox génère un fichier dans lequel les marqueurs de champs Shoebox sont remplacés par des balises XML, les balises fermantes étant ajoutées à la fin du contenu de chaque champ.

\lx	bàa	<?xml version="1.0" encoding="ISO-8859-1" ?>
\ps	n	- <database>
\sn	1	- <lxGroup>
\gn	Dieu	<lx>bàa</lx>
\ge	God	- <psGroup>
\sn	2	<ps>n</ps>
\gn	pluie	- <snGroup>
\ge	rain	<sn>1</sn>
\xv	<b>bàa rāa</b>	<gn>Dieu</gn>
\xn	Il pleut	<ge>God</ge>
\xe	It's raining	</snGroup>
\lt	Dieu/pleure	- <snGroup>
\lt	God/weeps	<sn>2</sn>
\xv	<b>bàa tūf sāarē</b>	<gn>pluie</gn>
\nn	Se dit en début de saison des pluies.	<ge>rain</ge>
\ne	Said at the beginning of the rainy season.	</snGroup>
\xn	pluie/crache/la salive	- <xvGroup>
\xe	rain/spits/saliva	<xv>bàa rāa</xv>
\f	<i>prov</i>	<xn>Il pleut</xn>
\lv	<b>?à jō? dē bàa jè kitè?ē gā</b>	<xe>It's raining</xe>
\ln	On ne sème pas avec la pluie du menteur.	<lt>Dieu/pleure</lt>
\le	One doesn't sow with the liar's rain.	<lt>God/weeps</lt>
\dt	04/Jul/2000	</xvGroup>
\da	1988	- <xvGroup>
		<xv>bàa tōf sYarĀ</xv>
		<nn>Se dit en début de saison des pluies.</nn>
		<ne>Said at the beginning of the rainy season.</ne>
		<xn>pluie/crache/la salive</xn>
		<xe>rain/spits/saliva</xe>
		</xvGroup>
		- <lfGroup>
		<lf>prov</lf>
		- <lvGroup>
		<lv>à jō? dē bàa jè kitè?ē gā</lv>
		<ln>On ne sème pas avec la pluie du menteur.</ln>
		<le>One doesn't sow with the liar's rain.</le>
		</lvGroup>
		</lfGroup>
		</snGroup>
		</psGroup>
		<dt>07/May/2004</dt>
		<da>1988</da>
		</lxGroup>

Nous voyons ici que le fichier Shoebox est converti en un document XML <database> dans lequel chaque fiche est délimitée par le couple de balises <lxGroup>...</lxGroup>.

La hiérarchie des marqueurs (définis dans le type de base de données choisi pour ce lexique dans Shoebox) est retranscrite ici par une imbrication des champs :

- Pour chaque entrée lexicale (\lx ), le premier niveau de la structure hiérarchique est la catégorie grammaticale (\ps ). Il est représenté en XML par le couple de balises <psGroup>...</psGroup> et se répète autant de fois qu'il y aura de catégories grammaticales différentes dans la fiche.
- Pour une catégorie grammaticale (\ps ), le niveau hiérarchique suivant est la définition ou sens (\ge, \gn) qui peut être multiple. Il y aura alors autant de couples <snGroup>...</snGroup> qu'il y aura de sens différents dans une même catégorie grammaticale.

- Pour un sens donné (`\sn`), il peut y avoir plusieurs exemples illustratifs (`\xv`) dans la langue source. Chaque exemple forme un groupe avec sa traduction dans les deux langues cibles (`\xe`, `\xn`), ses éventuelles traductions littérales (`\lt`) et ses notes explicatives (`\ne`, `\nn`), d'où les couples de balises `<xvGroup>...</xvGroup>` qui englobe ces différents champs.
- De même, pour un sens donné (`\sn`), il peut y avoir d'autres données supplémentaires susceptibles d'être apportées, comme des locutions ou proverbes utilisant le lexème en question. Pour maintenir groupés le type d'information et les informations de ce type, un couple de balises `<lfGroup>...</lfGroup>` est créé. Ainsi ici, plusieurs groupes de proverbes `<lvGroup>` comportant le proverbe lui-même `<lv>` et ses traductions `<ln>` et `<le>` pourront être regroupés sous la dénomination "prov".

## Caractères spéciaux, Unicode et XML

Comme nous le voyons dans l'exemple illustré précédent, les caractères spéciaux ne sont pas correctement passés de Shoebox à XML. Le jeu de caractères (*charset*) standard affecté au document XML est par défaut le jeu standard européen "ISO-8859-1". Ce jeu de 256 caractères ne recouvre bien évidemment pas l'ensemble des caractères spéciaux de la langue *Tupuri* de notre exemple.

