



Université Ibn Zohr
Faculté des sciences – Département Informatique

XML eXtensible Markup Language

Ayoub SABRAOUI

Master OTI –S1

Année universitaire 2016/2017

1

Plan

■ Partie I : Le standard XML

- Objectifs
- Pourquoi XML ?
- Structure d'un document XML
- Document bien formé

■ Partie II : Definition des documents XML

- DTD
- XML Schema



■ Partie III : Mise en forme, Traitement et Transformations des documents XML

- DOM (Document Object Model)
- XPath (Chemins d'accès au arbre XML)
- Transformations XSLT

2

Objectifs


■ On veut représenter des données

- Facilement **lisibles** : 
 - par les **humains**
 - par les **machines**
- Selon une technologie **compatible WEB**
(à intégrer facilement dans les serveurs WEB)
- en séparant les aspects : 
 - **présentation** (*format, couleurs etc..*)
 - **information** (*données*)
- D'une manière **standardisée**

3

Etat de l'art

■ Formats existants :

- **HTML** = **HyperText** Markup Language
 - **SGML** = **Standard Generalized** Markup Language
- 
- *Langage à balises*

■ Autres notations :

- **ASN.1** = Abstract Syntax Notation (ITU-T)
- CDR, XDR = Common/eXternal Data Representation
- etc.....

4

Critique de HTML

- Langage **simple, lisible !** (*texte formaté*)
- **Compatible WEB !**
- **Non extensible !** (*Nombre fixe de balises et attributs*)
- **Mélange des genres !**
(*i.e. balise de structuration et de mise en forme : <H1> title 1 </H1>*)
- **Incompatibilité** entre navigateurs et versions !
- **Pas de preuve** sur le document
 - structure (*ordre des balises*),
 - données (*type, valeur*),
 - sémantique

5

Critique de SGML

- Langage **puissant, extensible, standard (ISO 8879-1986)!**
- **Méta-langage de documentation pour grosses applications**
(*i.e. automobile, avion, dictionnaire, etc...*)
- ...mais
- **Trop complexe ! -> Implémentation beaucoup trop lourde !**
- **Pas forcément compatible WEB !**

6

XML

Définition intuitive d'XML:

- XML :
 - variante de **HTML généralisé !**
(compatibilité WEB, lisibilité, syntaxe)
 - **sous-ensemble de SGML !**
(flexibilité, rigueur)
- **langage à balises configurables**
- pour la représentation hiérarchique de données
- <http://www.w3.org/XML/>

7

XML, qu'est-ce que c'est ?

- balises descriptives (signification des données) plutôt que procédurales (présentation des données)
- libre, indépendant des plateformes logicielles ou matérielles
- XML est extensible: ne contient pas un ensemble fixe de balises
- les documents XML doivent être bien formés suivant une syntaxe définie, et peuvent donc être formellement validés
- XML est particulièrement adapté à l'échange de données et de documents.

8

XML, qu'est-ce que c'est ?

Parsers et Décodage des documents XML

- L'extraction des données d'un document XML se fait à l'aide d'un outil appelé analyseur syntaxique (en anglais **parser**, ou parseur) qui permet :
 - d'extraire les données d'un document XML (analyse du document ou parsing)
 - éventuellement, de vérifier la validité du document.

9

DEFINITION

- eXtensible Markup Language
 - Recommandation (norme) du W3C
 - Spécifiant un langage
 - Constitué d'un ensemble d'éléments appelés balises
 - Utilisable pour créer d'autres langages
- 2 concepts fondamentaux
 - Structure et présentation sont séparés
 - Les balises ne sont pas figées

10

DEFINITION

■ Conséquences :

- XML est un format de document
- XML est un format de données (dialectes)
- XML est un méta-langage (une famille de langages)

■ En simplifié :

- « XML est un langage de description de documents structurés » (www.w3c.org/XML).

