

Un peu de rigueur ! ... Les DTD et les schémas XML

Pour qu'un fichier XML émis par un "émetteur" puisse être compris sans équivoque par le "récepteur", il faut que le récepteur en connaisse la structure. D'où la nécessité de construire un "modèle de contenu".

Rq : Un ensemble de données qui donne des informations sur d'autres données (comme une DTD ou un schéma XML) sont appelées des « métadonnées ».

DTD (Document Type Description)

Exemple de déclaration interne d'une DTD pour le fichier xml "Bottin" (cf. labo1):

```
<?xml version="1.0" ?>
```

```
<!DOCTYPE Bottin [  
  <!ELEMENT Bottin (Abonne+)>  
  <!ELEMENT Abonne (Nom, Prenom+, Prefixe*, Tel, Adresse, CodePostal)>  
  <!ELEMENT Nom (#PCDATA)>  
  <!ELEMENT Prenom (#PCDATA)>  
  <!ELEMENT Prefixe (#PCDATA)>  
  <!ELEMENT Tel (#PCDATA)>  
  <!ELEMENT Adresse EMPTY>  
  <!ATTLIST Adresse designation CDATA #IMPLIED>  
  <!ATTLIST Adresse nom CDATA #REQUIRED>  
  <!ATTLIST Adresse numero CDATA #IMPLIED>  
  <!ELEMENT CodePostal (#PCDATA)>  

```

```
<Bottin>
```

```
<!--debut du contenu du botin-->
```

```
  <Abonne>  
    <Nom>Cuvelier</Nom>  
    <Prenom>Charles</Prenom>  
    <Prefixe>064</Prefixe>  
    <Tel>263542</Tel>  
    <Adresse designation="rue" nom="du puits" numero="56"> </Adresse>  
    <CodePostal>7160</CodePostal>  
  </Abonne>  
  
  <Abonne>  
    <Nom>Dupuis etc...
```

Explications :

- Déclaration d'une DTD avec "<!DOCTYPE"> (Le "!" précise qu'il s'agit d'une déclaration). Elle suit la déclaration XML et précède le reste du document.
On retrouve ce genre de déclaration dans les page HTML pour indiquer quelle version de l'html on utilise, par exemple pour HTML4.01 :
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">` .
Pour html5 , c'est simplement `<!DOCTYPE html>`
- DOCTYPE est immédiatement suivi par le nom de la racine du document xml. (ici, c'est "Bottin").
- Ensuite viennent les nœuds de la structure, il sont déclarés par "<!ELEMENT>".
Toujours le "!" pour préciser qu'il s'agit d'une déclaration.

La syntaxe est: `< !ELEMENT NomElément (modèle de contenu)`. Si le contenu contient des enfants, ils se notent entre () séparés par des virgules -
ex: Abonne (Nom, Prenom+, Prefixe*, Tel, Adresse, CodePostal).

Des caractères spéciaux peuvent suivre le contenu :

- + signifie que l'élément peut apparaître une ou plusieurs fois dans le document
- * signifie que l'élément peut apparaître zéro ou plusieurs fois dans le document
- ? signifie que l'élément est facultatif (0 ou 1 fois)
- (none) (donc si rien indiqué) signifie que l'élément ne peut apparaître qu'une fois dans le document

Les éléments du contenu sont à leur tour définis:

ex : `<!ELEMENT Nom (#PCDATA)>` spécifie que l'élément Nom est constitué de données . (*parsed character data*) ou est vide de données: `<!ELEMENT Adresse EMPTY>`

- Les éléments peuvent être décrits par des "attributs". Ceux-ci sont déclarés comme suit:

ex: `<!ATTLIST Adresse nom CDATA #REQUIRED>`

La syntaxe est : `<ATTLIST NomElément NomAttribut Type DefaultDeclaration>`.

Avec *Type* qui peut prendre les valeurs :

CDATA : information de type caractère.

ID : clé unique (identifiant)

et quelques autres...

Avec **DefaultDeclaration** qui peut prendre les valeurs:

IMPLIED (facultatif)

REQUIRED (requis)

FIXED "value" (valeur par défaut) ex : #FIXED "000" .

"value" en combinaison avec une énumération :

("bleu"|"vert"|"rouge") "bleu" → bleu sera la valeur par défaut.

