

OUTILS INTERNET II



Présenter par Mme Bensalem Hana
Université de Badji-Mokhtar Annaba

Promotion: MasterI ,GADM
Département d'informatique

CHAPITRE II: LES DOCUMENTS SUR LE WEB

- **Introduction :**
- Les pages sur le World Wide Web comportent
 - du texte, des liens hypertextes,
 - des objets multimédias,
 - des formulaires, etc.
- Afin que tout navigateur soit capable d'afficher une page web, un langage universel non propriétaire de publication est utilisé :HTML(Hypertexte Markup Language).
- Le navigateur reçoit le fichier html, et applique une mise en page dictée par une feuille de style(en CSS).



HTML

- **HTML** : un document html comporte d'une part le texte qui sera affiché dans le navigateur, et d'autre part, des balises telles que <title>et </title>.
- Quelques versions du HTML: le **HTML 2.0**, (1995), **HTML 3.2**(1997) , **HTML 4.01**(1999),
- XHTML: publié par le W3C, son objectif était de créer un langage de balisage qui respecterait les standards plus stricts de **XML**. Cependant, XHTML n'a pas été largement adopté.
- En 2004 le groupe **WHATWG** (Web Hypertext Application Technology Working Group), a collaboré avec le W3C (World Wide Web Consortium) pour développer **HTML5**
- **HTML5**, publié en 2014, a introduit de nombreuses nouvelles balises et fonctionnalités(pour le support multimédia, le dessin de graphiques, l'interactivité, le stockage local et la géolocalisation. Cette version contient 101 balises.
- .



LES BALISES HTML

- Les balises HTML définissent des directives de mise en page d'un texte encadré par un couple balise ouvrante, balise fermante:
 - Un titre: `<H2>xxxx</H2>`
- Les balises permettent :
 - Indiquer la structure (titre, sous-titre, paragraphes)
 - Caractériser des portions de texte (citation, extrait de code, définition, ...)
 - Produire des tableaux et formulaires.
 - D'inclure des ressources multimédias
 - Définir des liens hypertextes
 - De donner les méta-information sur le document tel que le titre, mots-clés, auteur, etc



LA STRUCTURE HTML

- Structure d'une Page HTML

- `<!DOCTYPE html>`

- `<html lang="fr">`

- `<head>`

- `<!-- Bloc entête contenant Métadonnées, scripts... -->`

- `</head>`

- `<body>`

- `<!-- Bloc affiché dans le navigateur -->`

- `</body>`

- `</html>`



L'ENTÊTE

L'entête ne s'affiche pas
dans le contenu de la page

- L'entête: La balise <head> délimite la partie du document qui spécifie :
 - Des **métadonnées** sur le document qui sont utilisables par le navigateur ou les moteurs de recherche (pour indexation)
 - Des **scripts** (ex: javascript, php) contenant du code utilisable pour l'affichage du corps du document
 - Des **liens vers des scripts externes** contenant du code utilisable pour l'affichage du corps du document
 - Des **liens vers des feuilles de style** gérant la mise en forme du document
 - Un **lien vers une image servant d'icône** dans l'onglet du navigateur



CONTENU HTML

- La balise <body> délimite le corps/contenu du document. C'est la partie à afficher par le navigateur et peut contenir :
 - Titre, sous-titre
 - Paragraphe
 - Listes ordonnées ou non
 - Liens hypertextes
 - Images
 - Regroupement
 - *Formulaires*
 - *ETC.*



LIMITES DU HTML

- Un langage non extensible
- Le langage HTML a été conçu pour être assez simple de sorte que le nombre et la signification des balises est limité. Il est figé.
- Un document non structuré; la définition limitée de la structure d'un document.
- Pas de vérification de la structure. Exemple: un document commençant par une tête de chapitre H2 et poursuivant par une tête de chapitre H1.
- La recherche doit considérer le document HTML comme une chaîne de caractères.
- Non séparation du document et de sa présentation graphique.
- Non-orienté contenu : pas d 'information sémantique et inadapté à l'échange de données



XML

introduit en 1986 par C. Goldfarb. pour des documentations techniques de grande ampleur.

Sa grande complexité a freiné son utilisation en dehors de ces projets.

