XML – INTRODUCTION TECHNIQUE

(EN VUE DE LA MODELISATION DES ECRITURES PROCEDURALES)

2ème SEMINAIRE D'INFORMATIQUE JURIDIQUE DE MACOLIN

13 NOVEMBRE 2001

LUCIEN LAZZAROTTO

Pour en savoir plus :

« *XML A Primer* » (Simon St. Laurent, M&T Books, 2001) ;

« *The XML and SGML Cookbook* » (Rick Jelliffe, Prentice-Hall, 1998) ;

« *Structuring XML Documents* » (David Megginson's, Prentice-Hall, 1998) ;

« *Developing SGML DTDs : From Text to Model to Markup* » (Eve Maler/ Jeanne El Andaloussi, Prentice-Hall, 1996) ;

« *$GML : The Billion Dollar Secret* » (Chet Ensign's, Prentice-Hall, 1997) ;

« *XML How to program* » (H. M. Deitel, P.J. Deitel, T. R. Nieto, T.M. Lin, P. Sadhu, Prentice-Hall, 2001).

* * *

Sites utiles :

http://www.w3.org (notamment http://www.w3.org/TR et

http://www.w3.org/XML)

http://legalxml.org et http://lexml.de

http://wsba.org/c/ec2/xml/1999/eficourt.htm

http://www.oasis-open.org/cover/sgml-xml.html (articles divers)

http://www.utm.edu/departments/math/graph/ (théorie des graphes ; RDF)

http://metadata.net/dstc/ (éditeur de meta-données ;cf RDF)

http://legalxml.com/DocumentRepository/ProposedStandards/Clear/PS_10001/PS_10001_2000_07_24.htm (Legal XML Proposed standard No PS_10001_2000_07_24)

* * *

I. LES ORIGINES DE XML ET LA NOTION DE DOCUMENTS STRUCTURES

I.1 Documents structurés vs documents non structurés

Au sens « informatique » du terme, un document « non structuré » n'est pas un document dans lequel la pensée de l'auteur apparaît confuse ou un document dont la présentation en chapitres et sous-chapitres a été négligée. C'est un document dont la structure « de surface » (chapitres, sous-chapitres, titres, références, notes, tableaux, etc…) ou la structure « logique » interne (enchaînement des idées, thèse/antithèse, démonstration, processus, etc…) n'a pas été décrite, référencée ou codée sur le plan informatique. Un document non structuré est donc essentiellement un document que seul un lecteur humain est en mesure de « comprendre » et de parcourir de manière « intelligente ». Un document non structuré ne correspond, pour l'ordinateur, qu'à un enchaînement de caractères alphanumériques, sans hiérarchie ou liens particuliers; il ne permet pratiquement aucun traitement automatique.

A l'inverse, un document « structuré » permet, en raison de la technologie utilisée lors de sa création, de repérer et répertorier de manière automatique les différentes parties et éléments qui le composent, de manipuler lesdites parties, d'imposer un contenu minimum lors de sa création, de présenter l'information qu'il contient de différentes manières, que ce soit sous l'angle typographique ou logique, d'utiliser des outils de recherche qui tireront parti de sa structure sous-jacente, de le mettre à jour de manière aisée, etc…

Illustration : un tableau de données, décrivant par exemple les résultats financiers d'une société par produits et zones géographiques, peut être réalisé au moyen d'un traitement de texte, l'auteur créant l'effet de lignes et de colonnes grâce à de simples espaces blancs, mais il peut également être créé à partir d'une feuille de calcul ou généré par un gestionnaire de base de données. Un éventuel destinataire du premier fichier, au format « traitement de textes », sera dans l'impossibilité d'obtenir de manière automatique une représentation graphique des données reçues ou d'extraire au moyen d'une instruction simple une partie des données en cause, par exemple les résultats du produit X dans le pays Y, aux fins de création d'un second rapport. Le destinataire de la feuille de calcul ou du fichier de base de données disposera, en revanche, de toutes ces possibilités et de bien d'autres encore. Cela n'est pas surprenant, puisque les feuilles de calculs et les gestionnaires de bases de données sont spécialement conçus pour traiter des données fortement structurées, comme des informations financières et comptables; il semble d'ailleurs, à première vue, contre-productif d'utiliser un traitement de textes pour stocker de telles informations.

