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

2021 - 2022
WEBR3 / WEBR4 - XML

Introduction à XML

DTD

Ce document introduit la notion de *Document Type Definition* (DTD). Il est parsemé d'exercices mettant en œuvre les notions présentées.

Le contenu de ce TD est en partie inspiré de :

- *wikipedia* : https://fr.wikipedia.org/wiki/Document_type_definition
- *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 : 23 septembre 2021

1 Structure et contenu d'un document XML

Dans le TD *Document XML*, on a vu que pour être bien formé (*well-formed*), un fichier XML doit respecter une série de règles relatives à sa *structure*.

Voici un exemple de document XML bien formé :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- message_nodtd.xml -->
<message>
    <to>Fred</to>
    <from>Anne</from>
    <subject>XML</subject>
    <content>Hello World!</content>
</message>
```

Au delà de la vérification de la *grammaire* XML, il est possible de vérifier que le *contenu* d'un document XML respecte un *vocabulaire* donné. C'est ce qui s'appelle la *validation* d'un document XML. Un document XML bien formé peut dès lors être valide ou non valide.

Il existe divers standards pour spécifier le contenu attendu d'un document XML. Nous étudions la DTD (*Document Type Definition*) dans ce TD. Les schémas XML (*XML Schemas*) sont abordés dans un TD suivant.

Validation par DTD Il existe divers outils pour valider un document XML selon une DTD. Plusieurs éditeurs de texte le permettent.

L'utilitaire en ligne de commande `xmllint` [1] peut également être utilisé à cette fin. Pour valider le fichier `file.xml` au regard de la DTD qu'il renseigne (voir la section 2), exécuter :

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

Pour valider le fichier `file.xml` au regard de la DTD `file.dtd`, exécuter :

```
xmllint --noout --dtdvalid file.dtd file.xml
```

Dans les deux cas, si tout se passe bien lors de la validation du fichier XML, aucun affichage n'est produit.

2 Premières DTD

On peut écrire la DTD directement dans le document XML ou dans un fichier séparé. Dans ce dernier cas, il possible, mais pas obligatoire, de lier cette DTD aux fichiers XML auxquels elle s'applique.

Cette section propose un premier contact avec la syntaxe d'une DTD. Les détails sont fournis dans la section 3.

1. http://xmlsoft.org/xmllint.html (consulté le 12 février 2020).

2.1 DTD interne

Voici un document XML dont le contenu est identique à celui de la section 1, mais muni d'une DTD interne (*internal DTD*) :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- message_internaldtd.xml -->
<!DOCTYPE message [
    <!ELEMENT message (to, from, subject, content)>
    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT subject (#PCDATA)>
    <!ELEMENT content (#PCDATA)>
]>
<message>
    <to>Fred</to>
    <from>Anne</from>
    <subject>XML</subject>
    <content>Hello World!</content>
</message>
```

La déclaration d'une DTD se trouve en tête du document XML, ou juste après son prologue :

```
<!DOCTYPE message [
```

Elle se termine par :

```
]>
```

Le nom de l'élément racine du document, exactement sous la même forme que dans le document XML, préfixe d'espace de noms compris, doit suivre `DOCTYPE`. On trouve ensuite la définition du contenu auquel le document XML doit se conformer pour être valide.

2.2 DTD externe

Il est plus « propre » de placer la DTD dans un fichier externe. Voici le fichier `message.dtd` :

```
<!-- message.dtd -->
<!ELEMENT message (to, from, subject, content)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT content (#PCDATA)>
```

2.2.1 Identifiant système

Pour indiquer que la DTD du document ayant pour racine l'élément `message` se trouve dans le fichier `message.dtd`, on utilise le mot-clé `SYSTEM`. Voici ce que cela donne avec le fichier `message.xml` :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- message.xml -->
<!DOCTYPE message SYSTEM "message.dtd">
<message>
    <to>Fred</to>
    <from>Anne</from>
    <subject>XML</subject>
    <content>Hello World!</content>
</message>
```

La DTD est ici locale, elle se trouve sur le même système de fichier que le document XML. Son emplacement doit être renseigné par une *Uniform Resource Identifier*[2] (URI). Il pourrait également s'agir d'un fichier sur un réseau.

Exercice 1 Modifiez la déclaration `DOCTYPE` de `message.xml` comme suit :

