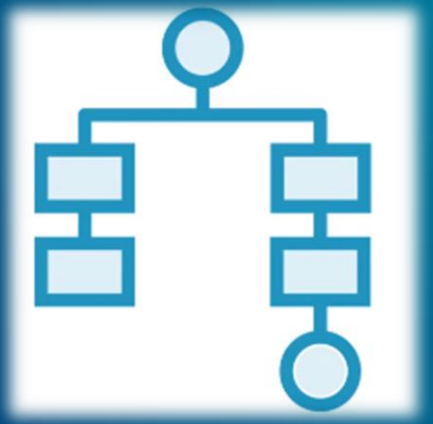
Université de BATNA 2
Département d'informatique

Cours : Les donnés semi-structurées

L3 ISIL



Chapitre 02 PART 1 : XML *core langage :* syntaxe de base.

Djennane, A
Djennane.am@gmail.com

2023 -2024

Contenu

01
Contexte et problématique

02
XML *Core* : syntaxe de base, espaces de noms.

03
La validation d'un document XML (DTD, XML Schema)

04
Xml et les base de données

05
La programmation XML : SAX, DOM, …etc

Contenu

01
Contexte et problématique

02
XML *Core* : syntaxe de base, espaces de noms.

03
La validation d'un document XML (DTD, XML Schema)

04
Xml et les base de données

05
La programmation XML : SAX, DOM, …etc

Contenu

01
Contexte et problématique

02
XML *Core* : syntaxe de base, espaces de noms.

03
La validation d'un document XML (DTD, XML Schema)

04
Xml et les base de données

05
La programmation XML : SAX, DOM, …etc

Contenu

01
Contexte et problématique

02
XML *Core* : syntaxe de base, espaces de noms.

03
La validation d'un document XML (DTD, XML Schema)

04
Xml et les base de données

05
La programmation XML : SAX, DOM, ...etc

Contenu

01
Contexte et problématique

02
XML *Core* : syntaxe de base, espaces de noms.

03
La validation d'un document XML (DTD, XML Schema)

04
Xml et les base de données

05
La programmation XML : SAX, DOM, ...etc

Assessment

-CC (40%) = Test+ Project (3-5 students) + quiz\ test.

- Final exam (60%).

Objectives de la matière

Cette matière va permettre aux étudiants :

- Faire la différence entre les données **structurées/semi-structurées** et **non structurées** (introduction pour le BIG DATA …).
- Avoir des connaissances liées aux formats d'échanges et de transportation de données entre les systèmes d'informations (XML, JSON, CSV).
- Savoir utilisez les technologies XML (XPath, XQuery….).

CHAPTER 2

Contenu de chapitre

On va voir dans ce chapitre :