Les feuilles de calculs et les gestionnaires de bases de données comportent toutefois deux inconvénients majeurs, lorsqu'il s'agit de produire et utiliser des informations textuelles classiques, dont l'auteur aimerait souligner la structure:

a) ils se prêtent mal à l'édition et au stockage de champs de grandes dimensions aux formats non connus à l'avance, leurs cibles étant plutôt

les données par nature fortement structurées et dont les éléments présentent des dimensions prédéterminées;

b) ils produisent des fichiers ayant un format « propriétaire » (dit « fichiers binaires »), c'est-à-dire des fichiers qui ne peuvent être lus, que ce soit par l'ordinateur ou l'humain, sans disposer du programme qui les a créés (sauf à utiliser des « filtres » et des « traducteurs », lorsqu'ils existent sur le marché), ce qui limite leur diffusion et leur échange entre utilisateurs de plates-formes matérielles et logicielles différentes.

Cela étant, le besoin d'outils permettant de structurer des documents textuels aux fins de modification, de recherche et d'indexation est très aigu dans bon nombre de domaines (songeons par exemple au monde de l'édition technique, qui produit les volumineux manuels de maintenance des machines et engins complexes de nos industries). La mise à jour et l'utilisation de certains documents seraient en effet un casse-tête et une opération très gourmande en ressources humaines si ces documents étaient produits par de simples traitements de textes standards. De plus, la possibilité de diffuser des documents en s'affranchissant des barrières logicielles et matérielles qui divisent les utilisateurs a de tout temps été un objectif d'une partie de l'industrie informatique et des académies. Enfin, l'explosion du WEB a montré l'importance et l'urgence de structurer les documents diffusés sur « la toile » pour permettre aux moteurs de recherche d'isoler de manière pertinente l'information que ces documents contiennent et qui sont souvent noyées au sein des milliards de pages disponibles.

XML, en tant que standard de structuration de l'information au sein de fichiers de type « texte », c'est-à-dire lisible par tout ordinateur et directement compréhensible par l'être humain, permet de résoudre, respectivement d'atteindre, les difficultés et objectifs susmentionnés, tout en faisant se rejoindre le monde des « données » et celui des « textes », puisqu'il offre un format unique pour traiter ces deux genres d'information.

I.2 Standard Generalized Markup Language - SGML

Les concepts et techniques sous-tendant XML ne sont pas récents. Dès la fin des années soixante, les problèmes et défis susévoqués ont conduit des chercheurs de la firme IBM à développer un « langage » de description de document fondé, d'une part, sur un système de balises (appelés « tags » en anglais), encadrant l'information traitée et s'emboîtant en fonction de la logique interne du document et, d'autre part, sur l'existence, pour chaque document, d'un fichier annexe ou d'une en-tête décrivant les balises utilisées et définissant la structure devant être respectée par ledit document. Ce fichier annexe ou élément de fichier est dit « Document Type Definition » ou DTD.

La puissance de ce concept résidait dans la liberté laissée à l'utilisateur de définir ses propres balises pour rendre compte des spécificités de son document, sans être restreint par un corpus de mots clés prédéfinis. Grâce aux DTD, il devenait également possible de contrôler le contenu d'un document au moment de sa

création, en imposant certaines règles de rédaction, permettant ainsi de gérer des masses d'information de manière standardisée.

Ce langage, qui n'a été normalisé sous forme d'un standard ISO qu'en 1986 (ISO 8879 :1986) porte le nom de Standard Generalized Markup Language ou SGML. L'industrie de l'édition en a tiré grand profit, mais l'implantation de ce standard dans les produits de bureautique diffusés dans le grand public s'est heurtée à la lourdeur et à la complexité dudit standard.

I.3 eXtensible Markup Language XML

SGML était une idée forte, mais les outils intégrant ce standard sont onéreux et lourds tant à utiliser qu'à développer. Quant aux évolutions de plus en plus riches et complexes du langage HTML (lui aussi inspiré de SGML, mais dans une optique essentiellement limitée à la présentation des documents au sein des navigateurs WEB et à l'utilisation des hyper liens) elles demeuraient inaptes à satisfaire le besoin croissant de structuration des documents au sein du World Wide Web.