```xml
<!DOCTYPE test SYSTEM "message.dtd">
```

Quelle erreur obtenez-vous lors de sa validation par `message.dtd` ?

Exercice 2 Remettez `message.xml` dans son état d'origine. Modifiez la première déclaration de contenu de `message.dtd` en :

```xml
<!ELEMENT test (to, from, subject, content)>
```

Quelle erreur obtenez-vous lors de la tentative de validation de `message.xml` au regard de `message.dtd` ?

2.2.2 Identifiant public

Il est possible de faire référence à une DTD publique :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- message_externalpublicdtd.xml -->
<!DOCTYPE message PUBLIC "-//HE2B ESI WEBR4//XML DTD//FR" "message.dtd">
<message>
    <to>Fred</to>
    <from>Anne</from>
    <subject>XML</subject>
    <content>Hello World!</content>
</message>
```

Le mot-clé `PUBLIC` est suivi par un identifiant public. Ici on a utilisé un *Formal Public Identifier*[3] (FPI). On a également fourni une URI — c'est facultatif — au cas où le *parser* ne trouve pas le fichier DTD à l'aide du FPI.

2. https://fr.wikipedia.org/wiki/Uniform_Resource_Identifier.
3. https://fr.wikipedia.org/wiki/Formal_Public_Identifier.

2.3 DTD interne et externe

Il est possible de mélanger DTD interne et externe. Cela permet d'utiliser et d'étendre ou de redéfinir [4] des déclarations externes.

Voici un exemple d'utilisation et extension de la DTD externe `message.dtd` utilisée en section 2.2 :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- message_alt_order.xml -->
<!DOCTYPE email SYSTEM "message.dtd" [
    <!ELEMENT email (from, to, subject, content)>
]>
<email>
    <from>Anne</from>
    <to>Fred</to>
    <subject>XML</subject>
    <content>Hello World!</content>
</email>
```

3 Contenu d'une DTD

Au sein d'une DTD, il est possible de définir :

- des éléments ;
- des attributs d'éléments ;
- des entités ;
- des notations.

3.1 Éléments

La structure générale d'une déclaration d'élément est :

```
<!ELEMENT name (content)>
```

où `name` est le nom de l'élément et `content` son contenu.

3.1.1 Nom

Dans sa déclaration, le nom de l'élément doit apparaître *exactement* comme il apparaît dans le document XML. Ceci comprend un éventuel préfixe d'espace de nommage. Dès lors, lorsqu'on utilise une DTD à des fins de validation, les noms de préfixes associés aux espaces de nommage sont fixés dans la DTD. Cette *grosse limitation* des DTD s'explique par la standardisation des espaces de noms *postérieurement* à celle des DTD. Les DTD sont antérieures au standard XML. La différence de syntaxe entre DTD et XML l'indique.

4. Seules les déclarations d'attributs (section 3.2) ou d'entités (section 3.3) peuvent être redéfinies.

3.1.2 Contenu

La déclaration de contenu d'un élément XML peut consister en :

- du texte ;
- un ou plusieurs autres éléments ;
- un mélange de texte et d'éléments ;
- rien ;
- n'importe quoi.

Texte Pour indiquer qu'un élément ne contient que du texte, on déclare :

```
<!ELEMENT subject (#PCDATA)>
```

où `PCDATA` vaut pour *parsable character data*. Il s'agit de caractères que le *parser* doit *parser*.

Éléments Plusieurs cas se présentent lorsqu'un élément contient un ou plusieurs éléments, mais pas de texte.

Élément unique Pour indiquer que l'élément `parent` contient un unique élément nommé `child`, on déclare :

