

BERNARD PROST



XML POUR L'ÉDITION

STRUCTURER • SAISIR • PUBLIER

EYROLLES

a'sfired

XML POUR L'ÉDITION

STRUCTURER • SAISIR • PUBLIER



La collection a'sfored

Cette collection résulte d'un partenariat avec l'Asfored, le centre de formation de l'édition. Experts dans leur domaine et pédagogues reconnus, les auteurs y livrent leurs méthodes de travail et conseils «métier». Un contenu précieux pour tous les professionnels qui conçoivent, produisent ou diffusent des supports imprimés ou multimédias.

Pour en savoir plus sur le centre de formation et d'expertise de l'édition : www.asfored.org.

L'édition vit une mutation dont la portée est considérable : le support papier voit arriver la concurrence de supports dématérialisés chaque jour plus nombreux (Web, *netbooks*, tablettes, téléphones, *readers*...). Pour l'éditeur, il s'agit désormais de publier pour ces supports en tenant compte de leurs spécificités, tout en minimisant les coûts et les délais de production.

Cet ouvrage, qui s'adresse aux éditeurs, aux lecteurs/correcteurs et aux fabricants, mais également aux managers qui souhaitent mieux comprendre les techniques sous-jacentes et l'influence du support sur la conception et la forme de l'objet de publication numérique, vise à donner les clés d'utilisation de XML dans la chaîne éditoriale. Il s'appuie sur un exemple concret d'article encyclopédique repris au fil des chapitres et qui répond à un cahier des charges éditorial simple :

- bibliographies, filmographies et discographies interactives ;
- publication sur papier, sur le Web, le téléphone, les tablettes...

L'auteur

Bernard Prost, diplômé de Centrale et de l'Institut d'études politiques, est le fondateur de Ligarane, société spécialisée dans la production de publications numériques sur une base XML, grâce à laquelle il accompagne les éditeurs (édition, presse) dans la mutation numérique de leur activité. Il anime à l'Asfored les stages XML pour l'édition et Bâtir sa stratégie d'édition multisupport.

XML POUR L'ÉDITION

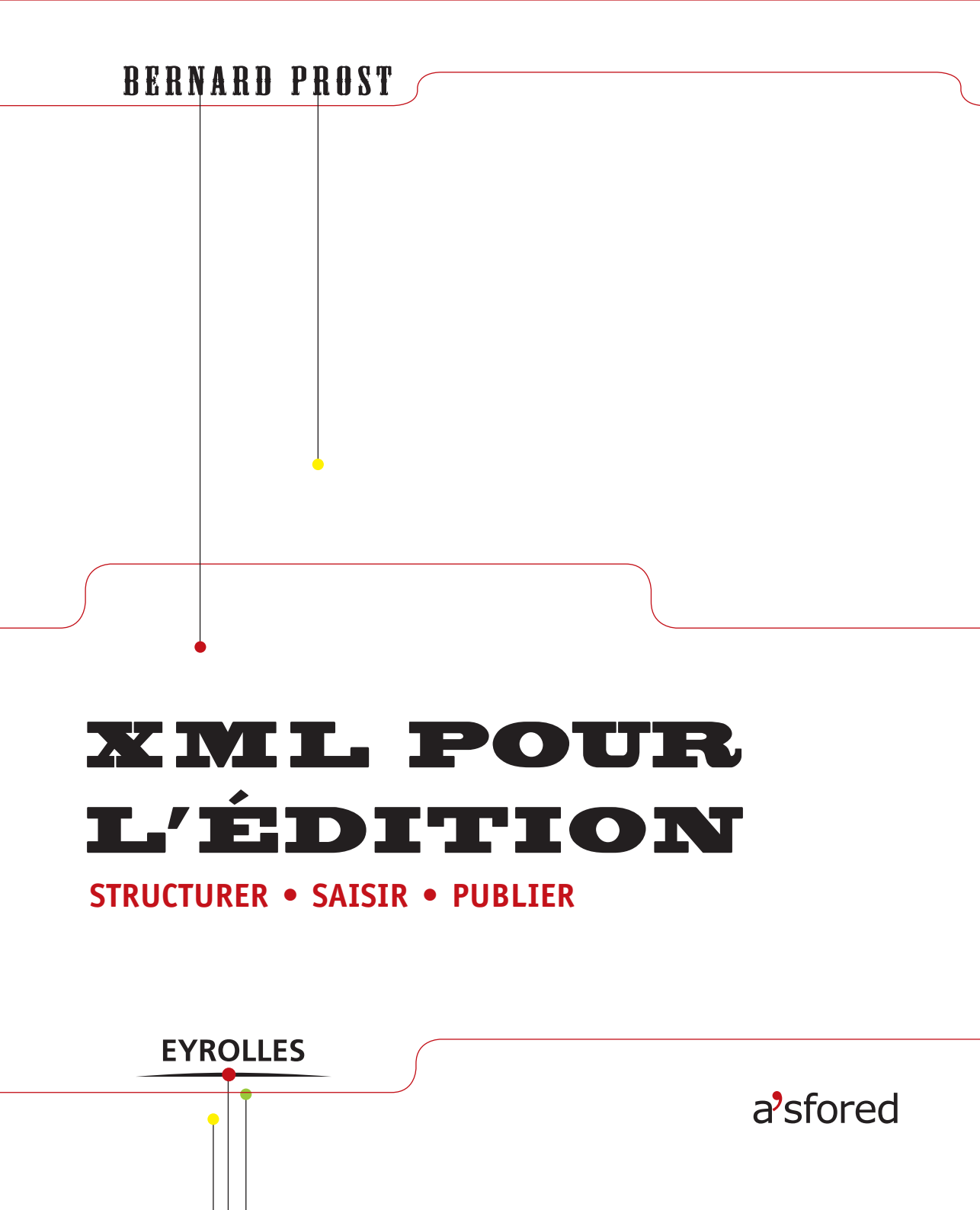
Dans la même collection

- M. Dournes, *L'image et le droit – Créer, protéger, reproduire, diffuser*, 2010, 224 pages.
É. Bacquet, *Préparer des images numériques*, 2009, 200 pages.

Chez le même éditeur

- H. Plaziat, D. Patte, *Manuel du secrétariat de rédaction* (à paraître).
A. Brillant, *XML – Cours et exercices*, 2010, 336 pages.
D. Tardiveau, *L'ActionScript 3 expliqué aux graphistes*, 2010, 224 pages.
J. Bracquemond, J.-L. Dusong, *Typographie, la lettre, le mot, la page*, 2010, 384 pages.
E. Lupton (sous la dir. de), *Books et fanzines, do it yourself!* 2009, 176 pages.
G. Lewis, *2000 accords de couleurs – Pour les graphistes et les designers*, 2009, 336 pages.
C. Breton-Schreiner, *L'essentiel de la PAO – Mise en pages, logiciels, polices, images*, 2009, 216 pages.
K. Johansson et al., *La chaîne graphique*, 2^e édition, 2009, 460 pages.
G. Davis, *500 grilles et feuilles de style pour l'imprimé et le Web*, 2008, 162 pages + CD-Rom.
E. Canivet, *InDesign, de la créa à l'exé*, 2008, 232 pages.
M. Lavant, *XPress en production*, 2007, 244 pages.
C. W. De Jong et al., *Créations typographiques*, 2007, 400 pages.
P. Evans, *PLV, Publicité, Packaging*, 2007, 268 pages.
L. Pasquet, Y. Delpuech, *Réaliser un magazine avec InDesign et Photoshop*, 2007, 164 pages.
P. Prévôt, *L'informatique de la chaîne graphique*, 2007, 116 pages.
P. Prévôt, K. Izoulet, *De la prépa de copie à la mise en pages*, 2006, 64 pages.
P. Prévôt, F. Rocher, *Techniques d'impression*, 2006, 100 pages.
J. Pieters, *Fabrication du document imprimé*, 2006, 280 pages.
K. Cheng, *Design typo, concevoir ses polices de caractères*, 2006, 236 pages.
W. Hill, *Le langage de la typographie*, 2006, 192 pages.
A. Lonjon et al., *Modélisation XML*, 2006, 498 pages.

BERNARD PROST

The cover features a minimalist design with red and black lines. A horizontal red line at the top has a rounded right end. Two vertical lines descend from it: one black line ending in a red dot, and one black line ending in a yellow dot. A horizontal red line with rounded ends is positioned below these. At the bottom, another horizontal red line has a rounded right end. From it, three vertical lines descend: one black line ending in a red dot, and two black lines ending in yellow and green dots respectively.

XML POUR L'ÉDITION

STRUCTURER • SAISIR • PUBLIER

— **EYROLLES** —

a'sfored

Éditions Eyrolles
61, boulevard Saint-Germain
75240 Paris cedex 05
www.editions-eyrolles.com

© Groupe Eyrolles 2011
ISBN : 978-2-212-12657-0

Le code de la propriété intellectuelle du 1^{er} juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée notamment dans les établissements d'enseignement, provoquant une baisse brutale des achats de livres, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie 20, rue des Grands Augustins, 75006 Paris.

Remerciements

Synthétiser les rapports entre XML et l'édition en un petit manuel est un exercice difficile qu'il est illusoire de penser mener à bien tout seul. Mes remerciements vont donc tout d'abord à Stéphanie Poisson, éditrice de cet ouvrage, à son équipe et à Véronique Dürr, qui l'a assistée dans la relecture : leur aptitude pour rendre lisible une pensée parfois tentée par l'excès de technologie est sans pareille.

Je remercie ensuite tous ceux qui travaillent ou ont travaillé avec moi autour de XML :

- les actionnaires de Ligan : Alain Pierrot, remarquable concepteur de taxonomies avancées, connaisseur de la suite Open Office, écrivain XSLT, expert de la numérisation de livres ; Xavier Maurin, pilote informatique et graphique du site MyBookForge, dont la vision numérique grand public est brillante ; Olivier Desnoux, ingénieur développeur à la méthodologie sans faille, au code élégant (et lisible !), co-concepteur du moteur de transformation au cœur de MyBookForge ; Adrien Vieilleribière, talentueux chercheur, grand artiste XSLT capable de tout mettre en ligne et rendant la transformation XML vers n'importe quel format cible à la portée de tous, également co-concepteur du moteur de transformation MyBookForge ; Patrick Pierre, ingénieur de talent et l'un des esprits les plus avancés en technologie de publication – sa maîtrise d'IDML (dont je parle très peu dans le livre) est remarquable ; Hugues Cochard, *serial-créateur* d'entreprises high-tech, pour l'heure à Tahiti mais très présent via Internet ;
- tous ceux qui m'ont fait confiance pour leurs projets professionnels ou scientifiques, en particulier Mai NGuyen et Lionel Ridoux pour qui information sur le médicament et XML n'ont plus de secret ;
- les amis de route : Christian Brugeron, pour ses scripts ingénieux destinés à contourner toutes les limitations d'à peu près n'importe quel logiciel de composition, InDesign en premier lieu, et Benoît Leprince, pour avoir bien voulu me livrer différents exemples de maquettes InDesign destinés à illustrer cet ouvrage.

Merci à toutes celles et ceux du tout jeune écosystème e-book, dont la croissance devrait suivre une pente vertigineuse, dans le monde et peut-être en France, en particulier Houriah Ghebalou (PREMICE, l'incubateur régional de Bourgogne), qui a apporté le financement des études préindustrielles du projet Ligarane/Mybookforge, la région Bourgogne, qui contribue au projet et à son implantation locale, et Nicéphore Cité, qui nous abrite à Chalon-sur-Saône et assure une relation permanente avec les start-up de l'image et du son.

Merci enfin à Ray Charles, qui a compris que le support influençait le contenu : sans la nécessité de retourner un 45 tours pour écouter la suite d'un long morceau de musique, le fameux *break* de *What I'd Say* n'existerait pas !



SOMMAIRE

Avant-propos	XII
--------------------	-----

Chapitre 1

Séparer la forme du fond

Modéliser le document	4
Identifier les 3 aspects d'un document	4
Identifier les classes de documents	6
Identifier les structures à repérer en fonction des médias cibles et des usages pressentis	9
Organiser les éléments structurels	12
Utiliser les balises pour marquer la structure	14
Balises	14
Éléments, relations et arbres	16
Attributs	17
Visualiser le document XML	19
Visualiser le XML dans un éditeur de texte	19
Visualiser le XML dans un navigateur	19

Chapitre 2

Les principaux composants structurels

Structure hiérarchique	24
< front >	26
< body >	28
< back >	29
Les structures blocs	30
Le paragraphe	30
La liste	31
Le tableau	32

Repérage de blocs particuliers.....	33
Conteneurs et contenus mixtes.....	34
Les structures inline.....	35
Enrichissement.....	35
Images et objets multimédias.....	36
Formules mathématiques, code.....	37
Notes.....	38
Liens hypertextes.....	38
Repérage d'éléments inline particuliers.....	39
Contenus mixtes.....	40
Structures navigationnelles.....	41
Table des matières.....	41
Index.....	42
Titres courants.....	42
Liens hypertextes.....	43
Les entités.....	44
Les entités caractères.....	44
Partage de composants XML.....	46
Métadonnées.....	47
Structure des métadonnées.....	47
Dublin Core.....	48
ONIX.....	49
Définir des métadonnées supplémentaires.....	49

Chapitre 3

Écrire/concevoir une DTD..... 51

Les règles de structures.....	52
Qu'est ce qu'une DTD?.....	52
Contrôler la DTD.....	53
Définir les balises et leur comportement.....	55
Décrire les attributs.....	60
Écrire et organiser la DTD.....	64
Tenir compte (ou non) des supports cibles.....	71
Le support papier.....	71

Le support électronique	72
Quelle DTD utiliser?	75
DTD généralistes publiques	75
DTD métiers.	76
DTD maison et DTD fournisseurs	77

Chapitre 4

Saisir le XML	79
Utiliser un éditeur XML	80
Qui doit utiliser un éditeur XML?	80
Comprendre la relation entre DTD, document XML et visualisation	81
Choisir et paramétrer son éditeur XML	82
Utiliser une DTD	90
Visualiser une DTD.	90
Documenter une DTD.	92
Utiliser un document modèle vide	93
Référencer la DTD dans le document XML	95
Saisir le texte et la structure	96
Utiliser l'éditeur XML	96

Chapitre 5

Préparer et diriger la saisie XML	103
Styler le document Word	104
Qu'est-ce qu'un style dans Word?	104
Objectifs du stylage.	105
Les limites du stylage à vocation XML	106
Déclarer un style dans Word	106
Nommer un style	107
Lister les styles de Word	108
Gérer les feuilles de styles.	109
Utiliser les styles pour produire du XML.	110
Identification des zones à styler	110

Nommage, choix typographiques et pose des styles.....	111
La liaison style Word/XML	113
Écrire les consignes de saisie	114
Que contiennent les consignes de saisie?	114
Qui doit écrire les consignes?.....	115
Comment rédiger les consignes?.....	115
Organiser la saisie XML.....	117
Désigner un responsable de DTD.....	117
Gérer les exceptions	117

Chapitre 6

Épreuver le XML..... 121

À quelle étape faire intervenir le XML?	122
Production XML en amont	122
Production XML en cogénération	124
Production XML en aval	126
Épreuver le XML?	127
Objectifs et principes de l'épreuve XML	127
Contrôler les caractères	129
Vérifier le contenu hiérarchique	130
Vérifier les images et les tableaux.....	131
Vérifier le balisage spécifique	131
Vérifier les liens.....	132
Vérifier une livraison de mise à jour	133
Les outils de comparaison des éditeurs XML	133
Les outils bureautiques.....	135

Chapitre 7

Transformer le XML avec XSLT . 137

La publication multisupport.....	138
Les transformations multisupports	138
Les cibles de publication	141

Le langage XSLT	146
Principe d'une transformation XSLT	146
Spécifier une transformation	147
Utiliser un processeur XSLT	149
Écrire un programme XSLT	150
X-Path	152

Chapitre 8

Publier pour l'électronique

Publier pour le Web	156
Quelle unité d'affichage?	156
Transformer les données pour le Web	160
Publier au format epub	170
Le format epub	170
Créer un epub	171
Convertir l'epub en mobipocket (Kindle)	179
Les supports cibles	181
Lire un epub : les logiciels de lecture	181
Les tablettes	185
Les téléphones	187

Chapitre 9

Publier pour le papier

XSL-FO : un langage XML ouvert de description de page	192
Comment produire un fichier XSL-FO?	193
Représentation de la mise en pages à l'aide d'XSL-FO	193
Mettre en œuvre une composition XSL-FO	198
De XSL-FO à PDF : les moteurs de rendu	202
Utiliser InDesign avec XML	206
La correspondance élément/styles	206
La correspondance attributs InDesign/styles	211
En guise de conclusion	215

Avant-propos

Wait a minute, wait a minute, oh hold it! Hold it! Hold it!

Hey (hey) ho (ho) hey (hey) ho (ho) hey (hey) ho (ho) hey
Ray Charles (*What I'd Say*)

If everything is under control, you are going too slow
Mario Andrett

L'édition vit une mutation dont la portée est considérable. Le support papier voit arriver la concurrence de supports dématérialisés chaque jour plus nombreux : le Web bien sûr, l'ordinateur ultra compact, appelé *netbook* – sans qu'il s'agisse d'un hasard –, mais surtout le téléphone et les outils nomades spécialisés comme le *reader* (liseuse) d'e-books ou la tablette qui mettent à la portée de chacun, *urbi et orbi*, des bibliothèques complètes d'œuvres littéraires ou professionnelles. Pour l'éditeur, il s'agit désormais de publier pour ces supports en tenant compte de leurs spécificités, tout en minimisant les coûts et les délais de production. Aux expériences nécessitant des reprises multiples du même fonds documentaire a succédé une approche plus industrielle – mais aussi plus normative ou contraignante – basée sur XML.

DÉFINITION **Liseuse**

Liseuse est le terme français proposé pour *reader* qui, chez les Anglo-Saxons, désigne le lecteur nomade de livres électroniques (e-book). La liseuse désigne un matériel dédié à la lecture utilisant la technologie d'affichage dite *e-paper*, terme marketing désignant un écran non rétro-éclairé, donc peu consommateur en énergie et réputé moins fatigant à la lecture. Associé à l'*e-paper*, le marketing a introduit la notion d'*e-ink* pour qualifier un pixel...

La souplesse et l'universalité de XML ont séduit aussi bien les éditeurs – en premier lieu les éditeurs juridiques, habitués il est vrai à SGML – que les informaticiens qui ont pu ainsi échanger des données entre ordinateurs aux fonctionnements hétérogènes.

Normalisé en 1999, XML a atteint la maturité : il existe désormais un écosystème XML où l'on rencontre des logiciels spécialisés (les éditeurs XML), des prestataires *on-shore*, *near-shore* et *off-shore* rompus à ce

langage, des développeurs d'applications sachant utiliser le DOM (*Document Object Model*) pour créer des produits électroniques toujours plus innovants, et des modèles documentaires spécialisés verticalement par secteur ou horizontalement par type de publication.

Pour autant, la pratique n'est pas encore stabilisée et les habitudes diffèrent considérablement d'un éditeur à l'autre. Le propos de cet ouvrage est d'apporter une vision pratique de l'utilisation d'XML dans les maisons d'édition, fondée sur des usages concrets ayant fait leurs preuves mais par nature limités à des cas particuliers. *XML pour l'édition* n'est ni une bible, ni un essai dogmatique sur le sujet, et chacun pourra adapter sa pratique aux exemples fournis.

DÉFINITION DOM

DOM (*Document Object Model*) est une modélisation informatique sous forme d'arbre de documents XML ou HTML. DOM est indépendant de toute taxonomie (voir définition plus loin). Cette modélisation permet une manipulation par programme des constituants (éléments) du document.

Structure de l'ouvrage

L'ouvrage est organisé en trois parties – structurer, saisir, publier – qui recouvrent l'ensemble du cycle XML pour une publication multisupport de type « livre ». *XML pour l'édition* s'adresse aux éditeurs, aux lecteurs/correcteurs, aux fabricants, en premier lieu, mais également aux managers qui souhaitent mieux comprendre les techniques sous-jacentes et l'influence du support sur la conception et la forme de l'objet de publication numérique. Enfin, les auteurs curieux des possibilités qu'apporte cette technique y trouveront matière à concevoir différemment leur œuvre.

L'ouvrage s'appuie sur un exemple d'article encyclopédique, de type Wikipedia, qui sera repris au fil des chapitres. Cet exemple fait appel à une structure spécifiquement développée pour le présent ouvrage (`article_v1.2.dtd`). Les balises ont été choisies en langue anglaise afin de se conformer aux usages internationaux. L'exemple répond à un cahier des charges éditorial simple :

- pouvoir publier l'article sur le papier, le Web, le téléphone ;
- en dotant les objets de publication électronique d'interactivité au niveau des auteurs, des bibliographies, des filmographies et des discographies, cette interactivité doit être indépendante des bases de données cibles.

Pour des raisons de simplicité, il n'est pas prévu de tableaux ni de formules mathématiques (en dehors de l'utilisation éventuelle de celles-ci sous forme d'images).

Structurer le XML

DÉFINITION **Taxonomie**

La taxonomie désigne le jeu de balises utilisé pour l'encodage XML d'un document. La taxonomie est généralement décrite à l'aide d'un langage spécialisé (DTD, XML Schema, Relax NG).

Le premier chapitre s'intéresse plus particulièrement à la modélisation des documents et à la démarche du balisage XML. Le second chapitre est, lui, consacré à la description des principales structures que l'on peut rencontrer dans un ouvrage,

ou plus généralement dans un document. Le troisième chapitre donne enfin les clés pour écrire une DTD, c'est-à-dire une des représentations (la plus simple en fait) d'une taxonomie.

Saisir le XML

Le quatrième chapitre concerne la saisie proprement dite. Dans la plupart des cas, ce travail est confié à un prestataire externe, mais l'éditeur doit de plus en plus être capable de modifier un document à l'aide d'un outil de saisie XML utilisé en interne pour rattraper de petites erreurs ou effectuer des corrections de dernière minute. Ce chapitre aborde en particulier la configuration d'un éditeur XML du marché et son utilisation dans le cadre d'une DTD particulière.

Le cinquième chapitre s'intéresse à la relation avec les sous-traitants : comment préparer la copie pour minimiser les risques d'erreurs d'interprétation au niveau de la structure par le prestataire et bâtir des consignes de saisie efficaces ?

Le sixième chapitre est ensuite consacré à une étape rarement décrite dans les processus de production : l'épreuve XML. Comment s'assurer que le XML fourni par le prestataire est conforme aux attentes de l'éditeur ? Ce chapitre aborde également les différents modèles de production XML selon que la saisie XML s'effectue en amont de la mise en pages papier, pendant celle-ci ou a posteriori.

Publier

Le septième chapitre expose, dans leurs grandes lignes, les techniques de transformation d'un document XML en un format cible qui peut être aussi bien du XML (par exemple en entrée d'InDesign), du XHTML ou tout autre format textuel. Bien que très technique, le langage de transformation XSLT n'a rien de mystérieux et il importe que les acteurs de l'édition en comprennent le fonctionnement pour apprécier l'impact d'une décision éditoriale.

Le huitième chapitre décrit brièvement la publication pour les supports électroniques en se limitant au Web, aux tablettes, aux liseuses et à l'iPhone choisi comme représentant le plus abouti (actuellement) de la lecture sur téléphone.

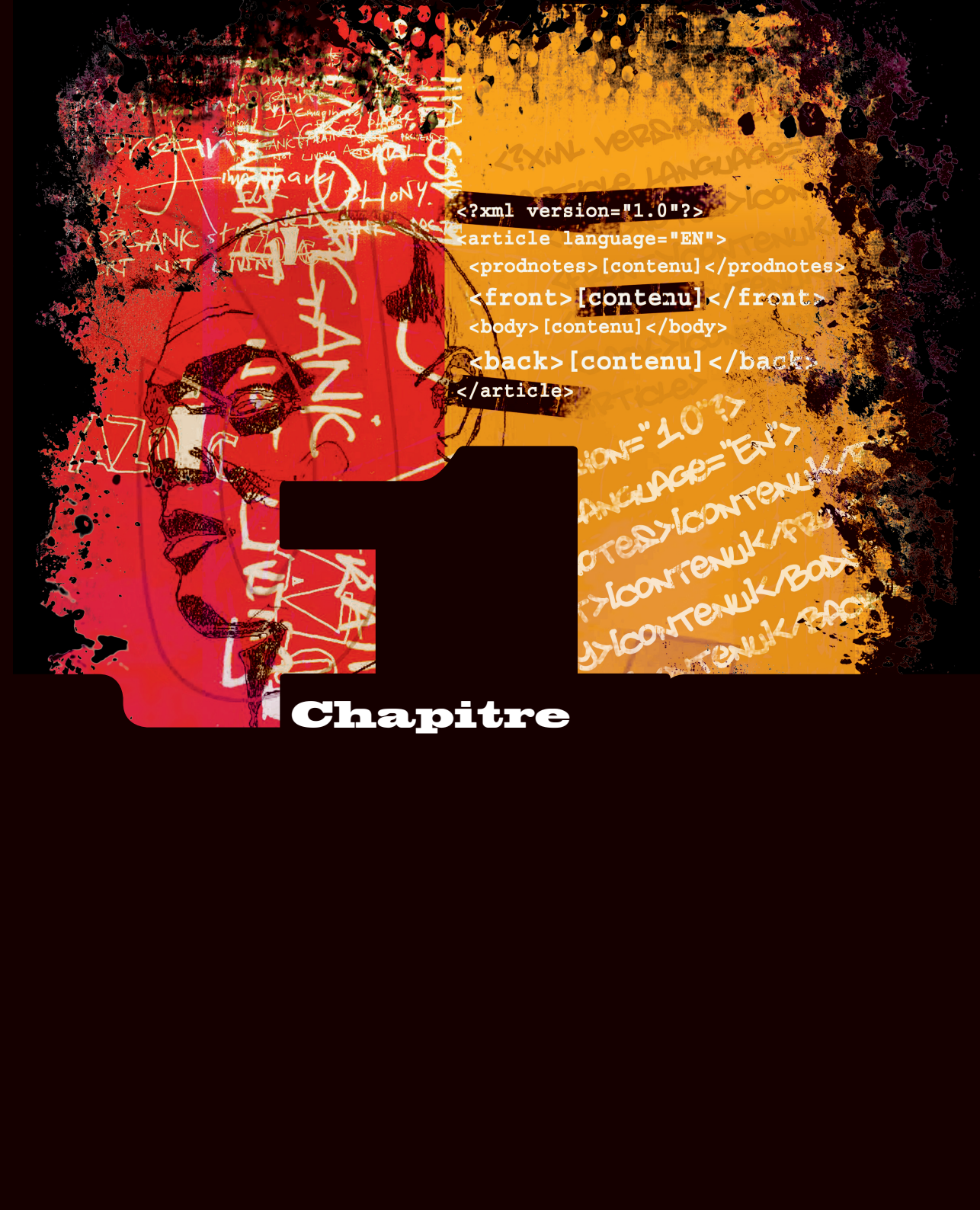
Le neuvième chapitre aborde enfin la publication sur papier à partir d'un document XML, selon deux approches :

- la transformation directe du XML en PDF en utilisant XSL-FO, langage de mise en pages lui-même exprimé en... XML ;
- l'importation du XML, éventuellement remanié, par un outil de PAO (ex. : InDesign).

Cet ouvrage vise à donner les clés d'utilisation de XML dans la chaîne éditoriale et se limite à l'essentiel de cette approche moderne de l'édition ; pour chacune des techniques abordées, il existe, aussi bien en français qu'en anglais, des ouvrages très complets auxquels le lecteur se référera avec profit.

À NOTER

Le vocabulaire du domaine XML est assez opaque. De nombreux termes sont chargés de références à SGML, aux feuilles de styles, etc., qui ont perdu leur sens d'origine et ne sont plus du tout représentatifs de leur fonction. Il faudra donc faire l'effort de les mémoriser indépendamment de leur sens usuel en français (ou en anglais) courant.



```
<?xml version="1.0"?>
<article language="EN">
<prodnotes>[contenu]</prodnotes>
<front>[contenu]</front>
<body>[contenu]</body>
<back>[contenu]</back>
</article>
```

Chapitre



Séparer la forme du fond



La question cruciale pour l'éditeur est de bâtir une méthodologie lui permettant, au prix d'une seule saisie initiale ou d'une saisie a posteriori la moins coûteuse possible, de publier pour de nombreux supports connus ou à venir. La première étape consiste à séparer autant que possible la forme du fond, bien au-delà des techniques mises en œuvre par les feuilles de styles des traitements de texte.



Modéliser le document

Un livre, ou plus généralement un document sous forme XML, nécessite de disposer d'une modélisation suffisamment générale pour s'adapter à tous les cas de figure probables. On modélisera abstraitement un ensemble de documents – une «classe de documents» – pour les soumettre à des traitements informatiques communs.

Identifier les 3 aspects d'un document

Après vous être familiarisé avec XML, vous ne verrez plus jamais un document comme par le passé. Au-delà du fond constitué par la juxtaposition des mots sans enrichissement typographique, et de la forme qui met en valeur partiellement la pensée de l'auteur, la (ou plutôt une) structure est une nouvelle composante documentaire apportant des fonctionnalités qui dépendent, en particulier, des usages papier et électronique envisagés.

Le fond

Le fond est le texte, c'est-à-dire ce qui se lit en dehors de toute considération de mise en forme. Sa version la plus libérée de la forme est l'enregistrement audio : le mot n'a que sa valeur sémantique et n'est pas souligné par une variation typographique, même si quelques variantes sonores peuvent lui donner un peu plus d'épaisseur.

La forme

La forme enrichit l'information. Elle s'appuie sur une traduction graphique fortement culturelle et linguistique qui rend implicite son décodage.

Dans nos sociétés, graisser un caractère a une signification de mise en valeur aussi bien au niveau d'un titre qu'au sein du texte courant. Le corps du caractère porte le sens du niveau d'importance, généralement couplé à un décalage

visuel vers la gauche : « plus gros et plus à gauche » exprime le plus souvent un titre de niveau hiérarchie plus important.

La structure

Mettre au singulier le mot « structure » est impropre : il n'existe pas une structure, mais une infinité de structures dépendant de ce que l'on souhaite identifier en vue d'un usage ultérieur.

- Dans le cas d'un roman que l'on veut éditer sur papier et en version électronique, on se limitera à repérer les chapitres, les titres de chapitres, les paragraphes, et à l'intérieur de ceux-ci les mises en valeur au sein du texte.
- Dans le cas d'un article de revue que l'on veut publier sur le Web avec des fonctionnalités de recherche automatique sur Google Scholar ou Google Books (ou n'importe quelle base bibliographique), on repérera les entrées bibliographiques, les noms et éventuellement prénoms des auteurs ainsi que les titres d'ouvrages ou de revues cités.

● Le fond (a) est constitué par le texte brut, celui que l'on peut restituer par la lecture à voix haute (ouvrage audio).

La forme (b) est un apport d'information supplémentaire, largement influencé par la culture et la pratique. Un titre se reconnaît à ce qu'il est écrit en plus gros et en gras. Il exerce une fonction de « repérage » (il s'agit d'un titre) et une fonction « résumé » permettant au lecteur de décider s'il va aller plus loin dans la découverte du texte.

La structure – identifiée ici par des bulles – est une représentation abstraite (souvent guidée par la forme lorsque celle-ci préexiste) destinée à un usage multisupport, sans choix a priori sur le rendu final.

Miles Davis
Miles Dewey Davis III (26 mai 1926, Alton, Illinois - 28 septembre 1991, Santa Monica, Californie) est un compositeur et trompettiste de jazz américain.
Une figure centrale du jazz Miles Davis fut à la pointe de beaucoup d'évolutions dans le jazz et s'est particulièrement distingué par sa capacité à découvrir et à s'entourer de nouveaux talents. Son jeu se caractérise par une grande sensibilité musicale et par une fragilité qu'il arrive à donner au son. Il a marqué l'histoire du jazz et de la musique du XXe siècle. Beaucoup de grands noms du jazz des années 1940 à 1980 ont travaillé avec lui.

(a)

Titre
(de l'article)



Illustration
(photo)

Mise en valeur

Titre
(de section)

Paragraphe
(de section)

Chapô

(b)

Identifier les classes de documents

Le modèle documentaire générique capable de représenter n'importe quel type de document n'existe pas. S'il existait, il serait d'une complexité telle que son usage deviendrait prohibitif. On s'attache dès lors à définir des « classes » de documents qui correspondent à des types d'organisation de l'information – par exemple un dictionnaire – ou aux regroupements naturels que sont les collections au sein des maisons d'édition.

Le processus de définition des classes de documents, appelé « Analyse documentaire », consiste à extraire d'un ensemble de documents préalablement identifiés comme étant similaires dans leur organisation, les composants structurels en vue d'un usage ultérieur. On part pour cela d'un nombre limité d'exemples d'ouvrages disponibles jugés représentatifs et l'on bâtit progressivement un modèle répondant aux besoins éditoriaux multisupports.

Ouvrages hiérarchiques

L'ouvrage hiérarchique le plus élémentaire est le roman ou l'essai : c'est le modèle à la fois le plus simple, le plus répandu, le plus intuitif, mais en même temps complexe car il existe une infinité de variantes structurelles qu'il faudra gérer (quitte à ne pas les prendre en compte ou à les simplifier pour les versions électroniques).

DÉFINITION **Label**

Le terme anglo-saxon Label désigne un indicateur (graphique, textuel ou numérique) de navigation : numérotation d'une liste, indication du numéro de chapitre...

Un essai est souvent (mais pas toujours) organisé en parties, elles-mêmes organisées en chapitres. Le chapitre est doté d'un titre optionnel, lui-même précédé (toujours optionnellement) d'un numéro ou d'un label permettant de le replacer dans l'organisation du livre. Lorsqu'il n'y a ni numéro, ni titre de chapitre, se pose dans la version électronique le problème du repérage de ces mêmes chapitres. Bien sûr, des solutions existent...

Le constituant structurel le plus répandu au sein du chapitre est le paragraphe : unité sémantique voulue par l'auteur et identifiable typographiquement par un retrait de première ligne – ce qui permet de le repérer lorsqu'il apparaît fortuitement en début de page – et un retour en fin de paragraphe. Au sein des paragraphes, l'auteur pourra procéder à des mises en valeur sémantiques qu'il exprimera, entre autres moyens, par du gras ou de l'italique.

Enfin, des variantes typographiques portant sur le paragraphe, fer à droite par exemple, permettent d'exprimer des notions différentes, comme une citation, un exergue, des épigraphes, etc., sans qu'il y ait de limite dans le nombre de variantes envisageables.

Dictionnaires

Chaque dictionnaire est une structure en soi et il n'est pas réaliste de parler de LA classe « dictionnaire » ; on trouvera plutôt des structures propres à chaque dictionnaire, l'objectif étant souvent de disposer de versions papier réduites (par exemple un dictionnaire de poche) ou de versions électroniques offrant des fonctionnalités avancées (liens hypertextes en particulier).

Un dictionnaire est plus proche d'une base de données que d'un document de type livre : il est doté d'entrées, le plus souvent classées par ordre alphabétique, organisées en unités sémantiques ordonnées à la manière des champs d'une base de données.

En pratique, on utilise des entrées modélisées en XML, sorte de micro-documents, que l'on intègre au sein d'une base de données. Cette dernière offre des fonctionnalités puissantes de gestion des entrées (par exemple l'assignation de telle ou telle entrée à tel auteur, le verrouillage d'une entrée lorsqu'une correction est en cours, la gestion des versions, etc.).

Dans le cas d'un dictionnaire comme le Littré, par exemple, on organisera l'information de la manière suivante :

Entrée

en-tête

terme

prononciation

nature grammaticale

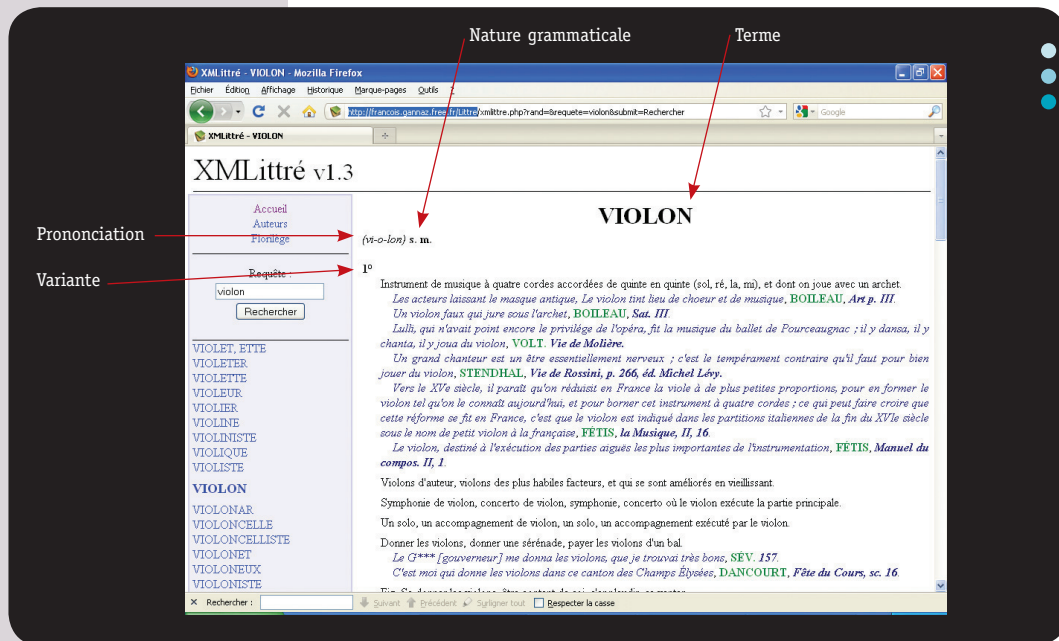
variante

numéro de variante

citation,

etc.

Au sein de chaque entrée, l'information sera organisée en autant de blocs qu'il y a de variantes de sens.



● Organisation générale de l'entrée du Littre V 1.3 (source : francois.gannaz.free.fr)

Revue et articles

Les articles de revue ont une structure assez facilement partageable et constituent une classe en soi. Les éditeurs de revues de sciences humaines et sociales ne s'y sont pas trompés et s'appuient sur des modèles suffisamment généraux pour être utilisés par pratiquement n'importe quelle revue, quitte à procéder à quelques ajustements structurels.

Un très bon exemple de cette approche est le site français lodel.org qui offre ainsi la possibilité à ses utilisateurs de produire simplement leur contenu, conformément à une classe de documents dont la taxonomie est publiée en licence.

Rappelons par ailleurs que le présent ouvrage s'appuie sur un exemple d'article encyclopédique simple (i. e. sans tableaux ni formules mathématiques) qui, au prix d'ajustements mineurs, pourrait être utilisé en production.

Autres classes de documents

Dans tous les cas où les modèles existants ne conviennent pas, il convient de procéder à une analyse documentaire spécifique. C'est ce qu'ont fait les édi-

teurs juridiques, qui ont mis au point des modèles propres à leurs ouvrages, ou encore les éditeurs de méthodes de langues, qui veulent à la fois publier sur différents supports et faciliter la localisation (i.e. la traduction) de leurs ouvrages en conservant tout ce qui concerne la langue apprise.

On s'intéressera également aux classes de documents ayant un intérêt transversal. Un éditeur scolaire, par exemple, voudra bâtir une banque d'exercices interactifs à partir de ses nombreux manuels destinés à l'éducation : dans ce cas, l'éditeur définira une classe «exercices» commune prenant en compte les types d'exercices envisagés en orientant la structuration vers une utilisation interactive.

Quelques éléments d'une classe «exercices» pour un éditeur scolaire

QCM	Questionnaire à choix multiples (plusieurs réponses correctes). L'utilisateur doit désigner des cases à cocher, avec plusieurs choix possibles.
QCU	Questionnaire à choix unique (une seule réponse correcte). L'utilisateur doit désigner un seul bouton radio.
À trous – saisie	Texte vide à l'affichage. L'utilisateur doit saisir le texte correct.
À trous – liste déroulante	Texte vide à l'affichage. L'utilisateur doit choisir le texte correct dans une liste. Possibilité de fausses réponses dans la liste.
Appariement – glisser/déposer	Deux listes organisées en items graphiques. L'utilisateur doit relier par glisser/déposer les items des deux listes à faire correspondre.
Appariement – labels	Deux listes organisées en items textuels, chaque item étant repéré par une lettre ou un numéro. La relation entre les items des deux listes s'effectue par saisie de la lettre ou du numéro en correspondance.

Identifier les structures à repérer en fonction des médias cibles et des usages pressentis

Chaque support a ses spécificités dont on doit tenir compte dans la démarche XML. Il est évident qu'un écran de téléphone s'accommode mal de titres longs (i.e. s'étalant sur quatre ou cinq lignes sur l'écran du téléphone, tandis qu'ils occupent une ou deux lignes sur la version papier) : il faudra soit diminuer la longueur du titre afin qu'il soit multisupport d'entrée de jeu, soit prévoir un titre adapté pour le média cible. Dans le pire des cas, on laissera le titre complet si l'on se limite aux supports capables de tronquer automatiquement les titres à l'affichage comme l'iPhone par exemple.

Le papier

Graphiquement, le papier est le média le plus riche qui soit ; il est également sous le contrôle total de l'éditeur. Celui-ci décide de tout : ligne graphique, choix des polices, interlignage, césure, taille de la page, nature du papier, encrage, etc.

On peut bien sûr utiliser XML pour produire l'information à destination du papier. Mais il se pose une difficulté : comment rendre éventuellement dans le XML toute la richesse typographique exprimée par l'impression, sachant que la démarche XML consiste à repérer plutôt les aspects sémantiques du texte que ses aspects typographiques dont le choix reste largement subjectif ?

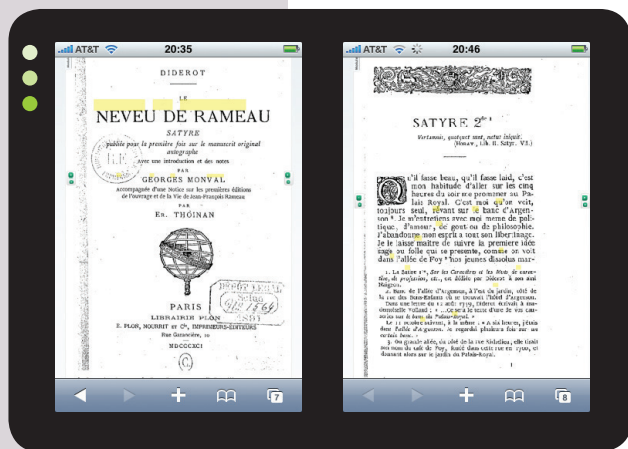
La première idée qui vient à l'esprit est de « forcer » la typographie dans le XML : c'est en effet possible et les représentations XML proposées par les éditeurs de traitement de texte ou de PAO reflètent cet aspect. La contrepartie

est une difficulté majeure ou même une impossibilité de produire des versions électroniques bien adaptées aux médias cibles. Si l'on tient à obtenir sur un support électronique un résultat aussi esthétique que sur papier, le plus aisé consiste à publier le PDF imprimeur ayant servi pour le papier avec le risque qu'il soit très difficile à lire compte tenu de la taille des caractères, ou qu'il soit non ergonomique, le défilement horizontal et vertical d'un PDF A4 sur ce même écran de téléphone constituant une expérience de lecture particulièrement éprouvante.

L'utilisation de XML pour produire du papier va dépendre de nombreux paramètres comme la complexité structurelle de l'ouvrage (mais cela se contourne), le rendu graphique (ainsi le présent ouvrage dont la structure, quoique très simple, cadre mal avec la richesse graphique de la maquette) et surtout le gain de productivité pour l'opérateur d'exécution PAO. Pour des ouvrages scolaires, très graphiques, le plus souvent contraints par le calendrier, la structuration XML préalable à la mise en pages n'est pas une option réaliste.

En matière de publication papier, un des choix stratégiques de l'éditeur est l'étape à laquelle il va introduire XML dans son cycle de production : en amont, préalablement à toute mise en pages ? Pendant la mise en pages sous forme d'une cogénération ? Ou entièrement après la mise en pages, comme si le document existait déjà dans sa forme papier ? Sur ce point, voir le chapitre 9.

● Pages PDF sur un écran de téléphone : on peut bien sûr faire défiler le texte après grossissement, ce qui revient à utiliser une loupe pour lire un document : peu ergonomique... (Le Neveu de Rameau, Diderot, ouvrage Gallica, BNF)



Quoi qu'il en soit, le choix de la structuration XML en vue d'une sortie papier va dépendre du degré d'automatisation recherché : pour un dictionnaire, l'automatisation est complète et l'inévitable travail de retouche humaine, ne serait-ce que pour optimiser la pagination, reste très réduit. Pour un roman ou un essai, le gain de productivité sera de 50 à 80 % en environnement InDesign suivant la complexité des notes et des éventuels index. Pour un ouvrage scolaire, le gain de productivité est négatif (tout au moins avec le niveau de complexité des maquettes actuelles) : utiliser XML pour produire la mise en pages d'un tel ouvrage est contre-productif. Ceci ne veut pas dire se passer de XML, mais dans ce cas, on produira les flux XML postérieurement à la publication papier, en simplifiant la structure et en l'optimisant en vue de la version électronique.

À NOTER

Il est d'usage de parler de « flux XML » pour parler du document XML. Ce vocabulaire vient de l'Internet où il est possible de commencer à interpréter la page HTML avant que la totalité de la page n'ait été transférée au navigateur. Dans la suite de l'ouvrage, on parlera indifféremment de flux XML ou de document XML.

L'édition électronique

L'édition électronique est contrainte en premier lieu par la taille de l'écran et par la connectivité synchrone Internet en second lieu. Elle est par ailleurs dépendante du caractère repaginable (*flowable* en anglais) ou non donné au texte lors de la publication : un texte est repaginable lorsqu'il s'adapte à l'écran d'affichage et se recompose au vol. La repagination est propre au support concerné (valeurs par défaut), mais elle dépend également du choix de l'utilisateur qui désire par exemple lire le texte en caractères plus gros ou dans une police particulière.

Si l'écran cible est de grandes dimensions et constitue l'unique support envisagé, l'éditeur peut se contenter de publier le PDF ayant servi à la mise en pages papier : il n'y a pas à proprement parler d'édition électronique, mais simplement une mise à disposition sous un format différent du document source papier.

Si l'écran est petit (téléphone ou Nintendo DS) ou si la cible est constituée de nombreux écrans aux tailles et caractéristiques différentes, il sera nécessaire d'en tenir compte : les titres, en particulier, risquent d'être trop longs pour un affichage sur téléphone. Il faudra en créer de plus courts (au prix d'un travail éditorial complémentaire) et prévoir leur emplacement dans la structure XML.

Il peut également être souhaitable d'introduire des informations de navigation qui auront une utilité uniquement en environnement électronique : la notion de titre courant doit ainsi être revue pour pouvoir s'afficher dans des mini-navigations en tête de chapitre ou de section.

DÉFINITION **Mini-navigation**

Système de navigation dans un document affiché sur un écran de très petites dimensions du type `toc>partie>chapitre>section`, chaque niveau étant accessible en hypertexte.

Par ailleurs, le texte lui-même doit prendre en compte les caractéristiques d'une publication multisupport : les renvois à des pages de la version papier sont assez déroutants en format électronique, sauf dans le cas d'un PDF produit à l'identique du papier et dans l'hypothèse où la pagination réelle, celle du papier, correspond exactement au nombre de pages du PDF, ce qui est, en général, loin d'être le cas...

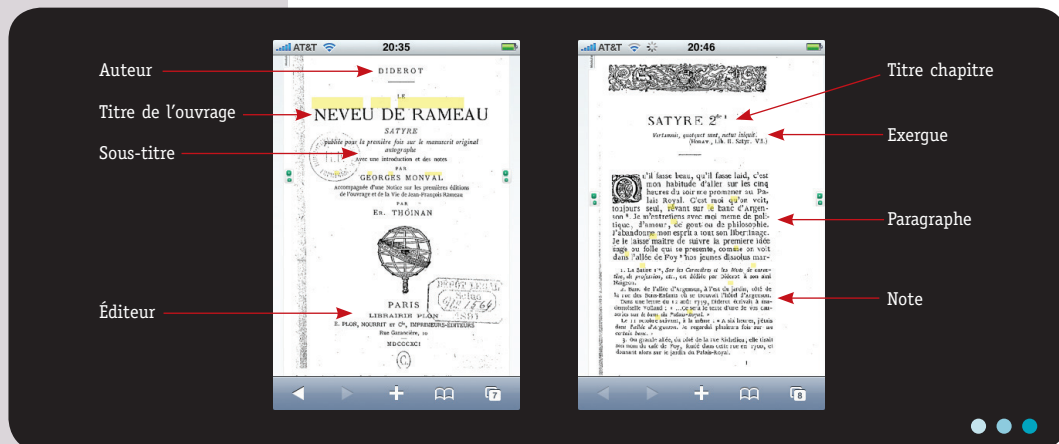
Organiser les éléments structurels

Nommer les éléments structurels

XML permet de définir son propre vocabulaire : il est donc possible de choisir `tit` ou `titre` ou `title` ou encore `Title` pour repérer un titre.

Lors de l'analyse documentaire, on se contentera de désigner en français courant les éléments structurels dans le vocabulaire de l'entreprise : on parlera par exemple de « titre de chapitre », « titre de section de niveau 1 », « titre

● L'identification des éléments structurels d'un roman, en utilisant des mots du vocabulaire courant.



de section de niveau 2», etc., indépendamment du choix final, sans que cela, d'ailleurs, ne préjuge du nom effectif de la balise.

Hierarchiser les éléments structuels

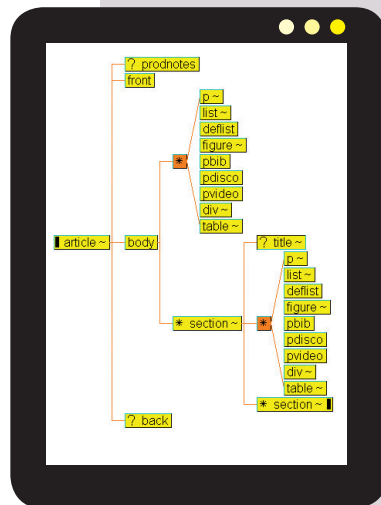
Parallèlement, les éléments structuels doivent être hiérarchisés à la manière d'une table des matières, en présentant à cette fin les composants de structure sous forme arborescente, l'objectif étant de constituer une représentation acceptable de la classe de documents étudiée.

Définir le comportement des éléments structuels

XML permet de définir des comportements élémentaires pour chaque élément structuel. Les éléments peuvent être « obligatoires » ou « facultatifs » : un titre d'article de presse sera généralement défini comme obligatoire (il doit nécessairement être présent pour que le document soit valide), tandis que le chapô sera facultatif. Les éléments peuvent être également « uniques » ou « répétables » : le titre d'un article sera ainsi unique, tandis qu'un paragraphe sera répétable.

Au niveau de l'analyse documentaire, on utilisera soit le français courant en combinant les adjectifs « unique », « répétable », « obligatoire », « facultatif », soit un éventuel signe distinctif (? , * , +), habituel dans l'écriture des DTD (voir le chapitre 3), que l'on accolera après le nom de l'élément décrit.

La représentation ainsi obtenue permet de se faire une bonne idée du modèle documentaire. Celui-ci sera affiné lors de l'écriture de la DTD (ou des autres systèmes de représentation de taxonomies) qui permettra de poser des contraintes très précises sur les éléments structuels.



● Représentation hiérarchique partielle d'une classe de documents (article encyclopédique)

À NOTER

Ces aspects de la modélisation peuvent se combiner : un élément pourra être défini comme unique et obligatoire (ex. : le titre d'un article), tandis qu'un autre pourra être défini comme répétable et facultatif (ex. : le paragraphe).

Utiliser les balises pour marquer la structure

Un jeu de balises – une taxonomie, donc – définies à l’aide d’un langage spécialisé (DTD, XSD ou Relax NG) formalise techniquement le modèle élaboré lors de l’analyse documentaire. En conjonction avec la DTD, les balises sont « posées » dans le texte à l’aide d’un éditeur XML afin de délimiter chaque élément structurel.

Le source HTML

Le source (sous-entendu « le fichier source », d’où le masculin) est le document HTML, mais également XML, XHTML, etc. que le navigateur ouvre de manière dynamique. L’interprétation des balises HTML conduit à une « mise en pages » graphique, conformément aux souhaits du concepteur web qui aura donc posé ces balises dans le document, soit à la main, soit par programme.

Balises

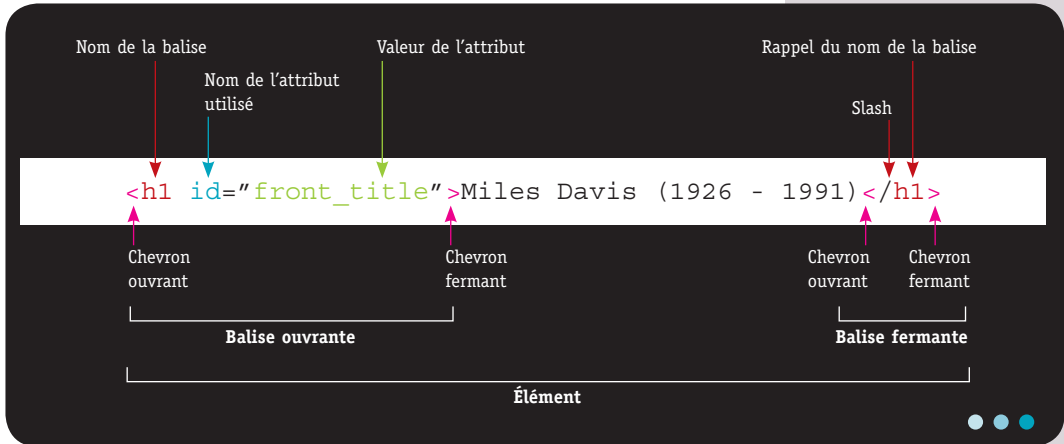
L’exemple de HTML

HTML est un exemple de taxonomie bien connue et facilement accessible : pour visualiser les balises d’une page HTML, il suffit de demander l’affichage du source depuis un navigateur (Affichage > Code source de la page sur Firefox).

La morphologie d’une balise au sein d’un document XML est simple :

- un chevron ouvrant (<);
- un nom sans espace et commençant par une lettre qui constitue le nom de la balise;
- éventuellement une succession d’informations complémentaires placées à l’intérieur de la balise (attributs repérables au signe « = » placé entre le nom de l’attribut et sa valeur entre guillemets);
- un chevron fermant (>).

Les balises au sein du document XML fonctionnent le plus souvent par paires : on parle alors de « balise ouvrante » (ex. : <h1>) et de « balise fermante » (ex. : </h1>), cette dernière ayant la caractéristique d’être munie d’un slash (/) avant le rappel du nom de la balise et de ne pas recevoir d’attributs.



L'ensemble de l'information balise ouvrante/contenu/balise fermante s'appelle un « élément ».

Pour être complet, mentionnons l'existence de balises vides qui ont pour fonction principale de recevoir des attributs. C'est le cas de la balise HTML `<img>` qui dispose d'un attribut `src` recevant le nom du fichier image (ex. : `<img src = "miles_1984.jpg"/>`).

Les balises XML

XML utilise un mécanisme comparable à celui de HTML, mais avec davantage de possibilités et de contraintes.

Au niveau des possibilités supplémentaires, XML permet de définir son propre jeu de balises, tandis que HTML impose les siennes (avec le laxisme déjà indiqué). On pourra donc organiser sa taxonomie en fonction de la finesse de repérage voulue, tout en choisissant un vocabulaire adapté. La description du comportement des balises est formalisée dans un système de déclaration qui peut prendre plusieurs formes techniques : la plus connue et la plus simple est la DTD, mais on peut recourir à XML Schema Description Language (XSDL souvent abrégé en XSD) ou à Relax NG.

Au niveau des contraintes, XML impose des règles au niveau du nommage des balises – on dit généralement qu'elles doivent être des « noms XML », terme commode pour désigner des règles de formation de mots conformes aux exigences XML – ainsi qu'au niveau de l'utilisation des balises et du contenu :

ACRONYME DTD

Définition de type de document

● Morphologie d'une balise munie d'attributs (ici une balise HTML)

XML, HTML, XHTML

HTML utilise un jeu figé de balises, contrairement à XML, et ne s'appuie pas sur une taxonomie contrôlée. Par ailleurs, HTML est utilisé de manière très laxiste au niveau de la casse des noms de balises : `<h1>` et `<H1>` sont considérés comme identiques en HTML, tandis qu'en XML cela désignerait deux éléments structurels différents. Le laxisme de HTML a été mis sous contrôle avec XHTML qui redéfinit les balises HTML mais en imposant les règles de XML. Dans le comportement XML, la casse des balises est imposée et les paires de balises doivent être correctement écrites : toute balise ouvrante doit avoir sa contrepartie fermante.

- les balises XML ne doivent pas commencer par le mot «XML» (quelle que soit la casse) : `<xmlsection>` est interdit;
- les balises XML ne doivent pas commencer par un chiffre : `<1Chapitre>` sera rejeté. En revanche, `<chapitre1>` est correct;
- les balises XML ne doivent pas contenir d'espace ou de retour chariot : `<titre section>` n'est pas autorisé;
- les balises XML ne doivent pas contenir de caractères de ponctuation autre que *l'underscore*, le trait d'union ou le point. `<titre_section>`, `<titre-section>` et `<titre.section>` sont autorisés, tandis que `<titre/section>` sera refusé. Le double point est autorisé dans des cas particuliers («espace de noms») : `<dc:language>` est une balise qui appartient à l'espace de nom `dc` (Dublin Core);
- les balises XML sont sensibles à la casse : `<TITRE>`, `<Titre>` et `<titre>` sont des balises différentes.

Au niveau du document XML, l'utilisation des balises est soumise à différentes contraintes :

- les balises XML doivent s'emboîter. Aucun chevauchement n'est autorisé : `<p>exemple de <i>texte</p> mis en valeur</i>` est interdit;
- toute balise ouvrante doit avoir sa contrepartie fermante : le début d'un paragraphe `<p>` doit avoir une fin de paragraphe `</p>`. Pour les balises vides, ce principe est respecté de fait, puisque la balise est à la fois ouvrante et fermante (d'où la syntaxe `<img/>` pour une image par exemple).

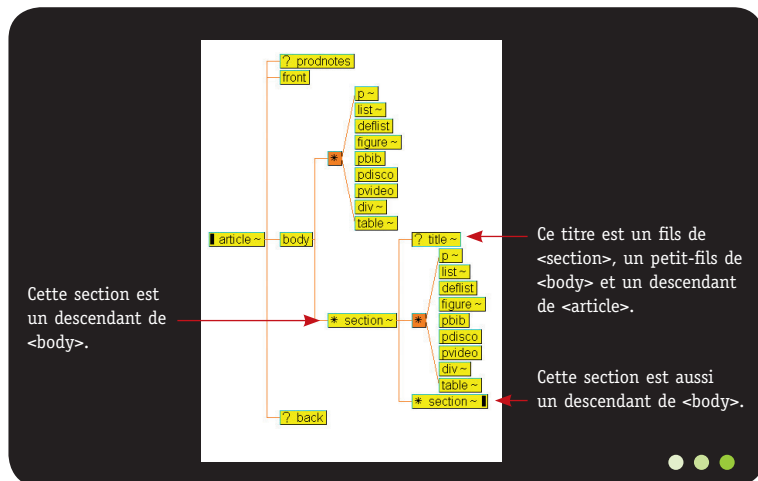
À NOTER

Le formalisme imposé au niveau des noms et des documents XML est indépendant de la présence ou non d'une DTD. Les règles XML générales introduisent des contraintes de nommage et d'appariement des balises. La DTD introduit, en plus, des contraintes de comportement des balises (apparition, fréquence).

Enfin, le contenu XML ne peut pas utiliser directement trois signes réservés : il s'agit du chevron ouvrant (`<`), du chevron fermant (`>`) et de l'esperluette – également appelée ampersand – (`&`). Il existe bien sûr un moyen d'exprimer ces signes en XML.

Éléments, relations et arbres

Les concepteurs de XML ont repris de SGML le mot «élément» pour désigner une structure ou une sous-structure repérable par des balises : on parlera ainsi de l'élément «article», de l'élément «p» ou de l'élément «i», dont la représentation dans le document XML commencera par les balises `<article>`, `<p>` et `<i>`.



La relation entre éléments est exprimée par un vocabulaire tiré du vocabulaire généalogique (limité au masculin). On parlera dès lors de père, de fils, d'enfants, d'ancêtre..., pour désigner la position d'un élément relativement à un autre.

La représentation de la structure et de ses sous-structures fait enfin appel à un vocabulaire plus botanique : on parle d'arbre, de branche, de feuille et de racine.

Attributs

Les balises seules ne suffisent pas à exprimer toute la richesse et les variantes possibles au sein d'un document. Le mécanisme d'«attribut» permet ainsi, sans multiplier le nombre des éléments, de définir des caractéristiques très fines et/ou très précises.

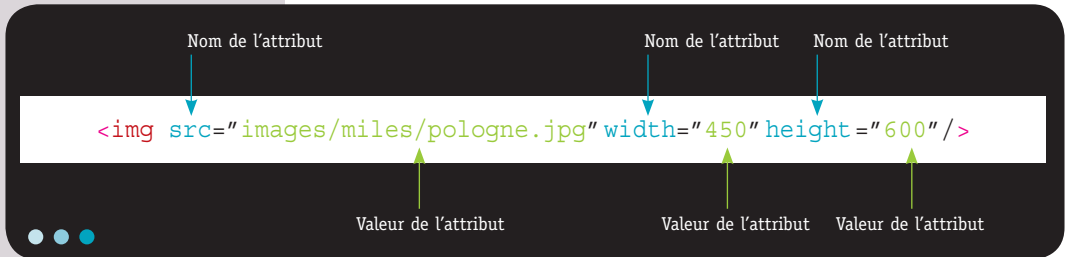
Un attribut est une information supplémentaire que l'on ajoute dans une balise ouvrante, en respectant un formalisme strict. Tout comme en HTML, les éléments peuvent recevoir de nombreux attributs dont le nom est toujours imposé par les règles XML et dont les valeurs (c'est-à-dire le contenu) répondent à des règles contraignantes mais simples :

- les valeurs d'attributs doivent être insérées dans des guillemets simples ou doubles. La présence de guillemets permet de prendre en compte des valeurs d'attributs où existent des espaces. On écrira ainsi ``;

SGML

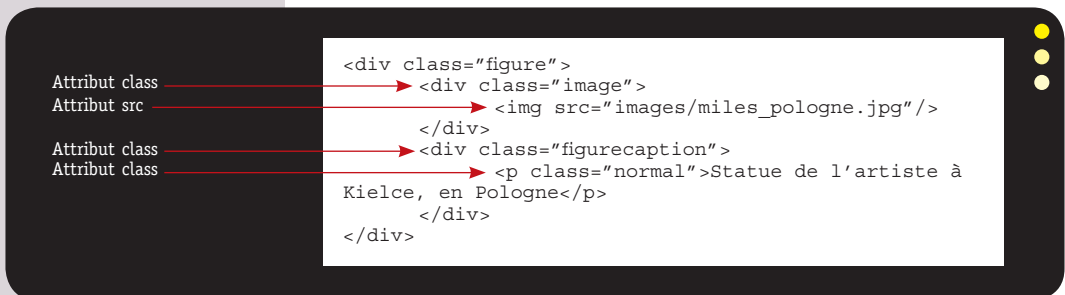
SGML (*Standard Generalized Markup Language*) a été élevé en 1986 au rang de norme internationale (ISO 8879) pour la structuration d'ouvrages à dominante technique. SGML est issu du travail d'ingénieurs d'IBM, Charles Goldfarb, Edward Mosher et Raymond Lorie (d'où les trois lettres GML) confrontés à la nécessité de produire industriellement la documentation des ordinateurs du constructeur. SGML permet de décrire des taxonomies évoluées. Trop compliqué pour les usages courants, SGML a inspiré à Tim Berners Lee la démarche HTML en 1990. HTML, trop orienté mise en pages et limité, conduit à la création de XML en 1999.

