DTD : Document Type Definition

Objectif

- Définir ce qui est autorisé, correct et compréhensible dans le contexte du document

- Il y a 2 solutions pour préciser les balises et attributs auxquels on a droit pour rédiger un document XML :
  - Les DTD (Document Type Definition)
  - Les schémas XML

Construction d'une DTD

- Les DTD fournissent les règles que le parseur (ou analyseur syntaxique = outil logiciel qui interprète le document) doit suivre pour la validation et l'interprétation correcte du document.
- Une DTD suit des règles strictes pour assurer que les résultats désirés soient obtenus et les données restituées correctement.
- Une DTD (ou un shéma XML) est requise pour qu'un document XML soit valide.

Types de DTD

- Une DTD a 2 représentations physiques possibles :
  - elle peut faire partie du document XML : interne
  - elle peut être un fichier à elle seule : externe
    - avantage : utilisable par d'autres documents XML
- Les DTD externes se séparent en 2 catégories :
  - privée : accessible en local
  - publique : disponible pour tous via un URI (Uniform Resource Identifier)

Exemple de document XML avec doctype interne

```
<?xml version = "1.0"
      encoding="UTF-8"
      standalone = "yes"?>
<!DOCTYPE CONTACTS [
<!ELEMENT CONTACTS ANY>
<!ELEMENT CONTACT (NAME, EMAIL)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT EMAIL (#PCDATA)>
<!ATTLIST CONTACT CONTACT_NUM ID #REQUIRED>
<!ATTLIST CONTACT MOTHER IDREF #IMPLIED>
]>
<CONTACTS>

   <CONTACT CONTACT_NUM = "N2">
   <NAME>Teri Mancuso</NAME>
   <EMAIL>teri@teri.com</EMAIL>
   </CONTACT>

   <CONTACT CONTACT_NUM = "N1" MOTHER = "N2">
   <NAME>Kristin Mancuso</NAME>
   <EMAIL>kristin@kristin.com</EMAIL>
   </CONTACT>

</CONTACTS>
```

Déclaration d'une DTD

- DTD interne :

```
<!DOCTYPE racine
   [déclarations des éléments, attributs,…]
 >
```

- DTD externe privée :

```
<!DOCTYPE racine SYSTEM "fichier.dtd" >
```

- DTD externe publique :

```
<!DOCTYPE racine PUBLIC "identifiant" "url">
```

exemple : <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3c.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >

Déclaration d'éléments

- La déclaration d'élément de base est structurée ainsi :

  <!ELEMENT nom_element DEF_CONTENU>

- Exemple : <!ELEMENT nom_de_famille (#PCDATA)>

- Cette déclaration indique que l'élément nom_de_famille doit contenir #PCDATA (Parsable Character Data), c'est-à-dire du texte brut ne contenant aucune balise ou sous-élément.

Déclaration d'éléments

DEF_CONTENU peut contenir :

- EMPTY : l'élément n'a pas de contenu, il est donc vide. Il peut cependant avoir des attributs.
  - <!ELEMENT elt EMPTY>
  - Utilisation : <elt/>
- ANY : l'élément contient des éléments quelconques.
  - <!ELEMENT elt ANY>
  - Vague, permet de dire qu'un élément existe sans savoir ce qu'il va contenir. Ne permet pas d'utiliser des éléments non déclarés. Besoin quand on utilise des applications associant un contenu d'applications XML inconnu de manière arbitraire.
- (#PCDATA) : l'élément contient du texte.
- (nom_element) : l'élément contient un autre élément de nom *nom_element*.
- Un ensemble d'éléments séparés par des opérateurs, le tout placé entre parenthèses.

