Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2021 - 2022
WEBR3 / WEBR4 - XML

Introduction à XML

Document XML

Ce document introduit les notions de document XML : balise, attribut et autres concepts de base. Il se termine par un exercice mettant en œuvre les notions présentées.

Le contenu de ce TD est en partie inspiré de :

- *wikipedia* : https://fr.wikipedia.org/wiki/Extensible_Markup_Language
- *w3school* : https://www.w3schools.com/xml/
- *Beginning XML, 5th Edition* : https://www.wrox.com/WileyCDA/WroxTitle/Beginning-XML-5th-Edition.productCd-1118162137.html
- *Essential XML Quick Reference : A Programmer's Reference to XML, XPath, XSLT, XML Schema, SOAP, and More* : https://www.pearson.com/store/p/essential-xml-quick-reference-a-programmer-s-reference-to-xml-xpath-xslt-xml-sc P100000683470/9780201740950

en accord avec leur règle pour un usage raisonnable (*fair use*).

Premier tutoriel Structurez vos données avec XML (pour commencer en douceur, un peu léger) : https://tutoriel-xml.rolandl.fr/.

Version du document : 17 septembre 2021

1 Premiers exemples

XML est l'acronyme d'*eXtensible Markup Language*. Il s'agit de définir des fichiers à la fois lisible facilement par les êtres humains et par les machines. Pour atteindre le premier objectif, on recourt au format textuel plutôt que binaire. Le second repose sur l'utilisation d'une grammaire stricte.

Il y a essentiellement deux types d'utilisation des fichiers XML : les fichiers pour le stockage de données (*data-centric*) et les documents avec l'ajout de métadonnées (*document-centric*).

1.1 Fichier XML orienté données

Voici un exemple de fichier orienté données :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Fred</to>
  <from>Anne</from>
  <subject>XML</subject>
  <content>ça avance ces TDs ?</content>
</note>
```

1.2 Fichier XML orienté document

Voici un fichier orienté document :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- https://fr.wikipedia.org/wiki/Casimir_(personnage) -->
<bio>
    <p><character>Casimir</character> est un personnage de fiction
    français créé par <author><firstName>Yves</firstName>
    ↪  <name>Brunier</name></author>
    et <author><firstName>Christophe</firstName>
    ↪  <name>Izard</name></author>
    dans les années <year>1970</year>.</p>
    <p>Le personnage apparaît pour la première fois à la télévision le
    <date>16 septembre 1974</date> sur la troisième chaîne couleur de
    l'<tvChannel>ORTF (C3)</tvChannel> dans l'émission télévisée pour
    enfants <program>L'Île aux enfants</program>, puis sur
    <tvChannel>TF1</tvChannel> jusqu'au <date>30 juin 1982</date>.</p>
</bio>
```

2 Éléments constitutifs

Cette section détaille les différents constituants d'un fichier XML.

2.1 Prologue

Un document XML *peut* commencer par une ligne appelée *prologue* ou *déclaration XML* (*XML declaration*). Le prologue est facultatif. S'il est présent, il *doit* être placé

en première ligne du fichier.

Voici un exemple de prologue :

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

Les versions possibles sont 1.0 et 1.1. L'encodage est, sauf rares exceptions, toujours de l'UTF-8.

On en trouve dans les deux fichiers de la section 1.

2.2 Commentaire

Il est possible de mettre des commentaires (*comments*) dans un document XML. Les commentaires commencent par la suite de caractères `<!--` et terminent par `-->`. Tout ce qui se trouve entre ces délimiteurs est ignoré par le processeur (*parser*) XML. Il est cependant interdit de placer la séquence de caractères `--` au sein d'un commentaire.

Voici un tel commentaire :

```
<!-- Casimir et la grotte mystérieuse -->
```

On en trouve un exemple dans le fichier de la section 1.2.

