

XML & Outils associés

Samir Bennani

sbennani@emi.ac.ma

PROGRAMME

- **Introduction à XML**
- ✓ **Les feuilles de style CSS**
- ✓ **XML avancé, Les DTDs, Les schémas**
- ✓ **Les feuilles de style et de transformation XSL**
- ✓ **Intégration XML, CSS, XSL, HTML, DTD,**
- ✓ **Intégration avec autres outils : JavaScript, ASP, ...**

XML : Introduction

Origine / objectifs

- La gestion de données s'est déclinée en deux branches vers 1970:
 - **BD structurées** (Réseau, Relationnel, Objet)
 - Tables objet-relationnel
 - Langage de requêtes SQL
 - **Gestion Electronique de documents** (GED)
 - Documents balisés
 - Recherche d'information par mots-clés
 - Moteurs de recherche (e.g., Google)
- XML est issu de la Gestion de Documents (GED)

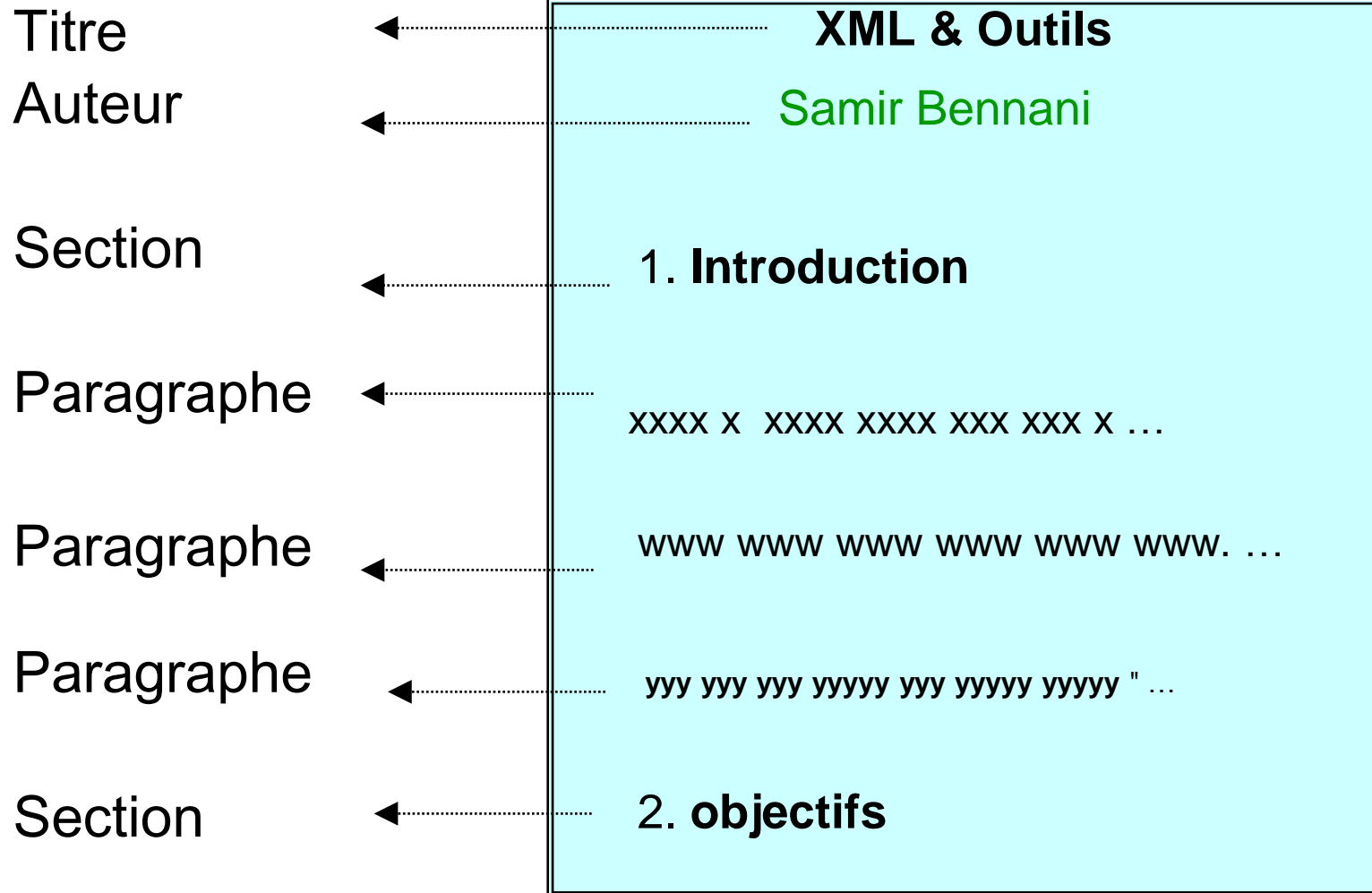
Document Web

- Un document = contenu + forme.
 - Contenu (fond) : structure + sémantique (contenu)
 - Forme : structure + présentation (Mise en forme)
- L'approche Web classique :
 - HTML pour la présentation et pour le fond
 - mélange le fond et la forme
 - SGML pour la structuration

Au commencement...

- SGML
 - Standard Generalized Markup Language
 - Langage à balises : représente documents structurés
 - Complexe, difficile à apprendre
 - Réservé aux professionnels - documentation
 - Difficilement extensible au Web
- HTML ! (dérive de SGML)
- Conçu pour afficher des « pages web »
 - ➔ C'est ce qui a fait son succès

Présentation et Structuration



Vue Balisée XML

<Livre>

<Titre> XML & Outils </Titre>

<Auteur> Samir Bennani </Auteur>

<Section titre = "Introduction">

<Paragraphe> xxxx x xxxx xxxx xxx xxx x </Paragraphe>

<Paragraphe> www www www www www www.. ...</Paragraphe>

<Paragraphe> yyy yyy yyy yyyyyy yyy yyyyyy yyyyyy ... </Paragraphe>

</Section>

<Section titre= " objectifs "> ...

</Section>

</Livre>

Insuffisances dans HTML

- **Pas de structure** de document
 - Entête, corps....
 - Difficultés d'imbrication de Balises P (sauf DIV et SPAN)
- document HTML = contenu & présentation (styles)
 - lecture fastidieuse
 - **mise à jour** du contenu difficile → engendre erreurs
 - **mise à jour de style** d'éléments de même nature délicate (omissions) et fastidieuse (répétitive)
 - accès non différencié selon utilisateur au
 - contenu : auteurs, rédacteurs
 - présentation : informaticiens, graphistes

Besoins en gestion des documents

- Difficile d'assurer **cohérence et MAJ** des documents
- **Besoin d'échange** et partage des documents
 - éviter les **ressaisies**
 - éviter les **conversions** en vue du traitement (e-commerce)
- De + en + de demande en **édition collaborative** de documents
 - les GED sont souvent complexes
- Besoins en **personnalisation** du contenu
 - l'information doit **s'adapter à l'utilisateur**, son profil, ses besoins
- Besoin de support **d'interopérabilité**
 - plusieurs modèles proposés par les architectures
 - CORBA, OMG, EJB, DCOM, ...
 - Hétérogénéité → Choix ???

XML : eXtensible Markup Language

- Outil de structuration de document
- Outil de description de contenu de site Web
- Outil de gestion de documents
- Outil de base pour applications de traitement des documents

XML : Objectifs

- Langage normalisé de description
 - Création libre des balises
 - plus simple que SGML (s/ensemble simplifié exploitable sur Web)
 - plus ouvert que HTML englobé (XHTML)
 - Structure hiérarchique vérifiable
 - Langage simple, documents « lisibles »
 - Facilite le traitement des données
 - Sépare le contenu et la forme
 - Indépendant des plateformes

XML : Objectifs

- Grande variété d'applications
 - Web et autres,
 - échange de Données,
 - 1 document :: plusieurs formes de diffusion (Web, PDA, PDF...)
 - Modèles
- conception de pages Web facile, rapide

XML

- issu de SGML (ISO 8879 - 1986)
 - SGML complexe appliqué à la GED
 - XML Plus simple & plus flexible
- XML normalisé par le W3C (1996)
 - par XML Working group
 - maintenant par XML Activity
 - version actuelle : XML 1.1 du 16 Aout 2006
- Autres outils associés
 - XSL : eXtensible Style Language
 - DOM : Document Object Model
 - RDF : Resource Description Framework
 - CSS : Cascading Style Sheet

XML

- Métalangage
 - utilisable pour créer son propre langage
- Implémentation selon type d'application et d'échange
 - entre machine et utilisateurs (Web)
 - entre applications
- 3 1/3 intervenant
 - analogie client/serveur
 - Client - Middle - Serveur BD
- Séparation contenu et présentation

les 10 objectifs de conception:

- XML doit pouvoir être utilisé **sans difficulté sur Internet**
- XML doit soutenir une grande **variété d'applications**
- XML doit être **compatible** avec SGML et HTML
- Il doit être **facile d'écrire des programmes traitant** les documents XML
- Le **nombre d'options** dans XML doit être **réduit** au minimum, idéalement à aucune

les 10 objectifs de conception

- Les documents XML doivent être lisibles par l'homme et raisonnablement clairs
- La spécification de XML doit être disponible rapidement
- La conception de XML doit être formelle et concise
- Il doit être facile de créer des documents XML
- La concision dans le balisage de XML est peu importante

XML: définitions de base

- XML :
 - méta-langage universel
 - pour représenter les données échangées sur le Web
 - qui permet au développeur de délivrer du contenu entre applications
- XML standardise la manière dont l'information est :
 - échangée
 - présentée
 - archivée
 - retrouvée
 - transformée
 - cryptée
 - ...