La nécessité de disposer d'une technique de structuration offrant une grande partie de la puissance de SGML, mais moyennant une complexité réduite, devenait donc évidente. C'est ainsi que naquit le standard « eXtensible Markup Language » ou XML, normalisé en 1998 par le W3Consortium (XML 1.0), qui est souvent décrit (à tort, car la formule est réductrice) comme le « SGML du WEB ». De manière imagée, on affirme que XML c'est 80% de la puissance de SGML pour 20% de sa complexité.

A l'instar de SGML, dont il est en quelque sorte un sous-ensemble, puisque l'un des impératifs de ce nouveau standard est d'être compatible avec SGML, XML repose sur un système de balises, décrivant des « éléments » et leurs attributs. Comme SGML, XML fait appel à la notion de DTD, qui permet à la fois de décrire la structure d'un document et de forcer l'utilisateur à la respecter. Toutefois, à l'inverse de SGML, le noyau des concepts et codes de XML est suffisamment réduit pour rendre son apprentissage et son utilisation très aisés.

Le terme « eXtensible » renvoie au fait que les balises utilisées pour structurer un document peuvent être définies et créées par l'utilisateur; elles ne sont donc pas figées dans le langage, ce qui rend ce dernier « extensible » à souhait et adaptable aux besoins du domaine à traiter.

II. XML : ASPECTS PRINCIPAUX DE SON FONCTIONNEMENT

Un document XML comporte en premier lieu une déclaration, encadrée par les signes **<? … ?>,** indiquant qu'il est rédigé en XML et précisant qu'il se conforme à la version xxx du langage (en l'état, seule la version 1.0 existe). Cette déclaration contient éventuellement d'autres informations, comme le système de codage retenu pour les caractères (par ex. : ASCII ou UNICODE), exemple :

<?xml version = «1.0» encoding= « UTF-8 » ?>

Le corps du document est ensuite composé de texte et d'un certain nombre d'autres balises ou « tags ».

II.1 Les balises

II.1.1 Formalisme

Une balise ou un tag est un mot clé commençant par le sigle « < » et finissant par le sigle « > » (ceci est une balise : <balise>).

II.1.2 Les éléments (ou briques de base)

Un document XML est essentiellement composé « d'éléments » qui s'organisent de manière structurée.

Un élément est une zone balisée, qui débute par un tag d'ouverture, contenant le nom de l'élément (ceci est une balise d'ouverture : <Element1>), et se termine par un tag de fermeture, comportant le nom de l'élément précédé du signe « / » (ceci est une balise de fermeture : </Element1>).

Tout document contient au moins un élément, mais il en comprendra en général plusieurs, comportant des sous-éléments. Les éléments ne peuvent en revanche se chevaucher :

Ceci est du XML correct :

<Chapitre1> blabla <Section1> blabla </Section1> blabla </Chapitre1>

Ceci n'est pas du XML correct :

<Chapitre1> blabla <Section1> blabla </Chapitre1> blabla </Section1>

Pour des questions de lisibilité, les éléments d'un document XML sont plutôt présentés ainsi :

```
<Chapitre1>  blabla
        <Section1> blabla </Section1>
        <Section2> blabla </Section2>
</Chapitre1>
```

L'exemple des chapitres d'un livre est le plus simple qui soit, mais le concept d'éléments peut être utilisé pour modéliser de nombreuses situations et objets, tel des tableaux de données, des formules, des processus, etc…

I.1.3 Le texte

Entre chaque balise se trouve le texte du document. Bien entendu, lorsqu'une application conçue pour traiter des fichiers XML affiche ou imprime un document XML, seul le texte apparaît, les balises demeurant invisibles pour le lecteur.

Dans la mesure où les applications XML partent du principe que chaque caractère rencontré est potentiellement une balise, les caractères utilisés pour créer des tags, tel les signes (<), (>), (**'**') et (**'**) sont interdits dans le corps du texte. Ils doivent être remplacés par des codes, en l'occurrence encadrés par les signes **&** et **;**.

Ces codes sont, soit tirés de la table UNICODE (technique qui permet d'ailleurs d'intégrer au texte des caractères d'alphabets différents du nôtre), soit ceux prédéfinis dans le langage XML pour les principaux cas de figure.