- une balise ne peut pas recevoir deux attributs ayant le même nom : on ne pourrait donc pas écrire `<section class="encadre" class="citation">`.



● Morphologie d'un attribut

Le mécanisme d'attribut donne une souplesse incomparable au XML en évitant d'avoir à déclarer un trop grand nombre de balises pour refléter toutes les subtilités demandées par la structuration documentaire. Reste toutefois à trouver un compromis entre balise et attribut pour identifier un élément structurel : dans quel cas choisir l'un plutôt que l'autre ? Il n'y a pas de réponse tranchée, et dans ce domaine, les spécialistes n'ont pas fini d'en débattre.



● Fragment de code HTML illustrant l'utilisation des attributs

Visualiser le document XML

Un document XML n'est qu'une combinaison de balises et de texte signifiant organisés en arbre dont la lecture est assez aride. Aussi cherche-t-on à visualiser les documents XML selon une représentation plus attrayante, sans perdre de vue la structuration de fond.

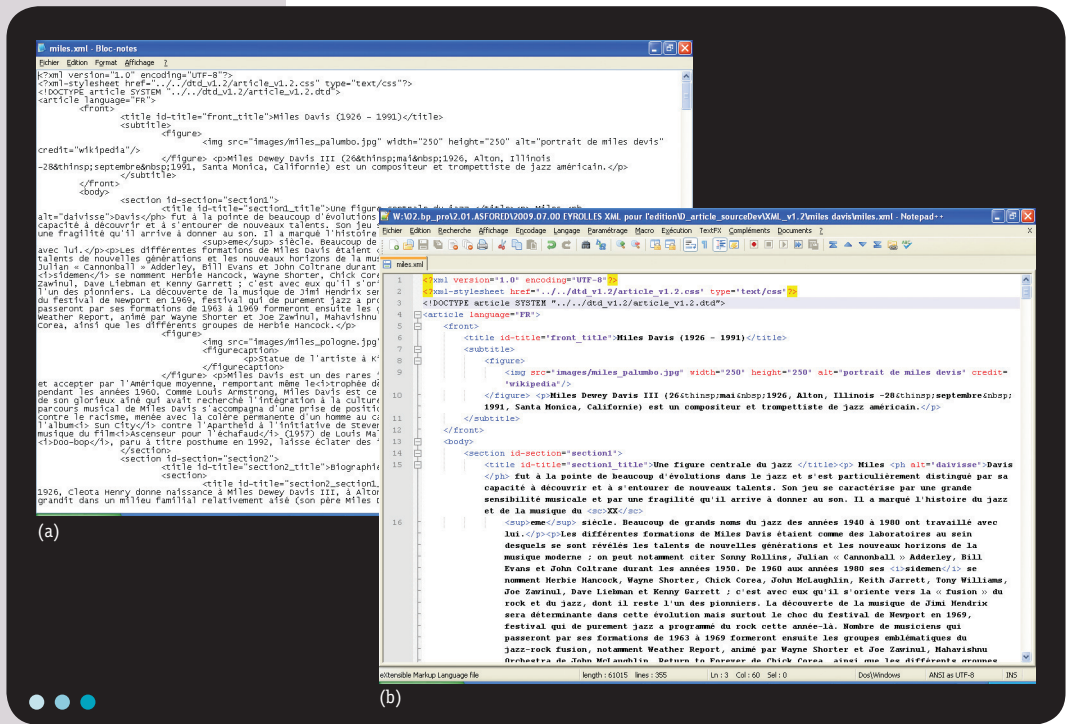
Visualiser le XML dans un éditeur de texte

Un document XML étant du texte pur, il est toujours visualisable dans n'importe quel éditeur de texte, le plus élémentaire soit-il. Dans ce cas, la structure arborescente n'apparaît pas au regard de l'utilisateur et la lecture est très difficile. Des éditeurs moins primitifs, comme Notepad ++ (voir figure page suivante), permettent d'organiser le texte XML afin de le rendre plus lisible, grâce à deux mécanismes :

- l'indentation, qui consiste à décaler vers la droite l'élément contenu par rapport à l'élément contenant;
- la coloration syntaxique permettant de différencier les constituants du flux XML : balises, attributs, valeurs d'attributs, texte.

Visualiser le XML dans un navigateur

Les navigateurs modernes sont tous capables de lire du XML. Si rien n'est spécifié au niveau du document XML, le navigateur restitue celui-ci selon un graphisme minimal consistant à indenter l'information XML pour représenter son arborescence. Quelques mini-fonctionnalités sont présentes, comme la possibilité de plier/déplier les éléments. C'est déjà beaucoup, mais cela reste encore bien aride.



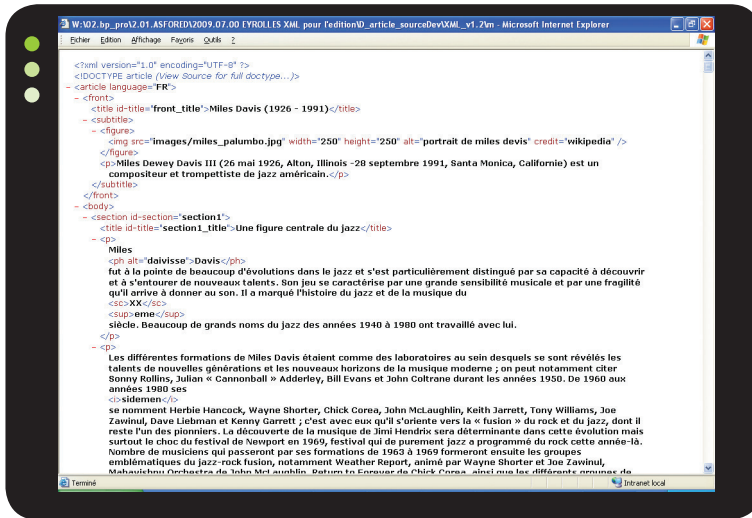
● Visualisation d'un document XML à l'aide du bloc-notes (a) et à l'aide de Notepad ++ (b). L'indentation et la coloration syntaxique aident considérablement la lecture.

L'utilisation d'une feuille de styles CSS

Une autre manière de visualiser le document XML consiste à lui accrocher une feuille de styles CSS (*Cascading Style Sheet*) : celle-ci va associer à chaque élément, en tenant compte du contexte, un rendu graphique plus attrayant pour représenter la structure. On choisira ainsi de décaler vers la droite les blocs d'information en dépendance hiérarchique : une sous-section sera ainsi décalée de 15 pt à droite pour mettre en évidence sa position dans l'arbre) et de mettre en couleur des balisages interne (du vert italique pour le nom d'une revue et du bleu pour le nom d'un livre). Le document XML devient alors lisible pour un collaborateur n'ayant que des connaissances très élémentaires en XML.

L'utilisation d'une transformation XSLT

La transformation par un script XSLT permet de modifier complètement le rendu d'un document XML et de l'adapter à ses besoins. XSLT, qui est abordé au chapitre 7, est l'outil de base de manipulation des documents XML en vue de leur publication.



- Visualisation d'un document XML dans d'un navigateur

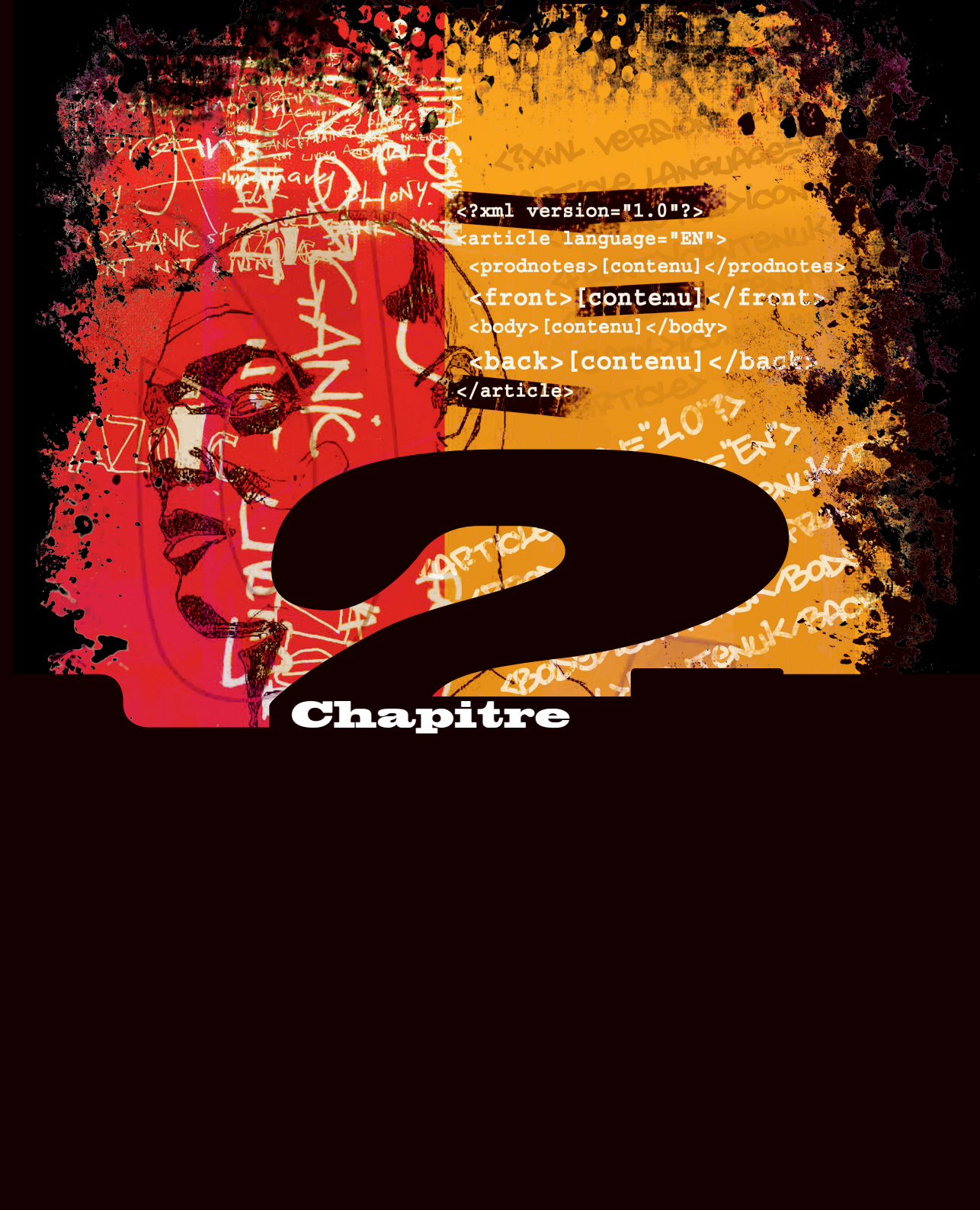
Attention

Le terme « script XSLT » n'est pas un terme officiel. On parle généralement de « feuille de styles XSLT », mais l'utilisation du mot feuille provoque la confusion avec les feuilles de styles CSS, les feuilles de styles des traitements de texte, etc.

Au stade de la visualisation d'un document XML, cette technique peut être utilisée au vol à l'aide d'un navigateur. Techniquement, on accroche au document XML, non plus une feuille de styles CSS, mais un script XSLT. Le navigateur prend alors en charge le document XML, exécute le script XSLT (grâce au processeur XSLT fourni en standard avec les ordinateurs modernes) et restitue une page HTML plus ou moins sophistiquée.

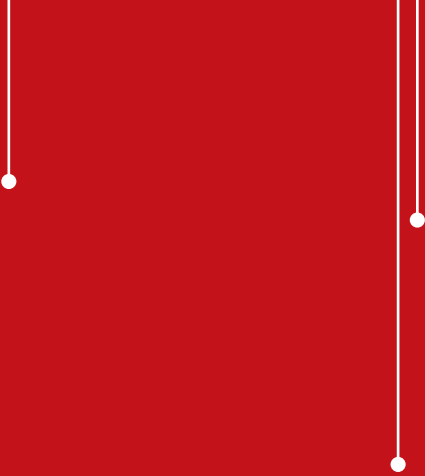


- Visualisation d'un document XML associé à un script XSLT à l'aide d'un navigateur. À ce stade, on ne cherche pas encore un rendu esthétique.



```
<?xml version="1.0"?>
<article language="EN">
  <prodnotes>[contenu]</prodnotes>
  <front>[contenu]</front>
  <body>[contenu]</body>
  <back>[contenu]</back>
</article>
```

Chapitre



Les principaux composants structurels



Même s'il existe une infinité de modèles possibles fonction des spécificités réelles ou supposées de tel ou tel ouvrage ou collection, il est possible de déterminer des composantes structurelles qui se retrouvent dans de nombreuses taxonomies. Une compréhension de celles-ci facilitera grandement la prise en main de XML dans l'édition.



Structure hiérarchique

L'organisation de l'ouvrage en parties, chapitres, sections est une constante de l'édition, parfois masquée par la complexité graphique des mises en pages. L'édition électronique renforce cette démarche en raison des automatismes à mettre en œuvre lors de la publication et de la nécessité de pouvoir se repérer électroniquement dans l'ouvrage.

Il est maintenant assez bien admis par les différents acteurs XML de l'édition qu'un document (livre, article, étude, essai, etc.) comporte trois divisions principales :

REMARQUE

Les outils de navigation popularisés par l'hypertexte étaient déjà utilisés dans les publications papier : la table des matières est un outil de navigation dans le livre, le système de liens se basant sur les numéros de page. De la même manière, les index sont une aide à la navigation proposée par l'auteur ou par des contributeurs spécialisés.

- un préliminaire composé de pages d'identification et d'introduction complété éventuellement d'éléments de structure comparables à un chapitre (préface, avant-propos). Il est usuel d'identifier cette partie par une balise `<front>` (littéralement : « ce qui est devant ») ;
- un corps où se trouve le contenu proprement dit. Pour un roman, le corps comporte généralement une succession de chapitres, éventuellement regroupés en parties. Ici également, l'usage a retenu la balise `<body>` (littéralement : « corps ») ;
- un postliminaire où sont regroupés les index, notes, glossaire et toute information complémentaire jugée utile par l'auteur. Le postliminaire est souvent balisé `<back>` ou `<rear>` (littéralement : « ce qui est derrière »).

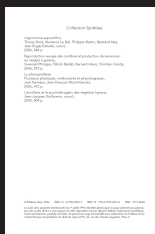
Front



Faux titre



Titre



Collection/copyright



Exerque



Remerciements



Table des matières



Avant propos

- Organisation générale d'un document, ici un livre : préliminaire (front), corps (body), postliminaire (back) (source : Éditions Quae)

Body



Introduction



Chapitre 1



Chapitre 2



Chapitre N

Back



Bibliographie



Glossaire



Abréviations



Index

Par ailleurs, il n'est pas nécessaire de tout structurer, car de nombreux éléments dérivent d'éléments existants :

- la table des matières n'est que la vue rassemblée des titres de chapitres, de sections et éventuellement de sous-sections associés au numéro de page physique. La contrepartie électronique est une table des matières active sous forme de liens hypertextes permettant un branchement direct vers l'élément concerné. Dans les deux cas, la table des matières n'a pas à être prévue en termes de structure car elle sera générée lors de la publication (on pourra toutefois prévoir dans le XML l'emplacement préférentiel retenu pour la table des matières) ;
- les notes de fin d'ouvrage n'ont pas non plus à être identifiées structurellement. Leur regroupement n'est que la conséquence d'un choix d'affichage en fin de volume au lieu d'un affichage en bas de page ou en fin de chapitre. Techniquement, la note est une sous-structure du texte, une grosse parenthèse, et doit (devrait) être saisie dans le flux de texte à l'endroit où l'auteur insère habituellement un appel de note ;
- de la même manière, il n'est pas utile de prévoir la table des figures qui peut facilement être générée à partir du balisage des figures dans le texte.

Le statut des index est moins tranché :

- si l'index est constitué d'une hiérarchie de mots ou de termes renvoyant aux pages de l'ouvrage, on se contentera de poser les repères d'index dans le corps du texte et d'actualiser l'index lors de la publication ;

À NOTER

On structure parfois les tables des matières lorsqu'on effectue des opérations de rétroconversion destinées à restituer presque à l'identique l'information disposée sur la page. Dans ce cas, il s'agit davantage d'une restitution XML de la présentation existante que d'une structuration XML en vue de la publication multisupport.

- si l'index incorpore des commentaires, comme il est d'usage en sciences humaines, il faudra alors le saisir au sein du fichier XML en posant bien sûr les liens entre l'entrée d'index et la « page » concernée.

<front>

Les pages liminaires d'un ouvrage papier comportent des pages de faux-titre – qu'il est d'usage de conserver dans un flux XML mais qui, en toute logique, ne devraient pas s'y trouver, la plupart des informations étant déjà disponibles sous forme de métadonnées – et un ensemble d'informations préalables pouvant se limiter à une ou deux pages (dédicace, exergue) ou à une structure un peu plus complexe.

Une modélisation possible consistera à structurer les préliminaires en éléments <page> et en <chapitre>.

- Les <page> regroupent en unités ce qui correspond aux pages physiques du début du livre ; elles seront restituées en blocs d'information relativement homogènes dans les versions électroniques. On y trouvera un exergue, une dédicace, des informations de copyright.
- Les <chapitre>, appelés ainsi parce qu'ils ont une structure comparable à celle des « vrais » chapitres, comprennent les préfaces et divers liminaires (discours, documents préalables, etc.).

<front>

The diagram illustrates the structure of the front matter of a book, organized into two rows of three elements each. The top row contains: Faux titre <page>, Titre <page>, and Collection/copyright <page>. The bottom row contains: Remerciements <chapitre>, Table des matières (Non saisie, sera générée), and Avant propos <chapitre>.

Pour un article, <front> regroupe le titre de l'article, le sous-titre, les auteurs de l'article et différents blocs optionnels comme les résumés en français ou en anglais.

● Organisation des préliminaires d'un ouvrage scientifique (source : Éditions Quae)

<body>

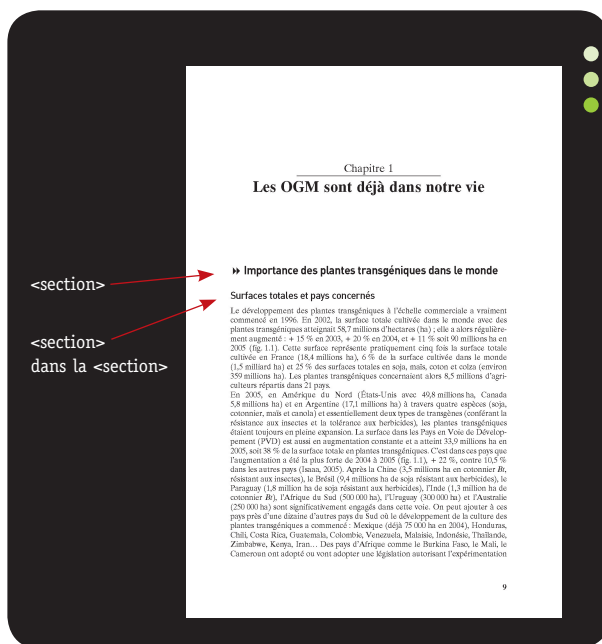
Le corps d'un livre est lui-même hiérarchisé : l'ouvrage peut ainsi être structuré en <partie> regroupant des <chapitre>.

À NOTER

Le vocabulaire adopté dans le présent ouvrage pour qualifier dans les exemples les parties, chapitres, sections, sous-sections, etc. est arbitraire. XML permet de choisir son propre vocabulaire, sachant que différents termes tendent à s'imposer sous l'influence de HTML : <p> pour un paragraphe, <i> pour l'italique, <body> pour le corps d'un document...

- La <partie> dispose généralement d'un <titre>, lui-même composé d'un repère de navigation («partie 1», «partie 2», etc., ou simple numéro), un libellé éventuel et divers ajouts (citation, exergue...). La partie commence souvent par un nombre plus ou moins important de paragraphes, de listes, de tableaux, etc., servant d'introduction aux chapitres proprement dits.
- Le <chapitre> se décompose comme n'importe quel document : la partie liminaire comporte éventuellement un titre plus ou moins complexe (i.e. avec surtitre, sous-titre, exergue, citation, etc.) suivi d'une structure introductive optionnelle constituée d'un mélange de paragraphes, de listes, de tableaux... puis d'une structure récurrente de sections. «Récursif» signifie que l'on peut trouver la structure à l'intérieur de la structure : une sous-section s'exprimera ainsi sous la forme d'une <section> dans la <section>, etc.

● Organisation récursive des sections au sein d'un chapitre
(source : Éditions Quae)



<back>

La fin d'un ouvrage ou d'un document pose quelque problèmes d'interprétation ou de choix : les annexes constituent-elles un postliminaire ou font-elles parties du corps de l'ouvrage ?

Ce qui nous semble important, c'est l'homogénéité au sein des structurations XML d'une maison d'édition. Si l'on choisit d'insérer dans les postliminaires l'index, la bibliographie et le glossaire, à l'exclusion des annexes conservées dans le corps de l'ouvrage, il est préférable que tous les ouvrages répondent à cet impératif ; plus facile à dire qu'à faire, chaque éditeur au sein d'une maison d'édition ayant sa propre vision des choses.

Les modèles documentaires prévoient généralement la possibilité de structurer les postliminaires avec une structuration simple (ex. : les <page> des préliminaires) ou plus complexe (ex. : <chapitre>), ce qui permettra d'organiser des glossaires et des index aux comportements sophistiqués dans la version électronique.



● Modélisation des postliminaires d'un ouvrage scientifique (source : Éditions Quae)

Les structures blocs

Les structures blocs correspondent à tout ce qui se composera, statiquement ou dynamiquement, dans un rectangle. L'exemple le plus intuitif est le paragraphe. Mais les blocs, en XML, vont bien au-delà et incorporent la structure de liste, de tableau, de figure ainsi que la définition de structures d'emballage ou « conteneurs ».

Le paragraphe

Le paragraphe, souvent désigné par la balise `<p>`, est l'unité sémantique la plus répandue : elle est presque entièrement sous la gouverne de l'auteur qui décide de l'endroit où le paragraphe va se terminer.

Sur le papier, il est d'usage de rendre explicite le début d'un paragraphe par un retrait de première ligne et sa fin par une ligne creuse. Le retrait a pour fonction de lever l'ambiguïté créée par un paragraphe se terminant exactement, une fois composé pour le papier, en fin de ligne (donc sans ligne creuse de fin de paragraphe). Bien sûr, le retrait n'est pas une information structurelle.

Il faut par ailleurs être capable, lors de la composition, d'empêcher que certains éléments qui pourraient prendre le statut de paragraphes, comme une strophe de poésie, ne se trouvent dotés d'un retrait. On prévoit dans ce cas une structure bloc particulière comme `<lg>` (pour *lines group*) englobant des `<l>`, c'est-à-dire des lignes.

D'un point de vue structurel, le paragraphe va être utilisé de manière très extensive afin de répondre à tous les cas de figure possibles : c'est ainsi que les notes sont structurées en paragraphes, même si la plupart d'entre elles n'en comportent qu'un seul. Une cellule de tableau sera également structurée en paragraphes, même si la plupart des cellules de tableau ne contiennent le plus souvent que très peu de texte.

Cela dit, on peut spécialiser certains paragraphes afin de ne pas surcharger leur structure de sous-éléments. On définira ainsi un paragraphe bibliographique (ex. : `<pbib>`) pour répondre à différents besoins :

- faire apparaître au sein de ces paragraphes un balisage spécifique (auteur, titre d'ouvrage, titre de revue...);
- extraire automatiquement d'un ensemble de publications dotées de ce type de paragraphe tous les éléments bibliographiques afin de produire une compilation thématique (bibliographie d'un domaine d'intérêt par exemple);
- rendre la bibliographie directement exportable dans les formats des principaux logiciels bibliographiques (EndNotes par exemple).

À NOTER

En XML, les paragraphes du corps du livre, les paragraphes d'une note ou les paragraphes d'une cellule de tableau ont souvent tous la même structure : on utilise donc la même balise, par exemple `<p>`. Mais comme ces paragraphes se trouvent dans des contextes différents (en l'occurrence le corps, la note, le tableau), leur apparence visuelle et même comportementale, lors de la publication, pourra être complètement différente.

- Exemple de paragraphe spécialisé « bibliographie » (`<pbib>` déclarée dans `article_v1.2.dtd`). Cette structuration est destinée à un usage électronique de recherche dans une base bibliographique comme Google books ou Google scholar.

La liste

La liste est un paragraphe doté d'une structure interne plus fine se décomposant en :

- une partie introductive optionnelle, structurée sous forme d'un paragraphe classique. On parle généralement de « paragraphe introductif »;
- un ensemble d'items eux-mêmes structurés en paragraphes. L'item se repère, lors de la publication, par un repère graphique : un casseau, une puce, un disque, etc.

Paragraphe
introductif de liste

Paragraphe
d'item de liste

Item de liste

- Morphologie d'une liste non ordonnée (i.e. non numérotée)

REMARQUE

Certains auteurs recourent à un nombre parfois considérable de repères graphiques (lettres, chiffres romains, signes graphiques) correspondant à de subtiles distinctions sémantiques. Cela n'est généralement pas souhaitable en édition électronique comme en édition papier, mais ne pose aucun problème en structuration XML.

Il est d'usage de distinguer les listes numérotées des listes repérées par un symbole typographique (généralement qualifiées de « listes à puces »). Pour être utile, cette différenciation doit correspondre à un souci sémantique de l'auteur : la liste numérotée est utilisée dans le cas où un ordonnancement est suggéré (i. e. il faut procéder dans un ordre imposé), tandis que la liste à puces est utilisée dans les cas de simples énumérations.

CALS

CALS est une DTD tableau développée pour les besoins du ministère de la défense américain (DoD) dans les années 1985 et reprise par OASIS. On parle désormais de CALS/OASIS. Cette structure de tableau est prise en compte par la plupart des éditeurs XML et est quasiment devenue une norme.

Pour en savoir plus :
<http://dtd.nlm.nih.gov/options/OASIS/tag-library/19990315/index.html>.

Le tableau

Le tableau a une structure hybride, à la fois typographique (l'organisation en cellules séparées par des filets) et sémantique (le typage d'informations sous forme de colonnes). Il s'agit d'une structure bloc dans la mesure où sa composition s'inscrit dans un rectangle. Il existe plusieurs modèles de tableaux, dont les plus connus sont HTML et CALS/OASIS.

La structure de tableau HTML est trop limitée pour les besoins de la publication papier mais est universellement utilisée pour la publication électronique. CALS/OASIS (CALS par abus de langage) répond mieux aux besoins du papier : aussi structurera-t-on les tableaux en CALS avant d'effectuer des transformations/simplifications pour l'édition électronique.

Une alternative consiste à produire les tableaux sous forme d'images, donc de manière externe au flux XML. L'inconvénient est que le tableau ne pourra plus se recomposer en édition électronique, à la fois en dimensions (si l'on veut garder la lisibilité) et en police.

CALS structure les tableaux en trois sous-ensembles :

- tête (thead);
- pied de tableau (tfoot);
- corps du tableau (tbody).

Pour des raisons de traitements informatiques, les concepteurs du modèle de tableau CALS ont imposé que le pied de tableau soit placé avant le corps du tableau. C'est un peu déroutant, mais logique : il suffit de penser aux tableaux longs où le pied de tableau doit être répété sur plusieurs pages imprimées.

Les tableaux sont organisés en colonnes et en lignes, le croisement définissant une cellule (appelée <entry> en CALS). Le contenu de la cellule est laissé à l'initiative de l'éditeur qui peut la structurer de manière très poussée en autorisant des listes, des images, etc.

Les cellules peuvent être fusionnées horizontalement (on parle de « colspan », ce qui veut dire approximativement « regroupement de colonnes ») ou verticalement (on parle alors de « rowspan »).

La largeur des colonnes est exprimée soit de manière absolue (en pixels ou en points), soit de manière relative : dans ce dernier cas, on saisit la largeur en pourcentage de la largeur totale du tableau. Les colonnes d'un tableau de 4 colonnes de largeur identique recevront ainsi la valeur 25%.

Attention toutefois : les éditeurs XML qui supportent CALS n'adoptent pas tous les mêmes conventions de nommage de la largeur des colonnes et des cellules fusionnées. L'éditeur XMetal utilise la notation 25*, tandis qu'Oxygen utilise 25%. Il est donc nécessaire d'imposer à son prestataire des conventions de nommage afin que l'ensemble de la production reste homogène.

ACRONYME Oasis

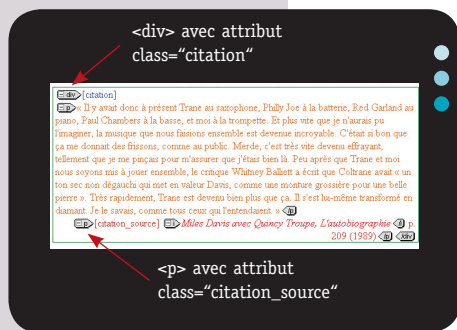
Organization for the Advancement of Structured Information Standards

	Variétés NT	Variétés résistantes aux herbicides	Var résistantes aux herbicides % Var NT
Rendement kg/ha	1 498	1 652	+ 10,3
Revenu brut (DC)*/ha	462	514,7	+ 11,4
Marge brute (DC)/ha	202,2	232,5	+ 15
* DC : Dollar Canadien.			

● Structure d'un tableau CALS/OASIS et les principales balises

Repérage de blocs particuliers

Il n'est pas toujours opportun de donner un nom différent à chaque structure bloc d'un document XML : la multiplication des balises rend la structure de plus en plus compliquée, difficile à mémoriser et à maintenir. En pratique, on utilise une structure bloc que l'on pourrait qualifier par abus de langage de « fourre-tout ». La tradition, héritée de HTML, veut qu'on l'appelle `<div>`, mnémotechnique de « division ». La `<div>` va permettre de délimiter des ensembles de blocs ; on lui donne éventuellement un titre et on la qualifie à l'aide d'un attribut (ex. : `class`), ce qui permettra d'absorber toutes les structures locales que l'on ne parvient pas à modéliser dans les structures hiérarchiques de plus haut niveau (chapitre et sections).



- Structure d'une `<div>` ne comportant pas de titre (`title` est optionnel dans la DTD correspondante).

- Conteneur de type bloc. La balise `<subtittle>` contient uniquement des paragraphes. Elle permettra de définir une restitution papier ou électronique spécifique de tout le sous-titre (sans avoir à définir le comportement typographique de chaque paragraphe).

La `<div>` doit toutefois être utilisée avec modération : la règle veut qu'on ne l'utilise que lorsque la structuration à l'aide de balises hiérarchiques classiques est impossible.

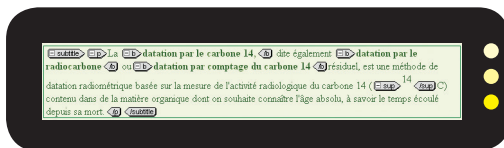
Conteneurs et contenus mixtes

Un aspect assez déroutant de XML est la présence de conteneurs (*containers*) permettant de repérer une structure sans que cela semble nécessaire à première vue : il s'agit de balises assurant un « emballage » d'autres balises. Les conte-

neurs vont permettre d'isoler des paquets d'informations eux-mêmes structurés, avec pour objectif de simplifier ou d'affiner les processus de composition papier ou électronique.

Les conteneurs sont presque toujours de type bloc, bien que l'on puisse trouver des conteneurs de type inline.

- La balise `<front>` est un conteneur de type bloc : elle ne contient que des balises et sert uniquement à identifier la partie préliminaire d'un ouvrage. Il aurait été possible de structurer un livre en s'affranchissant de cette balise ; mais sans celle-ci, l'identification de ce qui appartient au début d'un livre en vue de lui donner un traitement typographique spécifique lors de la mise en pages papier ou électronique est plus difficile.
- De la même manière, la balise `<div>` permet d'isoler un ensemble d'informations que l'on souhaite qualifier à l'aide de l'attribut `class`. L'exemple déjà cité est le bloc de citation(s) que l'on veut restituer typographiquement par un retrait gauche par rapport au texte courant : l'usage de `<div class="citation">` évite, en particulier, de déclarer une balise spécifique `<citation>`.



Les structures inline

Les structures inline correspondent à tout ce qui se composera dans le contexte d'une ligne. L'exemple le plus intuitif est l'enrichissement typographique de type italique ou gras. Les structures inline mélangent souvent du texte et d'autres balises : on parle alors de « contenu mixte ».

Enrichissement

La plupart des textes ont besoin d'enrichissement, soit pour exprimer une intention sémantique de l'auteur (exemple : mise en valeur de type italique ou gras), soit pour exprimer un rendu typographique signifiant « de type exposant » ou « indice » (exemple : H_2O et non pas $H2O$ pour exprimer la molécule d'eau).

Typographie sémantique

La typographie sémantique correspond à un enrichissement typographique, traduction visuelle de l'intention de mise en évidence de l'information par l'auteur. En XML, on devrait en toute rigueur définir une balise avec un terme non typographique : `<mise_en_valeur>` ou `<mev>`.

Dans la pratique, on se contente d'utiliser un terme typographique usuel, `<i>` pour italique, `<b>` pour gras (*bold* en anglais) ou encore `<sc>` pour les petites capitales (*small caps* en anglais) destiné à exprimer les siècles en chiffres romains, par exemple.

Typographie signifiante

Pour la typographie signifiante, on trouve souvent les balises `<sup>` pour l'exposant et `<sub>` (ou parfois `<inf>`) pour l'indice, par héritage des habitudes du HTML.

● Expression d'un exposant à l'aide de la balise `<sup>`

Le rapport $\frac{14}{12} \frac{^{14}C}{^{12}C}$ total est considéré comme uniforme dans l'atmosphère, la surface des océans et la biosphère en raison des échanges permanents entre les organismes vivants et leur milieu. ^②
De plus, on suppose que le flux de rayons cosmiques est constant sur une longue période de temps (première approximation). Par conséquent, le taux de production du radiocarbone est constant, donc le rapport $\frac{^{14}C}{^{12}C}$ total dans l'atmosphère, la surface des océans et la biosphère est constant (le nombre d'atomes produits égale le nombre d'atomes qui se désintègrent). ^③

Images et objets multimédias

XML est un langage purement textuel : une image ou un objet multimédia non descriptible en XML ne pourra pas être incorporé au flux XML. Seule sera introduite la référence à un nom de fichier.

Une image telle que `miles1984.png` sera donc « appelée » dans le flux XML et non pas incorporée. Comme le nom du fichier image n'est pas l'image elle-même, il est plus logique (ou au moins plus élégant) de renseigner le nom du fichier sous forme d'un attribut. L'habitude HTML a quasiment standardisé l'appel d'image par l'élément vide `<img/>` et l'attribut `src` (pour *source*). Il est également d'usage d'introduire les dimensions de l'image sous la forme d'attributs `width` et `height` recevant les dimensions en pixels. Il s'agit d'un héritage HTML peu utile dans une pratique multisupport. On préfère procéder, préalablement à la publication, au calcul de la taille optimale de l'image et à un retailage de celle-ci pour le support concerné : en version électronique, cela évitera de transmettre à un téléphone une image largement plus grande que l'affichage permis (sauf si l'on veut autoriser un zoom).

● L'encodage XML de l'image. On remarquera que la balise `<img>` est vide : elle ne contient rien et ne vit que par ses attributs : `src` pour le nom du fichier image, `width` et `height` pour ses dimensions, `alt` pour fournir le texte descriptif de l'image pour les non-voyants (via une synthèse vocale par exemple) et `credit` pour les informations de crédit d'image.

L'attribut `src` donne le nom du fichier image et son chemin relatif d'accès

Identifiant de la figure, pour les liens hypertextes depuis le corps du document

```
<figure id-figure="figure_1" >
  
  <figurecaption>
    <p>Courbe de calibration de la datation par le
    carbone 14 </p>
  </figurecaption>
</figure>
```

L'attribut `alt` sert à stocker un texte descriptif de l'image pour les non-voyants

Légende d'image

Par ailleurs, de plus en plus d'objets multimédia s'expriment directement en XML. SVG, par exemple, est un format (assez verbeux) descriptif d'un dessin vectoriel qui peut être directement incorporé dans le flux XML par un mécanisme spécifique d'appel, non abordé dans le présent ouvrage.

Formules mathématiques, code

Les formules mathématiques ont encore un statut hybride : dans la plupart des systèmes actuels, la formule mathématique est insérée sous forme d'une image qu'il a fallu produire par ailleurs. Les formules mathématiques sont toutefois représentables dans un vocabulaire XML spécifique (MathML) qui les rend indépendantes des supports de publication. Ces supports n'incorporant pas encore de visualiseur MathML, l'habitude est de faire appel à un élément spécifique (`<eqn>` ou `<formula>`) référençant le fichier image ou incorporant directement la formule exprimée en TeX.

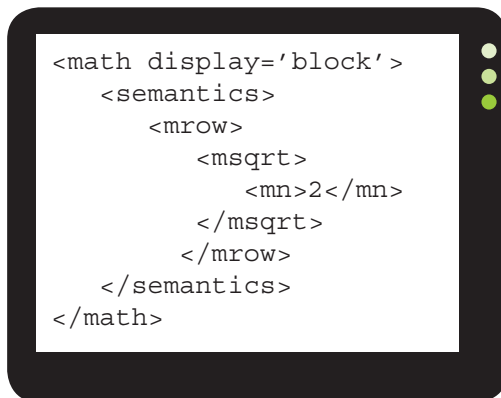
TeX (prononcez « tec ») est un langage de représentation (mais pas seulement) de formules mathématiques largement utilisé dans le monde universitaire et directement compréhensible par un non-voyant via un clavier braille ou une synthèse vocale. Il existe plusieurs variantes de TeX : il convient, lors de la saisie, d'être vigilant sur la version utilisée et de l'imposer à son prestataire.

Dans le domaine de l'informatique, la police Courier est souvent utilisée pour signaler du code dans le texte. En XML, on introduira de préférence la balise inline `<code>` qui encadrera toute référence à du code informatique et qui a l'avantage d'être sémantique. Dans la pratique, on trouve aussi la balise `<span>` munie d'un attribut `class="code"`, ce qui économise une balise dans la taxonomie.



● Encodage TeX d'une formule mathématique : solution acceptable à court terme avant que MathML ne se généralise.

● La même formule mathématique encodée en MathML



L'encodage XML des formules mathématiques

Afin de préserver l'avenir, l'habitude se répand de saisir les formules mathématiques en 3 formats : image, mathML et TeX.

- L'image suppose d'avoir figé la police de caractères utilisée pour les formules ainsi que le nombre de pixels, lui-même lié à la résolution du média cible. Il faut également fixer la position de l'axe mathématique, c'est-à-dire l'axe horizontal d'alignement de la formule (par exemple la barre de fraction).

- MathML est un encodage XML spécifique qui commence à être utilisé dans certains navigateurs via des plug-ins.

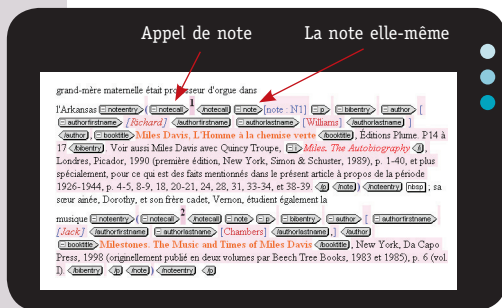
- TeX est utilisé comme alternative de composition à MathML et comme source de lecture braille.

Notes

DÉFINITION Identifiant

Un identifiant est un attribut spécial dont la valeur est nécessairement unique dans le document XML. Il permet de repérer tout type de structure, à condition bien sûr que sa présence ait été prévue. XML ne laisse pas toute liberté pour construire un identifiant : celui-ci doit répondre aux règles de constitution d'un nom XML, et en particulier commencer par une lettre, ce qui n'est pas intuitif.

● Note et son encodage XML



Les notes sont fondamentalement une structure inline. On trouve, certes, de nombreux modèles XML déclarant les notes sous forme d'un bloc structuré placé à des endroits précis d'un livre (fin de chapitre ou fin d'ouvrage) et associé à un mécanisme de liens ; cela ne constitue pas une pratique efficace, en particulier d'un point de vue technique lors des publications multisupports ultérieures.

Il est plus simple et plus performant de saisir la note là où elle est appelée, c'est-à-dire inline, en la munissant éventuellement d'un identifiant pour gérer le cas assez rare dans les livres, mais assez courant dans les articles, où la même note est appelée plusieurs fois.

Liens hypertextes

Le terme « Lien hypertexte » est utilisé pour désigner un lien cliquable. Il s'agit pourtant d'un concept aussi vieux que le livre. À la base, il s'agit du simple « voir » ou « cf. » utilisé par l'auteur pour renvoyer le lecteur à un passage de son propre ouvrage ou à un passage d'un ouvrage externe. Avec Internet, le concept s'est généralisé et étendu : les documents en ligne sont truffés de liens calculés ou saisis et le comportement du lien varie : ouverture de l'outil de courriel de la machine de l'utilisateur, lancement d'un téléchargement, ouverture d'une nouvelle page sur une machine distante, etc.

En XML, le lien est souvent une structure inline : il s'agit de poser à l'intérieur du document une balise associée à un ou plusieurs attributs spécifiques du lien attendu. Là aussi, l'héritage HTML a introduit des habitudes : on nommera `href` (qui est le mnémonique de *hypertext reference*) l'attribut exprimant un lien vers un site externe, et `mailto` le lien provoquant l'ouverture de l'outil de courriel et l'insertion de l'adresse électronique cible dans celui-ci.

Pour un lien vers une partie du document ou vers une figure, on utilisera un attribut spécifique (ex. : `refid-title` dans le premier cas et `refid-figure` dans le second cas). À cette fin, XML dispose d'un mécanisme propre permettant de s'assurer :

- que les identifiants (i.e. information d'identification d'un élément) sont uniques dans tout le document XML ;
- que toute référence à un identifiant, par exemple lors de l'introduction du lien hypertexte, correspond à un identifiant existant dans le document.

À NOTER

Ce mécanisme connu sous le nom d'ID/IDREF est expliqué plus précisément au chapitre 3.

● Encodage XML d'un lien hypertexte. La balise inline `<link>` est dotée d'un attribut `href` qui reçoit l'adresse du lien s'il s'agit d'un lien externe. Pour un lien interne vers une figure, on utilise l'attribut `refid-figure` qui reçoit la valeur d'un identifiant existant. Si l'identifiant mentionné n'existe pas dans le document, le parseur émettra un message d'alerte (voir la définition du terme « parseur » à la page 53).

Balise `<link>`

Attribut `href`

```
<section id-section="section1">
  <title id-title="section1_titre">Historique</title>
  <p> En 1960, Willard Frank Libby a reçu le prix Nobel
    de chimie pour le développement de cette métho-
    de cf. <link href="http://fr.wikipedia.org/wiki/
      Willard_Frank_Libby">article sur le chercheur
    </link> pour l'histoire de cette découverte).
  </p>
</section>
```

Repérage d'éléments inline particuliers

Comme pour les structures blocs, il n'est pas opportun de multiplier les éléments inline pour répondre à tous les besoins de typage. Une balise passe-partout, généralement appelée `<span>`, munie d'un attribut `class` est souvent utilisée à cet effet. Le terme `span`, hérité de HTML signifie en anglais « couverture/taille/portée » et veut dire à peu près en XML : « tout ce qui est inclus dans l'élément encadré par `<span>` et `</span>` ». Dans la pratique, on donne à l'attribut `class` un certain nombre de valeurs prédéterminées choisies pour être significatives dans le contexte étudié, ou bien on ajoute des attributs presque toujours facultatifs pour des raisons de compatibilité (voir le chapitre 3 sur la DTD).

DÉFINITION Typage

Le typage consiste à appliquer une balise XML à un élément structurel. Une fois typé, on dispose d'un « élément XML ».

Contenus mixtes

● Contenu mixte, théoriquement à proscrire. Les textes mathématiques ou chimiques introduisent souvent des formules en lignes en les isolant par un retour chariot. Pourtant, du point de vue du sens, on se trouve dans le même paragraphe et il n'est pas élégant, afin d'obtenir la composition voulue, de produire 2 paragraphes. On utilisera dans ce cas la balise `<img>` munie de son attribut `block="yes"` pour forcer le retour chariot.

(Source : Wikipedia)

Un élément structurel est dit «à contenu mixte» lorsqu'il mélange du texte et d'autres balises. Il s'agit d'un cas très courant dont le plus connu est le paragraphe : celui-ci contient du texte émaillé de balises typographique (*<i>*) ou sémantiques (*<auteur>*, *<langue>*, etc.). Cela dit, la définition et l'emploi non maîtrisé d'un contenu mixte peuvent donner lieu à des comportements parasites lors de la publication : mélanger du texte et des paragraphes est à proscrire, bien que l'on puisse tout à fait définir une telle structure au sein d'une DTD !

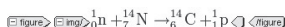
Conteneurs inline

Dans l'esprit du contenu mixte, et à l'image de ce qui se passe pour les blocs, on trouvera des conteneurs inline permettant, par exemple, de sécuriser la composition lors de la publication.

<p>Dans la haute atmosphère, des réactions nucléaires initiées par le rayonnement cosmique produisent un flux de neutrons libres. Après avoir été ralentis par collision avec les molécules de l'air, les neutrons dans une certaine gamme d'énergie (cinétique) réagissent avec l'azote pour former du radiocarbonate, selon l'équation bilan :</p>

(a)

► Dans la haute atmosphère, des réactions nucléaires initiées par le rayonnement cosmique produisent un flux de neutrons libres. Après avoir été ralentis par collision avec les molécules de l'air, les neutrons dans une certaine gamme d'énergie (cinétique) réagissent avec l'azote pour former du radiocarbone, selon l'équation bilan : ◀



(b)

☐ pbib ☐ bibentry ☐ author { ☐ authorfirstname [lan] ☐ authorfirstname ☐ authorlastname [Carr]
☐ authorlastname ☐ author ☐ booktitle Miles Davis ☐ booktitle William Morrow & Co, 1982, traduction
 française : Éditions Parenthèse, 1991 (ISBN 2-86364-057-7) : ☐ bibentry ☐ pbib

Titre de l'ouvrage
en typo pure

▣pbib Ian Carr, ▣I Miles Davis ▣ William Morrow & Co, 1982, traduction française : Éditions Parenthèse. 1991 (ISBN 2-86364-057-7) : ▣pbib

● Conteneur à contenu mixte. La balise `<author>` permet d'isoler le nom et/ou le prénom de l'auteur. Elle est éventuellement enchâssée dans une balise `<bibentry>` qui contient des balises repérant le titre de l'ouvrage écrit par les auteurs. Cela permettra, en édition électronique, de lancer une recherche automatique sur Google books ou Google scholar. Sans ce besoin de traitement lié à un comportement de la publication électronique finale, il est inutile de créer le conteneur `<author>`. Il suffit de respecter les usages typographiques et la convention de mise en italique des titres d'ouvrage.

Structures navigationnelles

Aux structures existantes pour le papier, l'édition électronique ajoute et résout ses contraintes liées à l'affichage parfois trop restreint (longueur d'un titre par exemple) et à la nécessité de se repérer et de naviguer de manière hypertextuelle dans et hors du document grâce aux structures navigationnelles.

Table des matières

Les tables des matières (ou TOC), et plus généralement les différentes tables de référence (table des illustrations, etc.), sont des vues synthétiques de la structure existante. Une TOC papier reprend simplement les titres des chapitres, sections et éventuellement sous-sections et permet la navigation par un renvoi à la page physique. En version électronique, la problématique est la même, mais soumise à des contraintes supplémentaires :

ACRONYME **TOC**

Table Of Content (table des matières)

- la longueur des titres ne doit pas dépasser 2 lignes sur l'écran cible. Dans le cas d'un téléphone, il faut raccourcir les titres et prévoir un mécanisme d'association entre le titre initial et le titre court ;
- le médium électronique ne donne pas la sensation physique de l'endroit où l'on se trouve, contrairement à un livre papier dont l'épaisseur des pages déjà lues est un indicateur de position. L'édition électronique suggère donc de multiplier les systèmes de repérage et de navigation, en introduisant des TOC partielles au niveau d'un chapitre, par exemple ;
- les chapitres qui n'ont ni numéro ni titre, assez courants dans les romans, posent un problème de génération de table des matières électronique : doit-on générer «chapitre 1», «chapitre 2», etc. ou bien introduire des titres alternatifs (début du chapitre par exemple) ?

L'indexation est un métier

Les éditeurs américains font appel à des indexeurs de métier. Les auteurs n'aiment pas s'atteler à ce travail – parfois très lourd et toujours minutieux – qui devrait être effectué dans la continuité de l'écriture. Un bon index est à la fois précis et concis (renvoi à 2 ou 3 pages maximum pour une entrée d'index).

Dans tous les cas, il faudra anticiper ces problèmes au sein de la structure et organiser la production dans ce sens.

Index

Les index sont un outil de recherche et de navigation largement utilisé pour le papier, en particulier dans le domaine professionnel. Pour être efficace, un index doit être doté d'entrées suffisamment précises pour bien cibler une recherche et chaque entrée ne doit pas pointer vers une multitude de pages.

Du point de vue structurel, on rencontre deux approches :

- soit la pose de l'entrée d'index s'effectue à l'endroit auquel l'index fait référence. Il s'agit alors d'une structure totalement inline. Suivant la complexité de l'index, on introduira une entrée principale et une entrée secondaire, dépendante de la première. Avantage : l'entrée d'index apparaît dans son contexte. Inconvénient : il faut à chaque fois saisir l'entrée ;
- soit la saisie de l'entrée d'index s'effectue dans un bloc réservé, par exemple en fin de document XML, associé à un repère d'index posé à l'endroit auquel l'index fait référence. Le repère est une structure inline, tandis que l'entrée d'index est une structure bloc. Cette approche offre une complète liberté éditoriale, en particulier celle de mettre des commentaires dans l'index, pratique assez courante dans les sciences humaines. Avantage : l'index est enrichi de commentaires. Inconvénient : la saisie est plus coûteuse, car cette méthode alourdit les procédures de contrôle qualité.

Titres courants

Le titre courant est un élément navigationnel du support papier dont la finalité (« savoir où l'on se trouve dans l'ouvrage ») n'est pas toujours respectée : il semble en effet peu utile de rappeler, dans l'édition papier d'un livre professionnel, le titre de l'ouvrage que l'on est en train de lire, tandis que la connaissance du chapitre et surtout de la section courante est très intéressante pour le lecteur.

En édition électronique, la problématique est d'importance : l'épaisseur physique du livre n'est pas là pour nous renseigner sur l'endroit où l'on se trouve : on introduira donc des éléments de navigation dépendants du support cible, par exemple un attribut `short-running-head` sur les titres susceptible de générer un petit système de repérage hypertextuel permettant de très facilement remonter à une section, un chapitre, une partie ou encore la TOC.

- L'utilisation de l'attribut `short-running-head` permettra de générer un mini-système de navigation pour l'édition électronique.

```
<title
id-title="section2_section5_title"
short-toc="1955-1957 : le 1er grand quintet"
short-running-head="1955-1957">
1955-1957 : le premier grand quintet
</title>
```

section title id=section2_section5_title 1955-1957 : le premier grand quintet short-toc : 1955-1957 : le 1er grand quintet short-running-head : 1955-1957 title

Liens hypertextes

Le lien hypertexte est essentiel dans une publication électronique et constitue une différence forte avec la publication papier en permettant d'être relié au reste du monde électronique. Pour le papier, le lien s'exprime par une parenthèse renvoyant à une figure, par exemple, ou par une abréviation latine (cf. pour *confer*) suivie de la cible du lien. En version électronique, les potentialités sont donc plus importantes. Techniquement, un lien nécessite deux informations :

- une information d'identification d'une cible, c'est-à-dire de l'endroit où doit conduire le lien lorsqu'il sera électroniquement activé. Si la cible est interne au document, il faut donc l'indiquer par un « identifiant » unique, généralement inséré au sein d'une « ancre ». Si la cible est externe, il faut indiquer une URL ou une adresse e-mail ;
- une information exprimant où se trouve le départ du lien dans le document. La balise correspondante se nommera par exemple `<link>` et sera munie d'attributs permettant de désigner la cible.

L'insertion d'identifiants n'exige pas toujours l'intervention d'un humain. Pour créer une table des matières interactive, il est nécessaire d'identifier chaque titre servant d'entrée dans la table des matières finale : cela peut se faire par logiciel.

```
<p>... comme on le voit sur la
<link refid-figure="fig15">figure 15</link> il est
nécessaire de procéder à des corrections.</p>
.
.
.
<figure id-figure="fig15">
<figurecaption>Courbe de calibrage de la datation par le
carbone 14</figurecaption></figure>
```

- La liaison entre une figure et sa référence dans le texte. Le lien référence la figure 15 (dont l'identifiant est « fig15 ») placée ailleurs dans le document.

Les entités

Les entités ne sont pas à proprement parler un composant structurel, mais constituent un mécanisme important de la structuration XML. Les entités sont utilisées pour encoder des caractères ou partager des composants XML réutilisables et modifiables a posteriori.

Les entités caractères

Il est possible en XML de représenter les caractères par une combinaison spécifique (et unique pour une DTD) de caractères appelée «entité» et non pas directement sous forme d'un caractère Unicode.

À l'origine du Web, cela était indispensable en raison de la disparité des systèmes de codage de caractères sur les différents ordinateurs : on préférait ainsi écrire `ç` ; pour noter le caractère ç, en laissant à chaque navigateur le soin de faire la correspondance à partir de la police de caractère disponible dans l'ordinateur. Dès lors, il n'était pas nécessaire que le ç soit informatiquement codé de la même manière sur un Mac ou un PC.

Avec Unicode, un seul caractère de n'importe quelle langue est représenté par un numéro unique. Sous réserve de disposer du caractère dessiné (« glyphe ») dans la police utilisée, la représentation est sans équivoque.

ACRONYME **NCR**

Numeric Character Reference

En théorie, il n'est donc plus utile de passer par le système d'entités ; il est toutefois prévu une correspondance numérique entre un caractère et

le nombre Unicode qui le représente sous la forme `&#NNN` ; ou `&#xNNNN` ; . On parle alors de NCR (*Numeric Character Reference*). Chaque caractère est disponible en deux représentations, soit décimale (nombre n'utilisant que les caractères de 0 à 9), soit hexadécimale (nombre n'utilisant que les caractères

de 0 à 9 et de A à F, qui eux-mêmes représentent les nombres décimaux de 10 à 15). On fait référence à chacune des deux représentations en parlant de «NCR décimal» ou de «NCR hexadécimal» que l'on complète, pour les caractères les plus courants des principales langues, par les entités ISO.

Une entité ISO est une entité répertoriée par l'ISO (*International Standard Organization*), donc normalisée dans ses différentes formes. C'est la raison pour laquelle les navigateurs en font l'interprétation de manière automatique, indépendamment du système de codage des caractères de la machine. Le nombre, qu'il soit référencé en décimal ou en hexadécimal, reste bien sûr unique pour un caractère donné. Pour les lettres «a» accentuées, on obtiendra ainsi les représentations regroupées dans le tableau ci-dessous.

À NOTER

Unicode va au-delà des langues en permettant d'encoder des signes diacritiques (signes modifiant les caractères usuels comme le tilde ~ dans le caractère espagnol ñ), des symboles mathématiques, techniques, etc.

Correspondance entre les caractères Unicode représentés par leurs glyphes, les représentations NCR et l'entité ISO. La représentation hexadécimale utilise indifféremment les lettres de a à f en majuscules ou en minuscules.

Caractère Unicode (image ou glyphe)	NCR décimal	NCR hexadécimal	Entité ISO	Mnémonique ISO
à	à ;	à ;	à ;	LATIN SMALL LETTER A WITH GRAVE
á	á ;	á ;	´ ;	LATIN SMALL LETTER A WITH ACUTE
â	â ;	â ;	â ;	LATIN SMALL LETTER A WITH CIRCUMFLEX
ã	ã ;	ã ;	ã ;	LATIN SMALL LETTER A WITH TILDE
ä	ä ;	ä ;	ä ;	LATIN SMALL LETTER A WITH DIAERESIS
å	å ;	å ;	å ;	LATIN SMALL LETTER A WITH RING ABOVE
æ	æ ;	æ ;	æ ;	LATIN SMALL LETTER AE
À	À ;	À ;	À ;	LATIN CAPITAL LETTER A WITH GRAVE
Á	Á ;	Á ;	Á ;	LATIN CAPITAL LETTER A WITH ACUTE
Â	Â ;	Â ;	Â ;	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
Ã	Ã ;	Ã ;	Ã ;	LATIN CAPITAL LETTER A WITH TILDE
Ä	Ä ;	Ä ;	Ä ;	LATIN CAPITAL LETTER A WITH DIAERESIS
Å	Å ;	Å ;	Å ;	LATIN CAPITAL LETTER A WITH RING ABOVE
Æ	Æ ;	Æ ;	Æ ;	LATIN CAPITAL LETTER AE

POUR EN SAVOIR PLUS

La totalité des valeurs Unicode et de leurs correspondances décimale et hexadécimale est disponible sur le site : www.fileformat.info/info/unicode

On notera parmi les avantages des entités ISO, la facilité de mémorisation des caractères.

En pratique, on utilisera quelques entités ISO pour repérer des caractères particuliers à des fins de vérification de l'encodage XML : les

espaces fines et espaces insécables, par exemple, gagneront à être repérées par les entités correspondantes (voir tableau ci-dessous).

L'encodage des espaces sous forme d'entités

Caractère unicode	NCR décimal	NCR hexadécimal	Entité ISO	Mnémonique ISO
espace fine	  ;	  ;	&thinsp ;	THIN SPACE
espace insécable	 ;	 ;	 ;	NO-BREAK SPACE
hair space	  ;	  ;	(aucune entité)	HAIR SPACE

Partage de composants XML

Il est parfois nécessaire de réutiliser un texte ou une portion de texte dans des documents XML différents, une information de copyright par exemple. Le mécanisme de l'entité répond à ce besoin : il s'agit d'un mécanisme du type rechercher/remplacer consistant à utiliser au sein du texte XML un mot encadré des signes & et ;. On écrira donc `©right ;` dans le document XML. À ce mot correspond un texte qui peut d'ailleurs être un fragment de texte XML, à condition que celui-ci soit structurellement conforme à son environnement à l'emplacement d'insertion.

L'association entre l'entité et son contenu se fait au sein de la DTD (voir chapitre 3) ; le contenu n'est restitué que lors de la composition. L'intérêt vient de ce que la mise à jour est globale : s'il faut changer un détail de la mention de copyright, on le fera au niveau de la DTD et la modification sera répercutée dans tous les documents y faisant référence, même les plus anciens.

Métadonnées

Les métadonnées sont essentielles dans l'approche XML pour l'édition : il s'agit d'embarquer dans le document XML des informations complémentaires utiles pour la publication électronique ou l'échange entre systèmes informatiques sous une forme la plus normalisée possible.

Les métadonnées existent déjà pour le papier : on peut par exemple citer l'ISBN qui constitue la « plaque minéralogique » d'un livre, ou encore le type d'ouvrage (roman, essai...) généralement inscrit sur les pages de couverture ou les premières pages. L'informatisation de l'édition a accentué cette démarche par la création de bases de données de gestion ou de référencement (Électre en France) s'appuyant largement sur ces métadonnées. XML l'a généralisée d'un point de vue technique et d'un point de vue semi-normatif avec des taxonomies de métadonnées publiques, en particulier Dublin Core et ONIX.

ACRONYME **ONIX**

ONline Information eXchange

Structure des métadonnées

On distingue deux catégories de métadonnées : celles attachées à une balise au sein d'un document XML et celles concernant le document XML dans son ensemble :

- le nom du fichier d'une image, par exemple, ainsi que ses informations de taille (largeur et hauteur) sont des métadonnées intervenant au niveau de la balise `<img/>` : nom du fichier et taille concernent la donnée mais ne sont pas la donnée elle-même et s'exprimeront respectivement par l'attribut `src` (pour le nom du fichier) et les attributs `width` et `height` pour la taille de l'image. Bien qu'il s'agisse de métadonnées – les informations n'apparaissent pas telles quelles dans le document publié – la pratique a conduit à ne parler que d'attributs ;

Remarque

Il se produit souvent une redondance entre les données balisées et les métadonnées. Ainsi, l'ISBN d'un ouvrage sera souvent identifié dans les préliminaires de l'ouvrage pour le soumettre à un traitement typographique spécifique, en particulier avec des séparateurs tirets, et sera présent dans les métadonnées sous forme normalisée (ex. : sans séparateur).

- les informations d'ISBN, de type de document, d'auteur, d'éditeur, sont attachées au document dans son ensemble et seront rassemblées dans des balises spécifiques appartenant le plus souvent à un jeu de métadonnées répertoriées : Dublin Core ou ONIX par exemple. La pratique a conduit à n'utiliser le terme de « métadonnées » que pour ce type d'informations.

Dublin Core

Dublin Core est le nom d'une taxonomie établie en 1999 par un groupe de travail qui a commencé ses travaux à Dublin (aux États-Unis). Cette taxonomie a pour objet de décrire sous forme de métadonnées les informations liées à la propriété intellectuelle de ressources numériques ou physiques. Elle compte 15 éléments et a été normalisée une première fois en 2003 avant d'être révisée en 2009 par l'ISO (ISO 15836 : 2009).

Liste alphabétique des balises du jeu de métadonnées « Dublin Core ».
Source : BNF et dublicore.org

Balise	Traduction française proposée	Commentaires
dc:contributor	Contributeur	Entité responsable de contributions au contenu de la ressource*
dc:coverage	Couverture	Périmètre ou domaine d'application du contenu de la ressource
dc:creator	Créateur	Entité principalement responsable de la création de la ressource
dc:date	Date	Date d'un événement dans le cycle de vie de la ressource
dc:description	Description	Représentation du contenu de la ressource
dc:format	Format	Manifestation physique ou numérique de la ressource
dc:identifier	Identifiant	Référence univoque à la ressource dans un contexte donné
dc:language	Langue	Langue du contenu de la ressource
dc:publisher	Éditeur	Entité responsable de la mise à disposition de la ressource
dc:relation	Relation	Référence à une ressource apparentée
dc:rights	Déclaration de droit	Informations sur les droits associés à la ressource
dc:source	Source	Référence à une ressource dont la ressource décrite est dérivée
dc:subject	Sujets ou mots-clés	Thème du contenu de la ressource
dc:title	Titre	Nom donné à la ressource
dc:type	Type	Type de la ressource

Cette taxonomie très simple est employée par de nombreuses organisations. Le format epub (pour les e-books) impose d'ailleurs l'utilisation de 4 métadonnées de type Dublin Core. Il est possible de raffiner les éléments : on parle alors de Dublin Core qualifié.

*Le terme ressource désigne de manière très générale l'objet numérique recevant les métadonnées.

ONIX

ONIX est une taxonomie de métadonnées de description d'ouvrages en vue d'échanges entre systèmes informatiques. Elle a été élaborée par un comité international. La première version a été publiée en 2000 ; la version 3, la plus récente, a été publiée en avril 2009. À titre d'information, 80 % de la production des livres aux États-Unis en 2009 utilisent des balises ONIX, ainsi qu'Amazon et Google.

La taxonomie, disponible sous forme de DTD, de schéma ou d'une description Relax NG, dispose de 250 éléments répartis en 26 groupes. Pour en tirer tout le bénéfice, il est préférable de travailler en Schema ou en Relax NG, car les déclarations de balise prévoient en particulier le format attendu pour les données : le code EAN, par exemple, est contraint à 13 caractères numériques, ce qui ne peut être exprimé en DTD.