Concepts du modèle

- **Balise (ou tag ou label)**

- Marque permettant de repérer un élément textuel
- Forme: <balise> de début, </balise> de fin

- **Élément de données**

- Texte encadré par une balise de début et une de fin
- Les éléments de données peuvent être imbriqués

<Client>

<adresse>

<rue> 1 Av de la Victoire</rue>

<ville> Rabat</ville>

</adresse>

</Client>

- **Attribut**

- Doublet nom= "Alami" qualifiant une balise
 - <Client nom= "Alami" num= "256015" region= " FES " >

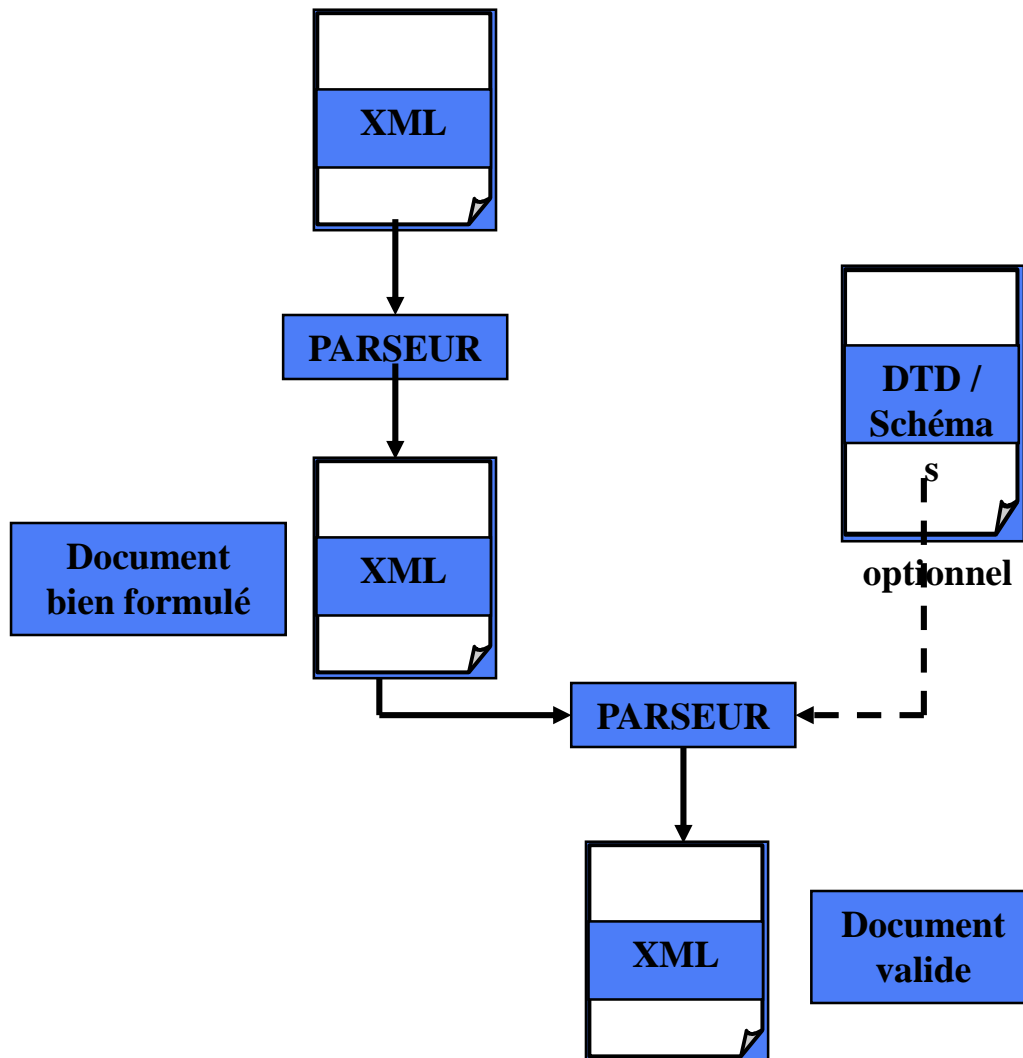
XML - technologies associées

- **HTML**
 - toujours présent parfois même avec XML
- Feuilles de Style en Cascade
 - **CSS** ; Cascading Style Sheet
 - associé à HTML ou XML
 - plusieurs styles pouvant s'appliquer en cascade
- **XSL** : eXtensible Style Sheet
 - spécifiquement conçu pour XML
 - langage de description de style, compatible avec CSS
 - choix de style contextuel des éléments
 - langage de transformation, présentation sélective

XML - technologies associées

- DTD : Document Type Definition
 - création de document XML conforme à un style de document
- Xlinks et Xpointers
- Caractère Unicode, DOM,

Traitement XML

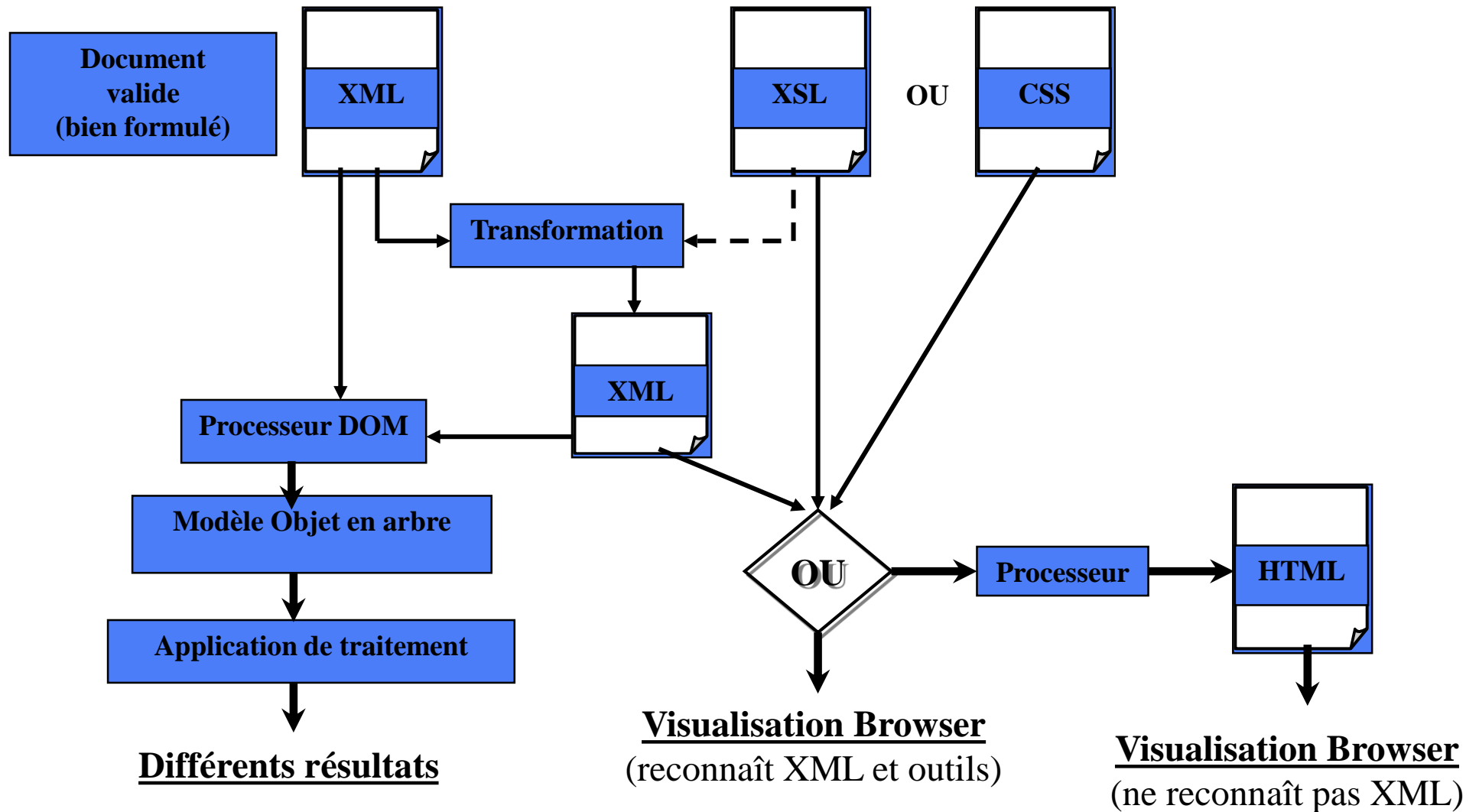


formulation correcte et validation
peuvent se faire simultanément
par les parseurs (processeurs)
XML

Parseurs :

- Analyses lexicale, syntaxique
- Validation (si DTD / Schémas),
Contrôle de conformité de la
syntaxe % norme
- Expansion des entités
paramètres si DTD
- Report d'erreurs en cas de
divergence.

