L'Altruiste : Le guide des langages Web

Le langage XML

Sommaire

1/**Introduction**
2/**Le prologue**
2.1/La déclaration du jeu de caractères
3/**Les commentaires**
4/**Les instructions de traitement**
5/**La Définition de Type de Document**
6/**Les DTD publiques**
7/**Les éléments**
7.1/La déclaration des éléments
8/**Les sections CDATA**
9/**Les attributs**
9.1/La déclaration des attributs
9.1.1/Le type d'attribut énumération
9.1.2/Le type d'attribut ID
9.1.3/IDREF et IDREFS
9.1.4/ENTITY et ENTITIES
9.1.5/NMTOKEN et NMTOKENS
9.2/L'attribut xml:space
9.3/L'attribut xml:lang
10/**Les espaces de noms (namespace)**
11/**La déclaration des entités**
11.1/Les entités internes
11.1.1/Les entités générales internes
11.1.2/Les entités paramètres internes
11.2/Les entités externes
11.2.1/Les entités générales externes
11.2.2/Les entités paramètres externes
11.3/Les entités analysables
11.4/Les entités non-analysables
11.5/Les entités prédéfinies
12/**Les notations**
13/**Les sections conditionnelles**
14/**Documents bien formés et valides**

1 / Introduction

Le XML (eXtended Markup Language) est un langage de balisage extensible standardisé par le World Wide Web Consortium (W3C) qui s'occupe également de la standardisation du langage HTML et des feuilles de style entre autres.

Le XML et les technologies associées comme le XSL (eXtended StyleSheet Language) seront certainement dans l'avenir, les successeurs désignés du langage HTML. Effectivement, plusieurs éditeurs de logiciels à l'image de Sun Microsytems ou de Microsoft travaillent activement à l'élaboration de nouveaux outils profitant pleinement des avantages du XML.

Le HTML permet de mettre en forme un document contenant diverses données formatées en titres, en paragraphes, en listes, en tableaux, etc. Ainsi, le langage HTML possède des balises destinées essentiellement à la présentation des données que le développeur lui fournit et en aucun cas, il ne tente de les interpréter.

Par contre, les balises du XML définissent plutôt la sémantique (le sens) des données. C'est-à-dire, que le balisage créé par le développeur donnera une signification précise des données fournies.

Par exemple, une liste de noms et de prénoms d'employés dans une entreprise quelconque sera balisé comme suit par du HTML :

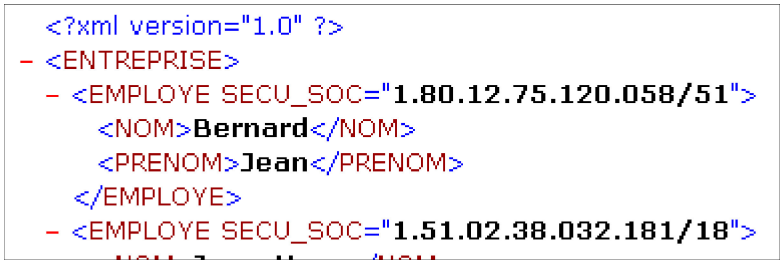
```
<ul>
 <li>Jean Bernard</li>
 <li>Jean-Yves Dupré<li>
 ...
</ul>
```

Alors que dans le XML, le balisage adoptera une autre forme plus adaptée aux données :

```
<ENTREPRISE>
 <EMPLOYE SECU_SOC="1.80.12.75.120.058/51">
 <NOM>Bernard</NOM>
 <pRENOM>Jean</pRENOM>
 </EMPLOYE>
 <EMPLOYE SECU_SOC="1.51.02.38.032.181/18">
 <NOM>Dupré</NOM>
 <pRENOM>Jean-Yves</pRENOM>
 </EMPLOYE>
 ...
</ENTREPRISE>
```

Comme nous pouvons le constater, le XML est une structure arborescente dont les noeuds de l'arbre contiennent des données. Dans l'exemple ci-dessus, le noeud principal est *<ENTREPRISE>* et des noeuds secondaires sont représentés par *<EMPLOYE>*. On remarquera que ce dernier possède un attribut *SECU_SOC* qui représente une donnée complémentaire et permet de différencier les noeuds secondaires entre eux.

Dans un navigateur compatible avec cette technologie comme Internet Explorer 5, le résultat à l'affichage est le suivant :



```
    </EMPLOYE>
</ENTREPRISE>
```

C'est pourquoi, pour afficher correctement ses données à l'instar du HTML, le XML a besoin du langage des feuilles de style, le CSS (Cascading Style Sheet) ou spécifiquement pour lui le XSL (eXtended StyleSheet Language). Le XSL est une grammaire du XML, un processeur de modèle qui transforme une grammaire XML en une autre ou en HTML.

Voici quelques éditeurs XML :

- XMLSpy est un très bon éditeur XML édité par Altova Inc.,
- Cooktop est un éditeur XML, XSLT, XPath et DTD puissant et totalement gratuit,
- EditiX est un éditeur XML complet réalisé en France.
- XMetal est un logiciel puissant et flexible pour la création de documents XML,
- XMLWriter est un puissant éditeur XML,
- XMLMind représente un très bon éditeur XML,
- JXMLEditor est un éditeur XML écrit en Java,
- KXML Editor est un éditeur XML pour KDE.

2 / Le prologue

Le prologue XML est une instruction de traîtement servant à identifier la version du langage XML et doit se trouver obligatoirement tout en haut du document XML.

Cette instruction est utilisée également pour **déclarer le jeu de caractères d'encodage** du document XML.

Enfin, elle permet de **spécifier si le document est autonome** (standalone="yes") ou s'il dépend, pour son fonctionnement, d'autres fichiers ou de toutes autres ressources externes (standalone="no").

Les instructions XML légales sont les suivantes :

```
<?xml attributs ?>
<?XML ATTRIBUTS ?>
```

Ces instructions possèdent trois attributs :

Instruction	Description
version="numéro-de-version"	indique la version de XML utilisée pour le traîtement du document.
encoding="type-d'encodage"	indique le jeu de caractères employé dans le document.
standalone="yes|no"	indique si le document est autonome ou s'il se réfère à d'autres fichiers.

exemple :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

L'instruction XML spécifie la version 1.0 de XML avec un encodage *Unicode compressé* et requiert des documents externes.

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

La déclaration suivante indique toujours la version 1.0, d'ailleurs la seule, un encodage correspondant au type *Latin-1, Europe occidentale* et enfin le document est autonome.

2.1 / La déclaration du jeu de caractères

Les documents XML doivent toujours être identifiés par un jeu de caractères approprié au langage dans lequel le document doit être rédigé.

La déclaration de l'encodage du document doit se situer dans la déclaration XML préalable par l'intermédiaire de l'attribut *encoding*.

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

Si la déclaration XML ne comporte pas d'attribut *encoding*, alors l'encodage par défaut sera le jeu de caractères Unicode compressé.

Encodage	Description
US-ASCII	Anglais
UTF-8	Unicode compressé (par défaut)
UTF-16	UCS compressé
ISO-10646-UCS-2	Unicode brut
ISO-10646-UCS-4	UCS Brut
ISO-8859-1	Latin-1, Europe occidentale
ISO-8859-2	Latin-2, Europe orientale
ISO-8859-3	Latin-3, Europe méridionale
ISO-8859-4	Latin-4, Europe septentrionale
ISO-8859-5	ASCII plus cyrillique
ISO-8859-6	ASCII plus arabe
ISO-8859-7	ASCII plus grec
ISO-8859-8	ASCII plus hébreu
ISO-8859-9	Latin-5, turc
ISO-8859-10	Latin-6, ASCII plus langues nordiques
ISO-8859-11	ASCII plus thaï
ISO-8859-13	Latin-7, ASCII plus langues baltes
ISO-8859-14	Latin-8, ASCII plus gallos et gaëllique
ISO-8859-15	Latin-9, Latin-10, Europe occidentale
ISO-2022-JP	Japonais
ISO-2022-CN	Chinois
KOI6-R	Russe
ISO-2022-KR	Coréen

Pour plus de renseignements sur les différents types d'encodage, consultez les standards Unicode à l'adresse suivante :

http://www.unicode.org

La liste officel des jeux de caractères disponibles se trouvent sur le site de l'INIA (Internet Assigned Numbers Authority) :

http://www.iana.org/assignments/character-sets

3 / Les commentaires

Les commentaires permetttent d'illustrer le code afin de le rendre compréhensible pour une meilleure efficacité dans la maintenance d'un site.

Les commentaires peuvent être placés n'importe où dans un document excepté à l'intérieur des balises. La définition de type de document peut contenir ces commentaires aux endroits autorisés par la grammaire.

Les commentaires dans XML ont la même forme que ceux du HTML.

```
<!-- commentaire -->
```

Les informations textuelles contenues à l'intérieur de ce balisage ne sont évidemment pas affichées à l'écran.

En outre, un processeur XML peut permettre à une application de récupérer le texte des commentaires.

La chaîne double trait d'union (--) ne doit pas apparaître à l'intérieur de commentaires en raison d'une incompatibilité.

```
<!----------commentaire---------->
```

Cette écriture est interdite car elle comporte deux séries de double trait d'union.

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<!--/-L'esperluette & peut être utilisée littéralement dans une instruction CDATA-/-->
<[CDATA[L'esperluette ou le & commercial]]>
<balise>
<!-- Explication -->
  Données
</balise>
```

4 / Les instructions de traitement

Une instruction de traitement permet de transmettre le type d'application et éventuellement différents paramètres permettant de traiter une entité externe non-analysable comme une vidéo (avi, mpeg, qtw, etc.), un document textuel (pdf, doc, rtf, etc.) un programme Java, Perl ou encore C++.

```
<?cible liste_attributs?>
```

La *cible* identifie l'application à laquelle l'instruction est destinée.

La liste d'attributs permet de spécifier des données particulières pour le traitement de l'instruction.

Les instructions de traitement débutant par la chaîne de caractères *xml* sont réservés à un usage réservé au standard XML.

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

Exemple :

```
<?tableau feuille="compte.xsl"?>
<?bml register demos.texteditor.MenuBarAdder?>
<?xml-stylesheet type="text/css" href="style.css"?>

<?xml version="1.0" standalone="no"?>
<!DOCTYPE document [
<!ELEMENT document (article)>
<!ELEMENT article (#PCDATA)>
<!ATTLIST article source ENTITY #REQUIRED>
<!ENTITY doc_word SYSTEM "/doc/article.doc" NDATA doc>
<!NOTATION doc PUBLIC
          "-//IETF//NONSGML Media Types application/msword//EN"
          "http://www.isi.edu/in-notes/iana/assignments
           /media-types/application/msword">
]>
<?doc WinWord?>
<document>
  <article source="doc_word">
  Spécification de XML 1.0
  </article>
</document>
```

5 / La Définition de Type de Document

Une DTD permet de déclarer la liste, le type et les relations des éléments, des attributs, des entités et des notations contenus dans le document XML.

La DTD contraint le document XML à se conformer aux règles énoncées. Si le document est conforme au DTD, alors ce document XML est considéré comme *valide*.

Chaque élément, attribut et entité doivent être déclarés dans la DTD avant leur utilisation.

De plus, la structure adoptée dans la DTD doit correspondre rigoureusement à celle construite dans le corps du document XML.

