LE LANGAGE XML

(Une petite introduction)

Tout comme son prédécesseur HTML, ***XML*** (*Extensible Markup Language*) fonctionne sur un système à *balises* (*Markup*) issu de *SGML*.

Mais tout l'oppose au HTML. Ainsi, il n'est pas dans les prérogatives d'*XML* de définir la mise en forme des données (gras, alignement, listes, etc), pas plus que ce langage ne propose de balises prédéfinies ayant des actions attendues (c'est d'ailleurs en ce sens qu'il est extensible).

L'objet d'XML est de séparer la partie informationnelle pure (les données, le document, etc) de tout traitement qui lui est associé, aussi bien pour la mise en forme que pour l'exploitation du contenu.

On pourrait dire que XML est aux bases de données ce que HTML est au traitement de texte.

I Constitution des documents XML

Un document XML est un ensemble d'informations structurées par des balises. Il doit répondre à une syntaxe de référence et possède une structure standardisée. Pour rendre l'affichage plus convivial ou lisible, on pourra recourir à des feuilles de style qui permettent la transformation d'un arbre XML selon une mise en forme. Syntaxe des balises XML

Bien que XML ne définisse pas les balises que l'on peut utiliser, il leur impose le respect d'un certain nombre de contraintes pour qu'une page XML soit considérée comme bien formée. Ainsi, les balises :

- sont sensibles à la casse
- doivent avoir une partie ouvrante<…> et fermante</…>. Remarque : une donnée vide peut être résumée par <…. />
- ne peuvent se chevaucher : on doit les fermer dans l'ordre inverse où on les a ouvertes
- doivent respecter les obligations de nommage d'éléments (commencer par une lettre ou _, suivi de chiffres ou des caractères – ou ., ne peut débuter par xml, sans espace)
- les commentaires s'écrivent entre <!–…–> (et donc l'utilisation de – est interdit dans un commentaire). Remarque : les commentaires ne sont pas forcément transmis du serveur au client, ils ne peuvent donc fournir des informations utiles au client.
- Elles peuvent comporter des attributs (<balise attribut='valeur' …>), les valeurs étant forcément entre apostrophes (' ') ou entre guillemets (“ ”). Remarque : ces attributs pourraient aussi être représentés par des balises. Ils apportent une certaine compacité dans le code.

Structure du document XML

Un fichier écrit en XML décrit une arborescence d'informations rattachées à une racine unique (la première balise). <balise_Racine> …corps du document </balise_Racine> Ensuite, chaque nœud (node) pourra comporter des valeurs (on parle de Parsed Character DATA ou PCDATA) ou d'autres nœuds enfants.

```
<balise_Racine>
```

```
<!--…corps du document …-->
      <balise_Nœud1>donnee_PCDATA1  </balise_Nœud1>
      <balise_Nœud2>
               <balise_sousNœud2_1>donnee_PCDATA2_1  </balise_sousNœud2_1>
               <balise_sousNœud2_2>donnee_PCDATA2_2  </balise_sousNœud2_2>
      </balise_Nœud2>
 </balise_Racine>
```

Pour des raisons de compatibilité avec les évolutions futures, il est recommandé de commencer un document XML par l'instruction

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
```

qui précise :

- la version utilisée,
- le type d'encodage du document (UTF-8 plus compact pour l'ASCII [donc sans accent] ou UTF-16 lorsque des caractères internationaux sont utilises, mais aussi ISO-8859-1 ou d'autres)
- standalone qui indique si le fichier est autonome ou dépend d'un autre document (page appelée, page incorporée, etc).

La suite du document correspond à l'enchaînement des balises (on parle d'un élément) et du texte (PCDATA) qui correspond aux valeurs des propriétés représentées par les balises.

Cette partie textuelle ne doit pas utiliser les caractères spécifiques aux balises (< >) ou particuliers (& ' " etc). On aura recours à leur équivalent codifié (respectivement &lt, &gt, &amp, &apos, &quot, etc) ou bien, lorsqu'un nombre trop important de caractères spéciaux est utilisé, on pourra utiliser la syntaxe :

```
<![CDATA[texte_qui_n_est_pas_interprété]]>
```

Mise en forme du XML avec CSS

Pour dissocier la mise en forme du contenu informatif, XML préconise le recours à des feuilles de mise en forme externes selon le principe des CSS (Cascade Style Sheet) qui vont permettre de préciser le mode d'affichage d'une information, selon l'endroit où elle se trouve dans l'arborescence.

On utilisera alors un ensemble de présentations nommées, enregistrées dans un fichier ayant l'extension .css. La feuille de style sera incluse dans la page XML avec l'instruction :

