



République Tunisienne
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique



Technologies XML

Chap2 : DTD (Document Type Definition)

Njeh Maissa



Audiences: D-LSI-ADBD

Année universitaire : 2023/2024

Plan



1 Qu'est-ce c'est DTD ?

- DTD .Document Type Définition ou Document Type Déclaration
- DTD est l'ensemble **des règles et des propriétés** que doit suivre le document XML.
- DTD est une **grammaire** dont le rôle est de **définir précisément d'un document.**
- C'est **l'ensemble de contraintes** que doit respecter un document pour être valide:.....

Un document est dit valide si :

- ✓ Il est bien formé : lorsqu'un document XML répond aux règles de base du XML et ne comporte pas de DTD
- ✓ Il est conforme a la DTD ou au schéma qui lui associé

- Un document est valide par rapport a une DTD si ce doument XML est bien formé et conforme a cette DTD==

- DTD permet de spécifier **une grammaire pour un langage** et de **tester** automatiquement son respect par un document donnée
- DTD définit la structure d'un document :
 - ***Les éléments et les attributs*** qui y sont autorisés
 - ***Les types*** de contenus qui y sont permis
- Elle permet de faire la différence entre un document bien formé et un document valide

L'avantage est de :

- Faciliter l'échange et la mise en commun de document produits par différents personnes
- Aider les développeurs qui convient des outils automatiques pour traiter les documents respectant la même DTD

- La déclaration de la DTD doit être placée **dans le prologue** du fichier XML
- La DTD peut être :

INTERNE : directement incluse dans le documents

EXTERNE : une référence vers un autre document contenant la DTD

MIXTE : constituée d'une partie interne et d'une partie externe

- La déclaration d'une DTD est comme suit :

<!DOCTYPE root-element....>

Mot clé de
déclaration de DTD

Nom de l'élément
racine du document

La déclaration est de la forme suivante:

- Les déclarations constituent la définition DTD

```
<!DOCTYPE root-element [ declarations]>
```

Exemple:

```
<!DOCTYPE simple [< !ELEMENT simple (#PCDATA)>  
]>
```

- Le nom de la balise racine **est simple**
- **Ne peut contenir que de texte (PCDATA: Parsed Characters DATA) et pas d'autre élément**

```
<simple> contient seulement du texte</simple>
```



- Les déclarations d'une DTD interne sont écrites juste après le prologue du document XML.
- Les déclarations doivent être faite dans l'ordre:
 - Mot-clé DOCTYPE
 - Élément racine du document ;
 - Contenu de la DTD elle-même, entre crochets

Remarque :

*Si la DTD **interne**, il faut déclarer dans le prologue d'un fichier XML **standalone="yes"***

*Lorsqu'il **externe** l'attribut **standalone="no"***



```
<?xml version="1.0" encoding="iso-8859-1"
standalone="yes"?>
```

```
<!DOCTYPE parent
[
  <!ELEMENT parent (garcon,fille)>
  <!ELEMENT garcon (#PCDATA)>
  <!ELEMENT fille (#PCDATA)>
]>
```

```
<parent>
```

```
  <garcon>mahdi</garcon>
```

```
  <fille>chayma</fille>
```

```
</ parent >
```




```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE bibliotheque[
  <!ELEMENT bibliotheque (livre)*>
  <!ELEMENT livre (titre, auteur)>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT auteur (#PCDATA)>
]>
<bibliotheque>
  <livre>
    <titre>XML : Cours et exercices</titre>
    <auteur>Alexandre Brillant</auteur>
  </livre>
</bibliotheque>
```

Plusieurs

Type chaîne de caractère

Activer Windows

Accédez aux paramètres de

- Une DTD est un document texte avec l'extension **.dtd** , ce n'est pas un document XML
- Ne commence pas par une déclaration XML.
- Les déclarations d'une **DTD externe** sont écrites juste **après** le prologue du document XML.