```
<!DOCTYPE élément_racine [
  <!ELEMENT balise (#PCDATA)>
  <!ATTLIST balise attribut CDATA #REQUIRED>
  <!ELEMENT élément_racine (balise)>
  <!ENTITY e-aigu "&eacute;">
]>
<élément_racine>
  <balise attribut="valeur">donn&e-aigu;e</balise>
</élément_racine>
```

La définition s'énonce à la suite de la déclaration XML préalable par la commande suivante :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE Elément_racine...>
<Elément_racine>
...
</Elément_racine>
```

Cette définition se compose du nom de l'élément racine, suivi de la déclaration de type de document soit sous forme d'une énumération des éléments, attributs et entités, soit sous forme d'une URI (Uniform Resource Identifier) pointant vers une DTD externe auquelle est associée un mot-clé : *SYSTEM* ou *PUBLIC* combiné avec le nom public de la DTD externe.

```
<!DOCTYPE Elément_racine [Enumération...]>
<!DOCTYPE Elément_racine SYSTEM "URI">
<!DOCTYPE Elément_racine PUBLIC "Nom_DTD" "URI">
```

Il est également possible de combiner une DTD externe à une DTD interne. Ainsi, la DTD du document XML se composera d'une sous-DTD externe et d'une sous-DTD interne.

```
<!DOCTYPE Elément_racine SYSTEM "URI" [Enumération...]>
```

Il existe, donc, deux mots-clé pour cette instruction :

Mot-clé	Description
SYSTEM	indique la localisation d'une URI d'une DTD utilisée dans le document.
PUBLIC	est employé lorsque la DTD possède un usage général.

La valeur diffère selon le mot-clé spécifié.

Valeur	Description

"URI"	permet d'indiquer une adresse d'une DTD lors de la spécification du mot-clé *SYSTEM*.
"Nom_DTD" "URI"	représente, lors d'une spécification *PUBLIC*, le nom de la DTD que le processeur XML utilisera pour retrouver la définition du type de document et une adresse permettra de se substituer au nom de la DTD précitée en cas d'impossibilité d'être atteinte.

Exemple :

```
<!DOCTYPE document SYSTEM "../dtd/definition.dtd">

<!DOCTYPE document
                PUBLIC
                "-//NomDTD//dtD//EN"
                "http://www.monsite.com/dtd/definition.dtd">

<!DOCTYPE document [
  <!ELEMENT paragraphe (phrase)>
  <!ELEMENT phrase (#PCDATA)>
  <!ATTLIST paragraphe auteur CDATA #IMPLIED>
  <!ENTITY e-dans-o "&oelig;">
  <!ENTITY a-dans-e "&aelig;">
]>
```

6 / Les DTD publiques

Lorsque la spécification d'une DTD dans un document XML nécessite une ressource externe, en l'occurrence un fichier de Déclaration de Type de Document, il est nécessaire de l'appeler par l'intermédiaire de son nom et par son URI (Uniform Resource Identifier).

Cette dernière est employée si le nom de la DTD ne suffit pas pour être atteinte par le processeur XML.

```
<!DOCTYPE Elément_racine PUBLIC "Nom_DTD" "URI">
```

Le nom de la DTD externe obéit à une formulation précise :

- si la DTD appartient au standard ISO, alors le nom commencera par *ISO*,
- autrement, le nom de la DTD débutera par un signe plus (+),
- si elle n'a pas été approuvé, le nom devra commencer par un signe moins (-),
- ensuite, deux barres obliques interviennent suivi par le nom du propriétaire,
- à nouveau deux barres obliques précèdent un descripteur de la DTD,
- enfin, un bigramme représentant la langue est séparé par deux nouvelles barres obliques.

```
ISO|+|-//nom_propriétaire//dtD Descripteur//Bigramme_langue
```

Exemple :

```
-//Jean Frédéric//dtD Librairie//FR
-//W3C//DTD HTML 4.0 Transitional//EN
```

7 / Les éléments

Les éléments XML sont des balises particulières à l'instar de celles du langage HTML, hormis que dans ce cas, elles sont le fruit de l'auteur.

Tous les Eléments doivent être déclarées dans la Définition de Type de Document (DTD).

Les balises XML peuvent être de deux sortes.

La première contient, entre la balise de début et la balise de fin, des données diverses comme du texte.

```
<Elément>donnée</Elément> <Elément/>
```

La seconde est du type vide, c'est-à-dire, des balises ne contenant pas de données à proprement parler à l'instar des balises HTML *<img...>* ou *
*. Dans ce cas, contrairement au HTML, les éléments doivent être impérativement fermées par une balise de fin ou le caractère slash (/) placé juste avant le caractère *supérieur à* (>).

```
<Elément></Elément>
<Elément/>
```

En effet, le balisage XML doit respecter des règles strictes comme la fermeture obligatoire de tous les types d'éléments comme précité.

```
<centre>
  <image src="http://monsite.com/photo.jpg">
</centre>
```

Cet exemple ne pourra pas fonctionner car la balise image n'est pas fermée.

De plus, Les éléments doivent être correctement imbriquées sous peine de dysfonctionnements.

```
<tableau>
  <gras><souligne>Titre</gras></souligne>
</tableau>
```

Dans cet exemple, les éléments *gras* et *souligne* sont incorrectement imbriquées.

Le nom d'un élément doit commencer par une lettre (a-zA-Z et les caractères accentués), le caractère de soulignement (_). Le nom peut comporter un nombre quelconque des caractères précités auxquels s'ajoutent les chiffres (0-9), le tiret (-), le point (.), les caractère deux points (:). Ce dernier est en général utilisé comme séparateur entre un espace de nom (namespace) et le nom d'élément.

```
<2main/> <!--Balise incorrecte-->
<&variable/> <!--Balise incorrecte-->
<premier Pas/> <!--Balise incorrecte-->
<départ/> <!--Balise correcte-->
<_depart/> <!--Balise correcte-->
<football:milieu/> <!--Balise correcte-->
<fin/> <!--Balise correcte--> <coord.y/> <!--Balise correcte--> <POINT/> <!--Balise correcte-->
```

Pour en savoir plus sur la construction des noms, consultez le site du World Wide Web Consortium (W3C).

De même, un nom d'élément ne peut commencer par la chaîne de caractère *xml*, *XML* ou toutes autres combinaisons formant ce dernier, réservée pour un usage spécifique.

Les noms d'éléments XML sont sensibles à la casse.

```
<balise>
<balise>
<bALISE>
```

Les trois balises ci-dessus sont toutes strictement différentes. Le processeur XML interprètera chacun de ces éléments de façon unique, si bien qu'ils seront incompatibles entre eux.

```
<balise>donnée</balise>
```

Cet exemple provoquera une erreur puisque les balises de début et de fin ne sont pas équivalentes.

7.1 / La déclaration des éléments

Les éléments, y compris l'élément racine, apparaissant dans le document XML doivent être au préalable définis dans la Déclaration de Type de Document (DTD).

Cette déclaration permet de spécifier un nom et éventuellement un contenu ou un type de contenu à l'élément concerné.

Un processeur XML peut émettre un avertissement lorsqu'une déclaration mentionne un élément pour lequel aucune déclaration n'a été fournie, toutefois cela ne constitue pas une erreur rédhibitoire.

L'instruction *ELEMENT* permet de déclarer les éléments composant le document XML.

```
<!ELEMENT Nom_Elément Spécifications>
```

Les spécifications de la déclaration comportent plusieurs valeurs possibles.

Valeur	Description
ANY	indique que n'importe quelle donnée peut être contenue par l'élément XML.
EMPTY	permet de déclarer un élément XML vide.
(#PCDATA)	signifie que l'élément XML n'acceptera que des données textuelles analysées (Parsed Character DATA) soit tout types de caractères hormis les balises d'éléments XML.
(Elément,..., ElémentN)	spécifie l'inclusion d'un ou plusieurs éléments enfants au sein des balises de l'élément concerné.
(Elément|Elément2)	signifie l'inclusion du premier élément enfant ou du second au sein des balises de l'élément concerné.

Les éléments enfants peuvent faire l'objet de récurrences à l'intérieur de l'élément parent.

Indicateur	Description
(Elément?)	implique l'apparition de l'élément une seule fois ou pas du tout (0-1) dans l'élément parent.
(Elément+)	implique l'apparition de l'élément une ou plusieurs fois (1-N) dans l'élément parent.
(Elément*)	implique l'apparition de l'élément plusieurs fois ou pas du tout (0-N) dans l'élément parent.

La répétition d'un même élément enfant dans un élément parent permet de contraindre l'élément conteneur à acceuillir le nombre d'itérations précitées de l'élément enfant.

```
<!ELEMENT Parent (Enfant, Enfant, Enfant)
```

Dans ce cas, le nombre d'itérations de *Elément_enfant* dans *Elément_parent* est de trois. Alors l'élément parent devra accueillir trois et seulement trois éléments enfants pour se conformer à la déclaration.

Exemple :

```
<?xml version="1.0" standalone="yes"?<
<!DOCTYPE bibliothèque [
  <!ELEMENT bibliothèque (livre)>
  <!ELEMENT livre (titre)>
  <!ELEMENT titre (#PCDATA)>
]>
<bibliothèque>
  <livre>
  <titre>La bible XML</titre>
  </livre>
</bibliothèque>
```

La DTD déclare un élément *livre* contenant un élément enfant *titre* composé de données textuelles analysées. Par ailleurs, l'élément racine *bibliothèque* doit contenir l'élément *livre*. L'élément racine doit faire l'objet d'un déclaration complémentaire afin de lui donner ces propres caractéristiques.

```
<?xml version="1.0" standalone="yes"?<
<!DOCTYPE bibliothèque [
  <!ELEMENT bibliothèque ANY>
  <!ELEMENT livre (titre,numero,couverture?)>
  <!ELEMENT collection (titre,numero)>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT numéro (#PCDATA)>
  <!ELEMENT couverture EMPTY>
]>
<bibliothèque>
  La bibliothèque de l'université contient :
  <livre>
  <titre>La bible XML</titre>
  <numéro>XML0023</numéro>
  <couverture/>
  </livre>
  <collection>
  <titre>XML, par la pratique</titre>
  <numéro>XML1023</numéro>
  </collection>
</bibliothèque>
```

L'élément racine *bibliothèque* peut tout contenir du texte ou encore des éléments enfants. L'élément *livre* doit contenir trois éléments enfants : *titre*, *numero* et couverture.

Ce dernier étant associé à un point d'interrogation peut apparaître une seule fois ou pas du tout. Par ailleurs, l'élément *couverture* est déclaré comme élément vide (EMPTY).

```
<?xml version="1.0" standalone="yes"?<
<!DOCTYPE bibliothèque [
  <!ELEMENT bibliothèque (livre+,collection*)>
  <!ELEMENT livre (titre,numero,couverture?)>
  <!ELEMENT collection (titre,numero)>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT numéro (#PCDATA)>
]>
<bibliothèque>
  <livre>
  <titre>La bible XML</titre>
  <numéro>XML0001</numéro>
  <couverture/>
  </livre>
  <livre>
  <titre>La programmation XML</titre>
  <numéro>XML0023</numéro>
  </livre>
  <collection>
  <titre>XML, par la pratique</titre>
  <numéro>XML1023</numéro>
  </collection>
</bibliothèque>
```

L'élément parent *bibliothèque* doit être composé d'au moins un élément *livre* et de zéro à plusieurs collections.

```
<?xml version="1.0" standalone="yes"?<
<!DOCTYPE bibliothèque [
  <!ELEMENT bibliothèque (livre+,index?)>
  <!ELEMENT livre ((titre,numero)?|volume+)>
  <!ELEMENT volume (titre,numero)>
  <!ELEMENT index (lettre+)>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT numéro (#PCDATA)>
  <!ELEMENT lettre (#PCDATA)>
]>
<bibliothèque>
  <index>
  <lettre>H</lettre>
  <lettre>C</lettre>
  <lettre>X</lettre>
  </index>
  <livre>
  <titre>La bible XML</titre>
  <numéro>XML0001</numéro>
  </livre>
  <livre>
  <volume>
     <titre>XML, par la pratique</titre>
     <numéro>XML1023</numéro>
  </volume>
  </livre>
</bibliothèque>
```

L'élément racine *bibliothèque* peut contenir de aucun à plusieurs éléments *livre*, au moins une *volume* et de zéro ou un *index*.

L'élément *livre* doit contenir un *titre* et un numéro ou (|) au moins un seul *volume*.

8 / Les sections CDATA

Les sections CDATA sont utilisées pour habiller des blocs de texte contenant des caractères qui seraient autrement reconnus comme balisage.

