2

Structure des documents XML

Ce chapitre vous apprend à structurer un document XML. La compréhension de ses composants est indispensable pour aborder les grands principes de XML.

Structure d'un document XML

Commençons par prendre un exemple simple de document XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Date de création : 30/09/07 -->
<cours titre="XML">
    <intervenant nom="alexandre brillant">
    </intervenant>
    <plan>
        Introduction
        XML et la composition de documents
    </plan>
</cours>
```

On peut d'ores et déjà analyser qu'un document XML est un document texte lisible. Sans comprendre nécessairement l'intégralité de la syntaxe, on en déduit qu'il s'agit de la description d'un cours dont l'intervenant n'est autre que l'auteur de cet ouvrage. Le plan du cours ressort également du document.

Nous allons maintenant décortiquer la syntaxe et en comprendre les tenants et les aboutissants.

L'en-tête : le prologue

Il s'agit de la première ligne d'un document XML servant à donner les caractéristiques globales du document, c'est-à-dire :

- La version XML, soit 1.0 ou 1.1, sachant que la très grande majorité des documents sont en version 1.0 et que la version 1.1 est assez décriée (recommandation du W3C en version 2 du 4 février 2004, qui note la résolution d'une incompatibilité avec les *mainframes* IBM).
- Le jeu de caractères employé (*encoding*). Pour fonctionner, le parseur va avoir besoin de distinguer le rôle de chaque caractère, certains étant réservés à la syntaxe et d'autres représentant des données. Pour définir un jeu de caractères, XML s'appuie sur des standards ISO et Unicode (voir *http://www.unicode.org/*). Notre standard Europe de l'ouest (ou iso-latin) est qualifié par ISO-8859-1. Lorsque l'encodage n'est pas précisé, c'est le standard UTF-8 qui est employé (avantage d'une compatibilité ANSI). Il existe beaucoup d'autres standards, citons ISO-2022-JP, Shift_JIS, EUC-JP… UTF-16 a la particularité de nécessiter l'encodage de chaque caractère avec 2 octets (un même fichier sera donc deux fois plus lourd encodé en UTF-16 qu'en UTF-8).
- Le champ `standalone` désigne l'indépendance du document, au sens où il n'existe aucun élément externe qui puisse altérer la forme finale du document XML fourni à l'application par le parseur (références d'entités, valeurs par défaut…). Ce champ prend les valeurs `yes` ou `no`. Dans la pratique, au-delà de l'aspect informatif, il n'a pas grand intérêt et vous pouvez l'ignorer.

Si nous reprenons notre exemple précédent, nous avons donc comme prologue :

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

À noter que ce prologue est encapsulé par `<?` et `?>` et qu'il n'existe pas d'espaces (de blancs) entre le début du document et cet élément. Autre remarque qui peut surprendre : le prologue n'est pas obligatoire et, dans ce cas, le parseur utilise un comportement par défaut (version 1.0 et encoding UTF-8).

Les instructions de traitement

Les instructions de traitement (*processing instruction* ou PI) n'ont pas de rôle lié aux données ou à la structuration de votre document. Elles servent à donner à l'application qui utilise le document XML des informations. Ces dernières sont totalement libres et dépendent avant tout du concepteur de l'application de traitement. On les positionne à n'importe quel endroit du document (après le prologue, bien entendu). Un cas typique est l'utilisation avec les navigateurs Mozilla Firefox ou Internet Explorer pour effectuer la transformation d'un document XML en document XHTML affichable avec l'instruction :

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

L'emploi de cette instruction et le rôle du document `affichage.xsl` seront éclairés dans la partie consacrée aux feuilles de styles.

> **Remarque**
> Attention à ne pas confondre ces instructions de traitement avec le prologue qui, s'ils sont de syntaxe similaire, n'ont pas le même rôle.

Enfin, ces instructions de traitement ont également pour fonction de pallier des manques dans la spécification XML, par exemple pour effectuer un lien vers un langage de validation non standard (vérifiant donc la conformité du document par rapport à une grammaire) de type Relax NG (*http://www.relaxng.org/*).

Les commentaires

Il y a peu de chose à dire sur les commentaires. Ce sont les mêmes qu'en HTML (ceci est dû au lien de parenté avec SGML). Ils se positionnent n'importe où après le prologue et peuvent figurer sur plusieurs lignes.

Notre exemple contient le commentaire suivant :

```
<!-- Date de création : 30/09/07 -->
```

Point important : les caractères `--` sont interdits comme commentaires pour une raison que l'on comprendra aisément (ambiguïté d'analyse pour le parseur).

La déclaration du type de document

Cette déclaration optionnelle sert à attacher une grammaire de type DTD (*Document Type Definition*) à votre document XML. Elle est introduite avant la première balise (racine) de votre document sous cette forme :

```
<!DOCTYPE racine SYSTEM "URI vers la DTD">
```

`racine` est le premier élément (la première balise). L'URI peut être absolue ou relative au document. Il est généralement préférable soit d'utiliser une URI relative, pour pouvoir déplacer le document XML et sa grammaire sans difficulté, ou bien d'exploiter une URL disponible depuis n'importe quel endroit (sur Internet/Intranet, par exemple).

Exemple :

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

Dans cet exemple, la DTD `cours.dtd` est localisée relativement à notre document XML.

Le mot-clé `SYSTEM` est important et indique qu'il s'agit d'une DTD qui vous est propre. L'alternative est le mot-clé `PUBLIC`.

Exemple :

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Strict//EN" >
```

Ici, l'URI est remplacée par un identifiant public composé de :

- - : indique qu'il s'agit d'un format non compatible ISO (sinon on utilise +).
- IETF : organisme gérant le format.
- DTD : indique la nature du document.
- HTML 2.0 Strict : description du document.
- EN : un code langue (ISO 639).

