Cours de XML - Concepts de base

Par G. Chagnon

Date de publication : 16 mai 2002

Dernière mise à jour : 13 janvier 2009

Ce cours présente un bref historique et les concepts de base de XML.

I - Historique : de SGML à XML

XML dérive d'un langage développé dans les années 80, le **SGML**. Ce langage était complexe à apprendre et utiliser quotidiennement. Une version allégée, le **HTML** a donc été développée ; mais ce dernier, malgré de nombreuses adaptations, ne pouvait pas être étendu à l'infini, au fur et à mesure de l'augmentation des besoins des développeurs. C'est alors que fut créé le **XML**.

I-A - SGML et HTML

I-A-1 - Le SGML

Le **SGML** (Standard Generalized Markup Language, langage de balisage standard généralisé), premier essai de normalisation concernant les documents électroniques, a été adopté comme standard en 1986.

Le **SGML** est constitué de plusieurs composants. Ceux-ci changent pour chaque application **SGML** :

- Définition du type de document (**DTD**) : ce composant sert à préciser la structure du document ;
- Instance du document : il s'agit des données à stocker elle-mêmes, présentées sous une forme structurée selon les éléments **SGML** qui ont été définis dans la **DTD**. Même si une instance de document peut partager une **DTD** avec d'autres documents, elle ne peut se conformer qu'à une seule **DTD** ;
- Synthèse du document : ce composant sert à préciser les principaux aspects de l'application **SGML**. C'est à ce niveau que sont déterminées les options et qu'est précisé le jeu de caractères qui sera utilisé ainsi que les autres fonctions similaires.

Ce langage servant à préciser la structure d'un document quelconque, il est compréhensible que sa généralité le rende difficile d'apprentissage, et complexe d'emploi. En particulier, il était inadapté à l'écriture de documents pour Internet. Il a donc été nécessaire d'en dériver le langage **HTML**.

I-A-2 - Le HTML

Comme il a été dit précédemment, c'est aujourd'hui le standard du développement web. Il a été étendu par le **XHTML**, un langage qui lui est extrêmement similaire, mais permettant la production de documents aux normes **XML**.

Ce langage est facile à apprendre et à utiliser ; il a d'ailleurs donné lieu au développement de nombreux outils de publication sur Internet :

- logiciels WYSIWYG -"*What You See Is What You Get*"- tels que **Nvu**, FrontPage ou **Dreamweaver** ;
- outils de publication de contenu comme **SPIP**, **eZPublish**, **Joomla**, **PHPNuke**...

... sans oublier qu'il est possible d'utiliser de simples éditeurs de texte comme **(X)Emacs** sous Linux ou bien le **Bloc-Notes** sous Windows.

Le **HTML** étant une application **SGML**, il est donc lié lui aussi à une **DTD**. Il en existe parfois plusieurs par version d'**HTML**. Celles de la version 4.01 se trouvent référencées sur le site web du **W3C** : *DTD HTML 4.01 Strict, Transitional, Frameset*...

Un inconvénient du **HTML** est son champ d'action limité : il n'est ainsi pas possible de définir autre chose qu'une page Web, ce qui est compréhensible puisque ce langage a été spécifiquement conçu pour cela.On ne peut par exemple pas ajouter de nouveaux éléments (on pourrait imaginer insérer des équations mathématiques, mais ce n'est pas possible en **HTML** stricto sensu).

Au bout de quelques années, la demande se faisant de plus en plus forte pour la définition d'un nouveau format, et le langage **HTML** rencontrant ses limites, le W3C a commencé à organiser **des groupes de travail sur XML**, un autre descendant du **SGML**.

I-B - XML

Le **XML** est un dérivé du **SGML**. Il tente de se servir des principes de simplicité du **HTML** et de la souplesse **SGML**.