Les sections CDATA sont identifiées de la manière suivante :

```
<![CDATA[ bloc de texte ]]>
```

Ces sections peuvent se trouver à n'importe quel endroit acceptable pour des données textuelles dans le document XML.

En outre, il n'est pas utile d'employer des références d'entités < et & pour dissimuler des caractères spéciaux tels que < et >, à l'intérieur d'une section CDATA, puisque seule la chaîne de caractères *]]>* est reconnue comme balisage de fin et donc qu'il n'y a aucun risque de confusion avec le balisage du document XML.

Enfin, les sections CDATA ne peuvent pas s'imbriquer.

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<racine>
  <![CDATA[Voici un élément vide
        correctement fermé : <balise/>.]]>
  <element>
      <![CDATA[Inutile de faire cela
            &lt;balise/&gt;.]]>
  </element>
</racine>
```

9 / Les attributs

Les attributs apportent des informations complémentaires à une balise XML.

Les attributs des éléments XML doivent être déclarés dans la Définition de Dype de Document (DTD) avant leur utilisation dans le document XML.

A l'image de ceux du HTML, les attributs sont formés d'un nom auquel est associé une valeur.

```
<livre edition="EditeurX" auteur="IndividuY">Titre</livre>
```

Dans cet exemple, deux attributs sont associés à l'élément *livre*, lui apportant deux caractéristiques supplémentaires.

Les attributs ne peuvent être présents que dans la balise de début de l'élément XML.

```
<balise attribut="valeur">...</balise>
<balise attribut="valeur"/>
```

La valeur des attributs doivent toujours être entre des guillemets doubles ("...") ou simples ('...').
Les attributs *pays='France'* et *pays="France"* sont tous les deux corrects.

Les attributs sont également sensibles à la casse de caractères (mahuscule/minuscule).
L'attribut *monnaie="francs"* est différent de *MONNAIE="francs"*.

Le nom d'un attribut doit commencer par une lettre (a-zA-Z et les caractères accentués), le caractère de soulignement (_) ou le caractère deux points (:) utilisé généralement pour des attributs réservés.
Ainsi, l'attribut *poids='120kg'* est correct alors que *#poids='72kg'* est invalide.

Enfin, une balise ne peut commencer par la chaîne de caractère *xml* ou *XML* réservée pour un usage spécifique, en l'occurrence pour des attributs réservés comme *xml:lang* et *xml:space*.

9.1 / La déclaration des attributs

Les attributs des éléments XML doivent être définis dans le DTD (Definition de Type de Document).

L'instruction *ATTLIST* permet de déclarer les attributs.

```
<!ATTLIST NomElément NomAttribut TypeAttribut ValeurDefaut>
```

Le langage XML supporte la déclaration multiple d'attributs sur plusieurs lignes à partir d'une seule instruction *ATTLIST*.

```
<!ATTLIST Elément Nom_Attribut Type_Attribut Valeur_Defaut
                  Nom2_Attribut Type_Attribut Valeur_Defaut
                  Nom3_Attribut Type_Attribut Valeur_Defaut>
```

Le type de l'attribut renseigne sur le type de données de l'attribut.

Type	Description
CDATA	indique l'utilisation de données textuelles ne comprenant pas de balises XML.
(Valeur|...|ValeurN)	déclare une liste de valeurs à utiliser.
ENTITY	correspond à une entité déclarée dans la DTD.
ENTITIES	correspond à la déclaration de plusieurs entités séparées par des espaces blancs dans la DTD.
ID	identifie des éléments d'une manière unique.
IDREF	correspond à la valeur d'un attribut ID.
IDREFS	correspond aux valeurs de plusieurs attributs ID séparés par un espace blanc.
NMTOKEN	invoque un token, un nom XML composé de lettres, de chiffres, soulignés, tirets, points et deux-points.
NMTOKENS	invoque plusieurs tokens XML séparés par des espaces blancs.
NOTATION	correspond à une notation déclarée dans la DTD.

La valeur par défaut permet d'initialiser l'attribut.

Valeur	Description
#REQUIRED	signifie que la valeur d'attribut est requise pour l'élément XML.
#IMPLIED	indique que la valeur de l'attribut peut ne pas être spécifiée.
#FIXED "valeur"	permet de fixer la valeur de l'attribut.
"valeur"	initialise la valeur de l'attribut.

```
<!ATTLIST livre auteur CDATA "nom">
<!ATTLIST livre gencode ID #REQUIRED>
<!ATTLIST porte ouvert (true|false) "true">
<!ATTLIST rugby point (essai|pénalité|drop|transformation) #IMPLIED>
```

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE liste [
<!ELEMENT commentaire (#PCDATA)>
<!ELEMENT editeur (#PCDATA)>
<!ATTLIST editeur adresse CDATA #REQUIRED>
<!ELEMENT liste (logiciel+)>
<!ELEMENT logiciel (nom, commentaire, editeur, prix)>
<!ATTLIST logiciel id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ATTLIST nom
    langue CDATA #REQUIRED
    systeme_exploitation CDATA #REQUIRED>
<!ELEMENT prix (#PCDATA)>
<!ATTLIST prix monnaie CDATA #REQUIRED>
]>
<liste>
 <logiciel id="CTP0124555">
  <nom langue="US" systeme_exploitation="Win">
   Cooktop 2.200
  </nom>
  <commentaire>
   Un editeur XML, XSLT, XPath et DTD puissant et totalement gratuit.
  </commentaire>
  <editeur adresse="http://xmleverywhere.com/cooktop/">
   XML Everywhere
  </editeur>
  <prix monnaie="$US">00.00</prix>
 </logiciel>
 <logiciel id="XSY325684">
  <nom langue="US" systeme_exploitation="Win">
   XML Spy 4.1
  </nom>
  <commentaire>
   Un editeur XML desormais mature.
  </commentaire>
  <editeur adresse="http://www.xmlspy.com/default.html">
   Altova Inc.
  </editeur>
  <prix monnaie="$US">199,00</prix>
 </logiciel>
 <logiciel id="XWR387795">
  <nom langue="US" systeme_exploitation="Win">
   XMLwriter v1.21
  </nom>
  <commentaire>
   Permet de creer des documents XML.
  </commentaire>
  <editeur adresse="http://xmlwriter.net/">
   Wattle Software
  </editeur>
  <prix monnaie="$US">75,00</prix>
 </logiciel>
</liste>
```

9.1.1 / Le type d'attribut énumération

Le type dans une déclaration d'attribut peut prendre la forme d'une liste de valeurs autorisées.

Ainsi, l'attribut de l'élément concerné pourra prendre pour valeur, l'une des valeurs énumérées dans la liste.

D'ailleurs, une des valeurs de la liste doit être indiquée par défaut entre des guillemets.

```
<!ATTLIST élément attribut (valeur|...|valeurN) "valeur_défaut">

<!ELEMENT etat_civil (#PCDATA)>
<!ATTLIST etat_civil sexe (homme|femme) "homme">

<!ELEMENT image EMPTY>
<!ATTLIST image visible (true|false) "true">

<!ELEMENT date EMPTY>
<!ATTLIST date mois (1|2|3|4|5|6|7|8|9|10|11|12) "1">
```

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier DTD : logitheque.dtd -->
<!ELEMENT categorie (logiciel+)>
<!ATTLIST categorie  nom CDATA #REQUIRED>
<!ELEMENT commentaire (#PCDATA)>
<!ELEMENT editeur (#PCDATA)>
<!ATTLIST editeur lien CDATA #REQUIRED>
<!ELEMENT logiciel (nom, commentaire, editeur, prix)>
<!ATTLIST logiciel
  id (02135844 | 10235684 | 13413558 | 13414170 | 13414421) #IMPLIED
  plateforme (Tous | Win | Linux | Unix | Mac) #REQUIRED
  support (CD | DISK | DVD) #REQUIRED
  disponibilite (non | oui) "oui"
  langue (FR | US) #REQUIRED
  code CDATA #IMPLIED
>
<!ELEMENT logitheque (categorie+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prix (#PCDATA)>
<!ATTLIST prix monnaie CDATA #REQUIRED>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE logitheque SYSTEM "logitheque.dtd">
<logitheque>
  <categorie nom="Edition Web">
    <logiciel
                id="13414170"
                plateforme="Win"
                support="CD"
                disponibilite="non"
                langue="FR">
      <nom>ShopFactory Pro v4.5</nom>
      <commentaire>Le commerce ...</commentaire>
      <editeur lien="http://www.3d3.com/">3D3</editeur>
      <prix monnaie="FRF">1 490,00</prix>
    </logiciel>
    <logiciel
                id="13414421"
                plateforme="Win"
                support="CD"
                disponibilite="oui"
                langue="FR">
      <nom>Ukanweb</nom>
      <commentaire>Votre site Web ...</commentaire>
      <editeur lien="">Ukantoo</editeur>
      <prix monnaie="FRF">925,00</prix>
    </logiciel>
    <logiciel
                id="13413558"
                plateforme="Win"
                support="DVD"
                disponibilite="oui"
                langue="FR">
      <nom>Ultra Edit 32 v8.0 (25-49 postes )</nom>
      <commentaire>Pour 25 à 49 utilisateurs.</commentaire>
      <editeur lien="http://www.3d3.com/">3D3</editeur>
      <prix monnaie="FRF">200,00</prix>
    </logiciel>
  </categorie>
  <categorie nom="Outils XML">
    <logiciel
                id="10235684"
                plateforme="Tous"
                support="CD"
                disponibilite="non"
                langue="US">
      <nom>Xalan 1.1</nom>
      <commentaire>Un processeur XSL ...</commentaire>
      <editeur lien="http://xml.apache.org/xalan-c/index.html">
            Apache Soft. Found.
          </editeur>
      <prix monnaie="$US">00,00</prix>
```

```
    </logiciel>
    <logiciel
            code="52136498"
            plateforme="Tous"
            support="CD"
            disponibilite="non"
            langue="US">
      <nom>Xerces 1.5.0</nom>
      <commentaire>Un analyseur syntaxique XML ...</commentaire>
      <editeur lien="http://xml.apache.org/xerces-c/index.html">
          Apache Soft. Found.
        </editeur>
      <prix monnaie="$US">00,00</prix>
    </logiciel>
    <logiciel
            id="02135844"
            plateforme="Tous"
            support="DISK"
            disponibilite="oui"
            langue="US">
      <nom>Sabblotron 0.60</nom>
      <commentaire>Un processeur XSL ...</commentaire>
      <editeur lien="http://www.gingerall.com/">
          Ginger Alliance
        </editeur>
      <prix monnaie="$US">00,00</prix>
    </logiciel>
  </categorie>
</logitheque>
```

9.1.2 / Le type d'attribut ID

Le type d'attribut ID permet d'associer à un élément un identificateur unique.

Cette méthode permet de mettre en correspondance des éléments entre eux à l'instar d'une base de données.

La valeur d'un attribut ID doit être un nom XML valide, c'est-à-dire, composé de lettres, de chiffres, de soulignés, de tirets, de points ou de deux-points.

De plus, cette valeur doit être évidemment unique. Dans le cas contraire, le processeur XML renverra une erreur d'analyse lorsqu'il rencontrera un second identificateur identique.

Par conséquent, le mot clé de valeur implicite *#FIXED* est incompatible avec le type d'attribut *id*.

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE magasin [
  <!ELEMENT magasin (service+)>
  <!ELEMENT service (produit)>
  <!ATTLIST service id ID #REQUIRED>
  <!ELEMENT produit (#PCDATA)>
  <!ATTLIST produit code ID #REQUIRED>
]>
<magasin>
  <service id="A001">
    <produit code="4DE205">
      Soupe
    </produit>
    <produit code="2TM206">
      Condiment
    </produit>
    <produit code="3KJ227">
      Conserve
    </produit>
  </service>
  <service id="A003">
    <produit code="1OU152">
      Lessive
    </produit>
    <produit code="8AH070">
      Essui-tout
    </produit>
  </service>
  <service id="D301">
    <produit code="240M002">
      Video
    </produit>
    <produit code="210K333">
      DVD
    </produit>
  </service>
  <service id="K853">
    <produit code="8KL025">
      Pantalon
    </produit>
    <produit code="9M2569">
      Robe
    </produit>
  </service>
</magasin>
```

9.1.3 / IDREF et IDREFS

Le type *IDREF* permet à une valeur d'attribut de faire référence à l'identificateur (ID) d'un autre élément.

