



Université Benyoucef Benkhadda - Alger 1
Faculté des Sciences - Département MI

Données Semi Structurées

3.Schema de Xml

DTD – Xml schema (XSD)

Présenté par : Mme. YDROUDJ

a_ydroudj@esi.dz

3ème Année Licence : SI/ISIL

2020-2021

DONNÉES



N

O

N

S

T

C

T

R

E

R

U

U

É

S

Plan du Cours

- ☐ Document valide
- ☐ Pourquoi valider un document Xml
- ☐ DTD : informations générales
- ☐ DTD : Rôle principal
- ☐ Déclaration de la DTD
- ☐ Définition des éléments
- ☐ Définition des attributs
- ☐ Définition des entités
- ☐ Exercices

Document XML valide

- Un document peut être valide par rapport à un schéma qui définit un vocabulaire et une structure.

Qu'est ce qu'un schéma?

Ensemble de contraintes/règles décrivant décrivent la grammaire utilisable dans un document XML

DTD (Document Type Definitions)

XML schema

Validation

Relax NG

Schematron

Valider en comparant chaque document XML à l'aide d'un outil de validation disponible en :

- Mode ligne de commande : xmllint, xmlstarlet, etc.
- En ligne: validator.w3.org
- Intégré par défaut dans des éditeurs xml : xmlSpy

► Pourquoi valider ?

- Uniformité « grammaire pour un groupes de documents »
 - Si plusieurs personnes travaillent sur un même document XML, la DTD garantit que tout le monde respectera la structure définie.
 - Rendre plusieurs documents XML utilisables par un même logiciel de traitement (exemple ERP dans une banque : transactions, marchés, conventions, etc.)
- La validation indique que soit le document est valide, soit il contient des erreurs comme : tel attribut de tel élément contient une valeur interdite par telle contrainte, il manque tel sous-élément dans tel élément, etc.

DTD : informations générales

DTD : description formelle (la grammaire) de la structure du document

Une DTD peut être :

- Locale au document (avec mode standalone=yes),
- Importée dans le document (avec mode standalone=no)
- Mixte (la partie interne est lue en premier, standalone=no)
- La DTD est facultative

Mais :

Un document qui déclare une DTD doit s'y conformer (on parle de la validité). Dans un contexte professionnel, on ne peut pas s'en passer.

Contenu principal

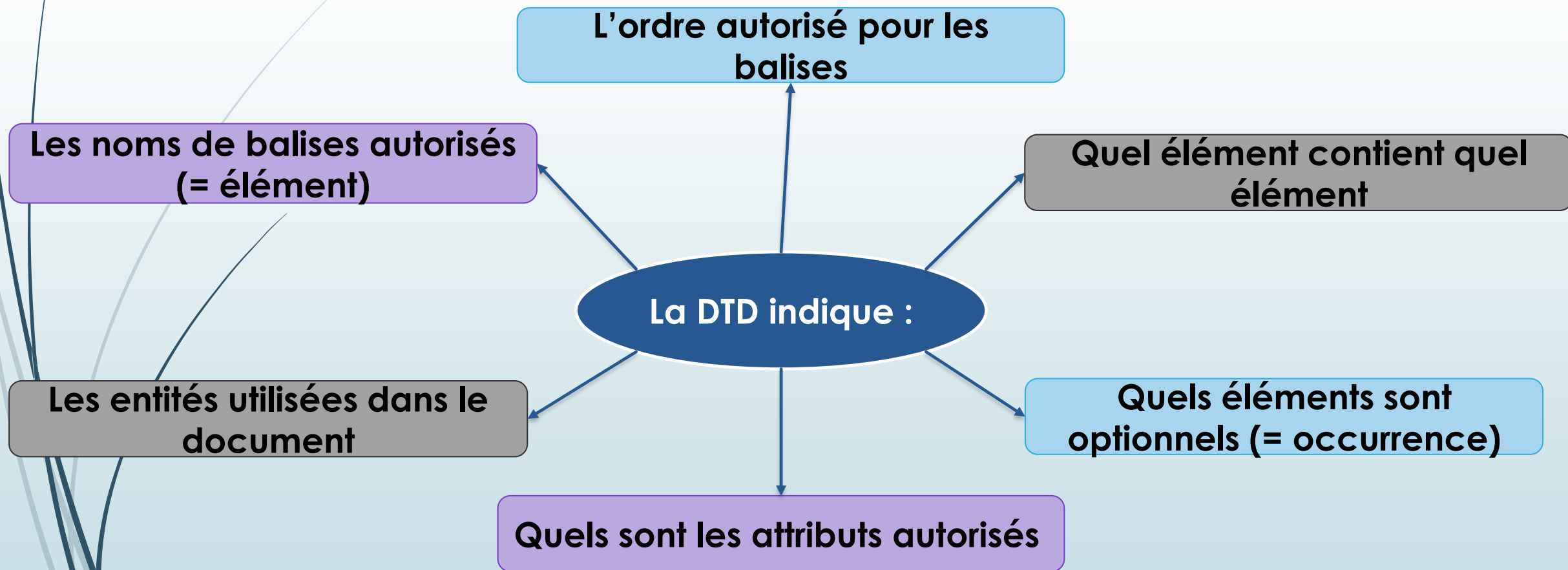
- Déclaration de chaque élément et de son contenu `<!ELEMENT ...>`
- Déclaration des attributs possibles d'un élément `<!ATTLIST ...>`
- Déclaration des entités possibles `<!ENTITY ...>`