Contrairement à Dublin Core, ONIX est rarement utilisé au sein du document XML, mais sert plutôt de format d'échange entre systèmes informatiques. On parle d'ailleurs de « message ONIX ». Il est toutefois fort pratique de stocker une partie des balises ONIX dans le document XML, ce qui rend le XML de l'ouvrage entièrement autodocumenté : il n'est plus utile d'aller chercher dans une base de données externe les informations.

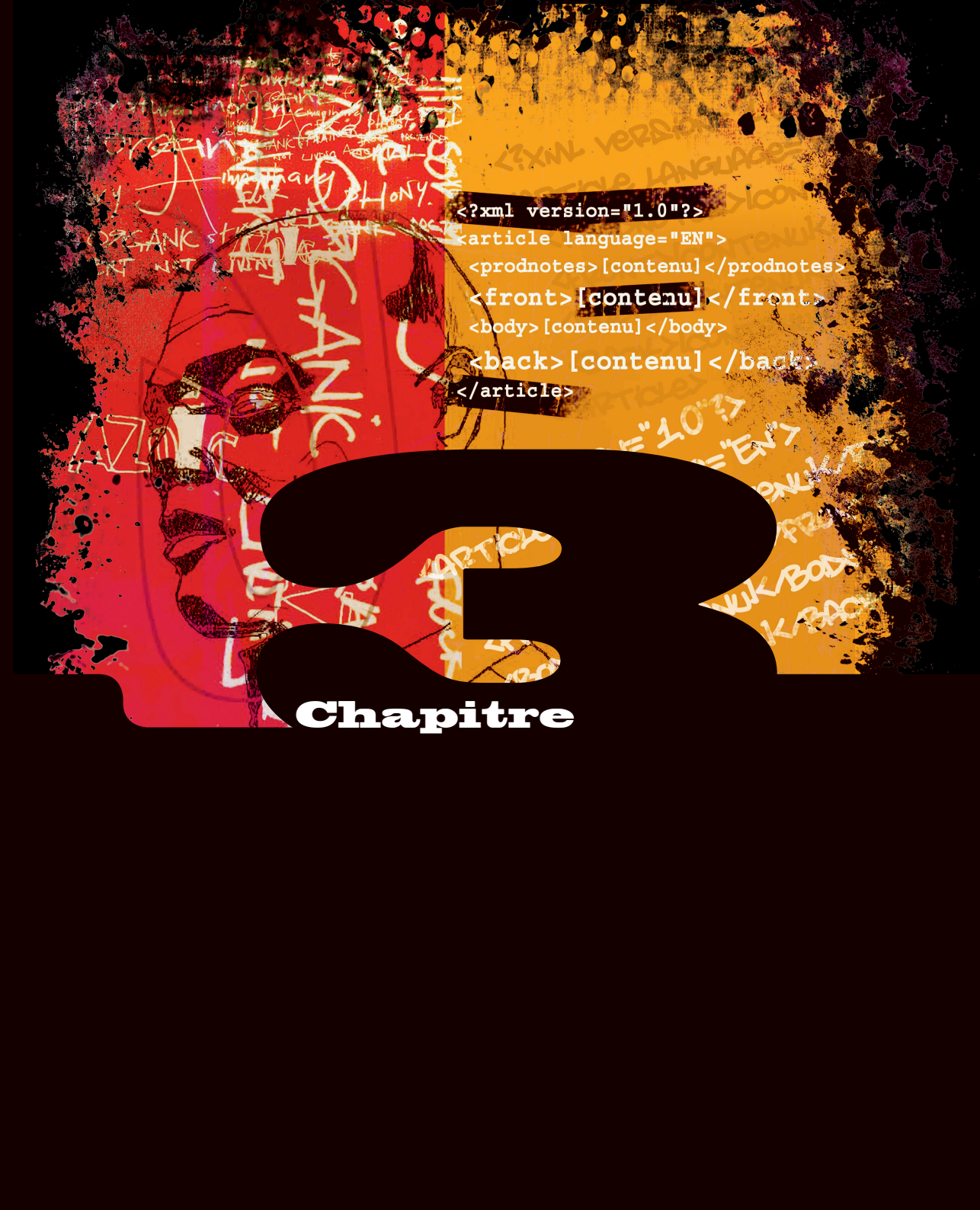
Définir des métadonnées supplémentaires

En fonction des besoins de gestion des documents, on pourra définir des métadonnées supplémentaires n'appartenant à aucune des taxonomies existantes. Ces informations visent un but fonctionnel comme l'attribution de descripteurs (ex. : mots-clés) contrôlés appartenant à un thésaurus ou encore le positionnement du document dans un système de classification hiérarchique, ce qui rend la recherche en principe plus précise.

Des articles scientifiques, par exemple, seront dotés de métadonnées permettant de décrire tous les index et classifications propres au domaine scientifique concerné : un document descriptif d'un médicament recevra ainsi des informations concernant son classement au sein d'un système international de classification (ATC : Anatomique, Thérapeutique, Chimique) et des informations d'indexation contrôlées (mots-clés appartenant à un thésaurus).

En pratique

Chaque éditeur opère une sélection parmi les balises proposées, tout en respectant bien sûr la taxonomie.



```
<?xml version="1.0"?>
<article language="EN">
  <prodnotes>[contenu]</prodnotes>
  <front>[contenu]</front>
  <body>[contenu]</body>
  <back>[contenu]</back>
</article>
```

Chapitre



Écrire/ concevoir une DTD



L'écriture de la DTD est une démarche simple et rapide, une fois la structuration réfléchie (l'architecture du modèle de document). Une DTD efficace est peu compliquée et utilise des noms de balise intuitifs, compréhensibles par tous les acteurs de la chaîne d'édition ; elle doit trouver un compromis entre automatisme et intervention humaine, en restant évolutive et modulaire pour les différentes publications.



Les règles de structures

DTD et langages de définition de taxonomies servent à déclarer les balises et leurs comportements; au-delà du formalisme imposé, c'est un exercice de compromis entre désir de tout contrôler et liberté à laisser aux auteurs et aux acteurs de la chaîne du livre. «*If everything is under control, you are going too slow*» (Mario Andretti).

Qu'est ce qu'une DTD ?

Une DTD (Définition de type de document) est une définition textuelle d'une arborescence documentaire, c'est-à-dire d'une taxonomie, écrite conformément à un certain nombre de règles.

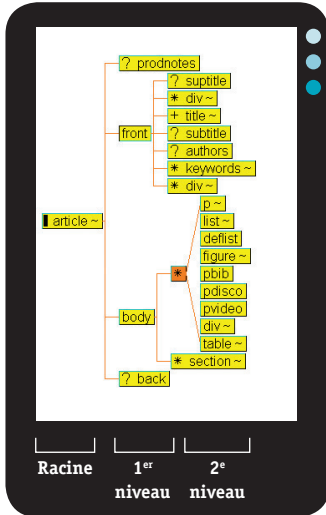
Conventionnellement, et de manière intuitive, une arborescence XML dispose d'une seule racine dont partent des branches multiples qui elles-mêmes se répartissent en d'autres branches, etc. Comme dans le vocabulaire arboricole, on parle de nœud : en première approximation, un nœud est une balise XML. En pratique, un nœud est plus précis qu'une simple balise : il s'agit aussi bien d'une portion de texte entre deux balises correspondant à une paire ouverte et fermante, que d'un attribut ou un commentaire. La notion de nœud prend toute son importance dans le cadre des transformations (voir le chapitre 7).

À NOTER

La DTD n'est pas le seul moyen de définir une taxonomie XML. *XML Schema Description Language* (XSDL) et *Relax NG* offrent aussi ces possibilités avec davantage de fonctionnalités et de précision.

La DTD permet de décrire complètement l'arbre en partant de la racine et en «détaillant» progressivement la structure de chaque branche. Si un document est doté d'un élément racine `<article>` et que cet article

est organisé en 3 sous-ensembles de même niveau (ex. : `front`, `body`, `back`), les 3 premières branches de l'arbre seront respectivement `<front>`,



```

8 <!--*****-->
9 <!--HIGHEST LEVEL ELEMENT-->
10 <!--*****-->
11
12 <ELEMENT article (prodnotes?, front, body, back?) >
13 <ATTLIST article language (FR|EN|SP|GE) #REQUIRED
14 first-published-date CDATA #IMPLIED
15 mp3-file CDATA #IMPLIED
16 >

```

● Représentation sous forme « d'un arbre horizontal » de la structure décrite par article_v1.2.dtd. On n'a représenté ici que 3 niveaux de structures en « dépliant » partiellement les éléments <front> et <body>; on pourrait continuer à déplier l'arborescence pour faire apparaître tous les niveaux possibles.

● Le début de la DTD article_v1.2.dtd correspondant à l'arbre de la figure ci-contre.

<body>, <back>. Chaque sous-branche sera ensuite décrite jusqu'à ce que la totalité de la structure ait été renseignée.

Contrôler la DTD

La DTD permet de définir des balises ainsi que les attributs qu'elles sont susceptibles de recevoir. Pour être valide, la DTD doit respecter des règles strictes d'écriture dont on confie le contrôle à un programme spécifique, le parseur (ou *parser* en anglais).

De son vrai nom « analyseur morphosyntaxique », le parseur est un programme chargé de vérifier que le fichier XML répond bien aux règles d'écriture XML d'une part et que les balises sont utilisées conformément à la DTD d'autre part (quand le fichier XML requiert une DTD).

Tous les parseurs effectuent à peu près les mêmes contrôles, la différence portant sur l'aspect plus ou moins explicite des messages d'erreurs. Le parseur est généralement intégré aux outils de mise au point de DTD et/ou de saisie XML ainsi qu'aux navigateurs. Il est également disponible de manière complètement autonome, mais cela est de plus en plus rare.

ATTENTION

Les parseurs diffèrent par leur degré d'intransigeance vis-à-vis des erreurs ou anomalies de DTD et de la lisibilité des messages d'erreurs. Certains parseurs « corrigent » même au vol les petites erreurs, ce qui est particulièrement handicapant dans un environnement industriel.

Pour l'éditeur Oxygen utilisé dans le cadre du présent ouvrage, le parseur est intégré : on ouvre la DTD avec Oxygen et on lance le contrôle de validité de la déclaration de structure.

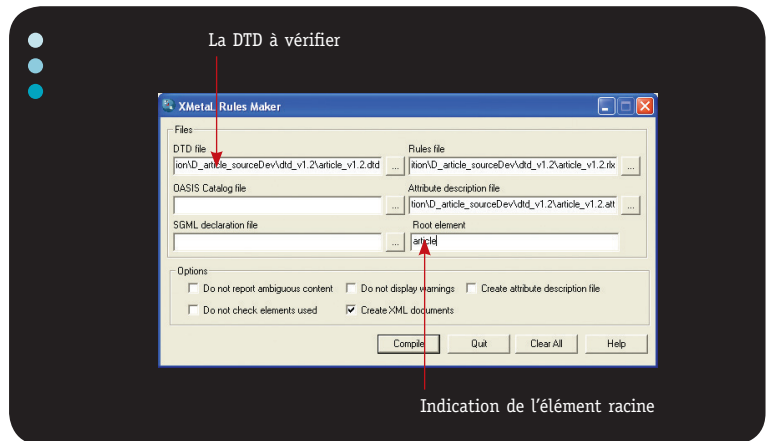
À NOTER

Ce n'est pas parce que la DTD parse qu'elle est satisfaisante : elle peut ne pas faire ce que son auteur attend ou prévoit, ou encore proposer des combinaisons d'éléments tellement nombreuses que le document XML qu'elle permet de bâtir devient imprévisible dans les phases de transformation.

Pour les anciennes versions de l'éditeur XMetal, le parseur est une application séparée (XMetal Rules Maker) qui prend en entrée la DTD et émet des messages d'alerte (*warning*) ou d'erreur le cas échéant. La mise au point d'une DTD étant une démarche

itérative conduisant à sans cesse effectuer des modifications suivies d'un parsing afin de vérifier que la dernière modification n'introduit pas d'erreurs, ce type d'outil est très pratique. La société éditrice de XMetal a toutefois abandonné cette approche (voir figure ci-dessous).

● Le parseur autonome de Xmetal V3 (Xmmkrules.exe). Lors de la compilation, le parseur renvoie des messages plus ou moins explicites.



Définir les balises et leur comportement

Définir une balise

Le mécanisme de déclaration d'une balise dans une DTD s'effectue dans le cadre d'une « ligne de DTD ». Celle-ci s'identifie par son début (juxtaposition de deux signes `< !`) et sa fin (signe `>`), indépendamment du nombre de lignes physiques nécessaires pour écrire les différentes déclarations.

Au sein d'une ligne de DTD, la déclaration d'une balise s'effectue selon un formalisme en 3 parties :

- utilisation du mot-clé : `ELEMENT` (en majuscule) ;
- déclaration de la balise elle-même, c'est-à-dire une simple mention de son nom. Celui-ci doit respecter les contraintes XML déjà mentionnées (chapitre 2) ;
- définition de son modèle de contenu décrit dans une paire de parenthèses, ouvrante et fermante. Ce contenu sera soit uniquement du texte, soit uniquement un jeu d'autres balises, soit un mélange de texte et de balises qui elles-mêmes auront été déclarées quelque part dans la DTD.

La déclaration :

```
<!ELEMENT article (prodnotes?, front, body, back>
```

traduite en français courant devient :

- introduction d'une nouvelle ligne de DTD (`< !`) ;
- déclaration d'un « élément » `article` dont le contenu est `prodnotes`, suivi de `front`, puis de `body` et enfin de `back` ;
- fin de la ligne de DTD (`>`).

La déclaration d'une structure complète nécessite de décrire successivement chaque niveau, jusqu'aux niveaux terminaux qui ne peuvent contenir que du texte.

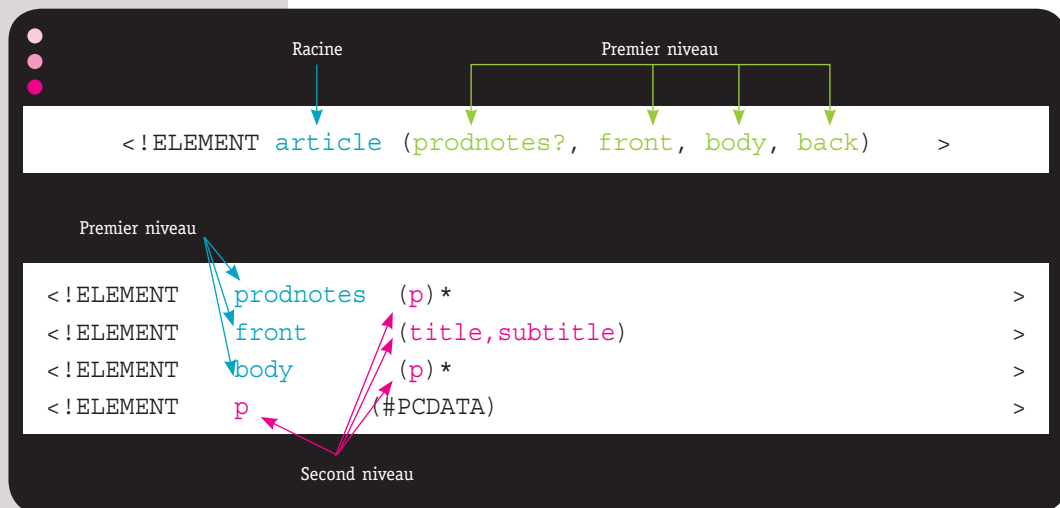
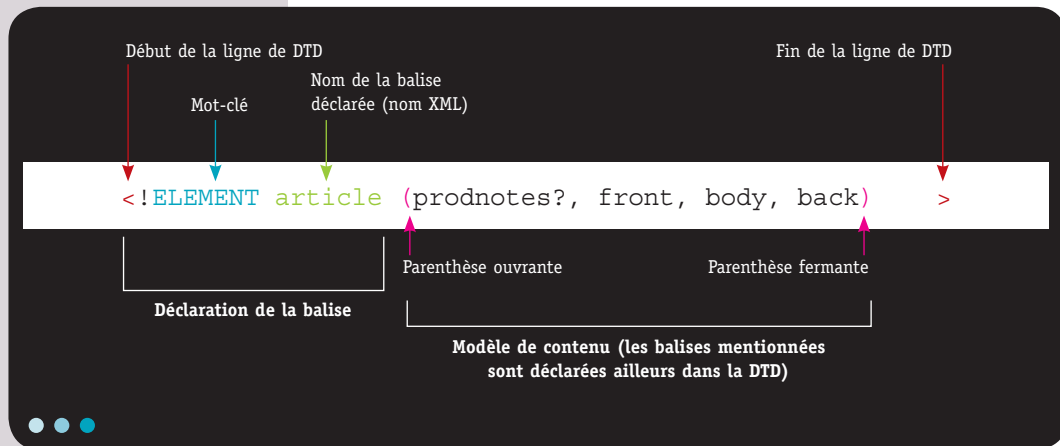
Il est à noter que le signe chevron (`<` ou `>`) n'est pas utilisé dans la DTD de la même façon que dans le document XML pour repérer les balises : un choix peu heureux lié au souci des concepteurs de SGML (l'ancêtre de XML) de ne pas trop « consommer » de caractères spécifiques et de ne pas heurter les habitudes des informaticiens... Pour le novice, c'est un effort supplémentaire à consentir que de retenir à tout moment le contexte dans lequel il se trouve (document XML ou bien DTD).

Par ailleurs, en termes de vocabulaire, on utilisera presque toujours le mot « élément » pour désigner la balise et son contenu dans le document XML :

● Morphologie d'une ligne de DTD ; elle se lit de gauche à droite. La ligne de DTD peut se poursuivre sur plusieurs lignes physiques.

on parlera donc de l'élément `article` pour désigner la balise `<article>` et son contenu.

Dans la DTD, le contenu que peut recevoir la balise, une fois utilisée dans le document XML, s'appelle un « modèle de contenu » et s'inscrit entre parenthèses après la déclaration du nom de la balise.



● Déclaration d'une structure complète limitée au premier niveau. Chaque déclaration de balise n'est suivie que du modèle de contenu du niveau immédiatement inférieur. Les 4 éléments `prodnotes`, `front`, `body` et `back` sont au même niveau. `p` est au niveau immédiatement inférieur.

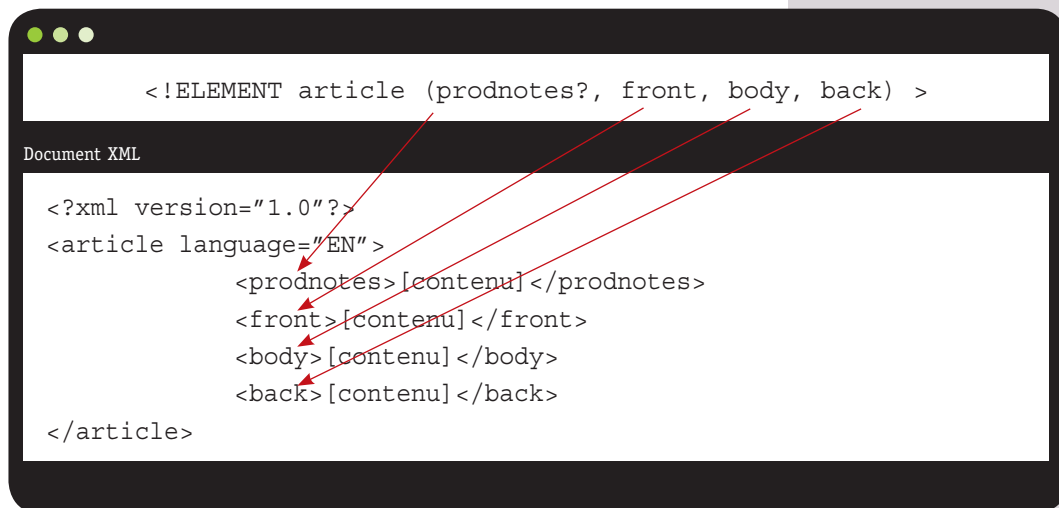
Correspondance DTD - document XML

Les premiers pas en XML sont un peu difficiles en raison de la similitude des informations exprimées dans la DTD et des balises utilisées dans le document XML.

La DTD se limite à la déclaration des balises et de leur contenu au niveau immédiatement inférieur jusqu'à épuisement des sous-structures, c'est-à-dire jusqu'au niveau où il ne reste que du caractère. La totalité des balises est déclarée. Le document XML, lui, utilise les balises en tenant compte de ce qui a été déclaré : ce qui est obligatoire doit être présent, tandis que les éléments facultatifs ne sont utilisés qu'en cas de nécessité.

Dès lors, on mélange facilement la DTD et le document XML, surtout si l'on utilise le même outil logiciel (ex. : Oxygen) pour mettre au point la première et effectuer une saisie XML. C'est la raison pour laquelle dans la suite de l'ouvrage, nous utiliserons de préférence trois outils différents :

- un éditeur de texte pour écrire la DTD ;
- un compilateur pour vérifier la conformité de la DTD à ses règles d'écriture ;
- un éditeur XML pour saisir le document XML.



● Correspondance entre la déclaration des balises et les éléments du document XML

Les connecteurs

L'écriture de DTD fait appel à deux caractères particuliers au sein du modèle de contenu (i.e. ce qui se trouve dans la parenthèse à droite du nom de la balise déclarée) pour spécifier soit la succession de l'ordre des balises, soit leur apparition exclusive dans le modèle. Ces signes, connus sous le nom de «connecteurs», sont respectivement la virgule et la barre verticale (la barre verticale de la touche 6 des claviers, généralement représentée par... deux petites barres l'une au-dessus de l'autre).

,	«ET» ordonné	(a,b) dans la DTD, signifie <a> suivi de dans le document XML.
	«OU» exclusif	(a b) dans la DTD, signifie soit <a>, soit dans le document XML.

- La virgule sert à exprimer que les éléments se suivent dans l'ordre indiqué. `<!ELEMENT article (front, body, back)>` dans la DTD, signifie que, dans le document XML, la balise `<article>` contiendra en premier la balise `<front>` (puis sa fermeture `</front>`) suivie de la balise `<body>` (et de sa fermeture) pour terminer par la balise `<back>` (et sa fermeture).
- La barre verticale sert à exprimer que les éléments sont exclusifs l'un de l'autre. `<!ELEMENT chapter (section|div)>` dans la DTD signifie que, dans le document XML, la balise `<chapter>` ne pourra recevoir qu'une

section ou qu'une div, exclusivement l'une de l'autre (en d'autres termes c'est soit l'un soit l'autre). En pratique, cette barre verticale est utilisée en même temps qu'un symbole d'occurrence (signe «*» ou signe «+») pour signifier le mélange d'éléments. Un peu difficile à comprendre au premier abord...

REMARQUE

Les séparateurs espaces n'exigent pas d'être uniques : il faut simplement qu'il y ait au moins une espace pour séparer des informations significatives dans la DTD (par exemple entre le mot-clé `ELEMENT` et le nom de la balise). En pratique, on se servira des espaces pour présenter proprement une DTD afin de la rendre plus lisible.

Les symboles d'occurrence

Les balises sont susceptibles d'être soumises à des règles d'apparition (obligatoire, facultatif) et à des règles de fréquence (unique, multiple). À cette fin, le formalisme DTD utilise 4 signes distinctifs, appelés «symboles d'occurrence» placés directement après le nom (et accolés à celui-ci) de la balise dans le modèle de contenu :

- absence de symbole : la balise correspondante dans le document XML est obligatoire unique;

- signe + : la balise, dans le document XML, est obligatoire répétable;
- signe ? : la balise est facultative unique;
- signe * : la balise est facultative répétable.

À NOTER

Le symbole d'occurrence est accolé à la droite du nom de la balise dans le modèle de contenu. Il n'y a pas d'espace entre le nom de la balise et le symbole d'occurrence.

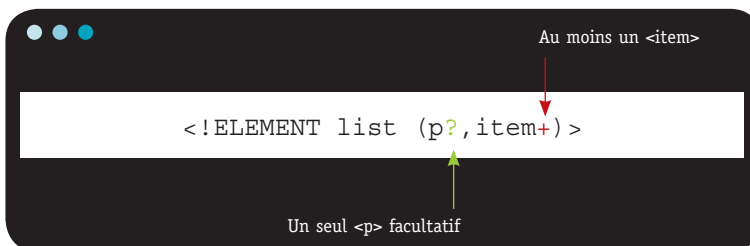
Les symboles d'occurrence et leur signification

Symbole	Signification	Exemple
	Obligatoire unique	(titre, p) dans la DTD signifie titre unique et obligatoire suivi de p unique et obligatoire dans le document XML.
+	Obligatoire répétable	p+ signifie au moins 1 balise p dans le document XML.
?	Facultatif unique	p? signifie 0 ou 1 balise p dans le document XML.
*	Facultatif répétable	p* signifie 0, 1 ou plus de balises p dans le document XML.

Comprendre la notation (a|b)* : mélange de a et de b

Notation dans la DTD	Possibilités dans le document XML
(a b)	Soit <a> unique, soit unique.
(a b)*	0, 1 ou plusieurs (a b) qui se suivent. Combinaisons possibles : (rien) <a> seul seul <a> suivi de <a> : <a> <a> <a> suivi de : <a> suivi de : suivi de <a> : <a> etc. Il s'agit donc d'un mélange d'éléments a et b.

Combinés aux connecteurs (, |), les symboles d'occurrence (+? *) permettent de décrire à peu près n'importe quelle structure, en particulier le mélange d'éléments. Dans la DTD (authorfirstname|authorlastname)* exprime que dans le document XML, la balise <author> contient un mélange de balises <authorfirstname> et <authorlastname>. C'est un peu difficile à comprendre mais clair après réflexion.



- Indication du comportement des balises en termes d'apparition et de fréquence au sein d'un modèle de contenu : la liste (élément list) peut contenir un paragraphe introductif unique (signe ? accolé à p), suivi d'au moins un item (signe + accolé à item). On remarquera que les symboles d'occurrence sont placés dans la DTD directement à droite du nom des balises du modèle de contenu.

Attention

Pas d'espace entre < et ! en début de ligne de DTD. Pas d'espace non plus entre < ! et le mot-clé ELEMENT. En revanche, on peut mettre tous les espaces et retours chariot que l'on souhaite dans et hors la parenthèse du modèle de contenu.

UN ENSEMBLE DE MOTS-CLÉS

Le formalisme DTD comporte différents mots-clés dont l'usage est très précis, comme ELEMENT, ATTLIST, CDATA, etc., dont on rencontrera quelques exemples dans la suite de l'ouvrage.

Mélange de texte et de balises

La DTD utilise un mot-clé pour indiquer qu'un contenu est de nature texte, c'est-à-dire qu'il ne contient que des lettres et des chiffres. Il s'agit de #PCDATA. Dire que la balise <sup>, balise utilisée par exemple pour exprimer un exposant dans le document XML, ne contient que du texte s'écrit dans la DTD :

```
<!ELEMENT sup (#PCDATA)>
```

On notera au passage qu'il n'est pas utile de mettre un symbole d'occurrence (* ou +) après la parenthèse entourant le mot-clé #PCDATA. Certains parseurs, d'ailleurs, le refusent.

L'utilisation du mot-clé #PCDATA est très souvent couplée à celle du connecteur | pour exprimer un contenu mixte, c'est-à-dire un contenu pouvant avoir du texte et des balises. Dire que <i> peut contenir un mélange de texte et de balises <sub> et <sup> s'écrit dans le formalisme DTD :

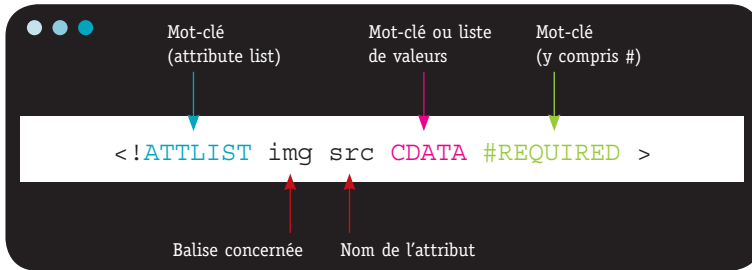
```
<!ELEMENT i (#PCDATA|sub|sup)*>
```

À noter : le mot-clé #PCDATA doit apparaître en premier dans un modèle de contenu. Écrire dans la DTD (sub|sup|#PCDATA)* provoque une erreur au niveau du parseur.

Décrire les attributs

Le mécanisme de déclaration des attributs au sein d'une DTD est un peu moins intuitif que celui utilisé pour les balises, mais il reste simple. La déclaration s'effectue au sein d'une ligne de la DTD (donc commençant par <! et se terminant par >) selon un formalisme en cinq parties :

- utilisation du mot-clé ATTLIST (en capitales, mnémonique de *attribute list*) pour signifier que l'on déclare un ou plusieurs attributs;
- rappel du nom de l'élément (i.e. la balise) concerné par les attributs en question;
- déclaration du nom de l'attribut (il doit respecter les règles des noms XML);
- déclaration des valeurs que peut prendre l'attribut. Ces valeurs répondent à des règles précises qui doivent être respectées pour que la DTD soit validée par un parseur. Ce sera soit un mot-clé (ex. : CDATA), soit une énumération (i.e. une liste de valeurs autorisées);
- déclaration de la qualité obligatoire ou facultative de l'attribut, à l'aide des mots-clés #REQUIRED et #IMPLIED. Notez que cette dernière déclaration peut être remplacée par une valeur par défaut lorsque l'attribut ne peut recevoir qu'un nombre fini de valeurs.



Règles de comportement des attributs : obligatoire vs facultatif

Chaque attribut dans le document XML est, à la manière d'une balise, obligatoire ou facultatif. Cet état s'exprime dans la DTD à l'aide des mots-clés :

- **#REQUIRED** pour les attributs obligatoires ;
- **#IMPLIED** pour les attributs facultatifs. Il faut bien reconnaître que le mot **IMPLIED** est peu parlant, contrairement à **REQUIRED**. À retenir tel quel donc, sans attache avec sa signification.

En termes d'usage, l'attribut obligatoire apparaîtra souvent au sein d'un élément vide, par exemple une balise spécifiant une image, tandis que les attributs facultatifs répondront à des besoins plus fins de balisage.

À noter :

- un attribut déclaré obligatoire dans la DTD et manquant dans le document XML provoquera une erreur au niveau du parseur ;
- les valeurs d'attributs ne peuvent pas comporter de balises. Il est ainsi faux d'écrire : `<img alt="photo de <b>Miles Davis</b> au concert de Prague en 1987">`.

Types d'attributs

XML prévoit différents types d'attributs afin d'apporter suffisamment de souplesse dans le comportement des balises. À chaque type correspond un formalisme spécifique.

- Lorsque la valeur de l'attribut est une chaîne de caractères, espaces compris, de longueur quelconque, on utilise le mot-clé **CDATA** (mnémonique de *character data*). On utilisera ce type d'attribut pour spécifier un texte de substitution décrivant une image à l'intention des non-voyants, par exemple (ex. : `alt="photo de Miles Davis au concert de Prague en 1987"`), mais aussi pour référencer correctement l'image dans un moteur de recherche, une fois la publication effectuée sur Internet.

● Déclaration d'un attribut `src` sur l'élément `img` ; il admettra comme contenu une chaîne de caractères (mot-clé **CDATA**). L'attribut `src` est obligatoire au sein de la balise `img` (mot-clé **#REQUIRED**).

Mot-clé : #FIXED

Il existe aussi le mot-clé **#FIXED** pour exprimer un attribut obligatoire de valeur prédéterminée (fixe). Il est rarement utilisé, car il est souvent possible d'exprimer les mêmes caractéristiques avec **#REQUIRED**.

Attention

Ne pas confondre **#PCDATA**, utilisé pour déclarer des modèles de contenu texte d'une balise, et **CDATA**, utilisé pour spécifier un type d'attribut.

● Déclaration d'attribut de type CDATA pour préciser le nom d'un fichier image et utilisation dans le document XML

Remarque

On peut déclarer des attributs faisant référence à plusieurs identifiants en les séparant par des virgules. La déclaration correspondante dans la DTD s'écrit à l'aide du mot-clé IDREFS (même chose que IDREF, mais avec un «s» à la fin). Ce type est moins souvent utilisé.

● Déclaration d'une énumération et utilisation dans le document XML

DTD

```
<!ATTLIST img src CDATA #REQUIRED >
```

Document XML

```
<?xml version="1.0"?>
<article language="FR">
  [contenu]
  <p>......</p>
  [contenu]
</article>
```

- Lorsque la valeur de l'attribut peut prendre différentes valeurs prédéfinies (on parle d'énumération), on listera toutes les valeurs possibles au sein d'une parenthèse, en séparant les valeurs par le caractère | (voir figure ci-dessous). Il est à noter que les valeurs ne doivent pas contenir d'espaces (contrairement à la chaîne de caractères déclarée par CDATA), ne pas commencer par un chiffre, etc., c'est-à-dire être des noms XML.
- Lorsque la valeur de l'attribut doit être unique dans TOUT le document, on parle «d'identifiant». Il doit être un nom XML, et donc, en particulier, ne pas commencer par un chiffre. Le mot-clé correspondant est ID en capitales. Lors du parsing du document XML, toute duplication d'identifiant au sein du document provoquera une erreur. Mais attention : ID utilisé dans la DTD étant un mot-clé, il est écrit en majuscules. Beaucoup d'auteurs de

DTD

```
<!ELEMENT article (prodnotes?, front, body, back?) >
<!ATTLIST article language (FR|EN|SP|GE) #REQUIRED >
```

Document XML

```
<?xml version="1.0"?>
<article language="EN">
  <prodnotes>[contenu]</prodnotes>
  <front>[contenu]</front>
  <body>[contenu]</body>
  <back>[contenu]</back>
</article>
```


DTD utilisent le mot `id` (en minuscules) pour désigner un attribut de type identifiant qui reçoit donc la valeur `ID` (en majuscules) dans la DTD. Il est facile de se perdre et il vaut mieux choisir des noms d'attributs explicites, comme `id-figure`, `id-tableau`... quitte à être un peu redondant.

- Lorsque la valeur de l'attribut fait référence à un identifiant (i.e. une valeur d'attribut unique dans tout le document), on utilise le mot-clé `IDREF` (mnémonique de «référence à un identifiant»).

● Déclaration d'un identifiant et d'une référence à un identifiant. L'utilisation correspondante dans le document XML ne sera valide que si le choix des identifiants, lors de la saisie, correspond aux contraintes XML (i.e. ce sont nécessairement des noms XML). Nous avons appelé l'identifiant `id-figure` (type `ID`, donc) et sa référence `refid-figure` (type `IDREF`) pour bien différencier leur usage.

DTD

```
<!ELEMENT figure (img,figurecaption?) >
<!ATTLIST figure id-figure ID #IMPLIED>
```

Document XML

```
<?xml version="1.0"?>
<article language="FR">
  [contenu]
  <figure id-figure="fig004"> [contenu] </figure>
  [contenu]
</article>
```

Unique dans le document XML

DTD

```
<!ELEMENT link (#PCDATA) >
<!ATTLIST link refid-figure IDREF #IMPLIED>
```

Document XML

```
<?xml version="1.0"?>
<article language="FR">
  [contenu]
  Voir la <link refid-figure="fig004">figure
  4</figure>
  [contenu]
</article>
```

Doit nécessairement exister sous forme d'identifiant dans le document XML

Écrire et organiser la DTD

TÉLÉCHARGER NOTEPAD

<http://notepad-plus.sourceforge.net>

L'écriture d'une DTD n'exige aucun outil spécifique : un éditeur de texte standard suffit. Pour notre part, nous utili-

sons Notepad ++, éditeur gratuit aux puissantes fonctionnalités et très simple à utiliser.

Au niveau de l'écriture des DTD (ou des autres langages de description de taxonomies), il est possible d'utiliser des couleurs afin de mettre en évidence les mots-clés, les caractères spéciaux, etc. Au niveau de la manipulation des fichiers XML, Notepad offre des fonctionnalités XML multiples, dont la coloration des balises et attributs (voir la figure suivante), la possibilité de nettoyer le flux XML des doubles espaces et retours chariots parasites (reportez-vous au chapitre 4 sur la saisie). Ces fonctionnalités XML avancées de Notepad nécessitent de télécharger un plug-in spécifique (XML Tools) et de l'installer.

Une alternative consiste à utiliser un éditeur XML polyvalent; on parle alors «d'environnement de développement intégré» (IDE : *Integrated Development Environment*). Un seul outil offre alors toutes les fonctions :

- éditeur et parseur de DTD ;
- éditeur de document XML ;
- outil de lancement des transformations, etc.

Il faut de l'expérience pour ne pas se perdre dans ce type d'outil. Aussi, dans le cadre du présent ouvrage, utilisons-nous trois outils :

- Notepad pour éditer la DTD ;
- XMmkrules pour vérifier la DTD ;
- XMetaL ou Oxygen pour saisir le document XML.

Organiser la DTD

Une DTD sert à décrire un arbre et est interprétée par les outils XML dans sa globalité : il n'y a donc pas d'ordre imposé pour écrire les différentes déclarations, à l'exception parfois de l'élément racine que certains outils exigent de voir apparaître en premier.

Cela dit, il est préférable d'organiser correctement sa DTD pour la rendre lisible directement, sans avoir à passer par des outils de visualisation de structure. On l'écrira donc de manière logique et organisée en adoptant l'ordre suivant :

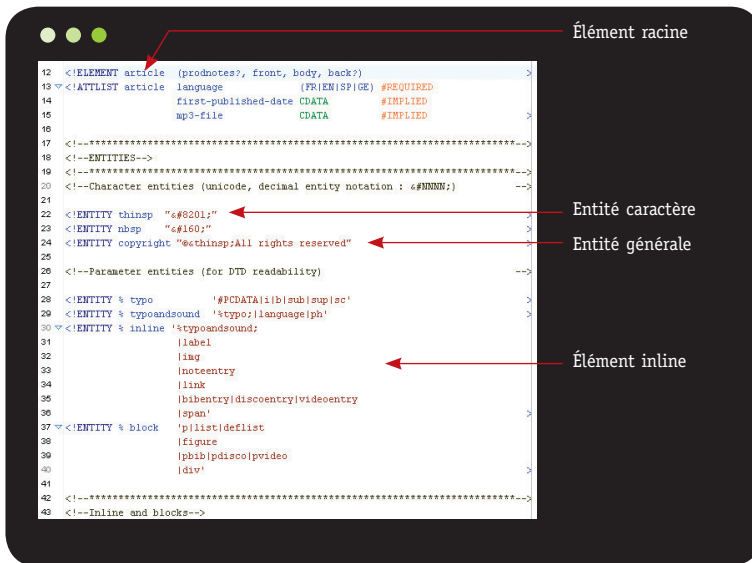
- déclaration de l'élément racine ;
- déclaration des entités ;

L'éditeur de texte

Un éditeur de texte est un programme permettant de manipuler du texte et de sauvegarder un fichier texte le plus élémentaire possible : des caractères et des espaces ou retours chariot les uns derrière les autres. Il ne s'agit pas de faire une mise en pages, domaine réservé aux traitements de texte ou aux outils de PAO. En revanche, les éditeurs de texte disposent d'outils d'aide à la rédaction (appariement de balises, etc.) et d'aide à la visualisation, en particulier la coloration syntaxique (pour la mise en valeur des balises dans le cas de XML) ainsi que l'indentation pour la mise en valeur des hiérarchies d'éléments.

- déclaration des éléments inline;
- déclaration des éléments blocs;
- déclaration des éléments hiérarchiques.

On s'efforcera, au sein de chaque partie, d'être logique et de déclarer les éléments dans l'ordre où ils apparaissent dans la hiérarchie. Cette façon d'opérer a bien sûr une limite, puisqu'en XML les éléments sont largement réutilisés (on trouvera des balises <title> à de nombreux endroits par exemple) et qu'un élément ne doit être déclaré qu'une seule fois dans la DTD.



● Organisation de la DTD en grandes parties commentées pour améliorer la lisibilité. On remarquera la coloration syntaxique qui n'a pas pour objectif l'esthétique, mais la mise en évidence des composants du langage.

Quelle langue d'usage pour les balises ?

On utilisera de préférence l'anglais pour spécifier les noms de balise, afin de les rendre compréhensibles aux ateliers offshore dont l'anglais est la seule langue commune. Un opérateur indien comprendra mieux le nom de balise <front> que celui de <prelim>... Par ailleurs, on choisira des noms de balises dont la signification est déjà connue au sein de taxonomies très utilisées, HTML en premier lieu. Un paragraphe sera de préférence appelé «p», même si des DTD largement répandues utilisent «para». On puisera également des noms signifiants de balises dans les grandes DTD publiques (TEI et DocBook) : l pour désigner une ligne de vers au sein d'une strophe par exemple.

Modulariser et commenter la DTD

Une bonne hygiène d'écriture de DTD passe par trois principes :

- modularisation;
- factorisation;
- commentaires.

Modularisation. L'écriture d'une DTD n'exige pas que celle-ci soit constituée d'un seul fichier. Il est possible d'appeler, depuis une DTD principale, des DTD satellites sous forme de modules : on parle alors de «DTD modulaire». On organisera mieux le partage de structures en isolant les grands composants. Dans une maison d'édition scolaire, on envisagera ainsi de créer et maintenir une DTD commune «exercices» qui sera appelée par toutes les DTD spéciali-

sées sur telle ou telle collection au lieu de recréer, pour chaque ouvrage ou collection, une DTD «exercices» spécifique plus ou moins compatible avec sa voisine.

La syntaxe d'appel du module externe étant un peu compliquée, nous ne l'expliquerons pas dans cet ouvrage.

Factorisation des modèles de contenu. Il est évident que des modèles de contenu partiels ou complets peuvent apparaître à différents endroits. Un mélange de texte de `<i>` et de `<b>`, par exemple, peut apparaître dans un paragraphe standard, un paragraphe bibliographique, une balise `<author>`...

Cependant, répéter chaque fois les balises dans les modèles de contenu où elles doivent apparaître risque de rendre la DTD peu lisible. On utilise donc le concept «d'entité paramètre» qui permet de mettre en facteur commun ce qui va être réutilisé dans la DTD. Une entité paramètre, dont le nom doit suivre les règles de constitution des noms XML, va correspondre dans la DTD, le plus souvent, à un mélange de caractères et de balises. Il suffira ensuite d'utiliser cette entité au sein de la DTD pour que le modèle correspondant soit utilisé. On choisit généralement un nom représentatif de ce que l'entité paramètre remplace : `typo` pour le texte pur et les balises typographiques, `inline` pour les structures disponibles en ligne (par exemple au sein d'un paragraphe ou d'un titre), `block` pour les structure de type bloc.

Attention : dans la DTD, l'entité paramètre se déclare différemment de l'entité caractère ou de l'entité générale par l'insertion du signe % dans la déclaration entre le nom de l'entité et son modèle de contenu. On écrira ainsi :

```
<!ENTITY % typo "#PCDATA|i|b|sub|sup|sc">
```

Son usage, dans la DTD, fait aussi appel au caractère % sous la forme classique d'une entité :

```
<!ELEMENT author (%typo;)*>
```

Attention

L'entité paramètre ne s'utilise que dans la DTD sous la forme du nom de l'entité précédé du signe % et terminé par ;. En cela, elle diffère des autres entités qui ne s'utilisent que dans le XML sous la forme du nom de l'entité précédé de & et terminé par ;. On notera aussi qu'une entité paramètre peut être appelée par une autre entité paramètre.

● Exemple d'entités paramètres utilisées pour rassembler sous un même nom les constituants d'un modèle de contenu

```
26 <!--Parameter entities (for DTD readability) -->
27
28 <!ENTITY % typo "#PCDATA|i|b|sub|sup|sc" >
29 <!ENTITY % typoandsound " %typo;|language|ph" >
30 <!ENTITY % inline " %typoandsound;|label" >
31
32 |liag
33 |noteentry
34 |link
35 |bibentry|discoentry|videoentry
36 |span" >
37 <!ENTITY % block " %inline;|p|list|deflist" >
38 |figure
39 |pbib|pdisco|pvideo
40 |div" >
41
```

`typo` regroupe un mélange de texte et de balises `i`, `b`, `sub`, `sup`, `pc`.

`typoandsound` regroupe `typo` et les balises `language` et `ph`.

`inline` regroupe `typoandsound` + les balises `label`, `langue`, `lien`, etc.

`block` regroupe les balises `p`, `list`, `deflist`, etc.

Commentaires. Il est utile de commenter la DTD afin de la rendre lisible pour d'autres utilisateurs, ou pour soi-même si l'on est amené à la modifier par la suite sans avoir régulièrement travaillé dessus. Un commentaire s'écrit à l'aide des ensembles de signes : `<!--` pour le début du commentaire et `-->` pour la fin du commentaire. Il est possible d'écrire ce que l'on veut à l'intérieur de ces deux signes, sauf un commentaire à l'intérieur d'un commentaire, erreur classique de débutant qui se traduit par des messages du parseur assez difficiles à interpréter.

À NOTER

Pas d'espace entre `<!--` et `--` en début de commentaire et pas d'espace entre `--` et `>` en fin de commentaire.

La même syntaxe (`<!-- -->`) est utilisée pour mettre des commentaires au sein d'un document XML.

Les commentaires doivent par ailleurs être écrits à l'extérieur d'une ligne de DTD : on ne peut commenter à l'intérieur d'une déclaration de balise. Ainsi, le commentaire suivant est incorrect :

```
<!ELEMENT article (prodnotes?, front, body, back)
<!-- definition de l'article -->
```

Il faut écrire :

```
<!-- definition de l'article-->
<!ELEMENT article (prodnotes?, front, body, back)>
```

Mettre au point et tester la DTD

La mise au point de la DTD s'accompagne d'un document XML de test appelé « monstre ». Un monstre sert à tester la plupart des combinaisons existantes (ex. : repérées lors de l'analyse documentaire) ou envisageables avec pour objectif :

- au niveau de la saisie, de répondre à la question « Est-il possible avec la DTD, telle qu'elle est écrite, de saisir telle ou telle information ? » ;
- au niveau des transformations (voir le chapitre 7), de vérifier que les objets de publication (numériques en particulier) sont viables et conformes à la forme et au comportement voulus.

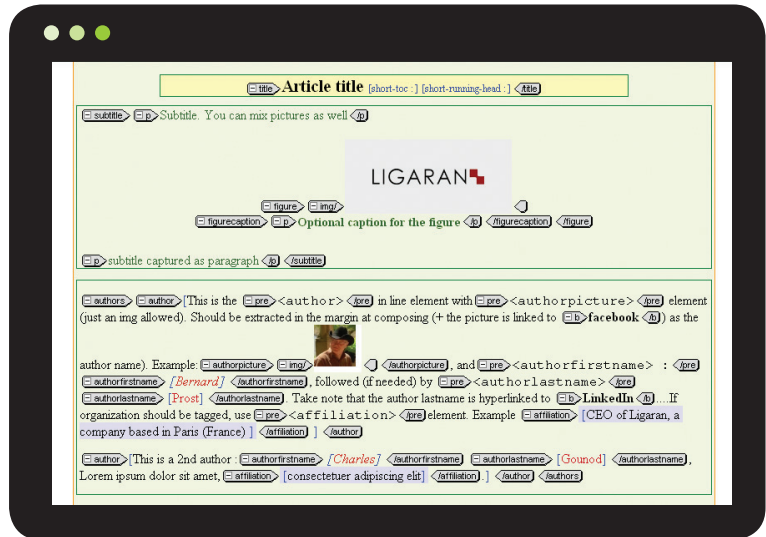
Commenter la DTD

Veillez à ne pas trop commenter à l'intérieur de la DTD, au risque de la rendre difficile à lire : dans ce cas, préférez un commentaire externe sous forme d'un module appelé par la DTD principale.

Anecdote

Le terme « monstre » vient de la musique classique : lors de l'écriture d'une symphonie, les compositeurs ont l'habitude de produire une version réduite pour piano qu'ils appellent « monstre ».

● Monstre (article_monster.xml) destiné à la validation de la DTD article_v1.2.dtd. Ce monstre servira aux tests de conversion vers les supports cibles.



Le monstre XML doit également reprendre toutes les entités caractères et entités générales déclarées dans la DTD, afin de vérifier leur apparence dans une police Unicode de référence. L'usage veut que l'on en fasse un tableau à placer dans la structure : «vrai» tableau si la DTD le permet, liste de paragraphes si la DTD ne l'a pas prévu (c'est le cas de article_v1.2.dtd).

Ce monstre servira également aux ateliers de saisie pour illustrer la saisie de cas particuliers.

Si la DTD est complexe ou la structure très grosse, on spécialisera éventuellement les monstres en fonction des structures à contrôler. Un éditeur scolaire, par exemple, construira trois monstres :

- un monstre pour la structure globale (ex. : monster_structure.xml);
- un monstre pour les tableaux, en reprenant tous les cas intéressants de contenus de cellules à problème (ex. : monster_tables.xml);
- un monstre pour les exercices (ex. : monster_exercises.xml).

Maintenir la DTD

Il ne faut pas rêver : la DTD ne sera jamais totalement stable et des modifications devront être apportées au cours du temps, soit parce que des cas de figure ont été oubliés, soit parce qu'il faut s'adapter à de nouveaux supports ou à une évolution de ceux-ci. Avant toute modification, on gardera à l'esprit une règle d'hygiène : modifier la DTD le moins possible.

Les modifications de DTD se répartissent en deux catégories :

- soit la modification rend la nouvelle DTD incompatible avec l'ancienne. C'est le cas lors de l'ajout d'un nouvel élément ou d'un attribut obligatoire sur un élément obligatoire existant : tout ancien document générera une erreur lors du contrôle vis-à-vis de la nouvelle DTD. Ce cas est à éviter autant que possible ;
- soit la modification laisse la nouvelle DTD compatible avec l'ancienne. C'est le cas de l'ajout d'un nouvel élément ou d'un nouvel attribut facultatif : tous les documents, qu'ils soient nouveaux ou déjà saisis, parseront correctement avec la nouvelle DTD, tandis que les nouveaux documents ne parseront pas toujours avec l'ancienne DTD. Par abus de langage, on dit que « les DTD sont compatibles ».

Versionner la DTD. Lorsque la nouvelle DTD est incompatible avec l'ancienne, les nouveaux documents peuvent ne pas être valides avec l'ancienne DTD. Dans ce cas, l'usage recommande de changer le nom de la DTD et de lui appliquer une convention de nommage tenant compte de ces modifications, par exemple en suffixant le nom de la DTD d'un v1.1 pour des modifications mineures ou d'un v2.0 si les modifications sont jugées majeures.

Dans le cas où la nouvelle DTD est compatible avec l'ancienne, on ne change pas le nom de la DTD, mais il est parfois souhaitable de conserver une trace de ces modifications, à la fois au sein de la DTD et dans les documents XML. La méthode habituelle consiste à prévoir un attribut, de préférence obligatoire, (ex. : DTDversion) sur l'élément racine que l'on renseignera lors de la production.

`<article DTDversion="1.04">` indique ainsi que le document XML a été saisi avec la 4^e modification compatible de la DTD v1. Cette information est généralement posée par les systèmes de production et n'introduit pas de charge de travail supplémentaire pour les opérateurs.

Décider d'une modification de DTD. L'intuition suggère de choisir une modification de DTD qui préserve le passé : en rendant facultatif un élément existant, en introduisant de nouveaux éléments optionnels ou encore en autorisant toujours plus de mélanges d'éléments existants.

À la limite, et c'est tentant, on pourrait autoriser n'importe quel élément dans n'importe quel élément. Cela aurait un coût : la DTD deviendrait plus laxiste et le document XML serait de moins en moins prévisible dans ses transformations en objets éditoriaux numériques.

DTD compatible ?

Une (nouvelle) DTD est compatible avec une (ancienne) DTD si tout document XML saisi avec l'ancienne DTD parse également avec la nouvelle. Il n'y a pas réciprocité, il n'est donc pas très rigoureux de dire que « les DTD sont compatibles ». On parle aussi de « compatibilité ascendante ».

Si l'on écrit :

```
<!ELEMENT section (titre|p)*>
```

cela signifie que, dans le document XML, il est possible de mettre la balise `<titre>` n'importe où au milieu des balises `<p>`, c'est-à-dire que l'on autorise un comportement du type « intertitres » au sens habituel du terme. Que vaut-il se passer lors de la construction de la table des matières ? Quel `<titre>` faudra-t-il choisir ? Le document a perdu sa structure stricte, garante d'une régularité de traitement lors des publications multisupports, tout en restant parfaitement XML !

En pratique, il vaudra mieux écrire :

```
<!ELEMENT section (titre?,p+)>
```

qui impose à la structure de commencer par un titre (optionnel) et de poursuivre par des paragraphes.

ATTENTION

La maintenance d'une DTD relève davantage de l'organisation de la production XML que d'une décision technologique (généralement ultrasimple). Il s'agit d'un acte de management général et pas seulement de production, compte tenu des enjeux. Voir le chapitre 5 consacré à l'organisation de la production XML.

Tenir compte (ou non) des supports cibles

On ne fait pas du XML pour le seul support papier ! L'objectif est de publier à partir d'une source unique (*single sourcing*) vers de nombreux supports ou canaux dont les caractéristiques ne sont pas toutes partagées : dès lors se pose la question de tenir compte (ou non) de ces spécificités dans le document XML.

Le support papier

Au sein des supports de publication, le papier relève de la voiture de luxe que l'on peut équiper d'une foule d'options : format de page, qualité et texture du papier, police de caractère, mise en pages, le tout sous contrôle total de l'éditeur. Avec l'accroissement de la puissance des logiciels de PAO, les mises en pages sont devenues de plus en plus complexes et de plus en plus subtiles. Inclure cette complexité et ces subtilités potentielles dans le XML est possible, mais au prix d'une complexification déraisonnable de la DTD, avec les conséquences que l'on devine sur les coûts de saisie et de traitement aval. Dès lors, deux options se présentent :

- simplifier la maquette. C'est tout à fait envisageable et il appartient au directeur artistique de tirer profit de la structuration pour imaginer des maquettes créatives, attractives, colorées, etc., sur la base de ces nouveaux critères ;
- laisser à la maquette toute sa complexité et produire a posteriori un XML épuré de l'ouvrage, à seule fin de publication électronique. Cela est acceptable, en particulier en édition scolaire où la contrainte de temps, majeure, ne permet pas de mettre en place un processus basé sur XML. On ne tiendra donc que partiellement compte du support papier en visant essentiellement à récupérer l'information textuelle à l'identique de ce qui a été publié. Pour cela, une application rigoureuse des styles permettra d'obtenir par exportation (InDesign par exemple) un XML brut qu'il suffira d'enrichir : on se met ainsi à l'abri d'une erreur de saisie lors d'un report de correction de mise en pages.

L'attribut MEDIA

En première analyse, on distingue trois média spécifiques : le papier (ex. : `media="print"`), l'écran d'ordinateur de bureau (ex. : `media="desktop"`) et l'écran de téléphone (ex. : `media="iphone"`).

En cas de mise en pages automatique – ce qui n'est pas possible dans de nombreux cas – on se réserve la possibilité d'introduire dans le flux XML des balises destinées au seul média papier. Le principe est de prévoir un attribut servant à préciser que cette mise en pages ne doit être utilisée que pour le papier. On écrira par exemple `<br media="print"/>` pour introduire une coupure forcée dans un titre lors de l'impression, tandis que cette balise sera ignorée dans la publication électronique.

Le support électronique

Les supports électroniques, moins riches graphiquement que le papier, disposent de fonctionnalités spécifiques, l'interactivité en premier lieu, et sont soumis à des contraintes, la taille de l'écran par exemple.

Fonctionnalités spécifiques

L'expression de nouvelles fonctionnalités nécessite d'introduire davantage de balisage dans le flux XML. Ainsi, pour une bibliographie interactive qui lancera, par exemple, la recherche d'un livre sur Google Books ou d'un article de revue savante sur Google Scholar, il faut repérer au minimum :

- pour un livre, le nom de l'auteur et le titre du livre. Dans le cas d'un ouvrage collectif, il faudra également repérer l'auteur principal, sauf à considérer qu'une règle implicite le place en premier de la liste;
- pour une revue, le nom de l'auteur, de la revue et le titre de l'article.

Une telle entrée bibliographique se déclarerait ainsi :

```
<!ELEMENT bibentry
(#PCDATA|author|booktitle|journaltitle
|articletitle)*>
```

etc.

pour une saisie XML qui se présenterait ainsi :

```
<p> <bibentry>
<author class="main_author">
<authorfirstname>Ian</authorfirstname>
<authorlastname>Carr</authorlastname>
</author>, <booktitle>Miles
Davis</booktitle> William Morrow & Co,
1982, traduction française : Éditions
Parenthèse, 1991 (ISBN 2-86364-057-7)
</bibentry></p>
```

Le comportement électronique se traduira par un enrichissement couleur du texte (ex. : bleu, souligné) afin d'indiquer une interaction possible par clic simple. La fonction sous-jacente transmettra à Google Books le nom de l'auteur et le titre du livre en demandant leur affichage dans une nouvelle fenêtre, par exemple.

La valeur ajoutée pour l'utilisateur est significative : il n'a plus à copier/coller dans la zone de recherche de Google pour lancer une recherche bibliographique sur un titre particulier. Accessoirement, ce balisage, plus riche que celui nécessité pour le seul papier, permettra de composer en italique le titre du livre.

Toujours dans le registre de la bibliographie, s'il s'agit d'apporter une valeur ajoutée plus fine à des chercheurs qui doivent pouvoir importer dans leur logiciel de gestion (ex. : EndNotes) leurs références bibliographiques préférées, il faudra prévoir un balisage encore plus fin des entrées, de manière à permettre dans la version électronique l'exportation automatique de ces références au format voulu.

Dans le cas de la synthèse vocale – parce que l'on veut rendre le texte accessible à des non-voyants – il faudra prévoir différentes balises spécifiques :

- une balise `<ph>` pour la phonétique permettant de substituer, lors de la synthèse vocale, un texte prononçable dans la langue choisie sans avoir à changer de locuteur. Une formule de mathématique balisée `<eqn>` aura ainsi comme texte associé : «somme de a jusqu'à b de f de x dx», ce qui s'exprimera en XML sous la forme `<ph text="somme de a jusqu'à b de f de x dx"><eqn>...</eqn></ph>`;
- une balise `<language>` pour repérer les portions de texte en langue étrangère au sein du texte, si l'on ne veut pas employer la phonétique, mais plutôt faire appel à un changement de locuteur. Le XML correspondant s'écrira alors : `<language language="EN">My taylor is rich</language>`.

Contraintes de taille d'écran

Il est évident que l'on ne peut pas s'exprimer de la même manière sur un écran de téléphone (320 × 240) que sur un écran d'ordinateur (1024 × 768 pour les plus petits affichages). Les images, formules, tableaux doivent être adaptés au support cible.

Pour les images, on exprimera des règles de retaillage sous forme de dimensions maximales en largeur et en hauteur : si les images sources sont produites pour le papier, il faudra les retailler physiquement à l'aide d'un logiciel en respectant des règles de tailles absolues en nombre de pixels. On choisira ainsi 800 pixels de large au maximum pour l'affichage sur écran d'ordina-

teur et 250 pixels pour l'écran de téléphone. Le plus simple est de prévoir un retaillage automatique lors de la publication sur les écrans cibles : on ne change pas le nom des images ni les chemins d'accès (ce qui évite de modifier les appels dans le fichier appelant), mais on stocke les images retaillées et le fichier issu de la transformation dans un autre répertoire.

L'avantage est double :

- le poids des fichiers à transmettre sur les réseaux de télécommunication est optimisé;
- les dimensions étant conformes à l'écran cible, il ne se produit pas d'effet parasite de redimensionnement des pages HTML, effet de bord classique sur un téléphone.

Pour les formules mathématiques, les exigences de lisibilité ne permettent pas de diminuer exagérément la taille de leur image. On adoptera alors une stratégie en 2 images pour le téléphone : une imagette que l'on fera apparaître dans la page, et l'image dans un format intermédiaire (ex. : largeur de 450 pixels pour un affichage à l'italienne sur un iPhone) pour un zoom dans une fenêtre séparée. La construction de ces images peut être faite de manière automatique à condition d'avoir repéré que l'image est une formule mathématique dans le texte source, ce qui se fera à l'aide d'un attribut `type` par exemple.

Pour les tableaux, le problème est souvent de nature éditoriale : pour être lisibles, ils doivent être entièrement affichables dans l'espace disponible, au moins en largeur. Le zoom n'est pas satisfaisant car on perd à la fois la tête et la correspondance des informations sur une ligne. Il faudra donc découper le tableau en le remplaçant par plusieurs petits tableaux, ce qui nécessite soit un travail éditorial supplémentaire, soit l'utilisation d'un programme astucieux de transformation, capable d'éclater un tableau en tableaux unitaires plus petits :

- dans le premier cas, on repérera ces tableaux destinés au seul affichage téléphone par un attribut `media="iphone"` par exemple, tandis que le tableau d'origine (qu'il soit image ou entièrement XML) recevra l'attribut `media="print"`.
- dans le second cas, on « délinéarisera » les tableaux en transformant automatiquement, lors de la publication électronique, les lignes de tableaux en autant de petits formulaires remplis (un formulaire par ligne). Cela n'est acceptable que si les tableaux ne sont pas trop longs.

Quelle DTD utiliser?

Le choix d'une taxonomie est crucial pour l'avenir du projet XML : DTD publiques, DTD métiers, DTD maison et DTD fournisseurs s'affrontent dans ce domaine. Quels en sont les avantages et les inconvénients?

DTD généralistes publiques

Les DTD publiques les plus connues sont au nombre de deux :

- TEI (*Text Encoding Initiative*), plutôt orientée sciences humaines ;
- DocBook, plutôt orientée ouvrages ou documentation technique.

Ces DTD étant publiques et bénéficiant de plusieurs années d'utilisation, elles ont suscité le développement de nombreux outils, en particulier de publication électronique ou papier : leur utilisation permet donc de faire l'économie de la réécriture des différents logiciels de conversion. Par ailleurs, elles assurent une interopérabilité (théorique) entre producteurs de documents XML fondés sur ces DTD publiques.

L'inconvénient de ces DTD, même si cela peut sembler étonnant au premier abord, est leur richesse : le même problème éditorial peut dès lors être traité de nombreuses manières – ce qui conduit alors à une explosion des combinatoires – et leur apprentissage est long.

À NOTER

En pratique, on sera amené à définir des sous-ensembles de l'une ou l'autre de ces DTD, en conservant une compatibilité complète avec la DTD générale, afin de préserver l'utilisation des outils existants créés par la communauté TEI ou DocBook.

TEI

La TEI (www.tei-c.org) a été créée pour les besoins des sciences humaines, en particulier pour l'encodage des manuscrits. Cette DTD permet d'encoder le texte, les variantes, les ratures, tout en étant très fine sur la position des mots correspondants dans le manuscrit, les renvois, les index.... Indispensable

pour la recherche en sciences humaines, elle reste, à notre avis, trop complexe et donc trop coûteuse pour l'édition institutionnelle.

On se contentera d'y puiser des éléments déjà déclarés et bien formalisés : la déclaration, désormais classique, des principaux éléments hiérarchiques d'un livre <front>, <body>, <back> vient de cette DTD.

DocBook

DocBook a été créée dans le dessein d'encoder des livres et des documents techniques. Elle a été initiée par l'éditeur O'Reilly pour ses besoins propres et a été mise en open source afin de bénéficier d'une communauté de contributeurs.

Comme la TEI, elle bénéficie de nombreux outils développés au fil des ans par les différents contributeurs. Et comme la TEI, elle est trop riche et dispose d'une combinatoire trop étendue pour traiter un même problème éditorial : on choisit donc un sous-ensemble avec des règles très strictes d'utilisation.

POUR EN SAVOIR PLUS

Le livre de référence de DocBook 5 est accessible en HTML :
www.docbook.org/tdg5/en/html/docbook.html

L'édition

La plupart des industries disposent aujourd'hui d'une taxonomie adaptée à leurs besoins, à l'exception notable de l'édition : la raison doit en être recherchée dans l'extrême diversité des modèles envisageables, rendant illusoire la conception d'une DTD unifiée.

