Le XML

Dans le terme AJAX, le X signifie XML. Ainsi, il convient d'introduire ou de rappeler quelques éléments de ce langage.

Le XML est un métalangage, soit un langage pour écrire d'autres langages.

Même s'il n'y a que peu de chances que vous conceviez un jour votre propre langage, le XML est une véritable révolution dans le panorama de la publication sur le Web. Il apparaît aujourd'hui comme incontournable car il est déjà à la base de toute une série de nouveaux langages qui sont, ou qui seront, utilisés dans la conception des pages web. Ces nouveaux langages, générés à partir du XML, en reprennent l'esprit, les règles et la syntaxe que vous découvrirez ci-après.

À la nuit des temps de l'informatique et d'Internet, existait déjà le SGML (*Standard Generalized Markup Language*), un langage normalisé pour la conception de langages de balises. Cette norme internationale (ISO8879) a comme objectif de décrire la structure et le contenu de différents types de documents électroniques. Le SGML a la particularité d'être très concis mais aussi très abstrait. En conséquence, il n'est que très difficilement abordable pour les non-initiés. Sa descendance est pourtant assez nombreuse et vous ne pouvez ne pas connaître un de ses enfants qui est un langage de balises utilisé pour la publication sur le Web : le HTML (*HyperText Markup Language*).

Le HTML ayant mal vieilli au fil des versions, le W3C Consortium, qui régit les règles de la publication sur le Web, a décidé de repartir d'une feuille blanche en revenant en quelque sorte aux sources. D'où le XML (*eXtensible Markup Language*), qui présente de fortes similitudes avec le SGML dont il est issu. Ainsi, le XML peut-être considéré comme un SGML simplifié ou un SGML qui serait plus abordable.

Le XML (*eXtensible Markup Language*) est donc un langage de balises comme le HTML, mais il est extensible ou, autrement dit, évolutif. En XML, les balises ne sont pas prédéfinies. C'est vous qui devez définir vos propres balises.

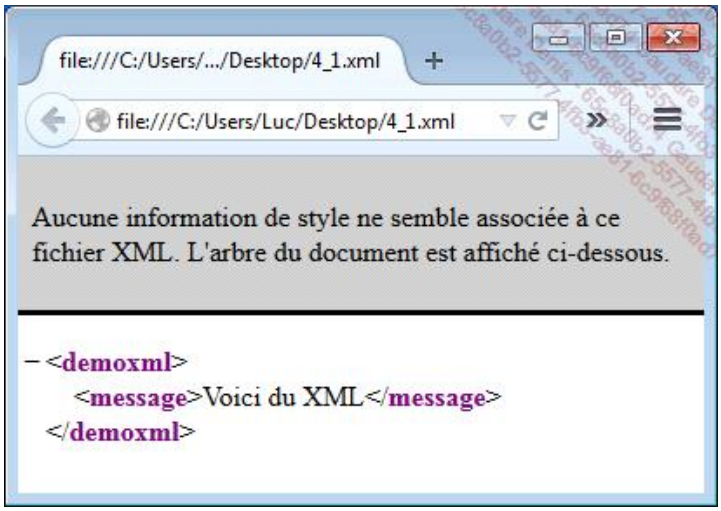
Si les navigateurs n'avaient pas de difficulté pour afficher les balises prédéfinies du HTML comme les <h1>,
 ou autres <table>, que doivent-ils faire avec vos balises du style <membre> ou <nouveau> ? Le XML a comme vocation de décrire de l'information et non pas d'afficher celle-ci. Ainsi le XML, pourtant créé en 1999, est resté durant quelques années un concept plutôt théorique, faute de navigateurs pour en afficher les résultats. Avec le développement de nouvelles techniques comme le XSL (*Extended Stylesheet Language* ou langage de feuilles de style extensible), il est devenu maintenant possible de percevoir concrètement les énormes potentialités de ce nouveau langage.

En conclusion, si le HTML a été conçu pour afficher de l'information, le XML a été créé pour structurer de l'information. À lui seul, il ne fait rien d'autre !

Voici un premier exemple de XML.

```
<?xml version="1.0"?>
<demoxml>
<message>Voici du XML</message>
</demoxml>
```

Ce qui, affiché dans Firefox, fournit le résultat suivant :



Pas de quoi pavoiser sur le plan esthétique... Mais le XML n'est finalement que de l'information structurée, encodée entre des balises. Il faudra d'autres éléments, comme une déclaration de style CSS ou un fichier XSL, pour que le navigateur puisse "comprendre" vos balises et afficher ce fichier sous une forme plus conviviale.

Le XML et le HTML

Lorsque les techniques de publication sur le Web sont abordées, une comparaison entre le HTML, le XHTML et le XML s'impose. Au contraire de ce qui a déjà été écrit par ailleurs, le XML n'est pas le successeur du HTML. Le XML, le XHTML et le HTML sont des langages distincts !