11

INTERÊT de XML

■ Richesse sémantique

- Dédié au traitement des données
- Soutenant une grande variété d'applications

■ Facilité de mise en œuvre

- Simple et lisible
- Portable et facilement utilisable sur Internet
- Assurant un développement aisé

12

Structure de documents XML

■ Prologue :

- Rôle équivalent au <HEAD> HTML,
- Meta-Information : {
 - **Instructions** de traitement
 - **commentaires**
(non interprétable par le parseur)

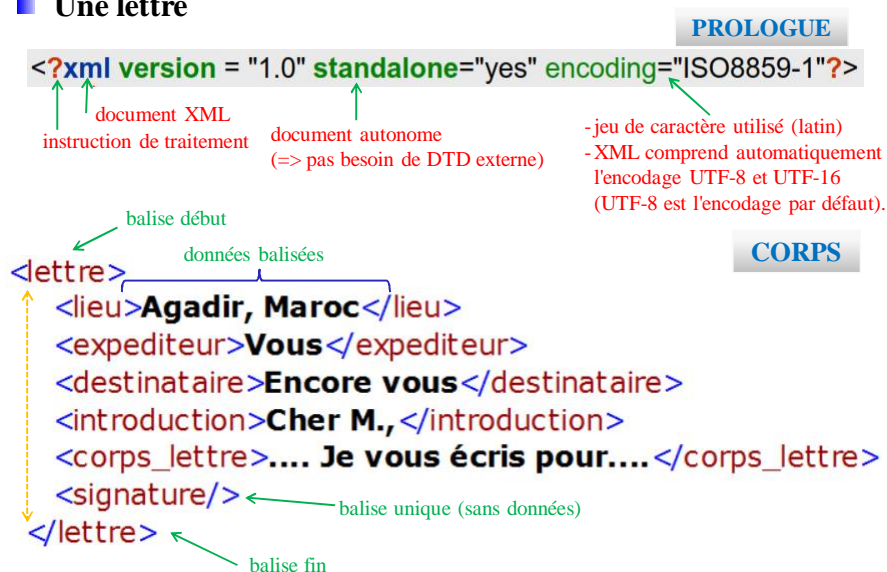
■ Corps :

- Rôle équivalent au <BODY> HTML
- Les données formatées: (structure arborescente) {
 - Balises d'encadrement
 - **Attributs associés** aux balises
 - **Données encadrées** par les balises

13

Exemple XML

■ Une lettre



Prologue d'un document XML

■ Exemple

document XML 1.0

ceci est un document XML non autonome
(il utilise une définition externe)

```
<?xml version="1.0" standalone="no" encoding="ISO8859-1" ?>
<!DOCTYPE liste_CD SYSTEM "CDs.dtd">
```

un commentaire spécial !
(il définit le type de document XML)

conforme à une définition externe
(spécifié dans le fichier "CDs.dtd")

15

Corps d'un document XML

■ Exemple

```
<liste_CD>
  <CD>
    <artiste type="individuel">Artiste un</artiste>
    <titre no_pistes="4">titre un</titre>
    <pistes>
      <piste>piste 1</piste>
      <piste>piste 2</piste>
    </pistes>
    <prix devise="dirham" paiement="CB">30.00</prix>
    <en_vente/>
  </CD>
  <CD>.....</CD>
</liste_CD>
```

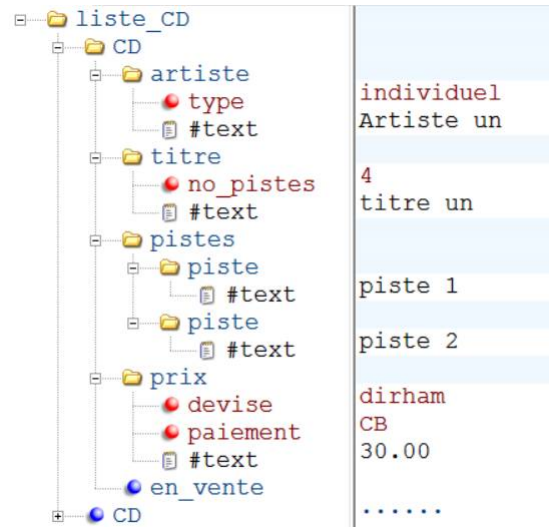
Diagram illustrating the structure of an XML document body with numbered annotations:

- 1: Start of the root element `<liste_CD>`
- 2: End of the root element `</liste_CD>`
- 3: Start of a child element `<CD>`
- 4: End of a child element `</CD>`
- 5: Start of an element `<artiste>`
- 6: End of an element `</artiste>`
- 7: End of an element `</prix>`
- 8: Start of an element `<pistes>`
- 9: End of an element `</pistes>`

16

Corps d'un document XML

■ Arbre des balises sur l'exemple



17

Corps d'un document XML

Explications sur l'exemple

- Balisage arborescent (*voir le transparent 12*)
- La **racine** du corps est **unique (1)(2)**.
- Les balises sont soit :
 - par paires : début (1), et fin (2),
 - uniques (4).
- Le contenu entre deux balises paires (3) est soit :
 - une valeur simple : chaîne de caractère (6), numéro réel (7), etc.,
 - une arborescence d'autres balises (9).
 - un mélange des deux (*pas présent dans l'exemple*).
- Certaines balises (de début) contiennent des **attributs (5)(8)**,

18

Structure des documents XML

Synthèse

- Un document XML : Prologue + Corps
(un arbre de balises)

- Balises du prologue :

`<?nom_balise_traitement ...?>`

Déclaration

`<!DOCTYPE ...>`

Type de document

- Balises du corps par paires (conteneurs pour les données) ou uniques

`<nom_balise nom_attribut1="val" nom_attribut2="val"> contenu
</nom_balise>`

`<nom_balise_simple/>`

XML : les commentaires

- en XML les commentaires se notent :

`<!-- texte du commentaire -->`

- Les contraintes d'utilisation sont

- pas de double tirets dans le texte,
- pas de commentaire dans un élément (l'exemple ci-dessous est incorrect),

`<produit
 nom="DVD"
 prix='100' <!-- en euros -->
>`

- les commentaires sont ignorés (plus ou moins),

XML : les balises (éléments)

■ Forme générale :

`<nom_d_élément> contenu </nom_d_élément>`

■ Les noms sont libres (contrairement à HTML). Ils obéissent à quelques règles:

- 1er caractère { alphabétique, «-», «_» },
- les autres caractères { alphabétique, chiffre, «-», «_», «:» }.
- pas de blanc,
- «xml» au début est interdit (maj./min.).

■ La balise de fermeture est obligatoire.

21

XML : les balises (éléments)

■ Le contenu d'un élément peut être

- vide (`<toc></toc>` ou `<toc/>`),
- du texte (sauf «<» et «&») basé sur l'encodage,
- un ou plusieurs éléments complets
`<toc> ... </toc>`
- une répétition de textes et d'éléments,
`<article> Le langage <def>XML</def> contient <liste>`
`<élément> du texte, </élément>`
`<élément> des éléments, </élément>`
`</liste></article>`
- Les blancs comptent: `<a> X ` est différent de `<a>X`.

22

XML : arbre d'éléments

- Un document XML est un et un seul arbre d'éléments. C'est à dire :
 - Pas de chevauchement d'éléments. La notation suivante :


```
<list> ... <item> ... </list> ... </item>
```

 est invalide. Il faut la corriger comme suit


```
<list> ... <item> ... </item> ... </list>
```
 - Un document XML est composé d'un seul élément. La notation suivante :


```
<?xml version="1.0" encoding="iso-8859-1" ?>
<article> ... </article>
<article> ... </article>
```

 est invalide. Il faut la corriger comme suit


```
<?xml version="1.0" encoding="iso-8859-1" ?>
<stock>
  <article> ... </article>
  <article> ... </article>
</stock>
```

23

XML : les attributs

- Un élément ouvrant peut être enrichi par des couples de la forme **attribut1="valeur1"** comme dans l'exemple


```
<produit nom="DVD" prix='200'>
```
- Syntaxe : **nom='valeur' ou nom="valeur"**
- La valeur doit être entourée d'apostrophes si elle contient des guillemets, et inversement.
- Caractères interdits : ^, % et &
- Le nom des attributs suit les mêmes règles syntaxiques que les noms d'éléments.
- Attributs comme ci-dessus ou sous-éléments ?


```
<produit>
  <nom>DVD</nom> <prix>150</prix>
</produit>
```
- L'attribut doit changer l'interprétation des données:


```
<prix monnaie="Euro"> 150 </prix>
```

24

XML : les attributs réservés

- `xml:lang='langue'` permet de définir la langue utilisée dans l'élément et tous les sous-éléments.