- Syntaxe XML (*core XML syntax*),

Structure d'un document XML

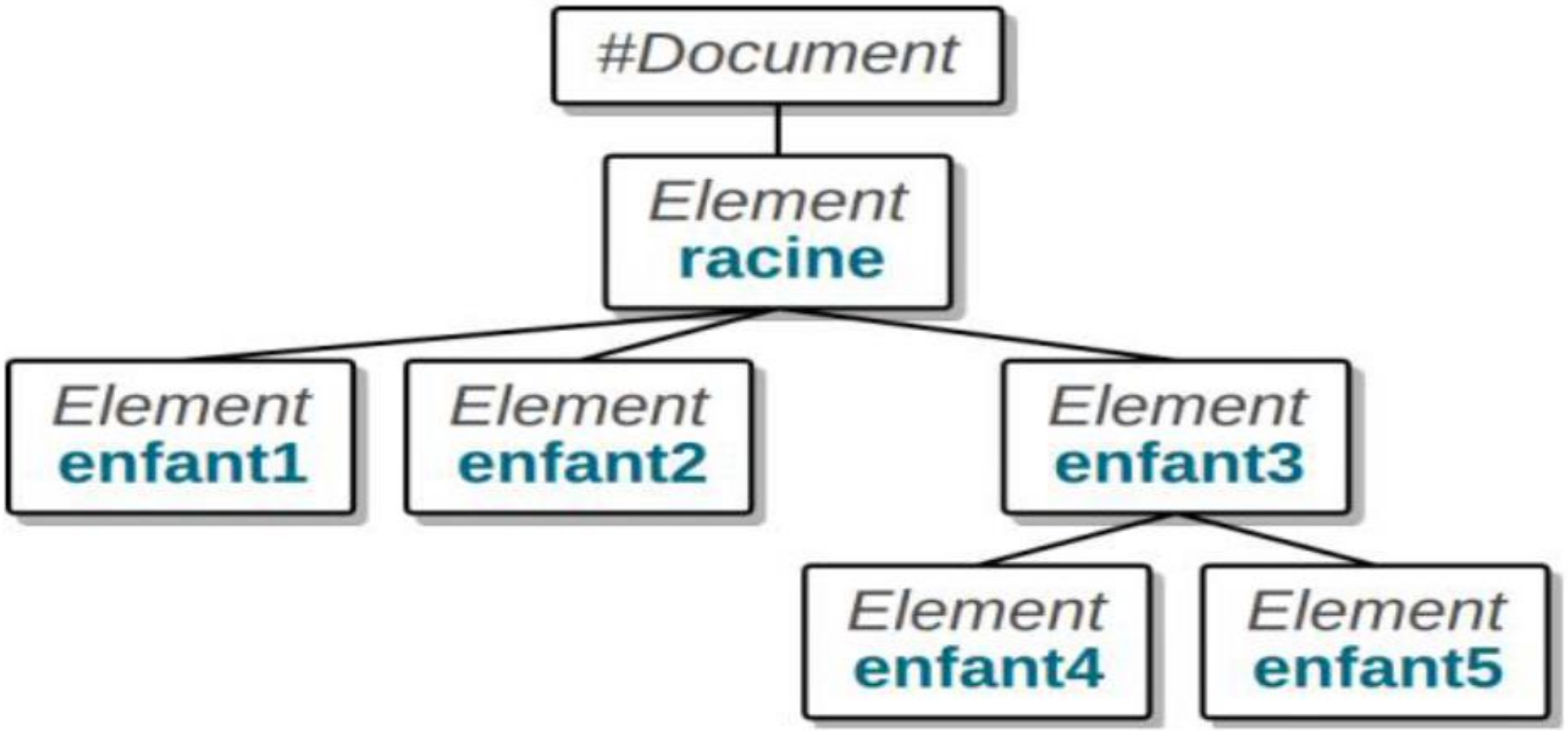
Arborescence d'éléments

Un document XML est composé de plusieurs parties :

- Entête de document précisant la version et l'encodage,
- Des règles optionnelles permettant de vérifier si le document est valide
- Un arbre d'*éléments* basé sur un élément appelé *racine*
  - Un *élément* possède un *nom*, des *attributs* et un *contenu*
  - Le contenu d'un élément peut être :
    - rien : élément vide noté `<nom/>` ou `<nom attributs.../>`
    - du texte
    - d'autres éléments (les éléments enfants).
  - Un élément non vide est délimité par une *balise ouvrante* et une *balise fermante*.
    - une balise ouvrante est notée `<nom attributs...>`
    - une balise fermante est notée `</nom>`

L’arbre XML

Soit cet arbre XML :



L'arbre XML

Voici comment on désigne les différents nœuds les uns par rapport aux autres :

- <racine> est le nœud **parent** du nœud **enfant** (*child*) <enfant3>, lui-même parent de <enfant4> et <enfant5>,
- <racine>, <enfant3> sont des nœuds **ancêtres** (*ancestors*) de <enfant4> et <enfant5>,
- <enfant4> et <enfant5> sont des **descendants** (*descendants*) de <racine> et <enfant3>,
- <enfant1> est un nœud **frère** (*sibling*[1]) de <enfant2> et réciproquement.

- On peut manipuler cette arbre (créer, rechercher, supprimer, …) par programmation (JAVA, PHP, …etc).