Les DTD sont issues de la norme SGML et n'ont pas la syntaxe XML.

Les DTD contiennent des déclarations des:

- Éléments
- Attributs
- Entités

DTD : Rôle principal



DTD : interne, externe et publique, mixte

Déclaration de la DTD dans le document XML

- La référence a la DTD doit être placée au début du fichier après le prologue :

Interne au document XML (1) : `<!DOCTYPE commission [...instructions...]>`

Externe au document, dans un fichier "voisin" (2) :

`<!DOCTYPE commission SYSTEM "(http://x.y/)exemple.dtd">`

Externe au document, publique (sur le web) (3) :

`<!DOCTYPE commission PUBLIC "ident-connu" "URL-sinon">`

- On peut combiner 1+2 ou 1+3 pour avoir une DTD mixte.

Dans ce cas, les déclarations internes sont lues avant la DTD externe pour ne pas avoir de conflits.

DTD externe: permet la réutilisation pour d'autres documents XML

DTD externe publique: définit un standard pour certains documents (CV, facture, publications, ...)

DTD : déclaration des éléments

Syntaxe: `<!ELEMENT le_nom (le_contenu) type_prédéfini>`

- Le contenu contient :
- Soit une séquence de sous éléments :
 - indique quel(s) élément(s) compose(nt) l'élément de la balise
 - liste de sous éléments séparés par une virgule
 - tous les éléments devront apparaître dans l'ordre prévu
- Soit un Choix à prendre parmi une liste de sous éléments :
 - indique quel(s) élément(s) compose(nt) le modèle
 - liste d'éléments séparés par une barre verticale
 - un seul élément de la liste devra apparaître et pas les deux

Un Auteur doit avoir un Nom
puis un Prénom :



`<!ELEMENT Auteur (Nom, Prénom)>`

Une coordonnée peut être un
numéro de tel ou une adresse
mail :



`<!ELEMENT coordonnée (numTel | email)>`

DTD : déclaration des éléments

Syntaxe: `<!ELEMENT le_nom (le_contenu) type_prédéfini>`

■ Type prédéfini contient :

- **ANY** : signifie que l'élément peut contenir n'importe quels éléments (définis dans la DTD) et textes (leur ordre d'apparition et leur nombre ne seront pas testés),

- **EMPTY** : signifie que l'élément doit être vide,

- **#PCDATA** : signifie que l'élément ne contient que des textes
son utilisation :