Ainsi :

& = & **< = <** **> = >** **' = '** **'' = "**

et les termes: « **ceci est un texte comportant les signes >'& suivis d'un point.** » se transcrivent ainsi :

<texte>ceci est un texte comportant les signes >'& suivis d'un point.</texte>

Bien entendu, les produits et environnements de développement permettant de créer des documents XML offriront à l'utilisateur le moyen de rendre ces opérations « transparentes » et lui éviteront le tracas d'éliminer les caractères interdits, en intégrant automatiquement leur équivalent codé.

Cela étant, il est des cas où l'auteur veut précisément utiliser des caractères interdits sans qu'ils soient confondus avec des balises. Dans ce genre de situation, il est possible d'ouvrir une section dite CDATA (pour « mélange caractères et données »), au sein de laquelle le texte ne sera pas contrôlé par l'application cible. Ces sections commencent par les signes **<![CDATA[** et se termine par les signes **]]>,** c'est-à-dire **<![CDATA[….]]>.**

I.1.4 Les attributs

Les éléments peuvent être précisés par des attributs, qui prennent leur place dans la balise d'ouverture de l'élément. Le nom de l'attribut est suivi du signe = et de la valeur de l'attribut encadré par des guillemets.

Ex. :

<Document Auteur= «Jean Dupont » Sujet=«Botanique»>
Ceci est un article de Jean Dupont sur la botanique
</Document>

Il n'est pas toujours aisé de déterminer s'il est préférable d'utiliser la notion de sous-élément ou d'attribut pour préciser un élément. XML offre les deux possibilités et le choix final de l'auteur dépendra en substance de la nature de l'application.

I.1.5 Les entités

Pour éviter d'avoir à saisir à de multiples reprises une portion de texte répétitive ou pour pouvoir modifier un terme répétitif postérieurement à la création du document (par exemple le nom des parties dans un modèle de contrat), XML offre le concept d'« entité ».

Une entité se déclare ainsi :

<!ENTITY *NomEntite* «*caractères_de_remplacement* »>

ou

<!ENTITY *NomEntite* SYSTEM «*NomDuFichierContenantUneEntité* »>

La deuxième formulation indique que l'entité est décrite dans un fichier se trouvant sur le même ordinateur.

Lors de la rédaction du corps du document XML, il suffira d'utiliser le code ***&NomEntite*** pour obtenir, au moment de la lecture du fichier, l'apparition des ***caractères_de_remplacement*** en lieu et place dudit code.

I.1.6 Les Namespaces

L'auteur d'un document XML étant libre de créer ses propres noms d'éléments et d'attributs, il est possible que certains noms choisis soient identiques à ceux utilisés par un autre auteur dans un but différent.

Lors d'échanges de documents, des incompréhensions ou conflits terminologiques peuvent se produire.

Afin d'éviter ce risque, XML offre une solution (purement optionnelle), qui consiste à donner à une zone du document (ou à l'ensemble du document) un « nom d'espace » (ou Namespace), au sein de laquelle, tous les éléments et attributs sont considérés comme précédés d'un préfixe correspondant à ce nom d'espace.

Un nom d'espace se déclare dans la balise d'ouverture de l'élément au sein duquel il doit s'appliquer (le cas échéant dans la balise d'ouverture de l'élément « racine » s'il doit s'appliquer à tous les éléments du document) :

<*NomElement* xmlns:prefix=«*IdentifiantDuNamespace*»>

où **xmlns** est un mot réservé pour annoncer un nom d'espace et où ***IdenfiantDuNamespace*** sera choisi de manière à être unique (de fait, il est fréquent d'utiliser un nom de domaine Internet ou un URL associé à l'auteur, car ce sont des noms uniques).

II.2 Les DTD et les SCHEMAS

II.2.1 DTD

Une DTD est une séquence de codes, placée dans un fichier séparé ou en-tête d'un document XML, définissant la structure dudit document. Elle précise le type, l'ordre, le nombre et les attributs des éléments du document.

Une section DTD se présente ainsi :

<!DOCTYPE *NomDocument* [

…

]>

Au sein de cette section, les éléments et leurs attributs sont décrits par des balises commençant par les signes « **<!** ».