La langue suit la norme ISO 3166 définie par la RFC 1766 (Request For Comment). Par exemple `fr` ou `en-US` ou `fr-FR`.

- `xml:space='preserve'` ou `xml:space='default'` permet de définir l'interprétation des espaces dans l'élément et tous les sous-éléments.
- `xml:id='identificateur'` permet d'associer une et une seule clef à un élément .
- `xml:idref='identificateur'` permet de faire référence à une clef.

```
<section id='intro'>
  <titre>introduction à XML</titre>
  ... </section>

<section>
  <p> après la section
  <xref idref='intro'>d'introduction</xref>
  nous allons passer au plat de résistance...
</section>
```

25

XML : les références d'entités

- Les entités sont des fragments de document XML définis dans la DTD. La référence d'entité se note :

`&nom_de_l_entité;`

- Il existe des entités prédéfinies :

```
* &amp; donne &
* &lt; donne <
* &gt; donne >
* &quot; donne «
* &apos; donne '
* &#nnn; donne le caractère de code décimal nnn,
* &#xnnn; donne le caractère de code hexadécimal nnn,
```

- Un exemple :

```
<texte> en HTML, la balise
&lt;p&gt; est très utile !
&#169;
</texte>
```

26

XML : section littérale

- Avec les sections littérales Il est possible de stopper l'interprétation des caractères spéciaux. La syntaxe est la suivante :

`<![CDATA[texte non soumis à l'analyse]]>`

L'exemple précédent devient

`<texte><![CDATA[en HTML, la balise
<p> est très utile !]]>
© </texte>`

27

XML : espaces de noms

- Un problème apparaît si on mélange deux textes XML dont les éléments ont le même nom. Par exemple

```
<produit>
  <nom>...</nom>
  <desc>...</desc>
</produit>

<fournisseur> <nom>...</nom>
  <desc> <adr>...</adr>
  <tél>...</tél> </desc>
</fournisseur>
```

- Pour régler ce problème on enrichit le nom de l'élément :

```
<fsa:produit
  xmlns:fsa=http://www.fsa.univ-ibn-zohr.ma>
  <fsa:nom>...</fsa:nom>
  <fsa:desc>...</fsa:desc>
</fsa:produit>
```

28

XML : espaces de noms

- Attention, le préfixe n'est qu'une macro. C'est l'espace de nom qui compte. Les deux éléments suivants sont les mêmes:

```
<fsa:produit
xmlns:fsa=http://www.fsa.univ-ibn-zohr.ma>
... </fsa:produit>
```

```
<inf:produit
xmlns:inf=http://www.fsa.univ-ibn-zohr.ma>
... </inf:produit>
```

- Les espaces de noms doivent être utilisés si le document XML rédigé est destiné à être mélangé à d'autres sources.

- On peut fixer l'espace de noms par défaut avec la syntaxe:

```
<produit xmlns=http://www.fsa.univ-ibn-zohr.ma>
  <nom>...</nom> <desc>...</desc> </produit>
</produit>
```

cela évite d'utiliser le préfixe.

29

Un Document XML bien formé

Un document XML avec une **syntaxe correcte** est dit **bien formé** =>

- Le document XML doit avoir un seul élément racine
- Les éléments (balises) XML doivent avoir une balise fermante
- Les balises XML sont sensibles à la casse (case-sensitive)
- Les valeurs des attributs doivent toujours être entre guillemets
- Les balises XML ne doivent pas se chevaucher

30

Un Document XML bien formé

- Conforme aux **règles syntaxiques** du langage XML !

contre exemple :

balises
chevauches

```
<?xml version = "1.0" standalone="yes"?>
<!-- Document XML pas bien formé ! -->
<peintre>
  <nom> Picasso
  <prenom>
    </nom> Pablo
  </prenom>
</peintre>
```

- Alors
 - Association possible avec une feuille de style
 - Peut être exploité par un parseur/analyseur syntaxique (i.e. pour parcourir l'arbre XML et le transformer)
 - Candidat pour être valide

