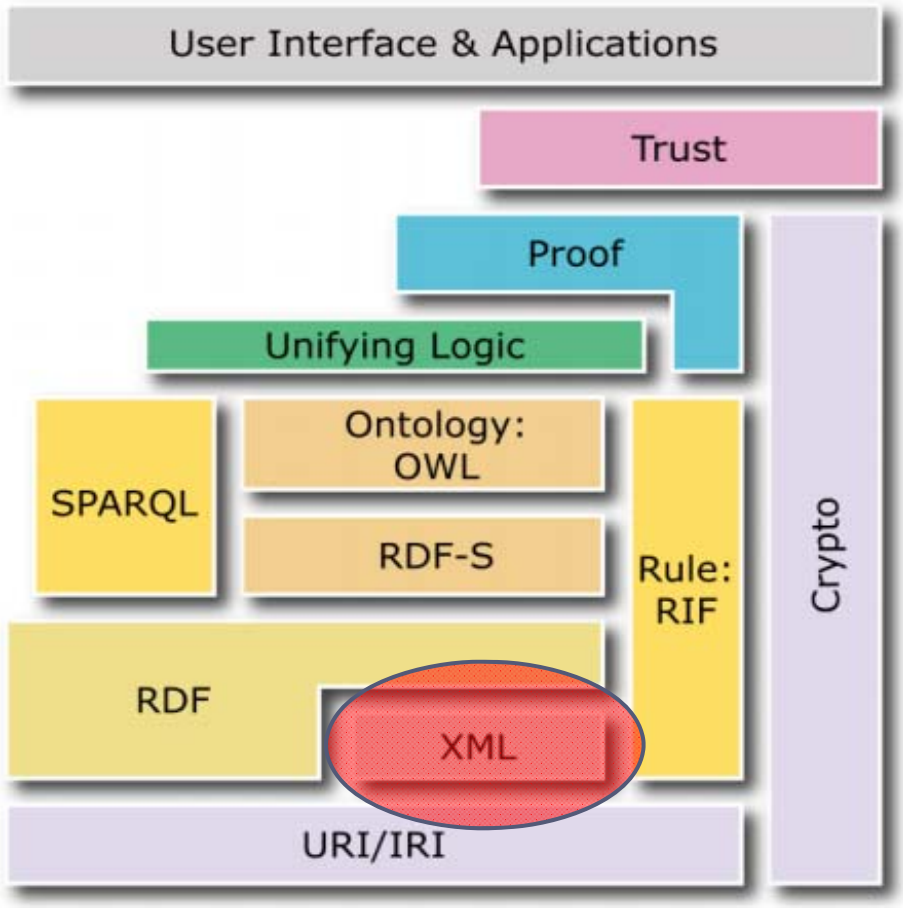
Web Sémantique & XML

Master IPS 2021/2022

Prof Ali OUACHA

Web sémantique

Approche en couches



Plan

Chapitre 1
XML

I – Introduction

II – Structure et Syntaxe

III – Espaces de nom

IV – Modèles de données : DTD

V – Modèles de données : Schéma XML

Introduction

I - Introduction

- XML (E**X**tensible **M**arkup **L**anguage) est un langage de balisage.
- Il est utilisé par des applications différentes afin de :
  - stocker
  - organiser
  - partager
- Le succès de XML est en grande partie dû à :
  - Séparation stricte entre contenu et présentation.
  - Structuration forte.
  - Simplicité, universalité et extensibilité.
  - Modèles de documents (DTD et Schémas XML)
  - Format texte avec gestion des caractères spéciaux.
  - Format libre.

I – Introduction

- Pour qu'un document XML soit correcte, il doit respecter les deux contraintes suivante :
  - **il doit d'abord être *bien formé* :**
    La première contrainte est de nature ***syntaxique***. Un document bien formé doit respecter certaines règles syntaxiques propres à XML.
  - **il doit ensuite, être *valide*:**
    La seconde contrainte est de nature ***structurelle***. Un document valide doit respecter un *modèle de document*.
- **Par Exemple :**
  Pour qu'une phrase en français soit correcte???
  - Orthographe
  - Grammaire

La différence essentielle :
la grammaire d'XML n'est pas figée.