Lorsque la DTD est « externalisée » sous forme de fichier séparé, la section DTD du document XML se limite à une balise contenant le nom du fichier externe

(sous forme d'URL où de nom de fichier précédé du mot clé SYSTEM lorsque le document se trouve sur le même ordinateur). Une DTD externe sera un document contenant uniquement une section DTD, identique à celle qui aurait été intégrée au document XML.

Un document XML peut faire référence à plusieurs DTD, considérées comme complémentaires.

Un document XML se présentera donc soit ainsi:

<?xml version = «1.0»?>

<!DOCTYPE *NomDocument* [

…

]>

<ElementRacine>….

<Sous-element>…

</Sous-element>

</ElementRacine>

soit ainsi:

<?xml version = «1.0»?>

<!DOCTYPE *Memo* SYSTEM memo.dtd>

<ElementRacine>….

<Sous-element>…

</Sous-element>

</ElementRacine>

La notation utilisée pour définir si les sous-éléments doivent apparaître dans un ordre précis ou non, s'ils sont optionnels ou non, s'ils peuvent être mélangés avec du texte ou non, s'ils peuvent être multiples ou doivent rester uniques, est la suivante :

\+ = **élément pouvant apparaître de multiples fois, mais au moins une**

***** = **élément pouvant apparaître de multiples fois ou aucune**

? = **élément ne pouvant apparaître qu'une fois ou pas du tout**

(NB : l'absence de signe signifie que l'élément est obligatoire et unique)

| = **alternative entre deux éléments**

, = **séquence d'éléments devant apparaître dans l'ordre défini**

Un élément peut également être défini comme vide (*NomElement* EMPTY) ou comme ne contenant que du texte, sans sous-éléments (*NomElement* #PCDATA).

Au sein d'une DTD, les attributs d'un élément se définissent quant à eux ainsi :

<!ATTLIST *NomElement NomAttribut TypeAttribut ValeurParDefaut*>

où *TypeAttribut* peut être une liste définie par l'auteur, une constante (mot clé #FIXED), une entité (voir chap. I.1.5), des caractères libres (mot-clé CDATA) ou des caractères restreints sans espaces (mot clé NMTOKEN).

et où *ValeurParDefaut* est optionnelle.

Un attribut peut également être défini comme #REQUIRED, pour indiquer son caractère nécessaire, ou #IMPLIED, pour indiquer que sa valeur sera attribuée automatiquement par l'application qui utilisera le fichier XML en question.

Quant aux entités, leur usage a été exposé plus haut.

La DTD d'un document nommé « Memo », contenant l'élément « Message », ainsi que les sous-élements « Auteur », « Destinataire », « Date » et « Texte », composés de texte libre, et, le cas échéant, deux éléments optionnels « Sujet » et « CopieConforme », de même qu'un attribut « Statut », pouvant prendre les valeurs « confidentiel » ou «public » (« public » étant la valeur par défaut), se présenterait donc ainsi :

<!DOCTYPE *Memo* [

<!ELEMENT Message (Auteur, Destinataire, Date, Sujet?, CC*, Texte)

<!ELEMENT Auteur (#PCDATA)

<!ELEMENT Destinataire (#PCDATA)

<!ELEMENT Date (#PCDATA)

<!ELEMENT Texte (#PCDATA)

<!ATTLIST Message Statut (Confidentiel | Public) Public)

]>

En tout état, l'apparente complexité de rédaction des documents XML et des DTD doit être relativisée, dans la mesure où les environnements de développement et autres outils bureautiques intégrant le standard XML permettront à l'utilisateur de créer ses modèles de documents de manière conviviale et graphique, en quelques clics de souris.

II.2.2 XML Schema

Les XML Schemas (recommandation W3C depuis le 2 mai 2001), offrent une alternative aux DTD comportant plusieurs avantages sur ces dernières, ce qui pourrait aboutir à l'abandon total de celles-là au profit de ceux-ci. Les XML Schemas autorisent en effet une description beaucoup plus fine des éléments du document, en étendant notamment les types d'attributs utilisables et la définition de contraintes. De plus, à l'inverse des DTD, les XML Schemas sont eux-mêmes des documents XML au sens propre, c'est-à-dire que leur présentation se conforme aux règles de rédaction des documents XML. Cela permet de facto d'utiliser les applications et outils de manipulation des documents XML pour traiter les XML Schemas.