Syntaxe de déclaration:

```
<!DOCTYPE nom-racine SYSTEM URI_dtd>
```

URI dtd: chaîne de caractères qui peut représenter :

- L'identification absolue du fichier contenant la DTD, par exemple:
 - `<!DOCTYPE html SYSTEM "http://www.w3.org/TR/DTD/xhtml1.dtd" >`
- Une adresse relative si document et DTD sont sur le même site:
 - `<!DOCTYPE guide SYSTEM " ../../DTD/rando/rando.dtd" >`
- Un simple nom de fichier si DTD et document sont dans le même répertoire:
 - `<!DOCTYPE personne SYSTEM " personne.dtd" >`

Les DTD externes peuvent être:

DTD privées

DTD publiques

DTD privées

- Accessibles uniquement en local, sur la machine de développement;
- De type SYSTEM

```
<!DOCTYPE element racine SYSTEM " URI_dtd " >
```

- Exemple:

```
<!DOCTYPE bibliotheque SYSTEM " bibliotheque.dtd " >
```

DTD publiques

- Disponibles pour tout le monde, sur un serveur distant
- On y accède grâce à une URI (Uniform Resource Identifier);
- De type PUBLIC

```
<!DOCTYPE element racine PUBLIC " URI_dtd " >
```

11 Déclaration de la DTD externe

- Le nom de la DTD appelée **URN** (Universal Resource Name) doit avoir la forme:

-//W3C//DTD catalogue //FR

Le nom du propriétaire suivie du type de document,
suivi de la langue _____

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "  
http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

parent.xml

```
<?xml version="1.0" encoding="iso-8859-1"  
standalone="no"?>
```

```
<!DOCTYPE parent SYSTEM " parent.dtd " >
```

```
<parent>
```

```
    <garcon>mahdi</garcon>
```

```
    <fille>chayma</fille>
```

```
</ parent >
```