Structure et Syntaxe

II - Structure et Syntaxe

Structure Globale d'un document XML

- La composition globale d'un document XML comprend deux parties consécutives :
  - **Prologue :** Il contient des déclarations facultatives.
  - **Corps du document :** **c**'est le contenu même du document.

```
<?xml ... ?>
...
```

Prologue

```
<root-element>
...
</root-element>
```

Corps

  - Un document XML peut aussi contenir :
    - **Commentaires.**
    - **Instructions de traitement.**

II - Structure et Syntaxe

Commentaires et Instructions de traitement

- **Commentaires**
  - En XML, les commentaires commencent par **<!--** et se terminent par **-->**.

```
<!-- ceci est correct -->
<elt>
<!-- ceci est correct aussi -->
Un peu de texte
</elt>
```

- **Instructions de traitement**
  - Une instruction de traitement est une instruction interprétée par l'application servant à traiter le document XML.

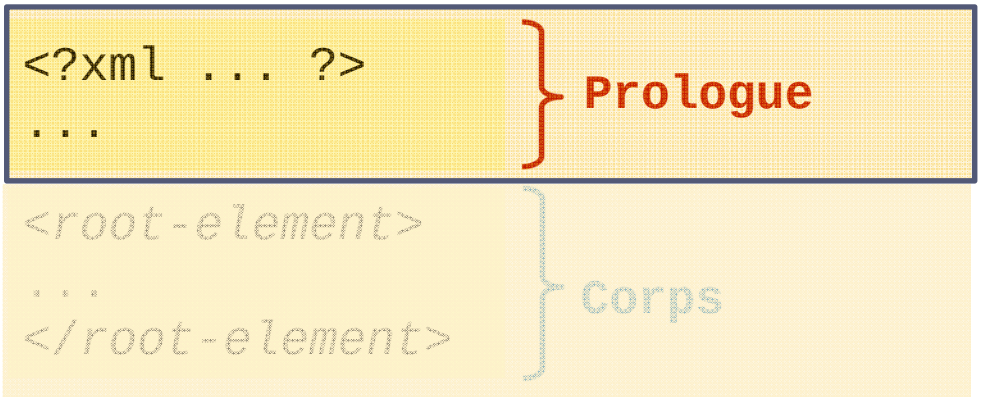
Les commentaires et les instructions de traitement peuvent être placés à n'importe quel endroit après le prologue tant qu'ils se trouvent à l'*extérieur* d'une autre balise.

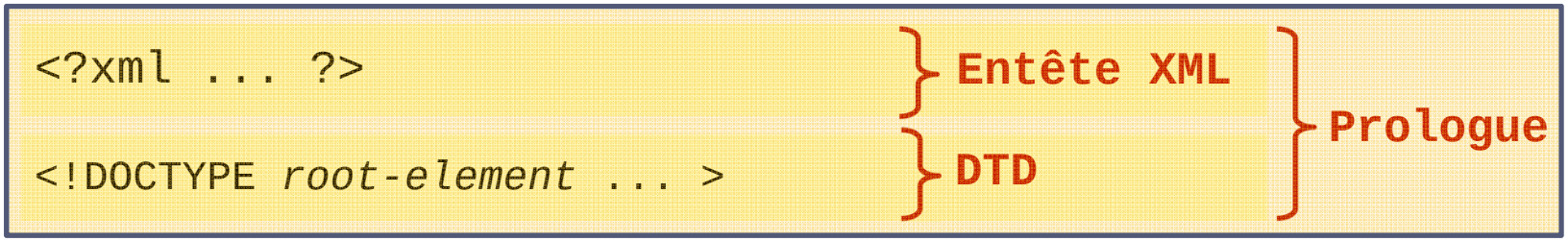
II - Structure et Syntaxe

Prologue

Prologue

- Le prologue contient deux déclarations facultatives :
  - L'entête XML.
  - La déclaration du type du document (DTD).





II - Structure et Syntaxe

Prologue - Entête XML

Entête XML

- Il est optionnel.
- Mais s'il est présent, il doit être la première ligne d'un document XML.
- Il sert à donner les caractéristiques globales du document.

- il a la forme générale suivante :

```
<?xml version="..." encoding="..." standalone="..."?>
```

- Exemples d'entête :

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

II - Structure et Syntaxe

Prologue - Entête XML

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

- **version** :
  précise la version d'XML utilisée.
  Les valeurs possibles actuellement sont **1.0** ou **1.1**
- **encoding** :
  précise le **codage des caractères** utilisé dans le fichier.
  Quelques valeurs possibles sont : **ASCII**, **ISO-8859-1**, **UTF-8**, …
  Ces noms de codage peuvent aussi être écrits en minuscule.
  Le codage par défaut est : **UTF-8**
- **standalone** :
  précise si le fichier est autonome, c'est-à-dire s'il existe des déclarations externes qui affectent le document.
  la valeur de cet attribut peut être **yes** ou **no**
  sa valeur par défaut est **no**.