```
<!ELEMENT parent (child)>
```

L'élément `child` doit également être déclaré pour que la DTD soit légale.

Liste d'éléments Pour indiquer que l'élément `parent` contient un ensemble ordonné d'éléments, on déclare :

```
<!ELEMENT parent (child1, child2, child3)>
```

Les éléments `child1`, `child2` et `child3` doivent également être déclarés pour que la DTD soit légale. L'ordre dans lequel ils apparaissent dans la déclaration de l'élément `parent` est celui dans lequel ils doivent apparaître dans le document XML. Chacun doit apparaître une et une seule fois sous l'élément `parent`.

Exercice 3 Reprenez les fichiers `message.xml` et `message.dtd` de la page 3. Après avoir échangé, dans `message.xml`, les éléments `to` et `from`, quelle erreur obtenez-vous lors de la validation ?

Exercice 4 Remettez le fichiers `message.xml` dans son état de base valide. Quelle erreur obtenez-vous lors de sa validation après avoir permuté les éléments `to` et `from` dans la déclaration de contenu de l'élément `message` dans `message.dtd` ?

Modificateur	Multiplicité
aucun	1
*	$[0, \infty[$
+	$[1, \infty[$
?	$\{0, 1\}$

TABLE 1 – Modificateur de multiplicité.

Choix d'éléments Pour indiquer que l'élément `parent` contient un élément parmi un ensemble d'éléments, on déclare :

```
<!ELEMENT parent (child1 | child2 | child3)>
```

Les éléments `child1`, `child2` et `child3` doivent également être déclarés pour que la DTD soit légale. Un et un seul d'entre eux doit apparaître sous l'élément `parent` dans le document XML.

Il est possible de construire des contenus complexes comme dans :

```
<!ELEMENT parent (child1, (child2 | child3))>
```

Cette déclaration impose que l'élément `parent` possède un élément `child1` suivi soit d'un élément `child2`, soit d'un élément `child3`.

Exercice 5 Interprétez la déclaration :

```
<!ELEMENT parent (child1 | (child2, child3))>
```

Quantification Les déclarations présentée jusqu'ici imposaient l'existence d'un élément unique sous un élément parent. Il est possible d'exercer des contraintes plus riches sur les nombres d'occurrences. Les quantificateurs disponibles sont détaillés dans la TABLE 1. Les modificateurs qui y sont listés doivent être renseignés immédiatement après le contenu dont la multiplicité doit être modifiée.

Exercice 6 Interprétez les déclarations suivantes :

1. `<!ELEMENT parent (child1)>`
2. `<!ELEMENT parent (child1*)>`
3. `<!ELEMENT parent (child1+)>`
4. `<!ELEMENT parent (child1?)>`
5. `<!ELEMENT parent (child1, child2)>`
6. `<!ELEMENT parent (child1, child2)*>`
7. `<!ELEMENT parent (child1, child2)+>`
8. `<!ELEMENT parent (child1, child2)?>`
9. `<!ELEMENT parent (child1*, child2)>`
10. `<!ELEMENT parent (child1, child2?)>`
11. `<!ELEMENT parent (child1+, child2*)>`
12. `<!ELEMENT parent (child1?, child2)*>`

13. `<!ELEMENT parent (child1* | (child2, child3)*)>`
14. `<!ELEMENT parent (child1? | (child2*, child3))+>`
15. `<!ELEMENT parent (child1 | (child2+, child3)?)*>`
16. `<!ELEMENT parent (child1 | (child2*, child3*))?>`

Contenu mixte Pour indiquer que l'élément `parent` contient du texte *et* des éléments, en nombre et ordre indéterminés, on déclare :

```
<!ELEMENT paragraph (#PCDATA | character | author | year)*>
```

Bien évidemment, les éléments `character`, `author` et `year` doivent également être déclarés pour que la DTD soit légale.

Voici un partie de document XML valide au regard de l'extrait de DTD ci-dessus :