De cette manière, il est possible de relier des éléments entre eux, à l'instar des clés d'enregistrements, dans une base de données, permettant de relier des tables entre elles.

IDREFS permet d'associer plusieurs identificateurs (ID) en les séparant par des espaces blancs dans la valeur d'un attribut.

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE société [
<!ELEMENT division EMPTY>
<!ATTLIST division
          id ID #REQUIRED
          services IDREFS #REQUIRED
>
<!ELEMENT divisions (division+)>
<!ELEMENT employe (nom, prenom)>
<!ATTLIST employe
          id ID #REQUIRED
          service IDREF #REQUIRED
>
<!ELEMENT employes (employe+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT service (nom)>
<!ATTLIST service id ID #REQUIRED>
<!ELEMENT services (service+)>
<!ELEMENT societe (services, divisions, employes)>
]>
<societe>
 <services>
  <service id="DG001">
   <nom>Direction générale</nom>
  </service>
  <service id="ST001">
   <nom>Service technique</nom>
  </service>
  <service id="SC001">
   <nom>Service commercial</nom>
  </service>
  <service id="SC101">
   <nom>Service clientèle</nom>
  </service>
  <service id="RH001">
   <nom>Ressources humaines</nom>
  </service>
  <service id="SQ001">
   <nom>Service qualité</nom>
  </service>
  <service id="SP001">
   <nom>Service production</nom>
  </service>
  <service id="RD001">
   <nom>Recherche et développement</nom>
  </service>
 </services>
 <divisions>
  <division id="A001" services="DG001 RH001"/>
  <division id="B001" services="SC001 SC101"/>
  <division id="C001" services="SP001 SQ001 RD001"/>
 </divisions>
 <employes>
  <employe id="RJ1002" service="DG001">
   <nom>Robierre</nom>
   <prenom>Jean</prenom>
  </employe>
  <employe id="LA1012" service="DG001">
   <nom>Lardut</nom>
   <prenom>Anne</prenom>
  </employe>
  <employe id="GA1013" service="ST001">
   <nom>Guilde</nom>
   <prenom>Angélique</prenom>
  </employe>
  <employe id="HP1022" service="SC001">
   <nom>Henry</nom>
   <prenom>Paul</prenom>
  </employe>
  <employe id="MM1045" service="RH001">
   <nom>Mortier</nom>
   <prenom>Marc</prenom>
```

```
    </employe>
    <employe id="LS1102" service="SQ001">
      <nom>Lebreton</nom>
      <prenom>Sophie</prenom>
    </employe>
    <employe id="JM1095" service="RD001">
      <nom>Jolie</nom>
      <prenom>Martine</prenom>
    </employe>
    <employe id="MT1036" service="SC101">
      <nom>Marcelin</nom>
      <prenom>Tania</prenom>
    </employe>
    <employe id="LL1029" service="SC101">
      <nom>Léger</nom>
      <prenom>Laurence</prenom>
    </employe>
    <employe id="DM1052" service="SC001">
      <nom>Duroi</nom>
      <prenom>Maxime</prenom>
    </employe>
  </employes>
</societe>
```

9.1.4 / ENTITY et ENTITIES

***ENTITY* permet de faire référence à une entité générale non-analysable déclarée dans la Définition de Type de Document (DTD).**

***ENTITIES* autorise la référence à plusieurs entités générales non-analysables, également déclarées dans la DTD.**

Une entité générale non-analysable correspond à un contenu composé de texte non-XML ou de données binaires et reste disponible à partir d'une adresse URL (Uniform resource Locator).

Ces entités sont préalablement déclarées par l'intermédiaire de l'instruction *ENTITY*.

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE site [
  <!ENTITY fichier SYSTEM "fichier.zip">
  <!ELEMENT téléchargement (#PCDATA)>
  <!ATTLIST téléchargement source ENTITY #REQUIRED>
  <!ELEMENT site (téléchargement)>
]>
<site>
  <téléchargement source="fichier">
  Cliquez ici pour télécharger le fichier
  </téléchargement>
</site>
```

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE collection [
  <!ENTITY image SYSTEM "http://www.monsite.com/image.gif">
  <!ENTITY image2 SYSTEM "http://www.monsite.com/image2.gif">
  <!ENTITY image3 SYSTEM "http://www.monsite.com/image3.gif">
  <!ENTITY image4 SYSTEM "http://www.monsite.com/image4.gif">
  <!ENTITY imageN SYSTEM "http://www.monsite.com/imageN.gif">
  <!ELEMENT album EMPTY>
  <!ATTLIST album source ENTITIES #REQUIRED>
  <!ELEMENT collection (album)>
]>
<collection>
  <album source="image image2 image3 image4 imageN"/>
</collection>
```

9.1.5 / NMTOKEN et NMTOKENS

***NMTOKEN* permet d'affecter un nom symbolique à l'attribut tel qu'un format de date, un format de fichier ou encore une abréviation.**

***NMTOKENS* offre la possibilité à la valeur de l'attribut de comporter plusieurs noms symboliques séparés par des espaces blancs.**

Les tokens sont utilisés dans l'énumération des signes des langages ou des jeux de mots clés qui satisfont ces restrictions dans la DTD.

Chaque nom symbolique ne peut être constitué que de caractères autorisés par XML, c'est-à-dire, des lettres, des chiffres, des soulignés (_), des tirets (-), des points (.) et des deux points (:).

Exemple :

```
<!ATTLIST fichier xml:lang NMTOKEN #REQUIRED 'fr'>

<!ATTLIST fichier type NMTOKENS #REQUIRED>

<!ELEMENT date EMPTY>
<!ATTLIST date année NMTOKEN #REQUIRED
              mois NMTOKEN #REQUIRED
              jour NMTOKEN #REQUIRED>

<?xml version="1.0" standalone="yes"?>
<!DOCTYPE gestionnaire [
  <!ELEMENT gestionnaire (répertoire)>
  <!ELEMENT répertoire (#PCDATA)>
  <!ATTLIST répertoire fichier NMTOKEN #REQUIRED>
]>
<gestionnaire>
  <répertoire fichier="index.htm">
  Ce fichier représente la page d'accueil de ce répertoire
  </répertoire>
</gestionnaire>

<?xml version="1.0" standalone="yes"?>
<!DOCTYPE France [
  <!ELEMENT France (région)>
  <!ELEMENT région (#PCDATA)>
  <!ATTLIST région département NMTOKENS #REQUIRED>
]>
<France>
  <région département="14 50 61">
  Basse-Normandie
  </région>
  <région département="04 05 06 13 83 84">
  Provence-Alpes-Côte d'Azur
  </région>
  <région département="75 77 78 91 92 93 94 95">
  Région Ile-de-France
  </région>
...
</France>
```

9.2 / L'attribut xml:space

L'attribut *xml:space* permet de définir le mode de traîtement des espaces blancs (espaces, tabulations, interlignes).

Deux mots-clés associés à cet attribut permettent de définir le comportement des applications envers les espaces blancs.

Mot-clé	Description
default	L'application procède à une interprétation par défaut des espaces blancs. Dans la plupart des cas, ces espaces sont ignorés.
preserve	L'application est obligée de tenir compte des espaces blancs dans le rendu final du document XML.

En fait, l'attribut *xml:space* pourrait correspondre à la balise *pre* permettant d'afficher un contenu tel qu'il a été formaté par l'auteur du document.

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE poème [
  <!ELEMENT poème (#PCDATA)>
  <!ATTLIST poème xml:space (default|preserve) "preserve">
]>
<poème xml:space="preserve">
  A quatre heures du matin, l'été,
  Le soleil d'amour dure encore.
  Sous les bocages s'évapore
  L'odeur du soir fêté.

  Là-bas, dans leur vaste chantier
  Au soleil des Hespérides,
  Déjà s'agitent - en bras de chemise -
  Les charpentiers.

  Dans leurs Déserts de mousse, tranquilles,
  Ils préparent les lambris précieux
  Où la ville
  Peindra de faux cieux.

  Ô, pour ces Ouvriers charmants
  Sujets d'un roi de Babylone
  Vénus ! quitte un instant les Amants
  Dont l'âme est en couronne.

  Ô Reine des Bergers,
  Porte aux travailleurs l'eau-de-vie,
  Que leur force soient en paix
  En attendant le bain la mer à midi.
</poème>
```

9.3 / L'attribut xml:lang

L'attribut *xml:lang* permet de définir une langue dans laquelle le contenu d'un élément est rédigé.

Cet attribut se déclare dans la DTD (Document Type Definition) par l'intermédaire de la commande suivante :

```
<!ATTLIST nom_élément xml:lang Type_attribut Valeur>
```

L'utilisation de l'attribut dans un élément s'effectue comme suit :

```
<nom_élément xml:lang="code_langue">  Données
<&nom_élément>
```

Le type d'attribut *CDATA* (Character DATA) permet de spécifier n'importe quel chaîne de caractères ne comprenant pas de balises XML, à la valeur de l'attribut *xml:lang*.

```
<!ATTLIST article xml:lang CDATA #IMPLIED> ...
<article xml:lang="Français canadien">  Texte
</article>
```

Le type d'attribut *NMTOKEN* permet d'affecter un nom symbolique à l'attribut en l'occurrence l'abréviation d'un nom de langue.

```
<!ATTLIST paragraphe xml:lang NMTOKEN 'fr'>
```

L'énumération permet de limiter les choix de langues à quelques unes parmi des centaines.

```
<!ATTLIST paragraphe xml:lang (fra|eng|deu|ita|esp) 'fra'>
```

Les codes de langues sont fournis par différents organismes.

- L'Organisation internationale de normalisation proposent le code ISO-639.
- L'INIA (Internet Assigned Numbers Authority) diffusant ces propres codes commençant par *i-*.
- Il est également possible de créer son propre jeu de codes de langues. Afin d'éviter toutes confusions ces codes seront précédés de *x-*.

Enfin, la valeur implicite peut être soit un code de langue par défaut, soit l'un des mots-clés suivants :

Valeur	Description
#REQUIRED	la valeur d'attribut est requis pour l'élément XML.
#IMPLIED	la valeur de l'attribut peut ne pas être spécifié.
#FIXED "valeur"	permet de fixer la valeur de l'attribut.

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE journal [
  <!ELEMENT recueil (poème)>
  <!ELEMENT poème (#PCDATA)>
  <!ATTLIST poème xml:lang (fra|eng) #REQUIRED>
]>
<recueil>
  <poème xml:lang="fra">
  Tout objet de beauté est une joie éternelle :
  Le charme en croît sans cesse ; jamais
  Il ne glissera dans le néant, mais il gardera toujours
  Pour nous une paisible retraite, un sommeil
  Habité de doux songes, plein de santé, et qui paisiblement respire.
  </poème>
  <poème xml:lang="eng">
  A thing of beauty is a joy for ever:
  Its loveliness increases; it will never
  Pass into nothingness; but stil will keep
  A bower quiet for us, and a sleep
  Fullof sweet dreams, and health, and quiet breathing.
  </poème>
</recueil>
```

10 / Les espaces de noms (namespace)

Les espaces de noms (namespace) permettent de regrouper des éléments XML autour d'un nom unique.

Ainsi, des éléments portant un nom identique peuvent faire partie de sources différentes, et partant, doivent pouvoir se combiner lors de fusions, ou doivent pouvoir cohabiter au sein d'un même document, tout en évitant des collisions.

Les éléments appartenant à un espace de noms se distinguent des autres éléments par l'ajout d'un préfixe symbolisant cette singularité.

Les préfixes d'espace de noms se placent au sein d'un marqueur XML, avant le nom de l'élément et séparés par le caractère deux-points (:).

Le préfixe doit être constitué de lettres (a-zA-Z et les caractères accentués), de caractères de soulignement (_), de chiffres (0-9) ou de tout autres caractères autorisés (voir W3C).