1. Une seule similitude : le SGML

Le seul point commun entre le HTML et le XML est qu'ils sont issus tous deux de la même "mère", soit le SGML (*Standardized Generalized Markup Language*), qui est le langage de référence en milieu professionnel pour tout ce qui concerne la gestion électronique des documents. Ils sont donc, tous deux, des langages de balises (voir le *Markup Language*). Ils ont également des caractéristiques communes héritées du SGML, qui consistent à transporter sur le Web des données en mode texte (*plain text*), compatibles avec n'importe quelle plateforme logicielle.

2. Les différences entre le HTML et le XML

Le HTML et le XML diffèrent sur de très nombreux points, dont certains ont trait à l'essence même du langage.

- Le XML décrit, structure, échange des données tandis que le HTML a pour vocation d'afficher des données.
- Le XML est un générateur de langages (métalangage). Le HTML est un langage normalisé de publication sur le Web.
- Le XML est extensible. Il permet de créer ses propres balises en fonction des données traitées. En HTML, les balises sont prédéfinies et elles sont donc figées.
- Le XML est un langage strict dont l'écriture doit être rigoureuse. La syntaxe du HTML est devenue très (trop ?) permissive à cause des navigateurs récents.

La syntaxe du XML

Le XML impose des règles de syntaxe très strictes et très spécifiques par rapport au HTML. On retrouvera ces mêmes règles de syntaxe dans tous les langages dérivés du XML, dont le XHTML.

Passons ces règles en revue :

- Le XML est un langage de balises. Mais au contraire du HTML où les balises sont définies, vous devez ou pouvez inventer vos propres balises. Rappelez-vous, le XML est eXtensible. Il faut donc définir soi-même le nom des balises utilisées.

  Il y a quelques règles pour la composition des noms (mais elles ne dérouteront pas les habitués du JavaScript) :

  - Les noms peuvent contenir des lettres, des chiffres ou d'autres caractères.
  - Les noms ne peuvent débuter par un nombre ou un signe de ponctuation.
  - Les noms ne peuvent commencer par les lettres xml (ou XML ou Xml...).
  - Les noms ne peuvent contenir des espaces.
  - La longueur des noms est libre mais on conseille de rester raisonnable.
  - On évitera certains signes qui pourraient prêter à confusion, comme le tiret, le point virgule, le point, la virgule, le signe plus grand que (>) et le signe plus petit que (<).
  - Les balises sont sensibles aux majuscules et minuscules (*case sensitive*). Ainsi, la balise `<Message>` est différente de la balise `<message>`. La balise d'ouverture et la balise de fermeture doivent donc être identiques. Ainsi par exemple `<Message> ... </message>` est incorrect et `<message> ... </message>` est correct.

  Un consensus se dégage pour n'écrire les balises qu'en minuscules, limitant ainsi les erreurs possibles.

- Toute balise ouverte doit être impérativement fermée.

  Les mauvaises pratiques du HTML, avec lesquelles on pouvait dans certains cas omettre la balise de fin, comme pour le paragraphe `<p>` ou l'élément de liste `<li>`, sont révolues.

  Ainsi en HTML, ce qui suit est affiché correctement :

  ```
  <p>
  <ul>
  <li>Point 1
  <li>Point 2
  ```

  Le XML est beaucoup plus strict. On devrait avoir :

  ```
  <p>
  <ul>
  <li>Point 1</li>
  <li>Point 2</li>
  </ul>
  </p>
  ```

  Les éventuelles balises uniques, appelées aussi balises vides, comme `<br>`, `<meta>` ou `<img>` en HTML, doivent également comporter un signe de fermeture soit balise/. Ainsi une balise `<meta/>` serait correcte en XML.

- Les balises doivent être correctement imbriquées. Le XML attachant beaucoup d'importance à la structure des données, des balises mal imbriquées sont des fautes graves de sens.

  Ainsi, l'écriture suivante est incorrecte car les balises ne sont pas bien imbriquées : `<parent><enfant>Marine</parent></enfant>`

  L'écriture correcte avec une bonne imbrication des éléments est : `<parent><enfant>Marine</enfant></parent>`

- Tout document XML doit comporter une racine.

  En fait, la première paire de balises d'un document XML sera considérée comme la balise de racine (*root*).

Par exemple :

```
<racine>
... suite du document XML ...
</racine>
```

En comparaison avec le HTML, l'élément racine est `<html> ... </html>`.

Tous les autres éléments sont imbriqués entre ces balises de racine.

Par exemple :

```
<racine>
<parents>
<enfants>
<petits_enfants> ... </petits_enfants>
</enfants>
</parents>
</racine>
```

- Les valeurs des attributs doivent toujours être mises entre guillemets. Le XML peut, comme le HTML, avoir des attributs comportant des valeurs. En XML, les valeurs des attributs doivent obligatoirement être reprises entre guillemets, à l'inverse du HTML où leur présence ou leur absence n'a plus beaucoup d'importance pour les navigateurs.

  Ainsi, l'écriture suivante est incorrecte car il manque les guillemets.