- Dérive de SGML (Standard Generalized Markup Language) et de HTML (HyperText Markup Language).
- XML ((eXtended Markup Language)) est un langage orienté texte et formé de balises qui permettent d'organiser les données de manière structuré
- Utilisé pour le stockage de document et la transmission de données entre applications.
- Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis son adaptation à de multiples domaines.
- Sépare structure et présentation d'un document



XML(1)

- les noms des balises XML sont libres. Le vocabulaire (nom) des balises est fixé par les auteurs, de manière consensuel, lors d'échange de documents.
- Cette liberté dans les noms de balises permet de définir des vocabulaires particuliers adaptés aux différentes applications
- Interopérabilité: xml permet l'échange et traitement des données en utilisant de nombreux types de logiciels.
- Validation: certaines règles doivent être respectées pour garantir la validité de tel document(appelé DTD)
- Tout un arsenal de technologies ont été développé autour d'XML pour les outils manipulant de tel documents.



XML(2)

- Les langages apparentées à XML:
- **XLink et XPointer** : un mécanisme pour matérialiser des liens entre des éléments d'un ou plusieurs documents
- **XPath**: un langage d'expressions permettant de sélectionner des éléments dans un document XML. Il est la pierre angulaire du langage XSLT.
- **XQuery** permet d'extraire des informations à partir d'un ou plusieurs documents XML et de synthétiser de nouvelles informations à partir de celles extraites.
- **Schémas XML** :(modèles de documents) Pour la validation de document. Les schémas XML remplacent les DTD héritées de SGML pour décrire des modèles de documents.il est souple et beaucoup plus puissants que les DTD.
- **XSLT** exprime facilement des transformations complexes des documents XML en d'autres formats comme PDF ou des pages HTML.



XML(3)

- Dialectes: ont été définis pour appliquer XML à des domaines très variés.
 - partagent la même syntaxe de base de XML
 - peuvent être utilisés pour spécifier et manipuler ces documents.
 - nul besoin de développer des outils spécifiques

- Exemple:

XUL (XML-based User interface Language): Langage de description d'interfaces graphiques développé par le projet Mozilla.

SVG (Scalable Vector Graphics): Description de dessins vectoriels

SMIL (Synchronized Multimedia Integration Language): Description de contenus multimédia

SAML (Security Assertion Markup Language): Langage d'échange d'authentifications et d'autorisations

DocBook: Format de documentation technique, les projets Ant ou Android utilisent XML pour le stockage de données et les fichiers de configuration.

,etc.



LES API XML

- DOM: est une API inter-langage du World Wide Web Consortium (W3C) pour accéder et modifier les documents XML.
- Pour PHP:
 - simpleXML: fournit des outils pour manipuler des ressources XML. Le principe est de transformer le XML en une hiérarchie d'objets PHP appelés [SimpleXMLElement](#). Pour plus d'info veuillez consulter: [Autres API XML \(univ-nantes.fr\)](#)
 - XmlReader
 - xmlWriter
- Pour Java :
 - **DOM**: Document object model
 - DOM4J est une API Java open source permettant de lire et écrire du XML, et faire des requêtes XPath sur des documents XML. Elle s'appuie assez largement sur l'API Collections, beaucoup moins lourde et plus performante que les API de références que sont Xerces et Xalan.
 - **JAXP** : Java API for XML parsing
 - **SAX** : Simple API for XML
 - **TrAX** : Transformation API for XML
 - **Xalan** : processeur XSLT
 - **Xpath**
- **Pour Python**: ElementTreeXML...