DTD métiers

De nombreux secteurs d'activité ont ressenti le besoin de se doter de taxonomies spécifiques, en premier lieu pour assurer l'interopérabilité de leurs données. L'industrie aéronautique, par exemple, s'est pourvue très tôt de taxonomies permettant de structurer la documentation technique des avions. Élaborées par l'*Air Transport Association* (ATA), les DTD ont été enrichies et améliorées au fil du temps : ATA100, créée dans les années 1980, ATA 2100, et jusqu'à la plus récente, ATA iSpec 2200 (*Information Standard for Aviation Maintenance*). Celle-ci compte de nombreuses DTD correspondant à des sous-ensembles documentaires comme l'*Aircraft Illustrated Part Catalog* (AIPC), l'*Aircraft Maintenance Manual* (AMM), etc.

Dans le domaine de la presse écrite et audiovisuelle, l'IPTC (*International Press Telecommunication Council*) a mis au point un ensemble de taxonomies, connues sous le nom de «IPTC G2 standard», pour décrire les messages XML représentatifs d'une dépêche, d'un événement ou de résultats sportifs assurant une interopérabilité planétaire aux informations produites par les acteurs du domaine.

DTD maison et DTD fournisseurs

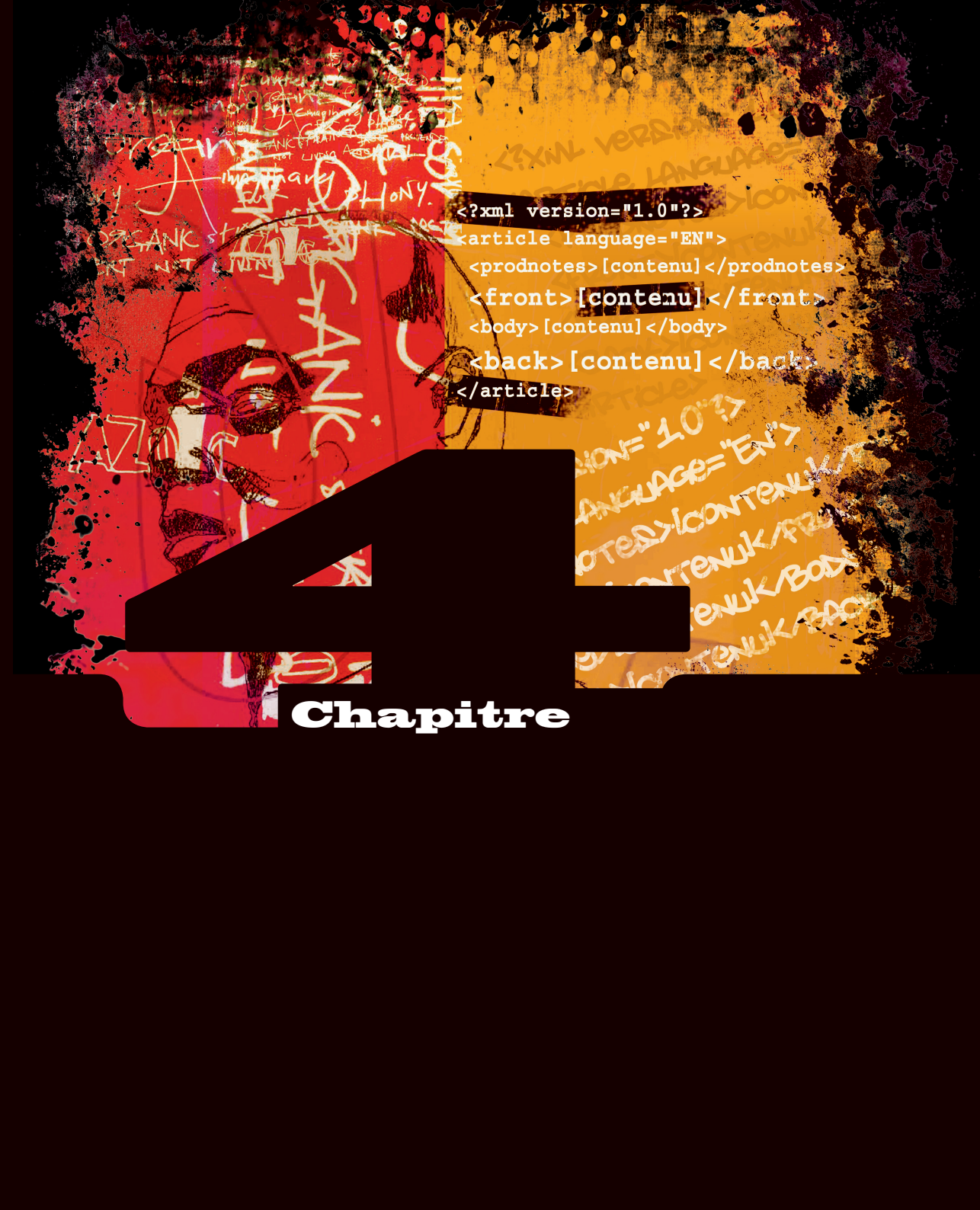
Chaque acteur du monde de l'édition propose sa propre DTD autour de laquelle des développements techniques et organisationnels parfois très lourds ont été consentis, ce qui rend difficile une migration vers une nouvelle taxonomie. Les prestataires doivent dès lors utiliser ces DTD et adapter leurs méthodes de production; celles-ci sont généralement bâties autour de leurs propres DTD, dites « DTD de production » (ou « DTD pivot »), que le prestataire va légèrement adapter pour assurer une bonne compatibilité avec celles de ses clients.

Cette démarche est dictée au prestataire par différentes exigences :

- être capable de répondre aux besoins de ses nombreux clients aux DTD disparates, sans avoir à tout réapprendre à ses équipes de production. Un paragraphe sera toujours balisé en interne par `<p>`, même si les clients ont choisi `<para>`, `<paragraphe>`, `<alinea>`, etc.;
- factoriser les efforts de développement autour de cette DTD de production en évitant de tout refaire pour chaque projet;
- ne pas modifier les processus de production interne, malgré la disparité des taxonomies;
- répondre rapidement à de nouveaux projets.

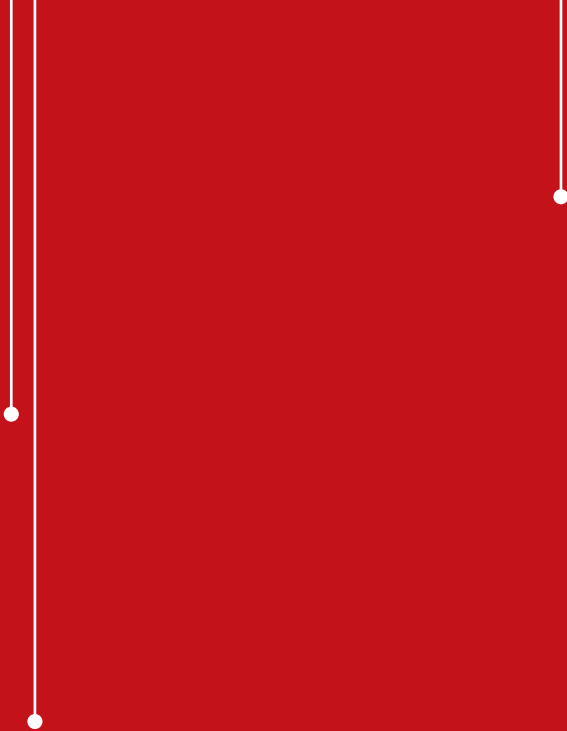
Tout cela est transparent pour l'éditeur qui recevra des documents XML conformes à sa propre DTD, indépendamment des méthodes de production utilisées par ses fournisseurs; c'est d'ailleurs l'objectif de XML que de rendre complètement étanches les systèmes de production, puisque seul compte le résultat.





<?xml version="1.0"?>
<article language="EN">
<prodnotes>[contenu]</prodnotes>
<front>[contenu]</front>
<body>[contenu]</body>
<back>[contenu]</back>
</article>

Chapitre



Saisir le XML

La DTD écrite et donc les règles d'utilisation des balises fixées, on fait appel à un traitement de texte XML pour poser les balises dans le texte. Les sites de production utilisent l'informatique, la copie étant généralement disponible en Word. L'éditeur qui intervient sur un texte XML doit pouvoir ouvrir un document XML avec un outil spécialisé pour apporter les corrections voulues sans faire appel à un prestataire.



Utiliser un éditeur XML

Le mot « éditeur » est polysémique : il signifie aussi bien maison d'édition, personne chargée de l'édition du texte que traitement de texte basique. Il s'agit de ce dernier sens lorsque l'on parle d'éditeur XML : un outil spécialisé permettant de poser les balises selon les règles édictées par la DTD.

Qui doit utiliser un éditeur XML ?

La théorie voudrait que le texte XML soit produit par l'auteur et que tous les acteurs de la chaîne du livre interviennent sur ce texte XML pour lui apporter les raffinements dont il a besoin. Une fois le texte XML finalisé, « il n'y a plus qu'à » produire les objets de publication, qu'ils soient sous forme papier ou numérique. En pratique et compte tenu de la maturité des intervenants vis-à-vis de XML, c'est plutôt à l'extrémité de la chaîne du livre, du côté des compositeurs, qu'il convient d'apprécier l'utilisation d'un éditeur XML.

Le prestataire de saisie ou de composition papier (quand la même société cumule ces deux métiers) est aujourd'hui celui qui fait le plus grand usage d'un éditeur XML. Les outils qu'il utilise sont des outils à vocation industrielle, fonctionnant souvent sur des ordinateurs UNIX et non pas des PC ou, plus rare encore, des Mac. Il travaille souvent avec une DTD de production qui lui permet de produire le XML dans de bonnes conditions de productivité et s'organise pour transformer ce XML intermédiaire en XML conforme à la DTD de son client. Le « contrat » entre ce prestataire et le client stipule de livrer un document XML conforme à la DTD client : ce document XML doit passer correctement l'épreuve du parseur et la saisie de texte ne doit pas dépasser un certain pourcentage d'erreurs, généralement explicité de manière contractuelle. Le prestataire a une obligation de résultat et non de moyens : peu importe l'éditeur utilisé, il suffit que le document XML livré soit conforme.

À NOTER

C'est d'ailleurs un des avantages de XML que de ne pas dépendre d'un outil spécifique (tandis qu'un texte saisi en Word ou dans InDesign ne pourra être ouvert qu'avec un nombre limité de logiciels).

Du côté de la maison d'édition, les acteurs impliqués sont le fabricant et l'éditeur : le fabricant parce qu'il est l'interlocuteur naturel du prestataire de saisie/composition, et l'éditeur parce qu'il intervient sur le contenu.

- En pratique, le fabricant doit connaître XML puisqu'il est amené à dialoguer avec son fournisseur. Il n'intervient pas sur le texte et n'a donc pas a priori besoin d'un outil d'édition XML.
- L'éditeur, en revanche, est impliqué en premier lieu : soit il travaille « à l'ancienne » et indique sur une copie papier les modifications à apporter au fichier par le prestataire, soit il effectue directement dans le XML les corrections mineures, raccourcissant ainsi considérablement le cycle éditorial. C'est ainsi que sont organisées les maisons d'édition les plus avancées ou spécialisées (l'édition juridique, une partie de l'édition scientifique). Dès lors, il semble opportun de s'approprier cette démarche, de choisir un éditeur XML, de le paramétrer et de l'utiliser, éventuellement avec l'aide des informaticiens maison dans la période de démarrage.

Comprendre la relation entre DTD, document XML et visualisation

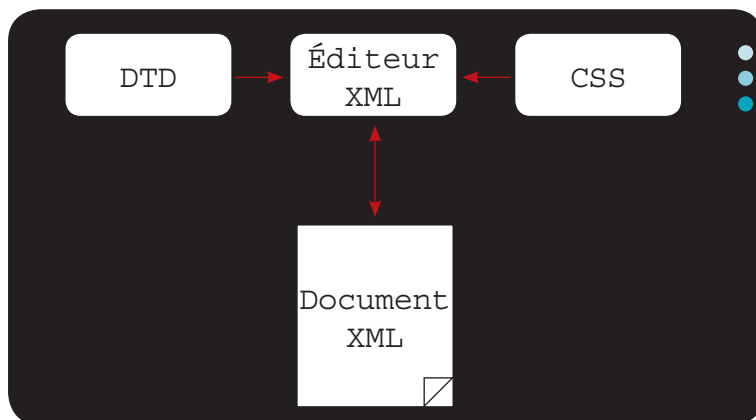
Il est difficile, lorsque l'on débute en XML, de comprendre la relation entre la DTD, le document XML et son rendu visuel lors de la saisie au sein d'un éditeur XML. En pratique, l'environnement technique de saisie XML fait intervenir trois fichiers :

- la DTD. On considère qu'il s'agit d'un seul fichier, mais elle peut être organisée physiquement en différents modules appelés depuis un fichier principal ;
- un fichier CSS de configuration permettant d'obtenir un rendu graphique à l'écran indépendamment du média final. Comme pour la DTD, la CSS peut être fragmentée en un fichier principal appelant différents modules ;
- le document XML proprement dit, produit avec l'aide de l'éditeur XML.

Il est important de comprendre que le rendu graphique d'un tel éditeur, contrairement à un traitement de texte classique, n'est pas pérenne et ne constitue pas une fin en soi : il s'agit d'une simple aide à la saisie. On parle parfois à ce sujet de WYSIOO : « What you see is one option », soit en français « ce que vous voyez n'est qu'une option », pour souligner l'aspect temporaire de cette mise en forme. Un éditeur XML produit un flux textuel de contenu et de balises sans enrichissement, sans images, sans objet multimédia ; ce n'est que du texte. D'ailleurs, celui-ci peut être ouvert avec n'importe quel outil de base sachant manipuler du texte basique, le Bloc-Notes par exemple.

L'auteur et le XML

En fait, l'auteur devrait être le premier utilisateur d'un éditeur XML. C'est en effet l'idéal mais relève de la pure théorie. Il faudra sans doute 15 ou 20 ans – une génération – pour que l'auteur s'approprie cette démarche de manière généralisée. Ce n'est pas totalement utopiste : il est des maisons d'édition qui ont su former leurs auteurs à cette approche, particulièrement dans le domaine scientifique.



La plupart des éditeurs proposent également une vue enrichie avec pseudo-balises ; celles-ci sont des visualisations graphiques des vraies balises et présentent l'avantage de permettre un positionnement précis du curseur en cours de travail.

Choisir et paramétrer son éditeur XML

Choisir l'éditeur

Il existe de nombreux éditeurs XML sur le marché, souvent disponibles dans les trois principaux systèmes d'exploitation utilisés sur les ordinateurs personnels (PC, Mac, Linux).

EN PRATIQUE

XMetal (justsystems.com) et Oxygen (oxygenxml.com) sont téléchargeables en version d'évaluation complète limitée à 30 jours.

Le choix est délicat, car chacun d'entre eux a ses spécificités, ses avantages ergonomiques, ses thuriféraires et ses contempteurs. Pour notre part, nous utilisons indifféremment XMetal ou Oxygen.

XMetal est un éditeur XML natif qui bénéficie de nombreuses années de développement et dont la productivité sur site de production n'est plus à démontrer. En outre, il est facilement programmable, ce qui permet :

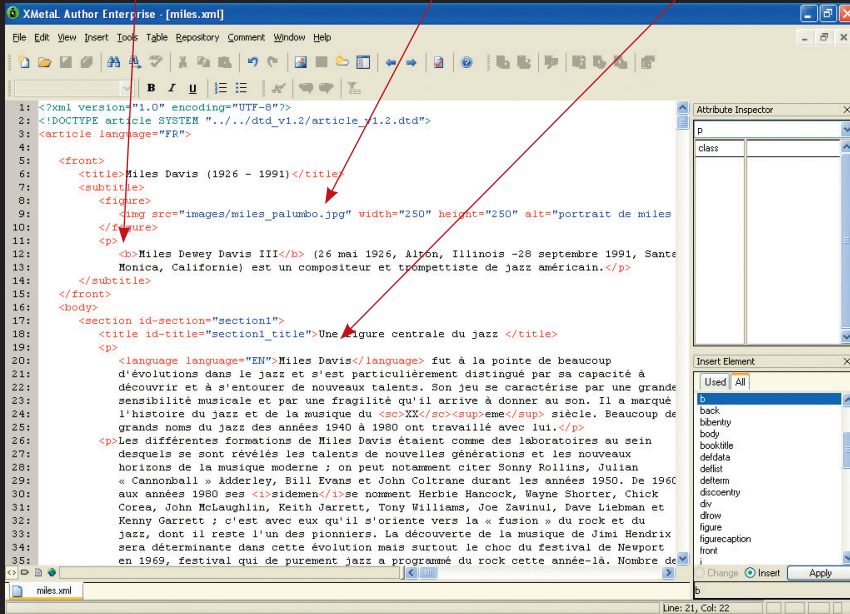
- de lui adjoindre des fonctionnalités propres à chaque DTD ;
- de l'épurer des fonctionnalités inutiles dans le cadre d'un projet éditorial spécifique.

XMetal, édité par la société Just System, est une application Windows et se décline en différentes versions en fonction du type d'utilisation :

Paragraphe de sous-titre

Référence du fichier image

Titre de section niveau 1

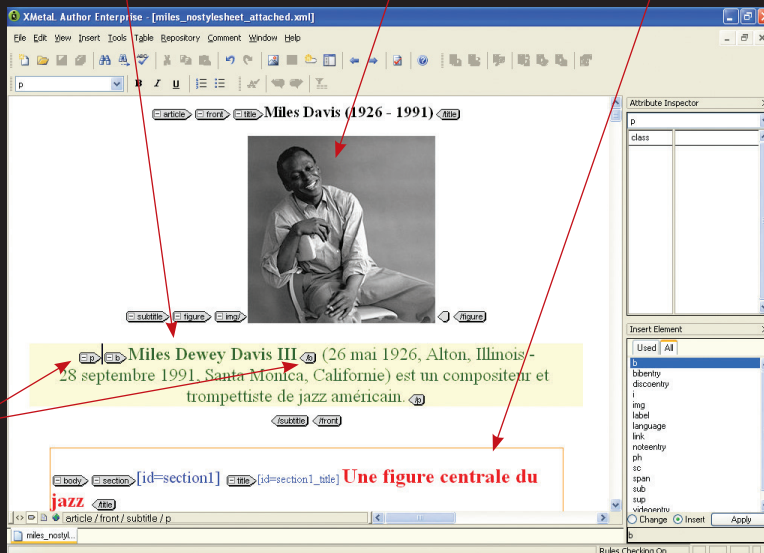


Paragraphe de sous-titre

Image visualisée

Titre de section niveau 1

Pseudo-balise



- Vue du flux XML et sa contrepartie « enrichie » dans l'éditeur XML.
- Les pseudo-balises sont ici visibles.

Remarque

À notre connaissance, la société ne distribue plus de versions localisées (c'est-à-dire adaptées au niveau de l'interface à une langue locale). Il faut donc être un peu familiarisé avec l'anglais.

- XMetal Author : il s'agit de l'éditeur XML complet, offrant toutes les fonctionnalités d'édition de documents XML. Il est également proposé en versions intégrables à des systèmes élaborés de gestion documentaire ou de collaboration comme Documentum (EMC), Sharepoint (Microsoft) et FileNet (IBM). C'est l'éditeur que nous utilisons le plus souvent dans le cadre de cet ouvrage ; toutes les copies d'écran correspondent à XMetal en version 5.5 ;
- XMetal Reviewer, destiné aux relecteurs ;
- XMetal Developper pour les informaticiens chargés de personnaliser ou reprogrammer XMetal Author.

Oxygen, édité par Syncro Soft, a un positionnement un peu différent : c'est un «environnement de développement intégré» ou EDI permettant d'effectuer à l'aide d'un seul outil tous les travaux XML, depuis la saisie du document XML jusqu'à la mise au point des transformations vers les formats éditoriaux cibles : XHTML, PDF par exemple.

En pratique, l'offre de l'éditeur s'organise en produits emboîtés :

- XML Diff pour comparer des fichiers XML ;
- SVN client pour interagir avec un entrepôt de fichiers piloté par un outil de gestion des versions de fichier (SVN = SubVersion) ;
- XML Author qui inclut les deux produits précédents. Comme son nom l'indique, il s'agit de l'éditeur XML utilisé par les auteurs ou les opérateurs de saisie. Il permet également de lancer les publications par transformation, à condition que celles-ci aient été mises au point dans le cadre de l'EDI ;

À NOTER

Deux autres points importants : Oxygen étant écrit en Java, il est disponible aussi bien sur Mac que sous Windows et sous Linux ; par ailleurs, l'interface est disponible en français.

- XML Editor, qui à son tour inclut XML Author, constitue l'EDI et permet d'effectuer toutes les opérations de mise au point dans un environnement XML : écriture et réglages des taxonomies (DTD en

particulier) et des transformations vers des formats de publication comme PDF ou XHTML, grâce à XSLT. Par ailleurs, l'outil prend en compte les principales taxonomies publiques, en particulier TEI et DocBook.

Parmi les autres EDI du marché, citons Stylus Studio (stylusstudio.com) et XML Spy (altova.com) qui proposent d'excellents produits résolument techniques mais, à dire vrai, peu orientés vers la saisie.

XML Copy Editor est un exemple d'éditeur XML gratuit (xml-copy-editor.sourceforge.net). Il permet de travailler sur le flux XML de manière native, comme avec un éditeur de texte classique du type Notepad, avec en

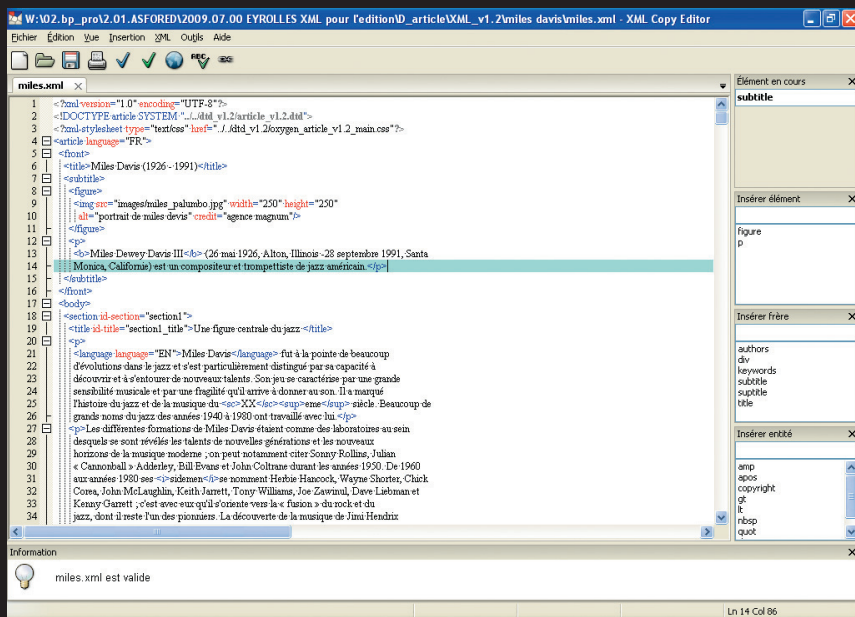
plus le contrôle de validité du document. Toutefois, il ne dispose pas d'un rendu graphique, ce qui limite son usage à un mode correction.

Signalons enfin XML Editor (xmlmind.com), disponible en version gratuite pour un usage non professionnel. L'interface est un peu déroutante, mais l'outil est suffisant pour un travail en mode correction.

Cela dit, les besoins d'un éditeur (ou d'un auteur) ne sont pas ceux d'un informaticien documentaire. On retiendra comme critères de choix pour un utilisateur non informaticien :

- la possibilité de visualiser facilement la structure (balises et attributs) grâce à un enrichissement typographique basé sur un langage connu, en l'occurrence CSS ;
- le basculement aisé entre la vue purement textuelle (c'est-à-dire le texte et ses balises) et la vue enrichie, avec ou sans pseudo-balises ;
- la possibilité de plier/déplier tout ou partie de la structure ;
- la possibilité de contrôler (de parser) tout ou partie du document ;

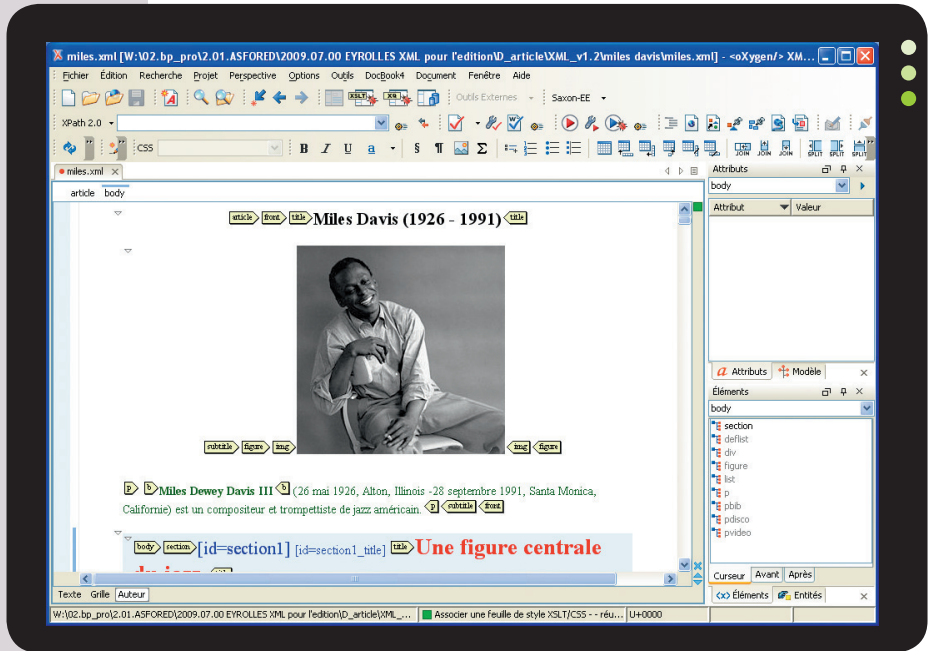
● Un éditeur gratuit, XML Copy Editor, version 1.2.0.6



● Le même document que celui de la figure de la page 83 ouvert dans Oxygen : la CSS ne donne pas exactement le même rendu et les pseudo-balises sont un peu différentes.

- la facilité d'apprentissage et d'utilisation. Ce critère est bien sûr assez subjectif : l'outil le plus facile à utiliser est toujours celui que l'on connaît...

Finalement, l'ergonomie ou le ressenti de l'ergonomie compteront largement dans le choix du logiciel, si l'on fait abstraction du prix, car celui-ci varie de la gratuité complète à plus de 1500 euros par poste de travail.



Paramétrer l'éditeur

Accrocher une CSS. La plupart des éditeurs offrent la possibilité de paramétrer le rendu typographique à l'aide d'un fichier CSS. L'association d'une CSS à un document XML se fait par l'adjonction d'une information au début du fichier XML. En pratique, il suffit d'ajouter la ligne suivante dans le document XML, après la déclaration XML et l'appel de DTD qui constituent généralement les deux premières lignes d'un fichier XML :

```
<?xml-stylesheet type="text/css"
href="../../../DTD_v1.2/article_v1.2.css"?>
```

Cette ligne associe la CSS `article_v1.2.css` au document XML en question; bien entendu, il convient d'écrire correctement le chemin d'accès à la DTD. En règle générale, on stocke la DTD dans un répertoire partagé (ou

directement sur l'Internet, mais dans ce cas elle est visible pour tous), ce qui explique l'écriture `../.. /dt_d_v1.2` qui signifie « remonter de deux niveaux (par rapport au niveau courant, c'est-à-dire celui du fichier XML) dans les dossiers et redescendre dans le dossier `/dt_d_v1.2` ».

Lors de l'ouverture dans l'éditeur XML, les instructions CSS sont interprétées pour donner un rendu graphique. Il est à noter que l'ouverture de ce même document XML dans un navigateur se traduira également par un enrichissement typographique plus ou moins similaire à celui obtenu dans l'éditeur.

Comprendre la CSS. Une CSS (*Cascading Style Sheet*) est une succession d'ordres typographiques élémentaires portant sur les balises ou sur une combinaison de balises : on peut donner une couleur rouge à un titre de niveau 1, une couleur bleue à un titre de niveau 2 (un titre de section dont le premier ancêtre est une section) et une couleur noire à un titre de niveau 3.

On écrira ainsi :

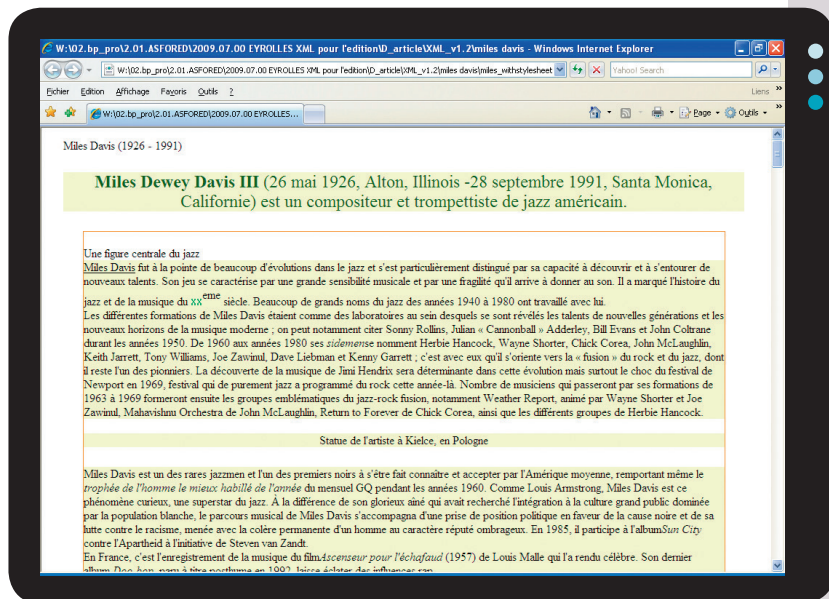
```
section section title {
color:blue;
}
```

pour avoir un rendu bleu sur un titre de niveau 2.

La ou le CSS ?

CSS (*Cascading Style Sheet*) signifie littéralement « feuille de styles en cascade ». Le mot feuille étant du genre féminin, on a pris l'habitude de parler de LA CSS. On peut utiliser le masculin si l'on sous-entend « le fichier CSS ».

● Le document de la figure de la page 83 ouvert dans un navigateur. On notera les différences de rendu avec celui de l'éditeur XML, en particulier l'absence d'image.



Ces ordres se propagent dans la structure : ils «cascadent», d'où leur nom. Si l'on associe à la balise <p> la police Garamond, alors tous les <p>, quel que soit leur emplacement dans la structure, hériteront de cette police... jusqu'à ce qu'un contre-ordre modifie ce fonctionnement. Le processus s'arrêtera en effet si l'on définit quelque part dans la CSS un comportement typographique différent pour un contexte différent : <p> dans un <subtitle> pourrait avoir un autre comportement typographique (ex. : italique) en substitution du comportement typographique par défaut. Ce comportement s'exprimerait de la manière suivante en CSS :

```
subtitle p {  
font-style:italic;  
}
```

Sans aller plus loin dans l'étude de CSS, précisons qu'il est possible de spécifier un comportement de type bloc (le contenu de l'élément s'inscrit dans un rectangle) ou de type inline (le contenu de l'élément reste dans la ligne contextuelle). Un titre dont le comportement est de type bloc se trouvera affecté de la commande suivante :

```
title{  
display: block;  
color: red;  
}
```

Une balise dont le comportement est de type inline recevra la commande suivante :

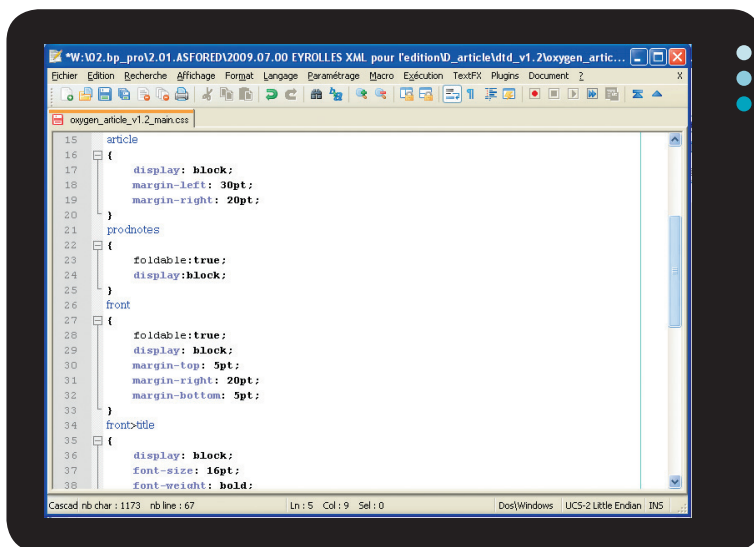
```
b {  
display: inline;  
font-weight: bold;  
}
```

Le langage CSS est très puissant et permet d'effectuer des tests sur un élément ou un attribut pour définir des comportements typographiques sophistiqués que l'on mettra en œuvre pour guider de manière sûre l'opérateur dans ses tâches de saisie.

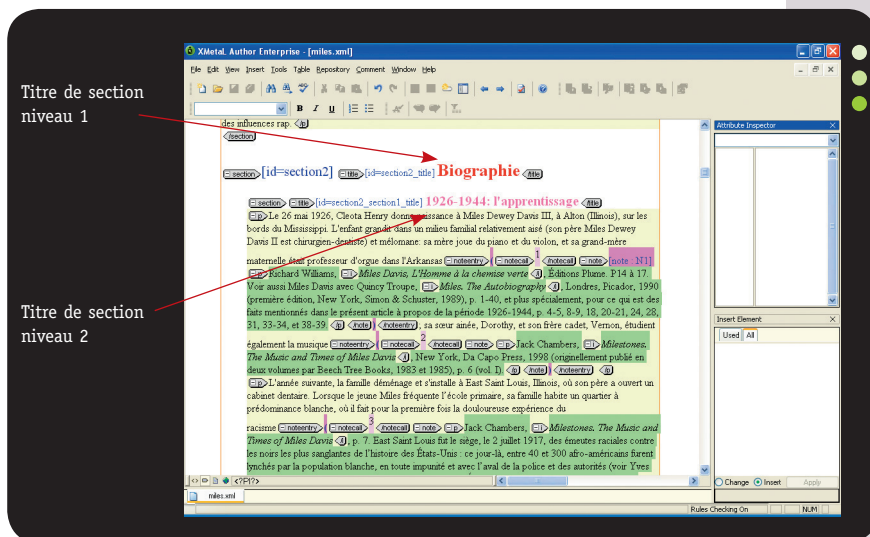
Note

Il existe des moyens plus puissants, mais très dépendants de l'éditeur, de personnaliser l'interface, par exemple sous l'aspect de formulaires.

Une remarque générale : chaque éditeur XML a son propre rendu en matière d'interprétation des commandes CSS. Par ailleurs, les éditeurs incorporent des commandes CSS propriétaires (c'est-à-dire propres à leur seul éditeur). Oxygen utilise ainsi la commande `foldable:true` pour exprimer qu'un élément peut être plié ou déplié dans l'interface. Cette commande est ignorée des autres éditeurs.



- Le début de la CSS d'article_v1.2.dtd. On notera la présence de la commande `foldable:true` propre à Oxygen.



- Exemple de configuration typographique de l'élément titre de section (`<section><title>`), titre de sous-section (`<section><section><title>`) et son rendu graphique dans l'éditeur XMetaL

Utiliser une DTD

La DTD mise au point grâce au concours de nombreuses personnes va être utilisée aussi bien par les prestataires que par les différents intervenants de la maison d'édition. Pour en tirer le meilleur parti, il convient de la visualiser et de la documenter avant de la mettre en œuvre au sein d'un éditeur XML.

Visualiser une DTD

Se faire une image de ce qu'est une structure n'est pas chose aisée : chacun d'entre nous dispose de son propre modèle mental. Dans tous les cas, la mémoire est sollicitée même si l'éditeur XML se charge de préciser à l'utilisateur l'élément à placer en fonction du contexte. Il est nécessaire que l'utilisateur sache presque « par cœur » ce qui peut être inséré à tel ou tel endroit.

Une bonne méthode consiste à produire un diagramme de la taxonomie : il s'agit simplement d'une vue arborescente de la structure définie par la DTD. Un outil comme Near & Far Designer produit des diagrammes simples et clairs. Les IDE, de leur côté, disposent également d'outils de visualisation, dont certains nécessitent de passer par le formalisme XML Schema ; ils sont souvent plus techniques et parfois mal adaptés à l'utilisateur final.

Avec Near & Far Designer

EN PRATIQUE

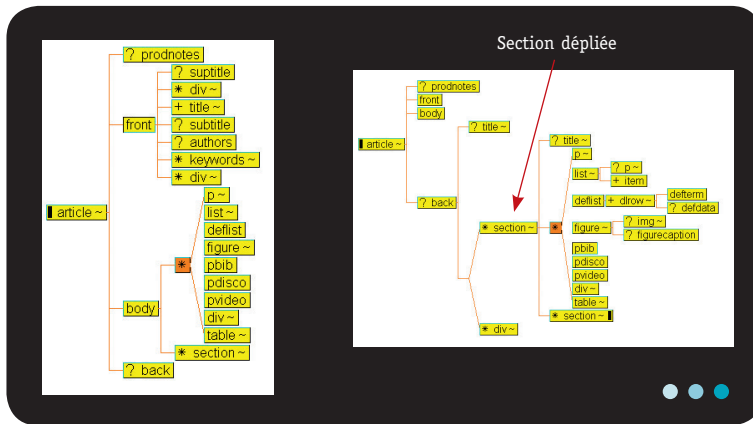
La découverte d'une DTD se fait en explorant l'arborescence depuis le premier niveau, en parcourant les branches grâce à la fonction plier/déplier implicite, disponible sur chaque nœud de l'arbre.

Near & Far Designer de la société Microsoft, maintenant disparue, est un produit Windows conçu à l'origine pour aider à la rédaction de DTD. Il est surtout utile pour explorer une DTD existante, la visualisation en diagramme simple de Near & Far étant suffisante pour un uti-

lisateur final. En fonction de ses propres préférences, il peut choisir un diagramme horizontal ou vertical.

Le formalisme est un peu désuet (la version que nous utilisons est la V3 qui date de... 1998) mais parfaitement intuitif et pédagogique. Les symboles d'occurrence des éléments sont indiqués de façon comparable à ceux de la DTD (* + ?), avec toutefois la différence que ceux-ci sont placés devant le nom de l'élément et non pas après comme dans une DTD.

En environnement éditorial, il est fort pratique de disposer de cet outil sur sa propre machine, ce qui permet à l'utilisateur de jeter un coup d'œil rapide à la structure, en même temps qu'il édite le document XML.



● Article_v1.2.dtd
visualisé à l'aide de
Near & Far Designer. La
structure peut être dépliée
à chaque nœud.

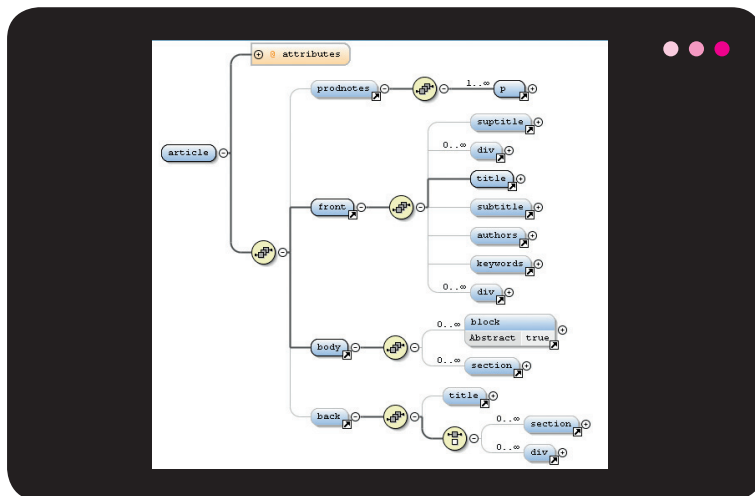
Avec Oxygen

Oxygen propose un outil élaboré de visualisation de la taxonomie. Il impose de convertir au préalable la DTD en .xsd (Schema), grâce à une commande intégrée à l'éditeur, afin que l'outil puisse bâtir la vue graphique ; c'est une petite contrainte qui ne pose pas de problème.

Comme pour Near & Far Designer, les nœuds sont cliquables et il est possible de plier/déplier l'arborescence avec toute la profondeur voulue, mais l'outil est très exigeant en espace d'affichage : un grand écran est impératif si l'on veut avoir une vue globale de la structure !

La représentation de la DTD se faisant au sein d'Oxygen, l'outil est beaucoup moins pratique en production ou en apprentissage car il faut jongler sans cesse entre les fenêtres.

● Article_v1.2.dtd.
Visualisation interactive au
sein d'Oxygen : chaque nœud
(élément, attribut...) peut être
plié et déplié.



Documenter une DTD

Documenter une DTD, ou plus généralement une taxonomie, est une démarche utile dans un environnement industriel et indispensable dans un environnement éditorial où la tâche de balisage XML est loin de constituer l'essentiel du travail. Disposer d'un manuel de référence, suffisamment concis et de préférence électronique (en PDF par exemple), permettra de retrouver la structure d'un élément peu utilisé sans avoir à rouvrir la DTD.

Plus prosaïquement, la DTD n'étant généralement pas accessible aux utilisateurs (elle est stockée sur le réseau et pas forcément visible), il est utile d'avoir cette documentation à portée de souris.

La documentation d'une DTD s'effectue selon deux approches :

- soit par une approche arborescente : on part des niveaux les plus élevés et l'on « descend » dans les niveaux inférieurs en adoptant une logique documentaire ;
- soit par ordre alphabétique des éléments, ce qui est nettement plus abstrait et ne peut répondre qu'à une question ponctuelle d'information sur une sous-structure.

Dans le premier cas, le travail se fait « à la main » à l'aide d'un traitement de texte classique : copie depuis l'outil de visualisation des DTD des diagrammes dépliés selon la profondeur désirée, associée à une explication minimale spécifiant l'objectif éditorial de telle ou telle sous-branche, accompagnée éventuellement d'un exemple de saisie. Une telle documentation n'est pas exhaustive,

mais largement suffisante pour un usage ponctuel, sachant que le dispositif pourra être complété par des consignes de saisie (voir le chapitre 5).

Dans le second cas, la documentation est produite directement par le logiciel. Oxygen dispose d'un outil de génération complet de la documentation de la structure. Celle-ci peut être enregistrée en PDF, ce qui permet de facilement la mettre à disposition des utilisateurs. Les inconvénients sont l'aspect systématique – en pratique, on n'a pas besoin de documenter tous les éléments – et l'approche peut-être un peu technique pour un utilisateur ponctuel.

Extrait de la documentation de article_v1.2.dtd générée par Oxygen. On remarquera le passage par le formalisme Schema (qui reste lisible... dès que l'on connaît XML).

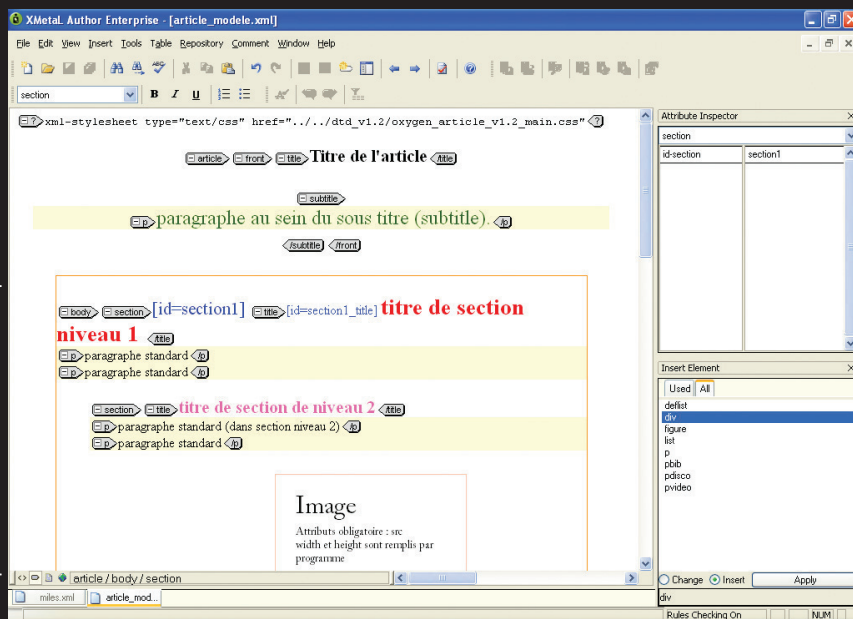
Schema de l'élément <article>

Utiliser un document modèle vide

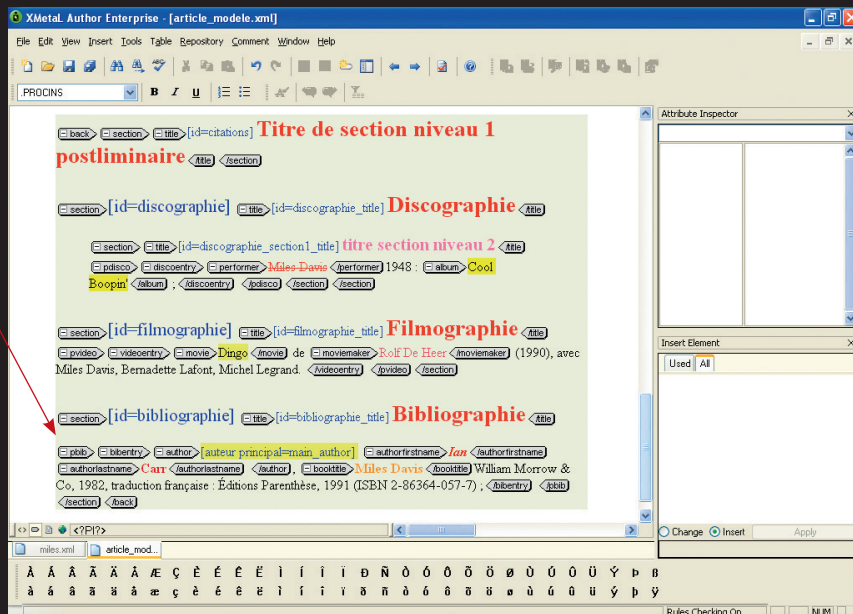
Une fois la DTD mise au point et la CSS d'affichage correctement configurée, il est possible de lancer la production. Dans la plupart des cas, les sites de production n'utilisent pas un éditeur pour la saisie, mais passent par différents processus bureautiques ou informatiques destinés à augmenter la productivité : par exemple, préparation de copie en Word stylé et conversion automatique produisant un XML brut qu'il restera à raffiner.

Du côté de la maison d'édition, il en va différemment : en phase de saisie à partir d'une page blanche (en réalité, cela ne se produit jamais...), ou lorsqu'on se lance dans du postbalisage – c'est-à-dire la pose de balises sur un texte existant dans une forme éditée (PDF, traitement de texte ou texte

Modèle de la structure hiérarchique



Modèle de la structure bibliographique



pur) – il est plus pratique de partir d'un document modèle, c'est-à-dire un squelette reprenant l'essentiel d'un document. Cette démarche présente trois avantages :

- l'utilisateur n'est pas complètement dépourvu face à la page blanche (même s'il dispose par ailleurs d'un mode d'emploi écrit minimal);
- il a, dès le démarrage, une bonne idée générale de la structure puisqu'il s'agit de remplir et d'amender un document canevas existant;
- enfin, le modèle est un très bon support pédagogique en phase d'apprentissage.

Le document modèle est constitué de textes saisis explicitant les niveaux de structure hiérarchique (section, sous-section, divisions) et donnant les exemples parmi les plus courants pouvant se produire dans le fonds documentaire traité.

À NOTER

Certains documents XML se soumettent facilement à une représentation de type formulaire; dans ce cas, le document modèle sera organisé en champs de saisie précédés de libellés explicitant les saisies attendues.

Référencer la DTD dans le document XML

Un document répondant à une structure explicitée dans une DTD doit pouvoir indiquer aux éditeurs XML (et à l'utilisateur...) la DTD utilisée. Pour cela, la DTD est « attachée » au document XML selon un formalisme comparable au rattachement d'une CSS. On insère immédiatement après la déclaration XML, qui constitue la première ligne d'un fichier XML, la ligne suivante :

```
<!DOCTYPE article SYSTEM
"../../dtd_v1.2/article_v1.2.dtd">
```

ce qui, traduit en français, signifie « rattacher à l'élément `article`, la DTD `article_v1.2.dtd` ».

Tout comme pour la CSS, le chemin relatif d'accès à la DTD est indiqué par le formalisme `../../dtd_v1.2/` (remonter deux fois et descendre dans le répertoire `/dtd_v1.2`).

En pratique, l'éditeur XML se charge de cette insertion, après avoir ouvert une boîte de dialogue demandant la DTD à utiliser. Si l'on fait appel à un modèle, le chemin d'accès à la DTD est déjà explicité et l'utilisateur n'a rien à faire.

Saisir le texte et la structure

L'éditeur XML configuré et la DTD mémorisée, le document structuré peut être produit. Le travail se limite généralement au postbalisage d'un flux de texte. La maison d'édition pourra apporter des corrections au texte XML (élément, attribut, entité, codage d'un caractère) ou même ressaisir ou compléter le texte.

Utiliser l'éditeur XML

Insérer des éléments

L'éditeur XML permet d'insérer un élément dans un contexte grâce à un modèle de visualisation assez simple : en fonction de l'endroit où se trouve le curseur, l'outil liste dans une fenêtre, souvent placée à droite et en bas de la zone de saisie et baptisée «Insert Element» dans XMetal, les éléments autorisés dans ce contexte. Il suffit de cliquer deux fois sur le nom de l'élément pour que celui s'insère dans le flux de texte. Cette méthode garantit que l'organisation du texte XML respecte la DTD.

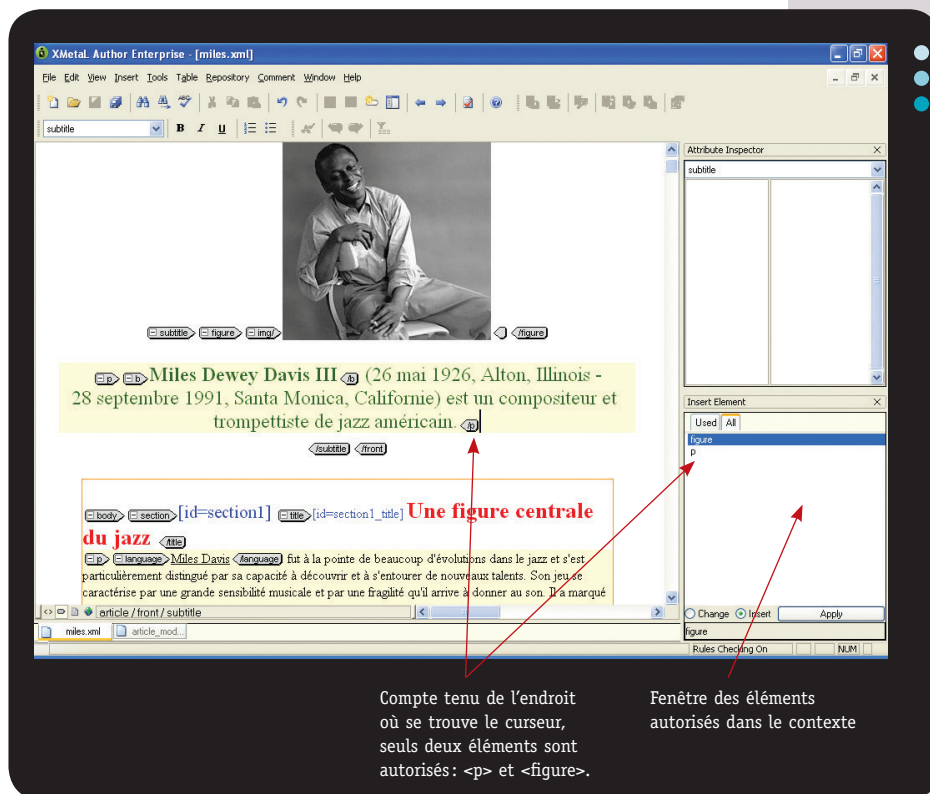
Sur la figure de la page suivante, le curseur est placé juste après la fin du paragraphe inséré dans la balise <subtitle>. D'après la DTD :

```
<!ELEMENT subtitle (figure|p)+>
```

<subtitle> ne peut contenir qu'un mélange d'éléments <figure> ou <p>, ce qui s'exprime par les seuls choix possibles proposés dans la fenêtre des éléments, à droite et en bas de la fenêtre principale.

Renseigner des attributs

Les attributs sont des informations complémentaires devant apparaître dans la balise ouvrante. En pratique, l'outil propose les attributs disponibles dans



une deuxième fenêtre située le plus souvent à droite, au-dessus de la liste des éléments autorisés, à l'emplacement du curseur. Les attributs obligatoires sont souvent indiqués en gras.

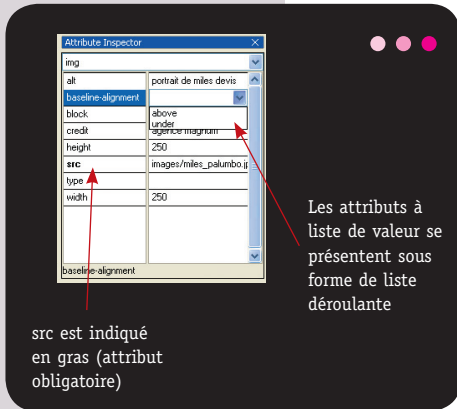
Suivant le type d'attribut, la fenêtre proposera une liste déroulante (cas des listes de valeurs pour les attributs) ou une zone de saisie. Dans le cas des images, `article_v1.2.dtd` prévoit les attributs suivants :

```
< !ELEMENT img EMPTY >
< !ATTLIST img type (formula|table) #IMPLIED
                src CDATA #REQUIRED
                width CDATA #IMPLIED
                height CDATA #IMPLIED
                baseline-alignment (above|under)
                                #IMPLIED
                block (yes) #IMPLIED
```

● Le curseur est placé derrière le paragraphe dans le sous-titre : seuls les éléments autorisés <figure> et <p> sont listés dans la fenêtre des éléments.

```
alt CDATA          #IMPLIED
credit CDATA       #IMPLIED>
```

L'attribut `src` est le seul obligatoire : il apparaîtra en gras dans la fenêtre des attributs.



● La fenêtre (Oxygen) d'édition des attributs de l'élément `img` `article_v1.2.dtd`. Visualisation interactive au sein d'Oxygen : chaque nœud (élément, attribut...) peut être plié et déplié.

● Le ruban (XMetal) d'insertion des caractères spéciaux

L'attribut `baseline-alignment` (utilisé pour positionner l'image au-dessus ou au-dessous de la ligne de base) ne propose que deux valeurs (`above` et `under`). Dans l'éditeur XML, ici Oxygen, ce choix se traduira par une liste déroulante.

Encoder des caractères

Les éditeurs modernes incorporent directement les caractères Unicode qui peuvent donc être saisis soit simplement depuis le clavier, soit à l'aide d'une barre de caractères spéciaux comme dans Xmetal. Selon l'encodage choisi pour le document XML, le caractère sera directement inséré soit sous forme de son glyphe (c'est-à-dire « l'image » du caractère), soit sous forme d'un code numérique de référence

appelé NCR (voir le chapitre 2) décimal ou hexadécimal.



L'encodage utilisé est précisé dans la déclaration XML du document, c'est-à-dire dans la toute première ligne du document XML :

```
<?xml version="1.0" encoding="UTF-8"?>
```

UTF-8 correspond à l'encodage Unicode standard. On trouve également deux autres encodage : ISO_8859-1 (dit « Latin 1 ») et ASCII. Dans ce dernier cas, le texte ne comportera aucun caractère accentué, ceux-ci étant remplacés par l'éditeur XML par leur valeur hexadécimale : le caractère accentué « é » sera ainsi remplacé par le NCR `	`.

UTF-8

```

1: <?xml version="1.0" encoding="UTF-8"?>
2: <!DOCTYPE article SYSTEM ".../dtd/v1.2/article_v1.2.dtd">
3: <?xml-stylesheet type="text/css" href=".../dtd/v1.2/oxygen_article_v1.2_main.css"?>
4: <article language="FR">
5:   <front>
6:     <title>Miles Davis (1926 - 1991)</title>
7:     <subtile>
8:       <figure>
9:         
11:     </subtile>
12:     <p><b>Miles Dewey Davis III</b> (26 mai 1926, Alton, Illinois -28 septembre 1991, Santa
13:       Monica, Californie) est un compositeur et trompettiste de jazz américain.</p>
14:   </front>
15:   <body>
16:     <section id=section="section1">
17:       <title id=section1_title>Une figure centrale du jazz </title>
18:       <p>
19:         <language language="EN">Miles Davis</language> fut à la pointe de beaucoup
20:         d'évolutions dans le jazz et s'est particulièrement distingué par sa capacité à
21:         découvrir et à s'entourer de nouveaux talents. Son jeu se caractérise par une grande
22:         sensibilité musicale et par une fragilité qu'il arrive à donner au son. Il a marqué
23:         l'histoire du jazz et de la musique du <sc>XX</sc><sup>eme</sup> siècle. Beaucoup de
24:         grands noms du jazz des années 1940 à 1980 ont travaillé avec lui.</p>
25:         <p>Les différentes formations de Miles Davis étaient comme des laboratoires au sein
26:         desquels se sont révélées les talents de nouvelles générations et les nouveaux
27:         horizons de la musique moderne ; on peut notamment citer Sonny Rollins, Julian
28:         « Cannonball » Adderley, Bill Evans et John Coltrane durant les années 1950. De 1960
29:         aux années 1980 ses <i>sidemen</i> se nomment Herbie Hancock, Wayne Shorter, Chick
30:         Corea, John McLaughlin, Keith Jarrett, Tony Williams, Joe Zawinul, Dave Liebman et
31:         Wayne Shorter.

```

ASCII

```

1: <?xml version="1.0" encoding="ASCII"?>
2: <!DOCTYPE article SYSTEM ".../dtd/v1.2/article_v1.2.dtd">
3: <?xml-stylesheet type="text/css" href=".../dtd/v1.2/oxygen_article_v1.2_main.css"?>
4: <article language="FR">
5:   <front>
6:     <title>Miles Davis (1926 - 1991)</title>
7:     <subtile>
8:       <figure>
9:         
11:     </subtile>
12:     <p><b>Miles Dewey Davis III</b> (26 mai 1926, Alton, Illinois -28&#x004D;septembre&#x004D;
13:       Monica, Californie) est un compositeur et trompettiste de jazz am&#x004D;ricain.</p>
14:   </front>
15:   <body>
16:     <section id=section="section1">
17:       <title id=section1_title>Une figure centrale du jazz </title>
18:       <p>
19:         <language language="EN">Miles Davis</language> fut &#x004D; la pointe de beaucoup
20:         d'&#x004D;volutions dans le jazz et s'est particuli&#x004D;rement distingu&#x004D;e; par
21:         d&#x004D;couvrir et &#x004D;s'entourer de nouveaux talents. Son jeu se caract&#x004D;
22:         sensibilit&#x004D; musicale et par une fragilit&#x004D;e; qu'il arrive &#x004D; donner a
23:         l'histoire du jazz et de la musique du <sc>XX</sc><sup>eme</sup> si&#x004D;cle. Beaucoup
24:         grands noms du jazz des ann&#x004D;es 1940 &#x004D; 1980 ont travaill&#x004D;e; avec lui.
25:         <p>Les diff&#x004D;erentes formations de Miles Davis &#x004D;taient comme des laboratoires
26:         desquels se sont r&#x004D;v&#x004D;l&#x004D;es les talents de nouvelles g&#x004D;n&#x004D;
27:         horizons de la musique moderne&#x004D;e; on peut notamment citer Sonny Rollins, Julian
28:         &#x004D; Cannonball&#x004D; Adderley, Bill Evans et John Coltrane durant
29:         aux ann&#x004D;es 1980 ses <i>sidemen</i> se nomment Herbie Hancock, Wayne Shorter, Chic
30:         Corea, John McLaughlin, Keith Jarrett, Tony Williams, Joe Zawinul, Dave Liebman et
31:         Wayne Shorter.

```

Tous
caractères
accentués
en NCR

Saisir des entités

Les entités (voir chapitre 2) sont introduites soit au niveau de certains caractères pour lesquels on veut s'assurer un contrôle précis, soit pour factoriser des portions de texte que l'on désire voir figurer de manière homogène dans le document.

Au niveau des caractères, on saisira sous forme d'entités les insécables et les fines. La DTD référencera ces entités de la manière suivante :

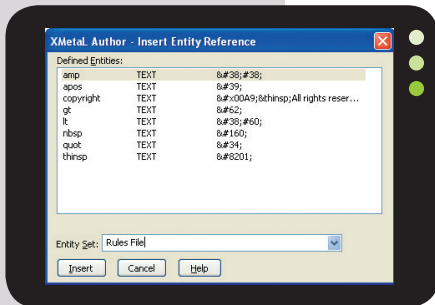
```
<!ENTITY thinsp "&#8201;" >
<!ENTITY nbsp "&#160;" >
```

À NOTER

Lors de la visualisation, sera remplacé par ; c'est-à-dire par son NCR décimal.

L'éditeur XML dispose d'un moyen d'insertion de ces entités. Pour XMetaL, il s'agit du menu Insert>Entity Reference qui donne la liste de toutes les entités déclarées dans la DTD ainsi que

des entités génériques pour différents caractères réservés (&, <, >).



