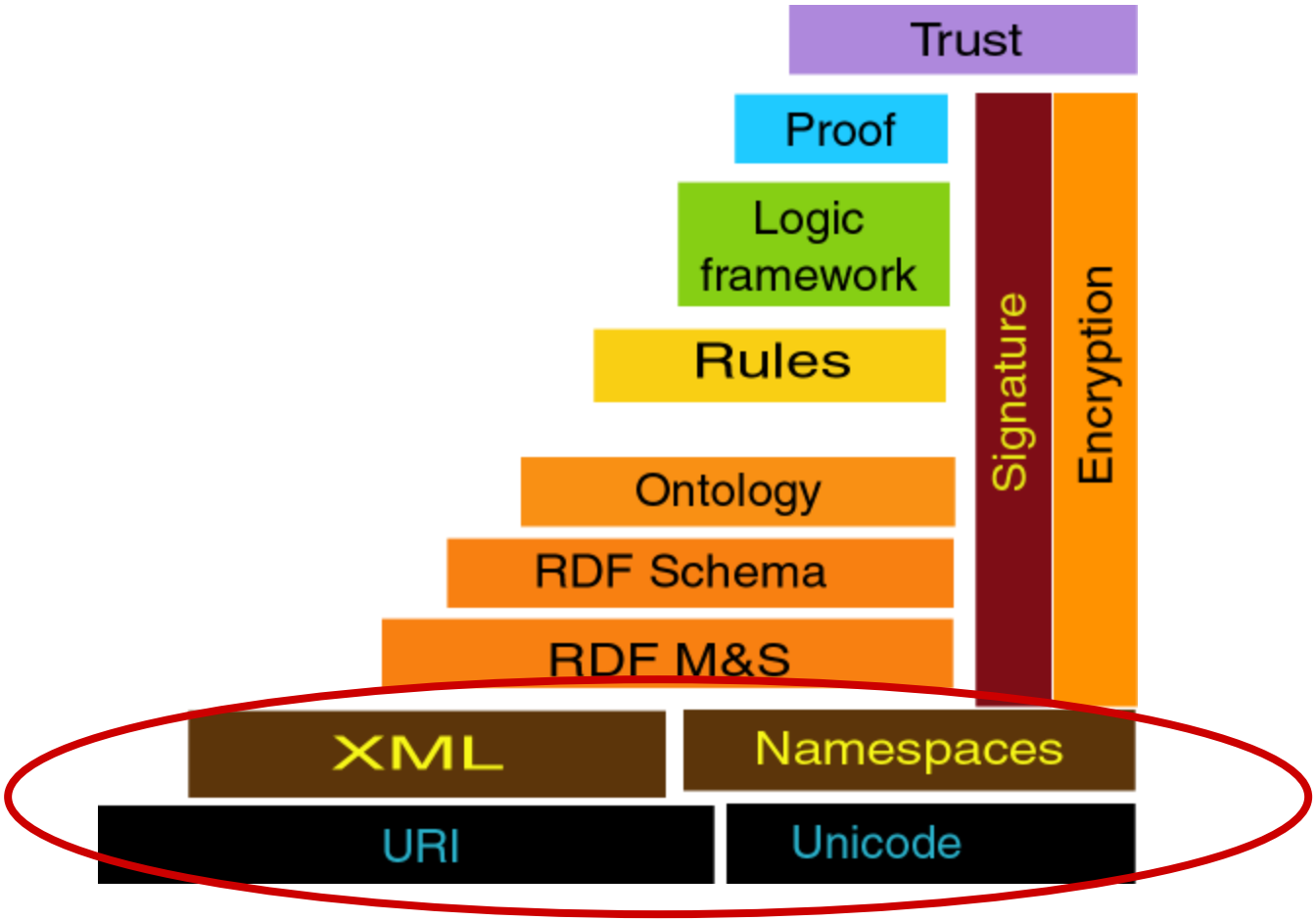
Ontologies et Web sémantique

Cours 3: XML

Dr. TA Tuan Anh
ttanh@ciid.vast.vn

Rappel


Trust
Proof
Logic framework
Rules
Ontology
RDF Schema
RDF M&S
Signature
Encryption
XML
Namespaces
URI
Unicode

Plan

- Syntaxe XML
- Modèles XML: DOM et SAX
- Espaces de noms (namespaces)
- DTD
- Schéma XML (XSD)
- Bases de données XML

Le langage de marquage XML

- XML = eXtensible Markup Language
- Recommandé par W3C pour
  - représenter les documents Web (généralisation de HTML),
  - encoder des données sur le Web pour
    - l'échange,
    - l'intégration et
    - l'interrogation
- XML permet de représenter des données avec une structure irrégulière, implicite et partielle
  - les nouvelles techniques d'intégration et d'interrogation de *données semi-structurées* peuvent être appliquées

Utilités de XML

- Séparer la structure logique des données de leur présentation
  - différentes présentations sont possibles pour le même document
- La forme sérialisée permet de stocker des données dans un document
  - indépendance des outils de gestion de données
- La forme arborescente permet de spécifier des manipulations de données XML
  - modèle de données semi-structurées

Syntaxe XML

- Un ensemble de catégories syntaxiques :
  - les éléments (et leurs attributs)
  - les commentaires
  - les instructions de traitement
  - les sections de texte
  - les sections littérales
- Ainsi que quelques règles sur la structure d'un document