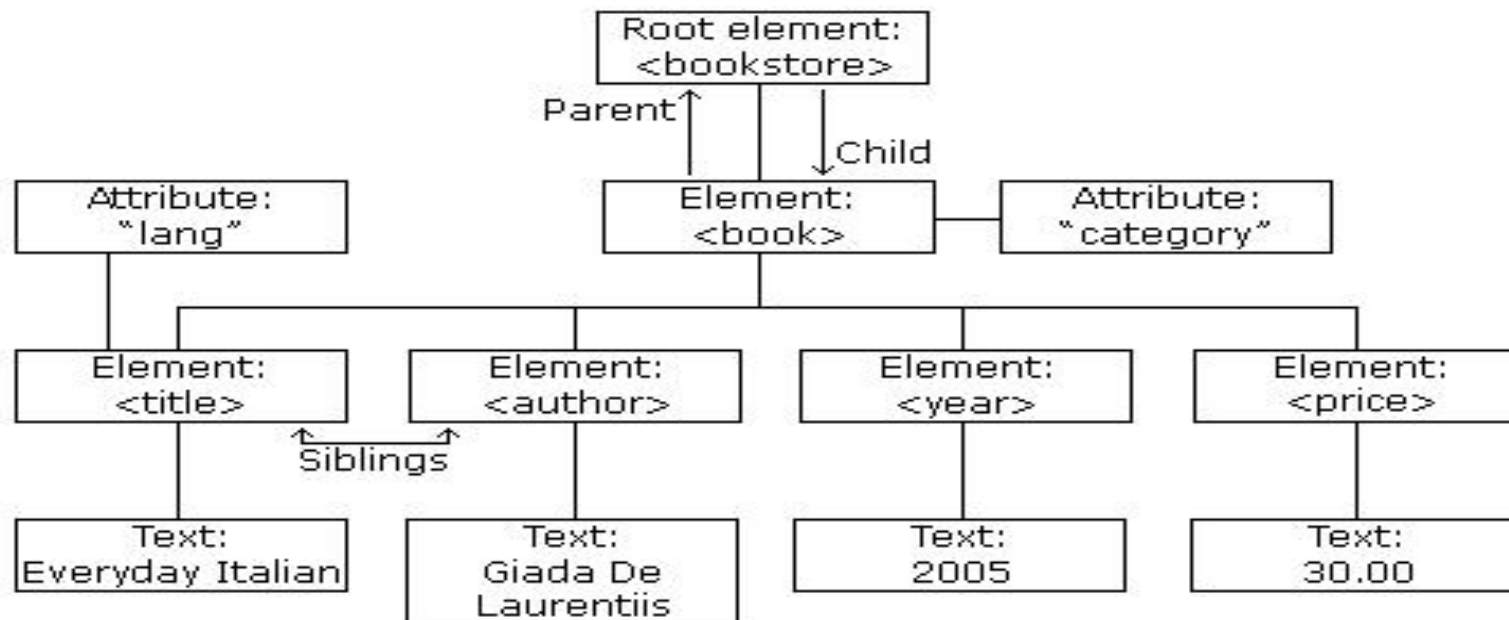
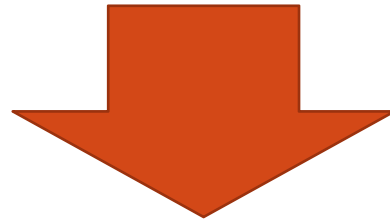


HTML&XML

- XML utilisé comme complément de Html, dans certaines applications web
 - Html est utilisé pour formater et présenter les données,
 - xml est utilisé pour stocker et transmettre les données



```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```



LA STRUCTURE ARBORESCENTE DANS XML

- Est un arbre d'éléments imbriqués, qui peuvent être:
 - Un seul élément racine(root)
 - des éléments enfant(Child)
- Chacun peut avoir :
 - Des attributs
 - Un contenu texte



STRUCTURE D'UN CONTENU XML

`<?xml version="1.0" encoding="UTF-8"?>` ← le prologue XML(optionnel)

`<message>` ← Une racine unique → `<root>`

`<note type="Note for Student">` ← 1^{ier} nœud fils

`<to>Master I GADM</to>`

`<from>Teacher</from>`

`<notification>Don't forget me this weekend!</notification>`

`</note>`

`</message>`

`<child>`

`<subchild>.....</subchild>`

`</child>`

`</root>`

Les fils du fils

la fermeture des balises doit se faire l'ordre



```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<message>
```

```
<note type="Note for Student">
```

```
<to>Master I GADM</to>
```

```
<from>Teacher</from>
```

```
<notification>Don't forget me this  
weekend!</notification>
```

```
</note>
```

```
</message>
```

ÉLÉMENT

- Peut contenir du texte, des attributs, d'autre éléments(des fils) ou un mélange de tout ces composants.
- Les éléments dans cet exemple :
`<message><note>...`
- `<to>`, `<from>`, `<notification>`: contiennent du texte.
- `<message>`, `<note>` contiennent des éléments.
- `<note >` a un attribut ,`type= "Note for Student"`
- Un élément vide est sans contenu
`<elementvide></elementvide>` ou
`<elementvide/>`



LES ATTRIBUTS

- Un élément peut avoir un ensemble d'attributs : type est un attribut de note.
- Paraissent dans la même balise de l'élément concerné.
- Les valeurs des attributs XML doivent toujours être entre côtes ou guillemets:
'Note' or " Note«
- Chacun ne dispose que d'une seule valeur;

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<message>
```

```
<note type='Note for Student'>
```

```
<to>Master I GADM</to>
```

```
<from>Teacher</from>
```

```
<notification>Don't forget me this weekend!</notification>
```

```
</note>
```

```
</message>
```