```
<paragraph><character>Casimir</character> est un personnage de fiction
français créé par <author>Yves Brunier</author> et <author>Christophe
Izard</author> dans les années <year>1970</year>.</paragraph>
```

Pour indiquer un contenu mixte, le mot-clé `#PCDATA` doit apparaître en premier dans la liste des éléments contenus.

Élément vide Pour indiquer que l'élément `parent` n'a aucun contenu, on déclare :

```
<!ELEMENT parent EMPTY>
```

Les éléments `<parent/>` ou `<parent></parent>` correspondent à cette définition.

Contenu quelconque Pour indiquer que l'élément `parent` possède un contenu, mais sans contrainte sur la nature de celui-ci, on déclare :

```
<!ELEMENT parent ANY>
```

On peut dès lors trouver n'importe quelle combinaison d'éléments et de texte dans l'élément `parent` *à condition* que les éléments en question soient déclarés dans la DTD.

Exercice 7 Écrivez une DTD pour le document XML `biographie.xml` reproduit ci-dessous :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- biographie.xml -->
3  <!DOCTYPE bio SYSTEM "biographie.dtd">
4  <!-- https://fr.wikipedia.org/wiki/Casimir_(personnage) -->
5  <bio>
6      <p><character>Casimir</character> est un personnage de fiction
7      français créé par <author><firstName>Yves</firstName> <name>Brunier
8      </name></author> et <author><firstName>Christophe</firstName>
9      <name>Izard</name></author> dans les années <year>1970</year>.</p>
10     <p>Le personnage apparaît pour la première fois à la télévision le
```

```
11      <date>16 septembre 1974</date> sur la troisième chaîne couleur de
12      l'<tvChannel>ORTF (C3)</tvChannel> dans l'émission télévisée pour
13      enfants <program>L'Île aux enfants</program>, puis sur
14      <tvChannel>TF1</tvChannel> jusqu'au <date>30 juin 1982</date>.</p>
15  </bio>
```

Vérifiez l'exactitude de votre fichier `biographie.dtd`.

3.2 Attributs

La structure générale d'une déclaration d'attribut d'élément est :

```
<!ATTLIST el_name att_name att_type att_value>
```

où `el_name` est le nom de l'élément, `att_name` celui de l'attribut, `att_type` le type de l'attribut et `att_value` une déclaration sur sa valeur.

Lorsqu'on désire assigner deux attributs à un élément, les deux déclarations suivantes sont équivalentes :

```
<!ATTLIST el_name att_name1 att_type1 att_value1>
<!ATTLIST el_name att_name2 att_type2 att_value2>
```

et

```
<!ATTLIST el_name att_name1 att_type1 att_value1
                  att_name2 att_type2 att_value2>
```

Ceci se généralise à un nombre quelconque d'attributs pour un élément.

3.2.1 Nom de l'élément et nom de l'attribut

Dans la déclaration d'un attribut, comme dans celle d'un élément (section 3.1.1), le nom de l'élément, mais aussi celui de l'attribut doiventt apparaître *exactement* comme ils apparaissent dans le document XML. Ceci comprend d'éventuels préfixes d'espace de nommage.

3.2.2 Type

La Table 2 reprend les types qui peuvent être donnés aux attributs d'éléments.

Voici quelques remarques supplémentaires.

Le type `CDATA` est le plus courant. Il s'agit de n'importe quels caractères sauf quelques caractères interdits qu'on peut obtenir par les (références d') entités standards (voir section 3.3).

Les valeurs d'identifiants (`ID`) ne peuvent pas commencer par un chiffre !

5. C'est-à-dire une chaîne qui respecte les règles de nommage d'un élément XML : elle ne commence pas par un chiffre, ne contient pas d'espace, etc.