Exemple XML

```
<?xml version="1.0"?>
<bookstore>
    <book category="roman">
        <title lang="en">Every Italian</title>
        <author>Giada De Laurrentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category=“fiction">
        <title lang="en">Harry Porter</title>
        <author>…</author>
        <year>2005</year>
        <price>25.00</price>
    </book>
</bookstore>
```

Eléments

- Un élément est représenté par une balise
  - `<nom_de_balise>…</nom_de_balise>`
- Un nom d'élément ne contient pas de blanc, ni de caractère accentué
- Les majuscules sont distinguées des minuscules
- Il existe une structure arborescente des éléments
  - un élément peut contenir des sous éléments pour son contenu
- Il existe une forme abrégée pour les éléments sans contenu
  - `<C></C>` peut s'écrire `<C/>`
- Tout document comprend un et un seul élément racine

Attributs

- Les attributs constituent un moyen de représenter le contenu d'un élément
  - <A attr1='…' attr2='…'>…</A>
- L'ordre des attributs n'est pas important
- Il doit toujours y avoir une valeur, encadrée par des guillemets (différent de HTML)
- Il ne peut pas y avoir deux attributs avec le même nom dans un élément

Sections de texte

- Un élément peut contenir une section de texte pour représenter son contenu
  - `<A>Le texte…</>`
  - Le texte s'écrit en syntaxe XML (i.e., avec le traitement de caractères spéciales)
- Une section CDATA représente un texte qui n'est pas analysé par le parseur (i.e., pas de traitement de caractères spéciales)
  - `<![CDATA[…]]>`
  - Exemple :

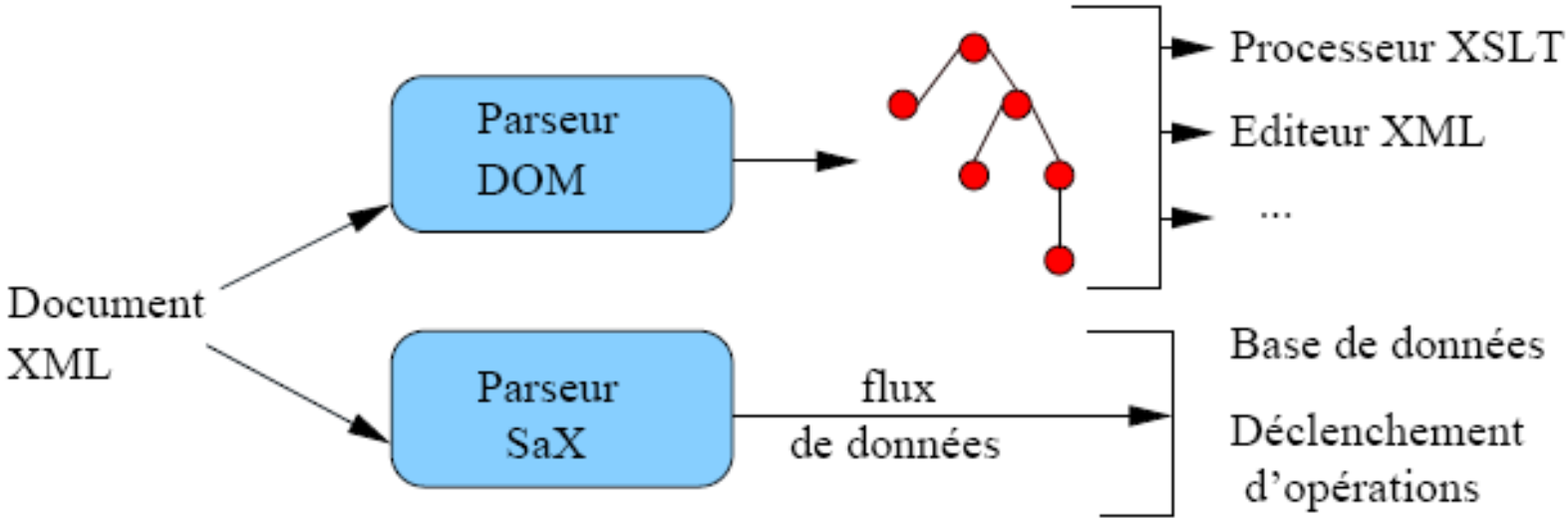
```
<?xml version='1.0'?>
<MYPROGRAM>
<![CDATA[if ((i < 5)&&(j > 6)) printf("error");]]>
</MYPROGRAM>
```

Programmer avec XML

- Deux modèles pour le traitement de documents XML :
  - DOM (Document Object Model), basé sur une représentation hiérarchique
  - SaX (Simple API for XML), basé sur des déclencheurs (événements/action)

