The eXtensible Markup Language (XML)

Stéphane Genaud

21 mars 2007

1 Historique

Dans les années 1980, de nombreux industriels de l'informatique essayèrent de promouvoir un format de données indépendant de toute plateforme [Fretter98]. Microsoft lança par exemple le RTF, et Adobe le PostScript, parent du PDF. Parallèlement, un groupe travailla à la normalisation d'un standard nommé *Standard Generalized Markup technology* (SGML), normalisé par l'ISO en 1986 (ISO 8879 :1986).

Durant les années 1990, SGML n'a pas rencontré le succès escompté : le monde de l'internet découvre à cette époque le langage HTML et sa facilité de mise en œuvre, en passant à côté de SGML. Conscients des faiblesses de HTML, le W3C à mis en place en 1996 un groupe pour définir une nouvelle version de SGML, baptisée *eXtensible Markup Language* (XML). La spécification XML à été développée de 1996, à 1998, par un partenariat d'industriels et d'organismes universitaires. La première spécification, XML 1.0 à été publiée en Février 1998 sous la forme d'une recommandation du W3C. Parmi les industriels, on compte Sun, HP, Microsoft, Netscape, Adobe, Fuji, Xerox, auxquels il faut ajouter des entreprises spécialisées dans le SGML, comme Arbor Text, Inso, SoftQuad, Grif, Isogen, Texcel. Dans la communauté universitaire, les principaux participants furent la Text Encoding Initiative (TEI), le NCSA, et James Clark. Plus récemment (après la définition de XML 1.0), IBM, Oracle et Omnimark se sont rajoutés au groupe. Aujourd'hui, la dernière version est 1.1[1].

2 Les besoins

2.1 Besoin d'un format universel

XML s'impose progressivement comme une technologie permettant de répondre aux problèmes liés aux *formats de données* informatique. Un nombre calculable d'heures de travail a été passé par les informaticiens, dans toutes les entreprises de par le monde, pour convertir des données écrites dans un format vers un autre format. Ce besoin peut provenir par exemple d'un changement de logiciel, et l'on souhaite ré-utiliser les anciens fichiers avec

[1] http ://www.w3.org/TR/2004/REC-xml11-20040204/

un logiciel nouveau qui n'accepte pas ce format.

Avec la multiplication des applications informatiques, le nombre de formats a aussi largement augmenté, et il devient impossible d'écrire des convertisseurs de données universels tant le nombre de combinaisons est élévé.

Pour illustrer les problèmes qui se posent, prenons l'exemple du format le plus utilisé de nos jours : HTML. Ce format est ouvert et normalisé par le W3C, ce qui assure sa pérennité et garantit son accessibilité (à contrario de beaucoup de formats propriétaires). Néanmoins, tranformer des documents HTML en un autre format nécessite des efforts importants.

2.2 L'exemple de HTML

Aujourd'hui des millions de pages sont écrites en HTML. Ce langage permet la *mise en forme* (ou *présentation*) du texte, ou *présentation* à l'aide de *balises*, à l'instar des traitements de textes précurseurs dans ce domaine, TEXet LATEX[Lamport85].

Ainsi, pour présenter l'adresse d'une personne sur une page web, on peut écrire le texte HTML suivant :

```
<i>Mr Charles Dupont</i><br>
13 Avenue des lilas<br>
44000 Nantes<br>
```

où `<i>` est une balise permettantde mettre en italique, et `<br>` indique qu'on doit passer à la ligne.

Cependant, les balises proposées ne donnent pas d'information sémantique. Pire même, le contenu du texte est entremêlé des balises indiquant quelle forme doit prendre le contenu, si bien qu'un programme est nécessaire pour extraire le contenu d'une page HTML.

2.3 Limitations de HTML

Un problème parmi beaucoup d'autres réside maintenant dans la classification de l'information, en particulier par les moteurs de recherche d'Internet. Comment distinguer dans quel contexte est utilisé le mot `lilas` présent dans le fragment précédent ? Un utilisateur qui chercherait, avec un moteur de recherche, le mot *Lila* en pensant à la fleur, pourrait se voir retourner l'adresse de cette page qui ne présente qu'une adresse postale. De manière plus générale, les échanges de données sous cette forme n'ont aucun intérêt car ils sont difficilement traités par des programmes. Il est donc nécessaire de structurer les documents selon la sémantique des mots contenus dans ce document.