Traitement XML (2)



Les applications XML

- Destinées à des documents XML en vue d'appliquer un traitement
 - par applications complètes ou plug-ins dans le navigateur
- Les applications de style
 - XSL, CSS, ...
- MathML
 - formulation d'expressions mathématiques complexes
 - prise en charge par le navigateur Amaya
- CML (Chemical Markup Language)
 - navigateur Jumbo
- SMIL : Synchronized Multimedia Integration Language
 - Real Player
- XHTML : eXtensible HTML
 - HTML en tant qu'Application XML => Document HTML bien formé

Les applications XML (2)

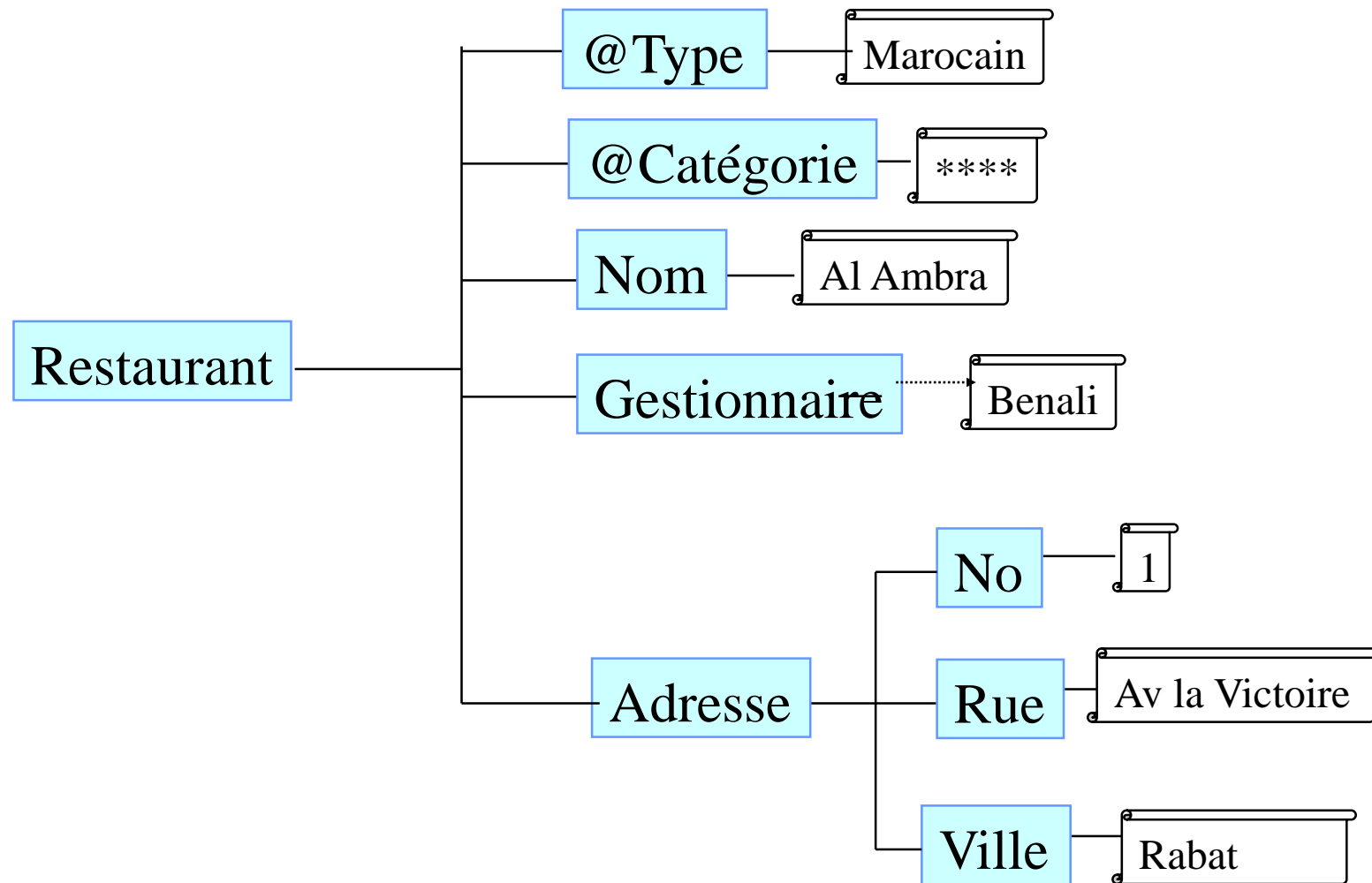
- RDF : Resource Description Framework
 - Utile pour la recherche bibliographique et lexicale
 - Utilisé par le Push dans IExplorer
- Description d'images vectorielles
 - SVG (Scalable Vector Graphic) W3C ou PGML (IBM...) ou VML (MS)
- MusicML
- Utilisation dans EDI, dans le domaine financier (OTP, OFX)
- WebDAV : HTTP Extensions for Distributed Authoring (IETF)
- XML Signature (W3C : état de travail pré-normatif)
 - intégrité d'un document numérique, authenticité et non répudiation
 - XML Security Suite de IBM

Les applications XML (3)

- P3P : Platform for Privacy Preferences
 - Protocole d'échange serveur/client avec respect de données privées
- SOAP : Simple Object Access Protocol
 - Expression d'objets en XML en vue de transport et invocation
 - implémentations en Java
- XML et les Bases de Données
 - Besoin d'échanger des données entre documents et BDs
 - Expression des requêtes : XML-QL
 - Logiciels multiples : DB2XML, ODBC2XML, XSQL Servlet,

Document XML -Structure

Modèle arborescent



Présentation textuelle

<Restaurant Type= " Marocain" Categorie="****">

<Nom>Al Ambra</Nom>

<Gestionnaire ... >Benali</Gestionnaire>

<Adresse>

<No> 1</No>

<Rue> Av la Victoire </Rue>

<Ville>Rabat</Ville>

</Adresse>

</Restaurant>

Structure d'un document XML

- 3 composantes principales

1 . Le Prologue

- déclaration XML
- instructions de traitement
 - applications éventuelles associées
 - références aux feuilles de styles
- Déclaration du type de document

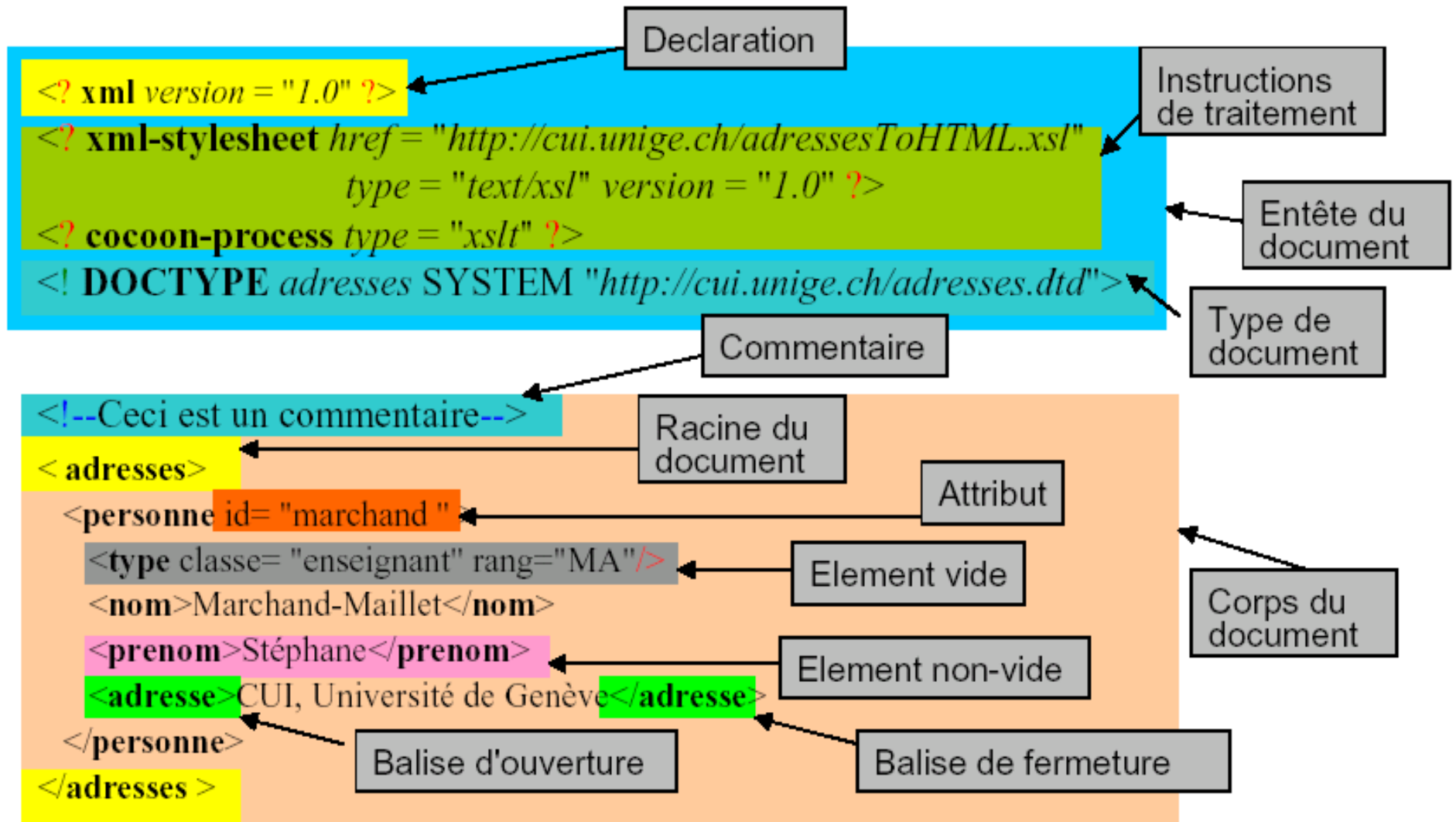
■ . L'arbre des éléments

- chaque élément dans une balise : `<élément> </élément>`
- ou éléments vides : `<élément/>`
- attributs associés à l'élément dans la balise
- références aux entités
 - insérer des caractères réservés, fichiers, caract répétés ...