Parseur DOM et SaX

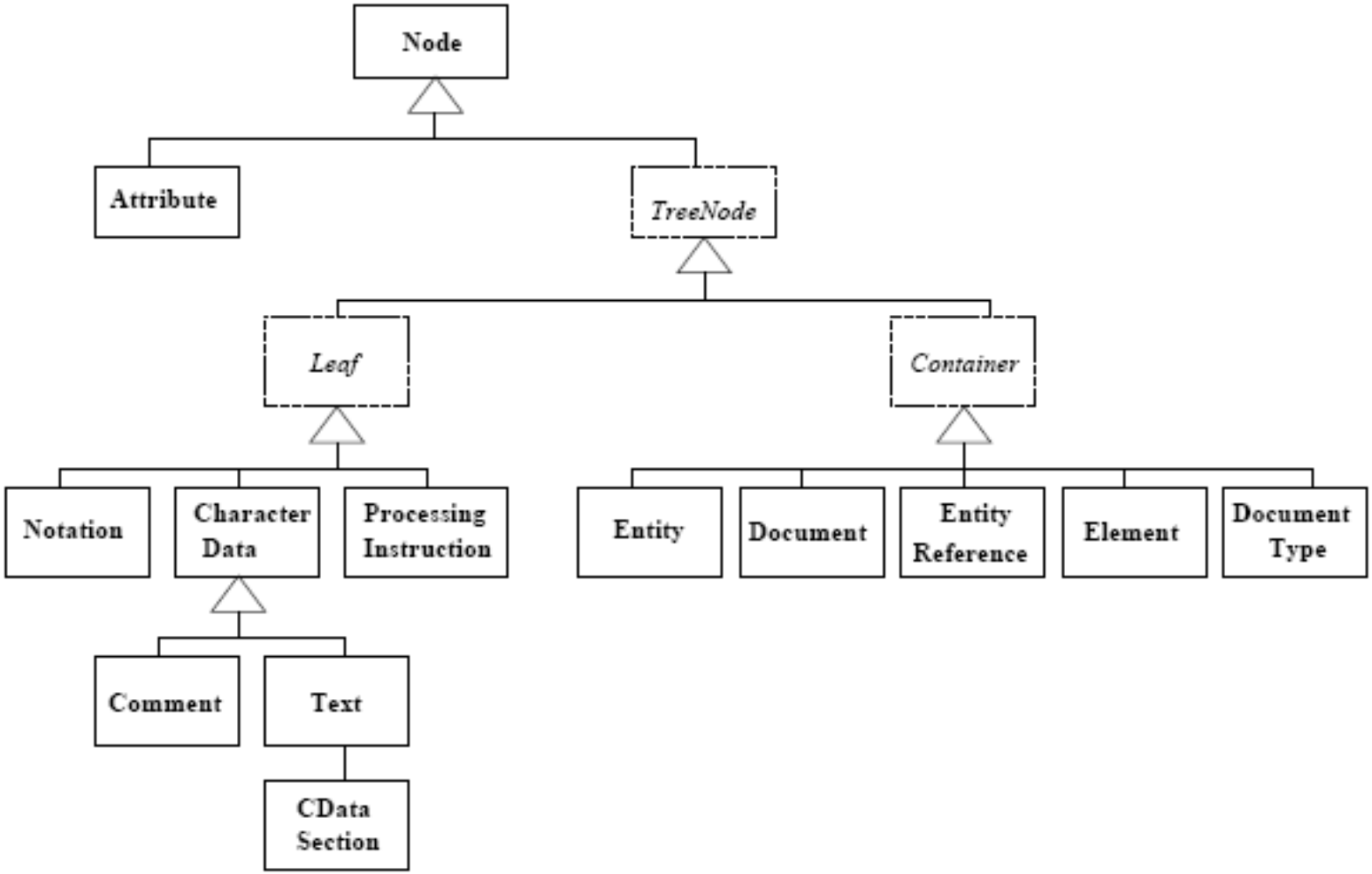
- Toutes les applications XML utilisent un parseur XML pour passer les documents par une phase préalable d'analyse



DOM

- Un parseur DOM prend en entrée un document XML et construit un arbre formé d'objets :
  - chaque objet appartient à une sous-classe de Node
  - des opérations sur ces objets permettent de créer de nouveaux noeuds, ou de naviguer dans le document
- DOM permet de représenter un document XML sous une vue arborescente de noeuds dont la racine est le noeud de document