2.4 Utiliser XML à la place

Avant de dire plus précisemment ce qu'est XML, donnons un exemple de document à balises qui satisgerait nos besoins : nous voulons mettre en évidence le contenu sémantique de l'exemple précédent. Pour cela, on enrichit le texte présenté ci-dessus par des balises (inventées pour le besoin) qui font référence à la nature de l'information, et non sa présentation (c.f. figure 1) :

```
<?xml version="1.0" ?>
<!DOCTYPE carnet-adresse>
<carnet auteur="Jean-Paul">
  <personne>
     <etatcivil>
        <titre>Mr</titre>
        <prenom>Charles</prenom>
        <nom>Dupont</nom>
     </etatcivil>
     <adresse>
        <rue>13 Avenue des lilas</rue>
        <ville>Nantes</ville>
        <cp>44000</cp>
     </adresse>
  </personne>
</carnet>
```

Fig. 1 – Exemple de texte XML pour un carnet d'adresse

On vient ici d'écrire un document XML. Car XML n'est pas un seul format de données, mais **un ensemble de règles** permettant de spécifier un format de données.

Dans notre exemple précédent, il se trouve que nous respectons les règles de XML que nous expliquerons dans la section suivante. Nous avons juste inventé un nouveau " langage " XML dont le but modeste est de représenter des carnets d'adresses. Notre document est un document XML bien formé et à ce titre il va pouvoir être exploité par tous les outils développés pour XML, et en particulier des convertisseurs.

3 Caractéristiques remarquables

Avant de préciser ce que recouvre la technologie et son utilisation, il faut remarquer deux caractéristiques générales des documents XML : les données sont en clair, et la présentation des informations doit être spécifiée à part. Ces caractéristiques sont le résultat de choix technologiques modernes.

3.1 Un document XML est lisible

La plupart des formats de données ont une forme dite “binaire” qui ne peut être lue facilement que par des programmes et qui est généralement plus compacte. (A titre d’exemple, essayez de lire le contenu d’un document produit par Microsoft Word sans ce même logiciel).

Les documents XML sont lisibles par des êtres humains : on arrive à comprendre la structure du carnet d’adresse et à en extraire les données. Lisible ne veut pas dire “clair et facilement lisible” : le nombre important de balises rend le texte évidemment très touffu et donc rébarbatif à lire. Il faut bien garder à l’esprit que des outils logiciels vont être normalement utilisés pour manipuler ces documents. Par exemple, on peut utiliser un éditeur de document XML comme Conglomerate[2] pour éditer le carnet d’adresse, comme le montre la figure 2.