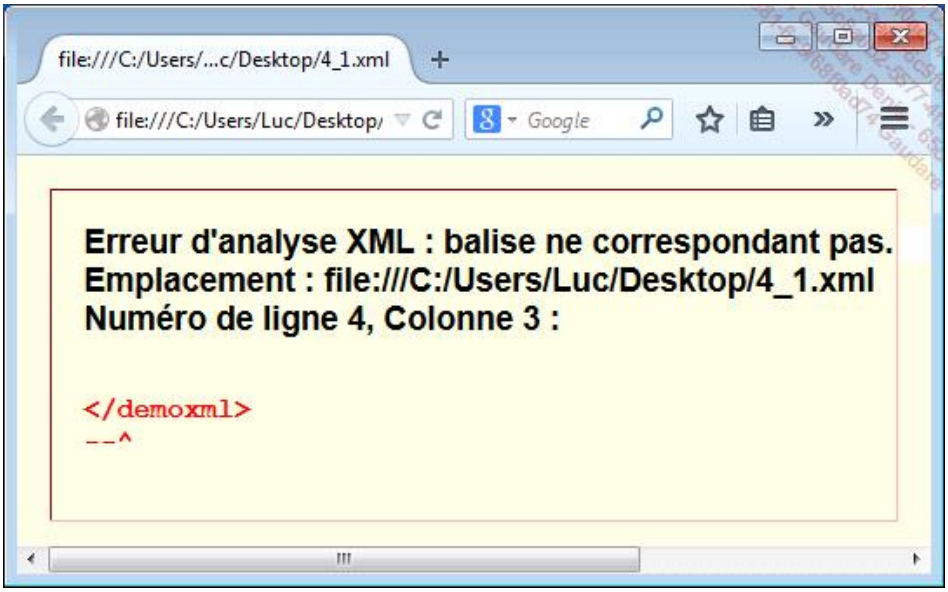
```
<enfant date_anniversaire=071185>
```

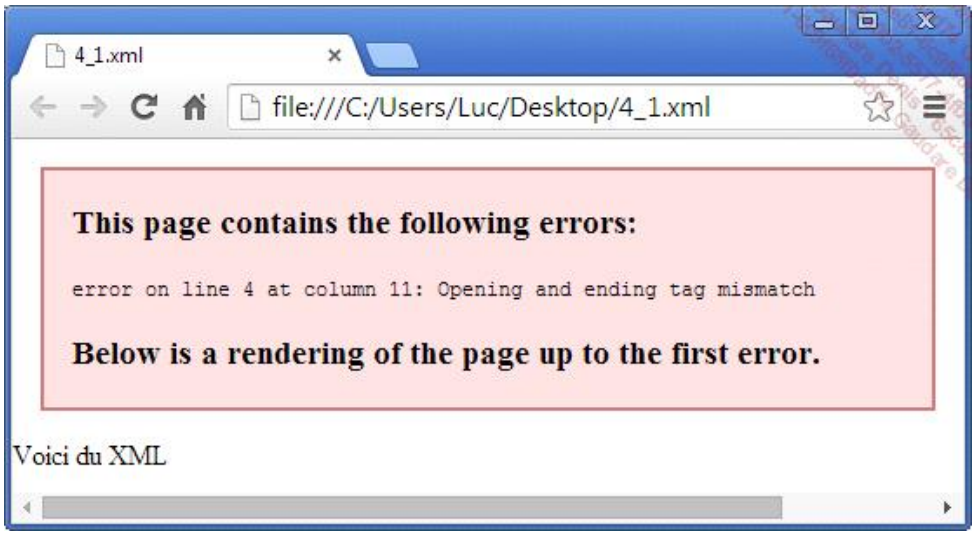
  La bonne écriture est :

```
<enfant date_anniversaire="071185">
```

- Le XML est un langage strict. Votre document doit impérativement respecter la syntaxe du XML. On dit alors que le document est **bien formé** (*Well-formed*). Seuls les documents "bien formés" sont affichés correctement. À la moindre erreur de syntaxe, le document n'est pas affiché !

  Voici ce qui est affiché dans Firefox et Google Chrome lorsqu'un document XML est incorrect.





Les navigateurs sont aussi un outil de débogage rudimentaire du code XML.

Un premier document XML

Voici un premier document XML qui sera étoffé en cours de chapitre.

```
<?xml version="1.0" encoding="UTF-8"?>
```

La déclaration `<?xml version="1.0"?>` indique au navigateur que ce qui suit est un document XML selon sa version 1.0. Vous remarquerez que cette balise ne comporte pas de signe de fermeture car cette balise n'est pas encore du XML.

Le jeu de caractères à utiliser est généralement notifié à l'intention de l'interpréteur XML (*Parser*). Le jeu de caractères UTF-8 a l'avantage d'accepter la plupart des lettres avec des accents et le signe euro. De façon native, le XML est conçu pour le jeu de caractères UTF-8.

```
<racine>
```

L'élément racine indispensable au XML. Vous pouvez utiliser, à votre convenance, n'importe quel nom pour cette balise de racine.

```
... suite du document XML ...
```

Votre document XML proprement dit, qui respecte bien entendu scrupuleusement la syntaxe du XML (bien formé).

```
</racine>
```