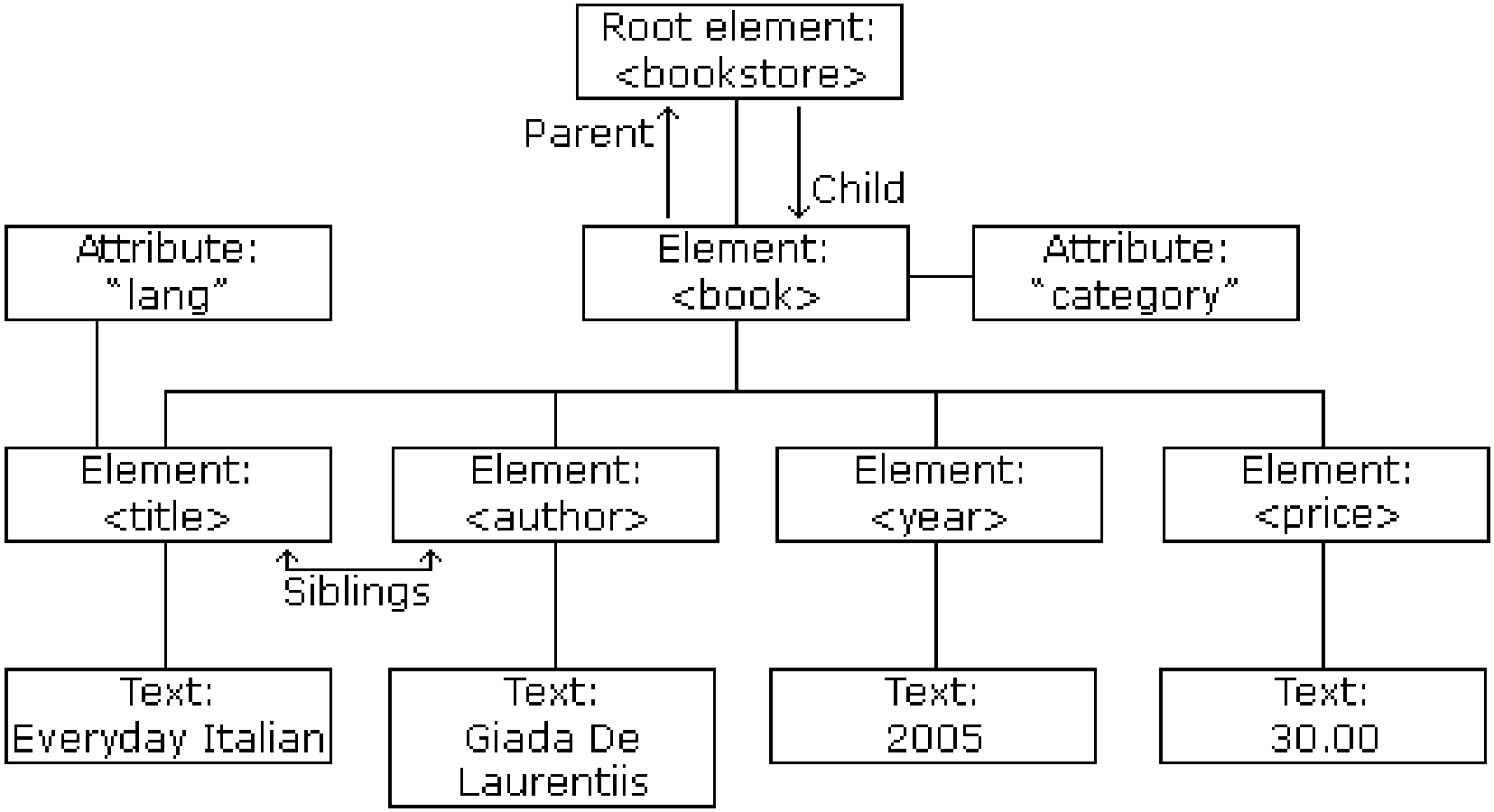
Les types de noeuds DOM



Passage à la représentation DOM

- Un document est analysé par le parseur DOM pour cree un arbre dont
  - le noeud racine est de type `Document`
  - les catégories syntaxiques (commentaires, balises, texte) se traduisent par différents types de noeuds (`Comment, Element, Text`)
  - les attributs se représentent par des noeuds de type `Attribute`
  - les noeuds constituent un arbre qui reflète l'imbrication des éléments dans la forme sérialisée

Exemple DOM


Root element:
<bookstore>
Parent
Child
Attribute:
"lang"
Element:
<book>
Attribute:
"category"
Element:
<title>
Element:
<author>
Element:
<year>
Element:
<price>
Siblings
Text:
Everyday Italian
Text:
Giada De
Laurentiis
Text:
2005
Text:
30.00

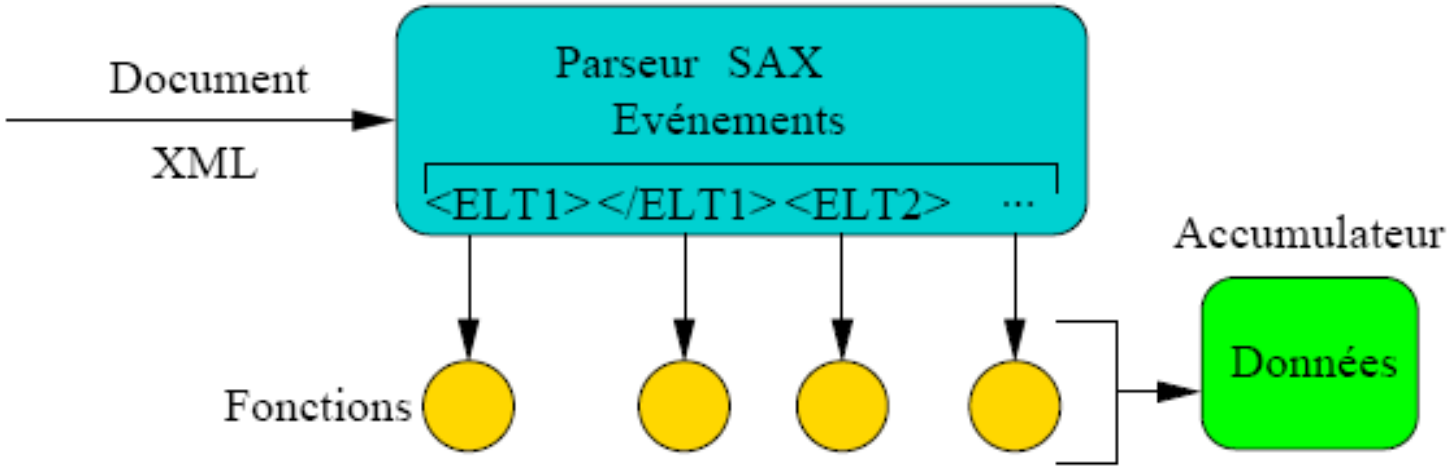
Exercice

- Construire la représentation DOM du document suivant

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<A>
   Le texte de A
   <B>Le texte de B</B>
   <D attr1="1" attr2="azerty">
         <C/>
         <![CDATA[2x < y]]>
   </D>
</A>
```

SAX

- Associer des événements aux balises
- Avoir pour but de traiter le document en mode de flux (temps réel)



Les événements SAX

- Un parseur SAX génère un événement à chaque fois qu'il rencontre
  - le début et la fin de document,
  - le début et la fin d'un élément,
  - une instruction de traitements
  - un commentaire, ...
- Les fonctions s'exécutent indépendamment

Exemple SAX