```
<Préfixe:Elément>
  donnée
</Préfixe:Elément>
```

La balise XML ainsi constituée, possède désormais un nom qualifié (*Préfixe:Elément*), qui est lui même composé d'un préfixe (*Préfixe*) et d'un nom local (*Elément*).

```
NomQalifié = xsl:stylesheet
Préfixe = xsl
NomLocal = stylesheet
```

Les espaces de noms peuvent être déclarés dans les éléments contenant les préfixes d'espace de noms ou pour plus de clarté et de commodité dans l'élément racine, par l'intermédiaire de l'attribut *xmlns*.

```
<Préfixe:Elément xmlns:Préfixe="URI">
  ...
</Préfixe:Elément>
```

L'adresse URI (Uniform Resource Identifier) permet d'associer une adresse Internet à l'espace de noms.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
```

Les espaces de noms multiples permettent d'accueillir au sein d'un élément parent, plusieurs préfixes différents.

Par exemple, les espaces de noms *xsl* associé à l'URI *http://www.w3.org/XSL/Transform/1.0* et *html* accouplé à l'URI *http://www.w3.org/TR/REC-html40*, pourraient constituer la structure d'une feuille de style (XSLT) et combiner ainsi deux types de préfixes : *xsl* et *html*.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
  <xsl:template match="/">
    <html:p xmlns:html="http://www.w3.org/TR/REC-html40"
            html:style="color:red">
      </xsl:apply-templates/>
    </html:p>
  </xsl:template>
</xsl:stylesheet>
```

Dans l'exemple ci-dessus les déclarations *xmlns* sont placées dans deux éléments différents. Il est également possible de les regrouper dans la balise parente pour un fonctionnement équivalent.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0"
                xmlns:html="http://www.w3.org/TR/REC-html40">
  <xsl:template match="/">
    <html:p html:style="color:red">
      </xsl:apply-templates/>
    </html:p>
  </xsl:template>
```

```
</xsl:stylesheet>
```

Un espace de noms implicite évite la répétition du préfixe dans les marqueurs XML tout au long du document. Sauf spécifications contraires, tous les éléments fils appartiendront au même espace de noms par défaut.

```
<stylesheet xmlns="http://www.w3.org/XSL/Transform/1.0">
  <template match="/">
    ...
  </template>
</stylesheet>
```

La déclaration de l'attribut de déclaration d'espace de noms (*xmlns*), dans la Définition de Type de Document (DTD) s'effetue comme suit :

```
<!ELEMENT Préfixe:Elément Spécifications>
<!ATTLIST Préfixe:Elément
        xmlns:Préfixe CDATA (#REQUIRED| #IMPLIED | #FIXED "URI")>

<!ELEMENT lal:balise ANY>
<!ATTLIST lal:balise
        xmlns:lal CDATA #FIXED "http://laltruiste.com">
```

Les éléments et attributs compris dans un espace de noms doivent être déclarés avec leur nom qualifié.

```
<!DOCTYPE NomQualifié [
  <!ELEMENT NomQualifié Spécification>
  <!ATTLIST NomQualifiéElement NomQualifiéAttribut Spécification>
]>
```

Les noms *xml* et *xmlns* sont interdits comme espace de noms, tout les deux faisant l'objet d'une utilisation spécifique.

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE lal:librairie [
 <!ELEMENT lal:auteur (#PCDATA)>
 <!ELEMENT lal:decription (xhtml:p)>
 <!ELEMENT lal:librairie (lal:livre+)>
 <!ATTLIST lal:librairie
      xmlns:lal CDATA #FIXED "http://www.laltruiste.com/">
 <!ELEMENT lal:livre (lal:titre, lal:auteur, lal:decription)>
 <!ATTLIST lal:livre lal:id ID #REQUIRED>
 <!ELEMENT lal:titre (#PCDATA)>
 <!ELEMENT xhtml:p (#PCDATA)>
 <!ATTLIST xhtml:p
      xmlns:xhtml CDATA #FIXED "http://www.w3.org/1999/xhtml">
]>
<lal:librairie xmlns:lal="../">
 <lal:livre lal:id="ISBN-2-7460-1969-8">
  <lal:titre>
   XML et XSL
  </lal:titre>
  <lal:auteur>
   Cyril Vincent
  </lal:auteur>
  <lal:decription>
   <xhtml:p xmlns:xhtml="http://www.w3.org/1999/xhtml">
    XML s'est imposé dans la plupart des champs de l'informatique
    moderne et est un outil de travail difficilement contournable
    pour les développeurs et les webmasters.
   </xhtml:p>
  </lal:decription>
 </lal:livre>
 <lal:livre lal:id="ISBN-2-84177-353-1">
  <lal:titre>
   XML en concentré
  </lal:titre>
  <lal:auteur>
   Elliotte Rusty Harold
  </lal:auteur>
  <lal:decription>
   <xhtml:p xmlns:xhtml="http://www.w3.org/1999/xhtml">
    XML est devenu un outil essentiel pour la création de sites
    Web dynamiques (B2C) et l'échange de données entre des
    systèmes hétérogènes sur Internet.
   </xhtml:p>
  </lal:decription>
 </lal:livre>
 <lal:livre lal:id="ISBN-2-84177-215-2">
  <lal:titre>
   XML Schema
  </lal:titre>
  <lal:auteur>
   Eric Van Der Vlist
  </lal:auteur>
  <lal:decription>
   <xhtml:p xmlns:xhtml="http://www.w3.org/1999/xhtml">
    Parmi les langages de définition de types récemment proposés
    pour succéder aux DTD, la Recommandation XML Schéma du W3C
    est certainement la plus utilisée.
   </xhtml:p>
  </lal:decription>
 </lal:livre>
</lal:librairie>
```

En savoir plus:

W3C

11 / La déclaration des entités

Les documents XML sont construits à partir d'une ou plusieurs entités internes ou externes.

Ces denières possèdent un contenu et un identifiant, en l'occurrence une valeur d'entité et un nom d'entité.

Tout document XML possède au moins une entité correspondant à la Déclaration de Type de Document et l'élément racine. Cette entité s'appelle *entité document*.

Les entités peuvent contenir des données XML bien formulées, d'autres formes de texte comme des courriers électroniques ou des données binaires comme des images ou des applets Java.

Les noms d'entités obéissent aux mêmes règles que les noms d'éléments ou d'attributs. Ils ne peuvent être constitués que de lettres, y compris les lettres accentuées, de chiffres, de soulignés (_), de tirets (-), de points (.) et de deux-points (:). Toutefois le nom d'entité ne peut commencer que par une lettre ou un souligné.

Deux catégories d'entités se distinguent. Il s'agit des **entités internes**, définies et utilisables dans le document XML, et les **entités externes**, définies dans une ressource externe et utilisable dans le document via une adresse URL (Uniform Resource Locator).

Deux autres types d'entités apparaissent également : les **entités analysables** dont le contenu est entièrement conforme à un XML bien formé, et les **entités non-analysables** correspondant à un contenu composé de texte non-XML ou de données binaires.

Enfin, les entités se divisent encore en deux parties. Les **entités générales** permettent d'utiliser leur contenu essentiellement dans le document XML hors DTD. Les **entités paramètres** sont utilisables exclusivement dans la Déclaration de Type de Document.

11.1 / Les entités internes

Les entités internes sont définies dans la DTD et utilisée dans la DTD elle-même (entités paramètres) ou dans le document XML (entités générales).

11.1.1 / Les entités générales internes

Les entités générales sont des entités analysables destinées uniquement à être utilisées dans le document.

La déclaration dans la DTD d'une entité paramètre s'effectue comme suit :

```
<!ENTITY nom "chaînes_de_caractères_de_remplacement">
```

Les références d'entités générales internes sont utilisées dans le document XML de la manière suivante :

```
&nom_entité;
```

Déclaration dans la DTD :

```
<!ENTITY onu "Organisation des Nations Unies">
```

Utilisation de l'entité dans le document :

```
<nom>&onu;</nom>
```

Les références d'entités internes générales peuvent être utilisées également dans une définition d'entité.

```
<!ENTITY un "Nations Unies">
<!ENTITY onu "Organisation des &un;">
```

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE organisation [
<!ENTITY un "Nations Unies">
<!ENTITY onu "Organisation des &un;">
<!ENTITY us "Etats-Unis D'Amérique">
<!ENTITY ru "Russie">
<!ENTITY chi "Chine">
<!ENTITY gb "Grande Bretagne">
<!ENTITY fr "France">
<!ENTITY sgm "Seconde Guerre Mondiale">
<!ENTITY ny "New York">
<!ENTITY ya "Yalta">
<!ENTITY sf "San Francisco">

<!ELEMENT description (#PCDATA)>
<!ELEMENT fondation (#PCDATA)>
<!ELEMENT membres EMPTY>
<!ATTLIST membres
        annee CDATA #REQUIRED
        nombre CDATA #REQUIRED
>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT organisation (nom, fondation, membres, description)>
]>
<organisation>
  <nom>&onu;</nom>
  <fondation>24 avril 1945</fondation>
  <membres annee="2002" nombre="191"/>
  <description>
    L'&onu; est une organisation internationale regroupant
    l'ensemble des états nations du monde, dont le siège est
    à &ny;. L'&onu; a été fondée à l'initiative des vainqueurs
    de la &sgm; suite aux conférences de &ya; (février 1945)
    et de &sf; (avril 1945). Les buts et principe de
    l'&onu; sont énumérés dans la charte des &un; qui est entrée
    en vigueur le 24 octobre 1945. L'une de ses missions principales
    consite à maintenir la paix et la sécurité internationale.De 51
    états membres à l'origine, l'&onu; comptent depuis 2002 191
    membres dont 5 permanents (&us;, &ru;, &chi;, &gb; et &fr;).
  </description>
</organisation>
```

11.1.2 / Les entités paramètres internes

Les entités paramètres sont des entités analysables destinées uniquement à être utilisées dans la DTD.

La déclaration dans la DTD d'une entité paramètre s'effectue comme suit :

```
<!ENTITY % nom "caractères_de_remplacement">
```

Les références d'entités paramètres sont utilisées dans la DTD de la manière suivante :

```
%nom_entité;
```

Déclaration dans la DTD :

```
<!ENTITY tout "ANY">
```

Utilisation de l'entité dans la DTD :

```
<!ELEMENT paragraphe %tout;>
```

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE poeme [
  <!ENTITY % def "default";
  <!ENTITY % pre "preserve";
  <!ENTITY % style "% def;|%pre;";
  <!-- Importation d'un fichier d'entités internes -->
  <!ENTITY latin SYSTEM "accents.ent">
  %accents;
  <!ELEMENT poeme (#PCDATA)>
  <!ATTLIST poeme xml:space (%style;) %pre;>
]>
<poeme xml:space="preserve">
  &Agrave; quatre heures du matin, l'&eacute;t&eacute;,
  Le soleil d'amour dure encore.
  Sous les bocages s'&eacute;vapore
  L'odeur du soir f&ecirc;t&eacute;.

  L&agrave;-bas, dans leur vaste chantier
  Au soleil des Hesp&eacute;rides,
  D&eacute;j&agrave; s'agitent - en bras de chemise -
  Les charpentiers.

  Dans leurs D&eacute;serts de mousse, tranquilles,
  Ils pr&eacute;parent les lambris pr&eacute;cieux
  O&ugrave; la ville
  Peindra de faux cieux.

  &Ocirc;, pour ces Ouvriers charmants
  Sujets d'un roi de Babylone
  V&eacute;nus ! quitte un instant les Amants
  Dont l'&acirc;me est en couronne.

