Introduction à XML

par Victor Stinner

Date de publication : 16/09/2003

Dernière mise à jour : 01/04/2005

Tutoriel d'introduction pour XML (eXtensible Markup Language). Téléchargez la version PDF. Téléchargez la version HTML/ZIP.

I - Introduction

I.A - Qu'est-ce que c'est ?

XML est l'abréviation de Extensible Markup Language. Contrairement aux « on dit », XML n'est pas un langage de programmation. On ne peut pas faire de tests, ni inclure un fichier dans un autre. En très gros, XML sert uniquement à stocker des données.

XML est simplement une méthode pour représenter les données. Celles-ci sont écrites entre des balises ou sous forme d'attributs, et l'ensemble est écrit sous forme d'un arbre.

I.B - Entête et encodage

Les documents XML doivent posséder une première ligne qui est de la forme :

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

Cette ligne défini le numéro de version (ici 1.0), et l'encodage des caractères. Pour l'encodage, il existe principalement :

- **ISO-8859-1** : Etats-Unis et Europe de l'Ouest. C'est le plus simple pour écrire un document en Français car les lettres accentuées ne prennent qu'un caractère (octet).
- **UTF-8** : Format international qui permet d'écrire dans n'importe quelle langue. Chaque caractère à un codage unique (Unicode). Les caractères latins (a-z et A-Z), chiffres arabes (0-9) et quelques autres caractères de ponctuation française sont codés sur un seul octet. Les autres caractères sont codés sur plusieurs octets (la taille peut varier de 2 à 4 caractères, peut-être plus ?). Les caractères accentués français prennent par exemple deux octets. Voyez le site d'Unicode dans la liste des liens en bas de page.


Pour écrire et lire en UTF-8, vos logiciels doivent supporter l'UTF-8 ! Sinon les caractères de plus d'un octet apparaîtront comme une chaîne de caractères bizarres.

I.C - Exemple

Exemple

```
<pere nom="Gilbert">
  <fils nom="Victor">
    C'est moi
  </fils>
  Mon père.
</pere>
```

Un document XML a une et une seule balise racine. Ici elle s'appelle pere :-) Une balise pour avoir un attribut, comme ici un nom. Les attributs sont un nom (chaîne de lettres A à Z, minuscule ou majuscule) dont la valeur est écrite en guillemets : nom="valeur". Une balise peut contenir du texte et/ou des balises "fils". On parle de balise parent (parent en anglais), et enfants (childrens) en référence aux arbres généalogiques. Une balise peut également être vide (ne contenir aucune données en dehors de son nom).

I.D - Règles à respecter

Pour écrire en XML, il existe quelques règles de base à respecter :

1 Un document XML ne contient qu'une balise racine (parent).

2 Une balise doit être refermée. Oubliez donc l'ancien HTML et ses balises fermées ou non, selon l'humeur du W3.org :-) Exemple courant : la balise img (image) doit être refermée. On n'écrit plus

```
<img src="image.png">
```

mais

```
<img src="image.png" />
```

3 Une balise vide peut s'écrire de différents manières. Prenons l'exemple de la balise HTML br. On peut l'écrire sous trois formes différentes :

```
<br/>
<br />
<br></br>
```

4 Les espaces (l'indentation) n'est pris en compte que dans le contenu des balises (pas dans la zone entre crochets < ... >).

```
<sac couleur="noir" marque="decathlon">
  Mon sac de cours.
</sac>
```

peut s'écrire plus "proprement" (à mon goût) :

```
<sac
  couleur="noir"
  marque="decathlon">
  Mon sac de cours.
</sac>
```

Par contre à l'intérieur des balises, chaque espace compte.

```
<p>Une petite phrase pour remplir la balise p ...</p>.
```

est différent de

```
<p>Une petite phrase
pour remplir
la balise p ...</p>
```

ou encore

```
<p>
  Une petite phrase pour remplir
  la balise p ...
</p>
```

Mais on verra qu'avec XSLT, il existe une option qui permet de supprimer les espaces et retours à la ligne inutiles (utilisés pour l'indentation).

II - Un exemple pour comprendre : un carnet d'adresse

II.A - Représentation d'une personne

L'exemple bidon mais classique : un carnet d'adresse. On va stocker une liste de personnes dont on possède différentes informations (nom, courriel, téléphone, etc.). Voyons déjà la représentation d'une personne. Représentation intuitive :

```
<personne>
  <nom>Victor STINNER</nom>
  <email>victor.stinner SUR haypocalc.com</email>
  <adresse>282, 7e rue à Québec (CANADA)</adresse>
</personne>
```

Cette représentation a un gros désavantage : on ne peut différencier le nom du prénom, et on ne peut retrouver le pays dans l'adresse. On peut alors utiliser une autre représentation :

```
<personne>
  <nom>
    <prenom>Victor</prenom>
    <nom>STINNER</nom>
  </nom>
  <email>
    <identifiant>victor.stinner</identifiant>
    <serveur>haypocalc.com</serveur>
  </email>
  <adresse>
    <numero>282</numero>
    <rue>7e rue</rue>
    <ville>Québec</ville>
    <pays>CANADA</pays>
  </adresse>
</personne>
```

La même adresse prend beaucoup plus de place et est plus longue à écrire. L'avantage ? Et bien, on peut correctement identifier les informations. On peut alors envisager d'effectuer un tri par nom, par prénom, par serveur d'email, par pays, etc. On peut également faire des recherches de personnes. On commence à voir se construire l'arbre des données. Le pays est stocké dans la balise adresse, qui elle même est stockée dans la balise père personne. Cette représentation est assez logique, et la localisation d'une balise (dans l'arbre XML) est une information en elle même. On pourrait trouver plusieurs balises nom dans l'arbre qui n'auront pas forcément le même sens : nom du carnet d'adresse, nom d'une personne, nom d'un groupe de personne, etc.