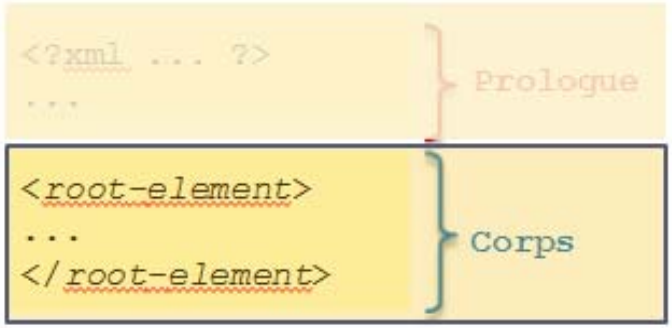
Déclaration de type de document (DTD)

- La déclaration de DTD définit la structure du document.
  - Elle précise le contenu de chacun des éléments
- Elle peut prendre plusieurs formes suivant que la définition du type est **interne** ou **externe**.
- Elle utilise le mot clé **DOCTYPE**
- Sa forme générale est comme suit : **<! DOCTYPE *root-element* ... >**

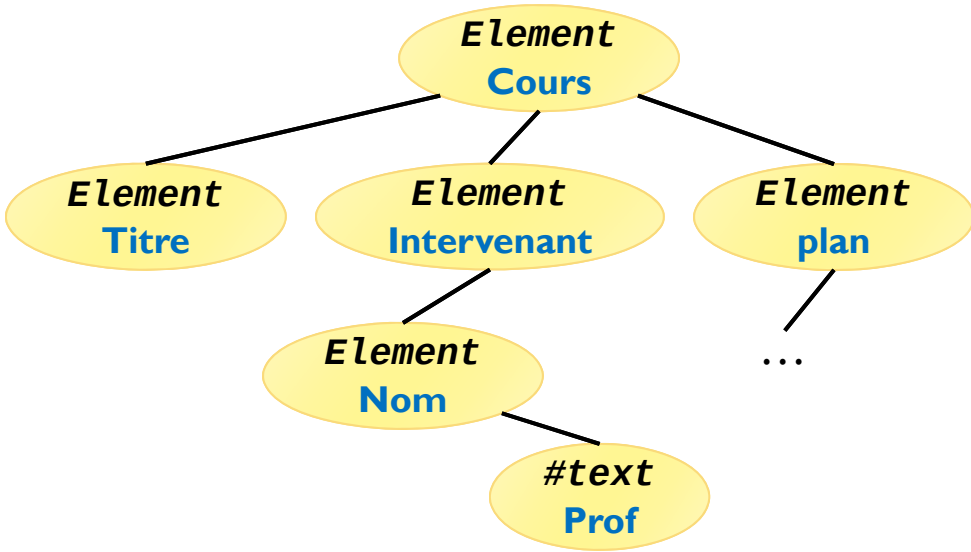
```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE cours SYSTEM … >
<cours>
    …
</cours>
```

Corps

- Un document XML se représente sous la forme d'une **arborescence d'éléments**
- Cette **arborescence** comporte un élément appelé **élément racine**
- Cette **racine** contient l'intégralité du document
- La racine doit être **unique**




```
<?xml version="1.0" ?>
<cours>
    <titre>XML</titre>
    <intervenant>
        <nom > Prof </nom>
    </intervenant>
    <plan>  ...  </plan>
</cours>
```



II - Structure et Syntaxe

Corps - Éléments

- Un *élément* est formé : d'une **balise ouvrante**, d'un **contenu** et de **la balise fermante** correspondante.
  - La ***balise ouvrante*** qui prend la forme `<name>`
  - La ***balise fermante*** qui prend la forme `</name>`
  - Le ***contenu*** qui peut être :
    - du texte simple,
    - d'autres éléments
    - un mélange des deux.