2.3 Éléments

Les éléments (*elements*) commencent par une *balise ouvrante* (*opening tag*) et terminent par une *balise fermante* (*closing tag*). Entre elles, on trouve le contenu de l'élément. Celui-ci est ordonné. On y peut y trouver des éléments, des instructions de traitement, des commentaires, des sections CDATA ou du texte.

2.3.1 Règles de nommage

Les noms des éléments, c'est-à-dire les textes servant de balises ouvrante et fermante, doivent respecter certaines règles pour que le fichier XML puisse être qualifié de bien formé. En voici les principales :

— ils commencent par une lettre ou un blanc souligné (*underscore*) ;

— ils sont constitués ensuite par un ensemble de lettres, blancs soulignés, chiffres et tirets (*hyphen*) ;

— ils ne peuvent commencer par la suite de lettres `xml` dans tous les arrangements de casse possibles (`xmL`, `XML`, etc.).

Soit l'élément de nom `element` :

— sa balise ouvrante commence par le caractère `<` directement suivi par le nom de l'élément, à la casse près, puis, éventuellement, des espaces et enfin le caractère `>` : cela donne par exemple `<element>` ;

— sa balise fermante commence par les caractères `</` directement suivi par le nom de l'élément, à la casse près, puis, éventuellement, des espaces et enfin le caractère `>` : cela donne par exemple `</element >`.

2.3.2 Élément vide

Un élément vide n'a pas de contenu. On peut l'obtenir avec un couple de balises comme dans :

```
<vide></vide>
```

Voici un exemple d'élément non vide :

```
<nonVide> </nonVide>
```

On peut alternativement produire un élément vide de contenu à l'aide d'une balise *auto-fermante* (*self-closing tag*) :

```
<vide/>
```

2.3.3 Imbrication

Les éléments peuvent s'imbriquer. Dans ce cas, les balises ouvrantes et fermantes ne peuvent se chevaucher. Un élément doit être complètement imbriqué dans son parent : il commence et termine au sein de son parent.

Voici un exemple valide d'imbrication d'éléments :

```
<parent><enfant></enfant></parent>
```

Voici une imbrication mal formée :

```
<parent><enfant></parent></enfant>
```

2.3.4 Racine

Tout document XML contient un élément particulier, son *élément racine* (*root element*), appelé aussi parfois *élément document* (*document element*). La particularité de cet élément est qu'il ne possède pas d'élément parent. Un fichier XML bien formé ne contient qu'*une seule racine*.

La racine du fichier de la section 1.1 est identifiée par les balises `<note>...</note>`. Celle du fichier de la section 1.2 par les balises `<bio>...</bio>`.

2.3.5 Attribut

Voici un nouvel exemple de document XML :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <notes>
3    <note id="_1">
4      <from>Anne</from>
5      <to>Fred</to>
6      <subject>XML</subject>
7      <content style="plaintext" lang="fr">ça avance ces TDs ?</content>
8    </note>
9    <note id="_2">
10     <from>Fred</from>
11     <to>Anne</to>
12     <subject>XML</subject>
```

```
13      <content style="plaintext" lang="fr">je m'y mets ce soir !</content>
14    </note>
15    <note id ="_0"/>
16  </notes>
```

Dans cet exemple, la racine `<notes>` contient deux éléments `<note>`. Chacun de ces éléments possède un *attribut* nommé `id`. Sa valeur est de `_1` pour la première occurrence de `<note>` et de `_2` pour la seconde. Par ailleurs, l'élément `<content>` possèdent 2 attributs : `style` et `lang`.

Nommage Les règles de nommage de attributs sont identiques à celles des éléments.

Valeur Tout attribut doit avoir une valeur.

Voici un exemple mal formé :

```
<element attribut >ko</element >
```

La valeur d'un attribut est renseignée entre guillemets `"` ou entre apostrophes `'` juste après le caractère `=` lui-même juste après le nom de l'attribut. Dans le cas où un attribut prend plusieurs valeurs, elles sont séparées par un espace.

Voici des exemples d'attributs avec leurs valeurs :