3 . Les Commentaires

- `<!-- commentaire -->`
- à tout endroit sauf en 1ère ligne

Structure d'un document XML



Exemple de document XML

```
<?xml version="1.0" standalone="no"?>
<!-- le prologue commence par la déclaration xml, le document n'est pas autonome -->
<!-- référence à la feuille de style qui sera appliquée : -->
<?xml-stylesheet type="text/css" href="biblio.css"?>
<!-- référence à la DTD optionnelle qui permettra de valider la document XML : -->
<!DOCTYPE BIBLIOGRAPHIE SYSTEM "BIBLIOGRAPHIE.dtd">
<BIBLIOGRAPHIE>
  <CITATION CLASSE="HOFSTADTER" ID="C1">
    <AUTEUR>Hofstadter, Douglas</AUTEUR>
    <TITRE>How Might Analogy, the Core of Human Thinking,
      Be Understood By Computers?</TITRE>
    <JOURNAL>Scientific American</JOURNAL>
    <MOIS>September</MOIS>
    <ANNEE>1981</ANNEE>
    <PAGES>18-30</PAGES>
  </CITATION>
  <CITATION CLASSE="TURING" ID="C2">
    <AUTEUR>Turing, Alan M.</AUTEUR>
    <TITRE>On Computable Numbers, With an Application to the Entscheidungs-problem</TITRE>
    <JOURNAL> Proceedings of the London Mathematical Society</JOURNAL>
    <SERIES>Series 2</SERIES>, <VOLUME>42</VOLUME>
    <ANNEE>1936</ANNEE> , <PAGES>230-65</PAGES>
  </CITATION>
</BIBLIOGRAPHIE>
```


XML : Les balises

- Un élément XML
 - commence par une balise ouvrante :
`<balise attributs éventuels>`
 - se termine par une balise fermante : `</balise>`
`<balise> élément XML </balise>`
 - sauf les balise vides : `<balise attributs éventuels />`
- Toute balise porte un nom
 - commence par un caractère alphabétique ou "_"
 - peut comprendre : "_", "-" et "."
 - on distingue Majuscules et minuscules
- Toute balise rencontrée est analysée
 - contrairement à HTML qui ignore les balises non valides

Les attributs**

- Une balise peut avoir un ou plusieurs attributs
 - syntaxe : `nomattribut="valeurattribut"`
ou `nomattribut='valeurattribut'`
 - `<TRIANGLE base="10cm" hauteur="12cm" />`
- Les noms des attributs
 - les noms d'attribut : mêmes règles que les noms de balise
 - les attributs d'une même balise doivent être différentes
- Les valeurs des attributs
 - considérées comme des chaînes de caractères
 - choix du délimiteurs " ou ' en fonction du contenu
sinon utiliser ' et "

XML La syntaxe

- Le texte d'un document XML peut être composé
 - de tous les caractères imprimables
 - RC, LF et Tab
- Les commentaires
 - compris entre "`<!--`" et "`-->`"
 - ne peut pas comporter "--" (2 tirets consécutifs)
 - à tout endroit du document
 - sauf en 1^{ère} ligne
 - et pas dans une balise
 - une balise peut être neutralisée par un commentaire

Références d'entités symboliques

- Certains caractères de texte risquent d'être interprétés
 - ils sont insérés à l'aide des références d'entité prédéfinis dans XML

Référence d'entité	Caractère symbolisé
&amp;	&
&lt;	<
&gt;	>
&quot;	=
&apos;	'

Ajout d'autres entités symbolique par : ENTITY Dans la DTD

Exercice 3 : XML et DTD- *Syntaxe XML*

Identifiez les balises XML correctes et celles erronées.

Indiquez les erreurs:

1. `<Drivers_License_Number>98 NY
32</Drivers_License_Number>`
2. `<Driver's_License_Number>98 NY
32</Driver's_License_Number>`
3. `<month-day-year>1/10/2005</month-day-year>`
4. `<first name>Alan</first name>`
5. `<àçttûä>øåú</àçttûä>`
6. `<PRENOM>Alan</prenom>`
7. `<month/day/year>1/10/2005</month/day/year>`
8. `<_4-lane>I-610</_4-lane>`
9. `<téléphone>0 123 456 789</telephone>`
10. `<4-lane>I-610</4-lane>`

Exercice de syntaxe

- Créez un document XML `annuaire.xml`
 - Construisant un `annuaire`
 - Comportant des entités de type `personne`
 - Chaque personne a une `identite`
 - L'`identite` stipule :
 - un `nom`, un `prenom`, et une `adresse`
 - Une `adresse` est donnée par :
 - une `rue`, un `code-postal`, une `ville` et un ou plusieurs numéros de `telephone`
 - On précise que le `telephone` peut être de `type` « `mobile` » ou « `fixe` »

Exercice Structure XML- DTD

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE annuaire SYSTEM "annuaire.dtd">
<annuaire>
  <personne>
    <identite>
      <nom>Alami</nom>
      <prenom>Driss</prenom>
      <adresse>
        <rue>25 avenue de la Victoire</rue>
        <cp>10000</cp>
        <ville>Rabat</ville>
        <tel>+21261259842</tel>
      </adresse>
    </identite>
  </personne>
  <personne>
    <identite>
      <nom>Benali</nom>
      <prenom>Mohamed</prenom>
      <adresse>
        <rue>18 Rue Gazza</rue>
        <cp>10000</cp>
        <ville>Rabat</ville>
        <tel>+21239584213</tel>
      </adresse>
    </identite>
  </personne>
</annuaire>
```

EDITEURS XML

- XMLwriter
- Visual XML
- XML Spy– <http://www.xmlspy.com>
- Microsoft XML Notepad :
<http://www.microsoft.com>
- ProtoXML
- XMLpro
- AelfredXML
- XMLStyler
- Lark

DTD

Définir une grammaire structurelle

- « DTD » (Document Type Definition)
- La DTD définit la filiation des éléments
 - Quelle est la racine du document?
 - Qui doit/peut avoir quels enfants ?
 - Combien d'enfants ?
 - Qui peut contenir du texte ?
 - Qui a des attributs particuliers ?
 - Quelles sont les valeurs de ces attributs ?
- Indiquée dans le document XML
- On parle de :
 - Document **bien formé** : Respect de la syntaxe XML
 - Document **valide** : Respect de la DTD spécifiée

Déclaration

- DTD (Document Type Definition)
 - Interne : en toutes lettres dans le fichier
 - Externe : dans un fichier extérieur [standalone='no']
- Définit comment le document sera rédigé dans un langage standard propre à la DTD
- Une DTD « n'est pas obligatoire »
 - Si on ne cherche pas à vérifier la validité du document (risqué)

Document XML

Prologue

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>  
<?xml-stylesheet type='text/xsl' href='test.xsl'?>
```

Déclaration

```
<?xml version='1.0' encoding='UTF-8'  
standalone='yes'?>  
<?xml-stylesheet type='text/xsl'  
href='test.xsl'?>  
  
<!DOCTYPE enseignant [  
<!ELEMENT enseignant (nom, prenom, grade,  
enseignements) >  
<!ELEMENT nom (#PCDATA) >  
<!ELEMENT prenom (#PCDATA) >  
<!ELEMENT grade (#PCDATA) >  
<!ELEMENT enseignements (cours+) >  
<!ATTLIST cours type CDATA #REQUIRED>  
<!ELEMENT cours (titre, volume) >  
<!ELEMENT titre (#PCDATA) >  
<!ELEMENT volume (#PCDATA) >  
>
```

Corps

```
<enseignant>  
  <nom>Idrissi</nom>  
  <prenom>Mohamed</prenom>  
  <grade>PES</grade>  
  <enseignements>  
    <cours type='cm'>  
      <titre>XML</titre>  
      <volume>24</volume>  
    </cours>  
    <cours type='td'>  
      <titre>MOO</titre>  
      <volume>33</volume>  
    </cours>  
  </enseignements>  
</enseignant>
```

DOCTYPE : DTD interne = []

Fichier exemple.xml

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE racine [
  <!ELEMENT racine (contenu)>
  <!ELEMENT contenu (#PCDATA)>
]>
<racine>
  <contenu>ceci est un contenu</contenu>
</racine>
```

« DTD »

DOCTYPE : DTD externe SYSTEM

Fichier exemple.xml

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

```
<!DOCTYPE  racine SYSTEM 'exemple.dtd'>
```

```
<racine>
```

```
    <contenu>ceci est un contenu</contenu>
```

```
</racine>
```

Fichier exemple.dtd

```
<!ELEMENT racine (contenu)>
```

```
<!ELEMENT contenu (#PCDATA)>
```

DOCTYPE : DTD externe PUBLIC

- On peut spécifier une DTD « publique » si elle est largement diffusée (standard) ou accessible sur le web

```
<!DOCTYPE racine PUBLIC ‘ ‘url’ ’>
```

Syntaxe (simple) de DTD

- Déclaration des éléments

`<!ELEMENT nom_element EMPTY>`

Élément vide `<Bidon/>`

`<!ELEMENT nom_element ANY>`

Peut contenir n'importe quoi

`<!ELEMENT nom_element (#PCDATA)>`

Element de données

`< nom_element >ceci est une donnée</ nom_element >`

Syntaxe (simple) de DTD

- Déclaration des sous-éléments obligatoires

```
<!ELEMENT nom_element (sous_element)>
```

< nom_element > contient un seul élément *< sous_element >*

```
<!ELEMENT enfant (age,sexe)>
```

<enfant> contient deux éléments

<age> et *<sexe>* DANS CET ORDRE

```
<!ELEMENT enfant (garcon|fille)>
```

<enfant> contient obligatoirement l'un ou l'autre des éléments
<garcon>, *<fille>*

Syntaxe (simple) de DTD

- Déclaration des occurrences

`<!ELEMENT nom_element
(sous_element)>`

Contient **UN SEUL élément**
<*sous_element*> (obligatoire)

`<!ELEMENT eleve (email?)>`

<eleve> contient **UN OU AUCUN**
sous-élément <email>

`<!ELEMENT livre (chapitre+)>`

<livre> contient **UN OU PLUS**
sous-éléments <chapitre>

`<!ELEMENT personne (tel*)>`

<personne> contient
N'IMPORTE QUEL NOMBRE de
sous-éléments <tel>

Syntaxe (simple) de DTD

- Cascade de déclarations

<!ELEMENT **personne**(age,sexe,profession)>

On peut bien sûr étendre le nombre d'éléments
obligatoires

<!ELEMENT **eleve** (nom,prenom,email?)>

<eleve> contient un <nom>, un <prenom>, et
peut-être un <email>

<!ELEMENT **recueil** (titre,(auteur,chapitre)+)>

<recueil> contient un <titre>, puis un nombre non nul de
couples <auteur>,<chapitre>

Syntaxe (simple) de DTD***

- Spécifier le type des attributs