Simplification de **SGML**, puisqu'il ne reste que 35 pages de spécification contre 155 pages en ce qui concerne le **SGML**, le format **XML** est rapidement apparu adapté à beaucoup plus d'usages que ses concepteurs le pensaient initialement. Il conserve certains aspects de **SGML**.

Le plus important point commun avec le **SGML** est le fait que tout document **XML** peut être basé sur une **DTD** ou un Schéma. Cette association n'est cependant pas obligatoire, et un fichier **XML** peut très bien se suffire à lui-même.

Une autre caractéristique importante est que dans un document **XML**, la mise en forme des données est totalement séparée des données elles-mêmes. Cela permet de séparer complètement l'information (le contenu) de son apparence (le contenant), et donc de fournir plusieurs types de sortie pour un même fichier de données, en fonction de l'utilisateur ou de l'application demandeuse (autre document **XML**, tableau, graphique, image, animation multimédia, fichier **HTML**, fichier **PDF**...).

De plus, la possibilité de créer les éléments que l'on désire permet de rendre le fichier lui-même lisible -et modifiable- par un être humain : on peut donner aux informations contenues dans un tel fichier les étiquettes que l'on veut, et les ordonner selon son désir.

Un document **XML** peut ainsi prévoir plusieurs cibles, comme par exemple l'écran d'un téléphone portable, celui d'un ordinateur de bureau, une base de données, une application logicielle, etc.

Il est également possible d'effectuer des sélections par tri, des générations automatiques de tables des matières et bien d'autres fonctions encore, grâce au langage de feuilles de style **XSLT**.

II - Mise en oeuvre

II-A - Procédure

XML permet de nettement séparer forme et fond. Par exemple, cela signifie que pour produire un document **HTML** à partir de données en utilisant le format **XML**, il est nécessaire d'écrire au *moins* deux fichiers, le premier contenant les données à mettre en forme, le second les informations nécessaires à cette mise en forme. En pratique, et dans un souci de normalisation et de généralisation, il peut s'avérer nécessaire d'ajouter un troisième fichier à l'ensemble :

1. *Optionnel* : un fichier peut être nécessaire pour définir a priori les balises auxquelles le document **XML** pourra avoir recours. Ce fichier peut être soit une *Document Type Definition* soit un Schema (voir plus loin dans le cours) ;
2. Le document contenant les données elles-mêmes, c'est-à-dire le document **XML** à proprement parler (une instance du document, au sens **SGML**) ;
3. Le document contenant les informations de mise en forme, permettant de produire un fichier dans le format de sortie voulu : une feuille **XSLT** (eXtensive Stylesheet Language Transformations).

Dans le cas d'un format de sortie **HTML**, il peut être également nécessaire d'ajouter une feuille de style **CSS**.

En résumé, si l'on veut produire un fichier **HTML** à partir de données mises sous format **XML**, il faut :

1. Créer éventuellement un fichier définissant les balises utilisables ;
2. Créer le fichier de données **XML** ;
3. Créer la feuille de style **XSL** permettant la production du fichier **HTML** ;
4. Créer éventuellement une feuille de style **CSS**.

Nous verrons au long de ce cours successivement les étapes 2, 1 et 3.