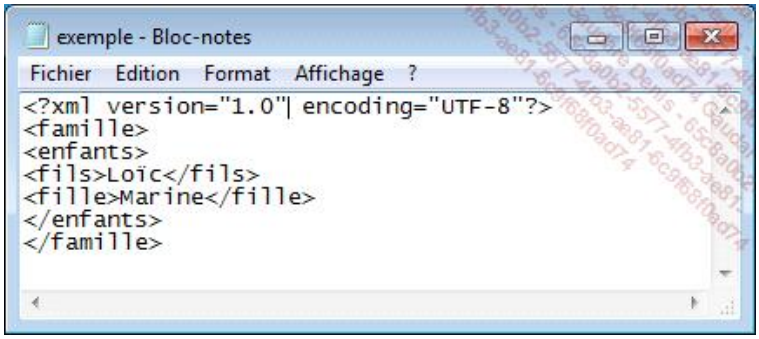
Le document XML se termine obligatoirement par la fermeture de la balise de racine.

Exemple

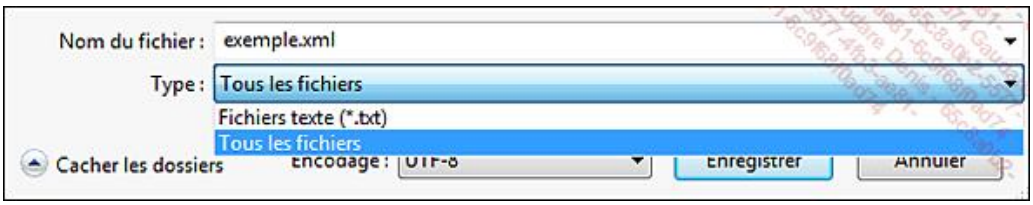
Voici un petit fichier XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<famille>
<enfants>
<fils>Loïc</fils>
<fille>Marine</fille>
</enfants>
</famille>
```

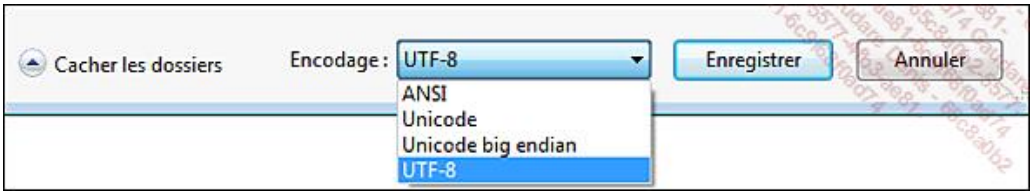
On le reproduit dans le programme Bloc-notes (*Notepad*) pour les utilisateurs de Windows.



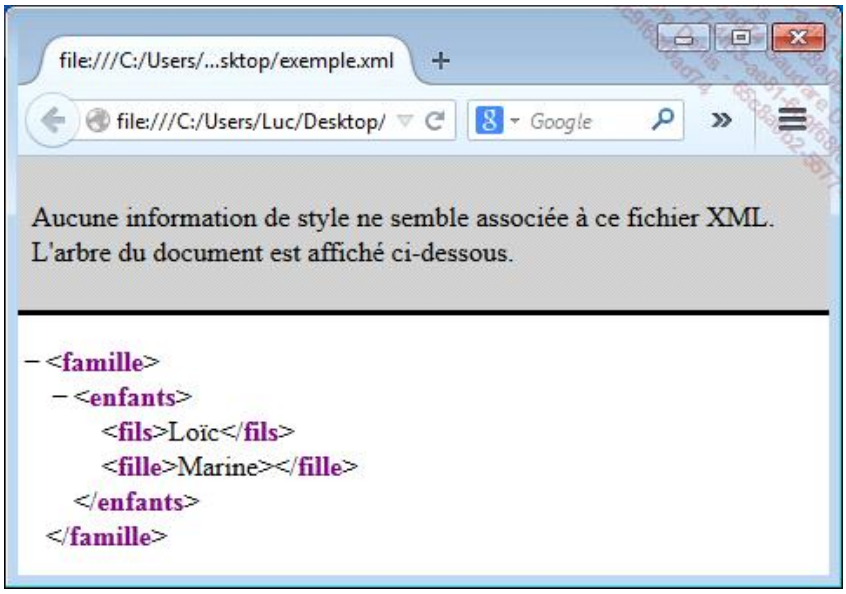
Et on l'enregistre en **Type : Tous les fichiers** sous un nom avec une extension .xml.



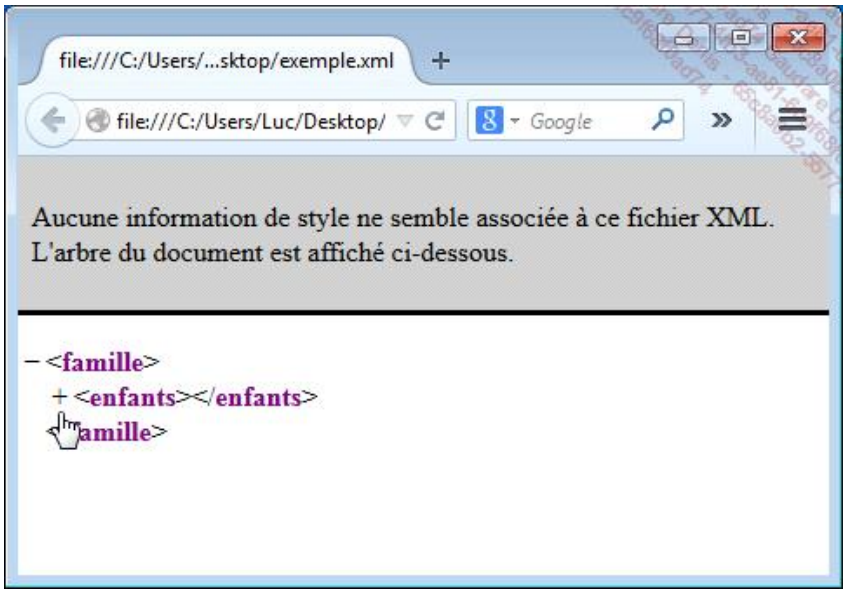
Et l'on n'oublie pas l'encodage en UTF-8.



Résultat dans Firefox :

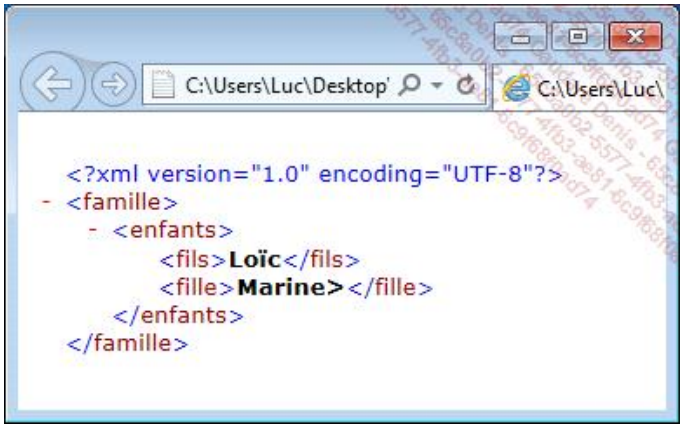


Vous remarquerez qu'il y a un petit signe **moins** affiché devant des balises englobantes. Il suffit de cliquer sur le signe pour masquer celles-ci, et bien entendu, de cliquer sur le signe **plus** pour les faire réapparaître.



Firefox fait remarquer qu'il n'y a aucune information de style associée au fichier XML afin de l'afficher valablement. Ainsi seule la structure du document (appelée aussi l'arbre) peut être affichée.

Résultat sous Internet Explorer :



Le DOCTYPE