On peut également préciser, après l'identifiant public, une URI. C'est utile si le parseur ne peut pas traduire l'identifiant public en URI exploitable.

La déclaration de type de document peut également héberger un bloc d'instructions propre aux DTD (on parle alors de DTD interne). Sans rentrer dans les détails, puisque cela sera repris dans l'analyse des DTD dans un prochain chapitre, voici un exemple de DTD interne qui sert à valider notre document XML :

```
<!DOCTYPE cours [
<!ELEMENT cours ( intervenant, plan )>
<!ELEMENT intervenant EMPTY>
<!ELEMENT plan (#PCDATA)>
<!ATTLIST cours titre CDATA #REQUIRED>
<!ATTLIST intervenant nom CDATA #REQUIRED>
]>
```

Les nœuds élément

Les éléments gèrent la structuration des données d'un document XML, un peu à la manière des répertoires qui servent à l'organisation des fichiers. On peut les qualifier de métadonnés, au sens où ils ne font pas partie réellement des données mais servent à en désigner la nature. À la place du terme élément, on peut utiliser les termes balise, *tag* ou encore nœud.

Pour se familiariser rapidement, reprenons l'exemple précédent. À l'intérieur, nous trouvons les éléments `cours`, `intervenant` et `plan`.

Pour décrire ce que contiennent les éléments, on parle de modèle de contenu. On trouve :

- Rien : il n'y pas de contenu, l'élément est vide.
- Du texte : nous détaillerons par la suite cette notion.
- Un ou plusieurs éléments : on peut les qualifier d'éléments fils, l'élément les contenant étant appelé un élément parent (à ne pas confondre avec un élément ancêtre,

qui indique qu'il existe une relation de conteneur à contenu et qui est donc plus large).

- Un mélange de textes et d'éléments : c'est une forme plus rare qui peut révéler une erreur de structuration. Elle reste cependant utile, lorsque l'on souhaite « décorer » un texte quelconque (cas du paragraphe en HTML avec des zones en gras, italique…).

Reprenons notre exemple XML et complétons-le un peu :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cours>
   <intervenant>
   Phileas
   </intervenant>
   <separateur/>
   <chapitre>
   Formation XML
       <para>Un paragraphe</para>
       <para>Autre paragraphe</para>
   </chapitre>
</cours>
```

- `cours` : élément racine contenant trois éléments fils : `intervenant`, `separateur` et `chapitre` ;
- `intervenant` : élément contenant du texte ;
- `separateur` : élément sans contenu ;
- `chapitre` : élément contenant du texte et des éléments fils `para` ;
- `para` : élément contenant du texte.

Remarque

Nous effectuons une mise en page minimale à l'aide de tabulations (on parle d'indentation) afin de faire ressortir la structure du document, comme les relations parent/enfant.

Si maintenant nous nous penchons sur la syntaxe, nous avons donc :

- `<element>` : balise ouvrante.
- `</element>` : balise fermante.
- `<element/>` : balise ouverte et fermée que l'on nomme balise autofermée. C'est l'équivalent de `<element></element>`. Elle désigne donc un élément vide.

Remarque

Cette dernière forme est propre à XML et n'apparaît pas dans HTML.

Exercice 1

Création d'un livre en XML

On souhaite écrire un livre en utilisant le formalisme XML. Le livre est structuré en sections (au moins 2), en chapitres (au moins 2) et en paragraphes (au moins 2).

Le livre doit contenir la liste des auteurs (avec nom et prénom).

Tous les éléments doivent posséder un titre, sauf le paragraphe qui contient du texte.

Proposez une structuration XML de ce document (avec 2 auteurs, 2 sections, 2 chapitres par section et 2 paragraphes par chapitre).

Vérifiez, à l'aide de l'éditeur, que votre document est bien formé.

Attention : ne pas utiliser d'attributs ; l'encodage utilisé est `ISO-8859-1`

Votre document sera nommé `livre1.xml`.

Les attributs d'un élément

Un attribut est un couple (clé, valeur) associé à la définition d'un élément. Il est localisé dans la balise ouvrante de l'élément. Un élément peut donc avoir de 0 à n attributs uniques. L'attribut est complémentaire de l'élément de par son rôle au sens où il ajoute une information à l'élément ou bien encore le complète dans sa définition.

Exemple :

```
<auteur nom="brillant" prenom="alexandre">…</auteur>
<contact email='a@a.fr'/>
```

`nom` et `prenom` sont des attributs de l'élément `auteur` alors que `email` est un attribut de l'élément `contact`.

On sépare les attributs par au moins un espace (blanc simple, tabulation, retour à la ligne). Les valeurs d'attributs peuvent figurer sur plusieurs lignes. On utilise soit les guillemets, soit les apostrophes pour encapsuler les valeurs.

Voici un exemple de document XML avec des attributs :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cours>
   <intervenant nom="fog" prenom="phileas"/>
   <introduction/>
   <chapitre numero="1">
       Formation XML
       <paragraphe>Détails du format</paragraphe>
   </chapitre>
</cours>
```

Choix entre éléments et attributs

L'attribut peut sembler superflu. En effet, ce qui s'écrit avec des attributs peut également l'être en s'appuyant uniquement sur des éléments.