II-B - Exemple : une bibliographie

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- La ligne ci-dessus est le prologue -->
<!-- Élément racine -->
<biblio>
  <!-- Premier enfant -->
  <livre>
    <!-- Élément enfant titre -->
    <titre>Les Misérables</titre>
    <auteur>Victor Hugo</auteur>
    <nb_tomes>3</nb_tomes>
  </livre>
  <livre>
    <titre>L'Assomoir</titre>
    <auteur>Émile Zola</auteur>
  </livre>
  <livre lang="en">
    <titre>David Copperfield</titre>
    <auteur>Charles Dickens</auteur>
    <nb_tomes>3</nb_tomes>
  </livre>
</biblio>
```

Exercice 1. Structuration d'informations 1

XML permet de *structurer* une information. Il est donc nécessaire, avant d'envisager d'utiliser ce format, de se familiariser avec cette structuration.

Le paragraphe suivant contient de l'information "en vrac". Réorganisez-la de manière à mettre en évidence sa structure logique, sans forcément passer par une mise en forme XML.

Une bouteille d'eau Cristaline de 150 cl contient par litre 71 mg d'ions positifs calcium, et 5,5 mg d'ions positifs magnésium. On y trouve également des ions négatifs comme des chlorures à 20 mg par litre et des nitrates avec 1 mg par litre. Elle est recueillie à **St-Cyr la Source**, dans le département du Loiret. Son code barre est 3274080005003 et son pH est de 7,45. Comme la bouteille est sale, quelques autres matériaux comme du fer s'y trouvent en suspension. Une seconde bouteille d'eau Cristaline a été, elle, recueillie à la source d'**Aurèle** dans les Alpes Maritimes. La concentration en ions calcium est de 98 mg/l, et en ions magnésium de 4 mg/l. Il y a 3,6 mg/l d'ions chlorure et 2 mg/l de nitrates, pour un pH de 7,4. Le code barre de cette bouteille de 50 cl est 3268840001008. Une bouteille de même contenance est de marque Volvic, et a été puisée à... **Volvic**, bien connu pour ses sources donnant un pH neutre de 7. Elle comprend 11,5 mg/l d'ions calcium, 8,0 mg/l d'ions magnésium, 13,5 mg/l d'ions chlorures et 6,3 mg/l d'ions nitrates. Elle contient également des particules de silice. Son code barre est 3057640117008.

PS : Volvic est dans le Puy-de-Dôme...

Correction

```
<cave>
  <bouteille>
    <marque>Cristaline</marque>
    <composition>
      <ion type="positif">calcium 71mg/l</ion>
      <ion type="positif">magnésium 5,5mg/l</ion>
      <ion type="negatif">chlorure 20mg/l</ion>
      <ion type="negatif">nitrate 1mg/l</ion>
      <autre type="metal">fer</autre>
    </composition>
    <source>
      <ville>St-Cyr la Source</ville>
      <departement>Loiret</departement>
    </source>
    <code_barre>3274080005003</code_barre>
    <contenance unit="cl">150</contenance>
    <ph>7,45</ph>
  </bouteille>
  <bouteille>
    <marque>Cristaline</marque>
    <composition>
      <ion type="positif">calcium 98mg/l</ion>
      <ion type="positif">magnésium 4mg/l</ion>
      <ion type="negatif">chlorure 3,6mg/l</ion>
      <ion type="negatif">nitrate 2mg/l</ion>
    </composition>
    <source>
      <ville>Aurèle</ville>
      <departement>Alpes Maritimes</departement>
    </source>
    <code_barre>3268840001008</code_barre>
    <contenance unit="cl">50</contenance>
    <ph>7,4</ph>
  </bouteille>

  <bouteille>
    <marque>Volvic</marque>
    <composition>
      <ion type="positif">calcium 11,5mg/l</ion>
      <ion type="positif">magnésium 8mg/l</ion>
      <ion type="negatif">chlorure 13,5mg/l</ion>
      <ion type="negatif">nitrate 6,3mg/l</ion>
    </composition>
    <source>
      <ville>Volvic</ville>
      <departement>Puy-de-Dôme</departement>
    </source>
    <code_barre>3057640117008</code_barre>
    <contenance unit="cl">50</contenance>
```