parent.dtd

```
<!ELEMENT parent (garcon,fille)>
```

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

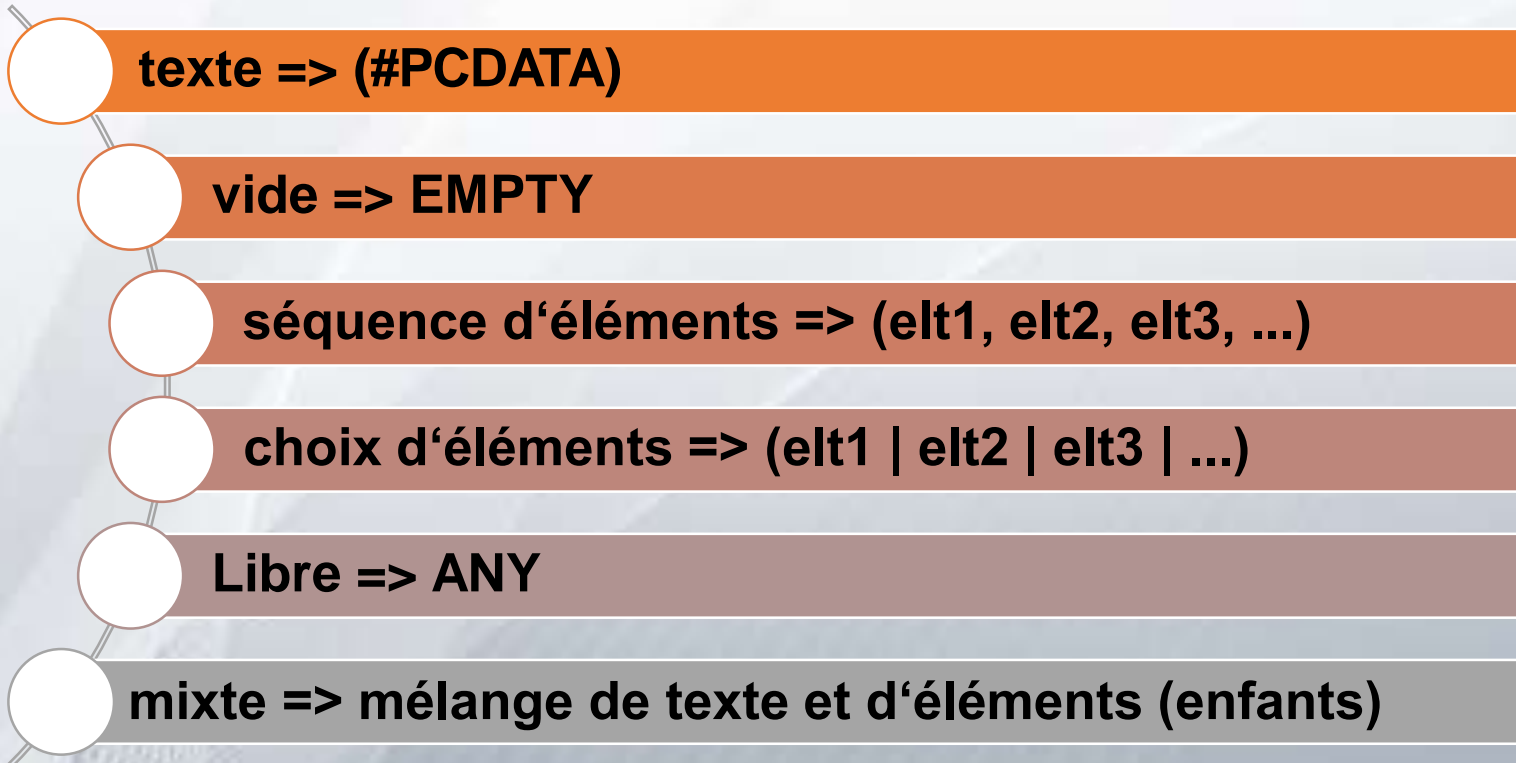
```
<!ELEMENT fille (#PCDATA)>
```

13 Déclaration les éléments

Syntaxe de déclaration d'un élément dans une DTD :

<!ELEMENT nom-element contenu-element>

- **nom_element** => nom d'une balise dans le fichier XML associe ;
- **contenu_element** : type auquel il est associé.
- Les valeurs possibles du type sont :



14 Déclaration les éléments

Syntaxe de déclaration d'un élément texte dans une DTD

```
<!ELEMENT nom-element(# PCDATA)>
```

Remarque:

Pour éviter les éventuelles erreurs du parseur, mieux vaut mettre le mot clé **#PCDATA** entre parenthèses.

Exemple d'utilisation :

.....

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

Se traduira par exemple dans le document XML :

.....

```
<titre> titre de livre</titre>
```


15 Déclaration les éléments

Syntaxe de déclaration d'un élément vide dans une DTD

```
<!ELEMENT nom-element EMPTY>
```

- L'élément vide n'a aucun contenu : pas de texte, ni même d'autres éléments.
- C'est une balise auto-fermante.

Exemple d'utilisation :

```
<!ELEMENT nom EMPTY>
```

....

Se traduira par exemple dans le document XML :

```
</nom>
```

Remarque:

Un élément vide peut tout a fait posséder des attributs.

Par exemple :

```

```

Élément a contenu mixte

- L'élément a contenu mixte est une liste de choix, avec des indicateurs d'occurrence bien sélectionnées.
 - Il peut contenir aussi bien du texte que des éléments enfants.

Syntaxe de déclaration d'un élément à contenu mixte dans une DTD :

..... `<!ELEMENT identite (#PCDATA | nom)>`

- **Exemple d'utilisation de cette déclaration :**

`<identite>`

M. `<nom>Dupond</nom>`

`<identite>7KI`

- Pour les séquences, les choix d'éléments et les éléments à contenu mixte, la notion d'indicateurs d'occurrence est mentionnée.
- Ces indicateurs permettent de définir les éléments XML qu'un élément peut ou doit contenir
- Ils définissent des règles d'utilisation, grâce à une syntaxe spécifique
 - **? : un seul ou rien**
 - **+ : un ou plusieurs**
 - *** : rien, un ou plusieurs (définition d'une option)**
 - **| : l'un ou l'autre mais pas les deux**
 - **, : obligatoires (dans l'ordre)**
 - **() : regroupement => les parenthèses permettent de regrouper les éléments pour leur appliquer les autres opérateurs**