```
<element attr1="val1" attr2='v2 v3 v4' attr3="l'arbre"/>
```

Ordre L'ordre des attributs, contrairement à celui des éléments, ne peut être imposé.

Ainsi l'élément :

```
<e a1="v1" a2="v2">ça avance ces TDs ?</e>
```

est totalement équivalent à :

```
<e a2="v2" a1="v1">ça avance ces TDs ?</e>
```

Unicité Un attribut donné pour un élément donné doit être unique.

Ainsi, la construction :

```
<e a="v1" a="v2">ça avance ces TDs ?</e>
```

n'est pas syntaxiquement correct, contrairement à celle-ci :

```
<e a="v1 v2">ça avance ces TDs ?</e>
```

Exercice 1 : Élément ou attribut ?

L'information se trouvant dans le document de la section 2.3.5 pourrait aussi être structuré comme suit :

Référence d'entité	Caractère correspondant
<	<
>	>
&	&
'	'
"	"

TABLE 1 – Références d'entité prédéfinies.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<notes>
  <note id="_1" from="Anne" to="Fred" subject="XML">
    <content style="plaintext" lang="fr">ça avance ces TDs ?</content>
  </note>
  <note id="_2" from="Fred" to="Anne" subject="XML">
    <content style="plaintext" lang="fr">je m'y mets ce soir !</content>
  </note>
</notes>
```

Dans ce dernier document nous avons préféré utiliser des attributs à la place de certains éléments.

1. Peut-on toujours remplacer un élément par un attribut ?
2. Peut-on toujours remplacer un attribut par un élément ?
3. Donnez quelques règles simples afin de choisir entre attribut et élément lors de l'élaboration d'un document XML. N'hésitez pas à utiliser Internet.

2.3.6 Contenu et références d'entité

Certains caractères ne peuvent pas apparaître dans les textes des éléments ou des valeurs d'attributs car ils sont interprétés par l'analyseur (*parser*) XML. Il s'agit, pour ce qui concerne les textes, des caractères `<` et `&`, et, éventuellement, des caractères `'` et `"` comme valeurs d'attribut. Si ils doivent néanmoins apparaître, on utilise les *références d'entités* (*entity references*).

Il existe 5 entités prédéfinies. Une référence d'entité est obtenue en préfixant l'entité d'une esperluette `&` (*ampersand*) et en la suivant d'un point-virgule `;` (*semicolon*). Les 5 références d'entité prédéfinies sont renseignées à la TABLE 1.

Ainsi, la construction mal formée :

```xml
<condition> x < 17 </condition>
```

est remplacée par celle-ci, bine formée :

```xml
<condition> x &lt; 17 </condition>
```

2.4 Instructions de traitement

Il est possible de renseigner dans un fichier XML des instructions de traitement (*processing instructions*) à la destination des programmes de manipulation du fichier. On en verra des exemples lors des transformations de fichiers XML.

2.5 Section CDATA

Si on désire insérer des portions de textes qui ne doivent pas être interprétées par le *parser* XML mais doit être considéré comme du texte brut, on peut utiliser une section CDATA (*character data*).

Une telle section XML peut dès lors contenir n'importe quel texte sans qu'il soit nécessaire d'échapper les caractères spéciaux XML : `< > &`.

Une telle section commence par la suite de caractères `<![CDATA[` et se termine par `]]>`. Elle peut, par exemple, être utilisée pour contenir du code informatique (par exemple JavaScript) :

```
<script type="text/javascript">
  <![CDATA[
    if (chars > 280 && mode == tweet) {
          div.innerHTML = '<b>Attention !</b>, la limite est dépassée !';
    }
  ]]>
</script>
```

Tout comme les commentaires, ces sections ne sont pas traitées par les processeurs XML. Mais, contrairement aux commentaires, le processeur a accès aux contenus des sections d'échappement.