```
<!ATTLIST cours type CDATA #REQUIRED>
```

Spécifie que l'attribut est une chaîne de caractères

```
<!ATTLIST client dossier ID #REQUIRED>
```

La valeur donnée à l'attribut doit être unique

```
<!ATTLIST cours niveau  
(debutant|intermediaire|confirme) #REQUIRED>
```

L'attribut prend une des valeurs spécifiées

Syntaxe de DTD

- Spécifier les attributs (ex: de type chaîne)

```
<!ATTLIST cours type CDATA #REQUIRED>
```

Spécifie que l'attribut doit toujours être fourni

```
<!ATTLIST cours type CDATA #IMPLIED>
```

l'attribut peut ne pas être fourni, dans ce cas l'application se charge de lui choisir une valeur

```
<!ATTLIST cours type CDATA #FIXED 'CM'>
```

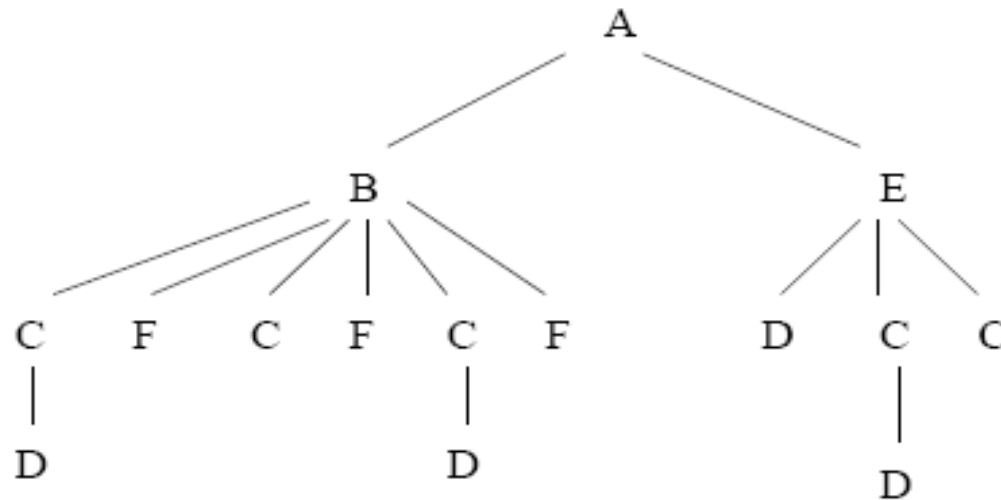
L'attribut a toujours la même valeur "CM"

Syntaxe (simple) de DTD

- Plusieurs attributs
 - Les uns à la suite des autres

```
<!ATTLIST cours type CDATA #IMPLIED niveau (deb|int|conf)  
#REQUIRED>
```

XML : Exercices



Questions :

- Définissez une DTD pour l'arbre : arbre.dtd
- Ecrire le Document XML correspondant

Exercice2 : Voici une DTD et une grammaire d'arbre:

```
<!ELEMENT dossiers (dossier*,
    personne*) >
<!ELEMENT dossier (consultation)* >
<!ELEMENT consultation (symptome+,
    prescription?) >
<!ELEMENT symptome (#PCDATA) >
<!ELEMENT prescription (medicament)*
    >
<!ELEMENT medicament (#PCDATA) >
<!ELEMENT personne (nom, prenom,
    tel?) >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
• <!ELEMENT tel (#PCDATA) >
```

Grammaire :

```
Dossiers -> dossiers (Dossier*, Patient*,
    Medecin*)
Dossier -> dossier (Consultation*)
Patient -> personne (Nom, Prenom)
Medecin -> personne (Nom, Prenom,
    Tel)
Consultation -> consultation
    (Symptome+, Prescription?)
Nom -> nom PCDATA
Prenom -> prenom PCDATA
Tel -> tel PCDATA
Symptome -> symptome PCDATA
Prescription -> prescription
    (Medicament+)
Medicament -> medicament PCDATA
```

- Est-ce que la DTD dtd valide tous les documents validés par la grammaire G
- Donnez un exemple de document validé par la DTD mais pas la grammaire G

Exercice 4 : XML et DTD

Écrire une DTD pour des documents XML - examen.

- Un examen contient un code de cours, un titre et une date composée du mois et de l'année.
- Ces éléments sont suivis par une liste de questions.
- Un examen a de 5 à 6 questions
- chaque question a une ou plusieurs parties.
- Une partie est un mélange de texte et d'autres parties.
- La valeur de mois doit être une chaîne de caractère valide.

Donnez une instance de document valide par rapport à la DTD (externe) et 2 exemples invalides.

XML

Notions Avancées

XSD

Un schéma XML est lui-même un document XML.

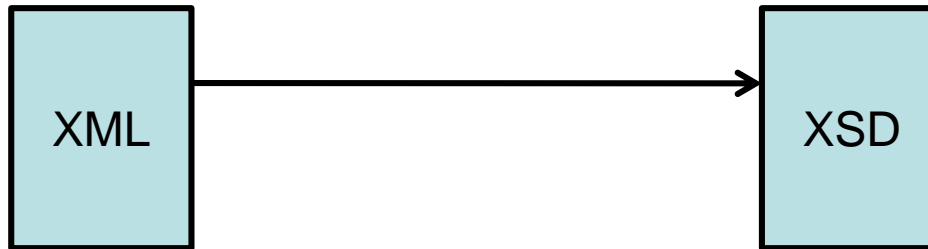
```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsd:schema  
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
.....
```

```
</xsd:schema>
```

XSD



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

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

```
</liste>
```

XSD vs DTD

- Typage des données: permet la gestion de booléens, d'entiers, d'intervalles de temps...
- Extension de types.
- Indicateurs d'occurrences (cardinalités) des éléments par un nombre. (limité à 0, 1 ou l'infini pour une DTD).
- Support des espaces de nom.

Élément simple/Complexe

- Un élément complexe est une balise qui contient la déclaration d'autres éléments et/ou des attributs.

<description>

 <date format="JJ.MM.AAAA">01.04.14</date>

</description>

<etudiant CNE="260949494">....</etudiant>

Élément simple/Complexe

- Un élément simple est une balise qui ne contient aucun attribut et n'imbrique pas d'autre balise.
- Exemple d'élément simple:

<date> 01.04.14 </date>

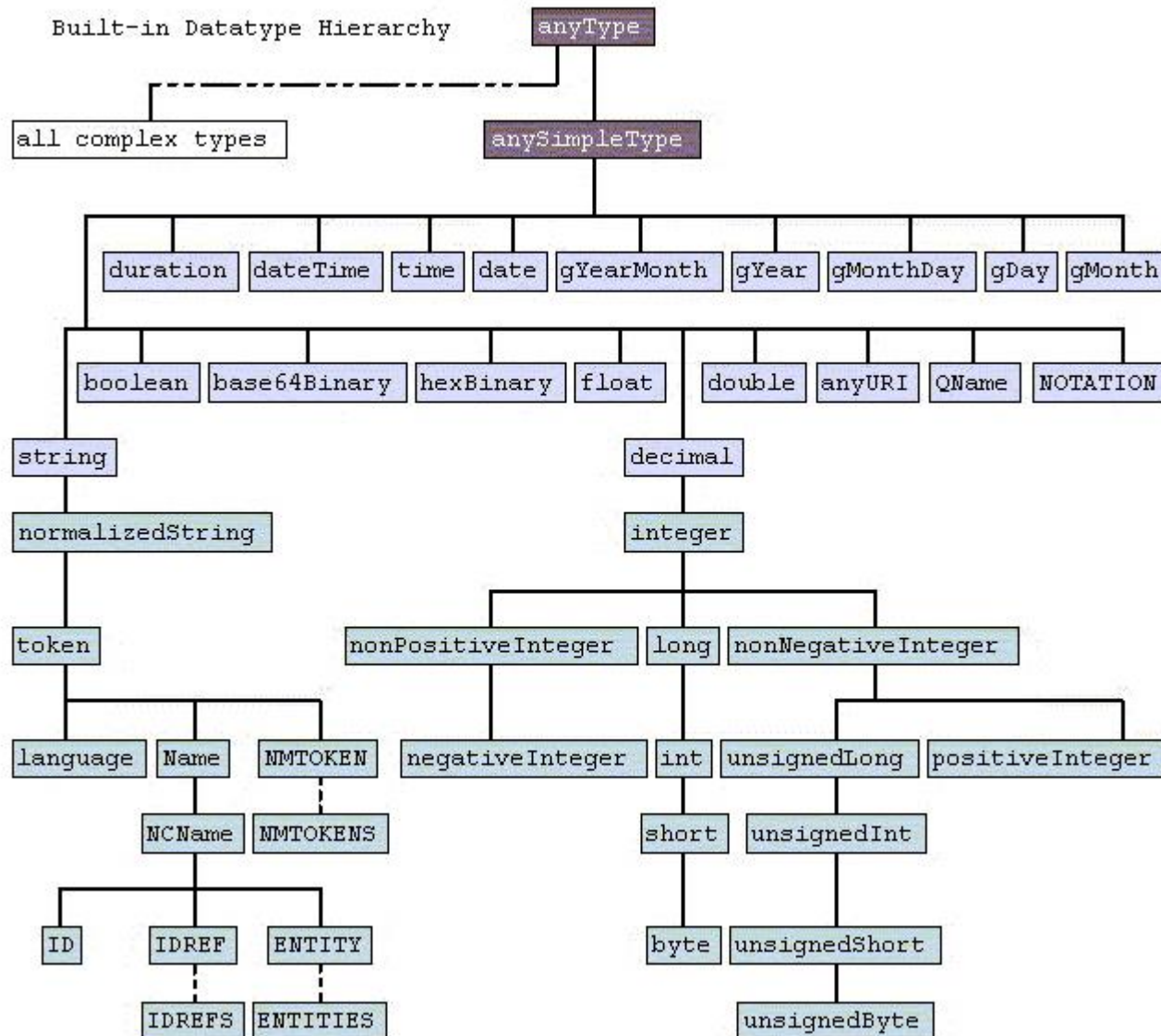
<nom>TAHIRI</nom>

<moyenne>14</moyenne>

Déclaration des éléments


- Pour définir n'importe quelle balise XML on utilise: `<xsd:element>`
- Exemple pour définir
- `<nom>Alami</nom>`
`<xsd:element name="nom" type="xsd:string" />`
- `<filier>Ginf2016</filier>`
`<xsd:element name="filier" type="xsd:string" />`

Déclaration des types



Déclaration des types

1. Définir le type étudiant