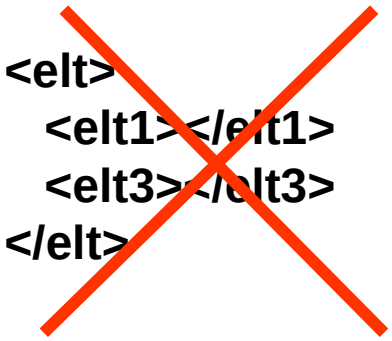
Déclaration d'éléments (suite)

- L'opérateur dans un ensemble d'éléments peut être l'opérateur de choix, représenté par le caractère | ou un opérateur de séquence représenté par le caractère ,.
- Lors de la déclaration, chaque élément de l'ensemble peut se voir attribuer une indication d'occurrence.
- Ex : <!ELEMENT elt (elt1, elt2?, elt3+, elt4*)>
- Signification :
  - pas d'indication : exactement une occurrence
  - ? : 0 ou 1 occurrence
  - + : 1 occurrence ou plus
  - * : 0 occurrence ou plus

Séquence d'éléments

- Une séquence d'éléments est la liste des éléments devant apparaître dans l'élément défini.
- Tous les éléments enfants doivent être déclarés dans la DTD. Ils doivent apparaître dans le fichier XML dans l'ordre de déclaration.
- Exemple : <!ELEMENT elt (elt1, elt2, elt3)>
- Utilisation :

```
<elt>
  <elt1></elt1>
  <elt2></elt2>
  <elt3></elt3>
</elt>
```



```
<elt>
  <elt1></elt1>
  <elt3></elt3>
</elt>
```

```
<elt>
  <elt1></elt1>
  <elt3></elt3>
  <elt2></elt2>
</elt>
```

Choix d'éléments

<!ELEMENT elt (elt1 | elt2 | elt3)>

- Sans indication d'occurrence, on ne peut en utiliser qu'un seul.

- Avec indication d'occurrence : <!ELEMENT elt (elt1*|elt2*|elt3*)>

```
<elt>
  <elt2></elt2>
</elt>
```

- Indication d'occurrence globale :<!ELEMENT elt (elt1|elt2|elt3)*>
  - Pas de contrainte sur l'ordre d'apparition des éléments

```
<elt>
  <elt2></elt2>
  <elt2></elt2>
</elt>
```

```
<elt>
  <elt3></elt3>
  <elt2></elt2>
</elt>
```

```
<elt>
  <elt2></elt2>
  <elt3></elt3>
</elt>
```

Quelques remarques

DEF_CONTENU peut être mixte.

- Dans ce cas il est uniquement de la forme (#PCDATA | terme1 | terme2 | terme3 | ...)*
- *terme1, terme2, terme3...* peuvent chacun être à son tour un élément ou un ensemble d'éléments.
- #PCDATA doit toujours être le premier de la liste

- Les mots clés EMPTY et ANY s'emploient sans parenthèses.

Déclaration d'attributs

- Les éléments peuvent contenir des attributs. Ils sont définis dans la DTD comme ceci :

  <!ATTLIST nom_element nom_attribut type obligation valeur_défaut>

- Exemple :

  <!ELEMENT produit (#PCDATA)>

  <!ATTLIST produit prix CDATA #REQUIRED>

- Types possibles : CDATA, ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN, NMTOKENS, énumération
- En général, on ne mentionne pas l'obligation lorsqu'il est fait mention d'une valeur par défaut (excepté pour l'obligation #FIXED). Ainsi :
  - #REQUIRED : attribut obligatoire (on ne mentionne pas de valeur par défaut)
  - #IMPLIED : attribut optionnel (on ne mentionne pas de valeur par défaut)
  - #FIXED : l'attribut est optionnel. Cependant s'il est présent sa valeur est nécessairement celle définie par défaut.

Déclaration d'attributs (suite)

- La valeur par défaut est donc alors présente (1) quand on ne mentionne pas d'obligation ou (2) lorsque l'obligation utilisée est #FIXED.
- Dans le premier cas, cela impliquera que si l'attribut n'est pas renseigné explicitement dans le document xml alors sa valeur sera celle par défaut mentionnée dans le doctype.
- Dans le second cas, l'attribut ne doit pas apparaître obligatoirement mais s'il apparaît alors sa valeur doit obligatoirement être celle qui est définie par défaut après le mot clé #FIXED.

Types d'attributs

- CDATA : n'importe quelle chaîne de caractères possible. C'est le type d'attribut le plus général.
- NMTOKEN : unité lexicale nominale : ne peut contenir que des lettres, des chiffres, un point [.] , un tiret [-], un trait de soulignement [_] et un deux-points [:].