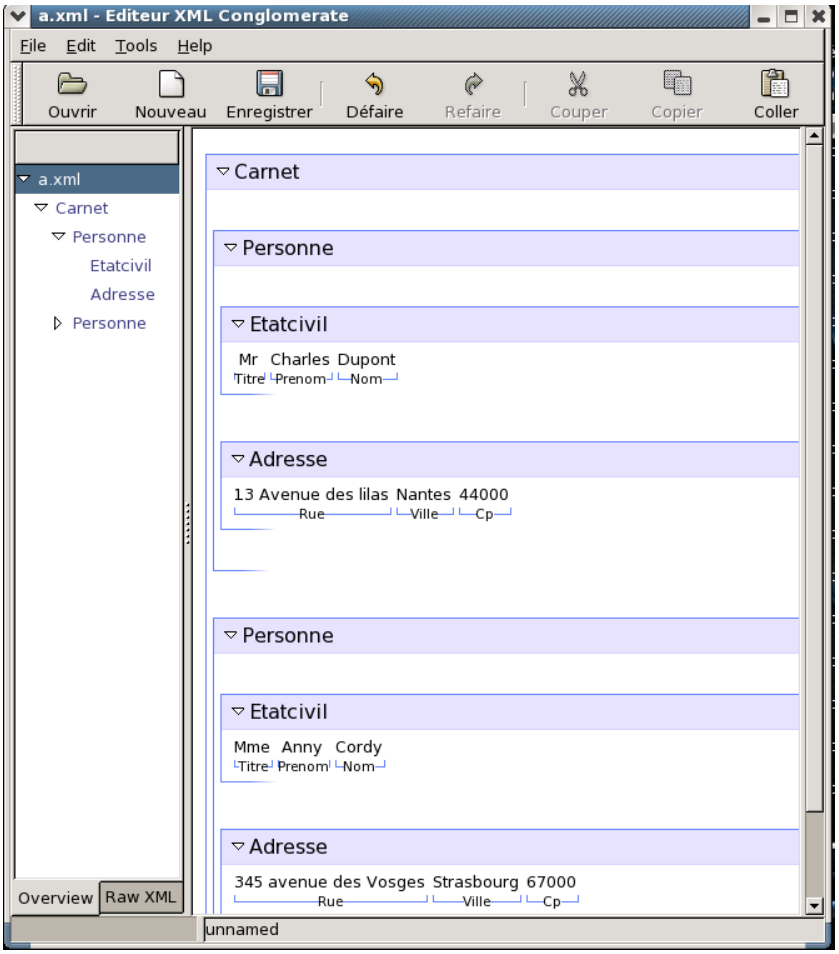

Fig. 2 – Notre carnet en XML, édité avec l’éditeur Conglomerate

[2]http ://www.conglomerate.org

Au moment de la conception de XML, les avantages de cette forme lisible l'ont emportés : un programmeur peut facilement modifier le fichier en cas de problème, et les techniques de compression actuelles permettent de transmettre des fichiers de cette forme dans un volume aussi compact que des fichiers binaires.

3.2 Présentation des données

Un fichier XML représentant des données, rien ne précise de quelle manière les données peuvent être visualisées (si besoin est). Les instructions de mise en page pour visualisation, dans un navigateur par exemple, sont volontairement écartées du document. La mise en page pour visualisation est assurée par une *feuille de style* (ou *stylesheet*).

Ce principe est maintenant largement adopté : on le retrouve en HTML avec les feuilles de style CSS, ou en Word (fichiers `.dot`). Pour XML, nous verrons que le concept de feuille de style est plus large, car il permet la transformation complète du document. En résumé, la mise en page n'est pas liée à l'information sémantique.

4 Les documents XML

Essayons maintenant de définir ce qu'est un document XML. Nous avons vu au travers de l'exemple du carnet d'adresse page 3, qu'il était possible de créer son propre " dialecte ". Pour qu'on puisse considérer ce document comme du XML, il faut respecter certaines règles de bases très simples qui portent sur l'imbrication des balises. On dit alors que le document est *bien formé*. Cependant, l'intérêt d'un document bien formé est limité si on ne dit pas aussi quelles balises sont autorisées dans notre dialecte. Si ces règles sont aussi respectées, on dit alors que le document est *valide*.

4.1 Document XML bien formé

Le *prologue* du document, indiquant quelles conventions respecte le document, est obligatoire. Ces conventions disent quelle version de XML est utilisée, ainsi que le le jeu de caractères (en anglais encoding) est utilisé dans le document (attribut facultatif, ici on spécifie qu'il s'agit du jeu ISO8859-1 (latin), pour permettre de prendre en compte les accents français). Un prologue typique est une ligne du type :

```
<?xml version="1.0" encoding="ISO8859-1"?>
```

Ensuite vient le document lui-même : c'est une imbrication de balises et de *données*. Une donnée est tout ce qui se trouve entre deux balises, une balise ouvrante `<balise>` et une balise fermante `</balise>`. Une donnée peut à son tour être une balise ou autre chose.

L'imbrication des éléments suit une forme particulière : elle est arborescente. En effet, XML impose deux règles qui font que tout document peut être représenté sous la forme d'un arbre.

– Il existe un seul attribut de plus haut niveau.
– Il est interdit de faire se chevaucher des balises, c'est-à-dire d'avoir une succession de balise du type :

```
<balise1>
<balise2>
</balise1>
</balise2>
```

– Toute balise ouvrante est fermante. Une exception est faite pour des balises qu'il est idiot de déclarer à la fois ouvrante et fermante (par exemple le retour à la ligne `<br>` en HTML). On utilise alors une balise à la fois ouvrante et fermante notée `<balise/>`.