Exemple :

Cas avec attributs :

```
<personne nom="brillant" prenom="alexandre"/>
```

Cas sans attribut :

```
<personne>
   <nom>
       brillant
   </nom>
   <prenom>
       alexandre
   </prenom>
</personne>
```

Cependant, l'inverse n'est pas vrai car un attribut ne peut pas être répété dans un élément (mais il peut l'être au travers d'éléments différents).

Exemple :

Cas avec éléments :

```
<carnet>
   <personne>…
   </personne>
   <personne>…
   </personne>
</carnet>
```

S'il fallait supprimer les éléments `personne` au profit d'attributs, il faudrait utiliser une convention de nommage complexe des attributs (avec une numérotation pour chaque personne…) ce qui ferait perdre tout intérêt à XML qui sert justement à structurer des données.

On peut définir cependant quelques règles simples pour déterminer s'il est préférable d'utiliser un attribut ou un élément. Lorsqu'une valeur est de taille modeste, a peu de chance d'évoluer vers une structure plus complexe, et n'est pas répétée, alors l'attribut peut tout à fait convenir. Dans tous les autres cas, l'élément reste incontournable. Si vous avez un doute entre un attribut et un élément, alors choisissez toujours l'élément qui est le plus ouvert et garantira le plus facilement une évolution.

Exercice 2

Utilisation des attributs

Conception de livre2.xml à partir de livre1.xml

On souhaite compléter la structure du document XML de l'exercice précédent par les attributs `nom` et `prenom` pour les auteurs et `titre` pour le livre, les sections et les chapitres.

Analysez la structure du nouveau document. Y a-t-il des simplifications possibles ?

Vérifiez, à l'aide de l'éditeur, que votre document est bien formé.

Les nœuds textes

Dans un document XML, ce qui est appelé donnée est le texte qui est associé à l'attribut, c'est-à-dire sa valeur, ou à l'élément, c'est-à-dire son contenu. Les données constituent le cœur du document, et tout le reste, le formalisme, ne sert qu'à séparer et classer ces données. Celles-ci sont dites terminales dans l'arborescence XML, dans la mesure où il ne peut y avoir de structuration supplémentaire. Dans le vocabulaire propre à DOM (*Document Object Model*), que nous aborderons dans un autre chapitre, la donnée apparaît comme un nœud texte. Elle est bien entendu liée au prologue puisque l'encodage sert à désigner le jeu de caractères utilisé dans votre document.

Dans le premier exemple XML que nous avons donné, le texte `XML` est une donnée liée à l'attribut `titre` de l'élément `cours`. `Introduction…` est une donnée liée à l'élément `plan`.

Comme certains caractères sont réservés à la syntaxe XML, il faut être vigilant lors de l'écriture des données.

Exemple :

```
<calcul>
   if ( a<b et b>c) ...
</calcul>
```

Dans cet exemple, on voit bien que `<b et b>` n'est pas une balise mais fait partie des données liées à l'élément `calcul`. Ce que nous comprenons facilement n'est pas sans ambiguïté pour un parseur qui identifiera une balise de syntaxe incorrecte.

Pour résoudre ce problème, nous disposons d'entités prédéfinies. L'entité en elle-même peut être perçue comme un raccourci vers une autre valeur, ce qui a l'avantage de faciliter la maintenance (changement de valeur avec répercussions) et de masquer les conflits syntaxiques.

Voici la liste des entités prédéfinies :

- `<` équivalent de < (*less than*) ;
- `>` équivalent de > (*greater than*) ;
- `&` équivalent de & (*ampersand*) ;

- `&quot;` équivalent de " (*quote*) ;
- `&apos;` équivalent de ' (*apostrophe*).

L'exemple précédent peut donc être correctement réécrit :

```
If (a&lt;b et b&gt;c)
```

Bien entendu, les entités prédéfinies ne servent qu'à lever une ambiguïté syntaxique pour le parseur.

Ces entités peuvent être utilisées dans un élément, pour du texte, ou dans un attribut, pour sa valeur.

Les entités prédéfinies présentes en trop grand nombre dans un même bloc peuvent alourdir inutilement le document. Dans le cas du contenu textuel d'un élément (et uniquement dans ce cas), nous disposons des sections CDATA (*Character Data*). Cette section doit être considérée comme un bloc de texte dont les caractères seront pris tel quel par le parseur jusqu'à la séquence de fin `]]>`.

Exemple :

```
<![CDATA[
<element>C'est un document XML
</element>
]]>
```

Dans cet exemple, `<element>` n'est pas considéré comme une balise de structuration, mais comme du texte.