31

Document XML valide

- Associé à une définition **DTD** (.dtd) ou un **Schema** (.xsd)

- définition:
 - interne au document XML -> **non recommandé** (dans le commentaire DOCTYPE)
 - externe -> **réutilisation des définitions, échange** (référéncé vers un fichier dans le DOCTYPE)

- **Conditions**

- document bien formé (syntaxe correcte),
- structure du document respectant la définition (voir les DTD),
- les références aux éléments du document soit résolues

- Alors

- Le document XML peut être échangé ! (format standardisé)

32

Plan

■ Partie I : Le standard XML

- Objectifs
- Pourquoi XML ?
- Structure d'un document XML
- Document bien formé

■ Partie II : Definition des documents XML

- DTD
- XML Schema

■ Partie III : Mise en forme, Traitement et Transformations des documents XML

- DOM (Document Object Model)
- XPath (Chemins d'accès au arbre XML)
- Transformations XSLT

33

Document bien formé et valide

■ Document bien formé

- Respecte les règles d'écriture syntaxique
- pas nécessairement conforme à une DTD ou XML schema

■ Document valide

- bien formé + conforme à une DTD (ou un schéma)

34

DTD

- Permet de définir le «vocabulaire» et la structure qui seront utilisés dans le document XML
- Grammaire du langage dont les phrases sont des documents XML (instances)
- Peut être mise dans un fichier et être référencé dans le document XML

35

Élément et attribut

- **<!ELEMENT *balise (contenu)*>**
 - Décrit une *balise qui fera partie du vocabulaire*.
 - Syntaxe:**
 - **<!ELEMENT tag (content) >** **Ou bien**
 - **<!ELEMENT element-name category>** (i.e. EMPTY, ANY, #PCDATA)
 - ✓ Exp. : **<!ELEMENT book (author, editor)>**
- **<!ATTLIST balise [attribut type #mode [valeur]]***
 - Définit la liste d'attributs pour une balise
 - ex : **<!ATTLIST auteur**
genre CDATA #REQUIRED
ville CDATA #IMPLIED>
<!ATTLIST editeur
ville CDATA #FIXED "Paris">

36

Elément et attribut

■ Nature des attributs : optionnels, obligatoires, valeur déterminée

- optionnel sans valeur par défaut
`<!ATTLIST personne att1 CDATA #IMPLIED>`
- optionnel avec valeur par défaut
`<!ATTLIST personne att1 "bidule">`
- obligatoire
`<!ATTLIST personne att1 CDATA #REQUIRED>`
- fixe
`<!ATTLIST personne att1 CDATA #FIXED "bidule">`

■ **Exemple :**

```
<!ATTLIST personne id ID #REQUIRED>
<!ATTLIST personne att1 CDATA #IMPLIED att2 CDATA #IMPLIED>
```

37

Structuration des balises

■ Structuration du contenu d'une balise

- (a, b) séquence **ex** (*nom, prenom, rue, ville*)
- (a|b) liste de choix **ex** (*oui/non*)
- a? élément optionnel [0,1] **ex** (*nom, prenom?, rue, ville*)
- a* élément répétitif [0,N] **ex** (*produit*, client*)
- a+ élément répétitif [1,N] **ex** (*produit*, vendeur+*)

38

Types de données

- **CDATA** : Données brutes qui ne seront pas analysées par le parseur (*Unparsed*) *Character DATA*
- **PCDATA** : Données de type texte dans l'encodage courant *Parsable Character DATA*
- **Enumération** : Liste de valeurs séparées par « | » (oui | non | peut-etre)
- **ID** : Identifiant pour l'élément (doit être unique dans le document)
- **IDREF, IDREFS** : Référence à un ID de ce document (resp. plusieurs références séparées par des espaces)
- **ENTITY, ENTITIES** : La valeur de l'attribut doit être le nom d'une entité déclarée dans la DTD (resp. un ensemble d'entités séparées par des espaces)
- **ANY** : Tout texte possible
- **EMPTY** : Vide