*Unicode*, standard qui se charge de créer un système unique d'encodage pour tous les caractères du monde, nécessite pour ce faire un encodage de ces derniers sur plusieurs octets, or Shoebox ne gère que des données sur 8 bits. Par contre étant capable d'affecter à chaque champ un encodage de caractères, Shoebox est à même d'utiliser plusieurs polices 8-bits spéciales, associés à des systèmes alphabétiques différents.

S'il existe des jeux de caractères standards définis pour certains groupes linguistiques, comme ISO-8859-1, tel n'est pas le cas pour les langues ayant un accès récent à l'informatisation. Le développement d'Unicode a tendance à rendre obsolète de telles démarches de standardisation par zone linguistique dans la mesure où un standard unique pourra permettre à toutes les données codées suivant ce standard d'être correctement interprétées sur tous les systèmes gérant l'Unicode.

Notre préoccupation est donc de trouver une procédure pour recoder les données issues de Shoebox dans le standard Unicode. En effet, Shoebox ne connaît pas encore, à notre connaissance, de logiciel équivalent sachant gérer Unicode, et donc nous continuerons à l'utiliser en état, mais il nous revient d'assurer la transition des données vers XML, qui est le format d'échange de données privilégié des nouvelles technologies de information, à condition que ces données soient recodées en Unicode.

## Unicode et UTF-8

En plus d'attribuer un code unique à chaque caractère, Unicode spécifie trois formes d'encodage permettant de transmettre des données à raison d'un, deux ou quatre octets par code de caractère. Chacune de ces formes d'encodage peut servir à encoder le même caractère et peut être transposée sans perte vers l'une ou l'autre forme d'encodage.

*UTF-8* est très populaire en HTML et dans les protocoles Internet. UTF-8 est un moyen de transformer tous les caractères Unicode dans un encodage en un nombre d'octets variable. Il présente l'avantage que les caractères Unicode correspondant au jeu de caractères ASCII (de 32 à 127) ont la même valeur qu'en ASCII et que les autres caractères Unicode transformés en UTF8 peuvent être gérés avec plus de facilité par la plupart des logiciel existants, sans trop de réécriture de logiciels.

*UTF-16* est apprécié dans les environnements où un juste équilibre doit être trouvé entre la souplesse d'accès aux données et l'économie d'espace de stockage. Il est raisonnablement compact, et les caractères les plus utilisés se trouvent codé sur 16 bits (2 octets), les autres étant accessibles par l'intermédiaire de couple d'unités de 16 bits.

UTF-32 est utilisé là où le problème de la mémoire de stockage ne se pose pas. Chaque caractère Unicode est encodé sur une unité de 32 bits (4 octets).

C'est donc UTF8 qui sera notre choix d'encodage des caractères dans les fichiers XML issus de l'exportation de données de Shoebox.

## Méthodologie de transcodage

Deux solutions se présentent à nous pour transcoder les caractères spéciaux en UTF-8.

- En aval, en travaillant sur le fichier XML produit par Shoebox. Il s'agira donc de recoder les caractères étendus (au delà de 127) des champs relevant de la langue Tupuri. Pour cela il faudrait programmer un outil par lequel les balises concernées seraient précisées, ainsi que la table de conversion des caractères de leurs codes 8 bits dans la police spéciale, vers leurs codes UTF-8.
- En amont, en recodant en UTF-8 la base de données Shoebox d'origine, puis en exportant ce fichier recodé en XML.

Si la première solution paraît plus logique puisqu'elle préserve la base de données Shoebox d'origine, elle nécessite la programmation de l'outil.

Or un outil semblable existe déjà, mais qui sait transcoder le contenu de **champs Shoebox** suivant des tables de correspondances 8-bits vers Unicode. Il s'agit de l'utilitaire *SFConv* du paquetage *TECkit* de la SIL.

Cet outil accepte en entrée un fichier au Format Standard (Fichier Shoebox en l'occurrence), et comme paramètre un fichier spécifiant les tables de correspondances à utiliser pour les différents marqueurs de champs. Ainsi pour chaque champ d'un fichier Shoebox, on pourra faire référence à une table de conversion mettant en correspondance les codes de la police spéciale utilisée pour ce champ et les codes Unicode correspondant. Ces tables de correspondance sont à établir une fois pour toutes pour chaque police utilisée et seront utilisables chaque fois que nécessaire. *SFConv* accepte un autre paramètre qui lui précise la forme d'encodage de sortie souhaitée, on choisira alors UTF-8.