● L'insertion d'une entité dans l'éditeur XMETAL

Quel que soit l'outil utilisé, c'est bien l'entité (c'est-à-dire) qui est insérée dans le document XML et non pas sa valeur déclarée dans la DTD, qui peut d'ailleurs être modifiée en fonction des besoins de production. On effectuera ainsi une substitution dans la déclaration de la DTD afin de rendre visible des insécables lors de l'épreuve. On déclarera par exemple :

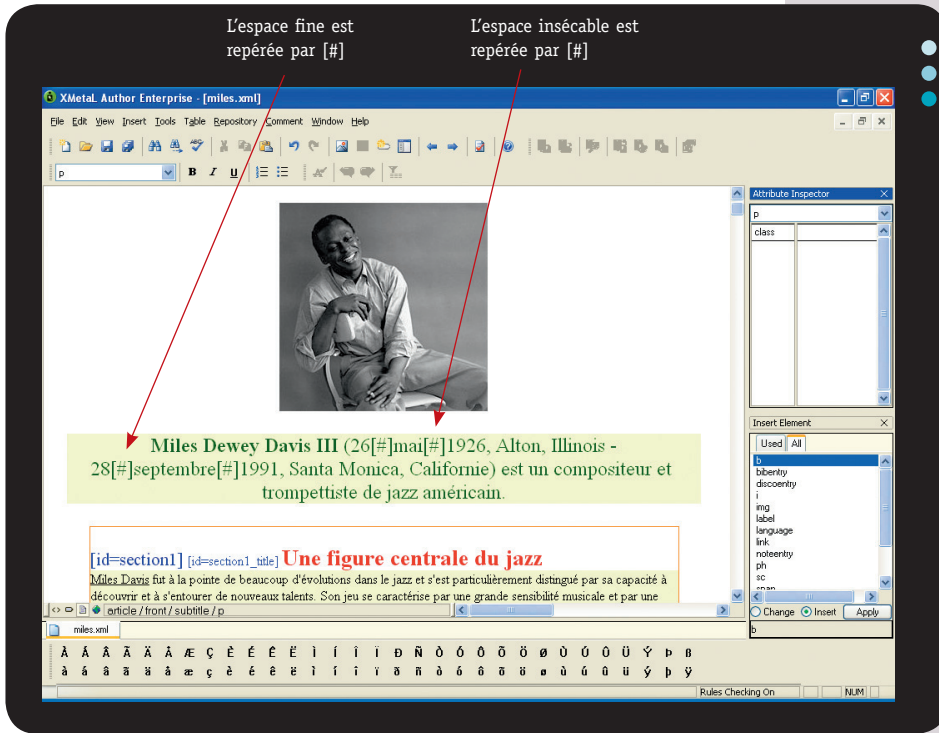
```
<!ENTITY thinsp " [#] " >
<!ENTITY nbsp " [_] " >
```

Lors de la visualisation sans balise, les caractères [#] et [] apparaîtront dans le document XML en remplacement des entités (on dit que l'on «résout les entités»), ce qui permettra de contrôler la bonne pose de ces entités.

```
<p><b> Miles Dewey Davis III</b>
(26&thinsp;mai&nbsp;1926, Alton, Illinois -
28&thinsp;septembre&nbsp;1991, Santa
Monica, Californie) est un compositeur et
trompettiste de jazz américain.</p>
```

L'insertion d'une entité correspondant à un texte factorisé suit le même cheminement. Dans la DTD, l'entité copyright sera déclarée ainsi :

```
<!ENTITY copyright "&#x00A9;&thinsp;All
rights reserved">
```



pour une utilisation dans le texte de la manière suivante :

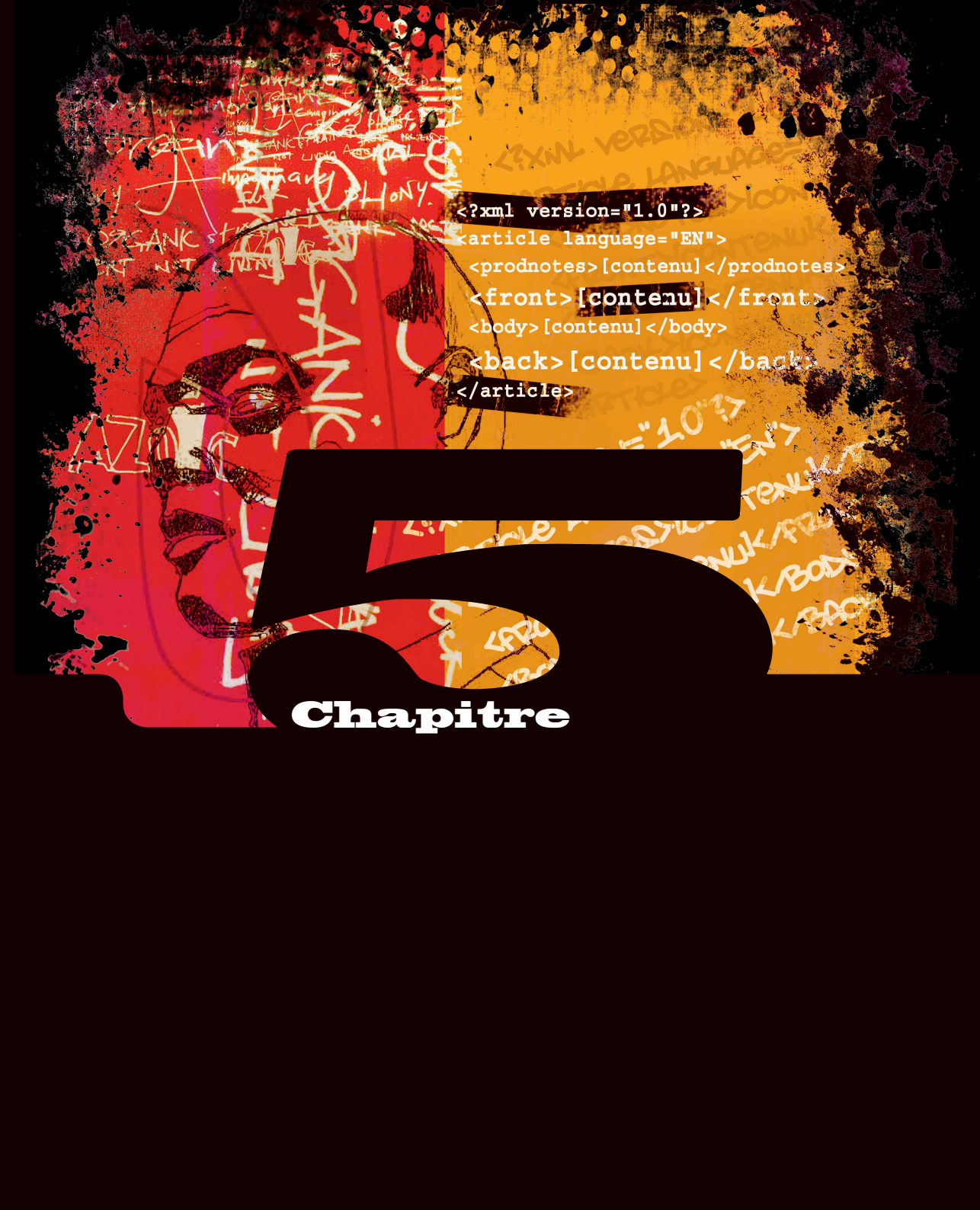
```
<p>&copyright;</p>
```

dont la restitution lors de la publication sera :

```
©[#]All rights reserved
```

On remarquera, au passage, que la déclaration de l'entité `©right;` fait appel à l'entité ` `. On veillera évidemment à ne pas croiser les déclarations !

- Visualisation des insécables et des fines dans XMetal (après modification des déclarations d'entités dans la DTD). Le lecteur correcteur peut immédiatement voir si une fine est manquante.



<?xml version="1.0"?>
<article language="EN">
 <prodnotes>[contenu]</prodnotes>
 <front>[contenu]</front>
 <body>[contenu]</body>
 <back>[contenu]</back>
</article>

5

Chapitre

Préparer et diriger la saisie XML

La saisie XML par l'auteur, rêve de tout éditeur, n'est pas pour tout de suite. La réalité s'appelle « Word » du côté de l'auteur, et « opérateurs de saisie XML multiclient et multi-DTD », du côté du prestataire.

La préparation et l'organisation de la saisie XML passent par des choix simples qui commencent dès la rédaction du document Word par l'auteur et se poursuivent par des consignes de saisie permettant au prestataire de minimiser les risques d'erreur.

Styler le document Word

Auteur et éditeur travaillent généralement sous Word. L'atelier de saisie utilise des outils orientés XML et rencontre des ambiguïtés structurelles dues à une interprétation du texte et au niveau disparate des opérateurs de saisie. Utiliser les fonctions de styles de Word pour décrire la structure XML peut lier ces deux mondes.

Qu'est-ce qu'un style dans Word ?

Tout traitement de texte moderne comme Word ou Open Office dispose de fonctionnalités de styles permettant sous un seul label de regrouper tous les paramètres typographiques qui pourront être appliqués à des paragraphes ou à des portions de texte. L'avantage recherché est de pouvoir changer glo-

balement l'apparence typographique de l'objet stylé : si tous les titres de niveau 1 ont reçu un style (par exemple Titre1), il sera possible en une seule opération de changer leurs attributs typographiques (retrait, taille de police, couleur...).

OPEN OFFICE

Open Office est le pendant open source de la suite Office de Microsoft. Bien que très puissante, agréable à utiliser et robuste, la suite Open Office n'a pas réussi à s'imposer auprès des auteurs et des entreprises. www.openoffice.org

Dans ses différentes versions jusqu'à 2007, Word distingue deux principaux types de styles :

- les styles de paragraphe qui correspondent à la notion de bloc en XML, c'est-à-dire un ensemble généralement composé dans un rectangle ;
- les styles de caractère qui correspondent à la notion de texte inline en XML.

Dans la suite de l'ouvrage, nous parlerons de style bloc (au lieu de style de paragraphe) et de style inline (au lieu de style de caractère) afin d'unifier le

discours entre les styles des traitements de texte, les styles CSS et les balises HTML lorsque la distinction est nécessaire.

REMARQUE

Word propose en réalité 4 types de styles : paragraphe, caractère, tableau, liste. Open Office en propose davantage.

Il est d'usage de parler de « feuille de styles » pour désigner le regroupement de tous les styles utilisés. Ce terme vient de ce que les documents Word, comme dans tout traitement de texte, peuvent s'appuyer sur un modèle externe au document – appelé *template* dans le jargon du domaine – qui regroupe ces styles. Celui-ci est créé à partir d'un document Word de départ comportant tous les styles et sauvegardé sous forme de modèle. Il porte l'extension .dot qui signifie : *document template*.

Le rattachement du modèle au document courant se fait par une commande du type Outils>Modèles et compléments qui diffère d'ailleurs d'une version de Word à l'autre...

Objectifs du stylage

Ce qui est recherché lors d'un stylage, c'est avant tout de la productivité dans la phase de mise en pages des publications.

En première intention, il s'agit de simplifier le coulage des textes dans les outils de PAO : grâce aux styles, les attributs typographiques sont plus facilement posés, le prestataire gagne en temps de mise en pages et l'éditeur en coûts.

En seconde intention, avec l'introduction de XML, le stylage est devenu le moyen de préparer la saisie XML. Le repérage des différents composants sémantiques d'un texte, titres de niveau, paragraphes, listes, attributs typographiques inline par des styles permet en effet par un mécanisme de conversion d'obtenir l'essentiel du XML attendu.

En même temps, cette démarche laisse à l'auteur son outil de prédilection et ne perturbe pas, du moins en théorie, ses méthodes habituelles de travail.

REMARQUE

Dans la pratique, le compuscrit (texte rédigé sur un ordinateur) fourni par l'auteur est rarement conforme à la feuille de styles, ce qui exige un travail de nettoyage a posteriori, faisant perdre une partie du bénéfice attendu du stylage.

Les limites du stylage à vocation XML

Le stylage à vocation XML présente une première limitation majeure : l'imbri-
cation des éléments à la manière des poupées russes, essentielle en XML, est
impossible au niveau des styles bloc (c'est-à-dire des styles de paragraphe).
Exprimer qu'un paragraphe appartient à un contexte de division en termes de
style se pose de la manière suivante : si l'on appelle `div` le style de division
et `p` le style du paragraphe standard, il faudrait pouvoir exprimer `div p`. La
seule manière de le faire est de créer un nouveau style bloc appelé par exem-
ple `div-p`. En admettant que la `div` puisse recevoir des paragraphes, des
listes, d'autres `div`, la combinatoire devient explosive. Le nombre de styles
est excessif et leur utilisation devient illusoire.

Le stylage présente une deuxième limitation : l'utilisateur du traitement de
texte peut à tout moment « surcharger » un style. Cela signifie qu'un style peut
être modifié localement pour répondre à un besoin éditorial. Un titre stylé
`Titre1` peut tout à fait être surchargé avec des paramètres typographiques
du style `Titre2` pour exprimer l'apparence d'un titre de second niveau :
le traitement de texte l'acceptera techniquement et le lecteur du document
interprétera cela comme un titre de niveau 2.

La troisième limitation des styles est leur extrême perméabilité à l'importation
d'autres styles. Il suffit de faire l'expérience à partir d'un copier/coller depuis
un document HTML pris n'importe où sur Internet. Après collage, la liste des
styles Word s'est enrichie (!) de nouveaux styles aux noms imprévisibles issus
des attributs insérés dans les balises HTML du texte importé.

La quatrième limitation est l'absence de règles de structure : il est tout à fait
possible de mettre un titre de niveau 2 (style `Titre2`) dès le début d'un
document Word, tandis que la structure XML interdirait ce cas de figure.

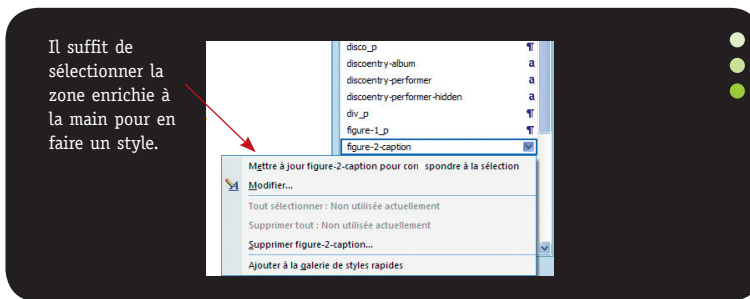
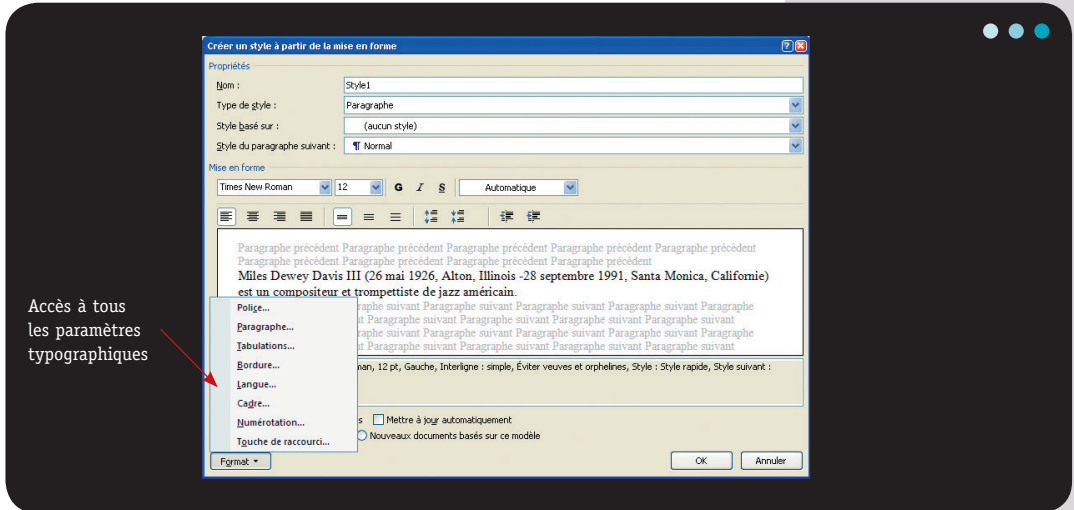
La cinquième limitation est le rendu différent de la même feuille de styles
dans les différentes versions de Word. Ce n'est pas très gênant, sauf pour
l'auteur qui se trouve confronté à une réalité différente visuellement de celle
proposée par les consignes d'écriture fournies par son éditeur.

Déclarer un style dans Word

Word, comme Open Office, propose deux manières de déclarer un style :

- directement, via des boîtes de dialogue permettant de choisir de manière
précise les attributs typographiques : police, taille, espace avant, espace
après... Cela impose une connaissance fine des paramètres typographiques
et une maîtrise des boîtes de dialogue correspondantes dans Word ;

- indirectement, via une mise en forme préalable et manuelle de l'objet à styler (paragraphe ou caractères), en le sélectionnant et en lui attribuant a posteriori un nom de style : c'est la méthode la plus intuitive, car guidée par une approche visuelle mais moins précise.



● Définition directe d'un style (Word 2007)

● Définition indirecte d'un style en s'appuyant sur une mise en forme préalable.

Nommer un style

Le nommage des styles est laissé à l'initiative de la personne qui manipule le traitement de texte. Dans une double perspective de préparation du texte au passage XML et de productivité, il convient cependant de respecter quelques règles :

- les noms de styles doivent être assez proches du nom de l'élément XML qui leur correspondra. On choisira ainsi le nom `bib_p` pour exprimer un paragraphe bibliographique dont l'élément XML est `pbib`;

- les noms de styles doivent incorporer, quand cela est perçu comme nécessaire ou utile, le pseudo chemin représentatif de la position dans la structure. On nommera ainsi `bibentry-booktitle` le titre d'un ouvrage dans une bibliographie, dont le correspondant XML est `booktitle`;
- les noms de styles sont accessoirement munis d'un suffixe pour exprimer le fait qu'il s'agit d'un style de paragraphe ou d'un style de caractère. On choisira par exemple `_p` pour les styles de paragraphe et `_c` (ou pas d'extension du tout) pour les styles de caractère.

L'intérêt de suffixer par `_p` les styles de paragraphe est de rappeler à l'utilisateur qu'il s'agit d'un style bloc lorsqu'il fait son choix dans la liste déroulante des styles. Certes, Word propose un petit indicateur de nature de style (à droite du nom de style), mais peu d'utilisateurs le connaissent ou le voient.

DES STYLES PARTOUT

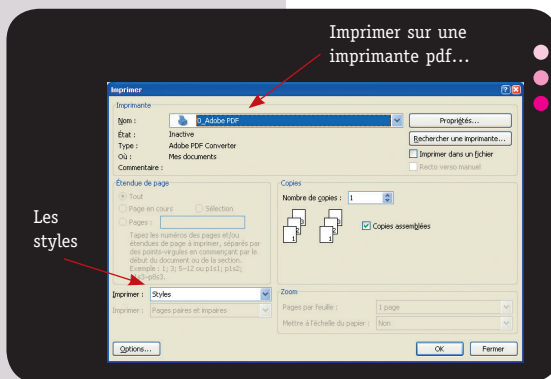
Encore une polysémie redoutable : il est difficile de s'y retrouver lorsque l'on parle de styles. Il existe le style du traitement de texte, celui de la CSS, celui des feuilles XSL, le style de l'auteur... Un effort intellectuel à consentir !

Par déduction, la plupart des autres styles ne sont pas suffixés. Il s'agit soit de styles de caractère, soit de styles réservés de Word, en l'occurrence les styles de niveau de titres. En effet, ceux-ci sont

bien connus des utilisateurs et sont automatiquement interprétés par les Word localisés, c'est-à-dire en d'autres langues que le français (l'ouverture de la feuille de styles dans une traduction américaine de Word remplacera `Titre1` par `Head1`, par exemple).

- Récupérer la liste des styles du fichier ouvert

Lister les styles de Word

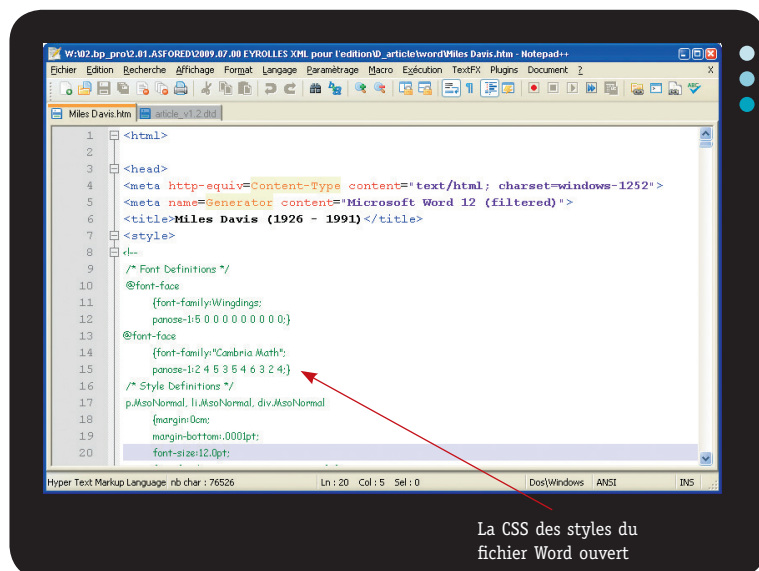


Il est fort pratique de disposer d'une liste des styles lorsque l'on veut gérer une feuille de styles de référence. Word ne propose pas un export direct : dans Word 2003 (la procédure est proche pour les autres versions), il faut passer par la commande `Fichier>Imprimer` et choisir `Imprimer:Styles` dans la liste déroulante. Le plus simple est d'imprimer en PDF afin de récupérer la liste des styles par copier/coller, soit dans votre traitement de texte, soit dans Excel.

Une autre méthode, plus technique, consiste à enregistrer le document Word en format «Page

Web, filtrée» et à ouvrir le HTML correspondant avec un éditeur de texte standard (ex. : Notepad ++). Les styles sont déclarés sous forme d'une CSS

intégrée en début de document Word. Il peuvent même être édités : on éliminera ceux qui ne servent pas, à l'exception des titres de Word qui sont des styles réservés, afin de disposer d'un modèle épuré.



● La CSS du fichier Word en mode Page Web filtrée

Enfin, dans Open Office, il suffit de récupérer le XML des styles (fichier style.xml) stocké dans le fichier .odt et d'extraire les styles intéressants, tout en nettoyant au passage ceux qui sont inutiles.

Gérer les feuilles de styles

Peut-on imaginer une seule feuille de styles pour toute la maison d'édition ? Oui, si la production est homogène : une maison qui ne fait que du roman pourra facilement suivre cette voie. Une maison aux collections diverses devra concevoir plusieurs feuilles, en gardant en commun tout ce qui peut être partagé : les titres, etc.

La gestion des feuilles de styles se doit par ailleurs d'être centralisée : une seule référence doit être disponible sur le réseau de l'entreprise, assortie de sa documentation. Une seule personne doit en être responsable. Dans la pratique, cette contrainte est rarement respectée, la feuille de styles circulant dans l'entreprise et évoluant au gré des besoins, ce qui rend l'approche nettement moins industrielle.

Utiliser les styles pour produire du XML

Utiliser les styles pour faciliter la production du XML depuis Word nécessite la prise en compte de deux paramètres : la facilité de pose des styles et leur configuration technique afin que le XML produit soit le plus proche possible du XML final.

Identification des zones à styler

Le but du stylage est de procéder à une conversion automatique de Word vers un XML brut, brut parce qu'il n'est pas possible sous Word de styler toute la finesse du XML cible, aussi travaille-t-on de manière plus grossière :

- stylage rapide sous Word, soit au stade de la rédaction par l'auteur, soit au stade du préprocessing de mise en pages chez le compositeur ;
- exportation/conversion en XML brut en aval.

Cette contrainte admise, on va chercher à styler l'essentiel de la structure, c'est-à-dire les trois grandes parties d'un article ou d'un ouvrage (*front*, *body*, *back*) et quelques parties plus laborieuses à traiter en XML, comme les bibliographies. Le travail s'effectue en trois étapes :

- définition de la structure XML intermédiaire à produire ;
- définition des styles blocs ;
- définition des styles inline, en prenant comme règle de ne pas dépasser trois niveaux d'emboîtement dans les cas les plus pointus.

La structure intermédiaire doit être pensée en compatibilité avec la structure XML finale. Un opérateur devra en effet ultérieurement ouvrir le XML brut avec la DTD du XML final afin d'effectuer le raffinement attendu : déplacement d'éléments, pose d'attributs, balisage plus fin (« surbalisage »).

Dans le cas de `article_v1.2.dtd`, on se limitera aux structures hiérarchiques suivantes :

- les blocs `front`, `body` et `back` que l'on identifiera par une séparation visuelle à l'aide de styles de paragraphes;
- les blocs de structuration du `front`, en repérant le titre de l'article, le sous-titre, le surtitre et le bloc des auteurs;
- les blocs des paragraphes bibliographiques, discographiques et filmographiques;
- un seul bloc pour repérer les `div`, en laissant au surbalisage la pose des attributs;
- les titres hiérarchiques, en conservant ceux de Word (Titre1, Titre2, Titre3);
- les listes classiques et listes de définitions (`deflist`).

Toutes ces structures seront rendues par des styles de paragraphe.

Les structures inline sont analysées en visant à ne conserver que des structures simples et en n'autorisant les imbrications que dans des contextes précis : note, liste de définitions, bibliographie, discographie, vidéographie.

Nommage, choix typographiques et pose des styles

Le nommage des styles n'est pas un processus anodin : de leur choix va dépendre la productivité de l'opérateur. Un principe de base consiste à regrouper les styles correspondant à un élément à baliser : tout ce qui concerne une entrée bibliographique devra être listé de manière contiguë dans la liste des styles de Word. Il faut donc préfixer le style et éventuellement ajouter une numérotation interne pour que le tri se fasse de manière satisfaisante.

Pour la bibliographie (élément inline `<bibentry>` dans le XML final), on choisira par exemple le nommage suivant pour les styles inline :

<code>bibentry-articletitle</code>	repérage d'un titre d'article de revue
<code>bibentry-booktitle</code>	repérage d'un titre d'ouvrage
<code>bibentry-journaltitle</code>	repérage d'un titre de revue

Le choix typographique sera dicté par des considérations simples :

- lorsqu'il s'agit d'un englobant inline (c'est-à-dire un style inline qui en contiendra d'autres), prévoir un encadré;

DOCX : XML et XML

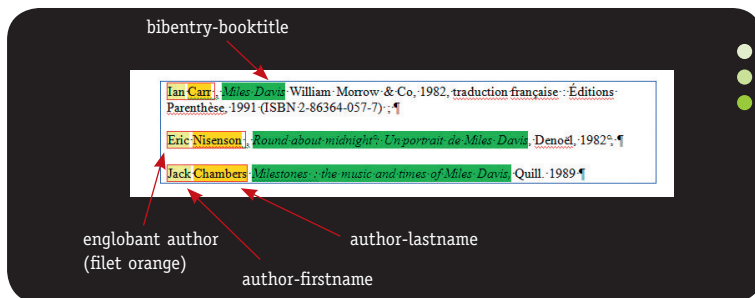
Un point important : Word dispose d'une commande `Fichier>Enregistrer sous>Format docx` qui crée un fichier XML. Ce fichier est au format XML Microsoft orienté mise en pages : il reflète la position des éléments sur la page et ne produit pas le XML conforme à la DTD attendue. Il est toutefois possible de l'utiliser pour une conversion vers le XML brut, à condition d'avoir préalablement stylé le document.

Remarque

Ces quelques règles, en jouant le plus logiquement possible avec les couleurs, donnent un rendu suffisamment discriminant pour que l'utilisateur pose rapidement les styles.

● Rendu du stylage d'une bibliographie

- lorsqu'il s'agit d'un titre hiérarchique (Titre1, etc.), d'un englobant bloc (figure, div) prévoir également un encadré;
- les styles ayant un caractère sémantique commun sont exprimés par une couleur de fond appartenant à une même gamme. Le prénom d'un auteur sera en jaune clair et le nom en orange. Il n'est pas toujours possible de le faire, mais c'est la démarche qu'il faut retenir.



La pose des styles s'effectue en utilisant les raccourcis de saisie permis par Word, en particulier la touche Contrôle (CTRL) qui permet d'effectuer des sélections disjointes multiples. La saisie des styles d'une deflist se fera donc selon la séquence suivante :

- stylage de tout le bloc deflist avec le style deflist_p;
- stylage de toutes les entrées (via la touche CTRL) avec le style deflist-1-defterm;
- stylage de toutes les définitions (via la touche CTRL) avec le style deflist-2-defdata.

● Stylage des deflist



On remarquera que le choix des styles se traduit par une présentation dans cet ordre logique au niveau de la liste des styles : il y a donc moins de mouvements de souris à prévoir et surtout une perte de temps largement diminuée par rapport à un nommage arbitraire. Le positionnement dans la liste peut se faire en rentrant la première lettre du regroupement de styles.

Dans le cas de la feuille utilisée pour article_v1.2, il suffit de retenir 8 entrées :

- 0 pour le découpage hiérarchique et front ;
- a pour authors ;
- b pour bib ;
- d pour deflist, disco et div ;
- f pour figure ;
- l pour link et listitem ;
- t pour Titre (mais l'usage est d'utiliser les raccourcis clavier) et typo ;
- v pour video.

À NOTER

Avec un peu d'habitude, ou un Post-it collé sur l'ordinateur, le stylage est extrêmement rapide.

La liaison style Word/XML

Les différents styles définis et testés, il reste à établir le tableau de correspondance des noms de styles avec la structure XML correspondante. Cette correspondance sera utilisée par le convertisseur Word vers XML pour produire le flux XML brut. En pratique, la correspondance se limite à l'association du nom de style aux balises ouvrantes attendues pour le flux XML, à charge pour le convertisseur d'effectuer les emboîtements corrects de balises.

Remarque

L'écriture des convertisseurs nécessite un savoir-faire particulier qui sort du cadre de cet ouvrage.

- La correspondance Styles/Structures XML (Excel)

	A	B	C	D	E
3	x	01 0 front	<front>		
4	x	01 1 front_article_suptitle_p	<titles><subitle>		
5	x	01 2 front_article_title_p	<titles><title>		
6	x	01 3 front_article_subitle_p	<titles><subitle>		
7	x	01 4 front_article_authors_p	<authors>		
8	x	02 body	<body>		
9	x	03 back	<back>		
10	x	author	<author>		
11	x	author-affiliation	<affiliation>		
12	x	author-firstname	<authorfirstname>		
13	x	author-lastname	<authorlastname>		
14	x	bib_p			
15	x	bibentry-articletitle	<articletitle>		
16	x	bibentry-booktitle	<booktitle>		
17	x	bibentry-journaltitle	<journaltitle>		
18	x	deflist_p	<deflist><drow>		
19	x	deflist-1-defterm	<defterm> (deuxième cellule)		
20	x	deflist-2-defdata	<defdata> (première cellule)		
21	x	disco_p	<pdisco>		
22	x	discoentry-album	<album>		
23	x	discoentry-performer	<performer>		
24	x	discoentry-performer-hidden	<performer hidden="yes">		
25	x	div_p	<div>		
26	x	div_p	<div> (div dans contexte front)		

Écrire les consignes de saisie

Des consignes de saisie écrites par l'éditeur sont pratiques dans le cas d'une saisie primaire et indispensables dans le cas d'une rétroconversion : il s'agit de rendre le prestataire le plus autonome possible dans ses choix et de minimiser le nombre d'allers-retours prestataire-éditeur sur des points de structure.

Que contiennent les consignes de saisie ?

CAPITALES OU BAS DE CASSE ?

Un autre exemple concerne la saisie de titres présentés en capitales dans le document source : il est en pratique préférable de les saisir en bas de casse accentués, quitte à restaurer la présentation initiale par une feuille de styles CSS à l'affichage. Il y a en effet plus de sémantique dans un texte accentué que dans un texte en capitales désaccentuées.

L'objectif premier des consignes de saisie est de minimiser les risques d'erreurs lors du balisage XML. Il ne s'agit pas d'erreurs de base triviales comme l'introduction d'un élément non permis par la structure : on attend en effet du prestataire une connaissance suffisante de XML et qu'il valide avant livraison le document vis-à-vis de sa DTD.

Différents types d'erreurs risquent de se produire :

- les erreurs d'interprétation du texte source qui conduisent à appliquer un élément pour un autre ;
- les erreurs de choix de balise quand la DTD permet différentes possibilités.

Un exemple du premier cas est celui d'un balisage sous forme d'italique (élément `<i>`) pour le titre d'un ouvrage cité dans une bibliographie, tandis que l'attente est une balise sémantique `<booktitle>`.

Un exemple du second cas est celui d'un balisage de la source d'une citation quand la DTD prévoit à la fois un élément `<source>` et un élément

`<p class="source">` : l'opérateur peut librement choisir entre l'une et l'autre. Ce cas est très fréquent avec les DTD très riches en éléments (ex. : DocBook) lorsque celles-ci ne sont pas utilisées de manière restrictive, c'est-à-dire en ayant pris soin d'enlever les éléments jugés non utiles au type d'ouvrage considéré.

Le deuxième objectif des consignes de saisie est de lever des interrogations sur des cas particuliers pour lesquels on dispose d'une réponse : une légende qui se trouve partiellement intégrée à la figure dans le texte source (parce que l'opérateur PAO a jugé cette approche plus esthétique pour la page) pourra avoir été répertoriée dans les consignes. Celles-ci indiqueront la marche à suivre :

- soit extraction de la légende pour la placer dans la balise `figurecaption`, ce qui suppose une retouche de l'image si l'on n'est pas parvenu à l'extraire proprement du PDF source en cas de rétroconversion ;
- soit intégration dans l'image au prix d'un détournement plus large permettant d'englober ladite légende.

Qui doit écrire les consignes ?

Le premier jeu de consignes est souvent écrit par le prestataire. Mais il appartient au fabricant ou à l'éditeur de l'affiner afin de répondre d'abord aux besoins éditoriaux de l'entreprise. Ceux-ci se traduisent par des questions simples :

- comment baliser tel ou tel élément ?
- comment lever une question d'ambiguïté structurelle ?

Comment rédiger les consignes ?

Les consignes de saisie poursuivent différents objectifs :

- servir de référence à un acteur (opérateur ou éditeur) pour contrôler l'orthodoxie d'une saisie ;
- servir de support de formation pour un nouvel opérateur ou un nouvel éditeur.

Le premier objectif conduit à des consignes portant uniquement sur les points difficiles ou ambigus : dans l'exemple `d'article_v1.2.dtd`, ce sera la description illustrée de la manière de styler une bibliographie, une discographie, une vidéographie ou encore une liste de définitions. Dans ce cas, un classement alphabétique et une sélection des seuls éléments renseignés dans les consignes semblent constituer la meilleure méthode.

Le second objectif milite pour une description plus arborescente de la structure, en partant du plus haut niveau et en recherchant l'exhaustivité. En pratique, on part des consignes destinées aux opérateurs en les réarrangeant pour les rendre accessibles.

Les consignes de saisie peuvent concerner uniquement la saisie XML à partir d'une source qui n'est pas un document Word stylé (cas de la rétroconversion). Dans ce cas, on se contente de puiser des exemples dans les documents sources et d'y associer le flux XML attendu, éventuellement complété par une explication.

Les consignes peuvent également concerner le stylage sous Word. L'approche est comparable, à la différence que l'exemple est un extrait du document Word stylé, complété éventuellement par l'expression du XML attendu.

● Consigne de stylage du Word pour les paragraphes discographiques avec la correspondance XML

The screenshot shows a Word document with a list of discographies. The list items are: 1948: Miles Davis Cool Boopin', 1949: Miles Davis Birth of the cool, sorti en 1959, 1951: Miles Davis Blue Period, and 1951: Miles Davis Dig. Red arrows point from labels 'discoentry-performer-hidden', 'discoentry-album', and 'discoentry-performer' to the respective parts of the list items. Below the list, an XML code block shows the structure: <pdisco> <discoentry>1948 : <performer hidden='yes'>Miles Davis</performer> <album>Cool Boopin'</album> </discoentry> </pdisco> <pdisco> <discoentry>1949 : <performer hidden='yes'>Miles Davis</performer> <album>Birth of the cool</album>, sorti en 1959 </discoentry> </pdisco> <pdisco> <discoentry>1951 : <performer>Miles Davis</performer> <album>Blue Period</album> </discoentry> </pdisco> <pdisco> <discoentry>1951 : <performer hidden='yes'>Miles Davis</performer> <album>Dig</album> </discoentry> </pdisco>

Organiser la saisie XML

L'écosystème XML fait intervenir des prestataires qui eux-mêmes pilotent des sites de production offshore. La copie arrivant en continu, des problèmes d'interprétation se posent lors du balisage XML : quelle balise utiliser pour quelle information, comment gérer les exceptions ?

Désigner un responsable de DTD

La DTD, bible de la structure, a une importance stratégique. Tout le dispositif de production des publications repose sur elle. Dès lors, elle doit être gérée comme telle : un actif stratégique qui ne peut être modifié que par une personne clé.

EN PRATIQUE

Ce pourra être un responsable de l'informatique éditoriale ou, si ce service n'existe pas, un éditeur. Les décisions de modification devront être prises en connaissance de cause.

Toute modification de DTD mise en production devra être tracée. Dès lors, l'entête du document XML (et donc la DTD devra le prévoir) comporte un élément obligatoire recueillant le numéro de la version de DTD.

Gérer les exceptions

La première réaction face à un cas non répertorié est de modifier la DTD de manière à autoriser l'exception : cela se traduit presque toujours par un relâchement de la structure, relâchement qui pose rapidement des problèmes d'automatisation de la mise en média en raison d'une combinatoire explosive.

REMARQUE

Quel que soit le niveau de détail et de sophistication de la DTD, des exceptions apparaîtront toujours au cours de l'exploitation.

Un exemple éclairera le risque apporté par cette démarche. Dans `article_v1.2.dtd`, les préliminaires sont indiqués comme suit :

```
<!ELEMENT front (suptitle?, div*, title,
                 subtitle?, authors?, keywords?, div*)>
```

Une application de mise en pages pourrait respecter l'ordre des éléments : le titre (`title`) est immédiatement suivi du sous-titre (`subtitle`) et du bloc auteurs (`authors`).

La demande d'une composition ponctuelle faisant apparaître directement les auteurs avant le sous-titre ne peut être obtenue sans modification de la DTD. Cette modification peut se faire de deux manières :

- soit en mettant un attribut par exemple `shift="yes"` sur `authors` exprimant que ce bloc devra être déplacé à la mise en média ;
- soit en rendant la DTD plus laxiste en autorisant le mélange `subtitle` et `authors`.

Dans le premier cas, la DTD s'écrirait :

```
<!ELEMENT front (suptitle?, div*, title,
                 subtitle?, authors?, keywords?, div*)>
<!ATTLIST authors shift (yes) #IMPLIED>
```

La saisie est complexifiée, car il faudra vérifier à chaque document s'il y a lieu ou non de poser cet attribut. Par ailleurs, cette approche introduit de l'information de mise en pages dans le flux XML, ce qui n'est pas une bonne pratique. XML a été conçu justement pour séparer la forme du fond !

Dans le second cas, la DTD devient :

```
<!ELEMENT front (suptitle?, div*, title,
                 (subtitle|authors)*, keywords*, div*)>
```

Cette écriture signifie que l'on peut mélanger les `subtitle` et les blocs `authors`, ce qui autorise au sein de `front` la présence de plusieurs `subtitle` éventuellement intercalés d'`authors`. Solution à proscrire, car les documents deviennent extrêmement permissifs en termes de structure.

On pourrait même aller plus loin en écrivant :

```
<!ELEMENT front
(suptitle|div|title|subtitle|authors|keywords|div)*>
```

Cela revient à autoriser n'importe quel élément dans n'importe quel ordre : en d'autres termes, on procède à une régression complète qui se rapproche du traitement de texte où n'importe quel style peut être introduit n'importe

où. L'aspect contraint du document disparaît et l'on s'avance vers de graves difficultés de rendu, une fois le XML transformé pour les média cibles.

Il ne faut donc pas céder face à une demande de ce type. La bonne méthode consiste à contraindre les cas particuliers à rentrer dans la structure. On explique à l'auteur que la présentation des articles est uniforme et répond à un besoin d'identification par le lecteur des informations sur une page, et que changer la géométrie de celle-ci n'est pas souhaitable.

Il est malgré tout des cas pour lesquels une modification de la DTD est légitime. On cherchera dans ce cas à rester compatible avec l'existant, c'est-à-dire à n'introduire que des éléments ou des attributs facultatifs : par définition, ceux-ci ne peuvent apparaître que dans les nouveaux documents ; tous les anciens documents parsent avec la nouvelle DTD, mais un document créé avec la nouvelle DTD peut ne pas parser avec l'ancienne. En introduisant, par exemple, l'attribut `lang` pour un élément `article` de la manière suivante :

```
<!ELEMENT article (contenu)>
<!ATTLIST article lang (EN|FR) #IMPLIED>
```

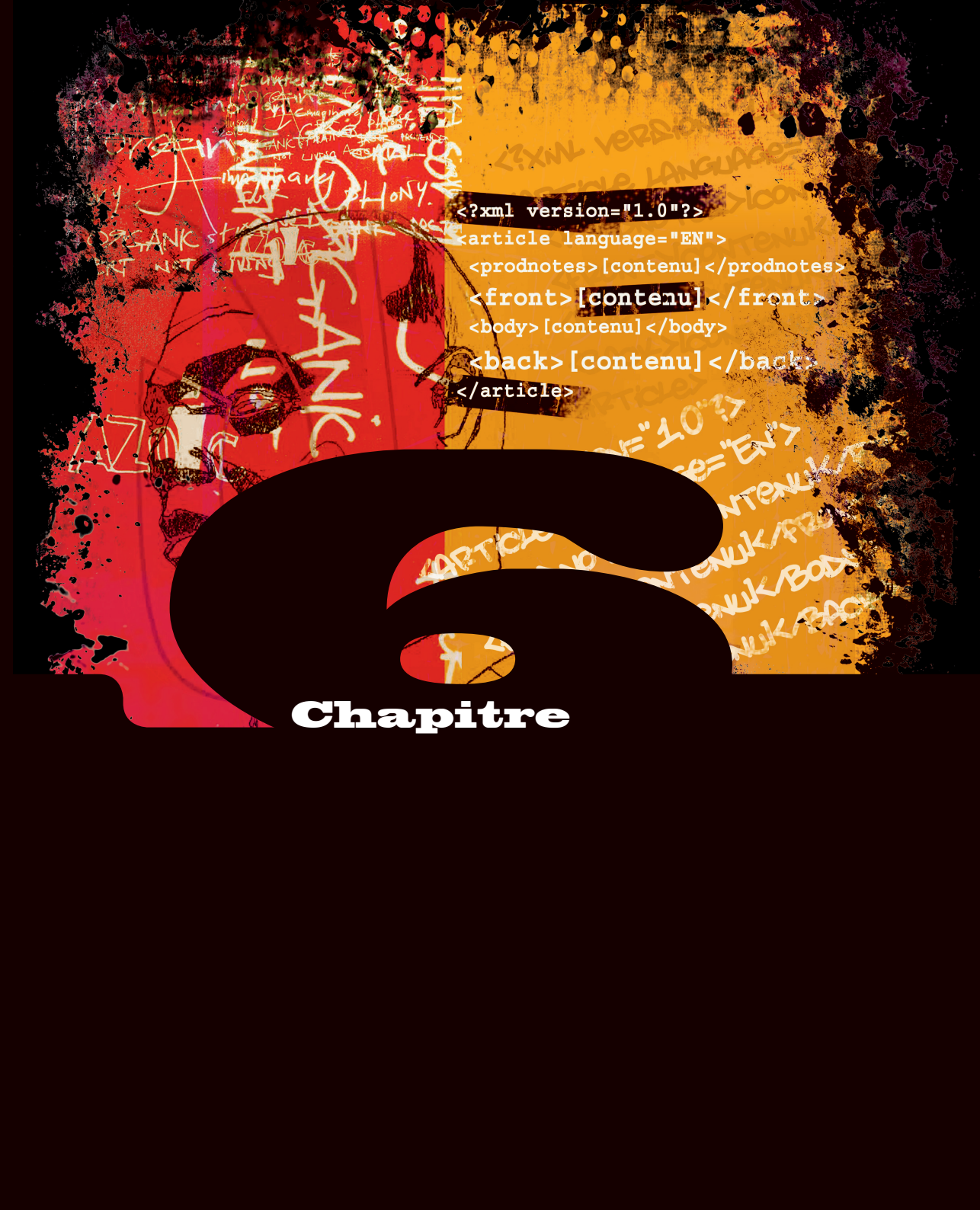
il est clair que tous les documents existants parseront avec la nouvelle DTD puisque l'attribut `lang` additionnel n'est pas obligatoire.

En revanche, si le nouveau document XML porte l'attribut `lang` sous la forme `<article lang="FR">`, il ne parsera pas avec l'ancienne DTD puisque celle-ci n'autorise pas cet attribut. Cela peut sembler évident, mais le problème se pose souvent...

À NOTER

Les modifications de DTD doivent être répercutées dans les consignes de saisie et plus généralement la chaîne de traitement : c'est tout sauf anodin !





```
<?xml version="1.0"?>
<article language="EN">
<prodnotes>[contenu]</prodnotes>
<front>[contenu]</front>
<body>[contenu]</body>
<back>[contenu]</back>
</article>
```

Chapitre



Éprouver le XML



La DTD commentée et les consignes de saisie renseignées ne suffisent pas à assurer un XML en tout point conforme aux attentes de l'éditeur : le contrôle doit porter sur le texte, la conformité des structures XML, la pose des entrées d'index et des liens, la taille et la qualité des images, l'aspect des tableaux... Des techniques d'épreuve doivent dès lors être mises en œuvre.



À quelle étape faire intervenir le XML ?

La production la plus contraignante est la mise en pages papier. La créativité graphique et les usages typographiques et de présentation compliquent la XMLisation qui peut avoir lieu avant, pendant ou après la mise en pages selon la complexité de l'ouvrage, le délai octroyé et la culture numérique de la maison d'édition.

Production XML en amont

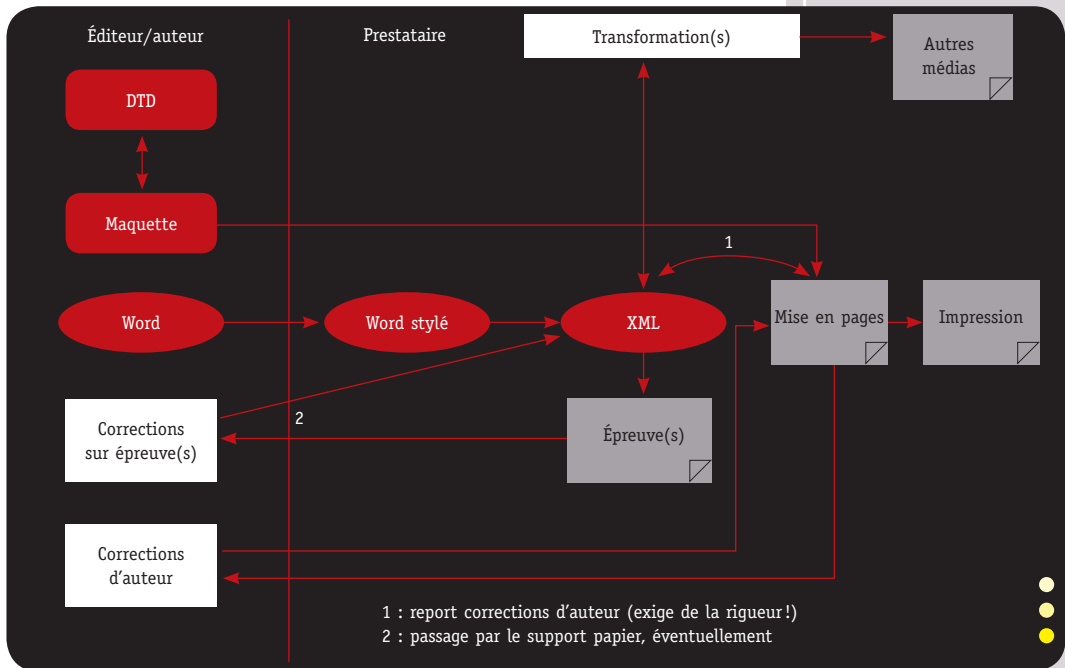
Générer le XML en amont constitue le cas théorique idéal : produire un seul fichier et travailler au niveau de l'éditorial jusqu'à ce qu'il soit sémantiquement correct pour servir de source à toutes les productions, semble en effet être la meilleure solution pour diminuer les coûts dans une perspective d'édition multisupport. La mise en pages papier se fera de manière automatique ou semi-automatique – en tout cas avec une diminution des coûts suffisante pour compenser le surcoût de la XMLisation – et les productions électroniques seront totalement automatiques.

Le cas le plus simple et le plus immédiat est le roman ne présentant pas de subtilité de mise en pages ou de typographie, c'est-à-dire la grande majorité des cas : la structure est généralement très simple – des chapitres organisés en paragraphes – et l'essentiel de la mise en pages peut être automatisé.

Entrent également dans cette catégorie les essais dont la structure de chapitres est une arborescence simple : des sections décomposées en sous-sections avec une profondeur qui peut aller en pratique jusqu'au quatrième niveau.

Le circuit de production est simple.

1. Le compuscrit, sous forme de fichiers Word, est transmis au prestataire qui lui applique un style afin de produire automatiquement du XML ; celui-ci est éventuellement surbalisé pour prendre en compte les finesses structurelles non rendues par les styles de Word.



2. Le XML est ensuite éprouvé sous différentes formes en fonction de la complexité du document traité. Pour un roman, on générera une épreuve PDF standard en respectant les contraintes typographiques décidées par l'éditeur (police, tailles et espacement des caractères) et en appliquant un dictionnaire de césure de la langue du document.

3. L'éditeur corrige l'épreuve, soit sous forme papier – donc après avoir imprimé le PDF – soit sous forme électronique en commentant directement le PDF avec les outils de révision d'Acrobat.

4. Une fois les boucles de lecture/correction passées – ce qui correspond le plus souvent à deux passages – le XML est injecté dans le système de mise en pages : il s'agit soit d'un moteur de composition offrant des possibilités de retouche manuelle, soit d'un logiciel de PAO comme InDesign en ayant préalablement transformé le XML pour que celui-ci soit facilement importable dans le système.

REMARQUE

Pour un essai ou un document comportant de nombreux éléments autres que du texte pur, des tableaux, des figures et des éléments navigationnels (index), on produira des vues spécifiques facilitant la lecture/correction.

● Production du XML en amont de la mise en pages papier

5. L'éditeur contrôle la mise en pages pour les dernières corrections, permettant au prestataire de terminer le document dans sa mise en pages papier.
6. Les corrections d'auteurs sont reportées dans le fichier XML. Ce report est soit automatique, lorsque le moteur ou l'outil de PAO et la simplicité de la structure le permettent, soit manuel.

Un point important concerne la récupération du XML mis à jour : le système de mise en pages ne permet pas toujours d'exporter un XML conforme à la DTD. Si les moteurs de composition font assez bien ce travail, les outils de PAO

LES LOGICIELS D'AUTOBALISAGE

Pour un roman, l'export ne pose généralement pas de problème. Pour un essai ou un document plus complexe, on utilisera de préférence un outil de balisage XML automatique travaillant sur du PDF, à condition que les styles typographiques soient suffisamment discriminants pour que l'automate donne de bons résultats ; si la maquette a été conçue dans ce sens, la récupération est satisfaisante.

sont moins performants et ne permettent pas la récupération simple des index et des notes de bas de page. Dans ce cas, il vaut mieux reporter à la main dans le flux XML les corrections d'auteur – en principe mineures. Ce report est toujours périlleux, car rien ne permet vraiment de vérifier la cohérence du document publié avec le contenu du fichier XML.

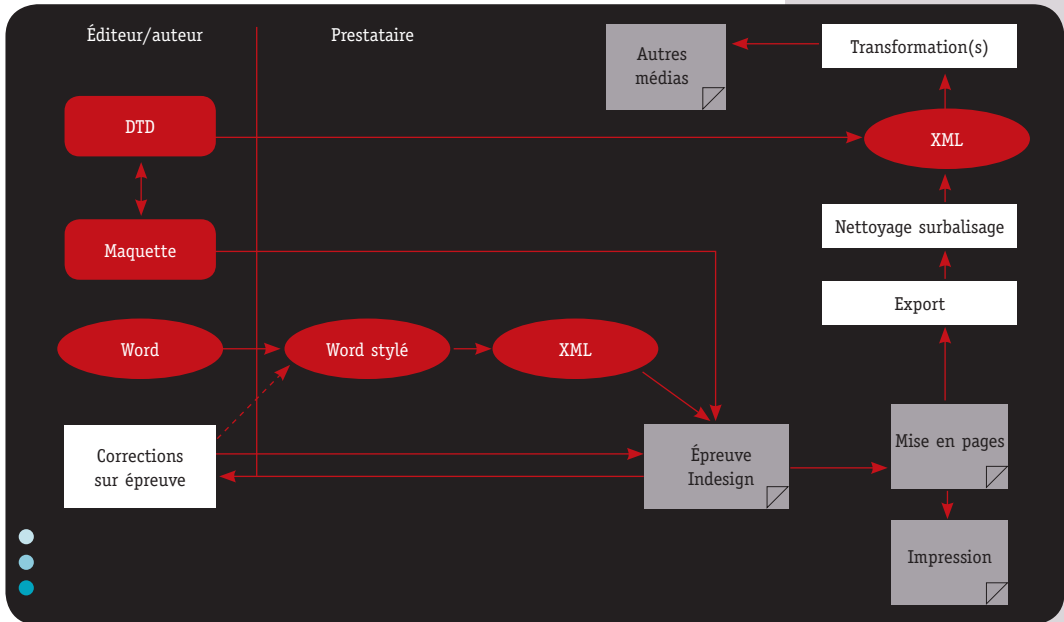
Production XML en cogénération

Les habitudes professionnelles des éditeurs ont évolué avec l'apparition des logiciels de PAO. À l'époque des photocomposeuses, ils recevaient du compositeur des placards papier sur lesquels étaient effectuées manuellement les corrections, à charge pour le compositeur de reporter celles-ci dans le flux de photocomposition. Une fois les placards validés, la mise en pages finale était lancée pour obtenir l'épreuve finale.

Les logiciels de PAO ont apporté un confort important à l'éditeur grâce à la possibilité de composer directement la mise en pages finale. Cela a eu deux conséquences :

- l'éditeur se trouve plus proche du rendu final et contrôle facilement la pagination de l'ouvrage, ce qui lui permet d'évaluer rapidement le coût de fabrication de l'ouvrage final ;
- le maquettiste peut prendre de plus en plus de liberté avec la composition « classique » : les livres deviennent de plus en plus riches graphiquement puisque les contraintes imposées par le placard disparaissent. Et de riches, ils deviennent complexes, éloignant de plus en plus le rendu visuel final d'un rendu visuel de placard, en augmentant le nombre de corrections d'auteurs, ce qui alourdit d'autant le coût final.

Avec l'arrivée du XML, ce confort n'est pas remis en question : si le XML est introduit dans la phase initiale, il faut une épreuve qui se rapproche de la



mise en pages finale. Or, les logiciels de PAO ne parviennent pas à transformer du XML natif en un rendu proche du rendu final, en particulier pour le traitement des tableaux. On procédera donc à une cogénération au prix d'un effort organisationnel important :

- la maquette doit être conçue de telle manière qu'une relative compatibilité soit installée entre la mise en pages InDesign et la DTD. Cette contrainte n'est pas toujours acceptée par les DA (directeurs artistiques) ;
- le premier traitement du compuscrit passe généralement par un document Word stylé, suivi d'une transformation XML éventuellement complétée par une retouche XML avant injection dans InDesign ;
- les retouches ultérieures s'effectuent directement dans le fichier InDesign.

La mise en pages est réalisée de manière automatisée et sert d'épreuve pour l'éditeur. Il est à noter que la pratique de la plupart des prestataires qui ne sont pas des *pure players* du numérique consiste à se limiter au seul éprouvage de la mise en pages papier, ce qui peut laisser passer de nombreuses imperfections du XML (voir plus loin « Éprouver le XML ? »).

L'étape finale de production du XML s'effectue par export du logiciel de PAO avec les limitations de celui-ci : il faudra inmanquablement surbaliser le XML d'export, en particulier rattraper les balisages de notes, d'index et de tableaux.

Production XML en aval

REMARQUE

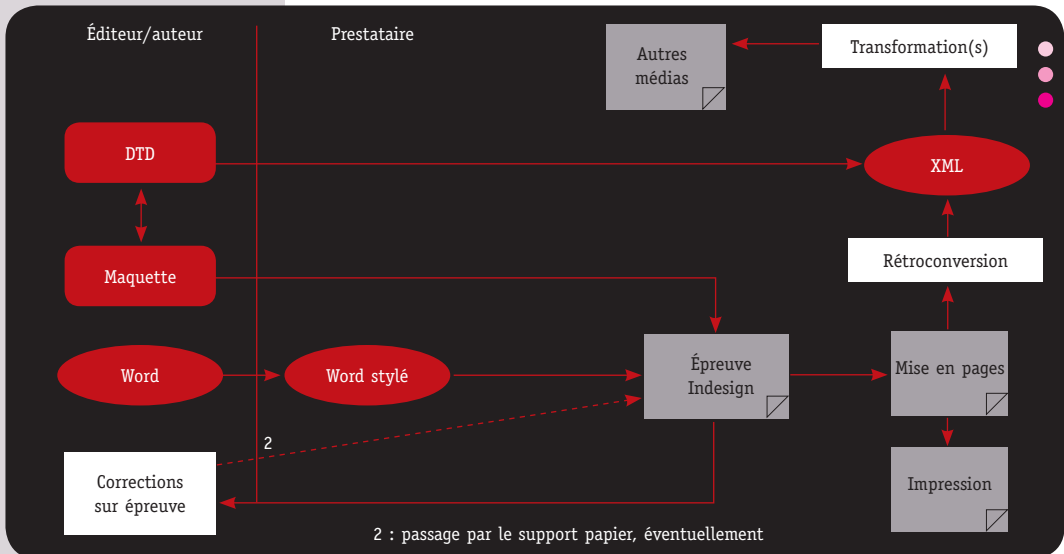
Ce cas de figure est celui du scolaire, où la mise en pages est à la fois contrainte par les habitudes – l'utilisation de la double page comme surface d'expression – et par le temps très réduit entre la mise à disposition par les pouvoirs publics des nouveaux programmes et la mise en place des spécimen par l'éditeur auprès des professeurs.

Ce cas est le plus simple du point de vue organisationnel : il revient à opérer une rétro-conversion d'un PDF sans modifier la chaîne existante.

Le circuit est tout à fait standard : le passage par le document Word stylé n'a rien d'obligatoire, mais il est souvent utilisé dans les faits.

Les corrections se font directement dans la mise en pages InDesign. La rétro-conversion intervient a posteriori, soit directement à partir du PDF avec ou sans logiciel de génération automatique d'un pré-XML. À des fins de productivité, on pourra procéder à un stylage du document InDesign qui permettra, par export, de récupérer un XML brut. Cela exige toutefois une grande rigueur que l'on n'arrive pas souvent à obtenir des prestataires, compte tenu des fortes contraintes de temps auxquelles ils sont soumis ou de la nature même du livre. Un texte de mathématiques comprenant des formules traitées en images (le travail de passage en MathML ou LaTeX reste donc à faire) ne donnera rien de satisfaisant à l'export. En pratique, mieux vaut partir du PDF.

- Production du XML a posteriori



Éprouver le XML ?

XML est plus informatif que le seul contenu lisible d'un document : il type sémantiquement les portions d'information selon les usages (ex. : pour les supports électroniques) mais est invisible dans le cadre de la mise en pages pour le médium papier. L'éprouvage XML est un moyen acceptable de contrôle du balisage par les éditeurs.

Objectifs et principes de l'éprouvage XML

L'éprouvage XML poursuit différents objectifs :

- vérifier que le contenu a correctement été saisi, ce qui est crucial dans le cas d'une rétroconversion à partir d'un original papier, du fait qu'il y a eu une étape d'OCR dont la qualité ne peut être assurée sans contrôle ;
- vérifier que les entités caractères ont été utilisées conformément aux règles édictées par l'éditeur ; il s'agit, en particulier, de s'assurer que les caractères Unicode ne remplacent pas l'entité déclarée dans la DTD ;
- vérifier que les structures XML sont correctes et que l'on n'a pas, par exemple, utilisé une balise `<i>` pour typer un nom d'ouvrage (`<booktitle>`) dans une référence bibliographique ;
- vérifier que les liens existent ou sont corrects : liens au niveau des index (création des index ou récupération des index pour le livre papier) et liens hypertextes propres aux médias électroniques ;
- vérifier que les ressources (images, formules mathématiques en LaTeX ou MathML) sont correctes ;
- vérifier que les éventuels traitements spécifiques des médias cibles (images, formules, tableaux) sont également satisfaisants.

Suivant la nature et la complexité de l'ouvrage, ces vérifications donneront lieu à la production de fichiers différents (un pour les entités, un pour les tableaux, un pour les index et les notes par exemple) ou d'un seul fichier rassemblant toutes les informations à contrôler. Ces fichiers sont généralement mis à disposition de l'éditeur selon un des deux formats PDF ou HTML :

- PDF : format plus précis, plus agréable à utiliser, orienté page (on mettra des folios, même s'ils ne correspondent pas aux folios de l'ouvrage papier final) mais figé : une modification de rendu exige de modifier les programmes de génération du PDF (ce qui reste malgré tout très simple) ;
- HTML : format brutal dans sa restitution mais, piloté par une feuille de styles CSS, il peut être modifié dans son rendu en retouchant uniquement celle-ci. En changeant au vol la CSS grâce à un module Javascript, on peut donner tous les rendus souhaités à partir d'un seul document HTML.

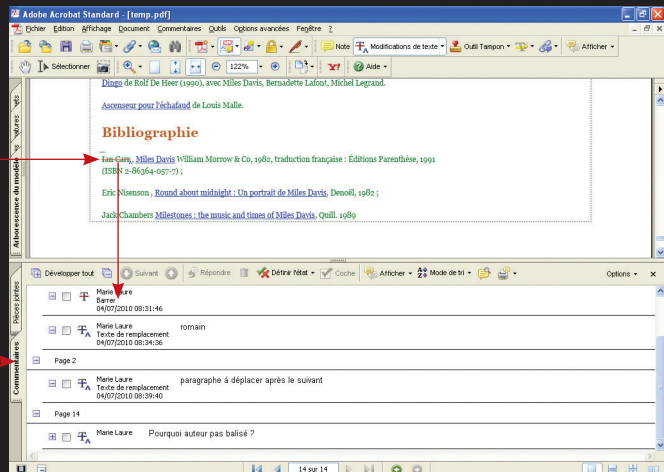
L'indication des remarques et modifications demandées se fera :

- soit sur un tirage papier selon les conventions classiques des corrections typographiques pour le fond ou la sémantique typo (gras, italique...) complétées le cas échéant par des indications portant directement sur les éléments. C'est la démarche la plus intuitive, mais lourde en logistique car il faudra renumériser les pages annotées pour les renvoyer au prestataire ;
- soit sur le PDF d'épreuve (ou une génération PDF du HTML). Cela suppose d'être muni d'Acrobat, de savoir utiliser les fonctionnalités de pose de notes

● Apposition des corrections et remarques sur l'épreuve à l'aide des outils d'Acrobat. Le curseur se positionne automatiquement en fonction du choix du commentaire.

Le curseur est placé en synchronisation avec le commentaire sélectionné

Liste des commentaires



ou de commentaires, et d'accepter de porter des remarques dans un document électronique, ce qui constitue un changement important d'habitude au sein des maisons d'édition. Avec l'habitude, c'est la meilleure formule.

Contrôler les caractères

Un des nombreux avantages de l'approche XML est de contrôler complètement le jeu de caractères utilisé. Une vérification à ce niveau permet de se prémunir de l'utilisation de caractères encodés dans des polices non Unicode et de vérifier strictement les entités caractères utilisées dans le document XML. Il est important d'avoir à l'esprit que la vérification doit se faire sur le fichier XML fourni et non pas sur une restitution qui aurait résolu (c'est-à-dire remplacé) les entités définies dans la DTD, auquel cas il serait impossible d'intercepter des erreurs telles que l'utilisation directe de l'Unicode en lieu et place de l'entité.

Les prestataires écrivent généralement des programmes de contrôles qui effectuent ce travail. Cela prendra la forme d'un tableau des caractères utilisés dont le code numérique Unicode se situe au-delà des caractères

accentués classiques (pour la langue française), en faisant apparaître le décompte si on le souhaite. Suivant que la DTD impose ou non des entités caractères explicites (ex. : `→` pour le signe $\rightarrow$), on fera ressortir les caractères dans leurs différentes possibilités de saisie :

- caractères directement encodés en Unicode sans entité : dans ce cas, on affiche directement le glyphe ($\rightarrow$) de la police de vérification ;
- caractères encodés sous forme d'entités caractères hexadécimal ou décimal (NCR) avec la visualisation correspondante : `→` ;
- caractères encodés sous forme d'entités nommées lorsque cela est prévu dans la DTD : `→` (mnémonique de *right arrow*).

REMARQUE

Ce tableau gagnera à être mis en tête de l'épreuve, ce qui permettra de détecter les anomalies d'encodage en première lecture.

Après analyse automatique du document, le caractère $\rightarrow$ a été utilisé une fois sous forme de caractère Unicode, 12 fois sous forme d'entité déclarée (dans la DTD), 3 fois sous forme d'entité hexadécimale et 1 fois sous forme d'entité décimale. Il est clair que, dans cet exemple, tout aurait dû être encodé en entité déclarée `→`.