- les autres éléments contenues d'un élément sont alors appelés « **éléments enfants** »
- l'élément contenant est appelé « **élément parent** »

II - Structure et Syntaxe

Corps du document - **Éléments**

- **Exemples** :
  - Élément contenant du texte :

```
<titre>Les Misérables</titre>
```

  - Élément contenant d'autres éléments :

```
<livre>
      <titre>L'Assommoir</titre>
      <auteur>Émile Zola</auteur>
      <couverture couleur="rouge" />
</livre>
```

  - Élément vides : il ne contient pas d'élément-enfant :

```
<couverture … />
```

II - Structure et Syntaxe

Corps du document - **Éléments**

- **Règle à respecter :**
  - Le nom d'un élément est un **nom XML**
  - Un ***nom XML*** peut commencer par une lettre `[a-zA-Z]`, le caractère `':'` ou le caractère `'_'`
  - Les deux caractères `'-'` et `'.'` ainsi que les `chiffres` ne peuvent pas apparaître au début des noms
  - Respecter la **casse** : XML **fait la différence** entre **majuscule** et **minuscule**.
  - Respecter l'**ordre d'imbrication** des balise
  - Toutes les balises portant un contenu non vide **doivent être fermées**.
  - Les balises n'ayant pas de contenu **peuvent se terminer par** `'/>'`
  - **Aucun espace** ne peut séparer le caractère `'<'` du nom de l'élément

II - Structure et Syntaxe

Corps du document - **Attributs**

- Les balises ouvrantes d'un élément peuvent contenir des ***attributs*** associés à des **valeurs**.
- L'association de la valeur à l'attribut prend l'une des formes suivantes:
  - ***attribute***='***value***' ou ***attribute***="***value***"
- **attribute** et **value** sont respectivement le **nom** et la **valeur** de l'attribut.
- Chaque balise ouvrante peut contenir **zéro**, **une** ou **plusieurs** associations ***attributs-valeurs***.
- Un élément dont le contenu est vide peut aussi contenir des attributs.

```
<tag1> ... </tag1>
<tag2 attribute="value"> ... </tag2>
<tag3 attribute1="value1" attribute2="value2"> ... </tag3>
<tag4 attribute="value"/>
```

II - Structure et Syntaxe

Corps du document - **Attributs**

- **Règle à respecter :**
  - Le nom de chaque attribut doit être un **nom XML**.
  - La valeur d'un attribut peut être une chaîne quelconque de caractères délimitée par des apostrophes «**'**» ou des guillemets «**"**».
  - Elle peut contenir les caractères spéciaux «**<**», «**>**», «**&**», «**'**» et «**"**» qui **doivent nécessairement** être représentés par les **entités prédéfinies.**
  - Si la valeur de l'attribut est délimitée par des apostrophes « **'**», les guillemets «**"**» peuvent être introduits directement sans entité et inversement

```
<xsl:value-of select="key('idchapter', @idref)/title"/>
<xsl:if test='@quote = "&apos;"'>
```

II - Structure et Syntaxe

Corps du document - **Attributs**

- **Règle à respecter :**
  - l'indentation est libre pour écrire les attributs d'une balise ouvrante

```
<tag    attribute1="value1"
        attribute2="value2"
         ...
        attributeN="valueN">
    ...
</tag>
```

  - L'ordre des attributs n'a pas d'importance.
  - Les attributs d'un élément doivent avoir des **noms distincts**.
    - Il est donc impossible d'avoir deux occurrences du même attribut dans une même balise ouvrante.

II - Structure et Syntaxe

Corps du document - **Validation**

- Respecte **les règles pour les éléments et pour les attributs**

Un document XML ***bien formé***.

II - Structure et Syntaxe

Corps du document - **Validation**

- **Quelques conventions de nommage**

Voici quelques conventions souvent employées dans les documents XML :

- Employer des minuscules pour les attributs et les éléments.
- Éviter les accents dans les noms d'attributs et d'éléments.
- Préférer les guillemets délimitant les valeurs d'attribut.
- Séparer les noms composés de plusieurs mots par les caractères **-**, **_**, **.** ou une majuscule.
- Essayer d'être homogène dans votre document en gardant la même convention.

II - Structure et Syntaxe

Corps du document – **Entités & CDATA**

- **Entités & CDATA :**
  - Quel est le problème avec cette syntaxe :

```
<comparaison>
   Si x<0 alors la valeur absolu de de x est -x
</comparaison>
```

  - Les caractères spéciaux «**<**», «**>**» et «**&**» ne peuvent pas être inclus directement dans le contenu d'un document.
  - Ils peuvent être inclus par l'intermédiaire :
    - **des entités prédéfinies**
    - **des *sections littérales* (*sections CDATA*)**

II - Structure et Syntaxe

Corps du document – **Entités & CDATA**

- **Entités prédéfinies:**
  - Les entités prédéfinies permettant d'inclure les caractères spéciaux «**<**», «**>**», «**&**», «**'**» et «**"**» dans les contenus d'éléments et dans les valeurs d'attributs.