Exemple de ligne de commande :

```
SFConv -8U -c conv.xml -i tupuri.db -o tupuriU.db
```

Le fichier *conv.xml* est un fichier au format XML spécifiant les fichiers de transformatage (.map) à utiliser suivant les champs. Par exemple :

```
<?xml version="1.0"?>
<!--Copyright (c) 2002 SIL International-->
<sfConversion defaultMapping="Windows-1252">
  <sfMarkers escape="\\"
    chars="abcdefghijklmnopqrstuvwxyz_ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    mapping="Windows-1252">
    <marker name="\x" mapping="Tupuri"/>
    <marker name="\xv" mapping="Tupuri"/>
    <marker name="\lv" mapping="Tupuri"/>
  </sfMarkers>
</sfConversion>
```

Ici, le contenu de tous les champs seront convertis suivant le fichier de transformatage *Windows-1252.tec*, excepté les champs \lx, \xv et \lv qui eux seront transcodés suivant *Tupuri.tec*.

Echantillons du contenu des fichiers de transformatage dans leurs versions non-compilées (.map)

### **;Windows-1252.map**

```
;Copyright (c) 2002 SIL International.
EncodingName      'WINDOWS-1252'
DescriptiveName   'Windows code page 1252'
```

```

ByteDefault      '?'
UniDefault      replacement_character
RHSFlags        (ExpectsNFC)

```

```

ByteClass [ascii] = ( 0 .. 127 )
UniClass [ascii] = ( U+0000 .. U+007f )
ByteClass [latin1] = ( 0xa0 .. 0xff )
UniClass [latin1] = ( U+00a0 .. U+00ff )

```

```

[ascii] <> [ascii]
[latin1] <> [latin1]

```

```

0x80 <> euro_sign
;0x81      undefined
0x82 <> single_low_9_quotation_mark
0x83 <> latin_small_letter_f_with_hook
0x84 <> double_low_9_quotation_mark
0x85 <> horizontal_ellipsis
...

```

### **;Tupuri.map**

```

EncodingName      "Tupuri"
DescriptiveName   "Tupuri"

```

```

ByteClass[CTL]=(0..128)
UniClass[CTL]=(0..128)
[CTL] > [CTL]

```

```

130 > U+030F
131 > U+0283
...
235 > U+0065 U+030F
236 > U+00EC
237 > U+00ED
238 > U+00EE
239 > U+0131 U+030F

```

Tupuri	UTF-8/décimal	UTF-8
è	e/204/143/	eï
ì	195/172/	Ã¬
î	195/173/	Ã-
ï	195/174/	Ã@
ÿ	196/177/204/143/	Ä±

Une fois la base de données Shoebox transcodée en UTF-8 par SFConv, elle sera réouverte dans Shoebox puis exportée en XML. Evidemment les données affichées par Shoebox ne seront plus correctement affichées puisque converties en UTF-8 que Shoebox ne sait pas interprétées, mais ceci ne pose pas de problème pour l'exportation. Par contre lors du passage à XML, Shoebox remplace les caractères étendus (au delà de 127) par des entités HTML ("í" devient &#237;), alors qu'ils devraient être conservés tels quels pour être interprétables correctement en UTF-8. Il faudra alors recourir, dans les options d'exportation en XML, à une table de remplacement CCT qui restituera les caractères UTF-8 juste après l'exportation.

Extrait de cette table :

```

"&#195;" > "Ã"
"&#196;" > "Ä"
"&#197;" > "Å"
"&#198;" > "Æ"

```

## Récapitulatif des différentes étapes

Shoebox	<code>\lx gètél-niini</code>	<code>\gn brosse à meule</code>
Texte	<code>\lx gètÁl-niini</code>	<code>\gn brosse à meule</code>
XML	<code>&lt;lx&gt;g&amp;#235;t&amp;#193;l-n&amp;#239;in&amp;#237;&lt;/lx&gt;</code>	<code>&lt;gn&gt;brosse&amp;#224; meule&lt;/gn&gt;</code>
sans conversion		
Shoebox UTF-8	<code>\lx geİ tÉ`İ l-nÄ±İ inÃ-</code>	<code>\gn brosse Ã meule</code>
XML UTF-8	<code>&lt;lx&gt;geİ tÉ`İ l-nÄ±İ inÃ-&lt;/lx&gt;</code>	<code>&lt;gn&gt;brosse Ã meule&lt;/gn&gt;</code>
Netscape/IE6	<code>&lt;lx&gt;ge t `l-nı ini&lt;/lx&gt;</code>	<code>&lt;gn&gt;brosse à meule&lt;/gn&gt;</code>

## Toolbox

Maintenant que Toolbox prend la relève de Shoebox en gérant l'Unicode, la base de données transcodée en UTF-8 peut être directement ouverte et affichée correctement grâce à un encodage de langue Unicode.

Pour cela il faudra créer

- un nouveau type pour le lexique avec `\lx` comme marqueur d'enregistrement
- un encodage de langue Unicode pour le Tupuri (ordre alphabétique...)

Une fois le fichier UTF-8 importé dans Toolbox sous le nouveau type, on affectera l'encodage Tupuri à chaque champ en langue Tupuri. Le lexique peut maintenant être géré directement en Unicode et être exporté en XML, encodage UTF-8.