39

Exemple de DTD Externe (*fichier .dtd*)

```

<!ELEMENT doc (livre* | article+)>
<!ELEMENT livre (titre, auteur+)>
<!ELEMENT article (titre, auteur*)>
<!ELEMENT titre(#PCDATA)>
<!ELEMENT auteur(nom, adresse)>
    <!ATTLIST auteur id ID #REQUIRED>
<!ELEMENT nom(prenom?, nomfamille)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nomfamille (#PCDATA)>
<!ELEMENT adresse ANY>

```

40

Exemple de DTD interne

```
<?XML version="1.0" standalone="yes"?>
<!DOCTYPE CATALOGUE [
  <!ELEMENT CATALOGUE (VOITURES +)>

  <!ELEMENT VOITURES (SPECIFICATION+, ANNEE, PRIX)>
  <!ATTLIST VOITURES NOM CDATA #REQUIRED>

  <!ELEMENT SPECIFICATION EMPTY>
  <!ATTLIST SPECIFICATION MARQUE CDATA #REQUIRED
    COULEUR CDATA #REQUIRED>

  <!ELEMENT ANNEE (#PCDATA)>
  <!ELEMENT PRIX (#PCDATA)>
]>
<CATALOGUE>
  <VOITURES NOM= " LAGUNA">
    <SPECIFICATION MARQUE= " RENAULT" COULEUR="Rouge"/>
    <ANNEE>2001</ANNEE>
    <PRIX>70 000 Dirhams</PRIX>
  </VOITURES>
  .....
</CATALOGUE>
```

41

Exemple de ID et IDREF

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT DOCUMENT(PERSONNE*)>
  <!ELEMENT PERSONNE (#PCDATA)>
  <!ATTLIST PERSONNE PNUM ID #REQUIRED>
  <!ATTLIST PERSONNE MERE IDREF #IMPLIED>
  <!ATTLIST PERSONNE PERE IDREF #IMPLIED>
]>
<DOCUMENT>
  <PERSONNE PNUM = "P1">Fatima</PERSONNE>
  <PERSONNE PNUM = "P2">Mohamed</PERSONNE>
  <PERSONNE PNUM = "P3" MERE="P1" PERE="P2">Ali</PERSONNE>
  <PERSONNE PNUM = "P4" MERE="P1">Hajar</PERSONNE>
</DOCUMENT>
```

42

Pourquoi des DTD externes ?

■ Modèle pour plusieurs documents

- partage des balises et structures

■ Définition locale ou externe

- `<!DOCTYPE doc SYSTEM "doc.dtd">`
- `<!DOCTYPE doc PUBLIC "www.e-xmlmedia.com/doc.dtd">`

■ Exemple de document

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE VOITURES SYSTEM "voitures.dtd">
...
```

43

Entités dans les DTD

■ Entité

- Permet la réutilisation dans une DTD

Syntaxe

- Déclaration interne:
`<!ENTITY entity-name entity-value>`
- Déclaration externe:
`<!ENTITY entity-name SYSTEM "entity-URL">`
- Pour la référencer → `&entity-name`
- Exemple (interne):
`<!ENTITY website "http://www.TheScarms.com">`
- Exemple (externe):
`<!ENTITY website SYSTEM "http://www.TheScarms.com/entity.xml">`

■ Dans un document XML:

```
<url>&website</url>
```

Sera évaluée à:

```
<url>http://www.TheScarms.com</url>
```

44

Synthèse DTD

■ Spécification de la structure du document

- déclaration de balisage : ELEMENT, ATTLIST, ENTITY;
- déclaration des éléments

✓ éléments simples : $\left\{ \begin{array}{l} - \textbf{Vide (EMPTY)} \\ - \textbf{Libre (ANY)} \\ - \textbf{Textuel (\#PCDATA)} \end{array} \right.$

✓ composition : $\left\{ \begin{array}{ll} - \textbf{Séquence d'éléments} \text{ liste ordonnée} & \rightarrow (a,b,c) \\ - \textbf{Choix alternatifs d'éléments} & \rightarrow (a|b|c) \\ - \textbf{Mixte hiérarchique} & \rightarrow (a,(b|c),d) \end{array} \right.$

✓ indicateurs d'occurrences : $\left\{ \begin{array}{l} - ? \text{ (zéro ou une),} \\ - * \text{ (zéro ou plusieurs)} \\ - + \text{ (une ou plusieurs)} \end{array} \right.$

45