18	Indicateur d'occurrence	
<!ELEMENT elem (elem1,elem2,elem3)>	elem doit contenir un élément elem1, un élément elem2 puis un élément elem3 dans cet ordre.	
<!ELEMENT elem (elem1 elem2 elem3)>	elem doit contenir un seul des éléments elem1,elem2 <u>ou</u> elem3	
<!ELEMENT elem (elem1,elem2?,elem3)>	elem doit contenir en ordre un élément elem1, <u>un ou zéro</u> élément elem2 puis un élément elem3	
<!ELEMENT elem (elem1,elem2*,elem3)>	elem doit contenir en ordre un élément elem1, <u>zéro a plusieurs</u> élément elem2 puis un élément elem3	
<!ELEMENT elem (elem1,(elem2 elem4),elem3)>	elem doit contenir en ordre un élément elem1, <u>un elem2 ou elem4</u> puis un élément elem3	
<!ELEMENT elem (elem1,elem2,elem3)*>	elem doit contenir une suite d'éléments elem1,elem2,elem3 <u>répéter 0 a plusieurs fois.</u>	
<!ELEMENT elem (elem1 elem2 elem3)*>	elem doit contenir une suite quelconque d'éléments elem1,elem2,elem3 <u>répéter 0 a plusieurs fois.</u>	
<!ELEMENT elem(elem1 elem2 elem3)+>	elem doit contenir <u>une suite non vide</u> d'éléments elem1,elem2,elem3	

Syntaxe de déclaration d'un attribut dans une DTD :

```
<!ATTLIST nom-element nom-attribut type-attribut mode >
```

.....

- **Nom_element** : nom de l'élément auquel cet attribut appartient ;
- **Nom_attribut** : nom de l'attribut en cours de définition ;
- **type attribut** : type de donnée de l'attribut :
 - **CDATA** => on affecte une chaîne de caractères à l'attribut
 - **énumération**
 - on définit une liste de valeurs possibles pour l'attribut (permet de limiter le choix de l'utilisateur) (Choix1| Choix2 | ...)
 - pour définir une valeur par défaut il faut faire suivre l'énumération par la valeur désirée entre guillemets => (Choix1 |Choix2 | ...) " valeur par défaut"

- **NMTOKEN** => Les attributs de type NMTOKEN ne peuvent contenir que des lettres, des chiffres, un point [.], un tiret [-], un trait de soulignement [_] et un deux-points [:]. Exp: bbb="a1:12"
- **NMTOKENS** => suite de mots NMTOKEN, séparés par des espaces. Exp: ccc=" 3.4 div -4"
- **ENTITY** => Les entités sont utilisées pour remplacer une chaîne de caractères par un symbole, puis utiliser ce symbole à la place de cette chaîne.
- **ENTITIES** => suite de noms d'entités, séparés par des espaces (suite de ENTITY, séparés par des espaces)

Remarque:

Par espace blanc, on entend un ou plusieurs espaces, retours chariot, sauts de ligne ou tabulations.

- **ID** => on définit un identifiant unique pour chaque élément
- **IDREF(S)** => renvoi vers un (des) ID utilisé(s) ailleurs dans le doc

Remarques

- Un **ID** ne peut être que

#REQUIRED ou #IMPLIED

- Un élément ne peut disposer au maximum que d'une seule attribut ID

- **mode** : précisions sur le type d'attribut :

- **#IMPLIED** => ...

attribut facultatif

- **#REQUIRED** => ..

attribut obligatoire

- **#FIXED** valeur => ..

...

attribut a valeur fixe ; la valeur est déjà fixé
dans la DTD

- **#DEFAULT** valeur => ..

valeur par défaut XXX

Exemple de fraction d'une DTD

- L'élément date est du texte.

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

....

- Cet élément dispose d'un attribut format obligatoire, ne pouvant prendre que la valeur EN ou FR.

```
<!ATTLIST date format (FR | EN) #REQUIRED >
```

....



Exemple de déclaration correcte dans le document XML:

```
<date format="FR">8 octobre 2010</date>
```

Exemple de fraction d'une DTD

```
<!ELEMENT auteur (#PCDATA )>  
<!ELEMENT livre (#PCDATA )>  
<!ATTLIST auteur numero ID #REQUIRED >  
<!ATTLIST livre reference IDREF #REQUIRED >
```

...