Type	Valeurs possibles
`CDATA`	Des caractères quelconques (*character data*)
Énumération	Une des valeurs de l'énumération (voir section 3.2.2)
`ID`	Un identifiant unique dans le document XML
`IDREF`	La valeur d'un identifiant (`ID`) présent dans le document XML
`IDREFS`	Une séquence d'`IDREF`s séparées par des espaces
`NMTOKEN`	Un nom XML valide[5] (*name token*)
`NMTOKENS`	Une séquence de `NMTOKEN`s séparés par des espaces
`ENTITY`	Une entité non *parsée*
`ENTITIES`	Une séquence d'`ENTITY`s séparées par des espaces
`NOTATION`	Une chaîne déclarée comme une notation.

TABLE 2 – Types d'attributs.

Le type `IDREF` contraint le domaine des valeurs possibles de l'attribut. Il n'est cependant pas possible de lier un attribut de type `IDREF` à un attribut de type `ID` spécifique. On voit ainsi dans `messages.xml` (p. 11) que les attributs `ref` des éléments `other` font tantôt référence à un `id` de `message`, tantôt à un `id` d'`other`... Les schémas XML pallient à cette lacune.

Les entités non *parsées* vont au delà de la matière de ce TD. Le lecteur intéressé trouvera de quoi répondre à ses questions ici : https://docstore.mik.ua/orelly/xml/xmlnut/ch03_06.htm.

Il en va de même pour le type `NOTATION`. Davantage d'informations ici : https://xmlwriter.net/xml_guide/notation_declaration.shtml, par exemple.

Énumération Pour déclarer une énumération, on place les valeurs possibles, sans guillemets ni apostrophes, séparées par des barres verticales, l'ensemble entre parenthèses :

```
(a | 42 | -23)
```

Voici une DTD qui utilise cette énumération :

```
<!-- enumeration.dtd -->
<!ELEMENT elts (elt*)>
<!ELEMENT elt EMPTY>
<!ATTLIST elt attr (a | 42 | -23) "-23">
```

Et un document XML valide au regard de cette DTD :

```
<?xml version="1.0"?>
<!-- enumeration.xml -->
<!DOCTYPE elts SYSTEM "enumeration.dtd">
<elts>
    <elt/>
    <elt attr="42"/>
    <elt attr='-23'/>
    <elt attr='a'/>
    <!-- <elt attr="- 23"/> -->
```

Valeur	Signification
`"value"`	Attribut facultatif avec une valeur par défaut
`#IMPLIED`	Attribut facultatif sans valeur par défaut
`#REQUIRED`	Attribut obligatoire
`#FIXED "value"`	Attribut facultatif de valeur fixée

TABLE 3 – Valeurs d'attributs.

```
    <!-- <elt attr='M'/> -->
</elts>
```

3.2.3 Valeur

Après le type de l'attribut, il faut renseigner une valeur ou un caractéristique de valeur. Les quatre cas possibles sont listés dans la TABLE 3.

Voici quelques remarques supplémentaires.

La valeur par défaut ou fixée doit être donnée entre apostrophes ou guillemets.

Si dans un document XML on ne renseigne pas un attribut facultatif avec valeur par défaut ou valeur fixée, le *parser* XML associe à l'élement en question l'attribut avec la valeur par défaut ou fixée. Si l'attribut est facultatif sans valeur par défaut et qu'on ne le renseigne pas, l'élément n'est pas muni de cet attribut par le *parser*.

Si on fournit dans le document XML une valeur pour un attribut `#FIXED`, *mais* que cette valeur n'est pas celle renseignée dans la DTD, le document XML est invalide.