- **ANY** et **EMPTY** : sans parenthèses

- **(#PCDATA)** : avec parenthèses

`<!ELEMENT Nom (#PCDATA)>`

`<!ELEMENT img EMPTY>`

`<!ELEMENT adresse ANY>`

DTD : déclaration des sous éléments

Quelques détails sur les sous éléments

- Une liste ordonnée déclarée dans l'élément père, dans laquelle chaque sous-élément peut être suivi d'un suffixe qui indique la fréquence de son apparition :

Opérateur	Cardinalités	Signification	Exemple
	(1,1)	Obligatoire (par défaut) : doit apparaître une et une seule fois	A
?	(0,1)	Optionnel : doit apparaître zéro ou une seule fois	A?
*	(0,n)	Multiple : peut apparaître plusieurs fois ou ne pas apparaître	A*
+	(1,n)	Multiple obligatoire : doit apparaître au moins une fois	A+
		L'élément A ou B peuvent être présents mais pas les deux, ordre n'est pas important	A B
,		L'élément A doit être présent suivi de l'élément B	A,B
()		Permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

DTD : déclaration des sous éléments

Exemple1 : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [
<!ELEMENT INVENTAIRE (TITRE)>
<!ELEMENT TITRE (#PCDATA | SOUSTITRE)*>
<!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci zéro, une ou plusieurs fois mais notre document XML ne peut contenir qu'un seul « TITRE ».

```
<INVENTAIRE>
<TITRE>
Thorgal
<SOUSTITRE>Et la chambre des secrets</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```



```
<INVENTAIRE>
<TITRE>
<SOUSTITRE>Et la chambre des secrets</SOUSTITRE>
Thorgal
</TITRE>
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemples : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [  
<!ELEMENT INVENTAIRE (TITRE)>  
<!ELEMENT TITRE (#PCDATA | SOUSTITRE)*>  
<!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci zéro, une ou plusieurs fois mais notre document XML ne peut contenir qu'un seul « TITRE ».

```
<INVENTAIRE>  
<TITRE>  
    Thorgal  
</TITRE>  
</INVENTAIRE>
```



```
<INVENTAIRE>  
<TITRE>  
<SOUSTITRE>Et la chambre des secrets</SOUSTITRE>  
<SOUSTITRE>la grande surprise</SOUSTITRE>  
</TITRE>  
</INVENTAIRE>
```




DTD : déclaration des sous éléments

Exemples : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [  
<!ELEMENT INVENTAIRE (TITRE)>  
<!ELEMENT TITRE (#PCDATA | SOUSTITRE)*>  
<!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

```
<INVENTAIRE>  
<TITRE>  
</TITRE>  
</INVENTAIRE>
```




Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci zéro, une ou plusieurs fois mais notre document XML ne peut contenir qu'un seul « TITRE ».

```
<INVENTAIRE>  
<TITRE>  
Thorgal  
<SOUSTITRE>Et la chambre des secrets</SOUSTITRE>  
thorgal  
<SOUSTITRE>la grande surprise</SOUSTITRE>  
</TITRE>  
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemple 2 : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [
  <!ELEMENT INVENTAIRE (TITRE)*>
  <!ELEMENT TITRE (#PCDATA | SOUSTITRE)>
  <!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci une et une seule fois (il s'agit donc d'un OU exclusif).

Mais le documents XML peut contenir zéro, un ou plusieurs « TITRE ».

```
<INVENTAIRE>
<TITRE>
Thorgal
<SOUSTITRE>Et lla chambre des secrets</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```



```
<INVENTAIRE>
<TITRE>
<SOUSTITRE>La grande surprise</SOUSTITRE>
Thorgal
</TITRE>
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemple 2 : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [  
<!ELEMENT INVENTAIRE (TITRE)*>  
<!ELEMENT TITRE (#PCDATA | SOUSTITRE)>  
<!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci une et une seule fois (il s'agit donc d'un OU exclusif).

Mais le documents XML peut contenir zéro, un ou plusieurs « TITRE ».