  &Ocirc; Reine des Bergers,
  Porte aux travailleurs l'eau-de-vie,
  Que leur force soient en paix
  En attendant le bain la mer &agrave; midi.
</poeme>

<-- Fichier accents.ent -->
<!ENTITY Agrave "&#192;">
<!ENTITY Aacute "&#193;">
<!ENTITY Acirc "&#194;">
<!ENTITY AElig "&#198;">
<!ENTITY Ccedil "&#199;">
<!ENTITY Egrave "&#200;">
<!ENTITY Eacute "&#201;">
<!ENTITY Ecirc "&#202;">
<!ENTITY Igrave "&#204;">
<!ENTITY Iacute "&#205;">
<!ENTITY Icirc "&#206;">
<!ENTITY Ograve "&#210;">
<!ENTITY Oacute "&#211;">
<!ENTITY Ocirc "&#212;">
<!ENTITY Ugrave "&#217;">
<!ENTITY Uacute "&#218;">
<!ENTITY Ucirc "&#219;">
<!ENTITY agrave "&#224;">
<!ENTITY aacute "&#225;">
<!ENTITY acirc "&#226;">
<!ENTITY aelig "&#230;">
<!ENTITY ccedil "&#231;">
<!ENTITY egrave "&#232;">
<!ENTITY eacute "&#233;">
<!ENTITY ecirc "&#234;">
<!ENTITY igrave "&#236;">
<!ENTITY iacute "&#237;">
<!ENTITY icirc "&#238;">
<!ENTITY ograve "&#242;">
<!ENTITY oacute "&#243;">
<!ENTITY ocirc "&#244;">
<!ENTITY ugrave "&#249;">
<!ENTITY uacute "&#250;">
```

```
<!ENTITY ucirc "&#251;">
```

11.2 / Les entités externes

Les entités externes permettent d'insérer leur contenu à partir d'une ressource externe pointée par un URI (Uniform Resource Identifier).

Elles peuvent être également des *entités générales* ou des *entités paramètres* accessibles à partir de fichiers externes.

11.2.1 / Les entités générales externes

Les entités générales externes permettent de construire un document XMl à partir de plusieurs autres documents XML complémentaires.

La déclaration dans la DTD d'une entité externe s'effectue comme suit :

```
<!ENTITY nom SYSTEM "URI">
```

Les références d'entités externes sont utilisées dans le document XML de la manière suivante :

```
&nom_entité;
```

Déclaration dans la DTD :

```
<!ENTITY documentation SYSTEM "http//www.site.com/doc.xml">
```

Utilisation de l'entité dans le document XML :

```
<aide>
  &documentation;
</aide>
```

Il est possible d'utiliser cette entité externe au sein d'un attribut. Pour cela, il suffit d'inidque dans la déclaration de l'attribut que sa valeur sera du type *ENTITY* et dans le document la valeur de l'attribut devra porter le nom de l'entité externe.

```
<!ENTITY nom SYSTEM "ressource.ext">
<!ATTLIST element attribut ENTITY #REQUIRED>

<element attribut="nom"/>
```

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE organisation [
<!ENTITY nu "Nations Unies">
<!ENTITY onu "Organisation des v">
<!ENTITY us "Etats-Unis D'Amérique">
<!ENTITY ru "Russie">
<!ENTITY chi "Chine">
<!ENTITY gb "Grande Bretagne">
<!ENTITY fr "France">
<!ENTITY sgm "Seconde Guerre Mondiale">
<!ENTITY ny "New York">
<!ENTITY ya "Yalta">
<!ENTITY sf "San Francisco">
<!ENTITY liste SYSTEM "membres.txt">

<!ELEMENT description (#PCDATA)>
<!ELEMENT fondation (#PCDATA)>
<!ELEMENT etats (#PCDATA)>
<!ELEMENT membres EMPTY>
<!ATTLIST membres
        etats ENTITY #REQUIRED
        annee CDATA #REQUIRED
        nombre CDATA #REQUIRED
>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT organisation (nom, fondation, membres, etats, description)>
]>
<organisation>
 <nom>&onu;</nom>
 <fondation>24 avril 1945</fondation>
 <membres annee="2002" nombre="191" etats="liste"/>
 <etats>&liste;</etats>
 <description>
  L'&onu; est une organisation internationale regroupant
  l'ensemble des états nations du monde, dont le siège est
  à &ny;. L'&onu; a été fondée à l'initiative des vainqueurs
  de la &sgm; suite aux conférences de &ya; (février 1945)
  et de &sf; (avril 1945). Les buts et principe de
  l'&onu; sont énumérés dans la charte des &un; qui est entrée
  en vigueur le 24 octobre 1945. L'une de ses missions principales
  consite à maintenir la paix et la sécurité internationale.De 51
  états membres à l'origine, l'&onu; comptent depuis 2002 191
  membres dont 5 permanents (&us;, &ru;, &chi;, &gb; et &fr;).
 </description>
</organisation>

<-- Fichier membres.txt -->
Afrique du Sud, Arabie saoudite, Argentine, Australie, Bélarus,
Belgique, Bolivie, Brésil, Canada, Chili, Chine, Colombie, Costa Rica,
cuba, Danemark, Égypte, El Salvador, Équateur, États-Unis d'Amérique,
Éthiopie,  Fédération de Russie, France, Grèce, Guatemala, Haïti,
Honduras, Inde, Iran, Iraq, Liban, Libéria, Luxembourg, Mexique, Nicaragua,
Norvège, Nouvelle-Zélande, Panama, Paraguay, Pays-Bas, Pérou, Philippines,
Pologne, République arabe syrienne, République dominicaine, Royaume-Unie
de Grande-Bretagne et d'Irlande du Nord, Tchécoslovaquie, Turquie, Ukraine,
Uruguay, Venezuela, Yougoslavie, Afghanistan, Islande, Suède, Thailande,
Pakistan, Yémen, Myanmar, Israël, Indonésie, Albanie, Autriche, Bulgarie,
Cambodge, Espagne, Finlande, Hongrie, Irlande, Italie, Jamahiriya arabe
libyenne, Jordanie, Népal,  Portugal, République démocratique populaire lao,
Roumanie, Sri Lanka, Japan, Morocco, Sudan, Tunisia, Ghana, Malaisie, Guinée,
Bénin, Burkina Faso, Cameroun, Chypre, Congo, Côte d'Ivoire, Gabon, Madagascar,
Mali, Niger, Nigéria, République centrafricaine, République démocratique du
Congo, Sénégal, Somalie, Tchad, Togo, Mauritania, Mongolie, République-Unie
de Tanzanie, Sierra Leone, Algérie, Burundi, Jamaïque, Ouganda, Rwanda,
Trinité-et-Tobago, Kenya, Koweït, Malawi, Malte, Zambie, Gambie, Maldives,
Singapour, Barbade, Botswana, Guyana, Lesotho, Yémen démocratique, Guinée
équatoriale, Maurice, Swaziland, Fidji, Bahreïn, Bhoutan, Emirats arabes unis,
Oman, Qatar, Bahamas, Allemagne, Bangladesh, Grenade, Guinée-Bissau, Cap-
Vert, Comores, Mozambique, Papouasie-Nouvelle-Guinée, Sao Tomé-et-Principe,
Suriname, Angola, Samoa, Seychelles, Djibouti, Viet Nam, Dominique,
Iles Salomon, Sainte-Lucie, Saint-Vincent-et-les Grenadines, Zimbabwe,
Antigua and Barbuda, Belize, Vanuatu, Saint Kitts-et-Nevis, Brunéi
Darussalam, Liechtenstein, Namibie, Estonie, États fédérés de Micronésie,
```

Iles Marshall, Lettonie, Lituanie, République de Corée, République populaire démocratique de Corée, Arménie, Azerbaïdjan, Bosnie-Herzégovine, Croatie, Géorgie, Kazakstan, Kirghizistan, Ouzbékistan, République de Moldova, Saint-Marin, Slovénie, Tadjikistan, Turkménistan, Andorre, Érythrée, ex-République yougoslave de Macédoine, Monaco, République slovaque, République tchèque, Palaos, Kiribati, Nauru, Tonga, Tuvalu, Serbie et Monténégro, République démocratique du Timor-Leste, Suisse

11.2.2 / Les entités paramètres externes

Les entités paramètres externes permettent de construire une DTD complexe et volumineuse à partir de plusieurs autres DTD complémentaires.

La déclaration dans la DTD d'une entité paramètre externe s'effectue comme suit :

```
<!ENTITY % nom SYSTEM "URI">
```

Les références d'entités paramètres externes sont utilisées dans la DTD de la manière suivante :

```
%nom_entité;
```

Déclaration dans la DTD :

```
<!ENTITY % règles SYSTEM "http//www.monsite.com/regles.dtd">
```

Utilisation de l'entité dans la même DTD :

```
%règles;
```

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE societe [
<!ENTITY % services SYSTEM "services.dtd">
%services;
<!ENTITY % divisions SYSTEM "divisions.dtd">
%divisions;
<!ENTITY % employes SYSTEM "employes.dtd">
%employes;

<!ELEMENT societe (services, divisions, employes)>
]>
<societe>
 <services>
  <service id="DG001">
   <designation>Direction générale</designation>
  </service>
  <service id="ST001">
   <designation>Service technique</designation>
  </service>
  <service id="SC001">
   <designation>Service commercial</designation>
  </service>
  <service id="SC101">
   <designation>Service clientèle</designation>
  </service>
  <service id="RH001">
   <designation>Ressources humaines</designation>
  </service>
  <service id="SQ001">
   <designation>Service qualité</designation>
  </service>
  <service id="SP001">
   <designation>Service production</designation>
  </service>
  <service id="RD001">
   <designation>Recherche et développement</designation>
  </service>
 </services>
 <divisions>
  <division id="A001" services="DG001 RH001"/>
  <division id="B001" services="SC001 SC101"/>
  <division id="C001" services="SP001 SQ001 RD001"/>
 </divisions>
 <employes>
  <employe id="RJ1002" service="DG001">
   <nom>Robierre</nom>
   <prenom>Jean</prenom>
  </employe>
  <employe id="LA1012" service="DG001">
   <nom>Lardut</nom>
   <prenom>Anne</prenom>
  </employe>
  <employe id="GA1013" service="ST001">
   <nom>Guilde</nom>
   <prenom>Angélique</prenom>
  </employe>
  <employe id="HP1022" service="SC001">
   <nom>Henry</nom>
   <prenom>Paul</prenom>
  </employe>
  <employe id="MM1045" service="RH001">
   <nom>Mortier</nom>
   <prenom>Marc</prenom>
  </employe>
  <employe id="LS1102" service="SQ001">
   <nom>Lebreton</nom>
   <prenom>Sophie</prenom>
  </employe>
  <employe id="JM1095" service="RD001">
   <nom>Jolie</nom>
   <prenom>Martine</prenom>
  </employe>
  <employe id="MT1036" service="SC101">
```

```
    <nom>Marcelin</nom>
    <prenom>Tania</prenom>
   </employe>
   <employe id="LL1029" service="SC101">
    <nom>Léger</nom>
    <prenom>Laurence</prenom>
   </employe>
   <employe id="DM1052" service="SC001">
    <nom>Duroi</nom>
    <prenom>Maxime</prenom>
   </employe>
  </employes>
</societe>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Fichier services.dtd -->
<!ELEMENT designation (#PCDATA)>
<!ELEMENT service (designation)>
<!ATTLIST service
        id ID #REQUIRED
>
<!ELEMENT services (service+)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Fichier divisions.dtd -->
<!ELEMENT division EMPTY>
<!ATTLIST division
        id ID #REQUIRED
        services IDREF #REQUIRED
>
<!ELEMENT divisions (division+)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Fichier employes.dtd -->
<!ELEMENT employe (nom, prenom)>
<!ATTLIST employe
        id ID #REQUIRED
        service IDREF #REQUIRED
>
<!ELEMENT employes (employe+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
```

11.3 / Les entités analysables

Le contenu d'une entité analysable est représenté par le nom de texte de remplacement, lequel fait partie intégrante du document.

En conséquence, des données pouvant être correctement analysée par un processeur XML, et partant, formant un contenu XML bien formulé, contribuent à rendre une *entité analysable*. Dans le cas contraire l'entité est dite *non-analysable*.

La déclaration dans la DTD d'une entité non-analysable s'effectue comme suit :

```
<!--Entité générale externe-->
<!ENTITY nom SYSTEM "URI">
<!--Entité générale interne-->
<!ENTITY nom "texte_de_remplacement">
<!--Entité paramètre-->
<!ENTITY % nom "texte_de_remplacement">
```

Les entités analysables sont appelées par leur nom au moyen d'un appel d'entité.