Différents structuration de la même information

<pre><note date="2024-02-16"> <to>Student</to> <from>Teacher</from> </note></pre>	<pre><note> <date>2024-02-16</date> <to>Student</to> <from>Teacher</from> </note></pre>	<pre><note> <date> <year>2024</year> <month>02</month> <day>16</day> </date> <to>Student</to> <from>Teacher</from> </note></pre>
---	---	--



<notification>Don't forget me this weekend!</notification>

On veut lui ajouter la date



<notification: content>Don't forget me this weekend!</notification: content>

<notification: date>16/02/2024</notification: date>

BALISE À PLUSIEURS INFORMATIONS

- une balise avec deux types différents d'information=> **un conflit** au niveau de cette balise.
- Différencier les deux types de valeurs par modification des balises :

<notification : content>, <notification:
date>



```
<note type='Note for Student'>
<to>Master I GADM</to>
<from>Teacher</from>
<notification>Don't forget me this
weekend!</notification>
</note>
```

```
<note >
<module>outils internet</module>
<Student:familynam>Lapointe</Student:familynam>
<Student:name>ALI</Student:name>
<notecc>14</ notecc >
<notexam>15</ notexam >
</note>
```



CONFLIT DE NOMMAGE

- Lors de combinaison de plusieurs documents XML de différents applications chacun.
- ceci conduit à un conflits, deux éléments portant le nom avec des structure hétérogènes
- Il faut utiliser un préfixe:
- Exemple:

```
<a:note type='Note for Student'>
....
</a:note>
<b:note>
<module>...</notexam>
</b:note>
```



```
<root xmlns:a= " URI1 "  
  xmlns:b= " URI2" >  
  <a:note type='Note for Student' >  
  ...  
</a:note>  
<b:note >  
  ...  
</b:note>
```

```
<racine xmlns:dc="http://www.purl.org/dc/elements/1.1/" >  
  ...  
<dc:title>langage de formel</dc:title>  
<dc:creator></dc:creator>  
  ■ <dc:date></dc:date>  
  ...  
  ■ </racine>
```

LES ESPACES DE NOM, OU XML NAMESPACE (XMLNS)

- Est un moyen d'éviter les conflits de nommage.
- Réutiliser les nombreux dialectes XML ayant été définis pour des utilisations diverses
- Associer une URI (Uniform Resource Identifier) à un préfixe.
- défini par l'attribut