```
<xsd:complexType name="EtudiantType" >  
.....  
</xsd:complexType>
```

2. Utiliser la définition de type étudiant dans la liste

```
<xsd:element name="liste">  
.....  
<xsd:element name="etudiant" type="EtudiantType" />  
.....  
</xsd:element/>
```

Exemple: etudiants.xml

```
<liste>
<etudiant code="E0203">
  <nom>Maaroufi</nom>

  <prenom>Manal</prenom>

  <telephone type="GSM">0662569856</telephone>

  <telephone type="Fixe">0535869584</telephone>

  <email>M_maaroufi@gmail.com</email>

  <adresse>N816 Residence alexandrie, Avenue Mohamed5 B.P.20000 Fes, Maroc.</adresse>

  <filiere>Genie Industriel</filiere>

  <moyenne>15</moyenne>
</etudiant>
</liste>
```

Exemple: etudiants.xml

- Question 1: Créer un schéma xml «etudiants.xsd» pouvant valider le document «etudiants.xml»

```
<xsd:schema  
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
.....
```

```
</xsd:schema>
```

```
<liste xmlns:xsi="http://www.w3.org/2001/XMLSchema-  
instance"
```

```
  xsi:noNamespaceSchemaLocation="etudiants.xs  
d">
```

Exemple: etudiant.xml

- Question 2: définir dans le schéma XML «etudiants.xsd» la structure des éléments simples suivants:

```
<nom>Maaroufi</nom>  
<prenom>Manal</prenom>  
<telephone>0662569856</telephone>  
<email>M_maaroufi@gmail.com</email>  
<adresse>N816 Residence Alexandrie, Avenue Mohamed5 B.P.20000 Fes,  
Maroc.</adresse>  
<filier>Genie Industriel</filier>  
<moyenne>15</moyenne>
```

Dans un schéma xml, un type simple est déclaré:

```
<xsd:element name="element" type="type"/>
```

Opérateur de séquence

- L'opérateur **xsd:sequence** définit une suite de sous-éléments dans un ordre déterminé

```
<xsd:complexType name="filiere_type">  
  <xsd:sequence>  
    <xsd:element name="nom" type="xsd:string" />  
    <xsd:element name="descriptif" type="xsd:string" />  
  </xsd:sequence>  
</xsd:complexType>
```

Opérateur de choix

- L'opérateur **xsd:choice** définit un nouveau type XML formé d'un des éléments énumérés.

```
<xsd:complexType name="filiere_type">  
  <xsd:choice>  
    <xsd:element name=" nom" type="xsd:string" />  
    <xsd:element name=" id_filiere" type="xsd:string" />  
  </xsd:choice>  
</xsd:complexType>
```

Opérateur d'ensemble

- un nouveau type dont chacun des éléments doit apparaître une fois mais dans un ordre quelconque.

```
<xsd:complexType name="filiere_type">  
  <xsd:all>  
    <xsd:element name="nom" type="xsd:string" />  
    <xsd:element name="descriptif" type="xsd:string" />  
  </xsd:all>  
</xsd:complexType>
```


Exemple: etudiants.xml

- Question 3: définir dans le schéma XML «etudiants.xsd» le type étudiant **etudiant_type** qui se compose de la séquence suivante:

<nom>Maaroufi</nom>

<prenom>Manal</prenom>

<telephone>0662569856</telephone>

<email>M_maaroufi@gmail.com</email>

<adresse>N816 Residence Alexandrie, Avenue Mohamed5 B.P.20000 Fes, Maroc.</adresse>

<filier>Genie Industriel</filier>

<moyenne>15</moyenne>

Correction

```
<xsd:complexType name="etudiant_type">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string" />
    <xsd:element name="prenom" type="xsd:string" />
    <xsd:element name="telephone" type="xsd:string" />
    <xsd:element name="email" type="xsd:string" />
    <xsd:element name="adresse" type="xsd:string" />
    <xsd:element name="filiere" type="xsd:string" />
    <xsd:element name="moyenne" type="xsd:decimal" />
  </xsd:sequence>
</xsd:complexType>
```

Exemple: etudiants.xml

- Question 4: définir dans le schéma XML «etudiants.xsd» l'élément racine **liste** qui se compose d'une séquence **d'étudiants**.
- Question 5: valider votre document XML (vérification lexicale et syntaxique)

Correction

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
<xsd:element name="liste">
```

```
<xsd:complexType>
```

```
<xsd:sequence>
```

```
<xsd:element name="etudiant" type="etudiant_type" maxOccurs="unbounded" />
```

```
</xsd:sequence>
```

```
</xsd:complexType>
```

Exemple: etudiants.xml

- Question 6: pour un étudiant donné, on doit indiquer soit son adresse, soit son email

Exemple: etudiants.xml

- Question 6: pour un étudiant donné, on doit indiquer soit son adresse, soit son email

```
<xsd:choice>
```

```
  <xsd:element name="email" type="xsd:string" />
```

```
  <xsd:element name="adresse" type="xsd:string" />
```

```
</xsd:choice>
```

Indicateurs d'occurences

- spécifier le nombre d'occurrences autorisées d'une balise (élément XML).

```
<xsd:element name="note" type="xsd:string" minOccurs="0"/>
```

```
<xsd:element name="note" type="xsd:string" minOccurs="0"  
maxOccurs="unbounded" />
```

Exemple: etudiants.xml

- Question 7: indiquer que le nom, prénom, filière et moyenne doivent figurer une et une seule fois dans l'arborescence étudiant,
- Question 8: la liste doit contenir au minimum un seul étudiant et au maximum N étudiants

Correction etudiant.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="liste">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="etudiant" type="etudiant_type" minOccurs="1" maxOccurs="unbounded" />

    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="etudiant_type">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="prenom" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="telephone" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="email" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="adresse" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="filiere" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="moyenne" type="xsd:decimal" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Valeur par défaut et valeur fixe

- Il est possible de donner une valeur par défaut ou une valeur fixe au contenu d'une balise XML mais à condition que le type de la balise soit simple.

```
<xsd:element name="title" type="xsd:string" default="Titre par défaut"/>
```

```
<xsd:element name="title" type="xsd:string" fixed="Titre fixe"/>
```

Exemple: etudiants.xml

- Question 9: Accorder à la filière la valeur fixe « Ginf2016 »
- Question 10: Accorder à l'email la valeur par défaut « @emi.ac.ma »

Exemple: etudiants.xml

- Question 9: Accorder à la filière la valeur par fixe « Ginf2016 »
- Question 10: Accorder à l'email la valeur par défaut « @emi.ac.ma »

```
<xsd:element name="nom" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="prenom" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="telephone" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="email" type="xsd:string" minOccurs="1" maxOccurs="1" default="@emi.ac.ma" />
<xsd:element name="adresse" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="filiere" type="xsd:string" minOccurs="1" maxOccurs="1" fixed="Ginf2016" />
<xsd:element name="moyenne" type="xsd:decimal" minOccurs="1" maxOccurs="1"/>
```

XSD: déclaration des attributs

- Pour déclarer un attribut `xsd:attribute`
- Exemple:

```
<xsd:attribute name="format" type="xsd:string" />
```

```
<xsd:attribute name="lang" type="xsd:string" use="required"/>
```

```
<xsd:attribute name="lang" type="xsd:string" use="optional"/>
```

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

L'attribut est déclaré après la séquence des éléments

Exemple: etudiants.xml

- Question 11: un étudiant est identifié par un code unique

Exemple: etudiants.xml

- Question 11: un étudiant est identifié par un code unique

```
<xsd:complexType name="etudiant_type">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="prenom" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="telephone" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="email" type="xsd:string" minOccurs="1" maxOccurs="1" default="@emi.a" />
    <xsd:element name="adresse" type="xsd:string" minOccurs="1" maxOccurs="1" />
    <xsd:element name="filieres" type="xsd:string" minOccurs="1" maxOccurs="1" fixed="Ginf20" />
    <xsd:element name="moyenne" type="xsd:decimal" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="code" type="xsd:string" use="required" />
</xsd:complexType>
</xsd:schema>
```

Ajouter un attribut au telephone

```
<xsd:element name="telephone" minOccurs="1" maxOccurs="2">  
  <xsd:complexType>  
    <xsd:simpleContent>  
      <xsd:extension base="xsd:string"> → 0537468051  
        <xsd:attribute name="type" type="xsd:string" /> → "Fixe"  
      </xsd:extension>  
    </xsd:simpleContent>  
  </xsd:complexType>  
</xsd:element>
```

```
<telephone type="Fixe">0537468051</telephone>
```

```
<telephone type="GSM">0652458565</telephone>
```


Les restrictions

- Restreindre les contenus en donnant une valeur minimale et/ou une valeur maximale avec les éléments `xsd:minInclusive`, `xsd:minExclusive`, `xsd:maxInclusive` et `xsd:maxExclusive`

Exemple de restriction

```
<xsd:element name="age">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:integer">  
      <xsd:minExclusive value="21" />  
      <xsd:maxExclusive value="25" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:element>
```