```
public class Trace extends DefaultHandler {
    public void startDocument() {
        System.out.println("start document");
    }
    public void endDocument() {
        System.out.println("end document");
    }
    public void startElement(String uri, String localName,
                String qName, Attributes attributes) {
        System.out.println("starting element: " + qName);
    }
    public void endElement(String uri, String localName,
                                        String qName) {
        System.out.println("end element: " + qName);
    }
    public void characters(char[] ch, int start, int length){
        System.out.println("character data, length " + length);
}
```

Espaces de noms (namespace)

- Un espace de noms XML (namespace) est une collection de noms d'éléments ou noms d'attributs
  - identifié par une URI
- Les espaces de noms permettent d'intégrer des documents en évitant des conflits de noms

Exemple

```
<mydoc>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
 <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
 </h:tr>
</h:table>
<f:table
   xmlns:f="http://www.w3schools.com/furniture/">
 <f:name>African Coffee Table</f:name>
 <f:width>80</f:width>
 <f:length>120</f:length>
</f:table>
</mydoc>
```

Noms qualifiés

Noms qualifiés	Noms universels
h:table	http://www.w3.org/TR/html4/table
h:tr	http://www.w3.org/TR/html4/tr
h:td	http://www.w3.org/TR/html4/td
f:table	http://www.w3schools.com/furniture/table
f:name	http://www.w3schools.com/furniture/name
f:width	http://www.w3schools.com/furniture/width
f:length	http://www.w3schools.com/furniture/length

Espaces de noms par défaut

```
<table xmlns="http://www.w3.org/TR/html4/">
 <tr>
  <td>Apples</td>
  <td>Bananas</td>
 </tr>
</table>
<table xmlns="http://www.w3schools.com/furniture/">
 <name>African Coffee Table</name>
 <width>80</width>
 <length>120</length>
</table>
```

Règles d'attribution d'un espace de nom

- Noms qualifiés sans préfixe:
  - espace de nom définit par l'attribut `xmlns` de l'ancêtre le 'plus près'
- Noms qualifiés avec un préfixe `pre`:
  - espace de nom définit par l'attribut `xmlns:pre` de l'ancêtre le 'plus près'

Exercices

- Identifier le nom universel des balises
- Exemple 1:

```
<?xml version="1.0"?>
<p:A xmlns:p="http://a.b.c/">
   <B attr1="val1“ p:attr1="val2"/>
   <p:C/>
</p:A>
```

- Exemple 2

```
<?xml version="1.0"?>
<A xmlns:p="http://a.b.c/">
   <p:B/>
   <p:B xmlns:p="http://x.y.z/">
       <p:C/>
   </p:B>
</A>
```

Suite…

- Exemple 3:

```
<?xml version="1.0" ?>
<A xmlns:p="http://a.b.c/"
   xmlns:q="http://a.b.c/">
   <p:B/>
   <q:B/>
</A>
```

- Exemple 4:

```
<?xml version="1.0" ?>
<A xmlns="http://a.b.c/" xmlns:vide="">
   <B/>
   <vide:B>
       <C/>
   </vide:B>
</A>
```

Documents XML valides et bien-formés

- Document XML bien-formé:
  - conforme la syntaxe imbriquée (arborescence XML)
- Document XML valide:
  - respecte une DTD (Document Type Definition)
    - décrivant formellement la structure du contenu
  - respecte l'intégrité référentielle
    - toutes les valeurs d'attributs de type ID sont distinctes
    - toutes les références sont valides

Exemple DTD

```
<!DOCTYPE BookStore [
    <!ELEMENT bookstore (book*)>
    <!ELEMENT book(title, author, year, price?)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT author (#PCDATA)>
    <!ELEMENT year (#PCDATA)>
    <!ELEMENT price (#PCDATA)>
    <!ATTLIST book category CDATA #REQUIRED>
    <!ATTLIST title lang CDATA “en”>
]>
```

ELEMENT

- Un élément est défini par un nom et un modèle de contenu
  - `<!ELEMENT nom (contenu)>`
- Modèle de contenu
  - Expression régulière sur l'alphabet des noms d'éléments
    - `A` = un et seulement un fils A
    - `A*` = zéro ou n occurrences
    - `A+` = au moins une occurrence
    - `A?` = zéro ou une occurrence
  - `EMPTY` = élément vide;
  - `ANY` = toute combinaison de tous les éléments;
  - `#PCDATA` = texte
  - Contenu mixte : `(#PCDATA | A | B ...)*`

Exemple

- Un cinéma est composé d'un nom, d'une adresse optionnelle et d'une suite de séances :
  - `<!ELEMENT cinéma (nom, adresse?, (séance)*)>`
- Une personne a un nom, plusieurs numéros de téléphone et au moins une adresse email :
  - `<!ELEMENT personne (nom, tel*, email+)>`

ATTLIST

- Un attribut est défini pour un élément par un nom et le type de données
  - `<!ATTLIST élément attribut type valeur>`
  - E.g., `<!ATTLIST category CDATA #REQUIRED>`
- Type d'attributs
  - `CDATA` : chaîne de caractères
  - `(en1|en2|..)` : énumération
  - `ID` : unique id
  - `IDREF` : une référence id
  - `IDREFS` : liste de références id
  - `NMTOKEN` : nom XML validé
  - `ENTITY` : une entité
- Valeur d'attributs
  - "…" : valeur par defaut
  - `#REQUIRED` : l'attribut doit être déclaré
  - `#IMPLIED` : l'attribut optionnel
  - `#FIXED` valeur : valeur fixé

Reference