3 Représentation sous forme d'arbre

Un document XML être représenté sous la forme d'un arbre.

L'élément racine du fichier XML correspond au *nœud racine* de l'arbre. À chaque élément du fichier correspond un *nœud élément*. Les textes qui apparaissent comme valeur d'éléments sont représentés par les *nœuds textes* de l'arbre. Ces derniers n'ont pas de nœud enfant. Ce sont donc les feuilles de l'arbre. En toute généralité, un élément vide est également une feuille.

Par exemple pour le document de la section 1.1, l'arbre correspondant est représenté à la Fig. 1. Il est à remarquer que si on désire une représentation stricte du document XML, incluant tous les espaces et retours à la ligne, il manque 5 nœuds textes, tous enfants de la racine. Il s'agit des retours à la lignes, souvent suivis d'espaces, après la balise ouvrante `<note>` puis après les balises fermantes `</to>`, `</from>`, `</subject>` et `</content>`.

3.1 Exercice 2

Représentez sous forme d'arbre le document XML `arbre.xml` disponible sur poESI.

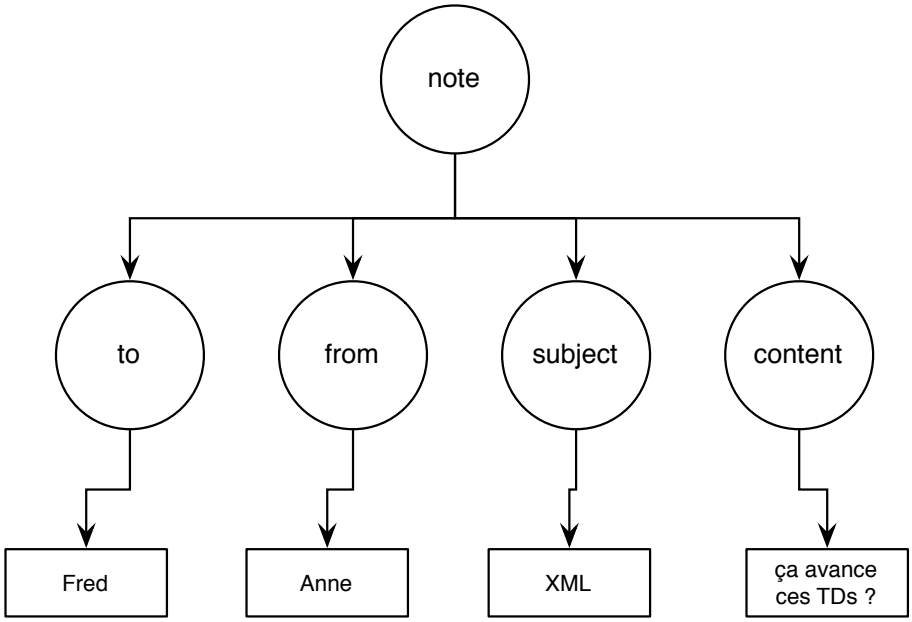

FIGURE 1 – Représentation sous la forme d'un arbre d'un fichier XML.

3.2 Exercice 3

Représentez sous la forme d'un arbre le document XML de la section 1.2.