Caractère	Glyphe Unicode ou glyphe de la valeur de remplacement de l'entité	Fréquence dans le document
$\rightarrow$	$\rightarrow$	1
<code>&rarr;</code>	$\rightarrow$	12
<code>&#x2192;</code>	$\rightarrow$	3
<code>&#8594;</code>	$\rightarrow$	1

Le cas des entités représentatives d'une espace doit être traité à part : les DTD définissent généralement les insécables et les fines sous forme d'entités

nommées (respectivement ` ` et ` `) auxquelles correspondent les valeurs hexadécimales ou décimales des entités caractères Unicode. Une DTD déclarera par exemple en décimal :

```
<!ENTITY thinsp "&#8201;" >
<!ENTITY nbsp   "&#160;" >
```

La vérification pose un problème : par définition, l'espace insécable encodée en ` ` n'est pas visible à la restitution. Pour la lecture/correction, on contourne le problème en rendant l'espace visible : il suffit d'effectuer une substitution lors de l'épreuve en remplaçant les valeurs d'entités par une chaîne de caractères atypiques et facilement reconnaissables. On utilisera ainsi dans le système d'épreuve quelque chose comme :

```
<!ENTITY thinsp "[#]" >
<!ENTITY nbsp   "[_]" >
```

Lors de l'épreuve, la restitution marquera clairement la présence d'une insécable ou d'une fine.

Une alternative consiste à utiliser des éléments au lieu d'entités pour spécifier des insécables ou des fines. On aura alors dans la DTD :

```
<!ELEMENT nbsp EMPTY >
<!ELEMENT thinsp EMPTY >
```

Le rendu sera spécifié par une commande CSS qui fera apparaître `[#]` pour l'insécable et `[_]` pour la fine, ce qui présente l'avantage de ne pas modifier la DTD en cours de production. Cette approche n'est pas souvent retenue ; elle est pourtant plus robuste que l'approche par entité qui nécessite des substitutions au niveau de la DTD en cours de traitement.

Vérifier le contenu hiérarchique

REMARQUE

En d'autres termes, il s'agit d'une table des matières « améliorée » faisant apparaître tout ce que le XML détient de manière sous-jacente en matière de structure hiérarchique.

La vérification du contenu hiérarchique s'effectue sur une épreuve mettant en exagération typographique toutes les informations de niveaux, ainsi que les informations de traitements spécifiques, par exemple les entrées de substitution (attribut `short-toc` dans `article_v1.2.dtd`) destinées aux tables des matières électroniques.

On fera également apparaître les éléments constitutifs du `front` dans une présentation de type formulaire, ce qui permet de le contrôler d'un seul coup d'œil.

Vérifier les images et les tableaux

Pour les images, le contrôle porte sur différents points :

- conformité de l'image appelée : l'image doit exister et l'image appelée doit être la bonne ;
- paramètres qualitatifs de l'image : détourage correct, conformité des éventuelles retouches apportées à l'image sur les spécifications de l'éditeur ;
- paramètres techniques de l'image : taille en pixels, résolution d'acquisition (généralement pour information) ;
- informations complémentaires attendues : légende, crédit.

À cette fin, l'épreuve proposera la liste des images dans leur contexte hiérarchique (s'il s'agit d'un éprouvage séparé) en mettant en marge ou sous l'image les informations suivantes :

- l'identifiant de la figure (et éventuellement de l'image) s'il n'est pas posé par programme, dans le cas d'un lien avec appels multiples par exemple ;
- nom du fichier et chemin déclaré dans le XML ;
- la taille de l'image en pixels : `width-px` et `height-px` ;
- la résolution de capture si cette information est prévue dans la DTD ;
- les crédits ;
- la légende.

Le contrôle des tableaux suit une démarche comparable. D'un point de vue XML, la difficulté vient de ce que l'encodage des tableaux varie souvent d'un prestataire à l'autre ou même d'un opérateur à l'autre. Les consignes de saisie imposent généralement les paramètres, ce sont donc eux qu'il faudra (rapidement) vérifier : nommage des colonnes, expression des dimensions (sous forme d'un pourcentage, sous forme d'un nombre de pixels, en unités métriques), non recouvrement des paramètres typographiques des filets.

Vérifier le balisage spécifique

Le balisage spécifique dépend beaucoup des structures explicitées dans les DTD. Au niveau de l'éprouvage, le principe général reste toutefois le même :

- mise en valeur typographique des informations sous-jacentes : dans une bibliographie, on rendra visible le balisage des auteurs, l'identification des titres d'ouvrages, des titres d'articles ou du nom d'une revue ;
- rattachement au point d'appel des informations parfois saisies à d'autres endroits de l'ouvrage : notes et entrées d'index par exemple.

L'objectif est de lever des ambiguïtés difficiles ou impossibles à détecter si l'on se limite au code typographique classique : l'italique dans une référence bibliographique sert à signaler le titre d'un ouvrage ou d'un article. C'est le lecteur qui effectue le rapprochement italique/titre d'ouvrage; mais pour un ordinateur, la seule indication typographique n'est pas suffisante.

Quelques exemples de contrôles d'éléments spécifiques

REMARQUE

Rien n'empêche de faire apparaître la note dans son contexte (dans le flux de texte); en effet, la note n'est qu'une grosse parenthèse, ce qui justifie d'ailleurs sa saisie dans le flux XML et non pas par un mécanisme d'identifiant avec un renvoi en fin de chapitre ou de livre, comportement sans intérêt en édition électronique.

Pour les notes, on signalera l'appel de note dans le texte et l'on fera apparaître en marge le contenu de la note ainsi que son éventuel identifiant si des appels multiples existent dans le document contrôlé. Pour les index, on fera apparaître en marge les appels d'entrées d'index, qu'il s'agisse d'une création d'index ou d'une rétroconversion :

- dans le cas d'un index en création (c'est-à-dire pour un nouvel ouvrage saisi directement en XML), il conviendra de produire l'index général de fin d'ouvrage afin de contrôler celui-ci dans sa globalité;
- dans le cas d'un index en rétroconversion, on fera apparaître les entrées d'index en les associant à chaque rupture de page, celle-ci étant saisie en rétroconversion à des fins de contrôle.

Vérifier les liens

Pour les liens internes du type renvoi vers un chapitre, il s'agit de s'assurer que la cible du lien est la bonne, sachant que la question de la vérification de l'existence de la cible ne se pose pas, le mécanisme XML permettant de s'en assurer : si la cible n'existe pas (c'est-à-dire que l'ID de la cible dans le texte n'existe pas), le document ne parse pas et l'erreur est détectée à la source.

La vérification s'effectuera en faisant apparaître dans le texte la présence du lien avec l'identifiant de la cible (par exemple entre parenthèses et dans une couleur reconnaissable) et dans la marge, le début du texte contenu par l'élément cible : si le lien renvoie vers un chapitre, on fera apparaître le début du titre de chapitre, ce qui permet de valider le lien d'un seul coup d'œil.

Pour les liens externes (URL, etc.), un programme de contrôle externe de vérification des liens cassés est passé sur le source XML avec pose d'un commentaire (`<!-- -->`) que l'on restituera entre parenthèses ou dans la marge de l'épreuve.

Vérifier une livraison de mise à jour

XML permet de vérifier très précisément les différences entre deux fichiers : de la modification d'un attribut à l'insertion d'un nouvel élément, en passant par une différence au niveau de la virgule ou des blancs significatifs lorsqu'ils sont prévus comme tels dans la DTD (portion de code informatique par exemple).

Presque tous les éditeurs XML disposent d'une fonction de comparaison, ce qui permet d'intervenir directement sur le document XML révisé pour le finaliser. Si cette approche paraît trop complexe, on fera appel aux fonctions de comparaison de fichiers des outils bureautiques comme Acrobat : il faudra dès lors s'assurer que les fichiers comparés font apparaître une partie des informations sous-jacentes du XML, comme les attributs.

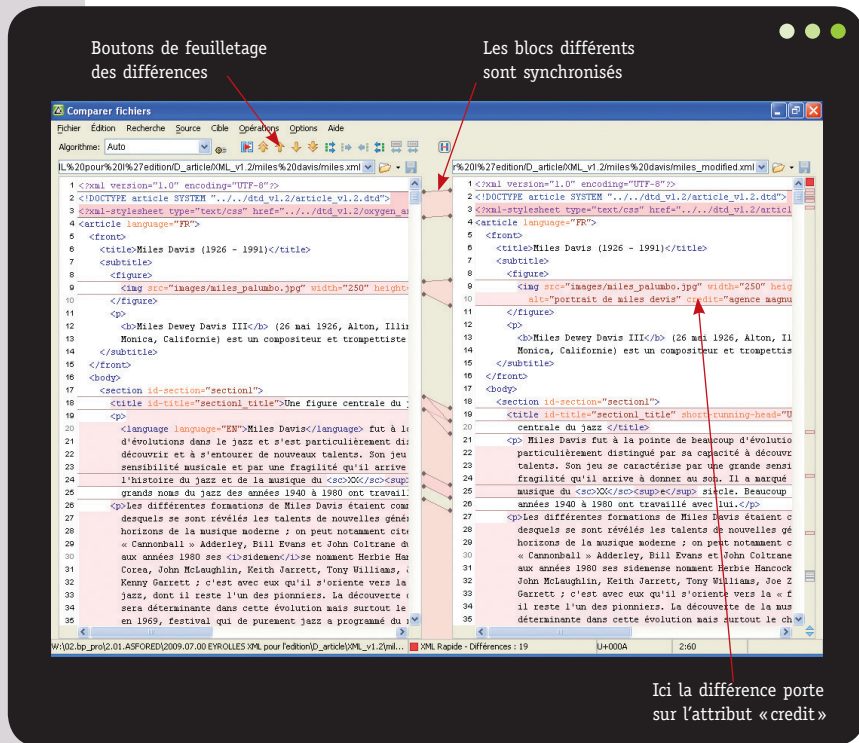
Les outils de comparaison des éditeurs XML

Un éditeur comme Oxygen dispose d'un outil de comparaison qu'il est possible d'appeler en mode autonome ou depuis l'éditeur afin de comparer deux à deux des fichiers (`DiffFiles.exe`). La présentation de la comparaison est un peu austère, car le XML est affiché en mode texte : aucun enrichissement ou interprétation typographique n'est proposé.

Les différences sont feuilletables une par une au niveau des blocs (paragraphe, div, listes...), le curseur se positionnant au début de celles-ci par un mécanisme de suivant/précédent simple. Si la différence ne porte pas sur l'élément lui-même, par exemple un élément inline, il est possible de zoomer sur

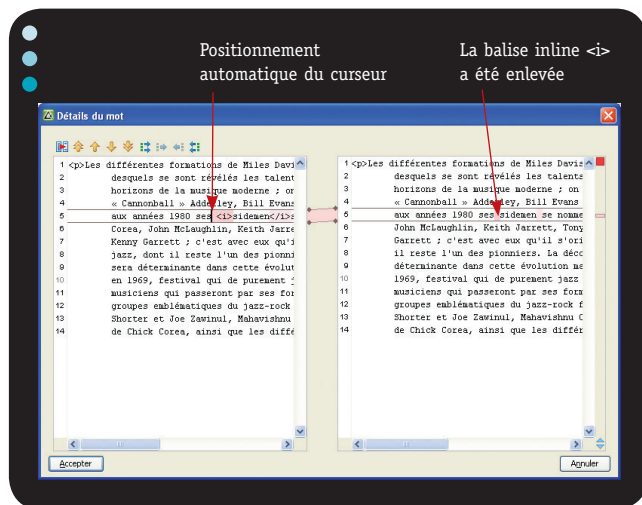
REMARQUE

Il convient d'être très rigoureux et de distinguer à tout moment le fichier de référence (fichier source) du fichier modifié dont on veut contrôler les différences.



● Utilisation de l'outil de comparaison de fichiers XML d'Oxygen

● Zoom au sein d'un bloc dans lequel une différence a été détectée : fenêtre Détails du mot



les différences au niveau du mot ou même au niveau du caractère. L'interface permet par ailleurs d'accepter ou non la modification et de corriger directement le XML examiné, ce qui en fait un outil puissant de retouche finale.

Dans un environnement d'édition, il faudra toutefois prévoir une formation à l'outil, en plus d'une formation générale à XML, afin d'en tirer pleinement parti. Pour les éditeurs que cela rebute, la solution passe par les outils de comparaison fournis notamment avec les applications bureautiques, Acrobat en particulier.

DiffFiles.exe est réglable de manière précise en fonction des objectifs du contrôle : la précision de l'algorithme de comparaison peut ainsi être choisie, permettant un traitement plus ou moins rapide. Le réglage sur «XML Exact» se traduit par un temps de calcul plus important, mais avec une précision accrue.

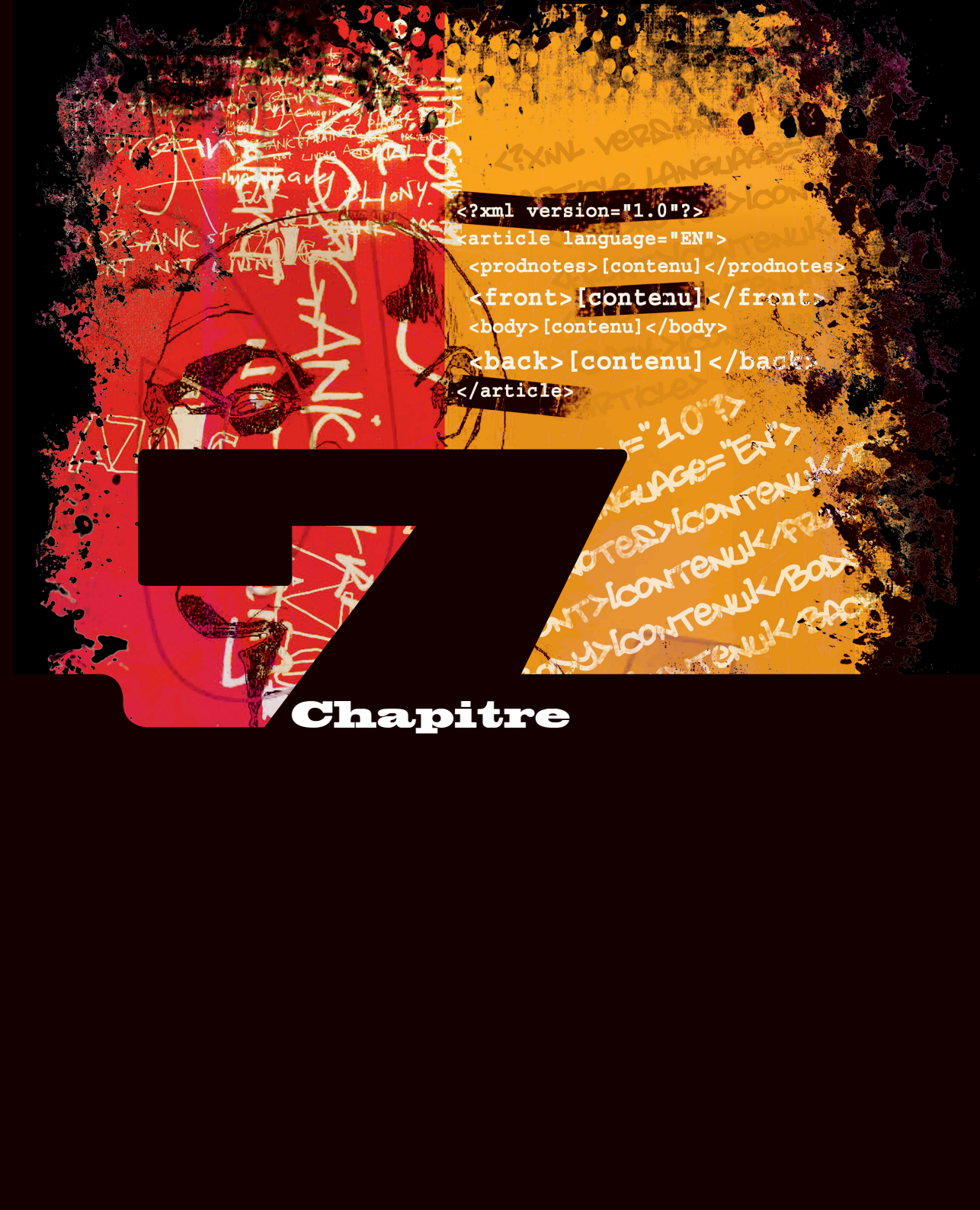
ORGANISER SON ESPACE DE TRAVAIL

Lors de la comparaison de fichiers, il convient de choisir une règle immuable de disposition, par exemple :

- le fichier source à gauche ;
- le fichier modifié à droite.

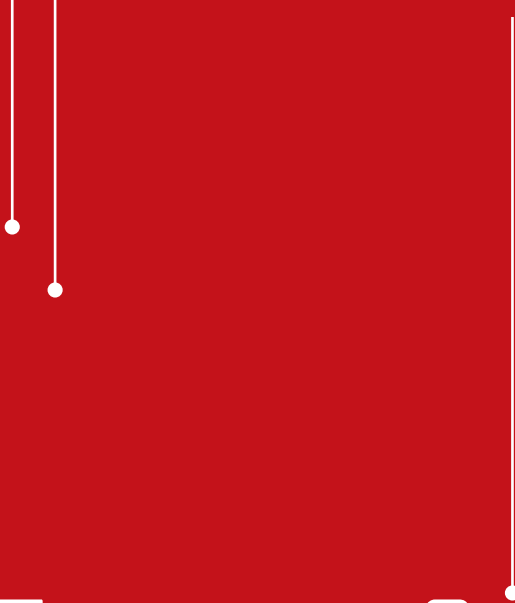
Les outils bureautiques

L'outil de comparaison d'Acrobat standard (version payante) permet de trouver les écarts de contenus textuel et typographique entre deux documents. Dès lors, on détectera très bien les écarts textuels et les disparités inline, à condition d'avoir choisi une différenciation typographique sur les éléments correspondants. L'éprouvage devra alors prendre en compte des informations sous-jacentes au XML, comme l'explicitation des attributs, ce qui permettra de visualiser les éventuelles différences.



<?xml version="1.0"?>
<article language="EN">
<prodnotes>[contenu]</prodnotes>
<front>[contenu]</front>
<body>[contenu]</body>
<back>[contenu]</back>
</article>

Chapitre



Transformer le XML avec XSLT

Le document XML, après épreuve et validation, est maintenant prêt à être publié. En pratique, l'étape de publication pour le rendre lisible sur un média particulier ou pour le communiquer à une autre machine, consiste à transformer le XML source en un format cible (XSL-FO, LATEX, HTML...). Ce format sera lui-même interprétable par un logiciel de restitution finale qui donnera au document son apparence habituelle de mise en pages.



La publication multisupport

Engager une démarche XML ne se justifie que si l'on vise la publication de tout ou partie du document sur différents supports, en restant conscient qu'un support cible tel qu'un écran de téléphone ne permet pas la même expression graphique que le support papier.

La démarche XML, qui suppose un investissement plus conséquent que celui imposé par la mise en pages papier, ne se justifie que si l'on vise la publication du document, ou d'un sous-ensemble du document, sur différents supports. La priorité aujourd'hui est encore donnée au papier qui constitue le principal support de communication de l'information textuelle (ou pour être plus précis, de l'information destinée aux médias non temporels). C'est aussi le support le plus exigeant avec des combinaisons graphiques incompatibles avec une automatisation complète. À l'autre extrémité se trouvent les médias sonores qui, par définition, sont complètement indépendants d'une représentation graphique. Entre les deux, on trouve toutes les combinaisons de complexité graphique imposées par les médias cibles.

CHOISIR LE XSLT

Le choix de ce langage n'a rien d'obligatoire, n'importe quel langage informatique pouvant faire l'affaire, mais il s'impose si l'on recherche la simplicité. XSLT est à la portée de n'importe quel utilisateur ayant codé un peu de HTML, le niveau requis étant celui de quelqu'un capable de manipuler Dreamweaver. Il en va différemment si l'on veut coder les transformations avec Perl, Java, PHP ou Python. Dans ces derniers cas, il vaut mieux être informaticien...

Les transformations multisupports

Il s'agit de transformer un document XML en différents formats cibles, en utilisant de préférence un seul langage de transformation dont le candidat naturel est XSLT.

Les transformations vont conduire à prendre le XML source et à lui imposer, via XSLT, des traductions vers

différents formats cibles qui eux-mêmes seront reconnus par des outils de lecture sur les médias finaux. En ordonnant les médias cibles par leur complexité graphique, du plus complexe à l'absence de graphisme, on obtient la classification suivante en termes de formats cibles :

- XML d'entrée dans les systèmes de PAO comme InDesign : il s'agit d'injecter dans l'outil de mise en pages un flux XML attendu par le système et qui réponde à une taxonomie spécifique. InDesign propose ainsi trois formats différents d'entrée pour le XML, le plus abouti étant proposé dans la version CS5 sortie en 2010. Ces formats sont des formats propriétaires Adobe, même s'ils sont ouverts – il s'agit chaque fois de XML. L'intérêt de passer par un outil PAO est de laisser à la maison d'édition la possibilité d'affiner aussi loin que voulu la mise en pages papier finale ;
- XSL-FO, format XML non propriétaire de représentation d'une mise en pages. Ce format a été élaboré par les instances internationales qui s'occupent de XML et est donc public. Le fichier XSL-FO, issu d'une transformation via XSLT du XML d'origine, n'est qu'une représentation technique de la page : il doit ensuite être pris en compte par un « interpréteur XSL-FO » capable de mettre en pages l'information dans son format technique final, généralement le PDF. Cette approche est celle d'une mise en pages d'esprit papier en termes de sophistication et de qualité, totalement automatisée grâce à des ordres d'optimisation de la disposition typographique. La qualité finale dépend toutefois de la maquette attendue, des paramètres de mise en pages définis par le DA et de la tolérance que l'on peut avoir vis-à-vis d'un montage automatique. Pour une publication PDF-écran, cette approche est très intéressante car plus prédictive qu'une mise en pages HTML, et elle propose toutes les fonctionnalités propres à un ouvrage PDF : liens, signets, notes de bas de page... ;
- XHTML à destination de l'affichage sur des sites web. On distinguera le XHTML d'encodage de pages destinées à être affichées sur un écran d'ordinateur de bureau, d'ordinateur portable ou encore de tablette (c'est-à-dire un écran dont la largeur est d'au moins 800 pixels), du XHTML destiné aux applications web pour téléphone (webapp) beaucoup plus contraint par la taille de l'écran ;
- XHTML comme élément constitutif d'un format d'agrégation supérieur comme epub pour les téléphones mobiles, les liseuses d'e-book et les tablettes dotées d'un logiciel de lecture de ce format. epub est un format très simple d'association de fichiers XHTML représentatifs d'un chapitre ou d'une

REMARQUE

Associé à un bon dictionnaire de césure et une maquette optimisée pour le coulage automatique, XSL-FO convient à une diffusion papier sans retouche complémentaire.

section (en fonction des choix faits au niveau de la segmentation du fichier source), correctement rassemblés dans un fichier unique zippé et généralement suffixé en .epub;

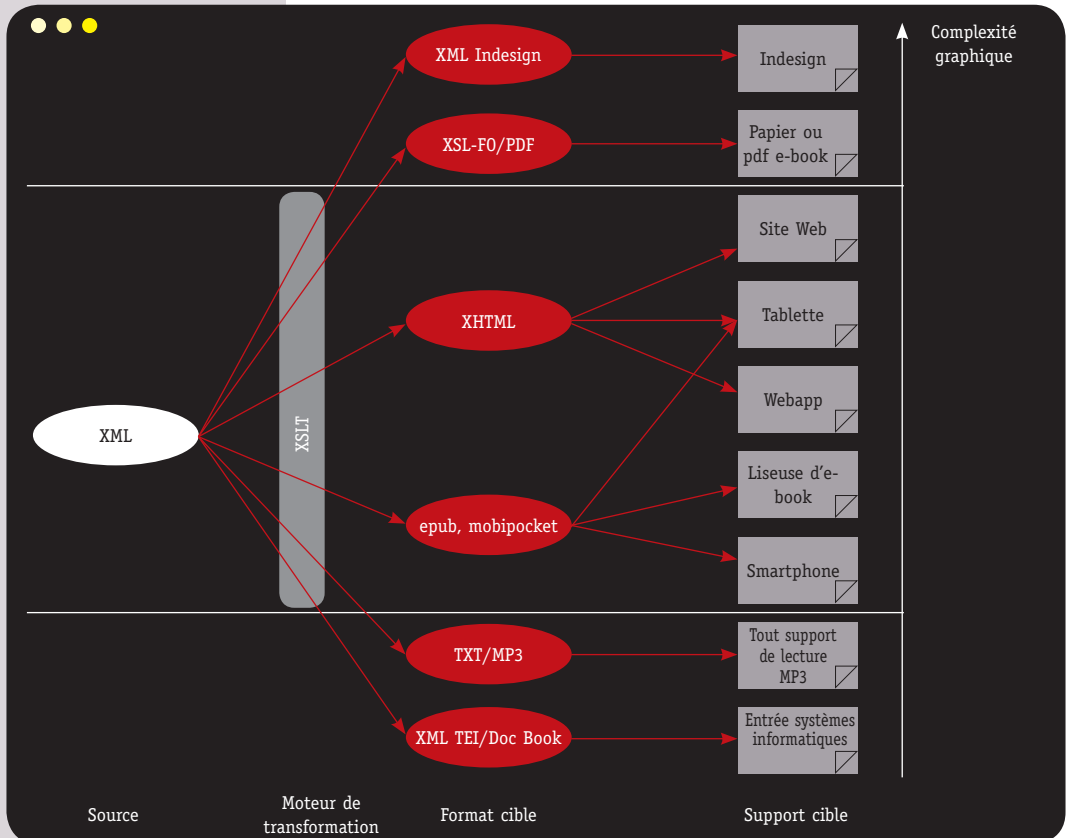
- format texte pur pour l'alimentation d'un synthétiseur vocal. Dans ce cas, la saisie XML aura éventuellement été dotée de balises spécifiques de ce média, pour typer les passages à prononcer en phonétique ou pour changer de locuteur par exemple;

À NOTER

En pratique, on utilise un ensemble de transformations XSL ou non XSL, donc écrites en faisant appel à d'autres langages plus adaptés que XSL dans certaines situations, le traitement multi-entrées par exemple.

- dialectes XML destinés à des applications s'attendant à recevoir un flux XML conforme à une taxonomie différente déjà utilisée, comme la TEI ou DocBook.

Tous ces formats pourront être produits grâce à différentes transformations ou successions de transformations XSL.



- Publier vers différents formats cibles

Les cibles de publication

Chaque cible de publication est dotée de caractéristiques propres qu'il convient de connaître afin d'apprécier la pertinence et les limites du codage XML correspondant. Pour faire simple : tout n'est pas automatisable depuis un flux XML et des compromis devront être trouvés entre l'automatisation du rendu et le niveau de retouche manuelle additionnelle à consentir pour une mise en pages sophistiquée.

Papier

La page a une géométrie prédéterminée et sa surface est perceptible dans son intégralité, contrairement à une page HTML qui exige un déroulement dès que celle-ci déborde la partie inférieure de l'écran. La géométrie devient un élément déterminant de la mise en pages et influe sur le fond, comme dans le scolaire où la communication écrite au sein d'un livre s'appuie explicitement sur la double page avec, par exemple, le cours à gauche et les exercices, commentaires ou illustrations à droite.

REMARQUE

Le papier présente la caractéristique d'être totalement sous contrôle de l'éditeur.

Les éléments graphiques sont tous figés : la police, les caractéristiques de la ligne, les césures sont sous contrôle, rien n'est laissé à l'initiative d'un logiciel prenant des décisions en fonction de paramètres fixés par l'utilisateur, comme sur un écran d'ordinateur.

La volumétrie du texte est également contrainte : certes, il existe de « gros » et de « petits » livres, mais la plupart des publications s'inscrivent dans des caractéristiques moyennes de pagination qui, à rebours, imposent une forme à l'objet livre.

Toutes ces contraintes disparaissent avec l'édition électronique. Les ressources intéressantes du papier, comme la mise en perspective de l'information sur la double page, deviennent même un handicap, l'association droite/gauche perdant sa sémantique avec l'affichage monopage.

PDF-écran

Le PDF-écran, par construction, n'est le plus souvent que la dématérialisation du papier. Parmi ses avantages pressentis, il a celui de conserver la qualité graphique du papier. Avantage qui peut se transformer en défaut quand on songe aux photos et illustrations en double page, classiques dans les magazines : à moins de disposer d'un écran large offrant un affichage confortable en double page, le lecteur se trouve confronté à une page simple dotée d'une

demie illustration ou d'une demie photo nuisant fortement à la lisibilité de l'ensemble.

Cela dit, le PDF-écran reste un format intéressant en édition électronique s'il est issu d'une génération automatique, car il préserve une bonne partie du libre arbitre graphique de l'éditeur : disposition des notes en bas de page ou ailleurs, titres courants, signets, table des matières interactive, index interactifs, tout en jouant sur la géométrie de la page, marges plus aérées, police plus grande, formats exotiques. Un beau livre pourra être présenté à l'italienne afin d'éviter le défilement vertical meurtrier sur la lisibilité d'une image, les images pourront être partitionnées pour devenir interactives...

REMARQUE

Rien n'empêche le PDF de compter 800 pages, là où l'ouvrage papier en compterait 300. L'édition électronique n'a pas de limite économique, si ce n'est la gêne éventuelle due à la faible bande passante d'une ligne ADSL.

Il reste un inconvénient : le PDF-écran étant très dépendant du matériel cible, il faut produire différents PDF pour différents écrans cibles – ce qui ne pose aucun problème dans un environnement totalement automatisé – et gérer différents objets de publication sur les réseaux de diffusion, ce qui est un problème un peu plus important.

Site web pour ordinateurs de bureau et portables

En termes de conception, le site web à destination des ordinateurs de bureau ou des portables doit être appréhendé comme un support bimédia : papier et écran.

L'impression écran. Support papier, le site web l'est, dans la mesure où de nombreux internautes lancent l'impression d'une ou plusieurs pages des sites qu'ils visitent. Leur finalité n'est pas toujours claire, mais le geste semble guidé par une remédiation de l'information électronique sans doute plus sécurisante, la trace papier semblant (à tort) plus pérenne que l'information jugée volatile dans son affichage écran. Sans oublier non plus les nombreux avantages du papier dont la mobilité constitue le principal atout : on peut reprendre le document indépendamment de la disponibilité d'un ordinateur, on peut l'annoter, le transmettre...

Il est donc important pour le concepteur de sites web de penser l'impression papier de la page comme un média et de s'organiser pour que cette impression soit conforme au média cible. On éliminera, par exemple, toutes les informations de navigation et d'informations émettées proposées sur les pages des sites, à la manière des unes de journaux qui cherchent à capter en simultanéité l'intérêt du lecteur, pour se concentrer sur une impression ramassée, conser-

vant l'essentiel et en en soignant la présentation. Cela est possible par un bon contrôle de la CSS dédiée à l'impression (`media="print"`) que trop peu de développeurs web connaissent. Le contrôle de l'impression de ces pages de site fait partie des exigences éditoriales que devrait avoir tout éditeur d'information.

L'écran. Le deuxième média propre au site web est l'écran. L'information s'exprime dans des dimensions assez confortables, avec toutefois la contrainte, par rapport au papier, du sens de visualisation à l'italienne : la largeur de la page visible est supérieure à sa hauteur, ce qui conduit aux divisions en 3 colonnes ou 4 colonnes de l'écran. Cette contrainte disparaît cependant avec les tablettes qui peuvent s'utiliser indifféremment à la française ou à l'italienne.

Le site web présente de nombreux défauts pour un éditeur habitué à contrôler sa mise en pages : l'internaute peut changer lui-même la taille des caractères et même la police ; les navigateurs interprètent de manière disparate les pages HTML ; la couleur des images est soumise à l'extrême variabilité des écrans, depuis leur réglage colorimétrique jusqu'à leur environnement lumineux, ce qui rend inutile tout travail de précision sur les caractéristiques de l'image. Enfin, le site web présente sa propre logique de navigation, dont la plus importante est la notion de « page-vue » par opposition à la notion de « page-précédente ». Cette remarque prend toute son acuité dans le cas d'un *reader* de PDF pour qui la notion de page-vue, souvent appelée page-précédente par l'internaute, est notoirement différente de celle de page-précédente du PDF qui est en fait la page N-1 du document, page que l'internaute n'a pas nécessairement feuilletée. Le PDF reste un média linéaire, contrairement à la page web qui est un média fondamentalement hypertexte.

REMARQUE

Avec un peu de cynisme, on pourrait aller plus loin : ne proposer que sous forme imprimée une partie de l'information en y incorporant une publicité contrainte. On aurait par exemple la mention « Cliquez ici pour imprimer la transcription exclusive des écrits téléphoniques de x ou y ». La presse en ligne ne semble pas encore avoir inventé cet outil à potentiel publicitaire...

● Page-précédente et page-vue précédente : la contradiction PDF/navigateur sur les flèches suivant et précédent qui ne signifient pas la même chose.



Site web pour téléphone

Le site web pour téléphone, à la différence du site web pour ordinateur n'est pas bimédia : l'utilisation ne se fait qu'à l'écran.

La surface d'écran très réduite, environ 480 × 320 pixels, impose une économie extrême dans l'expression graphique : les marges, pourtant nécessaires pour la respiration d'une page de livre, doivent être chichement octroyées ; les mises en page flamboyantes des livres scolaires se réduisent à l'essentiel – le cours, des images plein écran avec peut-être un peu de zoom, des exercices interactifs à l'allure spartiate. Les notes de bas de page, qui pour des raisons de performances techniques ne peuvent devenir actuellement des notes de bas d'écran, se trouvent le plus souvent renvoyées en fin d'ouvrage. Ceci conduit à des aberrations visuelles ou à des temps de chargement rapidement insupportables, le fichier des notes se trouvant être distinct du fichier d'affichage du chapitre ou de la section en cours de lecture.

Liseuses

Les liseuses sont de petits ordinateurs portables dédiés aux livres. Leur point fort revendiqué est de s'appuyer sur une technologie d'écran non rétro-éclairé, ce qui se traduit par une autonomie importante atteignant facilement un mois – l'autonomie se traduisant plutôt par un nombre de pages pouvant être vues qu'une durée – et un confort théorique de lecture plus important, la lisibilité provenant de la réflexion de la lumière.

Ce type de matériel présente la caractéristique d'afficher l'information strictement dans la surface de l'écran physique : en d'autres termes, l'approche propre aux navigateurs consistant à dérouler le texte en débordement inférieur n'existe plus. Retour à l'univers du livre papier (c'est-à-dire à la géométrie de la page physique imposée) avec les caractéristiques de l'électronique : possibilité de recomposition au vol de l'ouvrage après un changement de paramètres, choix de la justification, fonctionnalités de synchronisation entre plusieurs systèmes de lecture connectés (ex. : téléphone/liseuse).

Pour l'éditeur, la liberté graphique reste faible :

- d'une part, il doit se conformer à un format technique partagé par le plus grand nombre, aujourd'hui epub – format ouvert et public – et, dans une moindre mesure, mobipocket, format propriétaire d'Amazon ;
- d'autre part, les capacités graphiques de ces formats se limitent à un support très élémentaire de CSS. Bien sûr, cela ne peut que s'arranger, mais pour l'heure, les capacités graphiques sont décevantes.

Tablettes

Depuis le printemps 2010, l'avenir des liseuses est remis en question avec l'arrivée de l'iPad, premier d'une série de tablettes aux caractéristiques fort intéressantes pour l'édition : écran 1024 × 768 donc comparable à celui d'un ordinateur de bureau, reader d'epub (iBooks) moins élémentaire que ceux disponibles sur la place publique (Stanza) et doté d'une mise en pages moins spartiate, d'un support CSS un peu plus efficace et d'une prise en compte partielle de JavaScript. De quoi reprendre la main en termes de rendu visuel, sans compter l'arrivée prochaine (2011) de HTML5. Par ailleurs, les tablettes sont munies de navigateurs de dernière génération et il (re)devient possible d'envisager des publications de type HTML directes, sans passer par l'intermédiation d'un *reader* d'epub.

Autres cibles

Parmi les autres cibles de publication, mentionnons :

- la transformation du flux XML en flux de texte acceptable par un synthétiseur vocal – on prendra en compte dans le flux XML les contraintes du synthétiseur pour baliser une substitution phonétique ou le changement de langue, ce qui permettra de modifier le locuteur ;
- la transformation des documents XML bâtis selon une taxonomie d'entrée pour les mettre en conformité avec une taxonomie différente de sortie. Ce sera le cas de traducteurs XML vers des formats comme la TEI ou DocBook. Cette démarche peut être rendue nécessaire lorsque l'utilisateur cible a déjà beaucoup investi dans ces formats, tandis que l'éditeur à l'origine du document XML ne souhaite pas rentrer dans la complexité de la TEI ou DocBook.



Le langage XSLT

XSLT est un langage de transformation d'arbres XML en d'autres arbres XML – en particulier XHTML et XSL-FO – ou en différents flux textuels. Adossé à XPath, langage de spécification des nœuds de l'arbre, il permet la restructuration documentaire grâce à une syntaxe très simple à condition de comprendre la récursivité.

Principe d'une transformation XSLT

Une transformation XSLT consiste à sélectionner un ou plusieurs nœuds de l'arbre XML et à le(s) soumettre à une réorganisation/réécriture pour le(s) rendre conforme(s) à la syntaxe et aux attentes du format cible, par exemple HTML. Bien entendu, il faut une connaissance préalable du format cible pour écrire une transformation.

Les ordres de transformation sont incorporés dans une «feuille XSLT» qui peut être unique ou modulaire. Une feuille XSLT est un programme de transformation écrit en XML et dont les noms d'éléments et d'attributs sont figés de manière à pouvoir être compris par un outil logiciel complémentaire appelé «processeur XSLT». Par extension, on parlera de «langage XSLT» pour faire référence aux instructions XML spécifiques de ce vocabulaire.

FEUILLE XSLT

On parle de «feuille XSLT» parce qu'à l'origine, XSL (*XML StyleSheet Language*) a été conçu comme une feuille de styles. Au fil des travaux, la vision d'origine a été perdue et XSL s'est transformé en deux taxonomies XML particulières : XSLT pour les transformations et XSL-FO pour les mises en page. Cette terminologie opaque a perduré et il vaut mieux faire abstraction du terme «style» correspondant au S de XSL.

Une transformation XSLT est guidée par différentes considérations importantes à la base du langage XSLT :

- la transformation commence toujours par le « haut » de l'arbre XML, c'est-à-dire l'élément racine `<article>` dans le cas des documents XML conformes à `article_v1.2.dtd`;
- l'arbre est parcouru de haut en bas pour chaque branche et chaque nœud;
- les transformations sont récursives : si un traitement est spécifié pour un nœud (exemple une `div`), il sera appliqué à toutes les sous-branches de l'arbre portant le même nom de nœud. Toutes les `div` seront traitées de la même manière, sauf si l'on spécifie des conditions particulières, par exemple une `div` possédant un attribut `class` particulier, ou encore une `div` incluse dans une autre `div`. Ce comportement, un peu difficile à comprendre au début, ressemble à celui des CSS et ne devrait pas poser de problème à un concepteur graphique de sites web.

Le lancement d'une transformation XSLT nécessite de disposer d'un programme particulier capable de comprendre les instructions XSLT. Baptisé « processeur XSLT », il se présente soit de manière autonome, soit intégré à un environnement de développement comme Oxygen, soit encore intégré à un navigateur, ce qui est le cas de n'importe quel navigateur récent.

Le choix dépend essentiellement du confort de développement et du niveau de XSLT mis en œuvre : le langage en est à sa version 2 et tous les processeurs ne le supportent pas.

À NOTER

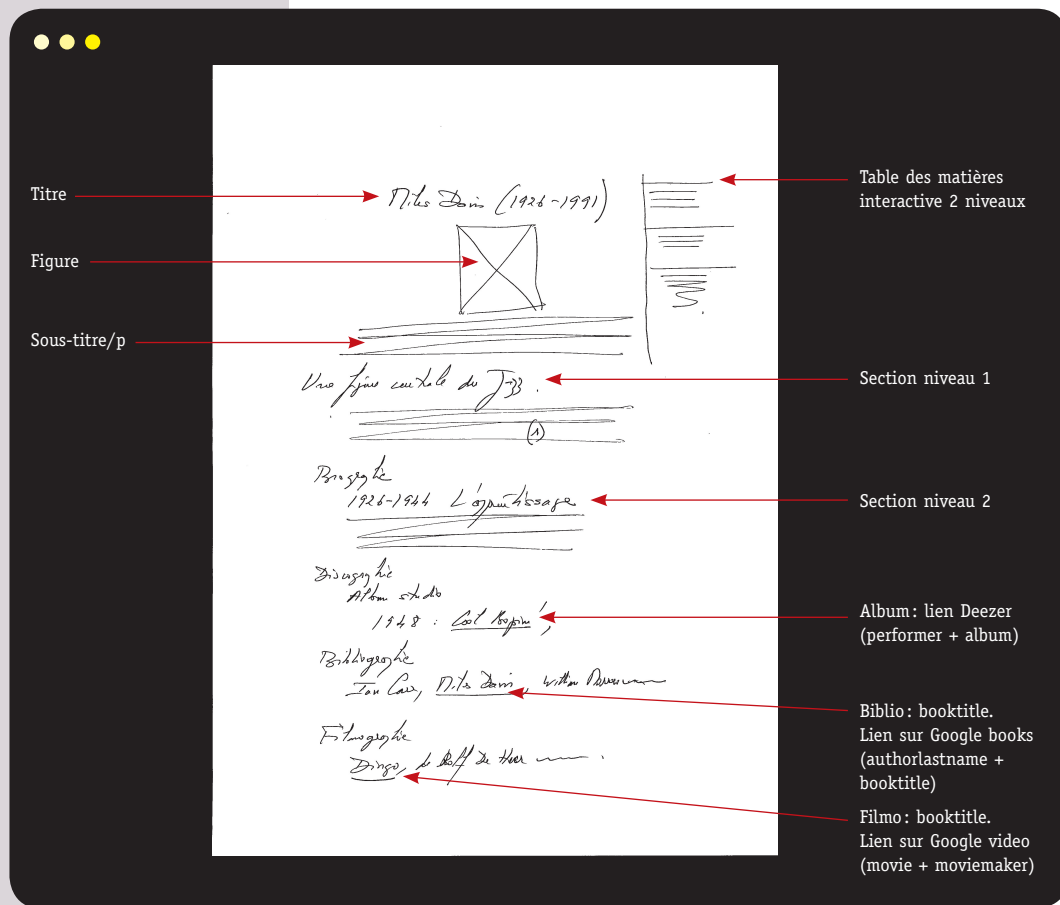
Pour les besoins du présent ouvrage, les exemples seront fournis en XSLT-1, sachant que l'orthodoxie commande de travailler en XSLT-2, cette dernière version présentant l'avantage de prendre en charge de manière native la transformation multi-documents cibles.

Spécifier une transformation

Une bonne compréhension de la taxonomie permet de concevoir l'objet de publication en sortie et de fournir les indications à la personne chargée de l'écriture du programme XSLT. Le plus souvent, cela passe par un crayonné commenté (voir la figure page suivante).

Il appartient au concepteur du site web de travailler de manière intelligente et moderne : la transformation XSLT doit produire un flux HTML épuré de toute mise en pages, mais présentant les informations de classes (c'est-à-dire des attributs `class`) permettant à une feuille CSS externe (également à la charge du concepteur de sites web) de mettre la page en présentation graphique. Cette manière de produire le XHTML ne relève pas d'une technicité particulière : c'est une décision managériale, souvent occultée dans la pratique où

trop de liberté est laissée aux équipes informatiques qui se débarrassent de la contrainte en codant plus ou moins « en dur » le rendu XHTML.



● Crayonné de spécification d'une transformation HTML cible (ici HTML desktop)

Utiliser un processeur XSLT

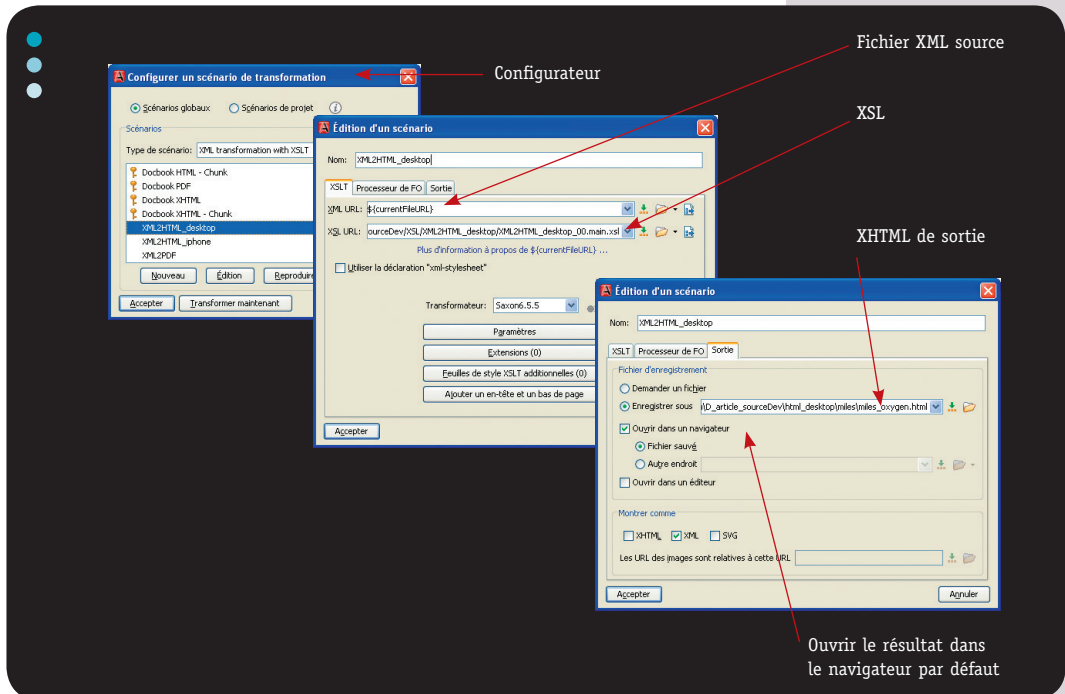
Un processeur XSLT prend le document XML source, lui applique le programme XSL voulu et restitue un résultat, si tout va bien, dans le format cible attendu. Le plus simple est de faire appel à une interface de transformation qui permet de spécifier le document XML à traiter, le programme XSLT à mettre en œuvre et le fichier résultat.

Pour l'exemple de l'article sur Miles Davis, on aura ainsi :

- fichier source : miles.xml ;
- fichier XSLT : XML2HTML_desktop_00.main.xsl ;
- fichier résultant : miles.html.

En pratique, la démarche est un tout petit peu plus compliquée, car il faut spécifier des chemins d'accès – les différents fichiers n'étant pas stockés au même endroit – et procéder à des copies de ressources comme les images, éventuellement retaillées pour le média cible, si l'on veut obtenir un jeu de fichiers homogènes faciles à mettre en ligne.

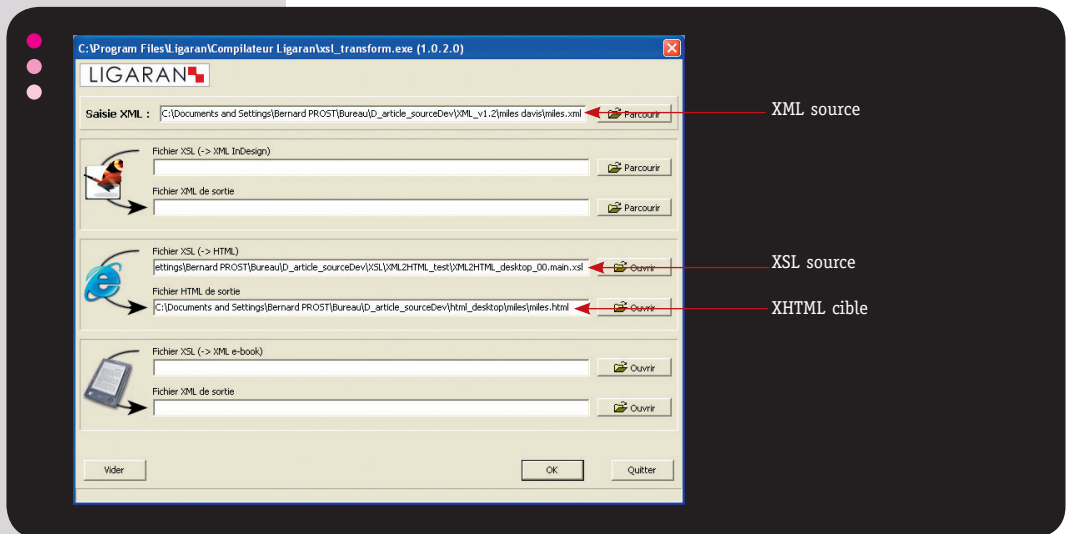
● Configuration d'une transformation XSLT de XML vers XHTML dans Oxygen



Dans le cas d'Oxygen, on spécifiera au niveau de l'interface générale les fichiers à mettre en œuvre sous forme d'un scénario de transformation à l'aide de l'outil Configurer un scénario de transformation. On précisera dans les options, pour plus de confort, que la sortie XHTML doit s'ouvrir dans un navigateur, le rendu graphique de l'interpréteur XHTML natif d'Oxygen étant trop limité.

On pourra également utiliser un lanceur local, pour les cas les plus simples, comme celui de Ligarán qui fonctionne en XSLT1. Il suffit de spécifier le fichier source, le fichier XSLT et le fichier cible avant de lancer l'opération qui ouvrira d'elle-même le navigateur pour afficher le HTML résultant.

● Configuration d'une transformation sur le lanceur Ligarán



Écrire un programme XSLT

Informations obligatoires

Un programme XSLT est d'abord un fichier XML : il en respecte les règles de base :

- déclaration XML : `<?xml version="1.0"?>`;
- présence d'un seul élément englobant toutes les commandes XSLT : `<xsl:stylesheet></xsl:stylesheet>`;
- déclaration de l'espace de nom réservé aux programmes XSL sous forme de deux attributs obligatoires :

- `xmlns:xsl=http://www.w3.org/1999/XSL/Transform;`
- `version="1.0"` (ou un indice plus élevé).

En pratique, le contenu de l'attribut `xmlns:xsl` étant simplement informatif, il n'a pas à nécessairement pointer vers une ressource existante sur le Web (un URI, par construction unique, suffit). Certains processeurs XSLT vérifient la valeur de cet attribut : il est plus prudent et plus simple de reprendre à l'identique la chaîne de caractères sans se poser de questions...

Écriture des ordres de transformation

Fondamentalement, XSLT parcourt le document XML et traite les seuls nœuds pour lesquels existe un modèle, appelé *template* en anglais, dans le programme XSLT : si le modèle est rencontré au cours du traitement, alors son contenu de transformation est appliqué.

DU TEXTE AU KM PAR DÉFAUT

Les processeurs XSLT disposent de réglages de transformation par défaut : en particulier, tous les nœuds sont traités en passant en sortie uniquement leur contenu textuel. Donc, une feuille XSLT a minima produira la sortie brutale de tout le texte, sans aucun enrichissement ni prise en compte des attributs, etc.

Voici un exemple de modèle décrivant pour le traitement de la balise `<authors>` une sortie HTML :

```
<xsl:template match="authors">
<div class="authors">
<xsl:apply-templates/></div>
</xsl:template>
```

Cet ordre XSLT permet de :

- déclarer un modèle (`<xsl:template>`);
- qui ne s'applique qu'à la balise `<authors>` (i.e. présence de l'attribut `match`);
- si elle est trouvée, écrire en sortie la balise d'ouverture : `<div class="authors">`;
- traiter tout le contenu de la balise `<authors>`. L'ordre `<xsl:apply-templates/>` signifie : appliquer les transformations XSLT, disponibles ailleurs dans le programme XSLT, sur le contenu de la balise `<authors>` (on traitera par exemple `<authorlastname>` dans une autre partie du programme XSLT). Le fait que le traitement ne soit pas local correspond à la récursivité;
- écrire en sortie la balise de fermeture `</div>`.

Place des *templates* dans le programme XSLT

XSLT est un langage décrivant des traitements sur les nœuds en parcourant l'arbre du document XML de haut en bas. L'emplacement des modèles (*templates*) dans le programme XSLT n'a, en principe, pas d'importance. On veillera seulement à regrouper les commandes selon une logique et une modularité en phase avec la DTD.

Lorsque le nœud `authors` est traité (il faut donc au moins qu'il existe), le programme XSLT donne donc la sortie HTML suivante :

```
<div class="authors"> (contenu traité par  
correspondance  
avec d'autres templates du programme XSLT) </div>
```

La CSS d'affichage du HTML, écrite par ailleurs, se chargera de mettre en valeur graphiquement la classe `authors` dans le navigateur.

Il existe différentes commandes permettant d'effectuer des actions particulières, comme `<xsl:apply-templates/>` qui permet d'appeler un modèle déjà défini quelque part dans la feuille XSLT sous forme d'un `match`. On aurait ainsi :

```
<xsl:apply-templates select="authors"/>
```

ce qui signifie que l'on applique à cet endroit précis, le modèle défini par `match="authors"`.

Un programme XSLT complet est donc essentiellement la succession de `match` sur des nœuds donnant la description des sorties à produire et éventuellement l'appel de ces modèles s'ils doivent s'appliquer à des endroits différents de celui de l'arbre XML auquel ils s'appliquent : cette caractéristique permet de comprendre pourquoi les documents XML peuvent complètement être réorganisés au moment de leur traitement par XSLT.

Le langage complet, bien que très économe en instructions, possède une panoplie suffisante de commandes pour traiter à peu près n'importe quelle transformation. La maîtrise complète et optimale du langage est affaire de spécialiste, mais sa mise en œuvre, très simple, est à la portée de n'importe quel éditeur qui gagnera à en comprendre le fonctionnement : cette compétence lui permettra de mieux agencer la DTD, en particulier de la rendre économe en nombre d'éléments grâce à une bonne compréhension de la récursivité.

X-Path

X-Path est le moyen de décrire, en particulier dans un modèle, le chemin d'accès à un nœud particulier par rapport au nœud courant (celui en cours de traitement).

La syntaxe est disponible en deux versions :

- une version explicite où la position généalogique des nœuds est exprimée en anglais (`descendant`, `child`, etc.);

- une version compacte, dont l'écriture ressemble à celle des chemins d'accès à un fichier dans un système de fichiers type DOS et qui permet de se repérer par rapport au répertoire (c'est-à-dire au dossier) en cours. C'est celle que nous retiendrons.

Écrire `../images` dans un système de fichiers signifie : « remonter d'un cran dans l'arborescence des répertoires, par rapport à l'endroit où l'on se trouve, et redescendre dans le dossier `/images` ».

XPath dispose du même genre de syntaxe, en particulier le slash pour indiquer l'emboîtement des nœuds, mais en plus sophistiqué car il permet de spécifier de manière précise les nœuds ancêtres, les descendants, les frères, etc., ainsi que leur position dans l'arbre.

Écrire `match="note/p[position()=last]"` permet de retrouver, dans le nœud courant, le dernier paragraphe à l'intérieur de la note.

Ou encore écrire `select="../section[@id]"` lorsque l'on est dans un modèle appliqué au titre d'une section (soit dans le document XML : `<section id="S05"><title>`) signifie : aller chercher dans le parent immédiat (ici `<section>`) la valeur de son identifiant et en faire quelque chose en sortie (le « quelque chose » dépend de ce qui est écrit avant le `select`. Si c'est une commande du type `<xsl:value-of select="../section[@id]"/>`, la sortie est le contenu du nœud, soit ici la valeur `S05` de l'identifiant de la section.

XPATh abrégé

La syntaxe XPath, selon le vocabulaire explicite en anglais pour spécifier un chemin d'accès à un nœud relativement au nœud courant, est trop verbeuse pour un usage courant. On pourrait ainsi écrire : `section/descendant::p` pour spécifier n'importe quel paragraphe enfant (i.e. quelle que soit la profondeur) de `section`; on notera le mot-clé « descendant » et le double double-point (`::`). L'écriture abrégée, plus compacte est : `section//p`.

Chapitre



Publier pour l'électronique



Publier pour l'électronique est un art éphémère. La principale difficulté est liée aux multiples cibles (site web, téléphone, tablette, liseuse) et donc à leurs caractéristiques propres de taille d'écran et de logiciels de lecture fortement dépendants de l'environnement employé. Cela impose pour le court terme de générer de multiples formats, et ainsi de produire des fichiers différents à gérer en aval au niveau des plates-formes de distribution.



Publier pour le Web

Le site web, cible actuellement la plus riche en mise en pages et en interactivité, oblige l'utilisateur à rester connecté durant la consultation. Cette contrainte sera éliminée par HTML 5. Mais la lecture HTML de livres reste un modèle lucratif, en témoigne le site de l'éditeur O'Reilly spécialisé dans les ouvrages TIC.

Quelle unité d'affichage ?

La question de l'unité d'affichage se pose lorsque l'on parle d'édition web. Comment découper l'ouvrage destiné à l'origine au papier afin de l'adapter à un affichage avec défilement vertical rappelant les rouleaux de l'ère pré-Gutenberg ?

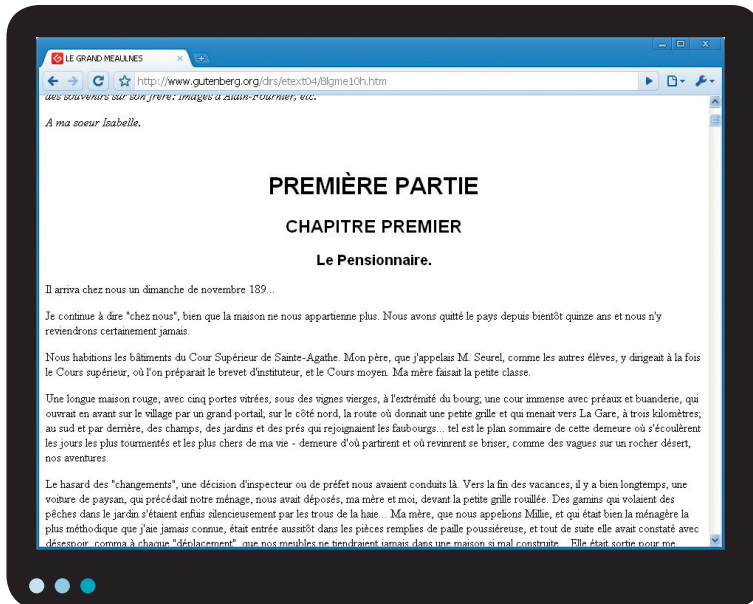
Nous distinguerons le cas des livres organisés en parties, chapitres et sections pour lesquels la question se pose avec acuité – il s'agit de transformer un livre en base de données, des ouvrages de type catalogue ou dictionnaire pour lesquels la question est inverse, leur nature étant essentiellement celle d'une base de données compilée sous forme d'un recueil papier.

Les livres

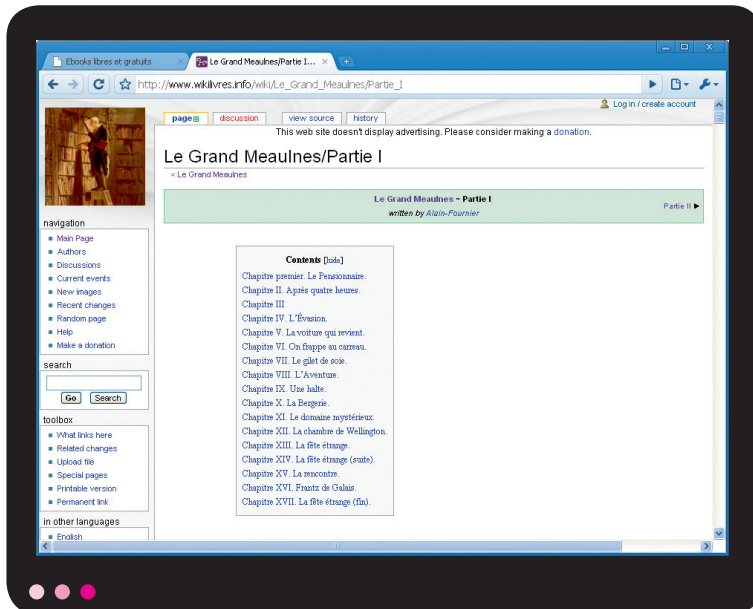
Rien n'empêche de publier un livre sous forme d'une seule grande page web incorporant la totalité de l'ouvrage. La navigation pourrait s'y faire par chapitre, section, etc., grâce à une table des matières suffisamment fine et à un couplage astucieux CSS et JavaScript reproduisant la navigation par chapitre.

Pour un roman dont le poids informatique est presque toujours inférieur à 1Mo, la démarche est parfaitement envisageable comme on peut le voir sur le site du projet Gutenberg (www.gutenberg.org) où sur le site de Wikilivres (www.wikilivres.info). En revanche, un livre comportant de nombreuses images ou formules traitées en images verra son poids rapide-

ment dépasser 10 Mo, avec pour conséquence un allongement significatif du temps de chargement.



● Le début du *Grand Meaulnes* en téléchargement libre sur www.gutenberg.org. Le roman est publié sous forme d'une seule page HTML.



● Le même titre sur Wikilivres. Le livre est découpé en parties (une partie = une page HTML); chaque partie est dotée d'une table des matières locale pointant sur les différents chapitres rassemblés dans la même page HTML.



Découpage
en chapitres

- Le roman de Théophile Gautier, *Fortunio*, en lecture par chapitre sur Wikisource

Une autre approche consiste à découper l'ouvrage en chapitres et à lui adjoindre une table des matières permettant la navigation directe entre chapitres. Cette démarche convient pour les romans (www.wikisource.org) pour lesquels la taille des fichiers unitaires à afficher reste modeste.

Dès que l'on doit afficher des illustrations, des photos, des diagrammes ou encore des formules-images – ce qui est le lot de la plupart des ouvrages professionnels –, la taille de la page à transmettre devient excessive. On adopte dans ce cas une navigation plus fine en découpant le chapitre au niveau de ses sections.

Cela impose de décider du traitement des titres de parties, de leurs informations introductives, ainsi que des titres de chapitres suivis de leurs textes introductifs (i. e. avant la première section) : le choix d'O'Reilly a été d'agréger à la première section les informations introductives de parties et de chapitres ; la même page HTML est ainsi affichée, que l'on pointe sur un titre de partie, le titre du premier chapitre ou le titre de la première section de celui-ci. La table des matières conserve la hiérarchie en parties/chapitres/sections, mais l'affichage agrège les informations au niveau de la première section.

Le titre de partie, le titre de chapitre et le titre de section (non dépliée ici) pointent vers la même page.

La partie, le titre de chapitre et son introduction sont concaténés à la première section sous forme d'une seule page HTML.

- ▶ Notes
- ▶ Bookmarks
- ▶ Search
- ▼ Contents
 - Table of Contents
 - Index
 - Copyright
 - Acknowledgments
 - Introduction
 - ▼ Pt. I. Core Markup
 - ▶ Ch. 1. Traditional HTML and XHTML
 - ▶ Ch. 2. Introducing HTML5
 - ▶ Ch. 3. HTML and XHTML Element Reference
 - ▶ Pt. II. Core Style
 - ▶ Pt. III. Appendices
 - Index

Part I: Core Markup

- CHAPTER 1 Traditional HTML and XHTML
- CHAPTER 2 Introducing HTML5
- CHAPTER 3 HTML and XHTML Element Reference

Chapter 1. Traditional HTML and XHTML

Markup languages are ubiquitous in everyday computing. Although you may not realize it, word processing documents are filled with markup directives indicating the structure and often presentation of the document. In the case of traditional word processing documents, these structural and presentational markup codes are more often than not behind the scenes. However, in the case of Web documents, markup in the form of traditional Hypertext Markup Language (HTML) and its Extensible Markup Language (XML)-focused variant, XHTML, is a little more obvious. These not-so-behind-the-scenes markup languages are used to inform Web browsers about page structure and, some might argue, presentation as well.