Prologue XML

La première ligne d'un document XML est appelée *prologue XML*. Elle spécifie la version de la norme XML utilisée (1.0 ou 1.1 qui sont très similaires[2]) ainsi que l'encodage :

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

```
<?xml version="1.0" encoding="iso-8859-15"?>
```

Norme Unicode

La norme Unicode définit un ensemble abstrait de plus de 245000 caractères représentable sur un ordinateur, par exemple é, €, «, ©. . . Ces caractères doivent être codés sous forme d'octets. C'est là qu'il y a plusieurs normes :

- ASCII ne représente que les 128 premiers caractères Unicode.
- ISO 8859-1 ou Latin-1 représente 191 caractères de l'alphabet latin n°1 (ouest de l'europe) à l'aide d'un seul octet. Les caractères € et œ n'en font pas partie, pourtant il y a æ.
- ISO 8859-15 est une norme mieux adaptée au français. Elle rajoute € et œ et supprime des caractères peu utiles comme ¤.
- UTF-8 représente les caractères à l'aide d'un nombre variable d'octets, de 1 à 4 selon le caractère. Par exemple : A est codé par 0x41, é par 0xC3,0xA9 et € par 0xE2,0x82,0xAC.

Commentaires

On peut placer des commentaires à peu près partout dans un document XML. La syntaxe est identique à celle d'un fichier HTML. Un commentaire peut d'étendre sur plusieurs lignes. La seule contrainte est de ne pas pouvoir employer les caractères -- dans le commentaire, même s'ils ne sont pas suivis de >

```
<!-- voici un commentaire -->
ceci n'est pas un commentaire
<!--
    voici un autre commentaire
    et ça -- , c'est une erreur
-->
```

Options après le prologue

Après le prologue, on peut trouver plusieurs parties optionnelles délimitées par <?...?> ou <!...>. Ce sont des instructions de traitement, des directives pour l'analyseur XML.

Par exemple :

- un *Document Type Definitions* (DTD) qui permet de valider le contenu du document.
- une feuille de style

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE personne SYSTEM "personne.dtd">
<?xml-stylesheet href="personne.css" type="text/css"?>
<personne>
    ...
</personne>
```

Nom des éléments

Les noms des éléments peuvent employer de nombreux caractères Unicode (correspondant au codage déclaré dans le prologue) mais pas les signes de ponctuation.

- L'initiale doit être une lettre ou le signe _
- ensuite, on peut trouver des lettres, des chiffres ou - . _

Attributs

Les attributs caractérisent un élément. Ce sont des couples nom="valeur" ou nom='valeur'. Ils sont placés dans la balise ouvrante.

```
<?xml version="1.0" encoding="utf-8"?>
<meuble id="765" type='table'>
    <prix monnaie='€'>74,99</prix>
    <dimensions unites="cm" longueur="120" largeur="80"/>
    <description langue='fr'>Belle petite table</description>
</meuble>
```

Remarques :

- Il n'y a pas d'ordre entre les attributs d'un élément,
- Un attribut ne peut être présent qu'une fois.

Attributs ou éléments

En ce qui concerne les propriétés de l'entité, on peut les placer :

- dans les attributs des éléments :

```
<voiture id="125" marque="Renault" couleur="grise"/>
```

- en tant que sous-éléments :

```
<voiture>
  <id>125</id>
  <marque>Renault</marque>
  <couleur>grise</couleur>
</voiture>
```

- ou employer un mélange des deux

Entités

Certains caractères sont interdits dans le contenu des éléments. Comme il est interdit de les employer, XML propose une représentation appelée *référence d'entité* ou *entité* pour simplifier. On retrouve le même concept en HTML.

Caractère	Entité
<	<
>	>
&	&
'	'
"	"

Exemple

Voici un exemple d'emploi d'entités :