4 Espace de noms

On rencontre souvent des éléments portant le même nom, c'est-à-dire utilisant les mêmes *tags*. Dans l'exemple ci-dessous les éléments `table` ne sont pas tous semblables :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <table>
    <tr>
      <td>Apples</td>
      <td>Bananas</td>
    </tr>
  </table>
  <table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
  </table>
</root>
```

Le premier représente une table formatée en html, le second contient une description d'un meuble (une table).

Afin d'éviter de les confondre les uns avec les autres, on définit les éléments de même

nom à l'intérieur d'un *espace de noms* ou *espace de nommage* (*namespace*).

Comme le but des espace de nommage est d'éviter les conflits de noms, il est vivement recommandé d'utiliser une URI (*Uniform Resource Identifier*) comme espace de noms, mais techniquement, on pourrait utiliser n'importe quelle chaîne de caractères.

4.1 Espace de noms préfixé

Une première approche consiste en la définition de l'espace de noms en même temps que d'un préfixe qui lui est associé :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <h:table xmlns:h="http://www.w3.org/TR/html4/">
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>
  <f:table xmlns:f="https://www.w3schools.com/furniture">
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

Dans cet exemple, on définit les espaces de noms :

— `http://www.w3.org/TR/html4/` de préfixe `h` ;

— `https://www.w3schools.com/furniture` de préfixe `f`.

Pour les préfixes, on a choisi `h` et `f`, mais on peut tout aussi bien utiliser autre chose. Par exemple, le document suivant est tout à fait équivalent à l'exemple précédent :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <brol:table xmlns:brol="http://www.w3.org/TR/html4/">
    <brol:tr>
      <brol:td>Apples</brol:td>
      <brol:td>Bananas</brol:td>
    </brol:tr>
  </brol:table>
  <chose:table xmlns:chose="https://www.w3schools.com/furniture">
    <chose:name>African Coffee Table</chose:name>
    <chose:width>80</chose:width>
    <chose:length>120</chose:length>
  </chose:table>
</root>
```

On peut définir un espace de noms et son préfixe sur n'importe quel élément.

La portée (*scope*) de l'espace de noms va de l'élément où il est défini à l'ensemble des éléments que celui-ci contient. Identiquement, son préfixe est utilisable sur l'élément

où il est défini et sur tous les éléments qu'il contient. C'est une erreur que d'utiliser un préfixe d'espace de nommage en dehors de la portée de son espace de nommage.

Par ailleurs, définir un espace de noms avec préfixe pour un élément donné *ne place pas* cet élément, ni son contenu, dans cet espace de noms. Il faut explicitement utiliser le préfixe sur les balises ouvrante et fermante d'un élément pour le placer dans l'espace de noms associé.

Placer un élément dans un espace préfixé de noms ne place pas non plus ses attributs dans cet espace de noms. Si on désire qu'un attribut soit mis dans un espace de nommage préfixé, il faut explicitement précéder son nom du préfixe voulu. Dans la mesure où les attributs d'un élément sont uniques, on y recourt rarement, sauf si on désire réutiliser des éléments XML définis par ailleurs dans un espace de nommage précis. Un exemple de ce dernier cas de figure est l'utilisation de l'attribut standard `xml:lang` qui se trouve dans l'espace de noms XML dont le préfixe est `xml`. Cet espace de noms et ce préfixes sont standards : il n'y a pas à s'en occuper, le *parser* XML les connaît.

L'exemple suivant est équivalent au précédent :

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:h="http://www.w3.org/TR/html4/"
      xmlns:f="https://www.w3schools.com/furniture">
  <h:table>
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>
  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

Dans le document ci-dessus, les espaces de noms sont définis sur l'élément racine... qui ne se trouve dans aucun espace de noms [1].

Le nom d'un élément (ou d'un attribut) muni d'un préfixe est appelé *nom qualifié* (*qualified name, QName*) de l'élément (ou de l'attribut). La partie de nom après le préfixe et les deux-points et dénommée *nom local* (*local name*).

4.2 Espace de noms par défaut

L'utilisation d'espace de noms peut rendre le document plus verbeux. Pour cette raison, le standard XML a prévu de permettre de définir un *espace de noms par défaut* (*default namespace*). Ceci est obtenu en n'associant *aucun préfixe* à l'espace de nom lors de sa définition. Tout élément égal ou contenu dans celui où l'espace de noms

1. On parle d'espace de noms vide (*empty namespace*) ou d'espace de noms nul (*null namespace*).

par défaut est défini et dont les *tags* ne portent pas de préfixe est automatiquement placé dans cet espace de noms :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.w3.org/TR/html4/"
      xmlns:f="https://www.w3schools.com/furniture">
  <table>
    <tr>
      <td>Apples</td>
      <td>Bananas</td>
    </tr>
  </table>
  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