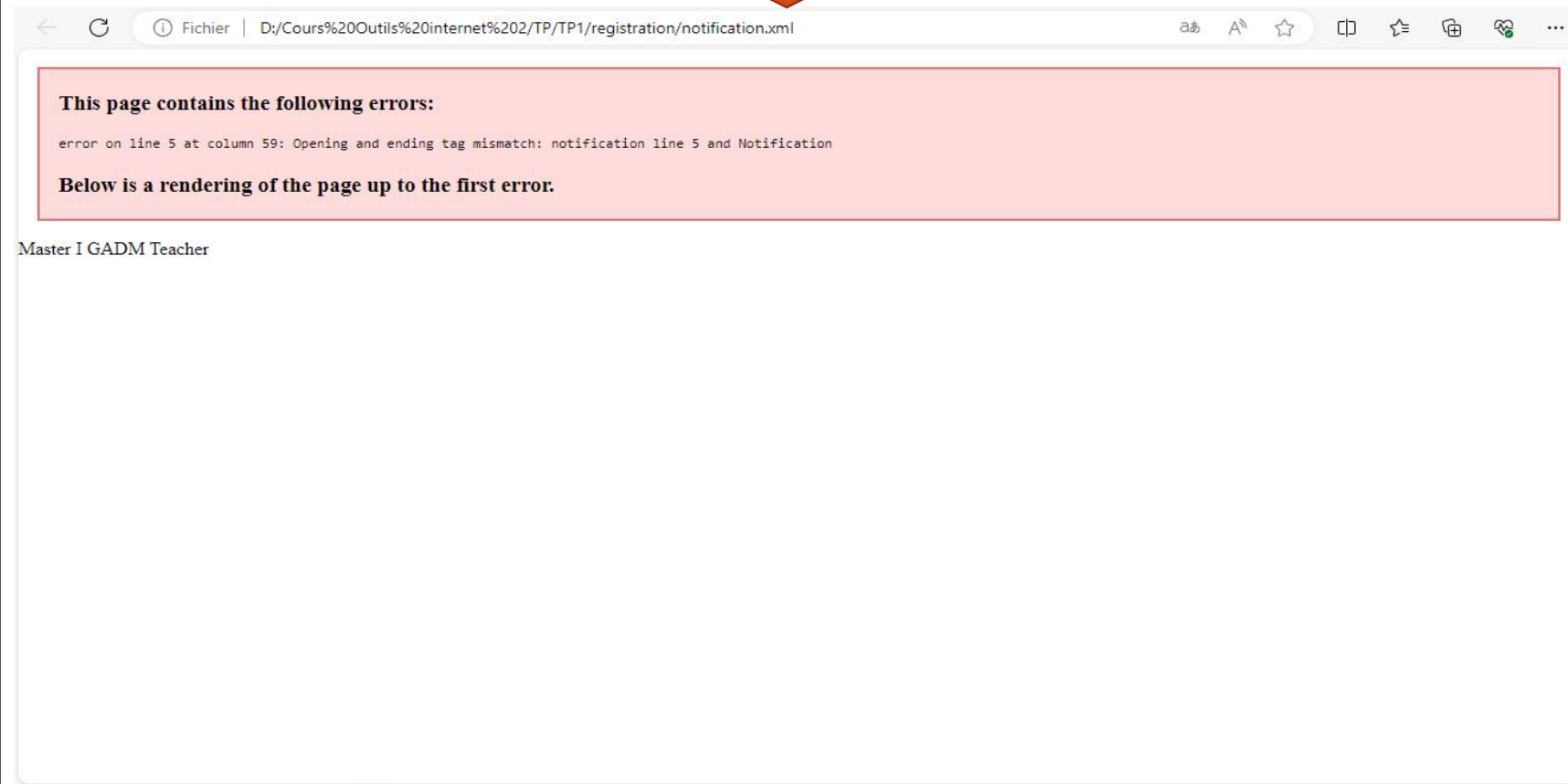
Xmlns dans la balise ouvrante:

Xmlns: prefix='URI'

- L'élément et tout ses fils seront associés au namespace associé à cette URI
- L'objectif est de donner un nom unique au namespace



```
<message>
<note type='Note for Student'>
<to>Master I GADM</to>
<from>Teacher</from>
<notification>Don't forget me this weekend!</Notification>
</note>
</message>
```



XML AVEC ERREUR

Exemple d'erreurs détectées par le navigateur:

Les éléments sont sensibles à la casse: notification ≠ Notification



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<message>
  <note type="Note-for-Student" age="30">
    <to>Master I GADM</to>
    <from>Teacher</from>
    <notification>Don't forget me this weekend!</notification>
  </note>
</message>
```

XML BIEN FORMÉ (WELL FORMED)

- Doit avoir un seul element racine
- Chaque élément doit avoir une balise fermante
- Les noms des balises sont sensible à la casse
- Les balises doivent être imbriquées correctement
- Les valeurs d'attributs doivent être entre guillemets.



- DTD: Document Type Definition
- DTD externe: insérer le lien au début du document

`<!DOCTYPE root-element SYSTEM "file.dtd">`

ou `< !DOCTYPE root-element PUBLIC "URL">`

- DTD interne: est incluse dans le document même.

XML VALIDE

- Un document XML bien formé est un document qui respecte la syntaxe XML.
- Un document XML valide est un document bien formé
- se conforme à la définition de la grammaire (DTD ou Schéma) à laquelle il est associé.