```
<INVENTAIRE>  
<TITRE>  
Thorgal  
</TITRE>  
</INVENTAIRE>
```



```
<INVENTAIRE>  
<TITRE>  
<SOUSTITRE>la grande surprise</SOUSTITRE>  
<SOUSTITRE>Les loups de la jungle</SOUSTITRE>  
</TITRE>  
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemple 2 : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [
<!ELEMENT INVENTAIRE (TITRE)*>
<!ELEMENT TITRE (#PCDATA | SOUSTITRE)>
<!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci une et une seule fois (il s'agit donc d'un OU exclusif).

Mais le documents XML peut contenir zéro, un ou plusieurs « TITRE ».

```
<INVENTAIRE>
<TITRE>
</TITRE>
</INVENTAIRE>
```



```
<INVENTAIRE>
<TITRE>
Thorgal
<SOUSTITRE>Et la chambre des secrets</SOUSTITRE>
Thorgal
<SOUSTITRE> Les loups de la jungle</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemple 3 : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [  
<!ELEMENT INVENTAIRE (TITRE)>  
<!ELEMENT TITRE (#PCDATA, SOUSTITRE)*>  
<!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

Elle signifie que pour un élément TITRE, on doit avoir :

- des données caractères

ET

- un élément SOUSTITRE

Et ceci zéro, une ou plusieurs fois

Mais notre document XML ne peut contenir qu'un seul TITRE.

```
<INVENTAIRE>  
<TITRE>  
Thorgal  
<SOUSTITRE>Et la chambre des secrets</SOUSTITRE>  
</TITRE>  
</INVENTAIRE>
```



```
<INVENTAIRE>  
<TITRE>  
<SOUSTITRE>La grande surprise</SOUSTITRE>  
Thorgal  
</TITRE>  
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemple 3 :


```
<INVENTAIRE>
<TITRE>
<SOUSTITRE>La grande surprise</SOUSTITRE>
<SOUSTITRE>Les loups de la jungle</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```




```
<INVENTAIRE>
<TITRE>
Thorgal
</TITRE>
</INVENTAIRE>
```



```
<INVENTAIRE>
<TITRE>
</TITRE>
</INVENTAIRE>
```



```
<INVENTAIRE>
<TITRE>
Thorgal
<SOUSTITRE>la grande surprise</SOUSTITRE>
Thorgal
<SOUSTITRE> Les loups de la jungle</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemple 4 : Soit la DTD suivante

```
<!DOCTYPE INVENTAIRE [  
<!ELEMENT INVENTAIRE (TITRE)*>  
<!ELEMENT TITRE (#PCDATA, SOUSTITRE)>  
<!ELEMENT SOUSTITRE (#PCDATA)> ]>
```

Elle signifie que pour un élément TITRE, on doit avoir :

- des données caractères

ET

- un élément SOUSTITRE

Et ceci une et une seule fois (il s'agit donc d'un ET exclusif)

Mais le document XML peut contenir zéro, un ou plusieurs « TITRE ».

```
<INVENTAIRE>  
<TITRE>  
Thorgal  
<SOUSTITRE>Et la grande surprise</SOUSTITRE>  
</TITRE>  
</INVENTAIRE>
```



```
<INVENTAIRE>  
<TITRE>  
<SOUSTITRE>La grande surprise</SOUSTITRE>  
Thorgal  
</TITRE>  
</INVENTAIRE>
```



DTD : déclaration des sous éléments

Exemple 4 :


```
<INVENTAIRE>  
<TITRE>  
<SOUSTITRE>La grande surprise</SOUSTITRE>  
<SOUSTITRE>Les loups de la jungle</SOUSTITRE>  
</TITRE>  
</INVENTAIRE>
```




```
<INVENTAIRE>  
<TITRE>  
Thorgal  
</TITRE>  
</INVENTAIRE>
```



```
<INVENTAIRE>  
<TITRE>  
</TITRE>  
</INVENTAIRE>
```



```
<INVENTAIRE>  
<TITRE>  
Thorgal  
<SOUSTITRE>la grande surprise</SOUSTITRE>  
Thorgal  
<SOUSTITRE> Les loups de la jungle</SOUSTITRE>  
</TITRE>  
</INVENTAIRE>
```