Exercice 3

Utilisation des entités prédéfinies

On se propose de créer un nouveau document `livre2bis.xml` reprenant l'exercice précédent (`livre2.xml`). Placez dans 2 paragraphes un bloc de texte contenant l'extrait suivant :

```
<element id="10">&gt;</element>
```

Pour le premier paragraphe, employez les entités prédéfinies.

Pour le deuxième paragraphe, employez une section CDATA.

Les entités du document

Nous avons déjà vu une première forme d'entité à travers les entités prédéfinies. Si vous concevez une DTD (*Document Type Definition*), sachez que vous pourrez à loisir créer votre propre entité et y faire référence dans vos documents. Nous aborderons la syntaxe des DTD dans un prochain chapitre, mais en attendant, voici un exemple d'utilisation :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE cours [
<!ENTITY auteur "Alexandre Brillant">
```

```
<!ELEMENT cours (#PCDATA)>
]>
<cours>
   Cours réalisé par &auteur;
</cours>
```

L'entité `auteur` est liée à un nom et un prénom.

La valeur de l'entité peut être interne ou externe (placée dans un autre fichier). Attention à ne pas confondre les entités que l'on trouvera dans un document HTML (comme ` `) avec celles que vous aurez à votre disposition. Dans les deux cas il s'agit bien d'une DTD qui en délimite l'utilisation. Si votre DTD ne comporte pas d'entités identiques à celles possibles dans un document HTML alors vous ne pourrez pas les employer.

Il existe également une forme d'entités, qui ne sert qu'à la DTD, appelée entités paramétriques. Nous les aborderons dans un prochain chapitre.

Enfin, il reste l'entité caractère. Cette entité va nous être utile pour employer un caractère issu de la table ISO/IEC 10646 que l'on retrouve également avec HTML (proche de UTF-16). On spécifie la valeur du caractère soit en décimal soit en hexadécimal.

Par exemple, pour représenter un retour à la ligne sous Linux/Unix (il faut en plus le caractère 13 pour la plate-forme Windows), on utilise :

```
Forme décimale :
&#10;
Forme hexadécimale :
&#xA;
```

Quelques règles de syntaxe

Ces règles de syntaxe sont à respecter impérativement pour qu'un document XML soit bien formé.

- Le nom d'un élément ne peut commencer par un chiffre.
- Si le nom d'un élément est composé d'un seul caractère il doit être dans la plage [a-zA-Z] (c'est-à-dire une lettre minuscule ou majuscule sans accent) ou _ ou :.
- Avec au moins 2 caractères, le nom d'un élément peut contenir _, -, . et : plus les caractères alphanumériques (attention, le caractère : est réservé à un usage avec les espaces de nom que nous aborderons par la suite).
- Tous les éléments ouverts doivent être fermés.
- Un élément parent est toujours fermé après la fermeture des éléments fils. Voici un contre-exemple où l'élément fils `b` est incorrectement fermé après la fermeture de son élément parent `a` : `<a><b></a></b>`.

Pour connaître plus précisément les caractères autorisés, vous pouvez vous reporter à la grammaire XML du W3C disponible à l'adresse *http://www.w3.org/TR/2006/REC-xml-20060816/#sec-well-formed*.

Quelques conventions de nommage

Voici quelques conventions souvent employées dans les documents XML :

- Employer des minuscules pour les attributs et les éléments.
- Éviter les accents dans les noms d'attributs et d'éléments pour des raisons de compatibilité avec les outils du marché qui proviennent souvent d'un univers anglo-saxon.
- Préférer les guillemets délimitant les valeurs d'attribut.
- Séparer les noms composés de plusieurs mots par les caractères -, _, . ou une majuscule. Essayer d'être homogène dans votre document en gardant la même convention.

Exemples :

```
<value-of/>
<valEntier></valEntier>
```

Quelques exemples XML

Voici quelques exemples simples de documents XML. Certaines formes de structuration d'usage courant ont été normalisées par différents organismes comme le W3C ou le consortium OASIS (*http://www.oasis-open.org*). Vous pouvez à titre d'exercice essayer de distinguer chaque composante de ces documents.

Le format MathML

MathML est une recommandation du W3C (*http://www.w3.org/Math/*) servant à d'écrire des expressions mathématiques.

```
<?xml version="1.0"?>
<math>
 <mrow>
  <msup> <mi>x</mi><mn>2</mn> </msup>
  <mo>+</mo>
  <mrow>
    <mn>4</mn><mo>&InvisibleTimes;</mo><mi>x</mi>
  </mrow>
  <mo>+</mo>
  <mn>4</mn>
 </mrow>
</math>
```

Très brièvement, `mo` désigne un opérateur, `mrow` une expression, `mi` et `mn` respectivement des opérandes variable (par exemple `x`) et nombre.

Ici nous avons décrit l'expression : `x^2+4x+4`.

Le format VoiceXML

VoiceXML est un standard pour décrire des choix (processus de questions/réponses) liés à des répondeurs vocaux (*http://www.voicexml.org/*).