```
<?xml version="1.0"?>
<!DOCTYPE annuaire [
<!ELEMENT annuaire (entree)>
<!ELEMENT entree (nom,telephone*)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT telephone (#PCDATA)>
] >
<annuaire>
<entree>
...
</entree>
```

</annuaire>

<!DOCTYPE root-element[declarations]>

DTD

DTD :une DTD définit les balises autorisées, leurs attributs et leur enchaînement

DTD interne :

C'est une DTD qui peut être incluse dans le document, son contenu est encadré par : '[' et ']', sous-forme:

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



- **PCDATA** : Parsed Character Data. Du texte contenu dans un élément, de la forme : `<element>text</element>`
- **CDATA** : Character Data ou chaîne de caractère non parsées, utilisé dans les attributs
 - exemple `<element attribut = "<hello/>">` .
 - `<hello/>` traitée comme une chaîne de caractère, et non une balise XML.
- **ID** et **IDREF** : permettent de lier différentes parties d'un document XML.
- ...

TYPES DE DONNÉES

Il y a 10 types de données définis dans la DTD. Les plus utilisées sont : PCDATA et CDATA.



- `<!ELEMENT nom_balise spécification contenu>`

Type	DTD	XML
Elément vide	<code><!ELEMENT elt EMPTY></code>	<code><elt/></code>
Elément contenant du texte	<code><!ELEMENT elt (#PCDATA)></code>	<code><elt>texte</elt></code>
Elément avec sous éléments	<code><!ELEMENT elt (sous-elt)></code> <code><!ELEMENT sous-elt EMPTY></code>	<code><elt></code> <code><sous-elt/></code> <code></elt></code>
Elément avec plusieurs sous éléments	<code><!ELEMENT elt (s1, s2)></code> <code><!ELEMENT s1 EMPTY></code> <code><!ELEMENT s2 EMPTY></code>	<code><elt></code> <code><s1/></code> <code><s2/></code> <code></elt></code>
Elément avec contenu variable	<code><!ELEMENT elt (#PCDATA s1)></code> <code><!ELEMENT s1 EMPTY></code>	<code><elt>texte</elt></code> ou <code><elt><s1/></elt></code>
Elément à contenu non défini	<code><!ELEMENT elt ANY></code> <code><!ELEMENT s1 EMPTY></code>	<code><elt>texte</elt></code> ou <code><elt><s1/></elt></code> etc.

LES ÉLÉMENTS

Un élément vide est sans contenu
`<elementvide></elementvide>` ou
`<elementvide/>`

Un élément feuille est sans fils=> ayant
seulement le type de la valeur de l'entité
qu'il englobe.

Un élément complexe qui contient à son tour
d'autre éléments



- `< !ELEMENT element(el)>` :
- Si `el=S` $\Rightarrow$ S peut avoir une et une seule occurrence
- Si `el=S*` $\Rightarrow$ S peut avoir 0 ou n occurrence
- Si `el=S ?` $\Rightarrow$ S peut avoir 0 ou 1 occurrence
- Si `el=S+` $\Rightarrow$ S peut avoir au moins 1 occurrence

CARDINALITÉ DES ÉLÉMENTS



DTD	XML
<pre><!ELEMENT elt EMPTY> <!ATTLIST elt attribut CDATA "default"></pre>	<pre><elt attribut="valeur"/> Par défaut : <elt attribut="default"/></pre>
<pre><!ELEMENT elt EMPTY> <!ATTLIST elt attribut ("v1" "v2") "v1"> 2 valeurs possibles : "v1" et "v2"</pre>	<pre><elt attribut="v2"/> ou (défaut) : <elt attribut="v1"/></pre>
<pre><!ELEMENT elt EMPTY> <!ATTLIST elt att1 CDATA "d1" elt att2 CDATA "d2"></pre>	<pre><elt att1="a" att2="b"/> par défaut: <elt att1="d1" att2="d2"/></pre>

LES ATTRIBUTS D'ELEMENTS

La forme de déclaration :

```
< !ATTLIST nom-element nom-attrib1 type
default-value nom-element nom-attrib2
default-value >
```

Exemple:

```
<!ATTLIST Livre lang CDATA>
```



Les attributs peuvent être (dans le cas où il est un élément vide):

- Obligatoire :

en DTD:

< !ATTLIST elt att CDATA #**REQUIRED**>

En xml: <elt att= "a"/>

- Non obligatoire :

en DTD:

< !ATTLIST elt att CDATA #**IMPLIED**>

En xml: <elt att= "a"/>

OU <elt/>

- à valeur fixe :

< !ATTLIST elt att CDATA #**FIXED** "a" >

En xml <elt att= "a"/>

LES ATTRIBUTS D'ÉLÉMENTS

Les valeurs des attributs XML doivent toujours être entre guillemets

Les attributs peuvent être:

Obligatoire

Non obligatoire ou à valeur fixe



- `< !ELEMENT nom "valeur">` ou `< !ELEMENT nom SYSTEM "file-ou-URL">`

- Exemple :

`< !ELEMENT eacute "! ;">`

dans XML s'écrit : `<balise>é ;gale</balise>` ⇔ égale

`< !ELEMENT explication SYSTEM "fil.xml">`

- Exemple: `< citation> &explication ;< /citation>`

sera remplacée par :

`< citation> contenu du fichier file.xml< /citation>`

LES ENTITÉS

Sont des unités de stockage des composants d'un document

Les entités peuvent être considérées comme des variables .

Certains caractères sont interdit dans XML, il doivent être remplacer par des références à ces entités



<message>salary < 1000</message>

ERREURS



<message>salary < 1000</message>

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

RÉFÉRENCE D'ENTITÉS

Certains caractères sont interdits dans XML, ils doivent être remplacés par des références à ces entités