```
    <ph>7</ph>
  </bouteille>
</cave>
```

Exercice 2. Structuration d'informations 2

Cet exercice est du même type que l'exercice précédent. Il s'agit de structurer, sous la forme d'un fichier XML, le texte suivant :

Il existe diverses variétés de nuages. La plupart de ceux dont nous allons parler ne produit aucun "hydrométéore", sauf le cumulonimbus, qui est accompagné d'averses (parfois sous la forme de neige, de grésil ou de grêle).

L'altocumulus et le cirrocumulus partagent les mêmes *"espèces" : lenticularis, stratiformis, castellanus et flocus*. On retrouve ces deux espèces également chez le cirrus, ainsi que les *espèces spissatus, uncinus et fibratus*. Les *espèces stratiformis, lenticularis et castellanus* sont quant à elles partagées également avec les strato-cumulus.

Ces derniers peuvent se traîner au ras du sol et monter à 2000m, mais certains nuages ont une altitude minimale à peine plus élevée, puisqu'elle n'est que de 200m pour les cumulus, et de 300m pour les cumulonimbus. Il est vrai que ces derniers compensent en montant jusqu'à une altitude maximale de 18000m, soit plus haut encore que les cirrus, qui plafonnent à 12000m. L'altitude minimale de ces derniers coïncide avec la fin de la présence possible des altocumulus, à 6000m. Et c'est autour de cette zone, entre 5000 et 7000m, que se trouvent les cirrocumulus. L'altitude minimale des altocumulus est de 2000m, soit quatre fois moins que l'altitude maximale des cumulus.

Ces pauvres cumulus ne sont pas favorisés en nom d'espèces, puisqu'ils se trouvent affligés de noms tels que *fractus, mediocris, humilis et congestus*... alors que les cumulonimbus ont des espèces aux noms plus... capillaires tels que *calvus, capillatus*. Les très gros cumulonimbus sont appelés *mammatus*.

Correction

```
<nuages>
  <nuage>
    <nom>
      altocumulus
      <espece>lenticularis</espece>
      <espece>stratiformis</espece>
      <espece>castellanus</espece>
      <espece>flocus</espece>
    </nom>
    <altitude max="6000" min="2000"/>
    <hydrometeores>Aucun.</hydrometeores>
  </nuage>
  <nuage>
    <nom>
      cirrus
      <espece>flocus</espece>
      <espece>castellanus</espece>
      <espece>spissatus</espece>
      <espece>uncinus</espece>
      <espece>fibratus</espece>
    </nom>
    <altitude max="12000" min="6000"/>
    <hydrometeores>Aucun.</hydrometeores>
  </nuage>
  <nuage>
    <nom>
      cirrocumulus
      <espece>lenticularis</espece>
      <espece>stratiformis</espece>
      <espece>flocus</espece>
      <espece>castellanus</espece>
    </nom>
    <altitude max="7000" min="5000"/>
    <hydrometeores>Aucun.</hydrometeores>
  </nuage>
```