Le DOCTYPE ou DTD (*Document Type Definition*) est l'ensemble des règles et des propriétés que doit suivre le document XML produit. Ces règles définissent généralement le nom et le contenu de chaque balise, ainsi que le contexte dans lequel elles doivent exister. Cette formalisation des éléments est particulièrement utile lorsqu'on utilise de façon récurrente des balises dans un document XML, ou que plusieurs personnes sont appelées à travailler sur le même document XML.

Les DTD sont écrites selon les prescriptions du SGML.

L'étude détaillée des DTD, dépasse de loin le cadre de cet ouvrage, mais un aperçu est cependant utile, surtout pour comprendre le fonctionnement des langages dérivés du XML (comme le XHTML), qui ne manquent pas d'utiliser ces fameux DTD.

Dans certains cas, plusieurs concepteurs peuvent se mettre d'accord pour utiliser un DTD commun pour échanger leurs données. C'est le cas du XHTML, où le DOCTYPE est signalé dans l'en-tête du document (strict, transitional ou frameset).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Cependant, dans la plupart des applications XML, le concepteur doit écrire sa propre définition de document.

1. Le DTD interne

On peut inclure le DOCTYPE dans le code du fichier XML. On parlera alors d'un DTD interne.

Un DTD interne suit la syntaxe suivante :