Entité	Caractère
<	<
>	>
&	&
'	'
"	"
…	…

```
<comparaison>
    Si x&lt;0 alors la valeur absolu de de x est -x
</comparaison>
```

II - Structure et Syntaxe

Corps du document - **Entités & CDATA**

- **Sections littérales (CDATA):**
  - Il est souvent fastidieux d'inclure beaucoup de caractères spéciaux à l'aide des entités.
  - Les *sections littérales* permettent d'inclure des caractères qui sont recopiés à l'identique.
  - Une section littérale commence par la chaîne de caractères «**<![CDATA[**»et se termine par la chaîne «**]]>**»
  - Une section littérale ne peut pas contenir la chaîne de caractères « **]]>**»
  - Il est impossible d'imbriquer des sections CDATA.

```
<![CDATA[ Contenu avec des caractères spéciaux <, > et & ]]>
```

Question 1

On souhaite écrire un livre en utilisant le formalisme XML. Ce livre est structuré en sections, en chapitres et en paragraphes. Le livre doit contenir la liste des auteurs (avec nom et prénom). Tous les éléments doivent posséder un titre, sauf le paragraphe qui contient du texte.

1. Proposez une structuration XML de ce document (avec 2 auteurs, 2 sections, 2 chapitres par section et 2 paragraphes par chapitre).
2. Vérifiez, à l'aide de l'éditeur, que votre document est bien formé.

Attention :

- Ne pas utiliser d'attributs.
- L'encodage utilisé est UTF-8.
- Votre document sera nommé livre1.xml.

Espaces de nom

III – Espaces de nom

Introduction

Introduction

- Dans le cas où plusieurs vocabulaires sont mélanger au sein d'un même document, il devient indispensable d'identifier la provenance de chaque élément et de chaque attribut afin de le valider correctement.
- Les espaces de nom :
  - sont introduits en XML afin de pouvoir mélanger plusieurs vocabulaires au sein d'un même document.
  - permettent d'éviter les conflits de noms entre différents vocabulaires.
  - permettent la réutilisation des vocabulaires déjà définis.
  - chaque élément ou attribut appartient à un espace est marquée par la présence d'un préfixe associé à cet espace de noms.

III – Espace de nom

Identification

- Un ***espace de noms*** est identifié par un IRI appelé ***IRI de l'espace de noms***.
  - Cet IRI garantit seulement que l'espace de noms soit identifié de manière unique.
  - Cet IRI est très souvent une URL.
  - Dans la pratique, l'URL permet, le plus souvent, d'accéder à un document qui décrit l'espace de noms.

- Quelques espaces de noms classiques

XML	**http://www.w3.org/XML/1998/namespace**
MathML	**http://www.w3.org/1998/Math/MathML**
XHTML	**http://www.w3.org/1999/xhtml**
Schémas	**http://www.w3.org/2001/XMLSchema**

III – Espace de nom

Déclaration

- Un espace de noms est déclaré par la forme suivante :

`xmlns:prefix="IRI-espaceDeNom"`

  - **`prefix:`** est un **nom XML** ne contenant pas le caractère « **:** »
  - **`IRI-espaceDeNom:`** l'IRI qui identifie l'espace de noms.

- Cette déclaration associe cet IRI au nom **`prefix`**.
- Ce **préfixe** est ensuite utilisé pour ***qualifier*** les noms d'éléments.

```
1 <hns:html xmlns:hns="http://www.w3.org/1999/xhtml">
2     <hns:head>
3         <hns:title>Espaces de noms</hns:title>
4     </hns:head>
5     <hns:body>
6         ...
7     </hns:body>
8 </hns:html>
```

III – Espace de nom

Déclaration

Remarques:

- Le choix du préfixe est complètement arbitraire.
- Même si les préfixes peuvent être librement choisis:
  - Ils doivent exprimer une cohérence entre sa déclaration et son utilisation.
  - Il faut aussi utiliser certains préfixes particuliers pour certains espaces de noms :
    - **`html`** pour XHTML
    - **`xsd`** ou **`xs`** pour schémas XML
    - **`xsl`** pour les feuilles de style XSLT
    - …
- Il est possible de déclarer plusieurs espaces de noms.
- C'est l'IRI associé au préfixe qui détermine l'espace de noms. Le préfixe est juste une abréviation pour l'IRI.
- Deux préfixes associés au même IRI déterminent le même espace de noms.