```
  <nuage>
    <nom>
      cumulus
   <espece>fractus</espece>
      <espece>humilis</espece>
      <espece>mediocris</espece>
      <espece>congestus</espece>
    </nom>
    <altitude max="8000" min="200"/>
    <hydrometeores>Aucun en général.</hydrometeores>
  </nuage>
  <nuage>
    <nom>
      strato-cumulus
   <espece>stratiformis</espece>
      <espece>lenticularis</espece>
      <espece>castellanus</espece>
    </nom>
    <altitude max="2000" min="0"/>
    <hydrometeores>Aucun.</hydrometeores>
  </nuage>
  <nuage>
    <nom>
      cumulonimbus
   <espece>calvus</espece>
      <espece>capillatus</espece>
      <espece>mammatus</espece>
    </nom>
    <altitude max="18000" min="300"/>
    <hydrometeores>Averses (parfois de neige, de grésil ou de grêle).</hydrometeores>
  </nuage>
</nuages>
```

III - Structure d'un document XML

III-A - Généralités

Les personnes ayant élaboré **XML** avaient en tête la simplicité de déclaration et la validation de la structure. De plus, elles souhaitaient tenir compte des expériences du passé en intégrant la grande souplesse syntaxique du **SGML** et la simplicité d'écriture du **HTML**.

Comme cela est illustré par l'**exemple** précédent, un fichier **XML** est composé d'un prologue, d'un élément racine et d'un arbre. Cet arbre est constitué d'éléments imbriqués les uns dans les autres (ayant une relation parent-enfant) et d'éléments adjacents.

- Les premières lignes forment le prologue, constitué dans l'exemple précédent de la déclaration **XML**, puis éventuellement d'une déclaration de type de document (une **DTD**) ;
- L'élément **biblio** est notre élément racine (en anglais : *document element*) ; il est constitué de trois éléments livre. Dans chacun d'entre eux nous retrouvons la même composition, c'est-à-dire : un élément **titre**, un élement **auteur** et éventuellement un élément **nb_tomes**. L'élément livre, de plus, peut avoir un attribut **lang** ;

Même s'il est simple de comprendre ce code, on s'aperçoit mieux d'une éventuelle erreur lorsqu'on visualise ce même fichier dans **un navigateur**.

III-B - Le prologue

III-B-1 - Déclaration XML

Cette déclaration fait partie des « instructions de traitement ». Exemple de déclaration **XML** :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

On distingue trois informations fournies dans cette déclaration :

1. **version** : version du **XML** utilisée dans le document, 1.0 en ce qui nous concerne (la dernière version du langage, 1.1, date de février 2004 mais ne change rien quant à ses bases) ;
2. **encoding** : le jeu de codage de caractères utilisé. Le jeu de caractères habituel pour le français est le ISO-8859-1. Il a tendance à être remplacé par l'ISO-8859-15 en attendant la généralisation de l'Unicode. Par défaut, l'attribut **encoding** a la valeur **UTF-8**. Cela permet à l'ordinateur de « savoir » quel caractère il doit afficher en réponse aux combinaisons de 1 et de 0 que contient le fichier sur le disque dur ;
3. **standalone** : dépendance du document par rapport à une **déclaration de type de document**. Si **standalone** a la valeur **yes**, le processeur de l'application n'attend aucune déclaration de type de document extérieure au document. Sinon, le processeur attend une référence de déclaration de type de document. La valeur par défaut est **no**.

Cette déclaration est facultative, mais il est préférable de l'utiliser. Dans ce cas les attributs version, encoding et standalone doivent être placés dans cet ordre. Si elle est utilisée, elle doit être placée en toute première ligne du document **XML**. Par exemple, il ne faut placer ni commentaire, ni même une simple ligne vide avant elle.

III-B-2 - Instructions de traitement

Une instruction de traitement est une instruction interprétée par l'application servant à traiter le document **XML**. Elle ne fait pas totalement partie du document. Les instructions de traitement qui servent le plus souvent sont la déclaration **XML** ainsi que la déclaration de feuille de style. Exemple d'instruction de traitement :

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

Dans cet exemple, l'application est xml-stylesheet, le processeur de feuille de style du **XML**. Deux feuilles de style différentes peuvent être utilisées, les **XSL** (propres au **XML**) ainsi que les **CSS** (feuilles de style apparues avec le **HTML**). L'attribut type indique de quel type de fichier il s'agit (text/css pour les feuilles de style **CSS**, par exemple) et l'attribut **href** indique l'URL du fichier. Cette instruction de traitement est notamment utilisée par les navigateurs Internet pour la mise en forme du document.

III-B-3 - Déclaration de type de document (DTD)

Cette déclaration, lorsqu'elle est présente, permet de définir la structure du document. Elle peut être de deux types, externe ou interne. Exemple de déclaration de type de document :

```
<!DOCTYPE biblio SYSTEM "biblio.dtd">
```

Ce type de déclaration est celui d'une déclaration de type de document externe (voir le **chapitre sur les DTD**). Elle définit l'ensemble des éléments utilisables dans le document, y compris l'élément-racine (ici **biblio**) ainsi que l'emplacement où se trouve le fichier **biblio.dtd** dans lequel se trouve définie la structure du document.

Bien que facultative, il est souvent très intéressant de posséder une **DTD**, en particulier externe, simplement pour vérifier la validité du document **XML**. Il est ainsi recommandé d'en utiliser une dans le cas du développement parallèle de plusieurs fichiers **XML** destinés à subir un traitement particulier (développement par plusieurs personnes par exemple).

L'autre type de document permettant de définir la structure d'un fichier, le **schéma XML**, s'utilise autrement, comme nous le verrons plus tard.

III-C - Les commentaires

En **XML**, les commentaires se déclarent de la même façon qu'en **HTML**, car ils reprennent la syntaxe du **SGML**. Ils commencent donc par <!-- et se terminent par -->. Ils peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'extérieur d'une autre balise.

Exemples de commentaires valides :