```
< ?xml-stylesheet type="text/css" href="URL_feuille_Style.css" ?>
```

Les styles sont définis dans un fichier comme suit :

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<!-- Fichier mesStyles.css -->
<!-- définit un style en gras et italique  -->
.grasItal {"font-style:italic";"font-weight:bold"}
<!-- définit un style nom qui met le texte en rouge -->
.nom {color:red}
```

```
<!-- définit un style grasItal qui s'applique à la balise nom -->
.nom [type="grasItal"] {font-style:italic ;font-weight:bolder}
<!--Redéfinit la balise p de mise en forme des paragraphes -->
p {color:orange;font-family:"Times new roman"}
```

Les styles seront ensuite appliqués aux balises dans le document XML par l'attribut class (pour les styles simples) :

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<?xml-stylesheet type="text/css" href="mesStyles.css" ?>
<Personne>
<Identité >
<titre class="grasItal">Monsieur</titre>
<nom class="nom">Hire</nom>
</Identité>
<Identité>
<titre>  Monsieur</titre>
<nom type="grasItal">Oquois</nom>
</Identité>
</Personne>
```

Un des inconvénients majeurs de l'utilisation de css est qu'il oblige à intervenir à l'intérieur du fichier de description des données. Or ceci n'est pas très conforme au principe de fonctionnement de XML.

C'est pourquoi l'on préférera avoirs recours à des techniques directement XML comme les pages de styles XSLT décrites ci-après.

II Exploitation de document XML : PARSEUR, XSLT, XPATH

Affichage standard

Pour rendre l'affichage correspondant à un fichier XML, les outils de visualisation (principalement les navigateurs) intègrent un processeur XML ou parseur, dont le rôle est d'effectuer l'analyse du code (en conformité avec les règles XML) et la conversion en un format affichable (en mieux que du HTML, avec possibilité de réduire des parties d'arbre).

Transformation de fichiers

Pour améliorer le rendu, ou destiner le contenu à un usage spécifique (tableur, PDF, HTML, etc) il est possible d'utiliser la syntaxe XSLT (Extensible Stylesheet Language for Transformation, définie en XML) qui précisera comment remplacer les nœuds ou leur contenu conformément à un modèle de sortie.

En clair, un document XML peut être transformé, grâce à XSLT, en un fichier de tableur, en une page web, en un document pour Open Office, Word ou Star Office ou même en une série de requêtes SQL. Le plus courant est tout de même le passage de XML vers du HTML ou une autre structure XML.

XSLT construit en mémoire l'arbre XML à partir des données brutes du document XML, puis applique les styles aux différents nœuds et feuilles de cet arbre. On n'intervient pas dans le fichier de données, sinon qu'en lui précisant au début à quel fichiers de transformation on l'associe par la ligne :

```
< ?xml-stylesheet type="text/xsl" href="URL_feuille_Style.xsl" ?>
```

Ainsi, avec XSLT, il est possible d'extraire les informations XML d'un document A et de les transformer en une autre représentation XML, B, destinée à être communiquée à un tiers (client, partenaire, etc).

Cela autorisera notamment à exploiter les données selon une interface spécifique à un endroit (par exemple une commande vue par le client) et selon une autre interface à un autre endroit (cette même commande vue par le fournisseur). On pense bien entendu aux échanges de données informatisés (EDI).

Le fichier de style XSLT fait appel à des moteurs de transformation diversifiés : outils standardisés comme le moteur Transform, ou FO (Formatting Object), outils propriétaires comme sait le faire Microsoft. En début de chaque document xsl, on doit donc indiquer à quel(s) moteur(s) on soumet les demandes de transformations, chaque moteur étant spécialisé dans un espace de noms (xmlns pour xml names-space).