```
<!DOCTYPE élément-racine [
déclaration des composants
]>
```

Du point de vue des DTD, tous les documents XML sont élaborés avec les composants suivants :

- Les éléments, ou plus communément les balises. Exemple : la balise `<body>` en HTML.
- Les attributs qui viennent compléter les balises. Les attributs sont toujours inclus dans la balise ouvrante.
  Exemple : l'attribut **border** de la balise `<table>` en XHTML.
- Les PCDATA (*parsed character data*), chaînes de caractères qui seront affichées par l'interpréteur XML (*Parser*). Exemple : le contenu de la balise `<p>Chapitre 1 : Le XHTML</p>` en XHTML.
- Les CDATA (*character data*), chaînes de caractères qui ne sont pas prises en compte et qui ne sont donc pas affichées par le navigateur XML. Exemple, du code JavaScript dans un document XHTML.
- Les entités, sortes de raccourcis qui permettent de déclarer plusieurs paramètres utilisables en appelant simplement l'entité plus loin dans le document.

Détaillons ci-après quelques règles pour définir ces différents composants.

a. La déclaration d'un élément

Un élément se déclare dans la DTD selon la syntaxe suivante :

```
<!ELEMENT nom_de_l'élément mot-clé>
```

ou

```
<!ELEMENT nom_de_l'élément (contenu)>
```

Exemple

`<!ELEMENT h1>` pour la balise `<h1>` en XHTML.

b. Les éléments vides

Un élément vide (*empty*) se déclare :

```
<!ELEMENT nom_de_l'élément EMPTY>
```

Exemple

`<!ELEMENT img EMPTY>` pour la balise `<img>` en HTML.

c. Les éléments comprenant des caractères à afficher

Les éléments comprenant des caractères à afficher, généralement le contenu des balises, se notent :

```
<!ELEMENT nom_de_l'élément (#PCDATA)>
```

Exemple

`<!ELEMENT p (#PCDATA)>` pour la balise de paragraphe `<p>` en HTML.

d. Les éléments avec des éléments enfant

```
<!ELEMENT nom_de_l'élément (elément_enfant,élément_enfant,...)>
```

Exemple

```
<!ELEMENT <enfants (fils,fille)>
```

Lorsque des éléments enfant sont déclarés, ces éléments enfant doivent apparaître dans le même ordre dans le document XML.

En outre, les éléments enfant doivent être déclarés.

```
<!ELEMENT <enfants (fils,fille)>
<!ELEMENT fils (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
```

e. Les éléments avec une seule occurrence

Les éléments enfant qui doivent apparaître seulement une fois dans le code de l'élément parent se notent :

```
<!ELEMENT nom_de_l'élément (elément_enfant)>
```

Exemple

```
<!ELEMENT adhérent (nom)>
```

L'élément enfant `nom` doit apparaître une fois et seulement une fois dans l'élément parent `adhérent`.

f. Les éléments avec une ou plusieurs occurrences

Les élément qui peuvent apparaître plusieurs fois se déclarent :

```
<!ELEMENT nom_de_l'élément (elément_enfant+)>
```

Exemple

```
<!ELEMENT adhérent (telephone+)>
```

Le signe + déclare que l'élément enfant `telephone` doit apparaître une fois mais peut aussi apparaître plusieurs fois dans l'élément parent `adhérent`.

g. Les éléments avec zéro, une ou plusieurs occurrences

Les éléments facultatifs qui peuvent ne pas apparaître (zéro occurrence) ou apparaître plusieurs fois se notent :

```
<!ELEMENT nom_de_l'élément (elément_enfant*)>
```

Exemple

```
<!ELEMENT adhérent (telephonefixe*)>
```

Le signe * déclare que l'élément enfant `telephonefixe` peut ne pas apparaître, tout comme il peut apparaître plusieurs fois dans l'élément `adhérent`.

h. Les éléments avec zéro ou une occurrence

Les éléments qui peuvent apparaître qu'une fois ou être absents se définissent :

```
<!ELEMENT nom_de_l'élément (elément_enfant?)>
```

Exemple

```
<!ELEMENT adhérent (adresseIP?)>
```

Le signe ? déclare que l'élément enfant `adresseIP` peut être absent ou apparaître une seule fois dans l'élément parent `adhérent`.

i. Les éléments alternatifs

Lorsque des options (le ou logique) sont autorisées dans les éléments enfants, celles-ci se déclarent :

```
<!ELEMENT nom_de_l'élément ((elément_enfant|élément_enfant))>
```

Exemple

```
<!ELEMENT telephone ((national|international))>
```

L'exemple déclare que l'élément parent `telephone` peut contenir l'élément enfant `national` ou l'élément enfant `international`.

Appliquons une DTD interne à notre fichier XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<famille>
<enfants>
<fils>Loïc</fils>
<fille>Marine</fille>
</enfant>
</famille>
```

Celui-ci devient :

```
<?xml version="1.0" "standalone="yes"?>
<!DOCTYPE famille [
<!ELEMENT famille (enfants?)>
<!ELEMENT enfants (fils*, fille*)>
<!ELEMENT fille (#PCDATA)>
<!ELEMENT fils (#PCDATA)>
]>
<famille>
<enfants>
<fils>Loïc</fils>
<fille>Marine</fille>
</enfants>
</famille>
```

Commentaires :

- Lorsqu'un DOCTYPE interne est défini, il faut déclarer que le fichier est indépendant (`standalone="yes"`) dans le prologue XML.
- L'élément racine est déclaré dans le début du DOCTYPE (`<!DOCTYPE famille`).
- Le contenu du DOCTYPE est encodé entre des crochets ouvrants et fermants.
- La ligne `<!ELEMENT famille (enfants?)>` déclare que la balise `<famille>` contient la balise `<enfants>`. Le signe ? prévoit le cas des familles sans enfants.
- La ligne `<!ELEMENT enfants (fils*, fille*)>` déclare que la balise `<enfants>` contient les

balises `<fils>` et `<fille>`. Le signe * est prévu pour le cas où une descendance n'est composée que d'un ou plusieurs fils ainsi que d'une ou plusieurs filles.

- Le DOCTYPE se termine par le signe >.