```
<vxml version="2.0">
<menu>
  <prompt>Faire un choix :<enumerate/></prompt>
  <choice next="http://www.meteo.exemple/meteo.vxml">Meteo
  </choice>
  <choice next="http://www.infos.exemple/infos.vxml">Info
  </choice>
  <noinput>Merci de faire un choix <enumerate/></noinput>
</menu>
</vxml>
```

Très brièvement, `menu` représente une liste de choix, `prompt` est une question, `choice` une réponse possible et `noinput` la phrase produite si aucun choix n'est réalisé ; le menu est rappelé avec `enumerate`.

Les espaces de noms

Les espaces de noms sont un concept très commun en informatique. Nous les retrouvons dans de nombreux langages de programmation afin de prévenir d'éventuels conflits. Par exemple, dans le langage de programmation Java, les *packages* servent à délimiter la portée d'une classe. En choisissant un package, le développeur lève tout conflit potentiel sur le nom, car la classe sera par la suite utilisée directement ou indirectement via son package et son nom.

Application des espaces de noms dans un document XML

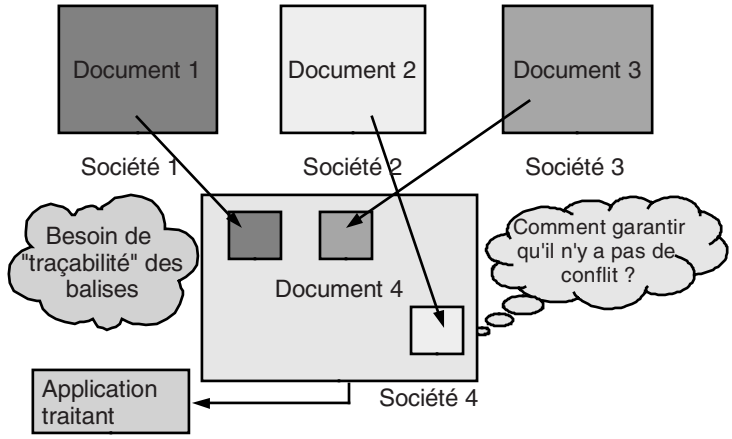
Lorsque nous créons un document XML, la structure du document est déterminée à partir des données que nous allons avoir à gérer. Mais ce que nous ne pouvons pas prévoir, ce sont les données d'origines diverses qui peuvent être en contradiction avec nos choix de structure initiaux. Il n'est pas non plus impossible que nous ayons à recouper plusieurs documents XML afin de produire un document de synthèse. Toutes ces opérations peuvent introduire autour de nos balises et attributs des conflits potentiels, car une balise ou un attribut peut avoir du sens dans un contexte mais pas dans un autre. C'est d'autant plus vrai que, ne l'oublions pas, un document XML est traité par une application qui ne doit pas rencontrer d'ambiguïté. Comment garantir, quand nous choisissons une balise ou un attribut, que personne n'a eu la même idée dans un contexte d'application différent ? Par exemple, le choix de l'élément `titre` peut représenter le titre d'un livre, mais pourquoi pas le titre de noblesse d'une personne…

Pour délimiter la portée d'une balise, d'un attribut ou d'une valeur d'attribut, nous disposons d'espaces de noms (*namespace*). Par analogie, cette notion est reprise dans la plupart des langages de programmation récents, comme les packages en java ou les

namespaces dans l'environnement .net. La notion d'espace de noms peut être perçue comme un groupe d'appartenance ou une famille. L'utilisation des espaces de noms garantit une forme de traçabilité de la balise et évite les ambiguïtés d'usage.

Les espaces de noms sont très souvent employés dans les spécifications W3C car ces documents peuvent être mélangés à d'autres et entraîner des conflits. On en trouve, par exemple, dans les feuilles de styles (XSLT) et dans les schémas W3C.

Figure 2-1
Croisement de documents



Dans la figure 2-1, nous avons 3 documents provenant de sociétés différentes. Ces trois documents sont intégrés dans un document utilisé par la société 4. Il est impératif que ces mélanges se fassent au mieux et sans conflit. Il peut être également intéressant, pour l'application de traitement, de savoir différencier ce qui vient de telle ou telle société.

Pour que les espaces de noms aient un sens, il faut pour chacun d'eux un identifiant unique. Dans le cas contraire, les espaces de noms pourraient eux-mêmes être en conflit. Cet identifiant unique peut être simplement l'URL, puisqu'il ne peut y avoir qu'un propriétaire pour une URL donnée (ceci est lié au nom de domaine présent dans l'URL et à son obtention auprès d'un organisme comme l'AFNIC en France). C'est un peu similaire à un bureau d'enregistrement.

> **Attention**
> L'URL ne signifie pas qu'il doit y avoir un document sur votre serveur HTTP. Ce n'est qu'un identifiant et n'importe quelle chaîne de caractères pourrait en réalité être employée.

Vocabulaire : qualification des éléments

Un élément qui est connu dans un espace de noms est dit qualifié ; dans le cas contraire, il est dit non qualifié. Lorsqu'on ignore l'espace de noms d'un élément, on dit qu'on s'intéresse à sa forme locale. Ce vocabulaire est à connaître lorsqu'on travaille avec un parseur dans les techniques dites SAX ou DOM.

Utilisation des espaces de noms dans un document XML

Il existe plusieurs méthodes pour intégrer des espaces de noms. Nous allons aborder les règles implicites et explicites sachant que cette dernière solution est davantage employée.

L'espace de noms par défaut

Un premier usage consiste à utiliser simplement l'espace de noms par défaut. Ce dernier est précisé par un pseudo-attribut `xmlns` (retenir que `ns` est pour *namespace)*. La valeur associée sera une URL garantissant l'unicité de l'espace de noms. L'espace de noms par défaut s'applique à l'élément où se situe sa déclaration et à tout son contenu.

Exemple :