Si on observe le carnet d'adresse figure 1, on constate bien que 1) le prologue est présent, et 2) les éléments peuvent être représentés sous la forme d'un arbre.
En supposant qu'on ait trois personnes dans le carnet, on pourrait dessiner l'arbre (dont la racine est en haut) de la figure 3 (certaines branches n'ont pas été développées pour des questions de place).

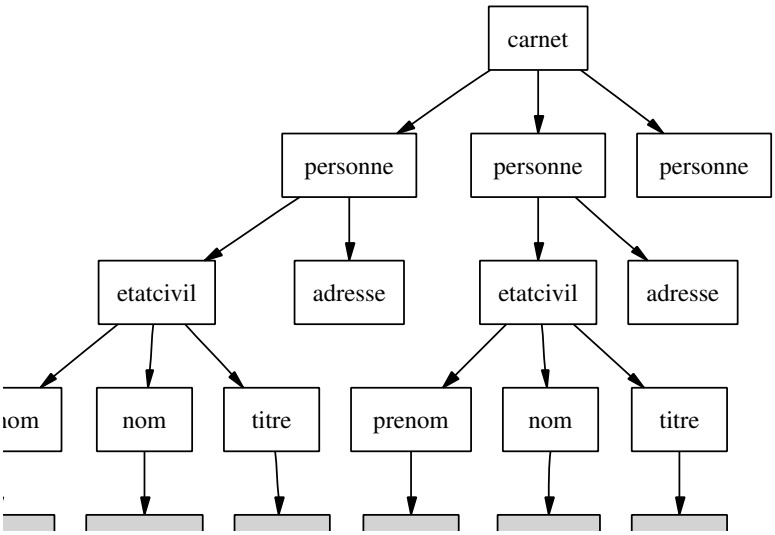

FIG. 3 – Les données du carnet sont structurées de manière arborescente.

4.2 Document XML et DTD

Etant donné ces règles, nous n'avons encore presque rien dit sur les contraintes à respecter pour qu'un document ait du sens. Une question tout aussi importante est de savoir, quelles balises avec quels attribut je peux utiliser, et comment ces balises peuvent s'agencer

les unes par rapport aux autres. Le seul exemple du carnet ne vous permet pas de savoir par exemple si vous pouvez spécifier deux adresses pour une personne.

Pour dire tout cela, les concepteurs d'un langage écrivent une *Document Type Definition* (DTD). La DTD est la *grammaire* du langage : elle définit les mots du langage (balises et attributs) ainsi que les règles de constructions syntaxiques possibles dans le langage. Définir la DTD que j'appellerai carnet-xml, c'est définir tout ce qu'il est possible de construire comme carnet d'adresses valides. On appelle ce travail la *modélisation de document*. Pour l'exemple de la figure 1, une définition pourrait être la suivante :

```
1  <!DOCTYPE carnet-adresse [
2          <!ELEMENT carnet (auteur, personne*)>
3          <!ATTLIST carnet auteur CDATA #REQUIRED>
4          <!ELEMENT personne (etatcivil,adresse)>
5          <!ELEMENT etatcivil (titre,prenom,nom)>
6          <!ELEMENT adresse (rue,ville,cp)>
7          <!ELEMENT titre (#PCDATA)>
8          <!ELEMENT prenom (#PCDATA)>
9          <!ELEMENT nom (#PCDATA)>
10         <!ELEMENT rue (#PCDATA)>
11         <!ELEMENT ville (#PCDATA)>
12         <!ELEMENT cp (#PCDATA)>
13      ]>
```

Examinons quelques détails de cette DTD : la ligne 2 déclare comment le document doit commencer, ou la *racine* du document. On déclare en effet un *élément* de nom `carnet` utilisé par aucun autre élément, et qui est constitué d'un auteur, et d'un ensemble de personnes (éventuellement aucune). La ligne 3 précise que l'auteur doit être impérativement spécifié. La ligne 4 définit ce qui constitue une personne : parmi ses constistuants, on trouve l'état civil et l'adresse, eux-mêmes constitués d'autres éléments. Cette DTD nous permet de voir qu'une seule adresse est possible pour une personne.