DTD : déclaration des attributs

Syntaxe: <!ATTLIST nom-element

nom-attribut1 type-att declaration-default

nom-attribut2 type-att declaration-default

nom-attribut3 type-att declaration-default

.... >

- Chaque attribut défini dans la liste possède un nom et un type.
- Déclaration par défaut d'un attribut peut être :
 - **#REQUIRED** : Une valeur doit être affectée à l'attribut (obligatoire).
 - **#IMPLIED** : l'attribut est facultatif (on peut inclure ou non l'attribut dans un élément de type associé)
 - **#FIXED** : valeur par défaut (et uniquement celle-là)
 - **'Valeur'** : Ou encore directement une valeur par défaut entre guillemets

DTD : type des attributs possibles

Type	Description
CDATA	Chaînes de caractères non analysées par le parseur.
Type énuméré (Valeur ... ValeurN)	Déclare une liste de valeurs à utiliser, une seule est choisie sur la liste.
ENTITY	Correspond à une entité déclarée dans la DTD.
ENTITIES	Correspond à la déclaration de plusieurs entités séparées par des espaces blancs dans la DTD.
ID	L'attribut en question est un <i>identifiant</i> dans le fichier XML. La valeur d'un attribut de type ID ne peut pas être utilisée deux fois dans le même document(Principe de la clé primaire). Il y a au plus un attribut de type ID par élément
IDREF	L'attribut en question correspond à la valeur d'un attribut ID déclaré précédemment. (principe de la clé étrangère)
IDREFS	Correspond aux valeurs de plusieurs attributs ID séparés par un espace blanc.
NMTOKEN	L'attribut prend un nom XML, sa valeur doit contenir des caractères d'unité lexicale : Caractères alphabétiques accentués , Chiffres , Caractères _ - . : et Caractères des autres langues non romanes
NMTOKENS	invoque plusieurs tokens XML séparés par des espaces blancs.

DTD : type des attributs possibles

Exemples :

■ CDATA :

attribut

Type chaîne
de caractères

Déclaration-défaut
(présence obligatoire)

élément

```
<!ATTLIST image source CDATA #REQUIRED  
hauteur CDATA #REQUIRED  
largeur CDATA #REQUIRED  
texte CDATA #IMPLIED  
>
```

```
<image source="bubulle.jpg"  
largeur="10cm"  
hauteur="5cm"  
texte="mon livre vert"/>
```

■ Énumération:

```
<!ATTLIST date mois ( janvier | février | mars |  
avril | mai | juin | juillet | août | septembre  
| octobre | novembre | decembre ) #REQUIRED  
>
```

```
<date mois="juin"/>
```

Chaque valeur de la liste doit respecter la syntaxe

DTD : type des attributs possibles

Exemples :

■ ID, IDREF, IDREFS :

```
<!ELEMENT user (#PCDATA)>
<!ATTLIST user id ID #REQUIRED>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
<!ATTLIST manager matri ID #REQUIRED>
<!ATTLIST project users IDREFS #REQUIRED>
```

↓

```
<user id="u1" />XX</user>
<user id="u2" />YY</user>
<user id="u3"/>ZZ</user>
<manager matri="mat1"/>ZZ</manager>
<project users="mat1 u1 u2">Foo</project>
<project users="u2 u3">Bar</project>
<project users="u1 u2 u3">Any</project>
```

```
<!ATTLIST a id ID #IMPLIED
           name ID #IMPLIED >
```



```
<!ATTLIST a id ID #IMPLIED
           name NMTOKEN #IMPLIED >
```



```
<a name="fichier1.pdf" id="link">fichier 1</a>
<a name="fichier2.pdf" id="link">fichier 2</a>
```



```
<a name="fichier1.htm" id="link1">fichier 1</a>
<a name="fichier2.htm" id="link2">fichier 2</a>
```