```
<!DOCTYPE Lib [
     <!ELEMENT lib (book*, person*)>
     <!ELEMENT book(title, author)>
     <!ELEMENT person (name)>
     <!ELEMENT author EMPTY>
     <!ELEMENT title (#PCDATA)>
     <!ELEMENT name (#PCDATA)>
     <!ATTLIST person personid ID #IMPLIED>
     <!ATTLIST author personref IDREF #REQUIRED>
]>
<lib>
     <person personid="toto">
          <name>Toto</name>
     </person>
     <book>
          <title>Book 1</title>
          <author personref="toto"/>
     </book>
     <book>
          <title>Book 2</title>
          <author personref="tata"/>
     </book>
</lib>
```

ENTITY

- Les entités sont des variables référencés aux textes communs
- Deux types de déclaration : interne et externe
  - `<!ENTITY entité "valeur">`
  - `<!ENTITY entité SYSTEM "URI/URL">`
- Exemple DTD :

```
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">
```

ou

```
<!ENTITY writer SYSTEM
   "http://www.w3schools.com/dtd/entities.dtd">
<!ENTITY copyright SYSTEM
   "http://www.w3schools.com/dtd/entities.dtd">
```

- Exemple XML :

```
<author>&writer; and &copyright;</author>
```

Résumé sur les DTDs

- Une DTD décrit la structure d'un ensemble de documents XML
- Tous les parseurs XML permettent de valider un document XML par rapport à une DTD
- Il existent des langages plus riches pour la description d'un document XML: XML Schema, Relax NG

Schéma XML

- Une alternative à base de XML pour DTD
  - Un schéma XML est lui même un document XML
- Typage et namespace
  - Séparation entre types et éléménts
  - Types complexes, abstraits et anonymes
  - Sous-typage par extension et restriction
  - Contraintes d’intégrité (clés, clés étrangères)

Types simples

- DTD: un seul type simple (#PCDATA) et 10 types d'attributs
- Schéma XML: 43 types simples
  - xsd:string, xsd:byte, ...
  - xsd:integer, xsd:long, xsd:float, xsd:double, ...
  - xsd:boolean
  - xsd:time, xsd:timeDuration, xsd:Date, xsd:year, xsd:month, ...
  - xsd:language, xsd:uriReference
  - xsd:ID, xsd:IDREF, xsd:NMTOKEN, ...

Restrictions de types simples

- On peut restreindre les types simples
  - par leur longueur (`length, minLength, maxLength`)
  - par des motifs (chaînes de caractères),
  - par énumération,
  - par des intervalles (`maxInclusive, maxExclusive, minInclusive, minExclusive`) et
  - autres (`precision, scale, encoding, period, duration`)

Restriction par motifs

- Numéro de téléphone: +33-(0)-1-34-45-67-89
- DTD: `#PCDATA`
- Schéma XML: Similaire aux expressions régulières Unix

```
<xsd:simpleType name='telType'>
   <xsd:restriction base='xsd:string'>
      <xsd:pattern value='+33-(0)-\d(-\d{2}){4}'/>
   </xsd:restriction>
</xsd:simpleType>
```

Eléments

- Déclaration d'éléments
  - `<xsd:element name="…" type ="…"`
    `[contraintes] [value]/>`
  - `<xsd:element ref="…" [contraintes]/>`
- [contraintes] :
  - `minOccurs="…"`
  - `maxOccurs="…"`
- [value] :
  - `default="…"`
  - `fixed="…"`
- Exemple:

```
<xsd:element name='nom' type='xsd:string'
                minOccurs='0' maxOccurs='2'/>
```

Attributs

- Déclaration d'attributs:
  - `<xsd:attribute name="…" [use] [value]/>`
- [use] :
  - `use="required"`
  - `use="optional"` par défaut
- [value] :
  - `default="…"`
  - `fixed="…"`

- Exemple:

```
<xsd:attribute name='langage'
   type='xsd:language' optional='true'/>
```

Types complexes

- Trois constructeurs de type:
  - `xsd:sequence`: séquence ordonnée d'éléments
  - `xsd:all`: séquence non-ordonnée d'éléments
  - `xsd:choice`: choix d'éléments (DTD: '|')
  - `xsd:group`: regroupement (DTD: '(...)')

Type complexe: Exemple

- Modèle de contenu DTD:

```
book(title, author, year, price)
```

- XML Schéma:

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
      <xs:element name="year" type="xs:year"/>
      <xs:element name="price" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Contenu mélangé (mixed)

- DTD :