```
<!-- ceci est correct -->
<elt> <!-- ceci est correct aussi -->
Un peu de texte </elt>
```

Remarque : En raison de la compatibilité **XML/SGML**, la chaîne de caractères -- est interdite dans un commentaire.

III-D - L'arbre d'éléments

III-D-1 - Introduction

Un document **XML** peut se représenter sous la forme d'une arborescence d'*éléments*. Cette arborescence comporte une racine (unique), des branches et des feuilles. Reprenons l'exemple précédent.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<biblio>
  <livre>
    <!-- Élément enfant titre -->
    <titre>Les Misérables</titre>
    <auteur>Victor Hugo</auteur>
    <nb_tomes>3</nb_tomes>
  </livre>
  <livre>
    <titre>L'Assommoir</titre>
```

```
    <auteur>Émile Zola</auteur>
    <couverture couleur="rouge" />
  </livre>
  <livre lang="en">
    <titre>David Copperfield</titre>
    <auteur>Charles Dickens</auteur>
    <nb_tomes>3</nb_tomes>
  </livre>
</biblio>
```

III-D-2 - Élément racine

L'élément-racine (en anglais : document element) est, comme son nom l'indique, la base du document **XML**. Il est unique et englobe tous les autres éléments. Il s'ouvre juste après le prologue, et se ferme à la toute fin du document. Dans l'exemple ci-dessus, l'élément racine est **biblio**.

III-D-3 - Les éléments

Les éléments forment la structure même du document : ce sont les branches et les feuilles de l'arborescence. Ils peuvent contenir du texte, ou bien d'autres éléments, qui sont alors appelés « éléments enfants », l'élément contenant étant quant à lui appelé logiquement « élément parent ».

Exemple d'élément contenant du texte :

```
<titre>Les Misérables</titre>
```

Exemple d'élément contenant d'autres éléments :

```
<livre>
  <titre>L'Assommoir</titre>
  <auteur>Émile Zola</auteur>
  <couverture couleur="rouge" />
</livre>
```

D'autres éléments sont vides : ils ne contiennent pas d'élément-enfant. Exemple d'élément vide :

```
  <couverture couleur="rouge" />
```

Il faut prendre garde à ne pas confondre la balise **<elt>** avec l'élément **elt**.

III-D-4 - Les attributs

Tous les éléments peuvent contenir un ou plusieurs attributs. Chaque élément ne peut contenir qu'une fois le même attribut. Un attribut est composé d'un nom et d'une valeur. Il ne peut être présent que dans la balise *ouvrante* de l'élément (par exemple, on n'a pas le droit d'écrire **</livre lang="en">**).

Exemple d'utilisation d'un élément avec attribut :

```
<instrument type="vent">trompette</instrument>
```

Exemple d'utilisation d'un élément vide avec attributs :

```
<img src="ours.png" alt="Gros ours" width="56" height="100" />
```

III-D-5 - Les entités

Il existe des entités définissables et définies. Elles peuvent être analysables ou non, internes ou externes. La déclaration des entités s'effectue au sein de la **DTD**. Elles peuvent être utilisées aussi bien dans la **DTD** que dans le document **XML**. Nous reviendrons plus en détails sur les entités et leur utilisation ultérieurement.