2. Le DTD externe

Le DTD externe suit la syntaxe suivante :

```
<!DOCTYPE élément-racine SYSTEM "nom_du_fichier.dtd">
```

Le même fichier que ci-dessus est alors :

```
<!DOCTYPE famille SYSTEM "parent.dtd">
<?xml version="1.0" standalone="no"?>
<famille>
<enfants>
<fils>Loïc</fils>
<fille>Marine</fille>
</enfants>
</famille>
```

Le fichier de DTD externe (ici dans le même répertoire) `parent.dtd` contient :

```
<!ELEMENT famille (enfants?)>
<!ELEMENT enfants (fils*, fille*)>
<!ELEMENT fille (#PCDATA)>
<!ELEMENT fils (#PCDATA)>
```

Mais il est aussi possible de faire référence à un DTD externe situé sur un autre site comme pour, par exemple, le XHTML :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Un document est dit valide s'il respecte les règles spécifiques de son DOCTYPE.

Il existe une autre méthode pour définir les DTD, non plus en SGML mais en XML, c'est le XML Schema.

Le XML Schema est un langage de description de format de document (*XML Schema Description*), encodé en XML, permettant de définir la structure d'un document XML. Ce fichier de description de structure tient le rôle de DTD et permet également de valider le document XML.

Afficher le XML avec CSS

Pour afficher les balises XML, on peut faire appel aux feuilles de style CSS, maintenant classiques dans le paysage de la publication sur le Web. Pour chaque balise "inventée" dans le fichier XML, un élément de style va être intégré pour l'affichage par le navigateur.

Voici un exemple des possibilités des feuilles de style CSS associées à un document XML.

Soit notre document XML de départ :

```
<?xml version="1.0" encoding="UTF-8"?>
<compilation>
<mp3>
<titre>Sarbacane</titre>
<artiste>Francis Cabrel</artiste>
<date>1990</date>
<editeur>Columbia Tristar</editeur>
</mp3>
<mp3>
<titre>Nickel</titre>
<artiste>Alain Souchon</artiste>
<date>1991</date>
<editeur>Virgin France</editeur>
</mp3>
<mp3>
<titre>>Sheller en solitaire</titre>
<artiste>William Sheller</artiste>
<date>1992</date>
<editeur>Mercury</editeur>
</mp3>
<mp3>
<titre>Caché derrière</titre>
<artiste>Laurent Voulzy</artiste>
<date>1993</date>
<editeur>Ariola</editeur>
</mp3>
</compilation>
```

Affiché dans le navigateur, ce code XML s'affiche comme suit :

Un fichier CSS (xmlcss.css) est ajouté, voici son contenu :

```
compilation , mp3 {}
titre { display: block;
        width: 250px;
        font-size: 12pt;
        font-family: Arial;
        font-weight: bold;
        background-color: #9cf;
        color: black;
        padding-left: 10px;}
artiste { display: block;
          font-size: 12pt;
          padding-left: 10px;}
date { display: block;
       font-size: 12pt;
       color: red ;
       font-weight: bold;
       padding-left: 10px;}
editeur { display: block;
          font-size: 11pt ;
          font-style: italic;
```

```
        font-family: Arial ;
        padding-left: 10px;}
```

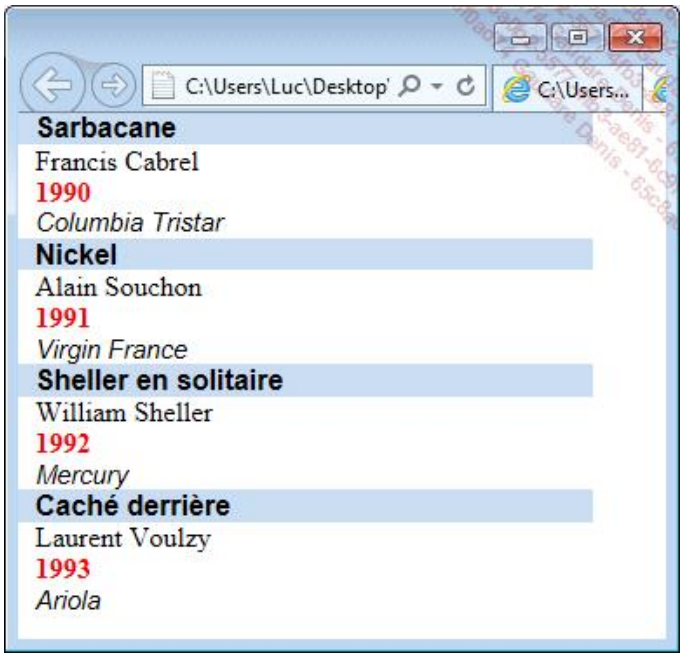
Après avoir ajouté un lien vers le fichier css dans le fichier XML :