```
<chapitre xmlns="http://www.masociete.com">
   <paragraphe>
   ...
   </paragraphe>
</chapitre>
```

Ici l'élément `chapitre` est dans l'espace de noms `http://www.masociete.com`. C'est également le cas de l'élément `paragraphe`, puisqu'il est dans l'élément `chapitre`.

Nous pouvons changer l'espace de noms par défaut même dans les éléments enfants : dans ce cas, une règle de priorité est appliquée. Attention, les espaces de noms ne sont pas imbriqués ; on ne peut appliquer qu'un seul espace de noms à la fois.

Exemple :

```
<chapitre xmlns="http://www.masociete.com">
   <paragraphe xmlns="http://www.autresociete.com">
   ...
   </paragraphe>
</chapitre>
```

L'élément `paragraphe` n'appartient pas à l'espace de noms `http://www.masociete.com` mais uniquement à l'espace de noms `http://www.autresociete.com`.

> **Attention**
> Un espace de noms par défaut ne concerne que les éléments. Les attributs et les textes n'y appartiennent pas. Le texte d'un élément n'est jamais dans un espace de noms puisqu'il représente la donnée.

L'espace de noms explicite

L'espace de noms par défaut présente l'inconvénient d'être peu contrôlable sur un document de taille importante. En effet, tout ajout ou modification d'un tel espace va se répercuter sur la totalité du contenu.

Pour disposer de davantage de souplesse dans ces espaces et pouvoir également les appliquer aux attributs et valeurs d'attributs, la syntaxe introduit la notion de préfixe.

Un préfixe est une sorte de raccourci vers l'URL de l'espace de noms. On peut faire l'analogie avec une forme de macro ou de constante. Autre analogie, dans le langage SQL : il arrive que des champs de même nom se trouvent dans des tables différentes et qu'une opération de jointure entre ces tables oblige à distinguer chaque champ en le préfixant par un alias lié à chaque table.

On déclare un préfixe comme un pseudo-attribut commençant par `xmlns:prefixe`. Une fois déclaré, il est employable uniquement dans l'élément le déclarant et dans son contenu. L'emploi consiste à ajouter en tête de l'élément, de l'attribut ou d'une valeur d'attribut, le préfixe suivi de `:`.

Exemple :

```
<p:resultat xmlns:p="http://www.masociete.com">
</p:resultat>
```

L'élément `resultat` est dans l'espace de noms `http://www.masociete.com` grâce au préfixe `p`.

Attention

Lorsqu'un élément est préfixé, son contenu doit l'être aussi si l'on souhaite que l'espace de noms s'applique également.

La notion de préfixe n'a de sens que par rapport à l'URL associée. Dans la pratique, l'application ne voit pas ce préfixe mais uniquement l'URL associée à telle ou telle partie du document. C'est comme si un élément était un couple (nom de l'élément, espace de noms), de façon analogue à un attribut et sa valeur.

Exemple :

```
Document 1 :
<p:res xmlns:p="http://www.masociete.com">
</p:res>
Document 2 :
<zz:res xmlns:zz="http://www.masociete.com">
</zz:res>
```

Les documents 1 et 2 sont strictement identiques malgré un préfixe différent ; `res` étant dans tous les cas un élément appartenant à l'espace de noms `http://www.masociete.com`.

On peut déclarer et utiliser plusieurs espaces de noms grâce aux préfixes.

Exemple :

```
<p:res xmlns:p="http://www.masociete.com" xmlns:p2="http://www.autresociete.com">
   <p2:res>
   </p2:res>
</p:res>
```

Le premier élément `res` est dans l'espace de noms `http://www.masociete.com` alors que l'élément `res` à l'intérieur est dans l'espace de noms `http://www.autresociete.com`.

Attention

On ne peut pas utiliser plusieurs préfixes en même temps sur un élément, attribut ou valeur d'attribut (exemple à ne pas faire `p:p2:res`).

La suppression d'un espace de noms

Aucun espace de noms n'est utilisé lorsqu'il n'y a pas d'espace de noms par défaut ni de préfixe.

Exemple :

```
<p:element xmlns:p="http://www.masociete.com">
   <autreelement/>
</p:element>
```

L'élément `element` est dans l'espace de noms `http://www.masociete.com` alors que l'élément `autreelement`, qui n'est pas préfixé, n'a pas d'espace de noms.

Pour supprimer l'action d'un espace de noms il suffit d'utiliser la valeur vide `""`, ce qui revient à ne pas avoir d'espace de noms.

Exemple :