DTD : type des attributs possibles

Exemples :

- **NMTOKEN, NMTOKENS:**

```
<!ATTLIST user date_connex NMTOKEN #REQUIRED>
```

```
<user date_connex="02-09-2020"/>
```

```
<!ATTLIST user date_connex NMTOKENS #REQUIRED>
```

```
<user date_connex="02-09-2020 13:23:45s"/>
```

```
<!ELEMENT exo (#PCDATA)>  
<!ATTLIST exo niveau NMTOKENS #IMPLIED>
```

```
<exo niveau="facile"> série des exercices <XML>  
</exo>  
<exo niveau="difficile facultatif"> exercice/1  
</exo>
```

DTD : type des attributs possibles

Exemples :

- **Type Notation :**

- Une notation décrit le format d'une donnée qui ne peut pas être analysée (contenu non xml tel que graphique, multimédia, films FLASH, etc.), ou bien identifie le programme utilisé pour traiter un format particulier.
- `<image source="toile.gif">` : utilisant une NOTATION, permettra de faire savoir à l'analyseur syntaxique XML que le contenu d'attribut `source` ne doit pas seulement être interprété comme une simple chaîne de caractères mais comme référence d'un fichier externe
- La valeur d'un attribut de type NOTATION est le nom d'une des notations déclarées dans la DTD et énumérées pour l'attribut.

Syntaxe : `<!NOTATION nom SYSTEM "url_emplacement_notation">`
`<!NOTATION nom PUBLIC "url_associe">`

DTD : type des attributs possibles

Exemples :

- **Type Notation :**

```
<!NOTATION jpeg      SYSTEM "image/jpeg">
<!NOTATION gif       SYSTEM "image/gif">
<!NOTATION png       SYSTEM "image/png">
<!-- ATTLIST image
      source      CDATA #REQUIRED
      hauteur     CDATA #REQUIRED
      largeur     CDATA #REQUIRED
      texte       CDATA #IMPLIED
      type        NOTATION (gif | jpeg | png) #REQUIRED
-->
```



```
<image      source="photo.jpg"
           largeur="10cm"
           hauteur="5cm"
           texte="mon poisson rouge"
           type="jpeg"/>
```

DTD : type des attributs possibles

Exemples :

- **Entity, Entities :**

- plusieurs classifications : interne ou externe, analysable ou non analysable, générale ou paramètre.

1- Interne ou externe : suivant sa localisation par rapport au document XML (déjà vue dans le chapitre 2)

2- Analysable ou non analysable : suivant le contenu de l'entité

- **entité analysable (parsed entity ou contenu xml) :** elles ne peuvent contenir que des caractères (texte). Le contenu est appelé texte de remplacement.
- **entité non analysable (unparsed entity ou contenu non xml) :** sont des ressources qui peuvent contenir tout type de données (fichier, images).

3- générale ou paramètre : suivant son usage comme étant un contenu dans le document xml ou comme une partie à remplacer dans la DTD.

- Nous verrons ici uniquement les entités externes non parsées et les entités paramètres (les autres voir chapitre2)

DTD : type des attributs possibles

- **Entité externe non analysable par xml :** Comme déclaration des images
- La valeur d'un attribut de type ENTITY comme l'attribut source de type ENTITY, est le nom d'une entité non parsée qui doit être déclarée dans la DTD

```
<!NOTATION jpeg      SYSTEM "image/jpeg">
<!ENTITY   photo     SYSTEM "photo.jpg" NDATA jpeg>
<!ATTLIST  image      source      ENTITY #REQUIRED
                        hauteur     CDATA  #REQUIRED
                        largeur     CDATA  #REQUIRED
                        texte       CDATA  #IMPLIED
>
```



```
<image      source="photo"
            largeur="10cm"
            hauteur="5cm"
            texte="mon poisson rouge"/>
```

photo est remplacée
par "**photo.jpg**"
dont le type est **jpeg**
défini dans la notation