3.2.4 Exemple

Voici un exemple de DTD avec des éléments munis d'attributs :

```
<!-- messages.dtd -->
<!ELEMENT messages (message*, other*)>

<!ELEMENT message (to,from,subject,content)>
<!ATTLIST message id ID #REQUIRED
                  replyTo IDREF #IMPLIED>

<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT content (#PCDATA)>

<!ELEMENT other EMPTY>
<!ATTLIST other id ID #REQUIRED
                ref IDREF #REQUIRED
                version CDATA #FIXED "1.0"
                misc CDATA "(void)">
```

Voici un document XML valide pour cette DTD :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- messages.xml -->
<!DOCTYPE messages SYSTEM "messages.dtd">
<messages>
    <message id="m1">
        <to>Fred</to>
        <from>Anne</from>
        <subject>XML</subject>
        <content>Hello World!</content>
    </message>
    <message id="m2" replyTo="m1">
        <to>Anne</to>
        <from>Fred</from>
        <subject>XML</subject>
        <content>Haha !</content>
    </message>
    <!-- <message></message> -->
    <!-- <other id="m1" ref="m1"/> -->
    <!-- <other id="o0"/> -->
    <other id="o1" ref="m1"/>
    <other id="o2" ref="o1"/>
    <other version="1.0" id="o3" ref="o1"/>
    <!-- <other version ="1.1" id="o4" ref="o1"/> -->
    <other id="o5" ref="o1" misc="n'importe quoi&lt;"/>
</messages>
```

Exercice 8 Dans la DTD `messages.dtd`, l'attribut `id` de l'élément `message` est défini comme identifiant et l'attribut `replyTo` comme `IDREF`. Notez les erreurs obtenue lors de la validation du fichier `messages.xml` lorsque vous effectuez les opérations suivantes sur ses données :

1. remplacer `id="m1"` par `id="1"` ;
2. remplacer `id="m2"` par `id="m1"` ;
3. remplacer `replyTo="m1"` par `replyTo="m3"`.

3.3 Entités

On a vu dans le TD *Document XML* qu'il existe 5 entités prédéfinies : `lt`, `gt`, `amp`, `apos` et `quot`. Dans un document XML, on y fait référence en précédant l'entité par une esperluette et en la suivant par un point-virgule : `<`, `>`, `&`, `&apos;` et `&quot;`.

Outre ces entités prédéfinies, il existe divers types d'entités : les entités générales internes ou externes, les entités paramètres internes ou externes et les entités *non parsées* (*unparsed*) générales externes.

On se limite ici à la définition d'entités générales internes *non parsées* (*unparsed*). Le lecteur intéressé par les autres types d'entités peut consulter avantageusement la page : https://www.liquid-technologies.com/DTD/Structure/ENTITY.aspx.

3.3.1 Entité générale interne

Ce type d'entité permet de définir des raccourcis pour certains caractères ou texte. Dans la DTD, on a :

```
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">
```

Une fois ainsi définies, il est possible de les utiliser dans un document XML :

```
<author>&writer;&copyright;</author>
```

4 Exercices récapitulatifs

Exercice 9 Écrivez une DTD pour le document XML `esi.xml` disponible sur poESI. Votre DTD doit définir les éléments et les attributs, en prenant soin de choisir le type le plus adéquat : `CDATA`, `ID`, `IDREF`, `IDREFS`, etc.

Produisez des documents XML qui permettent de vérifier que votre DTD est correcte. Par exemple un de ces documents contient 2 fois le même identifiant. Un autre document fait une référence à un `ID` qui n'est pas présent dans le document. Ou encore un document contient un élément pour lequel un enfant obligatoire ne s'y trouve pas.

Exercice 10 Les DTD sont-elles compatibles avec les espace de noms ? Que se passe-t-il lorsqu'on change dans un document XML le préfixe d'un espace de noms ? Ce document est-il toujours valide par rapport à la DTD ?

Piste de réponse De prime abord, il faut répondre négativement à la question précédente. Cependant, l'utilisation d'entités paramètres permet de s'affranchir du problème de la prise en charge des espaces de noms et du libre choix des préfixes. Nous n'allons pas plus loin, car cela dépasse la portée de ce TD. Pour le curieux, la section 2.5.1 de *Essential XML Quick Reference* en donne une mise en œuvre effective complète.