Exemple: etudiants.xml

- Question 12: la moyenne doit être entre 0 et 20

Moyenne entre 0 et 20

```
<xsd:element name="moyenne">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:decimal">  
      <xsd:minInclusive value="0" />  
      <xsd:maxInclusive value="20" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:element>
```

Restriction par énumération

- donner explicitement une liste des valeurs possibles d'un type donné

```
<xsd:element name="mention">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:enumeration value="AB" />  
      <xsd:enumeration value="B" />  
      <xsd:enumeration value="TB" />  
      <xsd:enumeration value="E" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:element>
```

Exemple: etudiants.xml

- Question 13: limiter la liste des filières à
 - ☐ Génie Civil
 - ☐ Génie Electrique
 - ☐ Génie Industriel
 - ☐ Génie Informatique
 - ☐ Génie Mécanique
 - ☐ Génie Minéral
 - ☐ Génie MIS
 - ☐ Génie des Procédés

Exemple: etudiants.xml

- Question 13: limiter la liste des filières

```
<xsd:element name="filiere">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Génie Civil" />
      <xsd:enumeration value="Génie Electrique" />
      <xsd:enumeration value="Génie Industriel" />
      <xsd:enumeration value="Génie Informatique" />
      <xsd:enumeration value="Génie Mécanique" />
      <xsd:enumeration value="Génie Minéral" />
      <xsd:enumeration value="Génie MIS" />
      <xsd:enumeration value="Génie des Procédés" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Expressions régulières: xsd:pattern

- Donner une expression régulière qui devra être vérifiée pour que la valeur soit acceptée.

```
<xsd:simpleType name="code">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[0-9]{10}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```


Exemple: etudiants.xml

- Question 14: ajouter au téléphone l'attribut type qui peut être soit fixe soit mobile

Exemple: etudiants.xml

- Question 14: ajouter au téléphone l'attribut type qui peut être soit fixe soit mobile

```
<xsd:element name="telephone" minOccurs="1" maxOccurs="2">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="type" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="fixe" />
              <xsd:enumeration value="mobile" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

D'autres opérateurs

- xsd:length
- xsd:minLength
- xsd:maxLength
- xsd:fractionDigits
- xsd:totalDigits

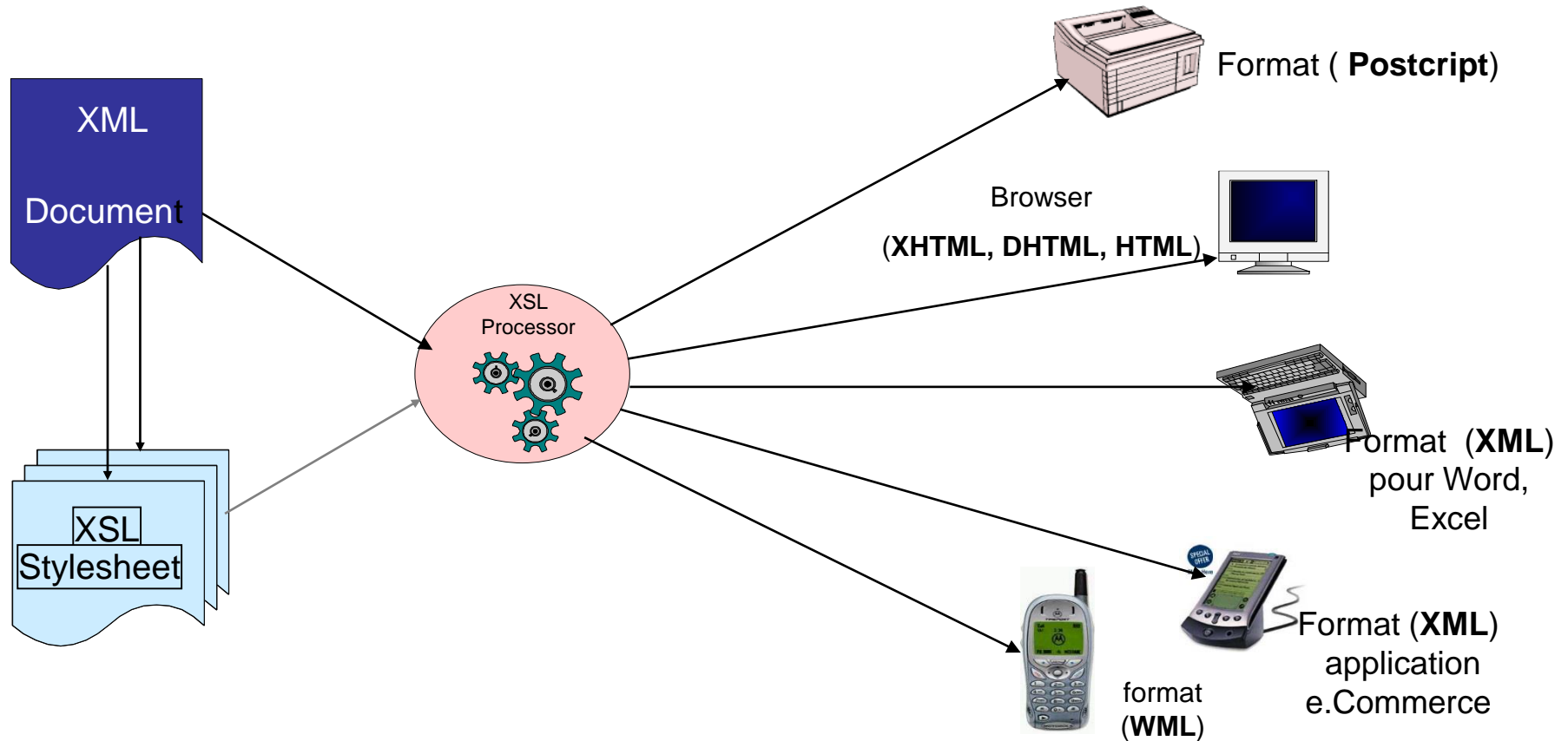
2. XSLT : la présentation

(eXtensible Stylesheet Language Transformation)

- Pour transformer un document
 - XML vers XML,
 - XML vers une présentation (HTML, texte, rtf, pdf, etc.)
- Un document xml est un arbre, XSL:
 - parcourt l'arbre
 - applique les règles de transformations vérifiées (à condition vraie) aux nœuds sélectionnés
 - produit un document en sortie

Publications avec XSL

- Plusieurs formats de publication pour un contenu



- XSL permet la présentation sur des terminaux variés¹⁰²

Les feuilles de style

- Une feuille de style XSL
 - est un document XML de racine `<xsl:stylesheet>`
 - contient une liste de règles de transformation `<xsl:template>`
- Chaque règle (`<xsl:template>`) précise:
 - Une condition spécifiant le sous-arbre du document d'entrée auquel elle s'applique (`match=`)
 - Une production spécifiant le résultat de l'application de la règle (`contenu`)
- Il s'agit de règles de production classiques
 - If `<condition>` then `<production>`
 - Codées en XML avec espace de nom `xsl:`

Exemple de document

```
<?xml version="1.0" ?>
  <Guide>
    <Restaurant Categorie="1">
      <Nom>Le Bahia</Nom>
      <Adresse>
        <Ville>Fès</Ville>
        <Dept>Fès-Boulemane</Dept>
      </Adresse>
    </Restaurant>
    <Restaurant Categorie="2">
      <Nom>Salè-Sucré</Nom>
      <Adresse>
        <Ville>Rabat</Ville>
        <Dept>Rabat-Salé</Dept>
      </Adresse>
    </Restaurant>
  </Guide>
```


Exemple de feuille de style XSL

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/xsl">
  <xsl:template match="/">
    <html>
      <head><B>Restaurant- XSL</B></head>
      <body><xsl:apply-templates/></body>
    </html>
  </xsl:template>
  <xsl:template match="Guide">
    <H1>BONJOUR EXEMPLE XML</H1>
    <H2>VOICI LE GUIDE</H2>
  </xsl:template>
  <xsl:template match="Restaurant">
    <P>      <l>Restaurant :</l>
      <xsl:value-of select="Nom"/></P>
  </xsl:template>
</xsl:stylesheet>
```

Règles de production

- Attributs
 - **match**: condition de sélection des nœuds sur lesquels la règle s'applique.
 - **name**: nom de la règle, pour invocation explicite (en conjonction avec <apply-template>)
 - **mode**: permet d'appliquer à un même élément des règles différentes en fonction du contexte
 - **priority**: priorité, utilisé en cas de conflit entre deux règles ayant la même condition
- Exemples
 - <xsl: template match="/">
 - <xsl: template match="auteur" name= " Hugo" priority="1">
 - <xsl: template match="auteur" name= " Hugo" priority="2">
 - <xsl: template match="auteur" name= "Hugo " mode="affichage1">
 - <xsl: template match="auteur" name= "Hugo " mode="affichage2">
 - <xsl:apply-template name="Hugo" mode="affichage1">

La génération du résultat

- Le contenu de l'élément `<xsl:template>` est la production de la règle:
 - Les éléments du namespace xsl sont des instructions qui copient des données du document source dans le résultat
 - Les autres éléments sont inclus tels quels dans le résultat
- Instructions pour:
 - Parcourir l'arbre du document source
 - Copier le contenu du document source dans le résultat
- Parcours de l'arbre:
 - `<xsl:apply-templates>`, `<xsl:for-each>`
- Copie du contenu du nœud sélectionné:
 - `<xsl:value-of select= ... >`

Résumé des commandes

<xsl:template>, définir une règle et son contexte

<xsl:apply-templates />, appliquer les transformations aux enfants du nœud courant

<xsl:value-of select ... />, extrait la valeur d'un élément sélectionné à partir du nœud courant