4.3 Document XML valide

En résumé, on distingue deux niveaux de validité d'un document XML :
- un document XML est dit *bien formé* si il respecte les règles précédentes (prologue et imbrication des balises)
- un document XML est *valide* si il est bien formé et si toutes ses balises appartiennent à une DTD.

4.4 Y a t-il une DTD pour moi ?

Quand arrive le besoin d'écrire un document, vous allez donc naturellement vous poser la question : " *Existe t-il déjà une DTD qui me permette d'exprimer mes informations en XML ?* ". La réponse est " probablement ". Il existe à l'heure actuelle des centaines de DTD dans tous les domaines. Besoin de stocker un graphique ? Il existe SVG (Scalable Vector Graphics) dont un exemple est présenté figure 4.

```
<?xml version="1.0" ?>
<!DOCTYPE svg PUBLIC "-//W3C/DTD SVG 200001102/EN"
       http://www.w3.org/TR/2000/CR-SVG-200001102/DTD/svg-200001102.dtd>
<svg>
  <desc>Un rectangle et un cercle</desc>
  <rect   fill="green" x="1cm" y="1cm" width="3cm" height="3cm"/>
  <circle fill="red"   cx="3cm" cy="2cm" r="4cm"/>
</svg>
```

FIG. 4 – Graphique décrit avec SVG

Il y a MathML pour les mathématiques, CML pour la chimie, cXML ou CBL pour le commerce (voir section 7), DocBook pour des livres (la figure 5 permet de voir quelques éléments de la DTD), etc.

La deuxième question légitime, après avoir trouvé une DTD adéquate pour le domaine visé est " *Cette DTD est elle pérenne ?* ". La réponse est difficile à donner pour la simple raison que les DTD ne sont **pas normalisées**. En fait, seul XML est fait l'objet d'une recommandation par le W3C[3]. La pérennité d'une DTD dépend donc son utilisation. Les précédentes DTD citées sont tellement répandues qu'elles sont des standards *de facto*. Beaucoup de personnes ont travaillé sur ces grammaires pour en corriger les défauts et elles couvrent en général presque toutes les besoins du domaine.

4.5 Les schémas XML

Les schémas XML (*XML Schema* en anglais) jouent le même rôle que les DTD. Pour simplifier l'exposé, nous n'avons parlé que des DTD, historiquement mieux installées. Les schémas XML décrivent aussi la grammaire d'un langage mais exprime cette grammaire en XML. Il a été souvent reproché aux DTD leur syntaxe particulière, et jugée trop compliquée. Effectivement, l'adoption d'un langage XML pour les grammaires semble légitime et depuis que le W3C a publié sa première recommandation sur XML Schema en 2001, son usage est en progression constante.