```
<element xmlns="http://www.masociete.com">
   <autreelement xmlns="">
       .. Aucun d'espace de noms
   </autreelement>
   <encoreunelement>
       ... Espace de nom par défaut
   </encoreunelement>
</element>
```

L'élément `element` est dans l'espace de noms `http://www.masociete.com` alors que l'élément `autreelement` n'est plus dans un espace de noms. L'élément `encoreunelement` se trouve également dans l'espace de noms `http://www.masociete.com`, de par l'espace de noms de son parent.

Exercice 4

Utilisation des espaces de noms par défaut et avec préfixe

Il s'agit de créer un document `livre3.xml` sur la base de `livre1.xml` en respectant les points suivants :

- Mettez tous les éléments dans l'espace de noms `http://www.masociete.com` sans utiliser d'espace de noms par défaut.
- Mettez la deuxième section dans un espace de noms `http://www.monentreprise.com`.
- Mettez le dernier paragraphe du dernier chapitre de la dernière section sans espace de noms.

Application d'un esspace de noms sur un attribut

Les espaces de nom peuvent s'appliquer via un préfixe sur un attribut ou une valeur d'attribut. Cet emploi peut servir, par exemple, à introduire des directives sans changer de structure. Cela peut également servir à contourner la règle qui veut que l'on ne puisse pas avoir plusieurs fois un attribut de même nom sur une déclaration d'élément.

Exemple :

```
<livre xmlns:p="http://www.imprimeur.com" p:quantite="p:50lots">
   <papier type="p:A4"/>
</livre>
```

Dans cet exemple, nous avons qualifié l'attribut `quantité` ainsi que les valeurs d'attribut `50lots` et `A4`.

Il existe une autre utilisation d'un espace sur une valeur d'attribut : un espace de noms permet de lever l'ambiguïté sur une valeur d'attribut. Prenons, par exemple, la valeur 1 : est-ce une chaîne de caractères ? Un nombre binaire ? Hexadécimal ? Un décimal ? Une seconde ? L'espace de noms sur une valeur d'attribut est couramment employé dans les schémas W3C que nous aborderons dans un autre chapitre.

Exercice 5

Utilisation des espaces de noms sur des attributs

Nous supposons que le livre des exercices précédents est maintenant disponible en plusieurs langues (au moins en français et en anglais).

Proposez une méthode pour gérer tous les titres et paragraphes en plusieurs langues.

Créez un document `livre4.xml` à partir de `livre1.xml`

Exemples de documents XML avec espace de noms

Pour vous exercer, vous pourrez analyser, dans les exemples ci-dessous, l'appartenance des différents éléments à tel ou tel espace de noms.

Le format XLink

XLink (*XML Linking Language* : *http://www.w3.org/XML/Linking*) sert à définir des liens entre différents documents. Dans la pratique, XLink est peu employé car il est plus simple d'introduire ses propres attributs ou balises pour réaliser des liens.

```
<annuaire xmlns:xlink="http://www.w3.org/1999/xlink">
 <persone
  xlink:href="etudiant/etudiant62.xml"
  xlink:label="Louis"
  xlink:role="http://www.campus.fr/etudiant"
  xlink:title="Louis Henri"/>
</annuaire>
```

`xlink` est le préfixe retenu pour réaliser un lien. Ce lien est effectué par l'attribut `href` vers un autre document ; le `label` est le texte affiché pour le lien ; le `role` est la nature du lien (un étudiant) et `title` est le titre du lien.

Le format XHTML

XHTML (*eXtensible HyperText Markup Language* : *http://www.w3.org/TR/xhtml1/*) est la version XML du standard HTML.

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
  <head>
    <title>Ma page</title>
  </head>
  <body>
    <p>Mon texte</p>
  </body>
</html>
```

Les éléments XHTML sont donc dans l'espace de noms `http://www.w3.org/1999/xhtml`. À noter que la présence de l'attribut `lang` avec ou sans préfixe `xml` n'est liée qu'à une problématique de compatibilité avec les navigateurs HTML. Le préfixe `xml` existe par défaut pour désigner un attribut propre au standard.

Autre exemple introduisant des éléments MathML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <body>
    <p>Formule mathématique</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply> <log/>
        <logbase>
          <cn> 3 </cn>
        </logbase>
        <ci> x </ci>
      </apply>
   </math>
  </body>
</html>
```

Outre les éléments XHTML, nous trouvons des éléments MathML dont l'espace de noms est *http://www.w3.org/1998/Math/MathML*.

Correction des exercices

L'ensemble des exercices a été réalisé avec le logiciel EditiX (*http://www.editix.com/*). Une version d'évaluation de 30 jours est librement téléchargeable (*http://www.editix.com/download.html*).

Exercice 1

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

<livre>
  <titre>Mon livre</titre>
  <auteurs>
    <auteur><nom>Brillant</nom><prenom>Alexandre</prenom></auteur>
    <auteur><nom>Briand</nom><prenom>Aristide</prenom></auteur>
  </auteurs>
  <sections>
    <section>
       <titre>Section 1</titre>
       <chapitres>
           <chapitre>
               <titre>Chapitre 1</titre>
               <paragraphes>
                       <paragraphe>Premier paragraphe</paragraphe>
                       <paragraphe>Deuxième paragraphe</paragraphe>
               </paragraphes>
           </chapitre>
           <chapitre>
               <titre>Chapitre 2</titre>
               <paragraphes>
                       <paragraphe>Premier paragraphe</paragraphe>
                       <paragraphe>Deuxième paragraphe</paragraphe>
               </paragraphes>
           </chapitre>
       </chapitres>
   </section>

   <section>
       <titre>Section 2</titre>
       <chapitres>
           <chapitre>
               <titre>Chapitre 1</titre>
               <paragraphes>
                       <paragraphe>Premier paragraphe</paragraphe>
                       <paragraphe>Deuxième paragraphe</paragraphe>
               </paragraphes>
           </chapitre>
           <chapitre>
               <titre>Chapitre 2</titre>
               <paragraphes>
```

```
                    <paragraphe>Premier paragraphe</paragraphe>
                    <paragraphe>Deuxième paragraphe</paragraphe>
                </paragraphes>
            </chapitre>
        </chapitres>
    </section>
  </sections>
</livre>
```