```
<letter>
Dear Mr.<name>John Smith</name>. Your order
<orderid>1032</orderid> will be shipped on
<shipdate>2001-07-13</shipdate>.
</letter>
```

- Schéma XML :

```
<xs:element name="letter">
 <xs:complexType mixed="true">
   <xs:sequence>
     <xs:element name="name" type="xs:string"/>
     <xs:element name="orderid" type="xs:integer"/>
     <xs:element name="shipdate" type="xs:date"/>
   </xs:sequence>
 </xs:complexType>
</xs:element>
```

Groupes: exemple

- DTD: `((B|C)*,D)+`
- XML schéma:

```
<xsd:complexType>
  <xsd:group minOccurs='1' maxOccurs='unbounded'>
    <xsd:sequence>
      <xsd:group minOccurs='0' maxOccurs='unbounded'>
        <xsd:choice>
          <xsd:element name='B' xsd:type='xsd:string'/>
          <xsd:element name='C' xsd:type='xsd:string'/>
        </xsd:choice>
      </xsd:group>
      <xsd:element name='D' xsd:type='xsd:string'/>
    </xsd:sequence>
  </xsd:group>
<xsd:complexType>
```

Extension de types complexes

```
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Bases de données XML

- XML est une sorte de SGBD avec un support de
  - stockage de données (les documents XML)
  - de schéma de données (DTD, XML Schémas, RELAX NG, etc.)
  - des langages de requête (XQuery, XPath, XQL, etc.)
  - des interfaces de programmation (SAX, DOM, etc.)
- Points faibles de SGBD XML
  - stockage efficace
  - indexation
  - sécurité
  - transactions et l'intégrité des données
  - etc.

Stockage de XML

- 3 solutions de stockage

1. Fichiers ‘plats’
   - Petits documents
   - Avantages : temps de chargement/reconstruction
2. Bases de données étendues
   - SGBD (objet-) relationnelle étendu avec des outils pour le traitement de documents XML
   - Avantages : modèle de stockage/interrogation avec SQL
3. Bases de données XML natives
   - Modèle conçu pour le stockage et l’accès à des arbres ordonnées
   - Avantages : chargement/mise-a-jour efficace de gros documents

Documents XML et Relations

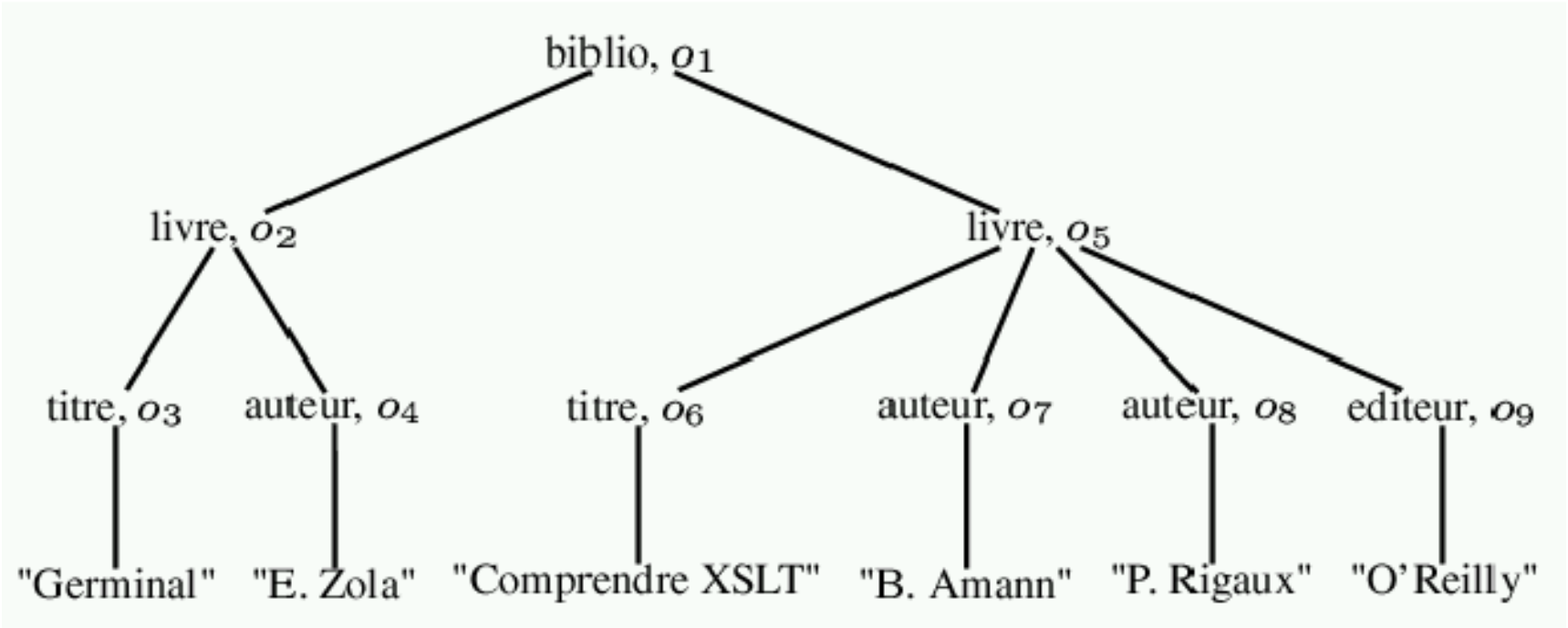
- XML:
  - modèle d’arbres ordonnés
  - structure irrégulière : éléments/attributs optionnels, éléments multiples
- Relations :
  - modèle ensembliste (relation = ensemble de n-uplets)
  - absence d’ordre
  - schéma obligatoire