[3]http ://www.w3.org/XML/Core/#Publications

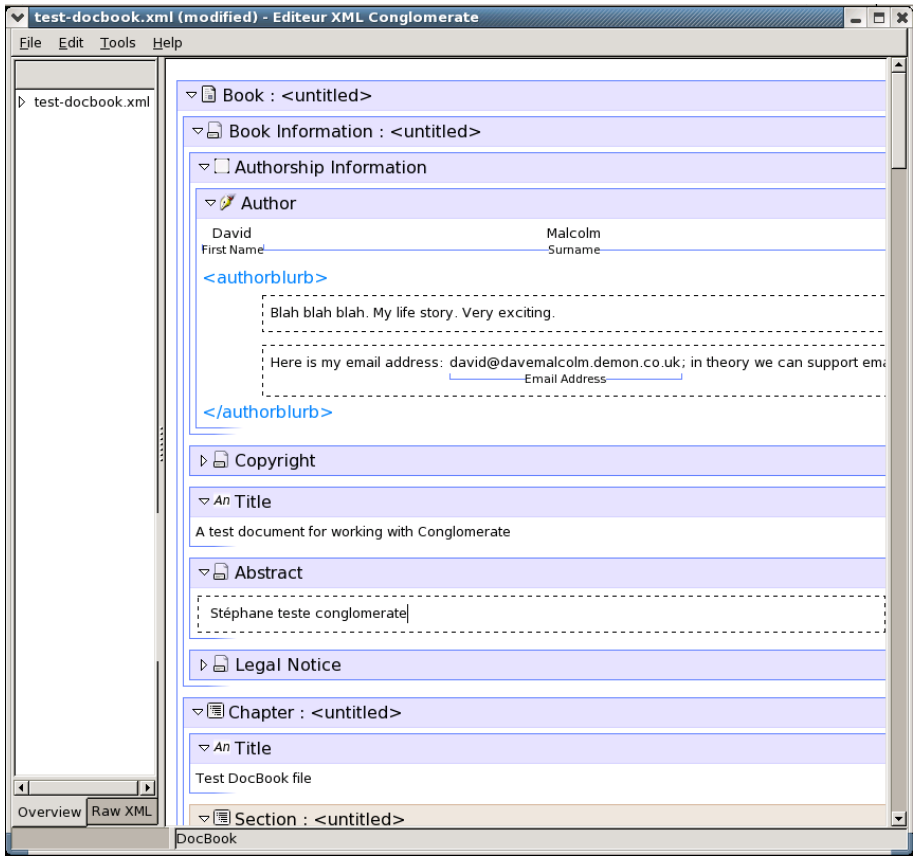

FIG. 5 – Ecriture d'un document DocBook avec Conglomerate

5 Mise en page de XML

Nous avions mentionné précédemment, que la mise en page d'un document XML requiert des informations supplémentaires qui ne figurent pas dans le document XML.

Ces informations de formatage sont apportées par ce qu'on pourrait appeler génériquement des *feuilles de style*.

5.1 Les feuilles de style : XSL

XSL *(eXtensible Stylesheet Language)* provient directement du *Document Style Semantics and Specification Language* (DSSL) utilisé pour SGML. Il définit des balises de formatage dans le contexte plus large d'un langage de transformations –les efforts de standardisation de XSL ont pour l'instant beaucoup porté sur la normalisation de ces transformations.

L'expressivité de XSL est bien plus grande que CSS : contrairement aux CSS, XSL permet aussi de retraiter un document XML afin d'en modifier totalement sa structure, ce

qui permet à partir d'un document XML d'être capable de générer d'autres types de documents (PostScript, HTML, Tex, RTF, ...) ou bien un fichier XML de structure différente.

On voit immédiatement tout l'intérêt que peuvent représenter ces transformations. Imaginons simplement une page d'information hébergée sur un serveur web, accessible à travers des équipements divers : un navigateur sur un ordinateur personnel, un PDA, et un téléphone portable WAP. Le serveur pourrait formater les informations dans le langage de présentation adapté à l'équipement qui se connecte, en l'occurrence du HTML pour le navigateur, du WML pour le téléphone portable, ou (par exemple) du format d'affichage Palm pilot. Pour réaliser ces transformations, XSL possède deux composantes :

5.2 Le moteur de transformation : XSLT

XSLT (eXtensible Stylesheet Transformation) est un langage permettant de décrire des règles de transformation, pour passer d'un document XML à un autre document XML.

Nous avons vu qu'un document XML peut être représenté comme une structure arborescente. Par exemple, dans l'exemple du carnet d'adresse, un fichier XML contenant deux personnes peut être représenté comme l'arbre de la figure 3.

Ainsi XSLT permet de transformer les documents XML à l'aide de feuilles de style contenant des règles appelées template rules (ou règles de gabarit en français).

Le processeur XSLT (composant logiciel chargé de la transformation) analyse le document XML et crée la structure arborescente correspondante. Les transformations selon les template rules contenues dans la feuille XSL produisent un arbre résultat représentant, par exemple, la structure d'un document HTML.

Chaque template rule définit des traitements à effectuer sur un élément (noeud ou feuille) de l'arbre source. L'arbre source peut ainsi être entièrement remodelé et filtré ainsi qu'on peut ajouter du contenu à l'arbre résultat, si bien que l'arbre résultat peut être radicalement différent de l'arbre source. C'est ce qui offre les facilités de transformations vers d'autres langages évoquées précédemment. Notons que le texte produit par la transformation a aussi la forme d'un arbre : on peut donc dire que XSLT permet de transformer du XML vers XML.