Les informations de la DTD peuvent être dans un fichier externe (.DTD) ;
dans ce cas, on doit avoir la déclaration de ce fichier dans le fichier .XML :

<!DOCTYPE Bottin SYSTEM "Bottin.dtd">

Complément d'information sur :

[http://edutechwiki.unige.ch/fr/Tutoriel/DTD#Quelques exemples d'identification de la racine et la DTD](http://edutechwiki.unige.ch/fr/Tutoriel/DTD#Quelques%20exemples%20d'identification%20de%20la%20racine%20et%20la%20DTD)

Les règles de combinaison:

On peut combiner selon les règles ci-dessous:

A et B = tags	Explication specification_contenu	Exemples
A?	A (un seul) est une option, (donc: A ou rien)	<!ELEMENT person (name,email?)>
A+	Il faut un ou plusieurs A	<!ELEMENT person (name,email+)>
A*	A est une option, il faut zéro, un ou plusieurs A	<!ELEMENT person (name,email*)>
A B	Il faut A ou B, mais pas les deux	<!ELEMENT person (email fax)>
A , B	Il faut A, suivi de B (dans l'ordre)	<!ELEMENT person (name ,email?)>
(A, B) +	Les parenthèses regroupent. Ici: un ou plusieurs (A suivi de B)	<!ELEMENT liste (name,email)+>

Éléments spéciaux

Voici comment définir un élément qui ne contient pas d'autres éléments.

Élément spéciaux	Explication specification_contenu	Exemples
#PCDATA	"Parsed Character Data" Données (non-interprétées par XML) dans le langage d'encodage courant.	<!ELEMENT email (#PCDATA)>
ANY	Mot clé qui indique que vous y insérez n'importe quel sous-élément non-défini et dans n'importe quel ordre (déconseillé). Uniquement utile dans la phase de développement.	<!ELEMENT person ANY>
EMPTY	L'élément n'a pas de contenu. C'est une balise auto-fermante comme dans 	<!ELEMENT br EMPTY>

Exercice 1 : Créer un DTD interne pour le fichier « stage » (généré en vba au labo1), imposez certaines contraintes (par exemple nom et prénom de l'étudiants requis...).

Valider le fichier xml dans le logiciel CookTop (Menu XML/ Option Valider) ou en ligne (<http://www.xmlvalidation.com>) .

Vérifiez que les contraintes fonctionnent bien en modifiant le fichier afin qu'il ne soit plus conforme (par exemple nom et prénom de l'étudiants requis,...) et valider le fichier .

Schémas XML

Historiquement, les DTD ont précédé l'apparition de XML. Elles sont issues du SGML, ancêtre de tous les langages de "marquage" (Markup Language). Elles sont toujours utilisées mais présentent certaines faiblesses. L'apparition du XML a entraîné l'arrivée d'une autre façon d'exprimer la structure d'un fichier XML en se basant sur le XML lui-même : ce sont les schémas XML.

Ainsi, un schéma XML pour le fichier "Bottin" ressemble à ceci:

```
<?xml version="1.0" ?>

<xs:schema id="Bottin" targetNamespace="http://tempuri.org/~vsF3.xsd" xmlns:msdtns="http://tempuri.org/~vsF3.xsd"
  xmlns="http://tempuri.org/~vsF3.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-
  microsoft-com:xml-msdata" attributeFormDefault="qualified" elementFormDefault="qualified">

  <xs:element name="Bottin" msdata:IsDataSet="true" msdata:Locale="fr-BE" msdata:EnforceConstraints="False">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="Abonne">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Nom" type="xs:string" minOccurs="0" />
              <xs:element name="Prenom" type="xs:string" minOccurs="0" />
              <xs:element name="Prefixe" type="xs:string" minOccurs="0" />
              <xs:element name="Tel" type="xs:string" minOccurs="0" />
              <xs:element name="CodePostal" type="xs:string" minOccurs="0" />
              <xs:element name="Adresse" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="designation" form="unqualified" type="xs:string" />
                  <xs:attribute name="nom" form="unqualified" type="xs:string" />
                  <xs:attribute name="numero" form="unqualified" type="xs:string" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element> </xs:schema>
```

Repérez dans le listing les remarques suivantes:

- Un schéma XML est donc un document XML (déclaré comme tel à la première ligne) mais il porte l'extension ".xsd".
- La racine s'appelle xs:schema ou xsd:schema. Celle-ci comporte :
- le nom de la racine du fichier xml associé : "Bottin"
- des "espaces de nom" ou "Name Space" de deux types : TargetNameSpace et XmlNameSpace (xmlns) Ces espaces de noms permettent de rendre non ambigus les noms utilisés dans le schéma.
- L'arborescence précisant les éléments (<xs:element> , leurs attributs <xs:attribute> qui sont "typés", leur structure <xs:simpleType> ou <complexType>.

A réaliser : Ouvrez le fichier "stage.xml" avec "Visual Studio Community" (on peut aussi faire cette manipulation en ligne : <https://xmlgrid.net/xml2xsd.html>) et créez le schéma grâce à l'option "XML/Créer un schéma" (peut varier en fonction de la version). On peut lier le document xml avec le schéma xsd en utilisant la syntaxe ci-dessous .