Exemple: XML ↔ Relations

```
<?xml version='1.0'?>
<biblio>
  <livre>
    <titre>Germinal</titre>
    <auteur>E. Zola</auteur>
  </livre>
  <livre>
    <titre>Comprendre XSLT</titre>
    <auteur>B. Amann</auteur>
    <auteur>P. Rigaux</auteur>
    <editeur>O'Reilly</editeur>
  </livre>
</biblio>
```

Bernd Amann, 2003

Arbre XML


biblio, o_1
livre, o_2
livre, o_5
titre, o_3
auteur, o_4
titre, o_6
auteur, o_7
auteur, o_8
editeur, o_9
"Germinal"
"E. Zola"
"Comprendre XSLT"
"B. Amann"
"P. Rigaux"
"O'Reilly"

Stockage en relation

- Une table binaire pour stocker l'ordre, les balises et la relation parent/enfant
- Une table unaire pour les valeurs

R:

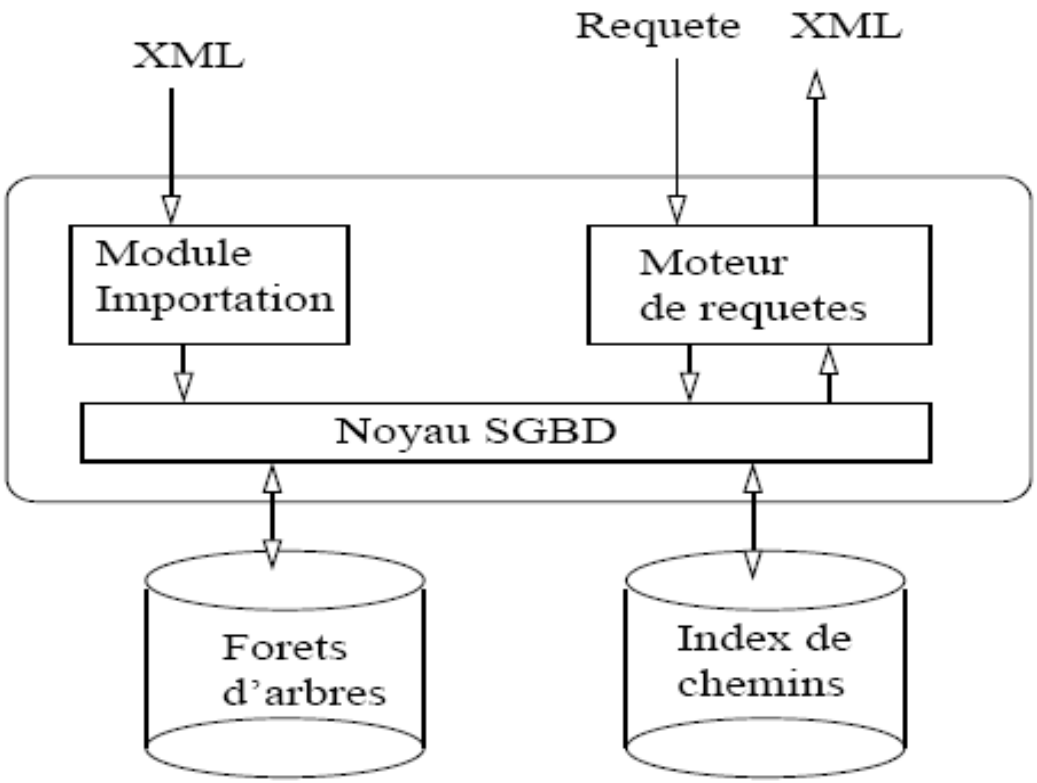
Part	Pos	Lab	Type	Id
o_0	1	biblio	ref	o_1
o_1	1	livre	ref	o_2
o_1	2	livre	ref	o_5
o_2	1	titre	cdata	o_3
o_2	2	auteur	cdata	o_4
o_5	1	titre	cdate	o_6
o_5	2	auteur	cdata	o_7
o_5	3	auteur	cdata	o_8
o_5	4	editeur	cdata	o_9

S:

Noeud	Val
o_3	Germinal
o_4	E. Zola
o_6	Comprendre XSLT
o_7	B. Amann
o_8	P. Rigaux
o_9	O'Reilly

Systèmes de stockage “natives”

- Tamino de Software AG
- Xyleme/Natix
- XIndice de Apache
- X-Hive DB
- IXIA Soft TextML



Langages de requête XML

- Comment interroger des documents XML?
- Solutions :
  - SQL : il faut stocker XML dans une BD relationnel
  - expressions XPath : extraction de fragments
  - règles XSLT : extraction + transformation (règles)
  - XQuery : vrai langage de requêtes pour XML

Ressources bibliographiques

- XML W3C
  http://www.w3.org/XML/
- XML / Database Links
  http://www.rpbourret.com/xml/XMLDBLinks.htm
- Cours DEA SIR, 2003/04, Module BD XML de Bernd Amann