II.3 Les Parsers

Un document XML qui respecte la syntaxe XML est un document « well-formed » ou « valide à la forme ». Un document XML qui respecte la DTD ou le Schema qui le définit est un document « valid » ou « valide au fond ».

Les applications dont le rôle est de contrôler la validité des documents XML sont des « parsers ». Il existe des parsers qui se limitent à contrôler le caractère « bien formé » du document (dits « non validating parsers »), opération indépendante de l'existence d'une DTD, et des parsers qui contrôlent la validité « matérielle » du document XML, en le comparant à sa DTD ou au Schema qui le définit (dits « validating parsers »)

Il existe, du point de vue informatique, deux techniques pour programmer un parser de manière à lui permettre d'appréhender un document XML. La première s'appuie sur le modèle DOM (Document Objet Model) et consiste à créer préalablement en mémoire un « arbre » à l'image des « branches » du document, qui permettra d'effectuer les tests de conformité nécessaires. La deuxième s'appuie sur le modèle SAX (pour Simple API for XML), qui consiste pour l'application à parcourir le document de manière linéaire et à réagir, c'est-à-dire

déclencher certaines fonctions données, lors de chaque « événement » consistant en la rencontre d'une balise XML. Chacun de ces modèles a ses avantages et ses inconvénients, ainsi que ses domaines d'application, mais ce sont là des considérations transparentes pour l'utilisateur.

II.4 Les outils de manipulation et présentations

II.4.1 XPath

XPath (recommandation W3C depuis le 16 novembre 1999) est une technologie complémentaire à XML, dont le but est le repérage des différentes parties d'un document XML, aux fins notamment de permettre la navigation au sein du document. XPath offre une syntaxe pour localiser les éléments, attributs, etc…du document..

XPath n'est pas un langage structuré, comme XML, mais plutôt un système déclaratif, comme celui utilisé pour identifier les répertoires d'un disque dur (cf c:/repertoire1/sous-repertoire1/…/fichier)

II.4.2 XSLT

Grâce à l'existence des balises de structures d'un document XML et à l'existence d'une DTD ou d'un Schema le décrivant, la manière dont un document XML va apparaître sur un écran ou à l'impression peut être aisément gérée et modifiée, puisqu'il suffit d'attacher de manière globale aux différents éléments décrits des attributs graphiques et typographiques particuliers. De fait, plusieurs formats de présentations peuvent être aisément créés pour un même document.

La notion de feuille de style, déjà utilisée avec le langage HMTL a été reprise pour XML, adaptée et étendue, pour aboutir à XSL (abréviation de XML Stylesheet Language) et à son sous-ensemble XSLT (pour XML Stylesheet Language Transformation), dont la version finale avant normalisation sous forme de recommandation W3C date du 24 août 2001. XSLT permet à la fois le formatage des documents XML et leur transformation en d'autres formats, tel HTML, RTF, etc…. Il s'appuie sur XPath pour le repérage des éléments au sein du document.

II.4.3 XPointer

XPointer, pour XML Pointer Language (recommandation W3C depuis le 27 juin 2001), offre un moyen de « pointer » sur l'information contenue dans un

document XML. Associé à la technologie XLink, il permet de créer des liens hypertextes sophistiqués.

Grâce au caractère structuré des documents XML, XPointer offre un outil toutefois beaucoup plus puissant que les liens hypertextes des documents HMTL, car un lien peut être établi avec une partie d'un document, sans que l'auteur dudit document ait lui-même prévu ce lien à l'avance par l'insertion d'un récepteur (dit « ancrages »), ce qui règle du même coup le problème de maintenance de liens par l'auteur du document cible.

II.5 CONCLUSION

En tant que technologie permettant de décrire de l'information de manière finement structurée dans un fichier texte, parfaitement indépendant de toute plate-forme matérielle ou logicielle, XML est une réponse idéale aux problèmes d'échanges de document et de données. De fait, développé au départ dans le monde de l'informatique documentaire, XML a conquis le monde de l'informatique des données et servira de plus en plus de « colle informatique» ou de pont entre les applications.

* * *