Dans cet exemple, l'espace de noms par défaut est `http://www.w3.org/TR/html4/`. Il est défini sur l'élément racine du document. Dès lors, la racine et tous les éléments du documents sont placés dans de *default namespace* lorsque leurs balises ne sont pas préfixées. Comme c'est le cas de la racine, ce dernier document n'est pas équivalent à ceux de la section 4.1 pour lesquels la racine était hors de tout espace de nommage.

L'utilisation d'un *default namespace* peut éventuellement nuire à la lisibilité du document car l'appartenance d'un élément donné à un espace de nom précis n'est plus explicitement indiquée.

Par ailleurs, les attributs ne sont jamais placé dans l'espace de noms par défaut. Soit on les préfixe et ils sont placés dans l'espace de nommage de ce préfixe, soit ils ne sont pas préfixés et ils ne sont dans aucun *namespace*.

5 Document bien formé

Un document XML est dit bien formé s'il est conforme aux règles de la syntaxe XML :

- si il est doté d'un prologue, celui-ci est la première séquence de caractères du fichier XML ;
- il doit avoir un élément racine et celui-ci est unique ;
- à toute balise ouvrante correspond une balise fermante ;
- les balises XML sont sensibles à la casse ;
- tous les éléments sont emboîtés correctement ;
- tout attribut possède une valeur, son nom et sa valeur sont séparés par un `=` ;
- toutes les valeurs d'attributs sont entre `"` ou `'` ;
- les entités XML doivent être utilisées pour les caractères spéciaux ;
- il existe quelques contraintes de nommage pour les éléments et les attributs : par exemple, un nom ne peut jamais commencer par `xml`, `XML`, `Xml`, etc.

Il existe divers outils pour tester la bonne forme d'un document XML. Plusieurs éditeurs de texte le permettent.

L'utilitaire en ligne de commande `xmllint` [2] peut également être utilisé à cette fin. Pour tester la bonne forme du fichier `file.xml`, exécuter :

```
xmllint --noout file.xml
```

Si le fichier est bien formé, aucun affichage n'est produit.

Exercice 4 : cherchez l'erreur

Une série de documents XML mal formés se trouve sur poESI : `xx_error.xml`.

Pour chacun de ces fichiers expliquez l'erreur et corrigez-la.

6 Exercice récapitulatif

Dans l'éditeur XML de votre choix écrivez un document XML « bien formé » mettant en évidence la structure des cours à l'ESI comme elle est présentée sur : https://heb-esi.github.io/he2besi-web/online/grilles.html

Sous la racine du document il y a 4 éléments :

- Les enseignants : pour chaque enseignant on a son trigramme (identifiant), son nom et son email.
- Les quadrimestres : pour chacun des 6 quadrimestres, un identifiant (Q1, Q2, Q3, etc.), et pour chacun de ces quadrimestres son nom en français (« Premier quadrimestre », etc.).
- Les sections : pour les 3 sections, leur identifiant (R, G ou I) et leur nom (« Télécommunications et réseaux », etc.).
- Les unités d'enseignement (UEs) : avec l'identifiant (WEBR4), le coordinateur (NVS), la section (R), le quadrimestre (4) et le nombre de crédits ECTS (5).
  On a également chacune des activités d'apprentissage (AAs) qui les constituent : les identifiants (DIN, XML), les nombres d'heures (48, 24), les listes d'enseignants (ARO, NVS, SRV ; NVS, TNI), les noms (« Développement Internet », « Introduction à XML ») et les objectifs des AAs (« Réalisation d'une application web [...] », « Avoir une vue de la constellation XML. [...] »).

2. http://xmlsoft.org/xmllint.html (consulté le 5 février 2020).