Exemple de déclaration correcte dans le document XML:

```
<auteur numero="a1">Thierry Boulanger</auteur>  
<auteur numero="a2">Alexandre Brillant</auteur>  
<livre reference="a1">XML par la pratique – Bases  
indispensables, concepts et cas pratiques</livre>  
<livre reference="a2">XML : Cours et exercices</livre>
```

Exemple de fraction d'une DTD utilisant les NMTOKEN/NMTOKENS :

```
<!ELEMENT attributes (#PCDATA)>
<!ATTLIST attributes aaa NMTOKEN #REQUIRED
                        bbb NMTOKEN #REQUIRED
                        ccc NMTOKENS #IMPLIED >
```



Exemple de déclaration correcte dans le document XML:


```
<attributes aaa="#d1" bbb="a1:12" ccc=" 3.4 div      -4"/>
```

Les attributs de type NMTOKEN ne peuvent contenir que des lettres, des chiffres, un point [.], un tiret [-], un trait de soulignement [_] et un deux-points [:].
Exp: bbb="a1:12"

suite de mots NMTOKEN,
séparés par des espaces. Exp:
ccc=" 3.4 div -4"

Exemple de fraction d'une DTD utilisant les NMTOKENS:

```
<!ELEMENT secureDocument EMPTY>  
<!ATTLIST secureDocument authorizedUsers NMTOKENS #IMPLIED >
```



.....

Exemple de déclaration correcte dans le document XML:

```
<secureDocument authorizedUsers="James.Bond M Miss.MoneyPenny"/>
```

Remplacement d'une chaîne de caractère par un symbole, puis son utilisation à la place de cette chaîne.

- Il existe deux types des entités:

- Les entités **générales**
- Les entités paramètre

- La différence entre entités **générales** et entités **paramètres** réside dans le contexte d'utilisation :

Les entités **générales** sont définies

- Dans la DTD et utilisées dans les documents XML correspondants

Les entités **paramètres** sont définies

- Dans la DTD et utilisées dans la DTD elle même

Syntaxe de déclaration d'une entité générale:

<!ENTITY nom-entité 'texte de remplacement' >

- Une entité générale est toujours invoquée sous la forme

&SYMBOLE

Au sein d'un document la ou devrait apparaître le texte de remplacement associé

Entités prédéfinies

- Cinq entités prédéfinies pour les caractères spéciaux qui sont interdits dans les nom XML

Entité	<	>	'	"	&
Caractère	<	>	'	"	&

Entités nom prédéfinies

- Permet la définition d'un texte sous un nom
- Trois formes de déclarations possibles :

```
<!ENTITY name "fragment">  
<!ENTITY name SYSTEM "url">  
<!ENTITY name public " fpi " "url">
```

} Entité interne

} Entité externe

- Invoquée sous la forme **&name** ; au sein d'un document la ou devrait apparaitre le texte de remplacement associé.
- Ordre de leur définition dans une DTD est indifférent

<!ENTITY non-entité 'texte de remplacement'>

<?xml version="1.0"?>

<!DOCTYPE test [

<!ENTITY lab "&abv; &long;" >

<!ENTITY long "tres tres longue">

<!ENTITY abv " ceci est une phrase ">

<!ENTITY cp " © edition la lune blanche ">]>

<test><p> &lab; & un peu obsoure </p>

< p> &cp; </p><test>

<test>

<p> ceci est une phrase tres tres longue & un peu
obscure </p>

< p> edition la lune blanche</p><test>

Désigne un fragment de document contenu dans un autre fichier

```
<?xml version="1.0"?>
<!DOCTYPE livre [
  <!ELEMENT livre (titre,chapitre *)>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT chapitre (titre,section+)>
  <!ATTLIST chapitre numero CDATA #REQUIRED >
  <!ELEMENT section (#PCDATA)>
    <!ENTITY ch01 SYSTEM " ch1.xml">
    <!ENTITY ch02 SYSTEM " ch2.xml">
    <!ENTITY ch03 SYSTEM " ch3.xml"> ]>
<livre><titre> cours xml</titre>
&ch01; &ch02; &ch03; </livre>
```

```
<?xml version="1.0"?>
<chapitre numero =‘3’>
<titre> troisième chapitre</titre>
<section> un</ section >
<section> deux</ section >
<section> trois</ section >
< /chapitre >
```