First Look at HTML and XHTML

In the case of HTML, markup instructions found within a Web page relay the structure of the document to the browser software. For example, if you want to emphasize a portion of text, you enclose it within the tags `<em>` and `</em>`, as shown here:

```
<em>This is important text!</em>
```

Le service Safari d'O'Reilly. L'unité d'affichage et de navigation est la section. La première section agrège le titre de partie, son introduction, le titre de chapitre, son introduction et toute la première section. (© O'Reilly)

Les catalogues et dictionnaires

Certains ouvrages se prêtent naturellement à une publication web sous forme de base de données, ainsi les catalogues pour lesquels l'unité d'affichage est une notice fortement structurée, la plupart des champs étant toujours renseignés. Dans ce cas, on fait appel à un programme de transformation, éventuellement écrit en XSLT, qui produira autant de fichiers XHTML que d'entrées disponibles.

Un programme complémentaire de remplissage de la base de données (créée par ailleurs) extrait les données intéressantes des balises et produit les index. Quant au document d'affichage et ses ressources de type image, il est stocké soit dans la base sous forme d'un champ composite (un *blob*), soit sous forme d'un ensemble de fichiers que le programme web viendra consulter. Pour l'internaute, le résultat visuel et comportemental est celui d'une application web renvoyant une réponse à une question : l'organisation en forme de livre a disparu.

Transformer les données pour le Web

À NOTER

Les différents principes exposés sur la base de la taxonomie décrite par `article_v1.2.dtd` conviennent également pour des livres à destination du Web, la seule différence résidant dans la multiplicité des fichiers.

L'unité d'affichage ayant été décidée, il faut concevoir les traitements de transformation en s'appuyant sur les quelques considérations méthodologiques, organisationnelles et pratiques suivantes :

- préserver l'avenir ou les évolutions graphiques, en séparant la forme du fond : la mise en écran de la page HTML doit être totalement pilotée par une CSS;
- optimiser la page pour la rendre facilement indexable et identifiable par un moteur de recherche, donc ne pas l'obscurcir par des couches graphiques insérées dans le HTML, et éviter de placer les menus en début de document;
- organiser le flux HTML de sortie en séparant le bloc de contenu, c'est-à-dire l'information elle-même, du bloc de navigation (i.e. table des matières ou menu) issu de la structure hiérarchique (titres de chapitres, sections, sous-sections...) du document.

Une fois ces principes posés, on s'attache à rendre facilement repérables dans le fichier HTML de sortie les éléments à piloter graphiquement par la CSS. La pratique veut que l'on choisisse :

- des identifiants pour les éléments définis comme uniques dans la DTD, comme le titre de l'article, le sous-titre, le bloc de contenu (`#content`) et le bloc de menu (`#menu`). Le sélecteur CSS correspondant commence par le signe `#`;
- des valeurs de classe pour tous les autres composants (`.bibentry`, etc.). Le sélecteur CSS correspondant commence par un point.

Afin de permettre la navigation interne au sein du document, le fichier HTML de sortie doit également porter des identifiants, le plus souvent générés par programme, afin de repérer :

- les éléments représentatifs de la hiérarchie (titres, sous-titres, etc.) que l'on fera apparaître dans la table des matières. Dans l'exemple de l'article, et afin de ne pas multiplier les transformations à ce stade, les identifiants ont été posés à la main (`id-title="section1_title"`) en adoptant une règle de nommage séquentielle intuitive;
- les identifiants de notes en cas d'appels multiples (`id-note="note1"`);
- les identifiants d'objets prévus dans la DTD, figure (`id-figure`), entrée bibliographique (`id-bibentry`), discographique et vidéographique, afin de permettre des liens depuis le texte vers ces objets;

- les identifiants de blocs (ex. : tableau) et d'objets inline (span et ancres d'index) le cas échéant.

Enfin, le comportement électronique des éléments sémantiques balisés dans le flux XML, comme les entrées bibliographiques, doit être encodé en sortie directement dans le flux HTML sous forme d'une requête à passer en lien hypertexte au site cible (ex. : Google Scholar pour un article de revue).

Séparer la forme du fond : les classes

En termes de bonnes pratiques, il faut rappeler la nécessité de séparer la forme du fond au niveau du fichier HTML de sortie. Celui-ci ne comportera donc aucune information de mise en pages, mais des attributs de classe rappelant la position hiérarchique d'accès à un élément du XML (ex. : `section_level2_title` pour la classe correspondant au titre de sous-section, ou des attributs d'identifiant déjà présents dans le XML d'origine (ex. : `note1` pour la valeur de l'`id-note` d'une note).

Éléments inline. Les éléments inline, à l'exception de ceux qui ont le même nom qu'en HTML, seront presque tous transformés en `<span>` munis d'un attribut `class` que l'on choisira le plus souvent identique au nom de l'élément XML. Cette méthode a pour atout de simplifier la lecture du code HTML de sortie dès que l'on connaît la DTD ; elle présente également l'avantage de simplifier considérablement l'écriture des programmes XSLT de transformation qui deviennent à leur tour simples à lire.

Une bonne pratique consiste à bâtir un tableau de correspondance élément XML/élément HTML qui sert d'aide-mémoire et de documentation. Pour les notes de bas de page, on aurait le tableau suivant (voir aussi figure suivante) :

<code><noteentry></code>	<code></code>
<code><notecall></code>	<code></code>
<code><note></code>	<code></code>

et pour les entrées bibliographiques, ce tableau-ci :

<code><bibentry></code>	<code></code>
<code><bibentry><author><firstname></code>	<code></code>
<code><bibentry><author><lastname></code>	<code></code>
<code><bibentry><booktitle></code>	<code></code>

EN PRATIQUE

Les considérations méthodologiques valables pour la publication web le sont pour les autres transformations, vers le format epub en particulier.

```

<noteentry>
  <notecall>2</notecall>
  <note>
    <p>
      <bibentry>
        <author>
          <authorfirstname>Jack</authorfirstname>
          <authorlastname>Chambers</authorlastname>,</author>
          <booktitle>Milestones. The Music and Times of Miles Davis</
booktitle>, New York, Da Capo Press, 1998 (originellement publié en deux volumes
par Beech Tree Books, 1983 et 1985), p. 6 (vol. I).</bibentry>
        </p>
      </note>
    </noteentry>
  
```

La note, en HTML, est «emballée» dans un `<span>` muni d'un id posé par programme à fin de manipulations JavaScript ultérieures.

Le lien Google books n'a pas été indiqué ici afin de simplifier l'exemple.

```

<span id="d8e93">
  <span class="noteentry">
    <span class="notecall">2</span>
    <span class="bibentry">
      <span class="bibentry_author_firstname">Jack</span>
      <span class="bibentry_author_lastname">Chambers</span>,</span>
      <span class="bibentry_booktitle">
        <a>Milestones. The Music and Times of Miles Davis</a>
      </span>,</span> New York, Da Capo Press, 1998 (originellement publié en deux
volumes par Beech Tree Books, 1983 et 1985), p. 6 (vol. I).</span>
    </span>
  </span>

```

● Correspondance entre l'entrée XML (`<noteentry>`) et la sortie XHTML pour les notes de bas de page

Dans ces tableaux, la profondeur du pseudo-chemin indiqué comme valeur de la classe dépend uniquement de l'usage que l'on veut en faire au niveau de la CSS. Si l'élément `<booktitle>` n'a pas besoin d'être différencié (en terme graphique et en terme d'usage) lorsqu'il se trouve au sein d'un élément `<bibentry>`, on se contentera d'écrire en sortie `<span class="booktitle">`. Si l'on a besoin d'être plus précis ou si l'on veut être plus explicite, on écrira `class="bibentry_author_firstname"`.

CLASSES : RECHERCHER L'EXHAUSTIVITÉ

Ne pas confondre stratégie de numérotation des identifiants qui doivent par construction être uniques et stratégie d'attribution des classes qui, elles, sont répétables. On écrira : `id=title="section1"` pour la première section de premier niveau et `class="section_level1_title"` qui caractérise n'importe quel titre de section de niveau 1.

Pour les éléments récursifs, c'est-à-dire pouvant se contenir eux-mêmes, on adoptera une stratégie d'écriture faisant apparaître le niveau de profondeur : `<div class="section_level3_title">`, pour styler un titre de sous-sous-section (i.e. `<section><section><title>`).

Au niveau de la mise en pages, on fera ensuite correspondre des sélecteurs CSS aux classes du document XHTML.

Correspondance entre la sortie XHTML et les classes CSS d’affichage pour l’entrée bibliographique.

<code><a></code>	<code>a { font-style:italic; color:black; text-decoration:none; }</code>
<code><a></code>	<code>a:hover { background-color:#FFD5AA; }</code>
<code>bibentry_author_firstname</code>	<code>.bibentry_author_firstname { font-variant:small-caps; }</code>
<code>bibentry_author_lastname</code>	<code>.bibentry_author_lastname { font-variant:small-caps; }</code>

● Le rendu dans un éditeur XML : la couleur verte du texte est héritée de l’élément portant l’id-back, contrepartie HTML de l’élément `<back>` du XML.

Éléments blocs. Pour les éléments blocs, la démarche de définition des classes est comparable à celle des éléments inline : la sortie XHTML portera des attributs `class` dont la valeur est le plus souvent le nom de l’élément XML d’origine. On aura ainsi pour les différents paragraphes bibliographiques, discographiques et vidéographiques d’article_v1.2.dtd :

<code><pbib></code>	<code><p class="pbib"></code>
<code><pdisco></code>	<code><p class="pdisco"></code>
<code><pvideo></code>	<code><p class="pvideo"></code>

Pour les paragraphes ordinaires `<p>` ou les `<div>` qui se distinguent par leur attribut `class` dans le XML source, la correspondance est encore plus simple puisque l’on conserve les attributs tels quels en sortie : la balise `<p class="citation_source">` du XML sera transférée sans modification dans le HTML de sortie.



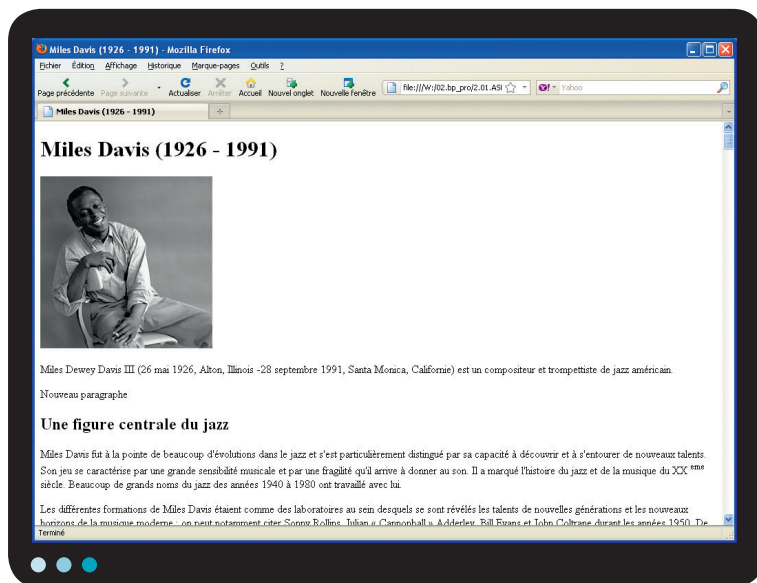
GÉNÉRER TOUTES LES CLASSES

Toutes les classes posées par le programme de transformation ne sont pas nécessairement utilisées par la CSS. Les générer reste une bonne pratique permettant au HTML de sortie de refléter assez bien la structure du XML, ce qui permet de styler ultérieurement la page sans avoir à retoucher les programmes de transformation.

● La page HTML affichée sans ses styles : retour à l'âge de pierre du Web, mais efficace pour le référencement.

Optimiser la page pour les moteurs de recherche

En matière de publication web, le référencement naturel est essentiel. Or, les moteurs de recherche sont plus à l'aise lorsqu'ils ont à indexer des pages très dépouillées et correctement hiérarchisées : on cherche donc à obtenir en sortie une page HTML organisée selon une hiérarchie simple faisant appel aux balises <hx> de HTML, le flux de texte étant organisé sous forme de div et de span emboîtés.



On veillera à organiser la page HTML en mettant en premier la div englobant le contenu de la page, et en fin de document HTML la div incorporant les éléments navigationnels (<div id="menu">) :

```
<html>
<head>
</head>
<body>
<div id="content">
  <h1 id="front_title">
  </h1>
  <h2 id="section1_title"> Titre de la section 1 <
  </h2>
</div>
```

```
<div id="menu">
</div>
</body>
</html>
```

Un bon moyen de vérifier l'application de ces principes consiste à afficher la page sans ses styles dans le navigateur : le résultat doit être une page correctement hiérarchisée, commençant par un titre de niveau 1 (<h1>) et suivi par des titres de niveau suivant, sans «trous» entre les niveaux.

DES HIÉRARCHIES COHÉRENTES

Veiller à organiser la page HTML en respectant la hiérarchie implicite des balises <hx>. Cette hiérarchie est explicitée dans le compte rendu produit par le site de validation HTML du W3C (validator.w3.org).

Poser les identifiants et valeurs de classe

Identifiants. La plupart des identifiants se posent par programme à partir du source XML : identifiants hiérarchiques, identifiants de blocs et identifiants d'éléments inline. Dans le cas d'`article_v1.2.dtd`, les identifiants sont posés de la manière suivante.

Pour les éléments hiérarchiques et englobants :

englobant content	<div id="content">
<front>	<div id="front">
<front><title>	<div id="front_title">
<body>	<div id="body">
englobant menu	<div id="menu">

Pour les éléments blocs :

<body><section>	<div id="section1">
<body><section><section>	<div id="section1_section1"> pour la première sous-section de la section 1 <div id="section1_section2"> pour la deuxième sous-section de la section 1...

Pour les titres, nécessaires à la génération des liens entre le menu de navigation et les titres eux-mêmes :

<body><section><title> (première section)	<h2 id="section1_title">
<body><section><section><title> (première sous-section)	<h2 id="section1_section1_title">
<body><section><section><section><title> (première sous-sous-section)	<h2 id="section1_section1_section1_title">

Si l'on craint que les identifiants deviennent trop longs, on adoptera un abrégé significatif : `s1_s1_s1_title` pour l'identifiant du titre de sous-sous-section.

Pour les éléments inline :

<code>note</code>	<code></code>
<code>bibentry</code>	<code></code>
<code>videoentry</code>	<code></code>

RAPPEL

Un identifiant XML (et donc XHTML) commence nécessairement par une lettre : `id="note1"` est valide, tandis que `id="1note"` est erroné.

Dans la pratique, il est inutile d'avoir un `id` signifiant : les programmes de pose se contentent de générer un numéro à plusieurs chiffres préfixés d'une lettre ou d'un ensemble de lettres.

Nota : lorsqu'il est fait référence de multiples fois au même objet, le lien est nécessairement posé à la main ; le programme de transformation doit

en tenir compte. C'est le cas des appels multiples à la même note. Selon la taxonomie définie dans `article_v1.2.dtd`, on écrira pour la première apparition d'une note à appels multiples :

```
<noteentry><callnote refid-
note="note1">1</callnote><note id-
note="note1">.....</note></noteentry>
```

et plus loin dans le document XML, l'appel à la note 1 :

```
.....<noteentry><callnote refid-
note="note1">1</callnote></noteentry>.....
```

Il va de soi que la valeur de l'appel de note (ici 1) peut être posée par programme ; seul l'`id-note` doit être posé manuellement par un opérateur, avec bien sûr une valeur unique pour tout le document.

Valeurs de classe. Les classes sont assignées à des éléments qui ne sont pas uniques par construction ; elles serviront à piloter le rendu typographique par une CSS. Une bonne pratique consiste à reprendre en attribut `class` soit les attributs de l'élément XML, soit le nom de l'élément XML.

On aura la correspondance suivante pour les blocs :

<code><div class="citation"></code>	<code><div class="citation"></code>
<code><div><title></code>	<code><h2 class="div_title"></code>
<code><div><div><title></code>	<code><h3 class="div_div_title"></code>
<code><deflist></code>	<code><table class="deflist"></code>

La `<deflist>` (liste de définition) est ici transformée en tableau à deux colonnes afin d'avoir un rendu plus agréable en sortie qu'une `<dl>` telle qu'elle existe en HTML.

Et pour les éléments inline :

<code><language></code>	<code></code>
<code><note></code>	<code></code>
<code><defterm></code>	<code><td class="defterm"></code>
<code><defdata></code>	<code><td class="defdata"></code>

Comportement électronique d'éléments sémantiques

Le comportement attendu en sortie d'un élément sémantique du flux XML fait appel à des transformations simples. Si l'on veut que le titre d'un livre dans une entrée bibliographique soit cliquable sous forme d'un lien dans Google books, il faut générer la requête dans le HTML de sortie sous une forme appropriée.

Un simple examen de la manière dont Google books passe sa requête une fois le formulaire de recherche rempli permet de répondre à plusieurs questions.

- Quelles sont les informations à baliser dans le XML source pour « populer » cette requête ?
- Comment passer les informations à Google books ?

La réponse à la première question conduit à limiter le balisage de la bibliographie au nom (et par élégance ou souci typographique au prénom, mais ce n'est pas utile pour la requête) de l'auteur et au titre de l'ouvrage. D'où la présence des balises `author` munies d'un attribut optionnel `main="yes"` dans le cas de l'auteur principal d'un ouvrage collectif, `authorfirstname`, `authorlastname` et `booktitle` dans la taxonomie `article_v1.2.dtd`.

On ne balisera la bibliographie de manière plus fine que dans l'hypothèse où l'on recherche un usage précis, par exemple la possibilité d'exporter les entrées bibliographiques dans un logiciel de gestion des références bibliographique (ex. : Endnotes). C'est un effort supplémentaire et coûteux dont il convient d'apprécier la légitimité économique.

Par extension, les éléments à repérer dans une entrée bibliographique portant sur un article de revue conduisent à baliser, outre l'auteur (`author` lui-même structuré en `authorfirstname` et `authorlastname`), le nom de la revue (`journaltitle`) et le titre de l'article (`articletitle`).

```

<p class="pbib">
<span class="bibentry">
<span class="bibentry_author_firstname">Ian</span>
<span class="bibentry_author_lastname">Carr</span>,
<span class="bibentry_booktitle">
  <a href="http://books.google.fr/books?as_q=&as_sauthors=Carr&as_vt=Miles+Davis">Miles Davis</a>
</span>
William Morrow & Co, 1982, traduction française : Éditions
Parenthèse, 1991 (ISBN 2-86364-057-7);
</span>
</p>

```

<authorlastname>

<booktitle>

● Correspondance entre éléments XML d'une entrée bibliographique et requête dans Google books. Seules deux balises XML sont utilisées : <booktitle> et <authorlastname>.

Une remarque toutefois : la grande disparité de notation des noms de revues – parfois mentionnés en abrégés et parfois en entier – au sein des bibliographies conduit à ne pas prendre en compte la balise `journaltitle` dans la requête : le bruit résultant à l'issue de l'exécution de la requête reste supportable, la combinaison du titre de l'article et du nom de l'auteur restant suffisamment restrictive.

Valider un XHTML

Il est de bonne pratique de s'assurer que les fichiers XHTML sont totalement conformes à leur DTD.

Rappelons que le W3C définit trois niveaux de XHTML :

- `xhtml1-frameset.dtd`;
- `xhtml1-transitionnal.dtd`;
- `xhtml1-strict.dtd`.

Ces DTD correspondent à différents niveaux de structuration, la plus contraignante, `xhtml1-strict.dtd`, est celle qui doit être retenue pour les transformations HTML : celles-ci indiqueront donc en tête du HTML résultant :

```

<!DOCTYPE HTML
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

```

Cette indication est utilisée par les services de validation en ligne du W3C (validator.w3.org), en choisissant l'onglet «Validate by file Upload».

Le validateur en ligne détecte les non-conformités par rapport à la DTD de référence et extrait du document HTML la hiérarchie fondée sur les balises de type `<hx>` : cela constitue un bon moyen de contrôler que les programmes de transformation n'utilisent pas les balises `<hx>` pour un rendu typographique.

- Le parseur de documents XHTML du W3C : `miles.html`, issu d'une transformation automatique comporte une erreur (présence de l'attribut `target`, non autorisé dans la DTD). On voit également, en bas de l'écran, le début de la structure (plan) du document basé sur les `<hx>`.

The screenshot shows the W3C Markup Validator interface in Mozilla Firefox. The browser title is "[Invalid] Markup Validation of miles.html - W3C Markup Validator - Mozilla Firefox". The address bar shows "http://validator.w3.org/check" and the page title is "[Invalid] Markup Validation of miles...".

Validation Output: 1 Error

Line 87, Column 91: **there is no attribute "target"**

... `<class="bibentry_booktitle"><a target="new" href="http://books.google.fr/books...`

You have used the attribute named above in your document, but the document type you are using does not support that attribute for this element. This error is often caused by incorrect use of the "Strict" document type with a document that uses frames (e.g. you must use the "Transitional" document type to get the "target" attribute), or by using vendor proprietary extensions such as "marginheight" (this is usually fixed by using CSS to achieve the desired effect instead).

This error may also result if the element itself is not supported in the document type you are using, as an undefined element will have no supported attributes; in this case, see the element-undefined error message for further information.

How to fix: check the spelling and case of the element and attribute, (Remember XHTML is all lower-case) and/or check that they are both allowed in the chosen document type, and/or use CSS instead of this attribute. If you received this error when using the `<embed>` element to incorporate flash media in a Web page, see the [FAQ item on valid flash](#).

Document Outline

Below is an outline for this document, automatically generated from the heading tags (`<h1>` through `<h6>`.)

- [h1] Miles Davis (1926 - 1991)
 - [h2] Une figure centrale du jazz
 - [h2] Biographie
 - [h3] 1926-1944: l'apprentissage
 - [h3] 1944-1948 : les années Bebop

http://validator.w3.org/feedback.html?uri=;errmsg_id=108#errmsg

Annotations:

- L'attribut `target` n'est pas autorisé par la DTD référencée.
- Structure du document HTML repérée à partir des balises `<hx>`.

Publier au format epub

epub est le format de publication préféré des éditeurs. Ouvert et élaboré par l'industrie de l'édition américaine via l'IDPF, il sert de base à la plupart des publications électroniques de livres. Universel et simple, il offre l'interopérabilité (lecture en synchronie d'un fichier sur écran, tablette et téléphone...).

Le format epub

ACRONYMES

IDPF : International
Digital Publishers Forum
(www.idpf.org)
OPF : Open Packaging
Format

Le format epub, quasiment devenu la norme pour la publication d'e-books, est un format très simple destiné à la publication de livres électroniques. Sa mise au point, inspirée de trois normes préexistantes (OPS, OPF, OCF) accessibles en ligne sur le site de l'IDPF, s'est appuyée sur des considérations de base et de bon sens :

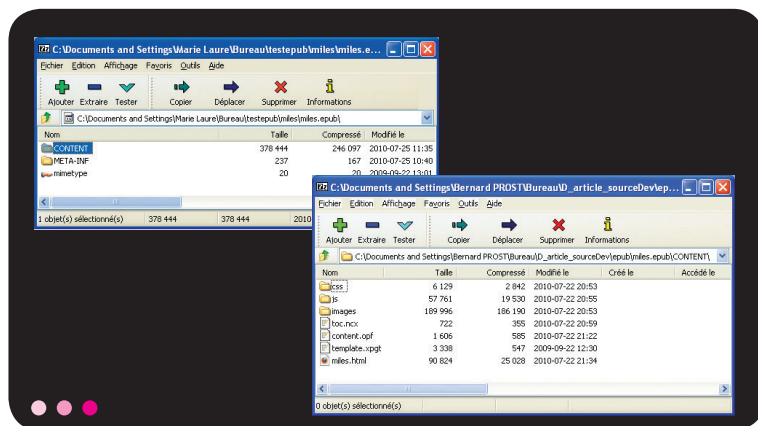
- le fichier de publication doit être unique pour être facilement transportable d'un système à l'autre ;
- le format doit être interopérable, c'est-à-dire fonctionner sur tout type d'appareil et de logiciel se conformant à la norme epub ;
- le texte doit être repaginable (i.e. *flowable*), c'est-à-dire s'appuyer sur un mécanisme permettant de l'inscrire dans l'écran de présentation quel que soit le matériel de restitution – tablette, ordinateur, téléphone – sans avoir à le faire défiler verticalement ou horizontalement ;
- l'affichage s'appuie sur une représentation XHTML (prochainement HTML 5), avec ou sans CSS ; il s'agit d'un HTML complet avec toutes les possibilités d'hyperliens. JavaScript est autorisé, mais son interprétation dépend des logiciels de lecture du fichier epub, ce qui, actuellement, freine son interopérabilité ;

- le découpage des fichiers est libre : on respectera par exemple le découpage en parties, chapitres et sections que l'on fera apparaître dans la table des matières. Rien n'empêche – à l'exception notable de certaines liseuses qui n'acceptent pas de gros fichiers (300 Ko maximum pour la liseuse de Sony), de n'utiliser qu'un seul fichier ;
- la navigation s'appuie sur un fichier XML, généralement suffixé `.ncx`, représentant l'ordre et la hiérarchie des entrées de la table des matières. Celles-ci pointent sur les fichiers XHTML grâce au mécanisme classique `id/idref` de XML ;
- l'ensemble des fichiers nécessaires à la publication, fichiers XHTML, images et plus généralement ressources (vidéo, son) est décrit dans un seul fichier XML, nécessairement suffixé `.opf`, ce qui permet au niveau des outils de contrôle de s'assurer que toutes les ressources invoquées dans l'epub sont bien présentes ;
- les fichiers sont organisés en trois groupes – un fichier unique et deux répertoires – respectant des règles d'organisation ;
- avec quelques contraintes de priorité, la totalité des fichiers est zippée en un seul fichier dont l'extension est `.epub` et non pas `.zip`.

Créer un epub

Ouvrir un epub existant

L'epub étant un fichier zippé, il est simple de l'explorer en l'ouvrant à l'aide de n'importe quel outil sachant reconnaître le format zip ; on choisira soit un outil open source comme TUGZip (www.tugzip.com) ou 7-Zip (www.7-



- Ouverture d'un epub avec 7-Zip : les fichiers peuvent être modifiés soit en les extrayant par copier/glisser sur le bureau et en les réintégrant après correction, toujours par copier/glisser, soit en les éditant directement (touche F4 à condition d'avoir préalablement associé un éditeur à 7-Zip via la commande Outils>Option).

zip.org), soit un outil commercial comme winzip (www.winzip.com) ou winrar (www.win-rar.com), tous fonctionnant en mode graphique sous Windows. Ce type d'outil permet de retoucher facilement l'epub (la CSS en particulier) en extrayant localement le fichier à modifier et en le réintégrant dans l'epub par copier/coller après correction.

À noter : en raison de leur gestion spécifique de l'ordre des fichiers, 7-Zip comme TUGZip en mode graphique ne permettent pas d'obtenir des epub irréprochables lorsque l'on crée une archive au format epub. En ouverture et retouche de fichiers en revanche, ils ne posent pas de problème.

Créer manuellement un epub

La création manuelle d'un epub est une opération simple, une fois le HTML disponible. Conformément à la norme, un epub comporte trois fichiers obligatoires, dont deux (`mimetype` et `container.xml`) ont un nommage imposé, et deux dossiers obligatoires dont un seul (`META-INF`) a un nommage imposé :

1. Le fichier `mimetype` (en minuscules et sans extension), dont le nom est imposé, ne comporte qu'une seule ligne de texte :

```
application/epub+zip
```

Ce fichier doit être placé en premier dans l'archive `.epub` et ne doit pas être compressé. Stocker un si petit fichier peut sembler bizarre, mais il permet à l'application de lecture – Stanza, iBooks, etc. – d'identifier très rapidement un fichier de type epub avec plus de certitude que la seule extension `.epub`.

2. Le dossier `META-INF`, dont le nom (en capitales) est imposé, contient au moins un fichier dont le nom (`container.xml`) est également imposé. Ce dernier est un fichier XML dont le vocabulaire, contrôlé par une DTD, est fixé par la norme. Son élément racine est `<container>`. Il contient à minima, et dans la plupart des cas, un accès à un fichier de configuration de l'epub (`xxx.opf`) :

```
<?xml version="1.0" encoding="UTF-8"?>
<container xmlns="urn:oasis:names:tc:opendocument:xml
lns:container" version="1.0">
  <rootfiles>
    <rootfile
      full-path="CONTENT/package.opf" media-
      type="application/oebps-package+xml"/>
  </rootfiles>
</container>
```

Les attributs `xmlns` et `media-type` doivent être repris tels quels ; leur complexité n'est qu'apparente. Contrairement à `mimetype` le dossier `META-INF` peut être compressé, mais ce n'est pas obligatoire, surtout en phase de mise au point de l'epub.

3. Un dossier dont le nom est libre, par exemple `\CONTENT` (on trouve presque toujours `OEBPS`), où sont regroupés tous les fichiers de contenu, HTML, CSS, JavaScript, etc. ainsi que deux fichiers obligatoires :

- `xxx.opf`, généralement appelé `package.opf` (on trouve aussi `content.opf` et `meta.opf`).

Ce fichier a nécessairement l'extension `.opf` et il doit être unique à porter cette extension au sein de `\CONTENT` ;

- `xxx.ncx`, presque toujours appelé `toc.ncx` bien que le nom ainsi que l'extension soient libres.

`Package.opf` et `toc.ncx` nécessitant d'être bien compris dans leur structure si l'on veut maîtriser le format epub, nous les décrivons ici de manière plus détaillée.

package.opf

Le fichier `package.opf` est un fichier XML unique dont le vocabulaire est contrôlé (élément racine `<package>`). Il comprend trois sections :

- les métadonnées (`<metadata>`), conformes à la taxonomie du Dublin Core (les éléments sont donc préfixés `dc`). Selon la norme, seules trois métadonnées sont obligatoires : `<dc:title>` pour le titre du document, `<dc:identifier>` pour stocker un identifiant, `<dc:language>` pour la langue principale du document :

```
<metadata>
  <dc:title>Miles davis</dc:title>
  <dc:identifier id="prostbookid-876523-456">prostbook</dc:identifier>
  <dc:language>FR</dc:language>
</metadata>
```

ATTENTION À LA COMPRESSION

Lors de la mise au point de l'epub, il est préférable de ne pas compresser les dossiers : on gagne en rapidité et en facilité, les paramètres étant fixés sur « aucune compression ». Ce n'est qu'au moment de la publication que l'on modifiera ceux-ci.

ACRONYME OEBPS

Open EBook Publication Structure

ACRONYME NCX

Signifie soit *Navigation Control for XML*, soit *Navigation Center eXtended*, suivant les écoles. Il s'agit de la table des matières.

- iBooks utilise deux métadonnées de fait obligatoires si l'on veut figurer correctement dans la bibliothèque d'e-books de l'iPad :
 - `<dc:subject>` pour faire apparaître la catégorie de rattachement du livre;
 - `<dc:creator>` pour faire figurer l'auteur.
- la déclaration de toutes les ressources (`<manifest>`), c'est-à-dire tous les fichiers utiles pour l'epub : la table des matières (`toc.ncx`), éventuellement le fichier de configuration d'Adobe pour Adobe Digital Edition (`.xpgt`), la CSS principale et ses CSS dépendantes s'il y a lieu, tous les fichiers XHTML, toutes les images, etc. Chaque ressource est décrite dans un élément `<item>` muni de 3 attributs : `id` pour identifier la ressource, `href` pour pointer sur le fichier ressource et `media-type` pour qualifier techniquement la ressource :

<MANIFEST>

To manifest signifie « montrer » en anglais. Un manifest est un fichier montrant toutes les ressources utilisées dans un projet. Ce mot, très général, se retrouve dans de nombreux environnements informatiques.

```
<manifest>
<item id="miles_germany"
href="images/miles_germany.jpg"
media-type="image/jpeg"/>
</manifest>
```

SPINE

Il est important de comprendre que le `<spine>` ne sert qu'à donner l'ordre d'affichage des fichiers dans les pages du livre lors du feuilletage. Si l'on met en premier le chapitre 2, le feuilletage du livre commencera par le chapitre 2, même si la table des matières est donnée dans la séquence correcte. Les deux notions sont donc indépendantes.

Ces informations sont, entre autres usages, utilisées par les outils de contrôle pour vérifier la conformité du format du fichier déclaré dans `href`.

- un fichier squelette (`<spine toc="ncx">`) donnant les identifiants des fichiers XHTML utilisés pour l'affichage du texte du livre dans les pages. Dans l'exemple d'`article_v1.2`, il n'y a qu'un seul fichier `xhtml`, donc une seule information dans `<spine>` :

```
<spine toc="ncx">
.....
<itemref idref="main"/>
.....
</spine>
```

L'attribut `toc` est obligatoire. Dans cet exemple, l'identifiant auquel il est fait référence (`idref="main"`) renvoie à un identifiant utilisé dans la zone `<manifest>` sous la forme `<item id="main" href="miles.html">`.

toc.ncx

Le fichier `toc.ncx` est un fichier XML unique à vocabulaire imposé (racine `<ncx>`); il regroupe :

- un en-tête `<head>` listant quatre valeurs de métadonnées sous forme de paires `name/content` :
- `<meta name="dtb:uid" content="xxx">` : `xxx` est un identifiant unique (`uid` signifie `user id`) pour l'ouvrage. Ce peut être un numéro ISBN ou tout identifiant généré par un programme permettant de produire des `uid` (*universal user id*);
- `<meta name="dtb:depth" content="N">` : `N` définit la profondeur de la table des matières sous forme d'un entier. `depth = "1"` correspond aux niveaux supérieurs d'un livre (`front`, `body` ou `back`);
- `<meta name="dtb:totalPageCount" content="N">` : métadonnée qui n'a pas d'effet pratique en matière d'e-books; en pratique, on met cette valeur à 0;
- `<meta name="dtb:maxPageNumber" content="N">` : métadonnée qui n'a pas d'effet pratique en matière d'e-books; en pratique, on met également cette valeur à 0.

● La correspondance entre les imbrications de `navPoint` et le rendu hiérarchique de la table des matières dans ADE



Miles Davis
Wikipedia

Une figure centrale du Jazz

Biographie

1926-1944 : l'apprentissage
1944-1948 : les années bebop
1948-1949 : naissance du Cool
1949-1955 : drogues et Hard Bop
1955-1957 : le 1er grand quintet
1957-1959 : vers le Jazz Modal
1960-1968 : le second grand quintet
1968-1975 : la révolution électrique
1981-1991 : Miles superstar

Citations et anecdotes

Discographie

Album studio

```
<?xml version="1.0" encoding="UTF-8"?>
<navMap>
  <navPoint id="main" playOrder="1">
    <navLabel>
      <text>Miles Davis</text>
    </navLabel>
    <content src="miles.html"/>
  <navPoint id="section1_title" playOrder="2">
    <navLabel>
      <text>Une figure centrale du Jazz</text>
    </navLabel>
    <content src="miles.html#section1_title"/>
  </navPoint>
  <navPoint id="section2_title" playOrder="3">
    <navLabel>
      <text>Biographie</text>
    </navLabel>
    <content src="miles.html#section2_title"/>
  <navPoint id="section2_section1_title" playOrder="4">
    <navLabel>
      <text>1926-1944 : l'apprentissage</text>
    </navLabel>
    <content src="miles.html#section2_section1_title"/>
  </navPoint>
  .....
</navPoint>
</navMap>
```

- un titre de document `<docTitle>`. On notera la majuscule sur le t de Title, la DTD de définition du vocabulaire `ncx` est un peu ancienne et a conservé les habitudes de la typo dite « pauvre-riche » censée rendre le code plus lisible. En pratique, aucun ereader ne tient compte de cette balise ;
- la navigation elle-même sous forme d'une structure d'emballage `<nav-Map>` de différentes entrées hiérarchiques (`<navPoint>`). L'élément `navPoint` étant récursif, il permet de définir n'importe quelle profondeur de hiérarchie : un `navPoint` dans un `navPoint` introduit ainsi une section dans la table des matières. `<navPoint>` exige deux attributs `id` et `playOrder`, ce dernier devant respecter une séquence stricte (1, 2, 3, etc.).

Produire l'epub

REMARQUE

Les outils de compression zip permettent, dans une seule archive, de mixer des fichiers avec des taux de compression différents. Le fichier `mimetype` doit nécessairement être stocké avec une compression nulle, tandis que les autres fichiers peuvent être compressés au maximum ou pas du tout compressé si on le souhaite.

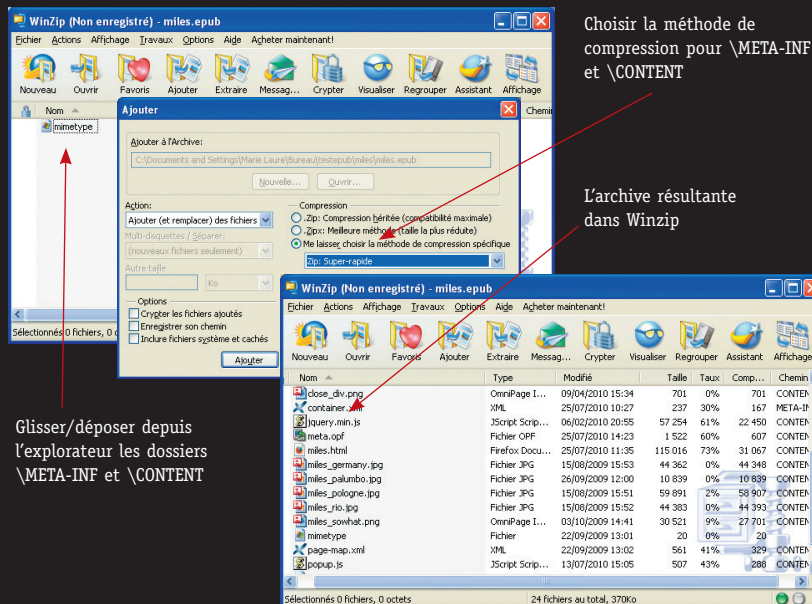
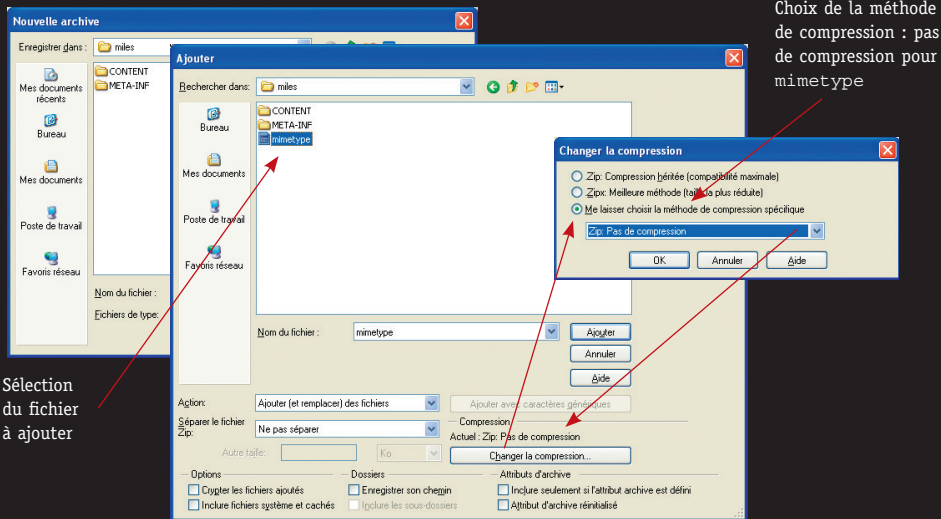
Dans le cas d'`article_v1.2` le processus a été simplifié au maximum : le fichier XHTML produit précédemment pour la publication web est repris comme fichier unique XHTML de l'epub. Les fichiers obligatoires `package.opf` et `toc.ncx` sont produits par transformations XSLT depuis le XML source en respectant la cohérence d'ensemble (ordre des fichiers dans le `<spine>` en particulier). Les fichiers sont finalement intégrés dans une archive unique suffixée `.epub` à l'aide de l'utilitaire de zippage préservant l'ordre des fichiers (ex. : Winzip), en deux étapes :

- création de l'archive que l'on baptisera d'entrée de jeu `xxx.epub` (inutile de passer par un nommage en `.zip` et un renommage ultérieur) et intégration du premier fichier `mimetype` sans compression ;
- ajout à l'archive des dossiers `\META-INF` et `\CONTENT` par glisser/déposer avec ou sans compression.

Une fois l'archive créée, il suffit de fermer l'epub et de le contrôler d'abord rapidement en l'ouvrant localement dans Adobe Digital Edition, de manière plus rigoureuse ensuite en le passant dans un *checker* d'epub (programme spécialisé de validation).

Valider un epub

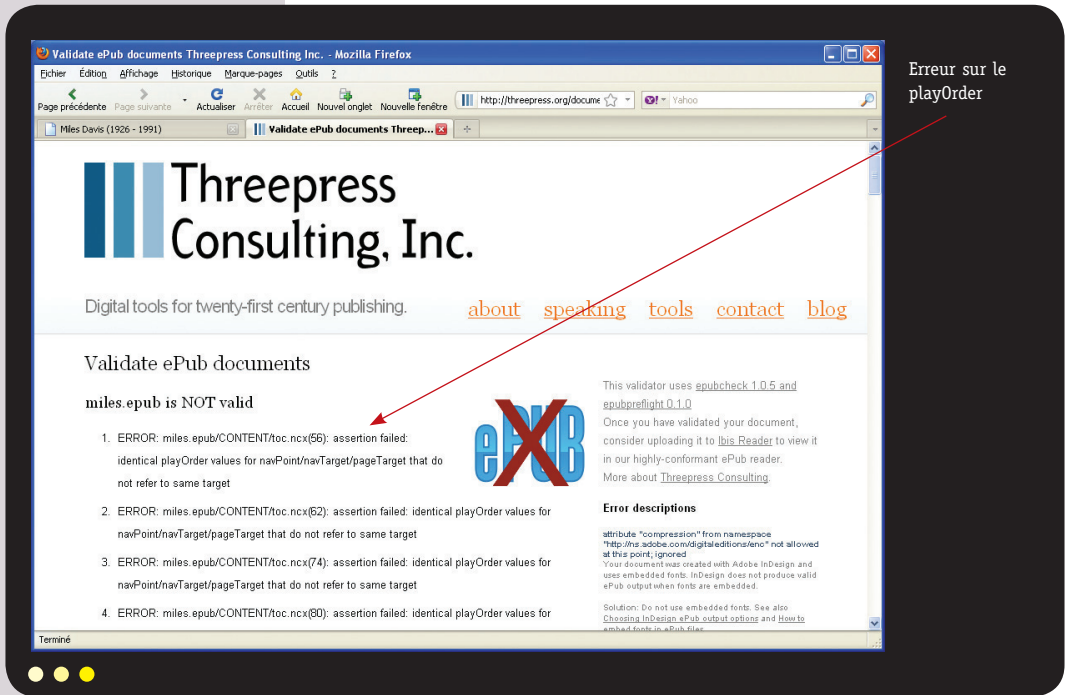
Le format epub comportant de nombreux fichiers dès que l'ouvrage est un peu complexe ou doté d'images, il est facile de lui faire perdre sa conformité à la norme, et cela même s'il s'ouvre sans difficulté dans Adobe Digital Edition : il convient donc de le soumettre, à l'instar de la validation des XHTML, à un



● Création de l'archive (baptisée .epub) et stockage du premier fichier mimetype sans compression à l'aide de Winzip

● Ajout par glisser/déposer des dossiers \META-INF et \CONTENT avec ou sans compression

contrôle de validité. Une bonne méthode consiste à utiliser un service en ligne comme celui de Liganan (www.mybookforge.com) ou de Threepress (www.threepress.org).



● Service en ligne de contrôle d'un epub : ici, le contrôle est négatif en raison d'une erreur de déclaration technique d'espace de nom sur les scripts JavaScript.

À noter : les utilitaires open source comme 7-Zip ou TUGZip, dans leur version graphique, ne sont pas en mesure de créer une archive se conformant aux contraintes de l'epubchecker en raison de leur méthode interne de stockage des fichiers : ainsi, il n'est pas possible de forcer le `mimetype` en première position. Cela devrait rapidement être corrigé dans les nouvelles versions de ces zippers compte tenu de la diffusion rapide du format epub.

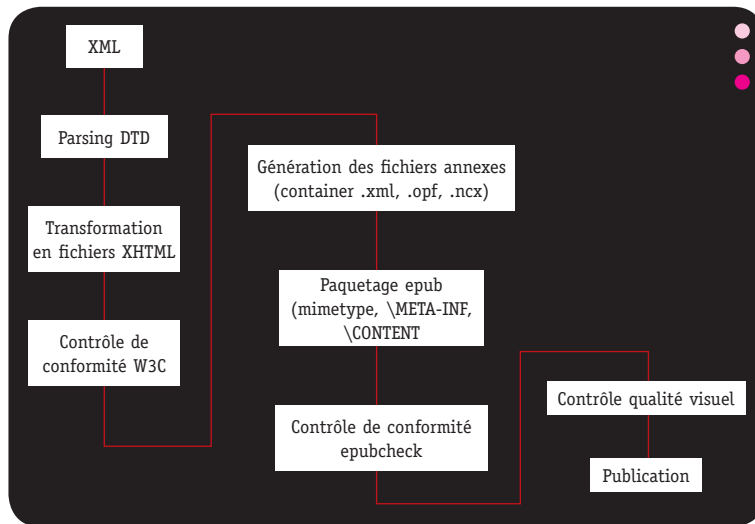
Convertir le XML au format epub

Dans la pratique, les opérations de conversion sont effectuées par une cascade de transformations partant du XML source pour produire les fichiers HTML cibles de la même manière que pour une publication HTML pour le Web. Elles peuvent se réaliser en utilisant n'importe quel langage informatique, l'essentiel pouvant être fait avec XSLT.

Il convient de s'appuyer sur une méthodologie afin de maîtriser la correspondance entre l'organisation du XML et la sortie obtenue, l'objectif étant d'être productif dans la transformation et de ne pas avoir à retoucher à la main l'epub. On adoptera en particulier une stratégie de contrôle à chaque étape de la transformation.

INDESIGN ET L'EPUB

Les logiciels de PAO comme InDesign proposent une sortie directe en epub sans avoir à passer par XML. Le résultat est satisfaisant pour des structures très simples comme des romans, mais nettement moins convaincant dès que l'ouvrage a une structure un peu plus complexe.



● Les principales étapes d'une chaîne de fabrication d'epub

Convertir l'epub en mobipocket (Kindle)

La conversion en mobipocket se fait très facilement à l'aide du convertisseur KindleGen en mode ligne de commande ; cette approche n'est pas très conviviale et réservée aux utilisateurs en possédant les rudiments.

KindleGen se télécharge sur le site d'Amazon (<http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000234621>). Le bouton de téléchargement est en fin de page. L'installation s'effectue de préférence sur le disque C:\ d'un PC (sinon il faut éditer une variable d'environnement) et le lancement de la conversion s'effectue dans une fenêtre DOS, par la commande :

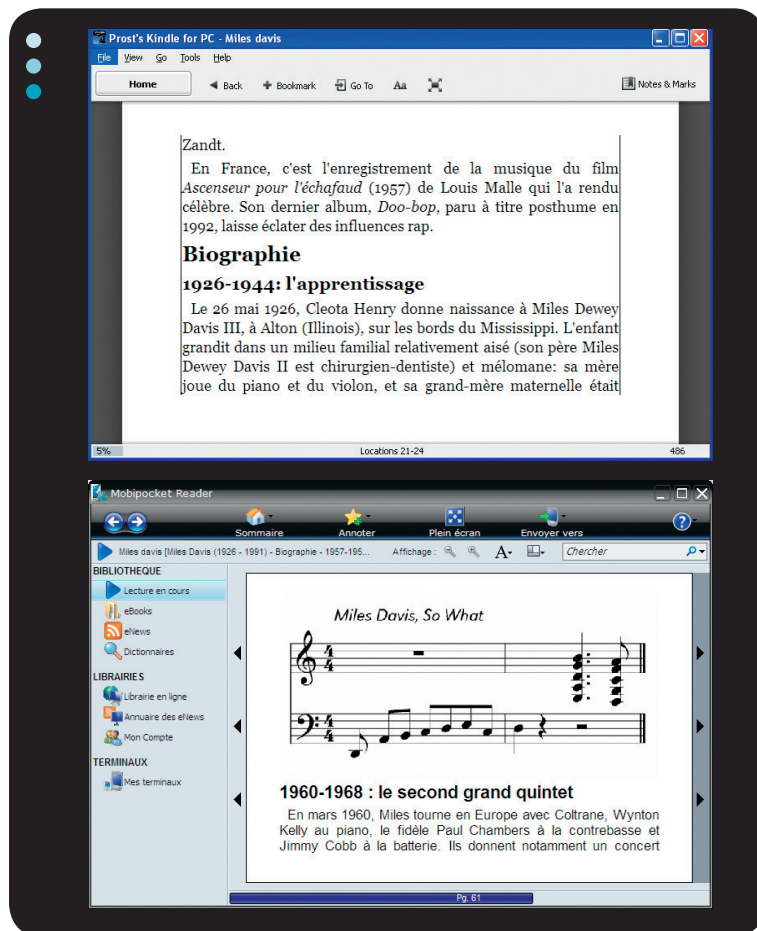
kindlegen miles.epub

Le résultat, si Kindlegen n'a pas détecté d'erreurs, ce qui suppose au moins que l'epub ait passé le contrôle de l'epubcheck, est un fichier suffixé .mobi (donc miles.mobi dans l'exemple) qui s'écrit dans le même répertoire que celui de l'epub.

Pour ouvrir le .mobi, et si l'utilisateur ne dispose pas d'une liseuse Kindle, il convient de télécharger sur son PC (Kindle for PC) ou son téléphone un lecteur de .mobi. Pour le téléphone, le lecteur se trouve sur le site de mobipocket (mobipocket est le nom du lecteur Kindle avant son rachat par Amazon) :

mobipocket.com>logiciels>mobipocket ebook reader

● Miles.epub converti en .mobi lu par le reader PC d'Amazon (Kindle PC) et le reader PC de mobipocket



Les supports cibles

La lisibilité du rendu visuel et l'interactivité possible dépendent des logiciels et matériels de lecture fort disparates actuellement. L'effervescence autour du livre électronique devrait susciter l'amélioration et l'émergence d'un ou deux lecteurs largement répandus, tel Acrobat Reader pour les fichiers PDF.

Lire un epub : les logiciels de lecture

epub n'est qu'un format ouvert de stockage et de transmission de l'information, à l'instar de HTML dont il s'inspire. La lecture d'un livre électronique nécessite l'utilisation d'un logiciel tiers capable d'interpréter les informations pour les rendre lisibles sur un écran en adoptant une métaphore plus ou moins proche de celle du livre papier.

DÉFINITION **ereader**

Ce mot anglais désigne un logiciel de lecture d'epub, et plus généralement de toute publication électronique de type livre.

Il n'est pas difficile d'écrire un logiciel de lecture d'epub puisque toute la structure du livre est décrite en XML (XHTML). Les logiciels qui existent ont l'inconvénient de ne pas proposer le même rendu visuel, de ne pas supporter de manière homogène les CSS et le JavaScript et, pour certains, d'introduire des compléments propriétaires censés apporter une meilleure expérience de lecture : la situation actuelle est donc comparable à celles des premières versions des navigateurs (Netscape, Internet Explorer) lors du décollage d'Internet, obligeant les développeurs à adapter leur code à chacune d'elles.

Cela constitue bien sûr un frein à la diffusion de la lecture électronique en attendant que le marché fasse lui-même ses choix.

En matière de lecteurs d'epub, on distingue actuellement deux catégories :

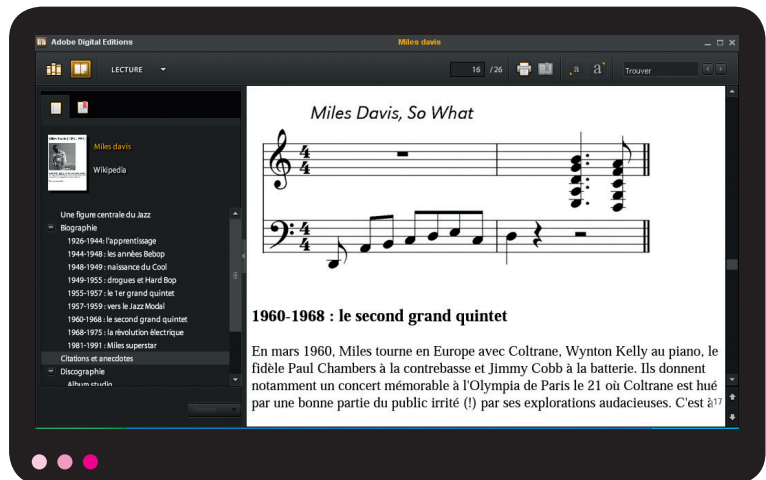
- les lecteurs *stand alone* nécessitant l'installation d'un logiciel sur la machine cible : ADE, Stanza, iBooks. Ils sont dépendants des caractéristiques logicielles des machines cibles, l'OS en particulier;
- les lecteurs web, qui « détricotent » l'epub et le transforment en publication web sous forme de pages HTML reliées entre elles par des boutons de navigation : Ibis Reader, MyBookForge Reader.

ADE

Adobe Digital Edition (ADE) est l'un des premiers lecteurs d'epub publiés. Il fonctionne en *stand alone* sur des ordinateurs de bureau Mac et PC, mais pas Linux, et n'existe pas en version téléphone. Il offre les fonctionnalités classiques d'un lecteur : table des matières, interactivité, repagination à la volée, support honorable des polices de caractères embarquées. Parmi ses inconvénients, on notera une esthétique peu engageante, un rendu sommaire, un faible support de CSS et de JavaScript et une tolérance excessive vis-à-vis des erreurs d'epub : un epub ayant échoué aux tests de validation pourra très bien fonctionner sur ADE, ce qui rend cet outil impropre à un contrôle qualité.

ADE utilise un fichier propriétaire (dont l'extension est `.xpgt`) installé dans `/CONTENT` afin de piloter différents paramètres de style dont certains déjà pris en charge par la CSS. Cette extension, directement issue du vocabulaire XSL-FO (voir le chapitre 9), permet de multicolonner la page et de préciser divers aspects géométriques, marges, etc. de la page dont le rendu ne sera perceptible que sur ADE.

● Miles.epub affiché sur
ADE PC (version 1.7.2 février
2010)



Stanza

Stanza, développé par la société Lexcycle (www.lexcycle.com), maintenant rachetée par Amazon, a été spécifiquement développé pour l'iPhone, avant d'avoir été porté sur l'iPad. Sa version PC, Stanza Desktop, actuellement dans sa version bêta est décevante.

Miles.epub sur Stanza
iPhone et Stanza iPad

Miles Davis (1926 - 1991)



Miles Dewey Davis III (26 mai 1926, Alton, Illinois - 28 septembre 1991, Santa Monica, Californie) est un compositeur et trompettiste de jazz américain.

Une figure centrale du jazz

Orange F 21:39

Lire Table des Matières

Miles Davis (1926 - 1991)

Une figure centrale du jazz

Biographie

1926-1944: l'apprentissage

1944-1948 : les années Bebop

1948-1949 : la naissance du Cool...

1949-1955 : drogues et Hard Bop

1955-1957 : le premier grand qu... ✓

Table des Matières Marque-pages Annotations

Cobb disait que ce disque
« avait dû être composé au paradis »



1960-1968 : le second grand quintet

En mars 1960, Miles tourne en Europe avec Coltrane,

Miles Davis (1926 - 1991)



Miles Dewey Davis III (28 septembre 1991, San Francisco) est un compositeur et trompettiste de jazz américain.

Une figure c

Miles Davis fut à la pointe le jazz et s'est particulièrement caractérisé par une grande fragilité qu'il arrive à dominer du jazz et de la musique grands noms du jazz des avec lui.

Les différentes formations

curieux, une superstar du jazz. À la différence de son glorieux aîné qui avait recherché l'intégration à la culture grand public dominée par la population blanche, le parcours musical de Miles Davis s'accorde position politique en faveur de la cause contre le racisme, menée avec la cause humaine au caractère réputé émancipateur à l'album *Sun City* contre l'apartheid de Steven van Zandt.

En France, c'est l'enregistrement de *Ascenseur pour l'échafaud* (1957) de Loïc célèbre. Son dernier album, *Doo-bop*, en 1992, laisse éclater des influences :



1960-1968 : le second

En mars 1960, Miles tourne en France avec Wynton Kelly au piano, le fidèle contrebasse et Jimmy Cobb à la batterie. Un concert mémorable à 21 où Coltrane est hué par une foule irritée (!) par ses explorations audacieuses, en avril, où Coltrane officie au sein du groupe de Miles Davis.

Miles retourne en Europe et à l'occasion du saxophoniste parlant de Miles se montre plus agressif et aussi plus proche d'un hard bop orthodoxe. Le mélodieux Hank Mobley tiendra le difficile rôle de remplaçant de Trane à partir de 1961 alors que Wynton Kelly est le pianiste du groupe. On peut l'entendre dans quelques titres de l'album *Someday My Prince Will Come* et dans les disques live *Miles Davis In Person : Friday Night & Saturday*

ne sans même m'en rendre compte (...) Chaque nuit, Herbie, Tony et Ron rentraient dans leur chambre et discutaient jusqu'au petit matin de ce qu'ils venaient de jouer. Le lendemain, ils remontaient sur scène et jouaient différemment. Et moi, soir après soir, il fallait que je m'adapte...

— Miles Davis avec Quincy Troupe, L'autobiographie (1989)

uniquement de compositions originales, le groupe reprend les standards du répertoire de Miles Davis (All of You, My Funny Valentine...) en concert. Lors de ces concerts, on entend le groupe à son meilleur. Shorty y montre toutes ses qualités de soliste et la section rythmique brille par sa cohésion et son inventivité prodigieuses.

évolution électrique

Alors que le rock et le funk se développent, Miles Davis va initier l'essor d'un jazz de style nouveau, fusionnant

Stanza (version 3.0.3, mars 2010) offre une expérience de lecture acceptable sur un téléphone comme sur l'iPad et une métaphore de page de livre meilleure que celle d'ADE. Il est également plus solide en matière de support de CSS, mais reste encore très primitif. Parmi ses avantages, l'intégration d'un ouvrage par téléchargement, très simple sur un iPhone : il suffit, sur la page de publication d'un site, de stocker le lien vers l'epub selon un pseudo protocole :

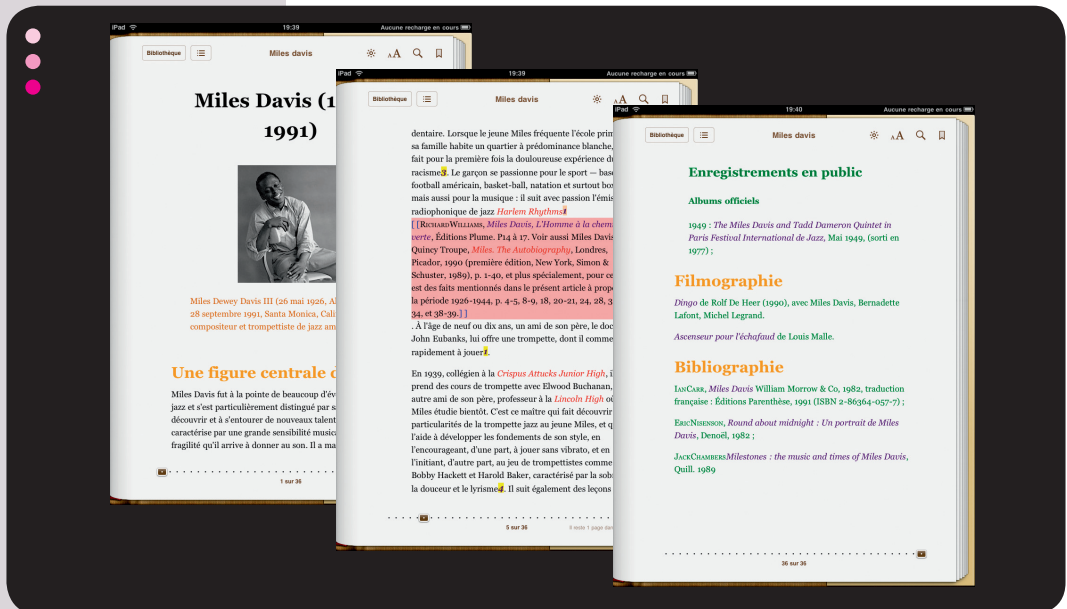
```
<a
href="stanza://{adresse_complète_page/fichier.
epub}">.....</a>.
```

Le livre se téléchargera automatiquement en cliquant sur le lien, à condition bien sûr d'avoir installé préalablement Stanza sur le téléphone ou l'iPad cible.

iBooks

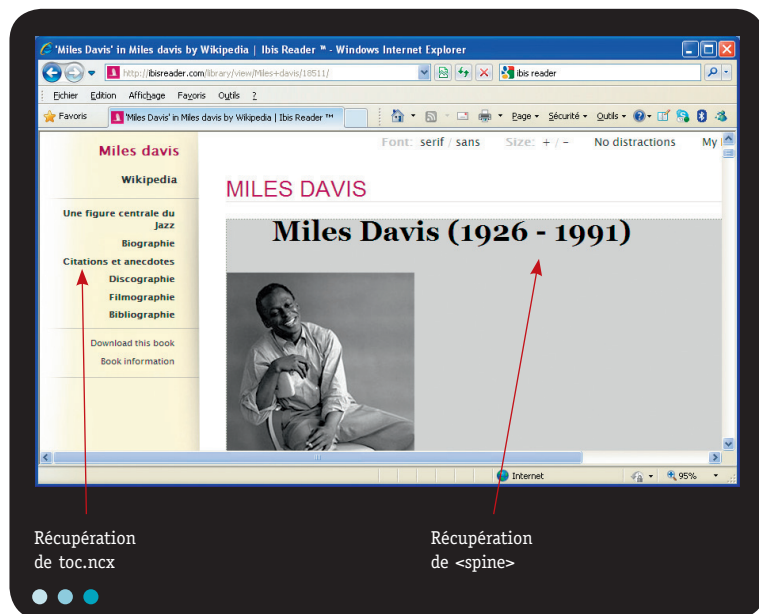
Le logiciel de lecture d'epub proposé par Apple sur l'iPad (version 1.1.1, mai 2010) est le plus abouti esthétiquement. Son support de CSS2 est meilleur que celui de ses compétiteurs, comme son support de SVG pour les graphiques. À l'instar de ses concurrents, il est indigent en matière de MathML : les formules ne sont pas lisibles (les flèches des vecteurs s'écrivent à la suite du segment de vecteur...), ce qui impose de publier celles-ci sous forme d'images.

● Miles.epub sur iBooks



Ibis Reader

Ibis Reader est un lecteur web qui « démonte » en ligne l'épub et restitue les pages HTML avec une navigation sommaire. Dans ses grandes lignes, le processus consiste à décompresser les pages HTML de \CONTENT et à leur adjoindre des éléments de navigation (boutons Suivant, Précédent) en se basant sur la succession des fichiers décrits dans <spine>. Si l'épub est monofichier, alors la sortie sera monofichier. L'approche revient donc à de la publication HTML, la qualité graphique en moins.



● Miles.epub dans Ibis Reader : on se rapproche de la publication HTML, sans l'esthétique ni les fonctionnalités.

Les tablettes

La grande révolution de 2010 en matière de lecture électronique est sans conteste l'apparition des tablettes tactiles : surface d'affichage suffisante pour une lecture soutenue, lisibilité excellente et rapport à l'objet rappelant le feuilletage du livre papier devraient en faire un vecteur de diffusion efficace pour la lecture électronique.

Le premier coup a été tiré par Apple, avant tous ses (nombreux) concurrents, avec une qualité qui positionne iBooks comme le premier lecteur logiciel sérieux du marché. Le transfert des epub vers l'iPad se fait via iTunes par glisser/déposer avec la convivialité que l'on connaît.