```
<!ELEMENT personne (#PCDATA)>
<!ELEMENT parent (fils*)>
<!ELEMENT fils EMPTY>
<!ATTLIST personne identifiant ID #REQUIRED>
<!ATTLIST parent identifiant IDREF #REQUIRED>
<!ATTLIST fils identifiant IDREF #REQUIRED>
<!-- XML -->
<personne identifiant="A">Mohamed</personne>
<personne identifiant="B">Amine</personne>
<parent identifiant="A">
<fils identifiant="B"/>
</parent>
```

IDENTIFICATEURS

- **Les identifiants** : Un attribut de type **ID** permet d'attribuer un identifiant à un élément d'un document XML.
- Cet élément sera référencé dans une autre partie du même document par un attribut de type **IDREF**.
- Les identifiants doivent toujours commencer par une lettre.



- Un schéma XML est un fichier XML dont l'élément racine est :
<xsd:schema>
- s'il n'y a pas d'espace de nom alors
xsi:noNamespaceSchemaLocation="URI/schema.xsd"

```
<?xml version="1.0" encoding="iso-8859-1"?>  
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
<!-- Déclarations d'éléments, d'attributs et types -->  
...  
</xsd:schema>
```

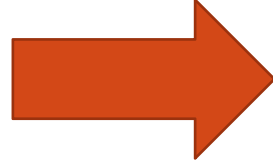
XML-SCHEMA

- est une amélioration des DTD.
- permet la spécification de : types d'éléments, Les relations de structuration, Les relations de spécialisation. Le typage des données (string, entier, réel..).
- Fournit les moyens d'exprimer : Les restrictions (cardinalités), et les collections d'éléments...
- se compose de déclarations d'éléments et d'attributs et de définitions de types.



```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

DTD



```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schema xml



- **Référencer un schema à partir d'un xml :**

- un schéma de validation local :

- **<element-racine**

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:noNamespaceSchemaLocation="reference.xsd">
```

- Lorsque le schéma est public, mis sur un serveur

- **< element-racine**

```
xmlns="http://www.iut-lannion.fr"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.iut-lannion.fr reference.xsd">
```

RÉFÉRENCER XML SCHEMA

- Se fait dans la balise ouvrante de l'élément racine.

- Via les attributs:

schemaLocation ou
noNamespaceSchemaLocation

- Le premier est utilisé lors de l'utilisation d'un espace de noms, le second dans le cas contraire

- Ces attributs se trouvent dans l'espace des instances schémas identifié par l'URI:

<http://www.w3.org/2001>

[/XMLSchema-instance](http://www.w3.org/2001/XMLSchema-instance)



- Exercice: donner un exemple de document valide par rapport à la suivante DTD:

```
1  <!DOCTYPE inscriptions [  
2  <!ELEMENT inscriptions (ue+,etudiant*)>  
3  <!ELEMENT ue (titre, resume, inscrit*)>  
4  <!ATTLIST ue code ID #REQUIRED  
5           niveau CDATA #REQUIRED>  
6  <!ELEMENT titre (#PCDATA)>  
7  <!ELEMENT resume (#PCDATA)>  
8  <!ELEMENT inscrit EMPTY>  
9  <!ATTLIST inscrit num IDREF #REQUIRED  
10           semestre CDATA #REQUIRED>  
11 <!ELEMENT etudiant (nom,adresse)>  
12 <!ATTLIST etudiant num ID #REQUIRED>  
13 <!ELEMENT nom (#PCDATA)>  
14 <!ELEMENT adresse (#PCDATA)>  
15 ]>
```

EXERCICE.1

À le faire sous forme
d'enregistrement



```
<?xml version="1.0"?>
<!DOCTYPE inscriptions SYSTEM "inscriptions.dtd">
<inscriptions>
  <ue code="BDAV" niveau="4">
    <titre>Bases de donnees avancees</titre>

    <resume>
      XML, Datalog, Securite
    </resume>
    <inscrit num="123456" semestre="2010/Automne"/>
  </ue>
  <etudiant num="123456">
    <nom>Toto</nom>
    <adresse>Villeurbanne</adresse>
  </etudiant>
</inscriptions>
```