```
&nom_entité; <!--Entité générale-->
%nom_entité; <!--Entité paramètre-->
```

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE poeme [
  <!-- Importation d'un fichier d'entités internes -->
  <!ENTITY % latin SYSTEM "latin.ent">
  %latin;
  <!ELEMENT poeme (#PCDATA)>
  <!ATTLIST poeme xml:space (default|preserve) "preserve">
]>
<poeme xml:space="preserve">
  &Agrave; quatre heures du matin, l'&eacute;t&eacute;,
  Le soleil d'amour dure encore.
  Sous les bocages s'&eacute;vapore
  L'odeur du soir f&ecirc;t&eacute;.

  L&agrave;-bas, dans leur vaste chantier
  Au soleil des Hesp&eacute;rides,
  D&eacute;j&agrave; s'agitent - en bras de chemise -
  Les charpentiers.

  Dans leurs D&eacute;serts de mousse, tranquilles,
  Ils pr&eacute;parent les lambris pr&eacute;cieux
  O&ugrave; la ville
  Peindra de faux cieux.

  &Ocirc;, pour ces Ouvriers charmants
  Sujets d'un roi de Babylone
  V&eacute;nus ! quitte un instant les Amants
  Dont l'&acirc;me est en couronne.

  &Ocirc; Reine des Bergers,
  Porte aux travailleurs l'eau-de-vie,
  Que leur force soient en paix
  En attendant le bain la mer &agrave; midi.
</poeme>

<-- Fichier latin.ent -->
<!-- Portions (C) International Organization for Standardization 1986 Permission to copy in any form is granted for
use with conforming SGML systems and applications as defined in ISO 8879, provided this notice is included in all
copies. -->
<!-- Character entity set. Typical invocation: <!ENTITY % HTMLlat1 PUBLIC "-//W3C//ENTITIES
Latin1//EN//HTML;"> %HTMLlat1; -->
<!ENTITY nbsp " ">
<!-- no-break space = non-breaking space, U+00A0 ISOnum -->
<!ENTITY iexcl "&#161;">
<!-- inverted exclamation mark, U+00A1 ISOnum -->
<!ENTITY cent "&#162;">
<!-- cent sign, U+00A2 ISOnum -->
<!ENTITY pound "&#163;">
<!-- pound sign, U+00A3 ISOnum -->
<!ENTITY curren "&#164;">
<!-- currency sign, U+00A4 ISOnum -->
<!ENTITY yen "&#165;">
<!-- yen sign = yuan sign, U+00A5 ISOnum -->
<!ENTITY brvbar "&#166;">
<!-- broken bar = broken vertical bar, U+00A6 ISOnum -->
<!ENTITY sect "&#167;">
<!-- section sign, U+00A7 ISOnum -->
<!ENTITY uml "&#168;">
<!-- diaeresis = spacing diaeresis, U+00A8 ISOdia -->
<!ENTITY copy "&#169;">
<!-- copyright sign, U+00A9 ISOnum -->
<!ENTITY ordf "&#170;">
<!-- feminine ordinal indicator, U+00AA ISOnum -->
<!ENTITY laquo "&#171;">
<!-- left-pointing double angle quotation mark = left pointing guillemet, U+00AB ISOnum -->
<!ENTITY not "&#172;">
<!-- not sign = discretionary hyphen, U+00AC ISOnum -->
<!ENTITY shy "&#173;">
<!-- soft hyphen = discretionary hyphen, U+00AD ISOnum -->
<!ENTITY reg "&#174;">
<!-- registered sign = registered trade mark sign, U+00AE ISOnum -->
<!ENTITY macr "&#175;">
<!-- macron = spacing macron = overline = APL overbar, U+00AF ISOdia -->
```

```
<!ENTITY deg "&#176;">
<!-- degree sign, U+00B0 ISOnum -->
<!ENTITY plusmn "&#177;">
<!-- plus-minus sign = plus-or-minus sign, U+00B1 ISOnum -->
<!ENTITY sup2 "&#178;">
<!-- superscript two = superscript digit two = squared, U+00B2 ISOnum -->
<!ENTITY sup3 "&#179;">
<!-- superscript three = superscript digit three = cubed, U+00B3 ISOnum -->
<!ENTITY acute "&#180;">
<!-- acute accent = spacing acute, U+00B4 ISOdia -->
<!ENTITY micro "&#181;">
<!-- micro sign, U+00B5 ISOnum -->
<!ENTITY para "&#182;">
<!-- pilcrow sign = paragraph sign, U+00B6 ISOnum -->
<!ENTITY middot "&#183;">
<!-- middle dot = Georgian comma = Greek middle dot, U+00B7 ISOnum -->
<!ENTITY cedil "&#184;">
<!-- cedilla = spacing cedilla, U+00B8 ISOdia -->
<!ENTITY sup1 "&#185;">
<!-- superscript one = superscript digit one, U+00B9 ISOnum -->
<!ENTITY ordm "&#186;">
<!-- masculine ordinal indicator, U+00BA ISOnum -->
<!ENTITY raquo "&#187;">
<!-- right-pointing double angle quotation mark = right pointing guillemet, U+00BB ISOnum -->
<!ENTITY frac14 "&#188;">
<!-- vulgar fraction one quarter = fraction one quarter, U+00BC ISOnum -->
<!ENTITY frac12 "&#189;">
<!-- vulgar fraction one half = fraction one half, U+00BD ISOnum -->
<!ENTITY frac34 "&#190;">
<!-- vulgar fraction three quarters = fraction three quarters, U+00BE ISOnum -->
<!ENTITY iquest "&#191;">
<!-- inverted question mark = turned question mark, U+00BF ISOnum -->
<!ENTITY Agrave "&#192;">
<!-- latin capital letter A with grave = latin capital letter A grave, U+00C0 ISOlat1 -->
<!ENTITY Aacute "&#193;">
<!-- latin capital letter A with acute, U+00C1 ISOlat1 -->
<!ENTITY Acirc "&#194;">
<!-- latin capital letter A with circumflex, U+00C2 ISOlat1 -->
<!ENTITY Atilde "&#195;">
<!-- latin capital letter A with tilde, U+00C3 ISOlat1 -->
<!ENTITY Auml "&#196;">
<!-- latin capital letter A with diaeresis, U+00C4 ISOlat1 -->
<!ENTITY Aring "&#197;">
<!-- latin capital letter A with ring above = latin capital letter A ring, U+00C5 ISOlat1 -->
<!ENTITY AElig "&#198;">
<!-- latin capital letter AE = latin capital ligature AE, U+00C6 ISOlat1 -->
<!ENTITY Ccedil "&#199;">
<!-- latin capital letter C with cedilla, U+00C7 ISOlat1 -->
<!ENTITY Egrave "&#200;">
<!-- latin capital letter E with grave, U+00C8 ISOlat1 -->
<!ENTITY Eacute "&#201;">
<!-- latin capital letter E with acute, U+00C9 ISOlat1 -->
<!ENTITY Ecirc "&#202;">
<!-- latin capital letter E with circumflex, U+00CA ISOlat1 -->
<!ENTITY Euml "&#203;">
<!-- latin capital letter E with diaeresis, U+00CB ISOlat1 -->
<!ENTITY Igrave "&#204;">
<!-- latin capital letter I with grave, U+00CC ISOlat1 -->
<!ENTITY Iacute "&#205;">
<!-- latin capital letter I with acute, U+00CD ISOlat1 -->
<!ENTITY Icirc "&#206;">
<!-- latin capital letter I with circumflex, U+00CE ISOlat1 -->
<!ENTITY Iuml "&#207;">
<!-- latin capital letter I with diaeresis, U+00CF ISOlat1 -->
<!ENTITY ETH "&#208;">
<!-- latin capital letter ETH, U+00D0 ISOlat1 -->
<!ENTITY Ntilde "&#209;">
<!-- latin capital letter N with tilde, U+00D1 ISOlat1 -->
<!ENTITY Ograve "&#210;">
<!-- latin capital letter O with grave, U+00D2 ISOlat1 -->
<!ENTITY Oacute "&#211;">
<!-- latin capital letter O with acute, U+00D3 ISOlat1 -->
<!ENTITY Ocirc "&#212;">
<!-- latin capital letter O with circumflex, U+00D4 ISOlat1 -->
<!ENTITY Otilde "&#213;">
```

```
<!-- latin capital letter O with tilde, U+00D5 ISOlat1 -->
<!ENTITY Ouml "&#214;">
<!-- latin capital letter O with diaeresis, U+00D6 ISOlat1 -->
<!ENTITY times "&#215;">
<!-- multiplication sign, U+00D7 ISOnum -->
<!ENTITY Oslash "&#216;">
<!-- latin capital letter O with stroke = latin capital letter O slash, U+00D8 ISOlat1 -->
<!ENTITY Ugrave "&#217;">
<!-- latin capital letter U with grave, U+00D9 ISOlat1 -->
<!ENTITY Uacute "&#218;">
<!-- latin capital letter U with acute, U+00DA ISOlat1 -->
<!ENTITY Ucirc "&#219;">
<!-- latin capital letter U with circumflex, U+00DB ISOlat1 -->
<!ENTITY Uuml "&#220;">
<!-- latin capital letter U with diaeresis, U+00DC ISOlat1 -->
<!ENTITY Yacute "&#221;">
<!-- latin capital letter Y with acute, U+00DD ISOlat1 -->
<!ENTITY THORN "&#222;">
<!-- latin capital letter THORN, U+00DE ISOlat1 -->
<!ENTITY szlig "&#223;">
<!-- latin small letter sharp s = ess-zed, U+00DF ISOlat1 -->
<!ENTITY agrave "&#224;">
<!-- latin small letter a with grave = latin small letter a grave, U+00E0 ISOlat1 -->
<!ENTITY aacute "&#225;">
<!-- latin small letter a with acute, U+00E1 ISOlat1 -->
<!ENTITY acirc "&#226;">
<!-- latin small letter a with circumflex, U+00E2 ISOlat1 -->
<!ENTITY atilde "&#227;">
<!-- latin small letter a with tilde, U+00E3 ISOlat1 -->
<!ENTITY auml "&#228;">
<!-- latin small letter a with diaeresis, U+00E4 ISOlat1 -->
<!ENTITY aring "&#229;">
<!-- latin small letter a with ring above = latin small letter a ring, U+00E5 ISOlat1 -->
<!ENTITY aelig "&#230;">
<!-- latin small letter ae = latin small ligature ae, U+00E6 ISOlat1 -->
<!ENTITY ccedil "&#231;">
<!-- latin small letter c with cedilla, U+00E7 ISOlat1 -->
<!ENTITY egrave "&#232;">
<!-- latin small letter e with grave, U+00E8 ISOlat1 -->
<!ENTITY eacute "&#233;">
<!-- latin small letter e with acute, U+00E9 ISOlat1 -->
<!ENTITY ecirc "&#234;">
<!-- latin small letter e with circumflex, U+00EA ISOlat1 -->
<!ENTITY euml "&#235;">
<!-- latin small letter e with diaeresis, U+00EB ISOlat1 -->
<!ENTITY igrave "&#236;">
<!-- latin small letter i with grave, U+00EC ISOlat1 -->
<!ENTITY iacute "&#237;">
<!-- latin small letter i with acute, U+00ED ISOlat1 -->
<!ENTITY icirc "&#238;">
<!-- latin small letter i with circumflex, U+00EE ISOlat1 -->
<!ENTITY iuml "&#239;">
<!-- latin small letter i with diaeresis, U+00EF ISOlat1 -->
<!ENTITY eth "&#240;">
<!-- latin small letter eth, U+00F0 ISOlat1 -->
<!ENTITY ntilde "&#241;">
<!-- latin small letter n with tilde, U+00F1 ISOlat1 -->
<!ENTITY ograve "&#242;">
<!-- latin small letter o with grave, U+00F2 ISOlat1 -->
<!ENTITY oacute "&#243;">
<!-- latin small letter o with acute, U+00F3 ISOlat1 -->
<!ENTITY ocirc "&#244;">
<!-- latin small letter o with circumflex, U+00F4 ISOlat1 -->
<!ENTITY otilde "&#245;">
<!-- latin small letter o with tilde, U+00F5 ISOlat1 -->
<!ENTITY ouml "&#246;">
<!-- latin small letter o with diaeresis, U+00F6 ISOlat1 -->
<!ENTITY divide "&#247;">
<!-- division sign, U+00F7 ISOnum -->
<!ENTITY oslash "&#248;">
<!-- latin small letter o with stroke, = latin small letter o slash, U+00F8 ISOlat1 -->
<!ENTITY ugrave "&#249;">
<!-- latin small letter u with grave, U+00F9 ISOlat1 -->
<!ENTITY uacute "&#250;">
<!-- latin small letter u with acute, U+00FA ISOlat1 -->
```

```
<!ENTITY ucirc "&#251;">
<!-- latin small letter u with circumflex, U+00FB ISOlat1 -->
<!ENTITY uuml "&#252;">
<!-- latin small letter u with diaeresis, U+00FC ISOlat1 -->
<!ENTITY yacute "&#253;">
<!-- latin small letter y with acute, U+00FD ISOlat1 -->
<!ENTITY thorn "&#254;">
<!-- latin small letter thorn with, U+00FE ISOlat1 -->
<!ENTITY yuml "&#255;">
<!-- latin small letter y with diaeresis, U+00FF ISOlat1 -->
```