```
xmlns:xsl =http://www.w3.org/1999/XSL/Format     <!-- moteur pour les noms XSL standard -->
xmlns:fo=http://www.w3.org/1999/XSL/Format     <!-- noms pour FO -->
xmlns:mscom="http://msdn.microsoft.com/aboutmsdn/rss/"  <!-- noms pour les transformations rss microsoft -->
XSL:FO et html2fo
```

Xsl:fo est une déclinaison du xsl permettant de produire les éléments de mise en forme riches que l'on peut trouver dans un traitement de texte : mises en forme de paragraphes, de caractères, de tableaux, etc.

Certains utilitaires prennent en charge toute la transformation partant d'un document décrit dans cette syntaxe xsl:fo pour générer un fichier formaté (rtf, pdf, etc).

Html2fo, permet une telle transformation automatisée. Le projet n'est pas encore finalisé et je n'ai pas réussi à le faire fonctionner.

Exploration de contenu

XPath est un langage spécifique défini pour naviguer à l'intérieur d'un fichier XML, sans se préoccuper de la structure arborescente sous-jacente, à la façon du SQL pour les bases de données.

Par exemple on peut extraire toutes les propriétés d'un type particulier (toutes les références d'articles, tous les montants de commande, etc).

Ce langage n'a pas été défini en XML pour des raisons d'efficacité et d'optimisation (on navigue directement dans le code sans passer par une représentation mémoire intermédiaire).

III Modèle objet de document (DOM)

Pour permettre à des applications d'exploiter simplement le contenu informatif des fichiers XML, un ensemble d'interfaces a été constitué, basé sur une représentation objet. Le fichier est contrôlé par le parseur, qui le convertit en un arbre mémoire. L'application utilise la syntaxe des interfaces DOM (Document Object Model) pour traiter le contenu de cette image. Ces interfaces permettent de gérer notamment :

- Les propriétés et le contenu du document (Document) cible à constituer : forme, disposition, couleurs, navigateur, etc, mais aussi inscription de données, extraction de contenu
- Les erreurs (DOMException), aussi bien dans la constitution du contenu que dans l'interaction avec l'extérieur.
- La navigation dans l'arbre XML relatif au fichier (NODE et ELEMENT)…

IV Espace de nom et DTD

Les documents XML ont pour vocation à être partagés entre différents opérateurs (utilisateurs à différents stade de la prise en charge d'une commande, relation client/fournisseur, donneur d'ordre/prestataire, entreprise/administration, etc.).

A ce titre, il faut procéder à certaines validations pour assurer la compatibilité.

Identification des balises

Lorsque l'on est amené à mettre en commun des fichiers XML définis indépendamment les uns des autres (par exemple dans une relation entre entreprises), il peut être intéressant de repérer les balises portant le même nom à différents endroits du réseau.

On parlera d'espace de nom, c'est à dire la possibilité offerte de préfixer les balises par le nom URL du site détenteur du document sous la forme :

```
<NomDuSite:nomDonnée>ValeurData</ NomDuSite:nomDonnée>
<!-- exemple -->
<central.fr:client>Robert</central.fr:client>
```

C'est ce que l'on utilise pour la syntaxe XSL.

Cet espace de nom permettra de déterminer le sens exact d'une balise.

Conformation à une syntaxe

Par ailleurs, lorsque l'on est amené à échanger des fichiers XML dans l'intention d'en exploiter le contenu de part et d'autre, il est important de disposer d'une syntaxe commune pour la structuration des informations.

On parle d'une DTD (Document Type Definition) qui décrit, dans un format XML particulier et précis

(type de donnée, obligation de présence ou non, etc), la structure correcte que devrait avoir un fichier XML correspondant à un traitement déterminé.

Grâce à un outil de contrôle (généralement intégré au parseur) on pourra vérifier qu'un fichier est conforme à une syntaxe DTD déterminée.

Cela s'avère très utile, par exemple, pour valider un fichier XSLT de transformation d'une syntaxe A vers une syntaxe B, passant du bon de commande du côté acheteur au bon de fabrication côté fournisseur par exemple.

V Utilisations de XML

Alors que l'on dispose déjà de nombreux formats standards (HTML, .dbf pour les bases de données, rtf pour les traitements de texte, csv pour les tableurs, etc.) et de langages d'exploitation (SQL en particulier), XML vient offrir une solution unique pour la représentation, la structuration et le stockage d'informations de tout type. Il se veut donc un outil universel, très léger (c'est du simple texte) donc facile à transporter sur internet.

Ce n'est pas un langage de plus, il se veut plutôt un langage pour toutes les informations.

Ainsi, on pourra aussi bien avoir des bases de données XML, des documents textuels XML, des tableaux XML, etc. Et tous pourront être exploités de la même manière par les langages XSLT pour la transformation d'un format à un autre, Xpath pour l'interrogation et l'exploitation, DOM pour la manipulation, la gestion d'erreurs, etc.

XML présente l'avantage de s'appuyer sur des éléments standard aujourd'hui (il peut être transmis par des serveurs Web, donc des outils largement répandus), il ne nécessite pas de conversion de format lors d'échanges à travers les réseaux, il est largement évolutif puisque les balises sont libres et peuvent servir à définir de nouveaux modes d'usage.

Une de ses applications initiales est de devenir l'outil de référence dans les échanges de données informatisés (EDI) entre entreprises d'un même secteur ou dans une relation avec l'administration fiscale.

Il a aussi été retenu comme langage de données dans la mise en place de services Web (applications disponibles en ligne, comme des traitements de texte, des outils de gestion, des programmes scientifiques, etc.).

Il est en train de devenir le langage de base pour la représentation et l'enregistrement des données dans les outils bureautiques (aussi bien dans le monde libre avec Open Office et le format Open Document Format que dans le monde Microsoft).

Des bibliothèques de documents XML de référence, associées à des DTD pour contrôler la validité de documents utilisateurs, devraient permettre la création et la diffusion d'applications prêtes à l'emploi.