SOLUTION.1

Notepad++:

- Module d'extension>gestion des modules d'extension>
- selectionner xmltools puis install

- SI ERREUR =DTD interdite=>

Module d'extension>xmltools>
option>MSXML feature> prohibitDTD

- Cliquer sur prohibitDTD plusieurs fois jusqu'à changer à sa valeur à False
- Puis valider à nouveau



- Une bouteille d'eau Cristaline de 150cl contient par litre 71mg d'ions positifs calcium, et 5,5mg d'ions positifs magnésium. On y trouve également des ions négatifs comme des chlorures à 20mg par litre et des nitrates avec 1mg par litre. Elle est recueillie à St-Cyr la Source, dans le département du Loiret. Son code barre est 3274080005003 et son pH est de 7,45. Comme la bouteille est sale, quelques autres matériaux comme du fer s'y trouvent en suspension.
- <bouteilles>
- <bouteille>
- <marque> </marque>
- </bouteille>
- </bouteilles>
- Une seconde bouteille d'eau Cristaline a été, elle, recueillie à la source d'Aurèle dans les Alpes Maritimes. La concentration en ions calcium est de 98mg/l, et en ions magnésium de 4mg/l. Il y a 3,6mg/l d'ions chlorure et 2mg/l de nitrates, pour un pH de 7,4. Le code barre de cette bouteille de 50cl est 3268840001008.
- Une bouteille de même contenance est de marque Volvic, et a été puisée à...Volvic, bien connu pour ses sources donnant un pH neutre de 7. Elle comprend 11,5mg/l d'ions calcium, 8,0mg/l d'ions magnésium, 13,5mg/l d'ions chlorures et 6,3mg/l d'ions nitrates. Elle contient également des particules de silice. Son code barre est 3057640117008.

EXERCICE.2

Réorganiser les informations contenue dans ce paragraphe sous forme d'un fichier xml.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<cave>
```

```
  <bouteille>
```

```
    <marque>Cristaline</marque>
```

```
    <composition>
```

```
      <ion type="positif">calcium 71mg/l</ion>
```

```
      <ion type="positif">magnésium 5,5mg/l</ion>
```

```
      <ion type="negatif">chlorure 20mg/l</ion>
```

```
      <ion type="negatif">nitrate 1mg/l</ion>
```

```
      <autre type="metal">fer</autre>
```

```
    </composition>
```

```
    <source>
```

```
      <ville>St-Cyr la Source</ville>
```

```
      <departement>Loiret</departement>
```

```
    </source>
```

```
    <code_barre>3274080005003</code_barre>
```

```
    <contenance unit="cl">150</contenance>
```

```
    <ph>7,45</ph>
```

```
  </bouteille>
```

SOLUTION.2



```
<bouteille>
  <marque>Cristaline</marque>
  <composition>
    <ion type="positif">calcium 98mg/l</ion>
    <ion type="positif">magnésium 4mg/l</ion>
    <ion type="negatif">chlorure 3,6mg/l</ion>
    <ion type="negatif">nitrate 2mg/l</ion>
  </composition>
  <source>
    <ville>Aurèle</ville>
    <departement>Alpes Maritimes</departement>
  </source>
  <code_barre>3268840001008</code_barre>
  <contenance unit="cl">50</contenance>
  <ph>7,4</ph>
```

</bouteille>

```
<bouteille>
```

```
  <marque>Volvic</marque>
  <composition>
    <ion type="positif">calcium 11,5mg/l</ion>
    <ion type="positif">magnésium 8mg/l</ion>
    <ion type="negatif">chlorure 13,5mg/l</ion>
    <ion type="negatif">nitrate 6,3mg/l</ion>
  </composition>
  <source>
    <ville>Volvic</ville>
    <departement>Puy-de-Dôme</departement>
  </source>
  <code_barre>3057640117008</code_barre>
  <contenance unit="cl">50</contenance>
  <ph>7</ph>
```

</bouteille>

</cave>

SOLUTION.2



RÉFÉRENCE

-  **Nicolas Lumineau**
- [1.1 Introduction à XML \[Mode de compatibilité\] \(univ-msila.dz\)](#)
- [XML Examples \(w3schools.com\)](#)
-