Production

Le lecteur de l'iPad supporte mieux que ses concurrents certains styles CSS et en ignore superbement d'autres ; il autorise six polices, tandis que l'iPad en supporte une trentaine. Pour un éditeur, la conséquence est d'importance : il lui faut concevoir sa publication électronique pour qu'elle soit la plus réussie sur l'iPad sans être trop dégradée sur les autres systèmes, sachant que la technologie progresse dans le même temps : tous les concurrents supporteront l'essentiel de CSS2 puis CSS3 ainsi qu'HTML 5.

Il lui faut aussi admettre que ses publications epub devront être régénérées à intervalles réguliers, annuellement ou tous les six mois, afin de tenir compte à la fois de l'évolution technologique des tablettes et de l'évolution du texte publié, pour les ouvrages professionnels ou techniques tout au moins. Techniquement, cela ne pose aucun problème dès que le source est en XML ; en revanche, les epub produits par exportation d'InDesign ou prochainement des outils bureautiques nécessiteront un travail manuel supplémentaire non négligeable, sans compter l'introduction de délais de publication cruciaux dans le domaine professionnel.

RETAILLE DES IMAGES

La retaille des images doit se faire automatiquement dans la chaîne de production : si un nouvel appareil concurrent de l'iPad permet de zoomer dans les images, il sera possible, à partir des sources, de produire ultérieurement des images que l'on bornera par programme à 900 points en largeur et 1000 points en hauteur.

Un autre impact est celui de la taille des images. L'écran de l'iPad compte 768 × 1024 points ; le lecteur d'iBooks offre 600 × 860 points utiles. Si le zoom d'image ne constitue pas une fonctionnalité décisive (exploration d'une œuvre d'art par exemple), il est inutile de charger l'epub d'images surdimensionnées. On se limitera dès lors à 600 points en largeur et environ 700 points en hauteur pour laisser de la place à la légende et être optimal pour l'iPhone 4.

Publication

REMARQUE

Rappelons qu'un des prérequis pour publier sur l'iBookstore est une conformité aux tests de validation de l'epubcheck.

Pour la publication sur l'iBookstore, le processus est à peine moins délicat que la publication sur l'Appstore pour le téléphone : mieux vaut actuellement passer par un agrégateur qui aura démonté les rouages des processus imposés par Apple.

Les téléphones

Le téléphone reste un support de lecture, certes assez primaire, mais largement répandu et surtout disponible en permanence. L'extrême réduction de l'écran d'affichage impose des contraintes qui se répercutent sur la production des objets de publication numérique : les tableaux devront être retraités pour qu'ils s'affichent de manière satisfaisante, la simple modification via une CSS orientée téléphone ne suffisant pas. Conséquences pour l'éditeur : l'objet numérique de publication pour téléphone est une publication à part entière qui suit sa propre chaîne de production.

Pour la publication principale, 3 approches peuvent être prises en compte :

- publication HTML en ligne à orientation téléphone comme l'a fait O'Reilly pour certains de ses ouvrages. La démarche est la même que pour la publication desktop : ouverture d'un compte et feuilletage en ligne ;
- publication sous forme d'une application dédiée. Cette approche est relativement coûteuse car il faut passer à chaque fois par un développement plus ou moins spécifique, sachant qu'il est toujours possible de packager une application HTML (i.e. la publication sous forme de web App) en une seule application par type de téléphone ;
- la publication sous forme d'un fichier epub que l'on diffusera via des distributeurs ou des agrégateurs.

Cela dit, il semble peu probable que la lecture d'ouvrages complets sur téléphone se répande ; à notre avis, l'édition pour téléphone s'orientera vers des textes courts ou des publications complémentaires à la publication numérique principale (i.e. sur les tablettes), surtout dans le domaine professionnel. On pourrait avoir :

- des publications de type *digest* ou aide-mémoire, qui apportent une réelle utilité à l'internaute par leur disponibilité permanente, tout en provoquant une frustration suffisante pour rendre l'ouvrage princeps indispensable ;
- des bandes annonces prenant le plus souvent l'aspect d'un extrait d'une partie du premier chapitre mais qui pourrait être l'ouvrage complet tronqué au niveau de chaque chapitre.

● Miles.epub en publication web-téléphone. Cette publication pourrait se packager sous forme d'App.



Production

Quelles que soient les approches retenues, l'impact en termes de production s'inscrit à différents niveaux :

- taille des images. Un processus de retaille automatique des images doit être mis en œuvre. L'iPhone 3 ayant un écran de 320 × 480 et l'iPhone 4 un écran de 640 × 960, on convertira les images dans les limites de l'iPhone 4, par exemple 600 × 800 ou 600 × 700 pour une compatibilité avec l'iPhone 4. Les styles CSS comporteront un sélecteur testant la largeur d'affichage du téléphone cible et adapteront l'affichage au matériel cible ;
- retraitement des tableaux. Le téléphone ne supporte pas visuellement les tableaux au-delà de quatre colonnes. Il convient donc soit de revoir éditorialement les tableaux en les restructurant avec quatre colonnes au maximum, soit de les retraiter par logiciel ;
- adaptation des styles : on optera pour une navigation proposant un menu statique ou un système de pliage/dépliage de la table des matières à la manière de Wikipedia mobile dans le cas d'une publication HTML. En mode epub, les styles utilisés pour les publications tablettes sont acceptables.

La spécificité de l'écran de téléphone a conduit au développement de bibliothèques JavaScript orientées mobiles :

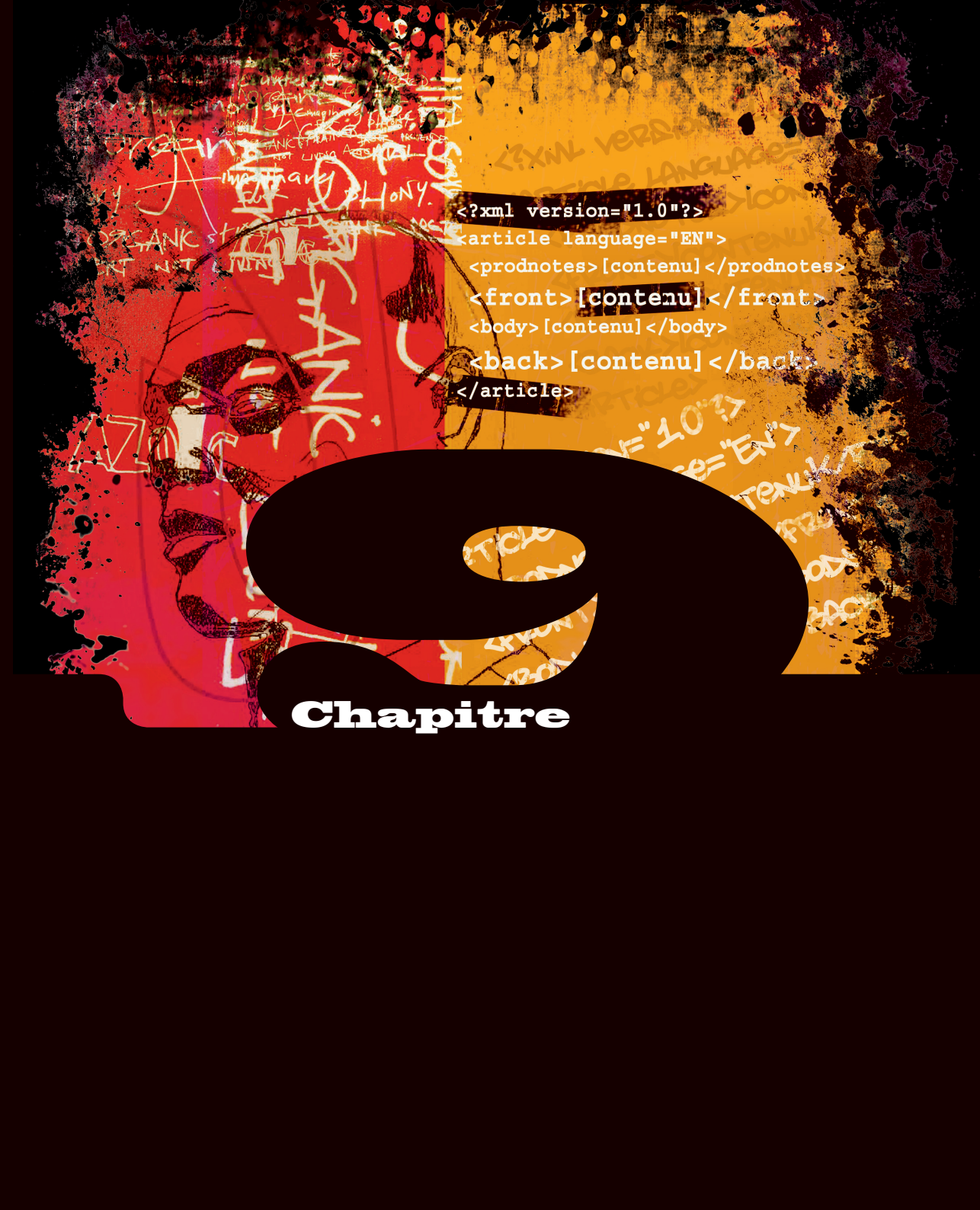
- jQuery (jquery.com) dont la visée va bien au-delà de la publication sur téléphone;
- XUI (xuijs.com) très similaire à jQuery dans sa syntaxe;
- IUI (iPhone User Interface, code.google.com/p/iui).

On mentionnera également iWebKit (iwebkit.net) qui permet de donner une apparence proche de celle d'une publication sur iPhone en n'utilisant que des styles CSS.

Ces environnements mobiles permettent « d'augmenter » la publication principes et donnent, pour certains d'entre eux, la possibilité de distribuer la publication sous forme d'application classique sur l'Appstore. Dans tous les cas, on veillera à conserver un processus automatisé partant du XML auquel on applique une ou plusieurs transformations, les fichiers HTML résultants appelant les bibliothèques listées ci-dessus.



- Affichage d'un tableau en 4 colonnes (© Bordas/Memobac)



```
<?xml version="1.0"?>
<article language="EN">
  <prodnotes>[contenu]</prodnotes>
  <front>[contenu]</front>
  <body>[contenu]</body>
  <back>[contenu]</back>
</article>
```

Chapitre



Publier pour le papier

XML peut servir à publier pour le papier. La technologie actuelle propose deux approches :

- l'automatisme complet, grâce au langage de description de page XSL-FO (technique à réserver aux ouvrages relativement simples dans leur mise en pages) ;
 - la semi-automatisation, où la source XML alimente un outil de PAO (pour les documents complexes, mais nécessitant une intervention humaine pour finaliser la mise en pages).
- 

XSL-FO : un langage XML ouvert de description de page

Le langage XSL («S» signifiant StyleSheet) prévu pour automatiser la composition, papier ou écran, s'est scindé en deux sous-langages, XSLT, pour décrire les transformations de toutes sortes, et qui connaît un grand succès, et XSL-FO, pour décrire la géométrie des pages et le placement de ses principaux objets, de diffusion plus confidentielle.

XSL-FO est un langage XML permettant de décrire graphiquement les pages d'un document et leur contenu afin qu'un processeur XSL-FO puisse les composer dans le rendu final selon différents formats, PDF en particulier. La manière d'obtenir le fichier XSL-FO descripteur du contenu de la mise en pages est

indifférente : le plus souvent, on écrit en XSLT les ordres de transformation du XML vers un fichier XSL-FO, mais rien n'empêche de créer celui-ci à la main (i.e. avec un éditeur de texte minimal) ou à partir de n'importe quel langage informatique.

ACRONYME **XSL-FO**

XSL Formatting Object.

Pour en savoir plus :

www.w3.org/Style/XSL

Le vocabulaire XSL-FO comporte une soixantaine de mots réservés destinés à identifier les blocs, les structures inline, la géométrie de la page, etc., chaque élément étant doté de nombreux attributs (au sens XML) destinés à décrire de manière très fine les attributs typographiques. Ce vocabulaire XML est assez verbeux, mais présente l'avantage d'être parfaitement explicite ; ainsi, un bloc (conceptuellement comparable à un bloc InDesign) se décrira sous la forme :

```
<fo:block>.....</fo:block>
```

On notera au passage le préfixe `fo:` accolé au nom de chaque balise, selon le mécanisme d'espace de noms comparable à celui de XSLT (qui, lui, utilise le préfixe `xsl`).

Les attributs destinés à spécifier tous les paramètres typographiques du bloc sont inspirés de CSS et donc très faciles à lire. L'écriture `<fo:block font-size="16pt" space-before="8mm">` spécifie que, lors du coulage, le texte du bloc se composera en corps 16 pt et que la mise en pages sera réalisée avec un espacement de 8 mm avant le bloc.

Comment produire un fichier XSL-FO ?

Il serait trop pénible de créer à la main un fichier XSL-FO représentatif d'une publication ; aussi, le principe consiste à appliquer au fichier XML une transformation XSLT produisant en sortie le flux XSL-FO cible représentatif des pages. Le fichier XSL-FO est ensuite repris par un processeur externe (« processeur XSL-FO ») qui va créer la représentation finale attendue, le plus souvent sous forme d'un fichier PDF ou PostScript.

Représentation de la mise en pages à l'aide d'XSL-FO

Le fichier XSL-FO étant un fichier XML standard, il comporte les deux composants de base d'un tel fichier :

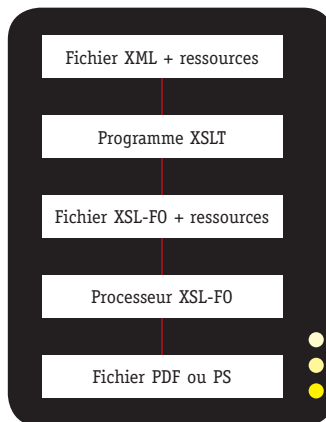
- la déclaration de version XML (`<?xml version="1.0"?>`) ;
- un élément unique racine `<fo:root>` doté d'un attribut `xmlns:fo` servant à déclarer l'utilisation du préfixe `fo:` dans la suite du fichier, selon le mécanisme standard de déclaration des espaces de nom (`xmlns = xml name space`) :
`<fo:root xmlns:fo="http://www.w3.org/1999/Format">`

L'élément `<fo:root>`, à son tour, se décompose en deux parties :

- une partie destinée à déclarer un jeu de modèles appelés « pages-maîtres », à la manière des gabarits d'InDesign : `<fo:layout-master-set>` ;

EN PRATIQUE

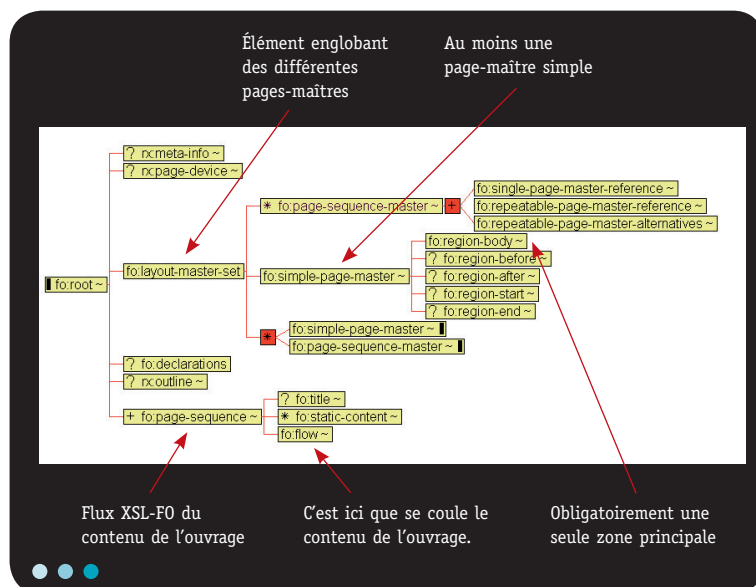
XSL-FO est actuellement disponible en version 1.1 (décembre 2006) ; la version 2.0 candidate est en évaluation depuis mars 2008. XSL-FO étant un langage XML contrôlé, il existe une DTD déclarant toutes les balises et leur comportement.



● Les étapes d'une transformation XSL-FO

● Organisation générale des informations dans un fichier XSL-FO représentatif d'une mise en pages

- une partie déclarant, dans l'ordre de la composition, les pages de contenu <fo:page-sequence>, celles-ci appelant les différentes pages-maîtres correspondant à la maquette.



Les pages-maîtres

Les pages-maîtres sont des modèles de page présentant des caractéristiques particulières aussi bien au niveau de leur géométrie qu'à celui de la disposition des informations dans la page. Par exemple, on déclarera pour la composition d'un livre de poche les pages-maîtres suivantes :

À RETENIR

Un fichier XSL-FO comporte au moins une page-maître.

- les pages de titres, de copyright, dédicaces, etc. ;
- les pages paires et les pages impaires du corps de l'ouvrage et pour chaque type de page (paire ou impaire) :
 - le contenu de la marge supérieure : titre courant rappelant le nom de l'ouvrage placé à gauche pour les pages paires et titre du chapitre placé à droite pour les pages impaires ;
 - le contenu de la marge inférieure : folio placé à gauche sur les pages paires et placé à droite sur les pages impaires.

Les pages-mâitres («page-master» dans le jargon XSL-FO) sont repérées par un identifiant placé en valeur d'un attribut `master-name`, et sont directement munies d'attributs typographiques spécifiant leur taille, les marges, etc. L'identifiant de la page-maître sera utilisé lors de la description de la séquence des pages (`<fo:page-sequence>`) pour appeler telle ou telle page-maître.

La déclaration d'une page-maître paire, dotée d'une marge uniforme de 1,5 cm s'effectuerait ainsi :

```
<fo:simple-page-master master-name="body-even"
margin="1,5cm" page-width="137mm" page-
height="189mm">
.....
</fo:simple-page-master>
```

ce qui déclare une page paire courante de 137 × 189 mm dotée d'une marge uniforme de 1,5 cm sur les quatre côtés. Cette page-maître est immatriculée par son identifiant `body-even`, ce qui permettra de l'invoquer lors de la description de la séquence des pages.

Une page impaire pourrait se déclarer ainsi :

```
<fo:simple-page-master master-name="body-odd"
margin="1,5cm" page-width="137mm" page-
height="189mm">
.....
</fo:simple-page-master>
```

La valeur des identifiants `master-name` (ici `body-odd` et `body-even`) est arbitraire ; il suffit qu'ils soient uniques dans le fichier XSL-FO. L'ordre des attributs est également indifférent, comme pour tout document XML.

À l'intérieur des pages-mâitres, il est possible de déclarer cinq zones, appelées «régions» dans le jargon XSL-FO, chacune ayant ses propres caractéristiques géométriques :

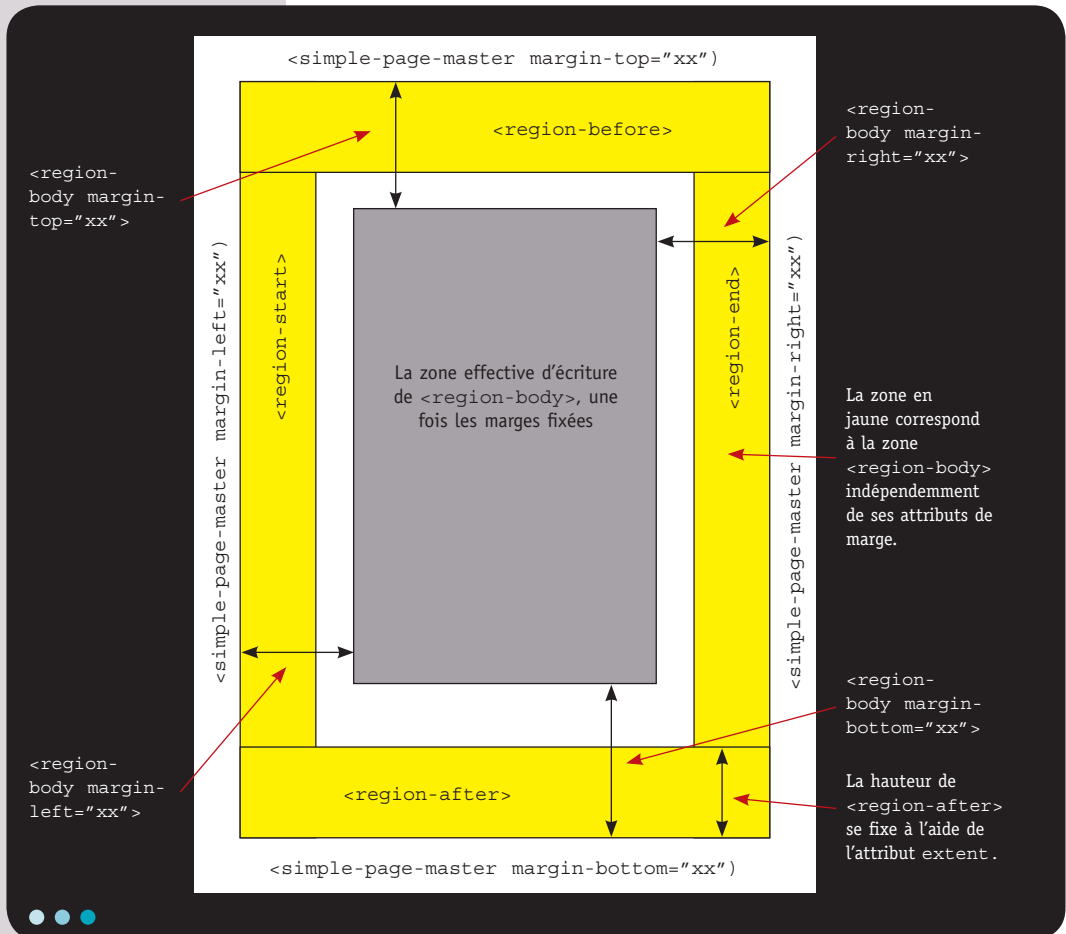
- une zone principale dans laquelle se coulera l'essentiel du texte : `<fo:region-body>`. Cette zone est obligatoire puisqu'elle reçoit le contenu textuel du livre ;
- quatre zones périphériques facultatives concernant le haut, la droite, le bas et la gauche de la `region-body` : `<fo:region-before>`, `<fo:region-end>`, `<fo:region-after>`, `<fo:region-start>`. Ces zones sont munies de différents attributs, extent en particulier, qui permettent d'en spécifier la taille. Il importe de retenir que ces zones s'ins-

DÉNOMINATION DES RÉGIONS

Les concepteurs de XSL-FO ont choisi des noms neutres pour repérer le début (start), la fin (end), le haut (before) et le bas (after) d'une page afin qu'ils soient indépendants de la géométrie physique de la page. En arabe ou en hébreu, le start d'une page est à droite; en chinois il est en haut, en français il est à gauche. Une dénomination de type CSS (top, right, bottom, left) aurait été plus claire.

● Géométrie d'une page-maitre simple (simple-page-master) : les informations prises en compte par XSL-FO.

crivent dans la region-body et qu'il faut choisir correctement leurs dimensions afin qu'elles ne se chevauchent pas inconsidérément (sauf si l'on cherche un effet de recouvrement spécifique...) : on veillera, en particulier, à doter <region-body> de marges suffisantes pour recevoir les informations de régions périphériques, le plus souvent <region-before> (haut



de page) pour les titres courants et `<region-after>` (bas de page) pour les folios.

Notez qu'il existe également des «super pages-maîtres» qui rassemblent dans le seul élément `<fo:page-sequence-master>` une séquence de pages-maîtres simples (`<fo-simple-page-master>`), ce qui permet pratiquement de décrire la complexité de n'importe quel ouvrage.

Les séquences de pages

Les pages-maîtres ne sont que des modèles : le coulage de l'information s'effectue dans des réceptacles appelant les pages-maîtres selon une séquence décrite dans des éléments `<fo:page-sequence>` successifs. Ceux-ci utilisent deux paramètres :

- d'une part, la référence du nom de la ou des pages-maîtres à utiliser ; l'appel s'effectue sous forme d'un attribut `master-reference` dont la valeur est l'identifiant `master-name` de la page-maître appelée : `<fo:page-sequence master-reference="body-odd">` signifie que la page-maître dont l'identifiant est `master-name="body-odd"` est utilisée.

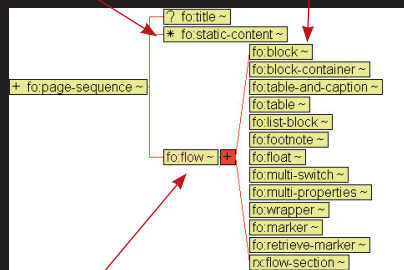
ATTENTION

On essaie de donner des noms signifiants aux pages-maîtres, par exemple le fait que l'on s'intéresse à une page impaire (ce que suggère le mot `body-odd`), mais ils ne sont pas connus de l'ordinateur. Il appartient à l'auteur du fichier XSL-FO d'être cohérent dans le choix des noms et les traitements attendus.

● Les éléments constitutifs d'une page-sequence

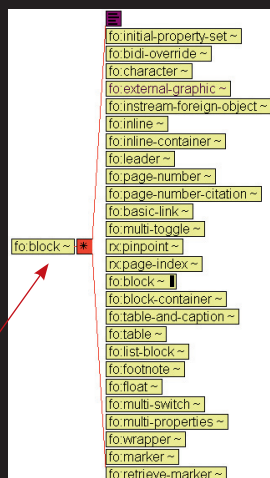
`<fo:static-content>`
pour spécifier les zones
fixes à remplir (titres
courants, folio, ...)

Conteneur bloc
simple



`<fo:flow>` :
conteneur général

`<fo:block>` :
conteneur
de texte, de
structures in-
line et d'autres
conteneurs-blocs



- d'autre part, des ordres de coulage dans les différentes régions, sous forme d'un élément `<fo:flow>` doté d'un attribut `name` explicitant la région qui recevra le coulage : `<fo:flow name="xsl-region-body">` signifie que le coulage se fera dans la `region-body`. Selon la région concernée, l'attribut `name` de `<fo:flow>` vaudra `xsl-region-body`, `xsl-region-before`, `xsl-region-after`, etc.

À l'intérieur de chaque `<fo:flow>` pour chaque région, des ordres de coulage spécifiques à différents objets sont prévus. Le plus courant est le contenu de bloc simple `<fo:block>`, mais on trouve des contenus spécifiques pour les listes (`<fo:list-block>`), les tableaux (`<fo:table>` et `<fo:table-and-caption>`), les images (`<fo:external-graphic>`), etc.

Sachant que chaque objet XSL-FO possède de très nombreux paramètres typographiques, y compris des ordres d'insécabilité afin d'empêcher qu'une légende ne se détache d'une figure par exemple, on comprend que l'on dispose d'une représentation très fine des pages.

En pratique, la mise en pages d'un ouvrage est donc représentée par un (gros) fichier XSL-FO indiquant, sous forme de suite, le placement de tout ce qui doit être composé. Il reste à rendre visible cette composition à l'aide d'un « moteur de composition » (également appelé « moteur de rendu ») : il s'agit d'un programme prenant en entrée le fichier XSL-FO ainsi que ses différentes ressources graphiques – images, signalétique, etc., pour produire en sortie une page visuellement organisée. Celle-ci sera générée dans un format technique de sortie qui peut être choisi, la plupart des moteurs offrant une sortie PDF ou PostScript.

Mettre en œuvre une composition XSL-FO

La mise en œuvre d'une composition XSL-FO à partir d'un flux XML ne diffère pas beaucoup de celle d'une transformation à destination de l'électronique ; elle s'effectue en trois étapes :

- création par un directeur artistique (DA) d'un gabarit visuel en recherchant un design optimal facilitant le coulage (ce qui suppose une bonne expérience) ;
- mise en œuvre d'un programme de transformation XSLT destiné à produire le fichier XSL-FO représentatif de la mise en pages finale ;
- composition finale par un moteur de rendu comme XSL Formatter d'Antenna House ou le logiciel libre FOP.

L'écriture du programme de transformation XSLT nécessite de revenir fréquemment sur les différentes étapes et éventuellement sur le fichier XML source selon une boucle classique :

- optimisation du design graphique afin que le coulage s'effectue sans heurt majeur (négociation et pédagogie programmeur/DA) ;
- mise au point des transformations XSL-FO pour régler la mise en pages automatique (programmeur) ;
- éventuellement retouche du XML par un programmeur ou, idéalement, un maquettiste, le plus souvent au niveau des tableaux et des images afin que la mise en pages reste élégante. On utilise à cette fin quelques attributs spécifiques du média cible (ici l'impression) que l'on aura prévus dans la DTD, comme `print-scale="x%"` destiné à régler à la main l'échelle de l'objet imprimé.

En termes de méthodologie, une bonne coordination doit être assurée entre le concepteur graphique et l'auteur du programme XSLT de transformation : il faut en effet que chaque bloc, chaque élément inline soit spécifié en termes typographiques. Si la maquette a été créée en PAO, il faudra que le DA transmette au programmeur, élément par élément, ce qu'il doit indiquer dans son programme de transformation XML vers XSL-FO, ou bien qu'il s'asseye à côté de lui afin de travailler conjointement au paramétrage.

La mise au point finale s'effectue toujours en binôme entre le graphiste et le développeur : le graphiste, au vu du résultat de mise en pages automatique, exprime en termes typographiques ce qui doit être modifié, sachant que si le programme XSLT est bien modulaire, sa lecture est aisée et parfaitement à la portée d'un maquettiste ou même d'un éditeur. Le plus souvent, le graphiste coulera dans son outil de PAO préféré des portions de texte ou d'informations correspondant au flux réel de données à traiter afin de travailler sur un prototype visuel réaliste ; il indiquera manuellement sur l'impression papier les principaux paramètres typographiques à retenir, sachant qu'une charte minimale aura été écrite afin de spécifier les polices de caractères et les paramètres typographiques minimaux.

Pages-maîtres

Pour les pages-maîtres, il est pratique de transmettre un tableau des principaux paramètres géométriques que le programmeur reprendra dans son programme de transformation. Dans le cas d'article_v1.2, la présentation des correspondances aura l'allure du tableau ci-dessous.

Principaux paramètres typographiques des pages paires et impaires de la mise en pages d'un article

	Page de gauche paire (even) (mm)	Page de droite impaire (odd) (mm)	
page-width	137	137	
page-height	189	189	
margin-top	16	16	
margin-bottom	16	16	
margin-right	18	25	Symétrie des pages paires et impaires
margin-left	25	18	
region-body/margin-top	16	16	Marge du haut de 16 mm sur la zone region-body afin de libérer de la place pour la région before (i.e. haut de page).
region-body/margin-bottom	16	16	Marge du bas de 16 mm sur la zone region-body afin de libérer de la place pour la région after (i.e. bas de page).
region-before extent	16	16	La hauteur réservée pour la zone region-before (i.e. en haut) est de 16 mm. Contenu : titres courants • impaire : titre article, fer à droite, • paire : catégorie ("WIKIPEDIA" dans l'exemple), fer à gauche.
region-after extent	16	16	La hauteur réservée pour la zone region-after (i.e. en bas) est de 16 mm. Contenu : folios • impaire : fer à droite, • paire : fer à gauche.

L'écriture XSL-F0, pour la page paire (even) s'exprime ainsi :

```
<fo:simple-page-master
    master-name="even"
    page-width="137mm"
    page-height="189mm"
    margin-top="16mm"
    margin-bottom="16mm"
```

```

margin-right="18mm"
margin-left="25mm">

<fo:region-body
  margin-top="16mm"
  margin-bottom="16mm"/>

<fo:region-before
  extent="16mm"/>

<fo:region-after
  extent="16mm"/>

</fo:simple-page-master>

```

Cette écriture n'est pas très intuitive en raison des noms choisis pour exprimer les régions : *before*, *end*, *after*, *start* correspondent en effet aux positions habituellement exprimées par *top*, *right*, *bottom*, *left*.

Contenu (flot)

Pour le contenu, généralement désigné par le terme « flot », on spécifie chaque type d'objet – titre, titres de sections de niveaux 1, 2, etc., liste, tableau, notes – à partir d'une représentation visuelle traduite en termes typographiques.

Les titres de sections pour les deux premiers niveaux (section et sous-section) s'exprimeraient comme dans le tableau ci-dessous.

Paramètres typographiques des titres de sections et de sous-sections

Titre 1 (section)	Times New Roman bold color #C12802 font-size 20 pt text-align center paragraphe solidaire (keep-with-next always) espace avant mini, optimum, maximum : 2 mm, 5 mm, 15 mm espace après mini, optimum, maximum : 2 mm, 4 mm, 4 mm
Titre 2 (sous-section)	Times New Roman bold color orange font-size 14 pt text-align left paragraphe solidaire (keep-with-next always) espace avant mini, optimum, maximum : 1 mm, 2 mm, 3 mm espace après mini, optimum, maximum : 1mm, 2 mm, 2 mm

Le XSL-FO correspondant, obtenu par transformation XSLT, s'écrit pour le titre 1 :

```
<fo:block
  font-family="Times New Roman"
  font-weight="bold"
  color="#C12802"
  font-size="20pt"
  text-align="center"
  keep-with-next="always"
  space-before.minimum="2mm"
  space-before.optimum="5mm"
  space-before.maximum="15mm"
  space-after.minimum="2mm"
  space-after.optimum="4mm"
  space-after.maximum="4mm">
  Une figure centrale du jazz
</fo:block>
```

On comprendra au vu de cet exemple que l'écriture du programme de transformation est minutieuse : on réservera donc l'approche XSL-FO pour des productions répétitives comme une collection.

De XSL-FO à PDF : les moteurs de rendu

Il existe peu de moteurs de rendu XSL-FO sur le marché. Ils sont coûteux et réservés à des professionnels, ce qui est assez injuste vis-à-vis de la norme, car XSL-FO est aussi simple que XML. Gageons que ce langage prendra de l'importance lorsque les navigateurs seront capables de l'interpréter : les mises en pages seront bien meilleures, en particulier au niveau de l'impression, que celles proposées aujourd'hui par défaut dans ces mêmes navigateurs.

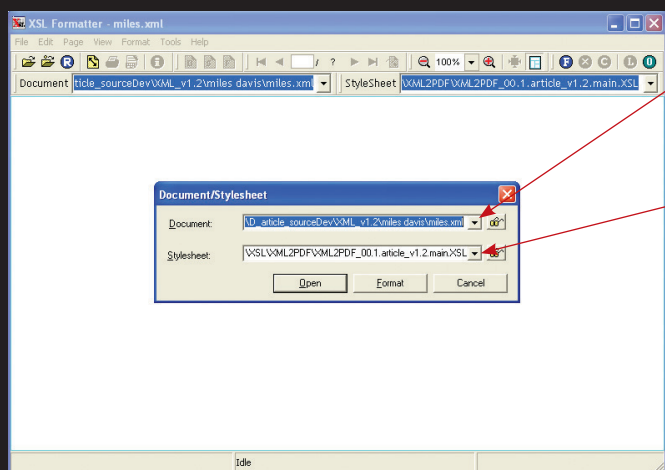
À NOTER

Des extensions assurant le support de SVG pour les graphiques et de MathML pour les équations sont disponibles sous forme d'options dans XSL Formatter.

Pour notre part, nous utilisons XSL Formatter de la société Antenna House. Ce logiciel, qui fonctionne sur toutes les plateformes (Windows, Linux et Mac), existe en version serveur et en version *stand alone*, cette dernière étant retenue dans le cadre du présent ouvrage en raison de sa simplicité d'utilisation.

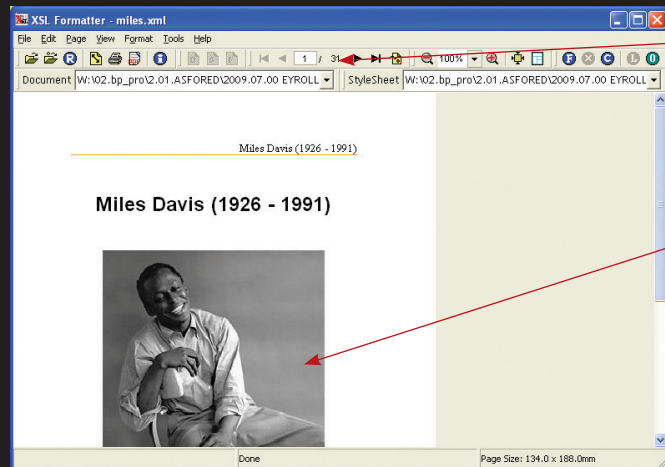
XSL Formatter masque à l'utilisateur la production du fichier XSL-FO intermédiaire qui existe bel et bien et qui est effectivement écrit sur le disque : il suffit de choisir le fichier XML d'entrée et le fichier XSLT de transformation pour obtenir le résultat sous forme d'une visualisation PDF. Il est toujours possible d'accéder au XSL-FO produit par la transformation à des fins de contrôle en cours de réglage. Mais une fois la transformation au point, il n'est pas utile d'y accéder et on règle le logiciel afin que le fichier soit effacé en fin de processus.

● L'interface de lancement de la transformation dans XSL Formatter : choix des fichiers XML et XSLT et résultat de la première page en arrière-plan



XML source

XSLT



Feuilleter. Ici 31 pages ont été générées en moins de 2 secondes.

Première page de l'ouvrage

Miles Davis (1926 - 1991)



Miles Dewey Davis III (26 mai 1926, Alton, Illinois - 28 septembre 1991, Santa Monica, Californie) est un compositeur et trompettiste de jazz américain.

WIKIPEDIA

ainé qui avait recherché l'intégration à la culture dominée par la population blanche, le parc Miles Davis s'accompagna d'une prise de position en faveur de la cause noire et de sa lutte contre le racisme avec la culture permanente d'un homme au caractère ombrageux. En 1985, il participe à l'album *L'Apartheid* à l'initiative de Steven van Zandt.

En France, c'est l'enregistrement de la *re Ascenseur pour l'échafaud* (1957) de Louis Malle qui le rend célèbre. Son dernier album, *Poo-bop*, paru à l'été 1992, laisse éclater des influences rap.

Biographie

1926-1944: l'apprentissage

Le 26 mai 1926, Cleota Henry donne naissance à Miles Dewey Davis III, à Alton (Illinois), sur les bords du Mississippi. L'enfant grandit dans un milieu familial relativement modeste. Miles Dewey Davis II est chirurgien général. Sa mère joue du piano et du violon. Son père maternel était professeur d'orgue dans une église, Dorothy, et son frère cadet, Vernon, était également musicien.

L'année suivante, la famille déménage à Saint Louis, Illinois, où son père a ouvert un cabinet d'avocats. Lorsque le jeune Miles fréquente l'école primaire, il habite un quartier à prédominance blanche, c'est la première fois qu'il expérimente la ségrégation raciale. Il se passionne pour le sport — baseball, football américain, basket-ball, natation et surtout boxe — mais aussi pour la musique — il suit avec passion l'émission radiophonique *Swing Time*.

4

WIKIPEDIA

qui lui est accordé, il saute dans le premier avion. Arrivé quelques heures plus tard, il attend pendant trois heures dans le hall de l'hôtel puis est finalement invité à monter sur scène pour le jazzman pour la séance. Miles lui dit : « Tu as 36 clichés et pas un de plus ». La séance commence puis à la 4^{ème} ou 5^{ème} photo, il demande à Miles de se mettre de profil avec sa trompette à l'horizontale. Miles Davis lui répond : « Sors d'ici, je ne joue pas la trompette à l'horizontale, je ne joue pas sur les champs de courses ». Le photographe repartait pour Paris dans l'heure.

Miles Davis avait l'habitude de jouer les yeux fermés et en tournant le dos au public. C'était selon lui pour mieux se concentrer en faisant abstraction de son environnement mais de nombreux spectateurs lui faisaient le reproche de ce qu'ils considéraient comme de l'impolitesse. Il répondait qu'un chef d'orchestre ne tourne jamais le dos à ses musiciens.

Discographie

Album studios

Miles Davis (1948) : *Cool Bop* ;
1949 : *Birth of the Cool*, sorti en 1959 ;
1951 : *Blue Period* ;
1951 : *Dig* ;

Bandes originales de films

1959 : Miles Davis *The Sound of Miles Davis*, TV Show, CBS (avec John Coltrane et Miles Davis) paru dans le DVD *The Great Jazz Film Ever*, Idem, 2003 et sous le nom *The Cool Jazz Sound*, Mvd Visual, 2005 ;

30

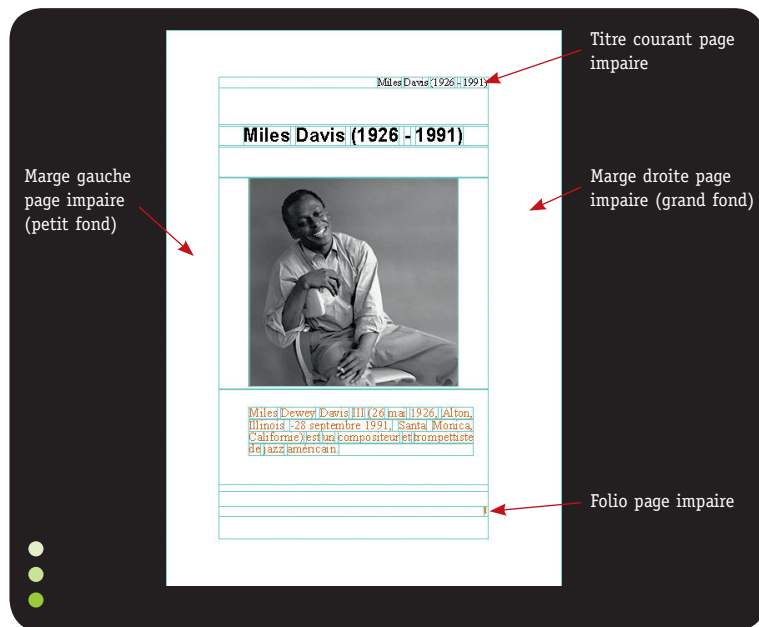
● Les différentes pages créées par l'interpréteur XSL-FO vers PDF de XSL Formatter

Deux points sont à souligner :

- le XSL-FO produit est indépendant de XSL Formatter et plus généralement de l'outil de mise en pages utilisé : il s'agit du résultat d'une simple transformation que l'on peut réaliser à l'extérieur de XSL Formatter, à l'aide de n'importe quel processeur XSLT comme MSXML ou Saxon ;
- le résultat visuel de l'interprétation du fichier XSL-FO dépend en revanche de l'outil utilisé : le même fichier XSL-FO traité par FOP (outil *open source*) donnera un rendu différent du traitement par XSL Formatter. Il convient donc de bien choisir son outil de mise en pages XSL-FO puisqu'il conditionne en partie l'écriture des programmes de transformation. XSL Formatter, en particulier, dispose de fonctionnalités supplémentaires (des pseudo-balises XSL-FO) qui ne seront interprétées que par ce logiciel : l'affichage d'un séparateur de colonne s'exprimera ainsi par différents attributs spécifiques préfixés `axf:`, compris de ce seul logiciel, le choix de l'outil est donc très structurant pour l'entreprise :

```
axf:column-rule-style="solid" axf:column-rule-width=".2pt" axf:column-rule-color="black" column-count="2" column-gap="5mm"
```

Par ailleurs, XSL Formatter dispose d'outils complémentaires de mise au point, comme la visualisation de toutes les zones PDF produites (i.e. des rectangles concernant les portions d'information élémentaire), ce qui permet de régler finement le logiciel de transformation de XML vers XSL-FO.



● Visualisation à l'aide de XSL Formatter des zones PDF produites après interprétation du XSL-FO. On retrouve la géométrie de la page décrite dans les pages-maitres ainsi que les principales structures. Le logiciel a toutefois calculé la meilleure position des informations en fonction des instructions typographiques XSL-FO, minimum, optimum et maximum.

Utiliser InDesign avec XML

InDesign prend en compte XML et deviendra sans doute le moteur de composition XML préféré des programmeurs. Nous décrivons ici les fonctions XML élémentaires d'InDesign (en vigueur depuis la CS2) qui augmentent la productivité si la maquette est optimisée pour les contraintes XML.

La correspondance élément/styles

La prise en compte de XML dans InDesign, dans sa forme la plus simple, consiste à transformer le flux XML d'entrée en un flux (très) simplifié permettant d'effectuer la correspondance éléments XML/Styles InDesign.

En pratique, on reprend la même démarche que sous Word lorsque l'on a bâti le tableau de correspondance styles Word/structure XML. La seule différence, ici, est que la structure XML à prendre en compte en entrée d'InDesign est très « plate » : elle se résume à une succession de structures paragraphes, produites dans l'ordre où ces paragraphes doivent apparaître dans la page ; le nom des éléments XML est généralement choisi pour être identique à celui des styles de paragraphes, mais ce n'est pas une obligation.

CORRESPONDANCE NOM DE BALISE/DE STYLE

Il n'est pas indispensable que les balises aient le même nom que les styles puisqu'InDesign dispose d'un outil de mise en correspondance : toute balise peut être associée manuellement au niveau d'une maquette avec un style. En pratique, il est beaucoup plus simple d'assurer la correspondance systématique, ce qui permet de lancer l'association automatique par la commande InDesign : panneau Structure > Faire correspondre les balises aux styles.

Au sein des paragraphes apparaîtront des structures XML inline dont les noms sont également choisis conformément aux styles de caractères (styles inline) du gabarit InDesign avec lequel on travaille.

La problématique, ensuite, se pose dans l'écriture d'un programme XSLT permettant de transformer le flux XML d'origine en flux XML d'entrée InDesign (on parle souvent dans ce cas du « XML AID »).

Le principe consiste à produire un flot de paragraphes texte et images, dans l'ordre où ils doivent apparaître, de telle sorte :

- qu'à un paragraphe soit associé un style de paragraphe préalablement chargé dans la maquette;
- qu'à une balise inline soit associé un style de caractère préalablement chargé dans la maquette.

Afin de ne pas se perdre dans les noms de balises (puisqu'il faut en créer de nouvelles pour le XML AID) et les noms de styles correspondants, on choisit des dénominations qui rappellent la position de l'information dans la structure arborescente du XML source. La correspondance XML source/XML AID se présentera par exemple ainsi dans le cas d'article_v1.2 :

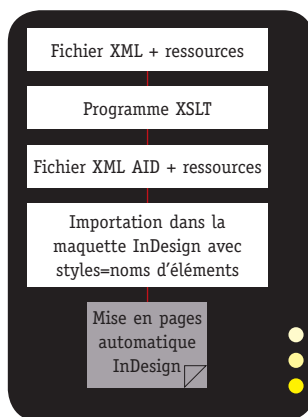
<front><titles><title>	<front-title-p>
<front><titles><subtitle><p>	<front-subtitle-p>
<body><section><title>	<section-level1-title-p>
<body><section><section><title>	<section-level2-title-p>

Ce flot est ensuite importé dans des blocs convenablement chaînés, généralement décrits dans les gabarits.

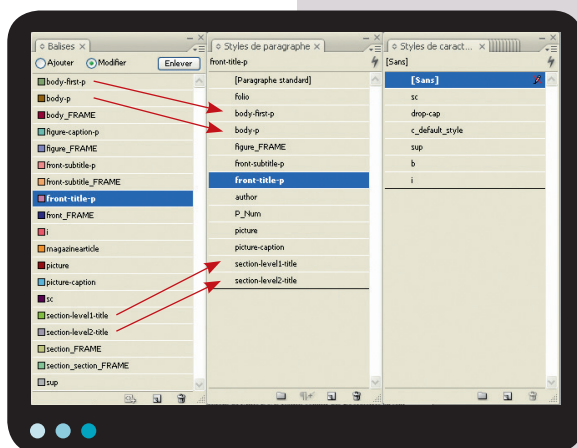
Le processus est donc extrêmement simple, puisque l'on crée un texte ne comportant que des paragraphes et des informations inline injectés dans InDesign.

Le plus difficile, et le plus déroutant par rapport à l'orthodoxie XML qui s'efforce d'imbriquer les éléments pour restituer la nature arborescente de l'information, est bien de penser le flot de texte comme une succession de paragraphes sans emboîtements ni styles inline.

- Dans InDesign, la correspondance balise/styles de paragraphe et styles de caractères. On notera que toutes les balises effectivement utilisées dans le fichier d'entrée importé sont présentes, mais seules sont significantes celles qui ont un contenu textuel ou image.



- On importe le fichier XML dans InDesign après l'avoir transformé via XSLT, afin d'avoir une correspondance stricte élément XML AID/style InDesign.



Méthodologie

Comme pour XSL-FO, la méthode consiste à partir de la maquette cible, elle-même basée sur des styles de paragraphe et des styles de caractères, à spécifier les noms de styles et de balises attendues.

La méthodologie de mise au point variera suivant que l'on travaille avec un maquettiste qui connaît les rudiments de XML ou non.

Dans l'hypothèse la plus courante où le maquettiste ne connaît pas XML, on procède en travaillant à deux (maquettiste et programmeur XML) selon le déroulé suivant :

- création de la maquette ;
- définition des noms de style en veillant à être en cohérence avec la structure XML source et à respecter quelques règles de nommage, à l'image de celles définies pour le stylage Word (voir le chapitre 5) : suffixe -p pour les styles de paragraphe, -c pour les styles de caractère, et _FRAME pour les englobants. On regroupera utilement les noms de styles et de balises XML attendues dans un tableau de correspondance (voir la figure p. 209) ;
- écriture du convertisseur XSLT permettant de passer du XML source au XML AID ;
- test d'importation et mise au point itérative.

● Powerpoint de définition des styles. L'utilisation exhaustive de styles évite de nombreuses retouches manuelles (ex : on aurait pu forcer la lettrine à la main).

The screenshot shows a PowerPoint slide about Miles Davis. Red arrows point from XML style names to specific parts of the slide:

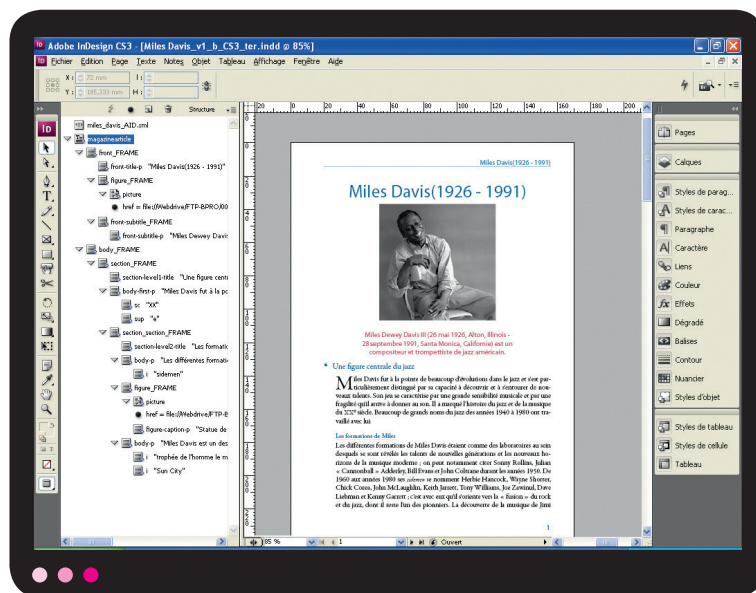
- `front-subtitle-p` points to the title "Miles Davis (1926 - 1991)".
- `front-title-p` points to the subtitle "Miles Dewey Davis III (26 mai 1926, Alton, Illinois - 28 septembre 1991, Santa Monica, Californie) est un compositeur et trompettiste de jazz américain."
- `picture` points to the photo of Miles Davis.
- `body-first-p` points to the first paragraph starting with "Miles Davis fut à la pointe de beaucoup d'évolutions dans le jazz et..."
- `body-p` points to the second paragraph starting with "Les formations".
- `section-level1-title-p` points to the section header "Une figure centrale du jazz".
- `section-level2-title-p` points to the section header "Les formations".

Une fois le système de conversion au point, c'est-à-dire fournissant de 60 à 90%, selon la complexité du projet, de la maquette cible montée, on passe en production :

- réglage éditorial du fichier XML source ;
- importation via XSLT du XML source dans InDesign pour finalisation manuelle.

styles InDesign	structure XML source	structure XML InDesign
front-suptitle-p	<front>	<front_FRAME>
front-title-p	<titles><suptitle>	<front-suptitle-p>
front-subtitle-p	<titles><title>	<front-title-p>
front-authors-p	<titles><subtitle>	<front-subtitle-p>
	<authors>	<front-authors-p>
	<body>	<body_FRAME>
body-first-p	<body><p>[position="1"]	<body-first-p>
body-p	<body><p>[position!="1"]	<body-p>
author	<author>	<author>
author-affiliation-c	<affiliation>	<author-affiliation-c>
author-firstname-c	<authorfirstname>	<author-firstname-c>
author-lastname-c	<authorlastname>	<author-lastname-c>
zbib-p	<pbib>	<pbib-p>
bibentry-article-c	<articletitle>	<bibentry-article-c>
bibentry-booktitle-c	<booktitle>	<bibentry-booktitle-c>
bibentry-journaltitle-c	<journaltitle>	<bibentry-journaltitle-c>
	<deflist><drow>	<deflist_FRAME>
deflist-defterm-p	<defterm>	<deflist-defterm-p>
deflist-defdata-p	<defdata>	<deflist-defdata-p>
pdisco_p	<pdisco>	<pdisco_p>
discentry-album-c	<album>	<discentry-album-c>
discentry-performer-c	<performer>	<discentry-performer-c>
div_FRAME	<div>	<div_FRAME>
figure_FRAME	<figure>	<figure_FRAME>
		<picture href="">
figure-caption-p	<figurecaption>	<figure-caption-p>
link-c	<link>	<link-c>
listitem-1-p	<list><item>	<listitem-1-p>
listitem-2-p	<list><item><item>	<listitem-2-p>
	<section>	<section_FRAME>

● Extrait du tableau de correspondance (feuille Excel) des styles InDesign et de la structure XML AID



● Vue de l'article Miles Davis importé automatiquement dans InDesign. On notera la structure XML AID à gauche (i.e. le XML après aplatissement par un programme XSLT).

Quelques astuces

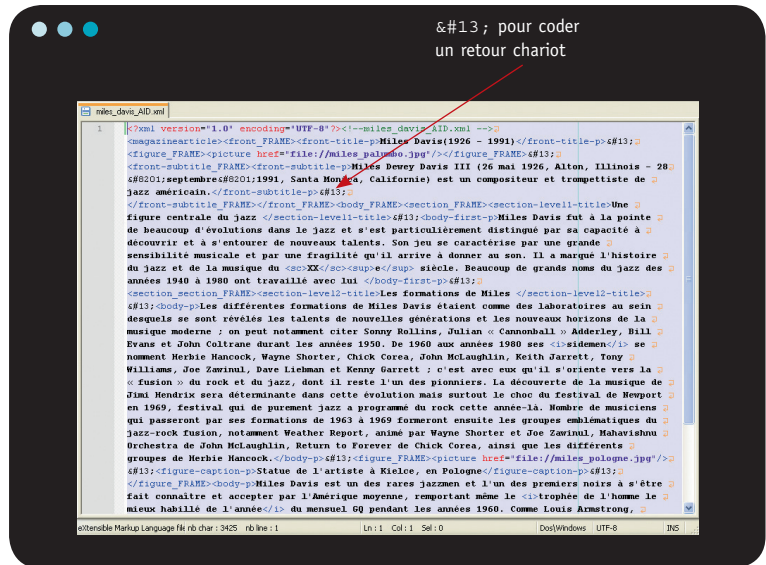
InDesign présente quelques particularités dans son interprétation du XML : le plus perturbant est sans doute le fait que le logiciel prend en compte les retours chariots à l'intérieur du flux XML. Normalement, les outils de formatage ne tiennent compte ni des retours chariot, ni des blancs interbalises. Mais InDesign en a besoin, notamment pour terminer un style de paragraphe.

CR ET LF

Ce que l'on appelle «retour chariot» est en fait l'insertion d'un code informatique interprété par les systèmes de visualisation comme un retour au début de ligne (retour chariot proprement dit, caractère `` ; ou CR pour «carriage return») suivi d'un saut de ligne (line feed, LF, caractère `
` ; ) sur les machines Windows. Mac se contente d'un retour de ligne seul et Unix d'un retour chariot seul. Quant à InDesign XML, la succession des deux caractères `` ; et `
` ; s'interprète comme deux retours chariots consécutifs !

● Une des contraintes XML d'InDesign : le flux XML ne doit contenir aucun retour chariot, tabulation, etc. Il est monoligne, mais les fins d'éléments stylés doivent être séparées par un caractère de type retour chariot, en l'occurrence `` ; afin que la mise en pages soit correcte.

Il va donc falloir insérer en fin de style un retour chariot simple, ce qui peut se faire facilement en XML en insérant l'Unicode hexadécimal, soit `` ; dans le flux, là où l'on mettrait un retour chariot au clavier. Dans ce cas, on ne mettra aucun retour chariot «en dur» dans le flux XML, ce qui le rend plus systématique et élimine le risque d'un formatage clavier involontaire.



La seconde particularité est liée à la manière d'appeler les images. InDesign impose en effet le vocabulaire spécifique suivant :

```
<picture href="file://images/miles.png"/>
```

pour appeler le fichier miles.png dans le sous-répertoire relatif images/. Il faut donc en tenir compte dans les programmes de transformation XSLT.

La correspondance attributs InDesign/styles

Force est de reconnaître que l'approche XML « aplati » consistant à faire correspondre le nom d'une balise au nom d'un style est très contraignante et même perturbante lorsque l'on souhaite récupérer en sortie d'InDesign le fichier XML selon sa DTD d'origine, ce qui peut être le cas lorsque des corrections d'auteur ou de dernière minute ont été apportées sur la mise en pages. Procéder à une rétroconversion du XML à plat vers le XML riche d'origine est une opération complexe, sinon impossible, en fonction de la politique de nommage choisie pour les différents éléments du XML InDesign.

Une alternative nettement plus séduisante, et d'un esprit un peu plus proche d'XML, est proposée par Adobe sous la forme d'attributs spécifiques d'InDesign à poser dans la structure XML d'origine. Techniquement, il s'agit toujours de faire correspondre un élément du fichier XML d'origine à un style InDesign. La différence majeure vient de ce que l'on ne modifie pas, ou très peu, le XML d'origine : on se contente de poser sur l'élément à styler (en style de paragraphe ou en style de caractère) un des deux attributs suivants :

- `aid:pstyle` pour un style de paragraphe ;
- `aid:cstyle` pour un style de caractère.

Le programme de transformation est très simple à écrire : il suffit d'identifier les éléments englobants d'un flux de texte ou d'image que l'on souhaite transformer en paragraphe ou en caractère stylé et de poser l'un des attributs précédents en précisant le style à attribuer.

Dans le cas d'`article_v1.2`, le XML AID du titre de l'article prendra la forme suivante :

```
<front><title id=title="front_title"
aid:pstyle="front-title-p"> Miles Davis (1926
- 1991)</title>
```

Rien n'a été changé au flux XML d'origine, si ce n'est l'ajout de `aid:pstyle="front-title-p"` qui signifie pour InDesign : styler selon le style de paragraphe `front-title-p` tout ce qui se trouve à l'intérieur de la balise `<title>` (du fait que l'attribut est posé sur cette balise).

Méthodologie

En termes de méthodologie, rien ne change par rapport à l'approche décrite précédemment : le choix des noms de styles est important pour comprendre facilement quel élément est balisé. La redondance liée au suffixe du style (-p pour les styles de paragraphes et -c pour les styles de caractères) et au nom de l'attribut (aid:pstyle ou aid:cstyle) est une sécurité permettant de très rapidement contrôler un flux XML.

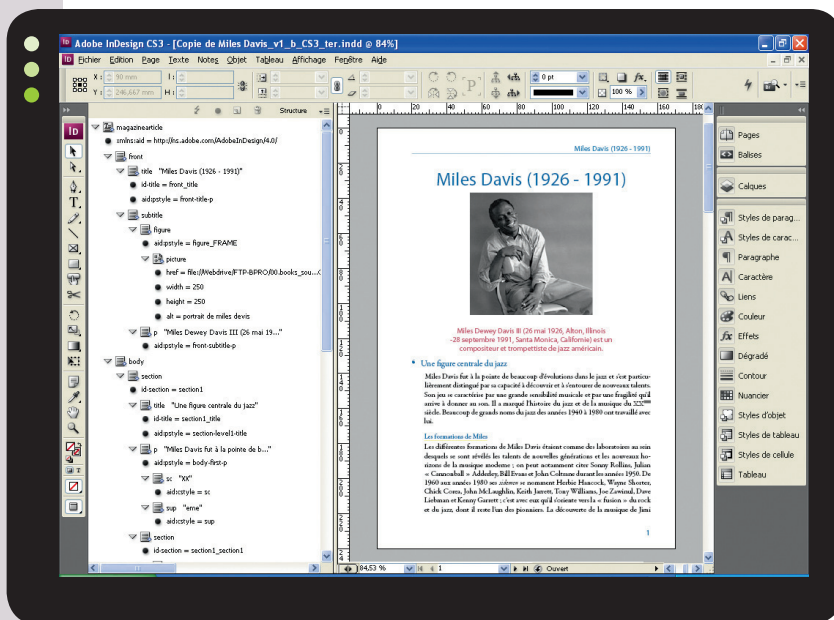
Les contraintes citées précédemment s'appliquent :

- l'élément doté d'un style doit être suivi d'un retour chariot sous la forme `` ; et le flux XML ne doit avoir ni tabulation ni retour chariot ou *line feed* « en dur » ;
- les images doivent être appelées selon la syntaxe `<picture href="file://"/>`.

Enfin, les attributs utilisés par InDesign étant préfixés par `aid:` (pour Adobe InDesign), il convient de déclarer un espace de nom sur l'élément englobant (`<magazinearticle>` dans l'exemple) selon la démarche habituelle des déclarations d'espace de noms :

```
<magazinearticle xmlns:aid="http://ns.adobe.com/AdobeInDesign/4.0/">
```

● Le résultat est le même qu'avec la méthode éléments/style. La structure XML est quasi identique à celle du XML source, la différence ne portant que sur les attributs `aid:` et les appels d'images.



Bien entendu, la pose des attributs `aid:` et la modification des appels d'image s'effectuent via un programme de transformation XSLT que l'on intègre à la maquette de telle manière que le chargement du XML soit complètement transparent pour l'utilisateur. En fonction de la complexité de la maquette, la mise en pages est automatisée de 60 à 100 % et, surtout, le XML est récupérable par exportation, ce qui permet d'être sûr d'avoir un XML représentatif de la publication papier, celle-ci constituant généralement la référence éditoriale.

IDML

Adobe, à partir de la CS4, a mis en place un format d'archivage remplaçant le langage INX (*InDesign Interchange*) existant auparavant. Baptisé IDML, pour *InDesign Markup Language*, il s'agit d'un jeu de fichiers XML, relativement complexes, descriptif de la publication. Tout comme de nombreux formats modernes (Open office, Word, epub), les fichiers sont emballés dans un zip unique suffixé `.idml`. Pour en examiner le contenu, il suffit de renommer le fichier en `.zip` et de l'ouvrir avec n'importe quel zippeur comme 7-Zip. La description de IDML sort du cadre du présent ouvrage.

En guise de conclusion

XML est désormais au cœur de la technique du livre : la profession dispose d'un format qui allie robustesse à souplesse tout en étant relativement simple à manipuler. Et surtout, il préserve l'avenir : le texte XML devient un véritable actif tactique permettant de répondre rapidement aux sollicitations du marché. Qu'un nouveau support de publication apparaisse, et il sera possible, en très peu de temps, de proposer son fond éditorial au moindre coût.

À rebours, XML peut devenir un tonneau des Danaïdes pour l'éditeur séduit par les possibilités offertes par la technologie : la tentation est forte de basculer vers la production audiovisuelle et de rajouter de la vidéo, de l'animation à ce qui est aujourd'hui un livre, c'est-à-dire un discours écrit, relativement linéaire, représentatif d'une pensée.

Mais on ne peut échapper à la pente technologique : autant connaître les menaces apportées par l'innovation et préserver ses activités et convictions en prenant le train de la publication sur support électronique ; sans négliger le port d'un gilet de sauvetage, XML, pour flotter sur les vagues de la modernité.