DTD : type des attributs possibles

- **Entité paramètre (à remplacer dans les DTD):**

syntaxe: `<!ENTITY %exempl "cours et TP">`

Appel : `%exempl;`

- ➡ **Différence avec les entités générales :**

- Appel avec le signe % au lieu de &
- Appel uniquement dans la DTD et jamais dans le contenu principal du document
- Permet de factoriser les définitions d'élément
- Une entité paramètre doit toujours être déclarée avant d'être appelée

Rôle principal :

- Limiter les répétitions de blocs de définition

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

DTD : type des attributs possibles

3-Entité paramètre (dans les DTD) : Exemple

```
<!ELEMENT dauphin (nom, taille, poids, date-naissance, commentaire?)>
<!ATTLIST dauphin id ID #REQUIRED
                espèce CDATA #IMPLIED
                nom-savant CDATA #IMPLIED
                photo ENTITY #IMPLIED
>
<!ELEMENT baleine (nom, taille, poids, date-naissance, commentaire?)>
<!ATTLIST baleine id ID #REQUIRED
                  espèce CDATA #IMPLIED
                  nom-savant CDATA #IMPLIED
                  photo ENTITY #IMPLIED
>
```

```
<!ENTITY % animal "nom, taille, poids, date-naissance, commentaire?">
<!ENTITY % animal-attr "id ID #REQUIRED
                        espèce CDATA #IMPLIED
                        nom-savant CDATA #IMPLIED
                        photo ENTITY #IMPLIED
">
<!ELEMENT dauphin (%animal;)>
<!ATTLIST dauphin %animal-attr;>
<!ELEMENT baleine (%animal;)>
<!ATTLIST baleine %animal-attr;>
```


DTD : type des attributs possibles

Entity, Entities :

```
<!DOCTYPE Université [
  <!ENTITY UA1 "Université Alger1">
  <!ENTITY UA1Desc SYSTEM "http://www.univ-
    alger1.dz/description.xml">
  <!ENTITY prof "Pr.X">
  <!ENTITY %spec "(mathématiques | Informatique)">
  <!ENTITY explication SYSTEM "annexe1.xml">
  <!ELEMENT dept %spec; #REQUIRED>
]>
```

```
<Université>
  <nom>&UA1;</nom>
  <directeur>&prof;</directeur>
  <dept> Informatique</dept>
  <desc>&explication;</desc>
</Université>
```

Entité
interne
générale

Entité
externe

Entité
paramètre

```
<université>
  <nom> Université Alger 1</nom>
  <directeur> Pr.X</directeur>
  <dept> Informatique </dept>
  <desc> Tout le contenu du fichier
    annexe1.xml ...</desc>
</université>
```

- DTD permet d'exprimer des contraintes assez basiques
 - Liste des éléments et de leurs attributs
 - Règles de structuration des éléments

Mais :

- Impossible de typer réellement les attributs
- Pas de contrainte de longueur de champs
- Il ne peut y avoir deux éléments de même nom dans deux contextes différents
- Pas d'héritage possible entre éléments : notation lourde
- Pas de prise en compte des espaces de noms



Autres langages de schéma plus complets, par ex. **XML Schema**

DTD Vs XML Schema

DTD	XML schema
N'est pas une syntaxe XML	Basé sur la syntaxe XML
Difficile à étendre	Facilement extensible
Données textuelles non typées	Supporte les types de données
Ne permet pas de spécifier exactement le nombre d'occurrences d'un élément	Permet de spécifier exactement le nombre d'occurrences d'un élément
Ne supporte pas les espaces de noms	Supporte les espaces de noms

Malheureusement, par contrainte du Temps, le XML schema ne va pas être traité pour cette année.