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

Le code complet devient :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="xmlcss.css" type="text/css"?>
<compilation>
<mp3>
<titre>Sarbacane</titre>
<artiste>Francis Cabrel</artiste>
<date>1990</date>
<editeur>Columbia Tristar</editeur>
</mp3>
<mp3>
<titre>Nickel</titre>
<artiste>Alain Souchon</artiste>
<date>1991</date>
<editeur>Virgin France</editeur>
</mp3>
<mp3>
<titre>Sheller en solitaire</titre>
<artiste>William Sheller</artiste>
<date>1992</date>
<editeur>Mercury</editeur>
</mp3>
<mp3>
<titre>Caché derrière</titre>
<artiste>Laurent Voulzy</artiste>
<date>1993</date>
<editeur>Ariola</editeur>
</mp3>
</compilation>
```

On obtient finalement dans le navigateur :



Assez efficace, non ? Mais il y a encore un autre moyen, plus performant et ayant des possibilités plus étendues : afficher du XML avec le XSL soit le langage de feuilles de style eXtensible : le pendant du XML aux feuilles de style CSS.

Afficher le XML avec CSS

Pour afficher les balises XML, on peut faire appel aux feuilles de style CSS, maintenant classiques dans le paysage de la publication sur le Web. Pour chaque balise "inventée" dans le fichier XML, un élément de style va être intégré pour l'affichage par le navigateur.

Voici un exemple des possibilités des feuilles de style CSS associées à un document XML.

Soit notre document XML de départ :

```
<?xml version="1.0" encoding="UTF-8"?>
<compilation>
<mp3>
<titre>Sarbacane</titre>
<artiste>Francis Cabrel</artiste>
<date>1990</date>
<editeur>Columbia Tristar</editeur>
</mp3>
<mp3>
<titre>Nickel</titre>
<artiste>Alain Souchon</artiste>
<date>1991</date>
<editeur>Virgin France</editeur>
</mp3>
<mp3>
<titre>>Sheller en solitaire</titre>
<artiste>William Sheller</artiste>
<date>1992</date>
<editeur>Mercury</editeur>
</mp3>
<mp3>
<titre>Caché derrière</titre>
<artiste>Laurent Voulzy</artiste>
<date>1993</date>
<editeur>Ariola</editeur>
</mp3>
</compilation>
```

Affiché dans le navigateur, ce code XML s'affiche comme suit :

Un fichier CSS (xmlcss.css) est ajouté, voici son contenu :

```
compilation , mp3 {}
titre { display: block;
        width: 250px;
        font-size: 12pt;
        font-family: Arial;
        font-weight: bold;
        background-color: #9cf;
        color: black;
        padding-left: 10px;}
artiste { display: block;
          font-size: 12pt;
          padding-left: 10px;}
date { display: block;
       font-size: 12pt;
       color: red ;
       font-weight: bold;
       padding-left: 10px;}
editeur { display: block;
          font-size: 11pt ;
          font-style: italic;
```

```
        font-family: Arial ;
        padding-left: 10px;}
```

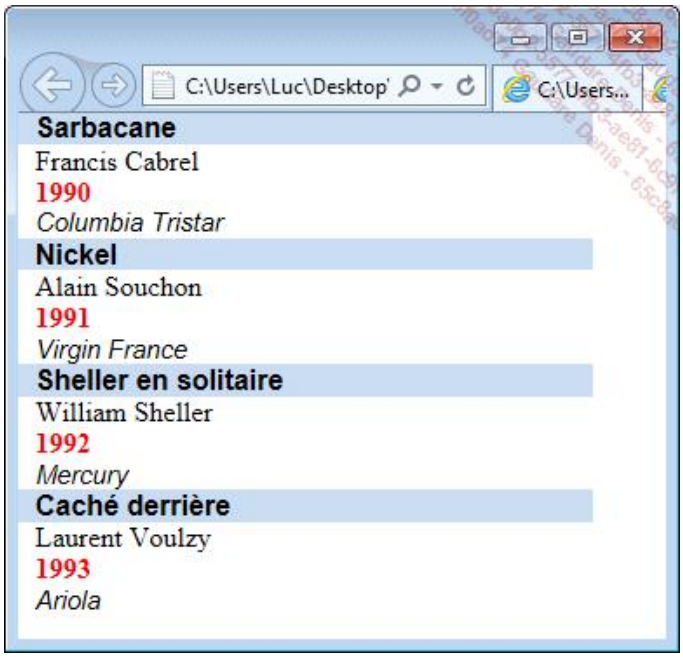
Après avoir ajouté un lien vers le fichier css dans le fichier XML :

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

Le code complet devient :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="xmlcss.css" type="text/css"?>
<compilation>
<mp3>
<titre>Sarbacane</titre>
<artiste>Francis Cabrel</artiste>
<date>1990</date>
<editeur>Columbia Tristar</editeur>
</mp3>
<mp3>
<titre>Nickel</titre>
<artiste>Alain Souchon</artiste>
<date>1991</date>
<editeur>Virgin France</editeur>
</mp3>
<mp3>
<titre>Sheller en solitaire</titre>
<artiste>William Sheller</artiste>
<date>1992</date>
<editeur>Mercury</editeur>
</mp3>
<mp3>
<titre>Caché derrière</titre>
<artiste>Laurent Voulzy</artiste>
<date>1993</date>
<editeur>Ariola</editeur>
</mp3>
</compilation>
```

On obtient finalement dans le navigateur :



Assez efficace, non ? Mais il y a encore un autre moyen, plus performant et ayant des possibilités plus étendues : afficher du XML avec le XSL soit le langage de feuilles de style eXtensible : le pendant du XML aux feuilles de style CSS.