Nous avons fait le choix de créer des balises supplémentaires telles que `auteurs`, `sections`, `chapitres`, `paragraphes` pour éviter de mélanger des ensembles distincts, comme le titre. Cela présente l'avantage de créer des blocs homogènes (tels que les auteurs, les sections, les chapitres…).

Exercice 2

```
<?xml version="1.0" encoding="iso-8859-1"?>
<livre titre="Mon livre">
  <auteurs>
   <auteur nom="Brillant" prenom="Alexandre"/>
   <auteur nom="Briand" prenom="Aristide"/>
  </auteurs>
  <sections>
   <section titre="Section 1">
       <chapitre titre="Chapitre 1">
           <paragraphe>Premier paragraphe</paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
       </chapitre>
       <chapitre titre="Chapitre 2">
           <paragraphe>Premier paragraphe</paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
       </chapitre>
   </section>
   <section titre="Section 2">
       <chapitre titre="Chapitre 1">
           <paragraphe>Premier paragraphe</paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
       </chapitre>
       <chapitre titre="Chapitre 2">
           <paragraphe>Premier paragraphe</paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
       </chapitre>
   </section>
  </sections>
</livre>
```

Comme l'élément `titre` disparaît au profit de l'attribut, nous pouvons alléger notre structure en éliminant les blocs superflus, comme `chapitres` ou `paragraphes`.

Exercice 3

```
<livre>
…
       <chapitre titre="Chapitre1">
           <paragraphe>&lt;element id="10"&gt;&amp;gt;&lt;/element&gt;</paragraphe>
           <paragraphe><![CDATA[<element id="10">&gt;</element>]]></paragraphe>
       </chapitre>
   </section>
</livre>
```

Exercice 4

```
<p1:livre titre="Mon livre" xmlns:p1="http://www.masociete.com"
➥xmlns:p2="http://www.monentreprise.com">
<p1:auteurs>
    <p1:auteur nom="nom1" prenom="prenom1"/>
    <p1:auteur nom="nom2" prenom="prenom2"/>
</p1:auteurs>
<p1:sections>
    <p1:section titre="Section1">
        <p1:chapitre titre="Chapitre1">
           <p1:paragraphe>Premier paragraphe</p1:paragraphe>
           <p1:paragraphe>Deuxième paragraphe</p1:paragraphe>
        </p1:chapitre>
    </p1:section>
    <p2:section titre="Section2">
        <p2:chapitre titre="Chapitre1">
           <p2:paragraphe>Premier paragraphe</p2:paragraphe>
           <p2:paragraphe>Deuxième paragraphe</p2:paragraphe>
        </p2:chapitre>
        <p2:chapitre titre="Chapitre2">
           <p2:paragraphe>Premier paragraphe</p2:paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
        </p2:chapitre>
    </p2:section>
</p1:sections>
</p1:livre>
```

Il y a plusieurs combinaisons possibles en fonction de l'utilisation du préfixe ou de l'espace de noms par défaut.

Exercice 5

```
<livre titre="Mon livre" en:titre="mybook" xmlns="francais" xmlns:fr2="francais"
➥xmlns:en="anglais">
<auteurs>
   <auteur nom="nom1" prenom="prenom1"/>
   <auteur nom="nom2" prenom="prenom2"/>
```

```
</auteurs>
<sections>
   <section fr2:titre="Section1" en:titre="Section1">
       <chapitre titre="Chapitre1" en:titre="Chapter1">
           <paragraphe>Premier paragraphe</paragraphe>
           <en:paragraphe>First paragraph</en:paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
           <en:paragraphe>Second paragraph</en:paragraphe>
       </chapitre>
   </section>
   <section titre="Section2" en:titre="Section2">
       <chapitre titre="Chapitre1" en:titre="Chapter1">
           <paragraphe>Premier paragraphe</paragraphe>
           <en:paragraphe>First paragraph</en:paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
           <en:paragraphe>Second paragraph</en:paragraphe>
       </chapitre>
       <chapitre titre="Chapitre2" en:titre="Chapter2">
           <paragraphe>Premier paragraphe</paragraphe>
           <en:paragraphe>First paragraph</en:paragraphe>
           <paragraphe>Deuxième paragraphe</paragraphe>
           <en:paragraphe>Second paragraph</en:paragraphe>
       </chapitre>
   </section>
</sections>
</livre>
```

Le choix d'utiliser un préfixe pour désigner une langue est discutable mais tout à fait opérationnel. Notre document garde toujours la même structure quelle que soit la langue et on peut déplacer une partie de l'ouvrage sans avoir à répéter l'opération pour chaque traduction.