5.3 Mise en œuvre

A titre d'exemple, voici un premier exemple simplissime qui transforme les balises `<para>` et `<emphasis>` du document XML en des balises HTML. Les transformations à exécuter sont décrites dans cette feuille de style XSL :

```
1 <?xml version='1.0'?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3                 version="1.0">
```

```
4
5  <xsl:template match="para">
6    <p><xsl:apply-templates/></p>
7  </xsl:template>
8
9  <xsl:template match="emphasis">
10   <i><xsl:apply-templates/></i>
11 </xsl:template>
12
13 </xsl:stylesheet>
```

Les règles de transformations sont exprimées dans les deux blocs de lignes 5 à 7, et 9 à 11. Il faut retenir qu'une règle s'applique toujours dans le contexte de l'un des nœuds du document source. La première règle se déclenche à la rencontre d'un nœud `para` : le processeur XSLT produit alors le fragment de texte spécifié à l'intérieur de la règle. En l'occurrence, ceci consiste à 1) produire le texte `<p>` 2) y ajouter le texte qui pourrait être produit par une application de règle (récursivité) 3) ajouter le texte `</p>`.
Avec cette feuille de style, le document XML suivant :

```
<?xml version='1.0'?>
<para>Ceci est un <emphasis>test</emphasis>.</para>
```

sera transformé en :

```
<?xml version="1.0" encoding="utf-8"?>
<p>Ceci est <i>test</i>.</p>
```

qui peut être lu par un navigateur n'interprétant que le HTML.

En effet, le processeur XSLT est descendu dans le nœud `para`, a produit le texte `<p>` puis a essayé d'appliquer une autre règle. N'ayant pas trouvé de règle pour le texte "Ceci est un", il le recopie. Il se trouve qu'il peut ensuite appliquer la règle sélectionnant le nœud `emphasis`, et il produit donc le texte `<i>`. Dans ce contexte (nœud `para/emphasis`) aucune autre règle ne peut s'appliquer, et la suite du texte est produite.

Prenons un autre exemple, un peu plus complexe, permettant de mettre en forme le carnet d'adresse. Ici, nous donnons une feuille de style permettant de faire la liste des personnes présentes dans un carnet.

```
1  <xsl:stylesheet version = '1.0'
2        xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
3
4  <xsl:template match="carnet">
```

```
<html>
  <head>
  <title>Carnet de : <xsl:value-of select="@auteur"/></title>
  </head>
  <body>
    <ol>
       <xsl:for-each select="personne">
       <li>
            <xsl:value-of select="./etatcivil"/>
       </li>
       </xsl:for-each>
    </ol>
  </body>
</html>
</xsl:template>

<!-- traitement personne -->
<xsl:template match="etatcivil"> <xsl:value-of select="titre"/>
<xsl:value-of select="nom"/> <xsl:value-of select="prenom"/>
</xsl:template>

</xsl:stylesheet>
```

Pour approfondir, un exemple plus conséquent, donné par [Harold01], est consultable en ligne. Pour une compréhension très complète du sujet le livre [AR02] est un excellent ouvrage.

6 Les logiciels pour XML

Nous donnons dans cette section, quelques pointeurs sur les logiciels utilisés dans le monde XML. Cette liste est en aucun cas exhaustive ; au contraire, l'effervescence actuelle autour de XML peut faire penser que le nombre de logiciels développés va croître très vite. Une source d'information est la page maintenue par le groupe Oasis [Cover00].

6.1 Les lecteurs

Côté serveur Pour que XML soit adopté par le monde internet, il est indispensable que tous les utilisateurs puissent lire des documents XML, quelque soit le navaigateur employé. Les prochaines générations de navigateurs seront capable de lire du XML, mais en attendant, il faut envisager une solution technique transformant du XML en HTML. Cette transformation doit être faite sur le serveur Web.

A titre d'exemple, voici la solution proposée par IBM : quand un navigateur demande un document XML au serveur, cette demande est filtrée par une couche appelée XML Enabler, qui ajoute à la requête, la demande d'une feuille style appropriée pour le navigateur. Le serveur retourne le document XML et une feuille XSL. Le XML enabler s'adresse alors à un outil de traduction qui, à partir de la feuille de style XSL et du document XML, rend un document HTML.