II.B - Représentation du carnet d'adresse

Maintenant qu'on a défini une personne, on va définir comment les personnes sont stockées dans le carnet d'adresse :

```
<carnet_adresse>
  <groupe>
    <nom>Amis</nom>
    <personne>...</personne>
    <personne>...</personne>
  </groupe>

  <groupe>
    <nom>Travail</nom>
    <personne>...</personne>
```

```
    <personne>...</personne>
  </groupe>

  <groupe>
    <nom>Famille</nom>
    <personne>...</personne>
    <personne>...</personne>
  </groupe>
</carnet_adresse>
```

J'ai regroupé les personnes dans des groupes. Ceci rajoute une information supplémentaire sur une personne : on sait à quel groupe de personnes elle appartient. J'ai donné un nom au groupe par une balise nom, mais on peut également utilisé un attribut :

```
<groupe nom="Amis">
  ...
</groupe>
<groupe nom="Famille">
  ...
</groupe>
```

Le choix entre une balise ou un attribut est laissé au programmeur. En pratique, la technique pour accéder aux données est assez semblable. Il faut par contre savoir qu'un attribut est unique, contrairement aux balises qui peuvent être répétées. Note: On peut également limiter le nombre de balises autorisées par les DTD ou XML Schéma, mais nous ne verrons pas ça dans cet article.

III - Caractères spéciaux et CDATA

Etant donné que les balises XML sont délimités par des crochets : '<' et '>', on ne peut pas écrire ces caractères directement. Il faut les écrire, respectivement, < et >. Pour s'en rappeler, lt est l'abréviation de lower than (plus petit que), et gt l'abréviation de greater than (plus grand que).

Autre solution : on peut écrire le code d'un caractère en décimal. Exemple : & représente le caractère &, code 38. Quelques codes utiles :

- 38 : caractère '&' (et commercial).
- 60 : caractère '<' (inférieur à).
- 62 : caractère '>' (supérieur à).
- 160 : caractère ' ' (espace). Ce caractère est représenté par la balise   en HTML. Personnellement, je l'ai abandonné au profit de la balises HTML pre.

Mais tout ceci est assez contraignant, plus particulièrement pour écrire du source (en langage C par exemple) qui est rempli de ces caractères. On peut alors utiliser des sections CDATA. Il suffit d'ouvrir une section par la balise <![CDATA[, et la refermer par]]>. A l'intérieur d'une section CDATA, on peut utiliser les caractères spéciaux sans problème. Par contre on ne peut pas y écrire]]> qu'il faut écrire]]>. Exemple de CDATA :

```
<p>Un peu de texte <b>HTML</b> qui ne sera pas mis en forme
car il est placé dans une section CDATA ;-)</p>
```

IV - Conclusion

IV.A - Avantages

1. Un document XML est hiérarchisé sous forme d'arbre. Cette représentation est logique est permet de faire des recherches très pointues à l'intérieur d'un document XML.
2. Le nom des balises, leur quantité, ainsi que les attributs ne sont pas limités. Si on veut les limiter, on peut passer par les DTD ou XML Schema.
3. Portabilité : un document XML est très portable et facilement lisible. Il existe des outils XML pour tous les langages courants (C/C++, Java, PHP, etc.).

IV.B - Désavantages

- La forme est tellement libre, qu'on peut avoir des formats incompatibles. Ex: une information peut-être stockée sous forme de balise, ou alors sous forme d'attribut dans deux documents différents. On peut utiliser les DTD ou XML Schema pour empecher ces problèmes.
- Le XML utilisé sous forme de fichier peut générer des fichiers très gros et difficilement éditables à la main. Pour des grands nombres d'enregistrement, un découpage en plusieurs fichiers ou l'utilisation d'une base de donnée est nécéssaire.

IV.C - Mon avis

Je pense que le format XML va être de plus en plus utilisé. Les disques durs et la mémoire sont de plus en plus gros, et se soucie de moins en moins de la taille des fichiers. Aujourd'hui, ce qui devient important c'est de retrouver une information dans une multitudes de documents. Le format XML est le début de la solution. D'un côté le XML est très souple, de l'autre il est très portable et se forme est définie selon des règles strictes. Il permet l'interopérabilités dans un environnement très hétérogène. Usez et abusez du XML, ce format a encore de beaux jours devant lui.

Le XML seul ne semble pas si intéressant. Il prend toute se puissant accompagné de son ami XSLT. Lisez rapidement mon article sur XSLT pour définitivement franchir le pas !