  <!ATTLIST journal annee NMTOKEN #REQUIRED>

  - Autorise "2006", "99", "mars", mais interdit "1990 02"
  - "30-01-2006" autorisé mais pas "30/01/2006"
- NMTOKENS : une ou plusieurs unités lexicales nominales séparées par des espaces blancs. Par espace blanc, on entend un ou plusieurs espaces, retours chariot, sauts de ligne ou tabulations.

Types d'attributs

- *Enumération* : liste de toutes les valeurs possibles séparées par des |
  - <!ATTLIST date mois (Janvier | Février | Mars) #REQUIRED>
- ID : nom XML unique dans le document. Un nom XML est un NMTOKEN avec la restriction suivante : il commence nécessairement par une lettre, ':' ou '_'. Dans l'exemple ci-dessous cela signifie que 2 balises *employe* ne peuvent pas avoir la même valeur pour l'attribut *numero_secu*.
  - <!ATTLIST employe numero_secu ID #REQUIRED>
  - <employe numero_secu="s123-45-6789"/>
- IDREF : attribut faisant référence à un attribut de type ID.
  - <!ATTLIST membre_equipe personne IDREF #REQUIRED>
- IDREFS : liste d'IDREF séparés par des blancs

Entités

- Permettent d'insérer du texte ou d'autres données de manière dynamique dans un document

- Entité interne :

  déclaration <!ENTITY MH "McGraw-Hill">

  utilisation &MH;

- Entité externe :

  déclaration <!ENTITY texte SYSTEM "texte.xml">

  utilisation &texte;

  - Dans ce cas, &texte; sera remplacé par le contenu de texte.xml.

- Le & étant utilisé pour référencer les entités, on utilise & (entité prédéfinie) pour écrire le symbole &.

IGNORE et INCLUDE

- Mots clés pour activer ou désactiver les éléments selon les besoins
- Permet de tester ou de déboguer, utile pour combiner plusieurs DTD en une seule unité.
- Issus du SGML, peu utilisés en XML

```
<![INCLUDE [
<!ELEMENT poeme (strophe+)>
]
]>
```

Entités paramétriques

- Les entités ne s'appliquent pas uniquement au document XML. Elles peuvent également servir à la réalisation de la DTD pour limiter les répétitions de blocs de définition (par exemple, un attribut présent dans plusieurs éléments). Cette forme d'entité est appelée entité paramétrique et doit être déclarée suivant la syntaxe :
  - <!ENTITY % nom "VALEUR">

- L'instruction %nom; sert à utiliser une entité paramétrique dans la DTD.

Entités paramétriques

- Exemple :

```
<!ENTITY % type_defaut "CDATA">
<!ATTLIST chapitre
 titre %type_defaut; #REQUIRED>
```

- Dans cet exemple, nous avons créé une entité paramétrique type_defaut qui est associée à un type (CDATA) pour un attribut. Cette valeur est ensuite employée pour définir le typage de l'attribut titre de l'élément chapitre.

Entités paramétriques

- Grâce aux entités paramétriques, il est également possible d'activer ou de désactiver des blocs de définition. Ces blocs suivent la syntaxe suivante :

```
<![%Valeur;[
 Partie de DTD
]]>
```

- Si Valeur vaut INCLUDE alors la partie de DTD est activée. Si Valeur vaut IGNORE cette partie est ignorée. Dans l'exemple suivant, on définit un attribut langue pour un élément chapitre uniquement si l'entité paramétrique anglais a la valeur INCLUDE.

```
<!ENTITY % anglais 'INCLUDE'>
<![%anglais;[
<!ATTLIST chapitre
 langue (anglais|français) "français">
 ]]>
```

Inconvénients des DTDs

- Manque de précisions dans la descriptions des contenus des éléments : impossible de mettre des contraintes sur les contenus textuels par exemple.
- Peu de flexibilité pour exprimer des cardinalités
- Un fichier DTD n'est pas un fichier XML. Il est donc nécessaire pour les logiciels traitant les docs XML d'avoir des modules supplémentaires pour gérer les DTDs. De plus ces DTDs ne sont pas manipulables par les nombreux outils qui existent autour du XML.
- Pas d'héritage
- Peu de type prédéfinis et pas de possibilité d'en créer
- Délicat de prendre en charge les espaces de noms
- ...

Ces lacunes sont comblées par les schémas XML