Côté client Beaucoup de projets de développement concernent la réalisation de navigateurs lisant des fichiers XML exprimés dans certaines DTD. Quelques un des projets phares [4] sont Amaya (www.w3.org/Amaya), Internet Explorer (www.microsoft.com/windows/ie/) et Mozilla (www.mozilla.org)[5]. On peut prendre l'exemple de MathML, que ces navigateurs devraient être capable de rendre, c'est-à-dire lire le XML et produire à l'écran des formules telles que nous les typographions habituellement. Vous pouvez pointer votre navigateur sur http ://www.w3.org/Math/testsuite pour vérifier.

6.2 Les bibliothèques logicielles

Pour analyser un document XML, deux API sont offertes :
- le *Document Object Model* (DOM), standardisé par le W3C, qui propose des fonctions pour manipuler les éléments d'un document.
- la *Simple API for XML* (SAX), qui déclenche des évenements lors du parsing du document. Si vous ne donnez auncune réponse à l'évenement, l'élément anlaysé est alors oublié. Cette façon de procéder permet, notamment sur de gros documents, d'économiser beaucoup de mémoire si seuls qquelques éléments vous intéressent.

7 XML et commerce électronique

L'utilisation de l'EDI est complexe à mettre en place (de très nombreux formats sont définis par ANSI X12), et de fait, réservée à des entreprises de grande taille. La première initiative de proposition d'échanges de données avec XML est provenue d'OASIS et de l'UN/CEFACT, sous le nom d'ebXML. Elle vise à élargir le nombre des entreprises pouvant utiliser cette technologie d'échange.

Bien que XML ne soit pas un remède miraculeux (XML en soi ne résout pas les problèmes d'interopérabilité) il a certains avantages :
- chacun peut écrire son propre langage de balises,
- il existe des outils pour passer d'un langage à l'autre,
- un humain est capable de comprendre intuitivement le contenu d'un fichier XML

[4] A la date de rédaction, les dernières versions sont Amaya 3.4, Mozilla 2.0 et IE 7

[5] Les Netscape 6.x et 7.X sont des dérivés de ce dernier logiciel

De fait, plusieurs organisations se sont penchées sur un langage (plus précisemment une DTD) qui serait appropriée pour le commerce électronique.

On peut citer **CBL** (Common Business Library), de Commerce One (www.xcbl.org), dont l'initiative est soutenue par Microsoft. Cette librarie est constituée de différentes DTD, comme par exemple des DTD pour les factures et paiements, la gestion des commandes ou échanger des enchères. Les DTD sont disponibles à http ://www.xcbl.org/xcbl30/DTD/doclist.html

Une autre proposition importante provient de Ariba (www.ariba.com) sous le nom de **cXML**. Développé depuis 1998, cXML est soutenu par de nombreuses entreprises, dont Philips, Nestlé, Visa, etc . Le guide utilisateur est disponible à l'adresse donnée dans la référence [Ariba02].

Références

[Lamport85] Leslie Lamport, *LaTeX - A Document Preparation System User's Guide and Reference Manual*, Addison-Wesley, Reading, MA, USA, ISBN 0-201-15790-X, 1985.

[Fretter98] Todd Fretter, *XML : Mastering Information on the Web*, Sun Microsystems, `http ://www.sun.com/980310/xml`, March 1998.

[Cover00] Robin Cover, *THe SGML/XML Web page, Extensible Markup Language (XML)*, `http ://www.oasis-open.org/cover/xml.html`, January 2000.

[AR02] Bernd Amann, Philippe Rigaux, *Comprendre XSLT*, Editions O'Reilly, 2002.

[Harold01] Elliotte Rusty Harold, *Chapter 17 of the XML Bible, Second Edition : XSL Transformations* , `http ://www.ibiblio.org/xml/books/bible2/chapters/ch17.html`, November 2001.

[Ariba02] Ariba Corporate, *cXML User Guide*, `http ://xml.cxml.org/current/cXMLUsersGuide.pdf`, Novembre 2002.