11.4 / Les entités non-analysables

Une entité non-analysable permet de déclarer un contenu non-XML dans un document XML.

Les entités non-ananalysables sont notamment des fichiers audios, vidéos ou images.

La déclaration d'une entité non-analysable s'effectue en spécifiant le type de données de l'entité par l'intermédaire du mot clé *NDATA*. Chacune de ces entités est associée à une notation, identifiée par une chaîne de caractères par l'intermédiaire de l'instruction *<!NOTATION...>*.

La déclaration dans la DTD d'une entité non-analysable s'effectue comme suit :

```
<!ENTITY nom SYSTEM "URI" NDATA notation>
```

Les références d'entités non-analysables sont utilisées dans le document XML de la manière suivante :

```
&nom_entité;
```

Déclaration dans la DTD :

```
<!ENTITY illustration SYSTEM "image.gif" NDATA GIF>
```

Utilisation de l'entité dans le document XML :

Ce genre d'entité ne pouvant être analysée doit être déclarée dans un conteneur afin d'être correctement traité par l'analyseur XML courant.

```
<image src="illustration"/>
```

Le nom des entités non-analysables est fourni par la valeur d'un attribut de type ENTITY ou ENTITIES dans l'instruction *<!ATTLIST...>*.

```
<?XML version="1.0" STANDALONE="yes">
<!DOCTYPE collection [
  <!ELEMENT album EMPTY>
  <!ATTLIST album src ENTITY #REQUIRED>
  <!NOTATION jpg SYSTEM "image/jpg">
  <!ENTITY photo SYSTEM "../mariage.jpg" NDATA jpg>
]>
<collection>
  <album src="photo">
</collection>
```

11.5 / Les entités prédéfinies

Les entités prédéfinies sont destinées à remplacer des caractères courants dans le contenu textuel d'un document XML.

En principe, tous les processeurs XML sont capables d'interpréter correctement ces entités.

Déclaration	Appel	Caractère
<!ENTITY amp "&">	&amp;	**&**
<!ENTITY apos "'">	&apos;	**'**
<!ENTITY gt ">">	&gt;	**>**
<!ENTITY lt "<">	&lt;	**<**
<!ENTITY quot """>	&quot;	**"**

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE code [
  <!ELEMENT code (#PCDATA)>
  <!ATTLIST code xml:space (default|preserve) "preserve">
]>
<code xml:space="preserve">
  &lt;?xml version=&apos;1.0&apos; standalone=&quot;yes&quot;&gt;
  &lt;!DOCTYPE poésie [
  &lt;!ELEMENT poésie (#PCDATA)&gt;
  &lt;!ENTITY e-dans-o &apos;&oelig;&apos;&gt;
  &lt;!ENTITY eacute &apos;&oelig;&apos;&gt;
  &lt;!ATTLIST poésie xml:space (default|preserve) &apos;preserve&apos;&gt;
  ]&gt;
  &lt;poésie xml:space=&quot;preserve&quot;&gt;
  Au gr&eacute; du destrier,
  Sans varlet, n&apos;&eacute;cuyer.
  L&agrave; près d&apos;une fontaine,
  Que mon c&e-dans-o;eur, mon c&e-dans-o;eur a de peine !
  &lt;/poésie&gt;
</code>
```

12 / Les notations

Les notations sont utilisées en conjonction avec des entités non-analysables, c'est-à-dire contenant des données non-XML (image, applet Java, courrier électronique, etc.).

Le nom de la notation est associé, au sein de l'instruction *<!ENTITY...>*, au mot-clé *NDATA* permettant ainsi de faire référence à un identificateur (ID) permettant de décrire le format de données non-XML concerné.

L'identificateur externe (ID_Externe) peut tout à fait être un type MIME (Multipurpose Internet Mail Extensions) comme "video/mpeg", "text.rtf", "application.pdf", etc..

Si l'identificateur est public alors une adresse URI (Uniform Resource Identifier) est nécessaire.

La déclaration d'une notation s'effectue comme suit :

```
<!NOTATION Name
        ((SYSTEM "ID_Externe"
        | PUBLIC "ID_Public" "ID_Systeme")
        | PUBLIC "ID_Public")'>

<!NOTATION nom_notation SYSTEM "ID_Externe">

<!NOTATION nom_notation PUBLIC "ID_Public" "ID_Systeme">

<!NOTATION nom_notation PUBLIC "ID_Public">
```

L'appel à une notation s'opère de la manière suivante :

```
<!ENTITY nom_entite SYSTEM "URI" NDATA nom_notation>
```

Exemple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE vidéothèque [
  <!ELEMENT vidéothèque (vidéo)>
  <!ELEMENT vidéo EMPTY>
  <!ATTLIST vidéo source ENTITY #REQUIRED>
  <!NOTATION avi SYSTEM "video/avi">
  <!ENTITY vacance SYSTEM "mes_vacances.avi" NDATA avi>
]>
<vidéothèque>
  <vidéo source="&vacance;"/>
<vidéothèque>
```

13 / Les sections conditionnelles

Les sections conditionnelles permettent d'inclure ou d'ignorer des portions de déclarations dans la DTD (Document Type Definition).

L'instruction suivante permet d'inclure le bloc de déclarations dans la DTD.

```
<![INCLUDE Bloc_Déclarations...]]>
```

L'instruction suivante permet d'ignorer le bloc de déclarations dans la DTD.

```
<![IGNORE Bloc_Déclarations...]]>
```

Il est possible de substituer la directive *INCLUDE* ou *IGNORE* par une entité paramètre, déclarée par l'instruction *ENTITY*.

De cette façon, l'entité paramètre pourra prendre comme valeur *INCLUDE* ou *IGNORE* afin d'inclure ou d'ignorer la section concernée.

```
<!ENTITY % directive "INCLUDE"> <![%directive; Bloc_Déclarations...]]>
```

Si une DTD externe est utilisée pour plusieurs documents XML, l'entité paramètre déclarée dans la sous-DTD interne combinée avec les sections conditionnelles déclarées dans la DTD externe permettent d'adapter cette dernière à tous les documents XML.

Exemple :

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE document SYSTEM "definition.dtd" [
  <!ENTITY % directive "IGNORE">
]>
<!-- document.xml -->
<document>
  ...
</document>

<?xml version="1.0" standalone="yes"?>
<!DOCTYPE document SYSTEM "definition.dtd" [
  <!ENTITY % directive "INCLUDE">
]>
<!-- snd_document.xml -->
<document>
  ...
</document>
<!-- definition.dtd -->
<!ELEMENT document ANY>
<![%directive; Bloc_Déclarations...]]>
```

14 / Documents bien formés et valides

Un document XML est dit *bien formé* lorsque le document est correct sans toutefois possèder une DTD.

Le prologue du document ne contient pas de Définition de Type de Document (DTD) et la structure arborescente du document respecte les standards XML (nom des balises et attributs, imbrication des marqueurs XML,...).

```
<?xml version="1.0" standalone="yes"?>
<magasin>
  <service>
    <produit>
      Vaisselle
    </produit>
  </service>
</magasin>
```

Un document XML est dit *valide* lorsque le document est *bien formé* et possède une DTD.

Le prologue du document contient une Définition de Type de Document (DTD) et l'arborescence des éléments XML respecte strictement, la structure définie par la DTD.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE magasin [
  <!ELEMENT magasin (service)>
  <!ELEMENT service (produit)>
  <!ELEMENT produit (#PCDATA)>
]>
<magasin>
  <service>
    <produit>
      Vaisselle
    </produit>
  </service>
</magasin>
```

Un document XML bien formé doit obéir aux règles suivantes :

- Le document doit commencer par une déclaration XML.

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

- Le document doit utiliser une DTD (Document Type Definition).

  ```
  <!DOCTYPE élément_racine [Définitions...]>
  ```

- Le document doit contenir un seul et unique élément racine, déclaré dans l'instruction *DOCTYPE*, dont les marqueurs encadrent une ou plusieurs autres balises.

  ```
  <!DOCTYPE élément_racine [Déclarations...]>
  <élément_racine>
  <balise>donnée</balise>
  ...
  </élément_racine>
  ```

- Le document doit contenir un ou plusieurs éléments. Si le document contient un seul élément, alors ce document sera composé du seul élément racine.
- Les valeurs d'attribut doivent être impérativement encadrées par des guillemets simples (') ou doubles (").

  ```
  <balise attribut="valeur" attribut2='valeur'>
  ```

- Une balise contenant des données doit impérativement être fermée.

```
<balise>donnée</balise>
```

- Une balise vide (ne contenant pas de données à l'instar de la balise IMG dans le HTML) doit également contenir un marqueur de fermeture.

```
<balise></balise>
<balise/>
```

- Les balises doivent être correctement imbriquées, ainsi la ou les balises fils doivent être encadrées par une balise parente de départ et une de fin.

```
<balise parent>
<balise fils>donnée</balise fils>
<balise fils/>
</balise parent>
```

- Les noms d'éléments et d'attributs ne peuvent être composés que de lettres, y compris les lettres accentuées, de chiffres, de soulignés(_), de tirets (-), de points (.) et de deux points (:).
- Les noms d'éléments et d'attributs doivent commencer par une lettre ou un souligné.

```
<_balise attribut9='valeur'/>
```

- Les noms d'éléments dans les balises d'ouvertures ou de fermetures devront respecter une casse rigoureusement identique.

```
<baLiSe>Donnée</baLiSe>
```

- Les noms d'attributs doivent également conserver une casse rigoureusement identique.

```
<balise AttribuT="valeur">donnée<balise>
<balise2 AttribuT="valeur">donnée<balise2>
```

- Le caractère *inférieur à* (<) sert uniquement à ouvrir une balise.

```
<élément/>
```

- Le caractère *esperluette* (&) est utilisé essentiellement pour faire appel à une référence d'entité soit prédéfinie, soit générale, déclarée au préalable dans la DTD.

```
&amp;

&entité_générale;
```

- Les caractères *inférieur à* (<), l'*esperluette* (&) ou la séquence]]> doivent être remplacés respectivement par les références d'entité *<*, *&* ou *]]>*.
- Les attributs des documents XML bien formés sans DTD seront considérés de type *CDATA*, d'ailleurs considéré comme type par défaut.