Certains caractères ayant un sens précis en **XML**, il est nécessaire de leur trouver un remplaçant lorsque l'on a besoin de les insérer dans un document. On a recours dans ce cas à des entités prédéfinies. Ces entités sont :

Caractère	Entité
&	&amp;
<	<
>	>
"	"
'	&aquot;

Table 1. Liste des entités prédéfinies

Il n'existe pas d'entité prédéfinie pour les lettres accentuées ou pour les alphabets latins. Il faut utiliser à la place les entités numériques du type (où **n** est une valeur décimale). La valeur numérique correspond au code ISO 10646 ; par exemple le caractère é est codé par l'entité numérique . Il est néanmoins possible d'importer des entités en provenance d'une autre **DTD**, par exemple **celle du HTML**.

III-D-6 - Les sections CDATA

Une section **CDATA** est une section pouvant contenir toutes sortes de chaîne de caractères. Une section **CDATA** permet de définir un bloc de caractères ne devant pas être analysés par le processeur **XML**. Cela permet entre autres de garder dans un bloc de texte un exemple de code à afficher tel quel.

Exemple d'utilisation de **CDATA** :

```
<![CDATA[Une balise commence par un < et se termine par un >.
```

III-E - Règles de composition

Un certain nombre de règles de base doivent être respectées :

1. Un nom d'élément ne peut commencer par un chiffre. Si le nom n'est composé que d'un seul caractère, ce doit être une lettre comprise entre « a » et « z » pour les minuscules, « A » et « Z » pour les majuscules. S'il est composé d'au moins deux caractères, le premier peut être « _ » ou « : ». Le nom peut ensuite être composé de lettres, chiffres, tirets, tirets bas et deux points. La syntaxe **XML** est sensible à la casse (le format distingue majuscules et minuscules).
2. Toutes les balises portant un contenu non vide doivent être fermées. La balise de début, la balise de terminaison et le contenu entre deux sont globalement appelés élément ;
3. Les balises n'ayant pas de contenu doivent se terminer par /> (voir la balise **** ci-dessus) ;
4. Les valeurs d'attributs doivent être entre *guillemets* ;

Un document respectant ces critères est dit *bien formé* (well formed).

Il est aussi possible de définir des règles plus strictes indiquant quelles sont les séquences et imbrications de balises ou les attributs autorisés. Cela se fait à l'aide d'une **DTD** ou d'un **Schéma**. Il est alors possible d'effectuer une validation des documents faisant référence à une **DTD** pour s'assurer qu'ils respectent bien les règles qui y sont mentionnées.

Un document bien formé dont la syntaxe est conforme aux règles stipulées dans une **DTD** ou un **Schema XML** est dit valide. Nous reviendrons sur cette notion ultérieurement.

IV - Support par les navigateurs

IV-A - Famille Mozilla et Internet Explorer

Ces navigateurs (à partir de la version 5 pour **Internet Explorer**) permettent l'affichage des documents **XML** sous la forme d'une arborescence dans le cas général ; si une feuille de style est spécifiée, le navigateur l'interprète (à partir de la version 6 pour **Internet Explorer**).

IV-B - Netscape

Le tour de la question est vite fait en ce qui concerne les navigateurs de la famille Netscape, puisque seules les versions égales ou supérieures à 6 affichent les fichiers **XML** à condition toutefois qu'une feuille de style **XSL** soit déclarée dans le prologue. Néanmoins son implémentation est encore partielle.

IV-C - Opera

Opera supporte **XML** depuis la version 8, et le langage **XSL** depuis la version 9.

IV-D - Conclusion

En définitive, deux solutions se présentent :

1. Soit on veut visualiser le source **XML**. Dans ce cas, mieux vaut ne pas fournir de feuille de style dans le prologue et utiliser **Internet Explorer** ou un navigateur de la famille **Mozilla**. ;
2. Soit on veut tester la feuille de style. Dans ce cas, on peut ajouter l'appel à la feuille de style dans le fichier **XML**, et constater le résultat sous n'importe quel navigateur récent.