III – Espace de nom

Portée d'une déclaration

- La portée d'une déclaration d'un espace de noms :
  - est limité à l'élément dans lequel elle est faite.
  - elle comprend les balises de l'élément qui la contient.

```
1  <html:myhtml xmlns:html="http://www.w3.org/1999/myxhtml">
2      <html:head>
3          <html:title>Espaces de noms</html:title>
4      </html:head>
5      <html:body>
6          ...
7          <mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML">
8              <mml:apply>
9                  <mml:eq/>
10                 ...
11             </mml:apply>
12         </mml:math>
13         <!-- L'espace de noms MathML n'est maintenant plus disponible -->
14         ...
15     </html:body>
16 </html:myhtml>
```

III – Espace de nom

Espace de noms par défaut

- L'utilisation d'un ***espace de noms par défaut*** permet **d'alléger l'écriture**.
  - Elle peut être spécifié par un pseudo attribut de nom **`xmlns`** dont la valeur est l'URI de l'espace de noms.
  - Lorsque celui a été spécifié, les **éléments** dont le nom n'est pas qualifié font partie de l'espace de noms par défaut.

```
1  <myhtml xmlns="http://www.w3.org/1999/myxhtml">
2      <!-- L'espace de noms par defaut est celui de MYXHTML -->
3      <head>
4          <title>Espaces de noms</title>
5      </head>
6      <body>
7          ...
8          <mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML">
9              <mml:apply>
10                 <mml:eq/>
11                 ...
12             </mml:apply>
13         </mml:math>
14         <!-- L'espace de noms MathML n'est maintenant plus disponible -->
15         ...
16     </body>
17 </myhtml>
```

III – Espace de nom

Espace de noms par défaut

- Tant que l'espace de noms par défaut n'a pas été spécifié, les éléments dont le nom n'est pas qualifié ne font partie d'aucun espace de noms.
- Il est possible de revenir à l'espace de noms par défaut non spécifié en affectant la chaîne vide `""` à l'attribut `xmlns` .

```
1  <myhtml xmlns="http://www.w3.org/1999/myxhtml">
2      <!-- L'espace de noms par défaut est maitenant l'espace de noms MYXHTML -->
3      <!-- Tous les éléments html, head, title, body, ... appartiennent
4         à l'espace de noms MYXHTML qui est l'espace de noms par défaut. -->
5      <head>
6          <title>Espaces de noms</title>
7      </head>
8      <body>
9          ...
10         <name xmlns="">
11             <!-- L'espace de noms par défaut n'est plus spécifié -->
12             <!-- Les trois éléments name, firstname et surname
13             n'appartiennent à aucun espace de noms. -->
14             <firstname>Gaston</firstname>
15             <surname>Lagaffe</surname>
16         </name>
17         ...
18     </body>
19 </myhtml>
```

III – Espace de nom

Attributs

Attributs

- **les attributs** dont le nom n'est pas qualifié **n'appartiennent à aucun espace de noms.**
- **Ils** ne font jamais partie de l'espace de noms par défaut. Cette règle s'applique que se soit l'espace de noms par défaut est spécifié ou non.

```
1  <book version="5.0"
2      xml:id="course.xml"
3      xmlns="http://docbook.org/ns/docbook"
4      xmlns:xml="http://www.w3.org/XML/1998/namespace">
5
6  </book>
```

- Dans l'exemple ci-dessous,
  - l'élément **book** appartient à l'espace de noms par défaut.
  - l'attribut **id** appartient à l'espace de noms XML
  - l'attribut **version** n'appartient à aucun espace de noms.

III – Espace de nom

Espace de noms XML

- Le préfixe **`xml`** est toujours implicitement lié à l'espace de noms XML identifié par l'**`IRI`** **`http://www.w3.org/XML/1998/namespace`**.
- Cet espace de noms n'a pas besoin d'être déclaré.
- Les quatre attributs particuliers suivants font partie de cet espace de nom:
  - **`xml:id`** Il permet d'associer un identificateur à tout élément
  - **`xml:lang`** utilisé pour décrire la langue du contenu de l'élément
  - **`xml:space`** permet d'indiquer à une application le traitement des caractères d'espacement.
  - **`xml:base`** permet de préciser l'IRI de base d'un élément

Document Type Definition (DTD)