```
<?xml version="1.0" encoding="utf-8"?>
<achat-si type-groupement="&amp;&amp;">
  <test>prix &lt; 50.00</test>
  <test>description &lt;&gt; <str value="&quot;&quot;"/></test>
</achat-si>
```

- Cela entraîne une erreur si on remplace ces entités par le caractère correspondant,
- Ces entités sont automatiquement remplacées par les caractères lorsqu'on traite le fichier.

Exemple d'entité interne

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE personne [
    <!ENTITY adresse "IUT de Lannion - Dept Informatique">
    <!ENTITY cours   "Cours XML">
]>
<personne>
    <lieu>&adresse;</lieu>
    <role>&cours;</role>
</personne>
```

Texte

Les textes font partie du contenu des éléments et sont vus comme des nœuds enfants. Il faut bien comprendre que la totalité du texte situé entre les balises, y compris les espaces et retour à la ligne font partie du texte.

```
<?xml version="1.0" encoding="utf-8"?>
<racine>
  texte1
  <enfant>texte2</enfant>
  texte3
</racine>
```

Section CDATA

Lorsqu'on veut écrire du texte brut à ne pas analyser en tant qu'XML, on emploie une section CDATA :

```
<?xml version="1.0" encoding="utf-8"?>
<fichier>
    <nom>exemple1.xml</nom>
    <md5><![CDATA[19573b741c0c5c8190a83430967bfa58]]></md5>
</fichier>
```

La partie entre <![CDATA[et]]> est ignorée par les analyseurs XML, on peut mettre n'importe quoi sauf]]>. Ces données sont considérées comme du texte par les analyseur.

Exemple


```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <svg-examples>
3      <example>
4          The following Scalable Vector Graphics document
5          describes a blue-filled and black-stroked
6          rectangle.
7          <![CDATA[<svg width="100%" height="100%"
8  version="1.1"
9  xmlns="http://www.w3.org/2000/svg">
10 <rect width="300" height="100"
11 style="fill:rgb(0,0,255);stroke-width:1;
12 stroke:rgb(0,0,0)"/>
13 </svg>]]>
14     </example>
15 </svg-examples>
16
```

- Le parseur va afficher le Code SVG, comme un texte.

Exercices

Bien formé

```
<list>
<item>Voiture</item>
<ITEM>Avion</ITEM>
<Item>Train</Item>
</list>
```

Mal formé

```
<list>
<item>Voiture</itm>
<item>Avion</ITEM>
<item>Train</item>
</list>
```

Mal formé

```
<text>
        <bold><italic>XML</bold></italic>
</text>
```

Exercices

Mal formé

```
<forbiddenNames>
        <A;name/>
        <last@name>
        <@#$%^()%+?=/>
        <A*2/>
        <1ex/>
</forbiddenNames>
```

- Les noms ne peuvent pas commencer par xml.

Mal formé

```
<forbiddenNames>
        <xmlTag/>
        <XMLTag/>
        <XmLTag/>
        <xMlTag/>
        <xmLTag/>
</forbiddenNames>
```

Exercices

Bien formé

Sélectionnez

```
<elements-with-attributes>
        <el _ok = "oui" />
        <one attr = "une valeur"/>
        <several first="1" second = '2' third= "333"/>
        <apos_quote case1="Aujourd'hui" case2='Il lança : "Salut, tout le
monde!"  '/>
</elements-with-attributes>
```

-

Mal formé

Sélectionnez

```
<errors>
        <wrong_char a*b = "23432"/>
        <mismatched_separator value = "12'/>
        <wrong_separator_type  value="aa"aa"/>
        <wrong_separator_type  value='bb'bb'/>
        <wrong_start XML-ID = "xml234"/>
</errors>
```

Exercices

Bien formé — Sélectionnez

```
<example>
        <isLower>
                23 &lt; 46
        </isLower>
        <ampersand>
                Dupond &amp; fils
        </ampersand>
</example>
```

-

Mal formé — Sélectionnez

```
<example>
        <isLower>
                23 < 46
        </isLower>
        <ampersand>
                Dupond & fils
        </ampersand>
</example>
```

Exercices

Bien formé

```
<example>
        <![CDATA[ <aaa>bb&cc<<<]]>
</example>
```

-

Mal formé

```
<example>
        <![CDATA[ <aaa>bb ]]>cc<<<]]>
</example>
```

Thank you