Exemple de fraction d'une DTD utilisant les entités générales

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<!DOCTYPE address [
```

```
    <!ELEMENT adresse (#PCDATA )>  
    <!ENTITY name 'tanmay'>  
    <!ENTITY company 'paint'>  
    <!ENTITY phone-no '(011)123-45678'>
```



Exemple de déclaration correcte dans le document XML:

```
<address>  
    &name  
  
    &company  
  
    &phone-no
```

Les valeurs de l'entité sont
référéncées en ajoutant le préfixe &
suivie du **nom de l'entité**;

Remarque :

Au moment de son interprétation, les références aux entités seront remplacées par leurs valeurs respectives

Syntaxe de déclaration d'une entité paramètre:

```
<!ENTITY %name-entité 'texte de remplacement'>
```

Exemple de fraction d'une DTD utilisant les entités paramètres:

```
<!ENTITY % commun "niveau, couleur">  
<!ELEMENT rectangle (%commun;, sommet+)>  
<!ELEMENT triangle (%commun;, sommet+)>  
<!ELEMENT disque (%commun;, centre, rayon)>
```

Exemple de fraction d'une DTD utilisant les entités:

```
<?xml version="1.0"?>  
<!DOCTYPE author [  
  <!ELEMENT author (#PCDATA)>  
  <!ENTITY email  
    "josmith@theworldaccordingtojosmith.com">  
  <!ENTITY js "Jo Smith &email;"> ]>
```



Exemple de déclaration correcte dans le document XML:

```
<author>&js;</author>
```

- Un élément peut posséder plusieurs attributs.
- Tous les attributs propres a un élément sont déclarés dans la même instruction

Exemple de fraction d'une DTD

```
<!ELEMENT livre EMPTY>  
<!ATTLIST livre titre CDATA #REQUIRED genre (roman |  
document | technique ) #DEFAULT " technique" #IMPLIED pages  
CDATA #REQUIRED >
```



Exemple de déclaration correcte dans le document XML:

```
<livre titre="Les DTD" genre="Technique" pages="60" />
```

- Le nombre d'apparitions d'un élément ne peut pas être contraint précisément on ne dispose que des quantifieurs ?,*,+
- **on ne peut pas dire qu'un élément doit apparaître plus de 3 fois mais toujours moins de 7.**
- On ne dispose pas de possibilité pour typer les contenus des éléments
- On ne dispose que d'un typage faible pour les valeurs des attributs
- On ne peut pas contraindre la forme de ces contenus (par exemple : entre 5 et 20 caractères, contenant un signe @.....)
- Il n'est pas possible de typer les références (elles peuvent référencier n'importe quel identifiant du document)
- Pour pallier ces manques d'autres propositions ont été faites , permettant de spécifier un langage XML de manière plus précise. Par exemple XML schéma et Relax NG

Ecrivez un document EX1.xml valide pour la DTD suivante: Mettez au moins deux années avec des blocs différents.

```
<!-- Racine -->
<!ELEMENT ecole (annee+)>
<!-- Année -->
<!ELEMENT annee (nom, age, matieres, effectif)>
<!ATTLIST annee bloc (maternelle | primaire | college | lycee) #REQUIRED>
<!-- Nom et prénom -->
<!ELEMENT nom (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT effectif (#PCDATA)>
<!-- Bloc matières -->
<!ELEMENT matieres (matiere+)>
<!ELEMENT matiere (#PCDATA)>
```

```
<?xml version="1.0" encoding="iso-8859-1"
standalone="no"?>
<!DOCTYPE ecole SYSTEM " ecole.dtd " >
```

```
< ecole >
```

```
<annee bloc = " lycee">
```

```
<nom>chayma</nom>
```

```
<age>21</age>
```

```
<matieres>
```

```
<matiere>XML</matiere>
```

```
<matiere>automate</matiere>
```

```
</matieres>
```

```
<effectif>30</effectif>
```

```
</annee>
```

```
<annee bloc = " college">
```

```
<nom>eya</nom>
```

```
<age>14</age>
```

```
<matieres>
```

```
<matiere>math</matiere>
```

```
<matiere>arab</matiere>
```

```
</matieres>
```

```
<effectif>25</effectif>
```

```
</annee>
```

```
</ ecole >
```