<xsl:for-each>, définir un traitement itératif

<xsl:element>, générer un élément

<xsl:attribute>, générer un attribut

<xsl:if>, définir un traitement conditionnel

XSLT

- XSLT un langage puissant de transformation d'un arbre XML en un autre.
- XSLT permet en particulier de publier des données XML sur le Web par transformation en un document HTML standard
- XSLT est très utilisé :
 - Pour publier des contenus XML
 - Pour transformer des formats (EAI, B2B)

3. XSL-FO : le formatage

XSL-FO : le formatage

- Permet des mises en pages sophistiquées
- Objets de mise en forme applicables aux résultats avec XSLT
- Distinction
 - Formatage des pages
 - Formatage des objets à l'intérieur des pages
 - Statiques
 - Dynamiques

CEV* (Customer Economic Value) Comparison

International Model: 4300 SBA LP 4X2 Competitor Model: Freightliner FL700

Application: Dry Van - 5 years/25,000miles/yr
(Applies to other applications with similar mileage)

Expected Differential Savings for International Trucks and Tractors versus Competition

VALUE CATEGORY	ISB 5.9L	ISC 6.3L	MBE 900 4.3L	MBE 900 6.4L
Resale value	\$ 800 - 1,200	\$ 800 - 1,200	\$ 1,600 - 2,400	\$ 800 - 1,200
Engine overhaul cost	Not Available	Not Available	Not Available	Not Available
Preventive maintenance costs	\$ 167 - 250	\$ 288 - 432	\$ (87) - (130)	\$ 27 - 41
Repairability	Not Available	Not Available	Not Available	Not Available
TOTAL (see how we save)	\$ 967 - 1,400	\$ 1,088 - 1,632	\$ 1,513 - 2,370	\$ 827 - 1,281

(Figures in stars are not differences in expected resale value or value of International vs. competitor)

VALUE NOTES

RESELLER VALUE Consistent used vehicle pricing

ENGINE OVERHAUL COST Regular cost savings for engine rebuilds

PREVENTIVE MAINTENANCE COSTS Lower preventive maintenance costs due to longer service intervals

REPAIRABILITY Lower repair costs resulting from replacing individual parts instead of the entire section

ADDITIONAL VALUE POINTS

VISIBILITY (PRODUCTIVITY) It's easier to see the gauges on our raised instrument panel. Improved visibility features like the one help drivers keep their eyes on the road.

VISIBILITY (PRODUCTIVITY) 1400 sq. in. hood package combines a 520 sq. in. radiator side-by-side with 475 sq. in. charge air cooler. It's highly efficient, light-weight, low profile design that advances forward visibility.

VISIBILITY (PRODUCTIVITY) With up to 2074 sq. in. of front windshield glass area, and 540 sq. in. of available glass area to each side the driver has a commanding view of the road. The repositioned A-pillars increase overall road view. A swept back angle helps deflect debris, thus minimizing glass damage. Standard tinted glass helps reduce glare.

RESALE VALUE (LIFECYCLE COST) International's galvanneal steel cabs are constructed with welded-in reinforcements, a deep-ribbed back panel, and a single piece vinyl door frame. It's built to withstand the toughest conditions—keeping you out of the shop and on the job.

RESALE VALUE (LIFECYCLE COST) Standard rear cab air suspension minimizes cab vibration, extending cab, and cab-mounted component life and delivering an exceptional ride, resulting in lower driver turnover and associated costs.

RESALE VALUE (LIFECYCLE COST) Our redesigned all-aluminum cab needs corrosion, and it weighs less. A lighter truck means you can increase your payload and your profits.

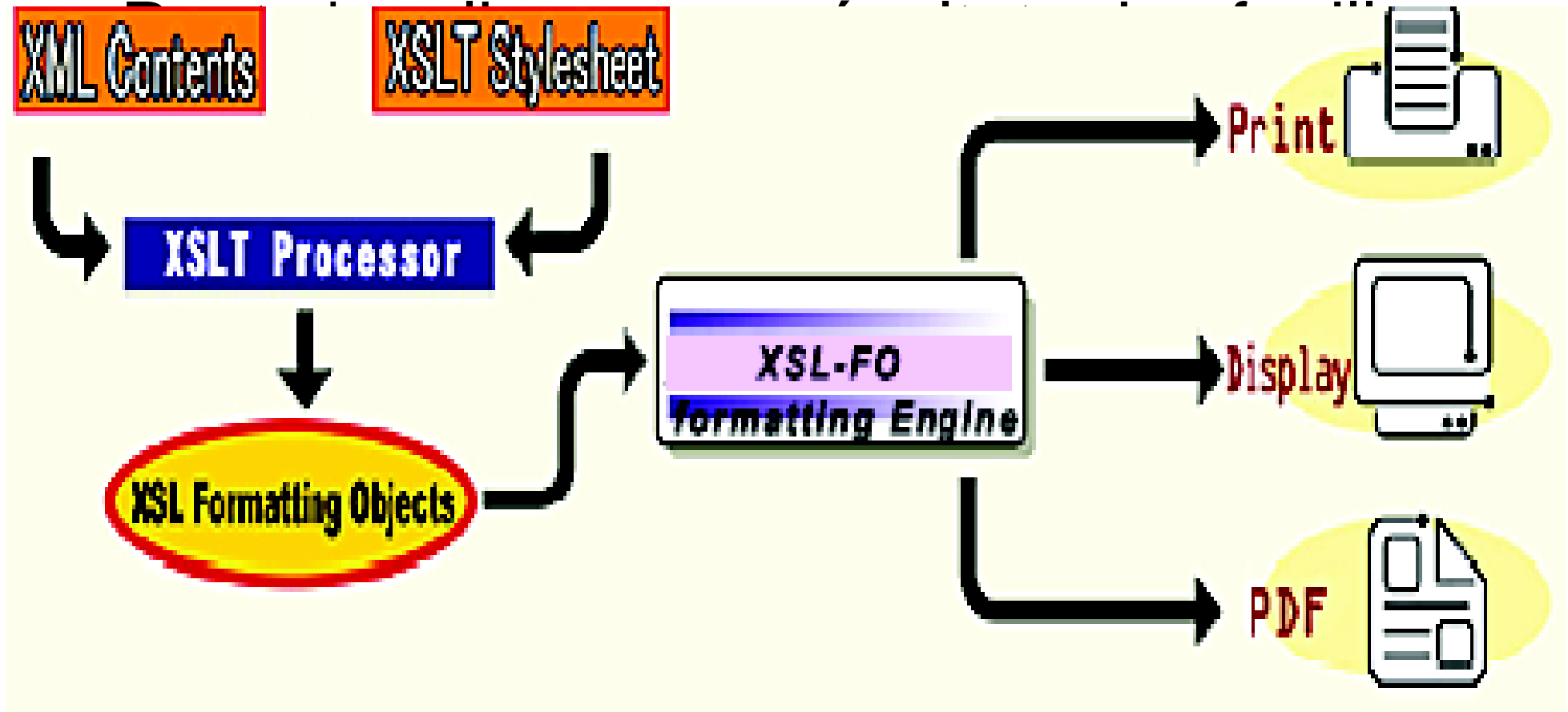
RESALE VALUE (LIFECYCLE COST) Polished aluminum wheels can increase resale value, improve image, and reduce chassis weight. Low A-tilt aluminum hubs to reduce chassis weight up to 100 pounds on the front axle.

BODY AND CHASSIS INTEGRATION (OPERATING EFFICIENCY) Whether you're in the construction business, hauling garbage or plowing snow, the International 1000 Series has been designed to render the clean CA necessary to mount bodies and equipment with ease.

BODY AND CHASSIS INTEGRATION (OPERATING EFFICIENCY) Front frame extensions with integral reinforcements provide the strength required to mount plus 100lbs. in combination with front PTO driven pumps and other heavy equipment.

CEV* (Customer Economic Value) Comparisons are intended for illustrative purposes only. Individual results will vary based on local market specific applications, trade cycle and miles per year.

Principles



Organisation du document

- Un document FO est formé d'un élément `fo:root` qui comprend deux parties distinctes
 - une description des modèles de pages
 - `fo:layout-master-set`
 - une description du contenu
 - `fo:page-sequence`
- Le contenu comporte :
 - Des flux contenant les données mêmes du document
 - Des éléments statiques dont le contenu se

Objets de formatage

- Les objets de formatage sont multiples :
 - `<fo:block>`
 - utilisé pour les blocs de textes, paragraphes, titres...
 - `<fo:display-rule>`
 - ligne de séparation
 - `<fo:external-graphic>`
 - zone rectangulaire contenant un graphisme (SVG)
- Ils possèdent de nombreuses propriétés
 - Pour un block on peut définir

Fonctionnalités

- Pages portrait ou paysage
- Pages recto-verso
- Page de tailles variées
- Marges multiples
- Colonnes multiples
- Entête et pieds de page
- Caractères unicode
- Multiple directions d'écritures
- Numérotation des pages
- Graphiques et SVG
- Tables, avec entêtes, lignes et colonnes fusionnables
- Listes
- Zones flottantes
- Tris à l'édition

XSL-FO: hello World

```
<?xml version="1.0"
  encoding="iso-8859-1"?>
<fo:root
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master
      master-name="my-page">
      <fo:region-body margin="2
cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-
reference="my-page">
    <fo:flow flow-name="xsl-region-
body">
      <fo:block>Hello
```

- **Element Root**
 - Permet de définir le namespace XSL-FO
- **Layout master set**
 - Permet de déclarer une ou plusieurs page masters (masque) et page sequence masters pour définir la structure des pages (ici une de 2 cm de marges)
- **Page sequence**
 - Les pages sont groupées en séquences et structurées selon la référence au masque.
- **Flow**
 - C'est le container du texte utilisateur dans le document. 117
Le nom du flot lit le texte à une zone de la page définie dans le

XSL-FO et XSLT : Exemple

- Définition de propriétés pour la racine

- **<xsl:template match='/>**

- <fo-display-sequence**

- font-style='italic'**

- start-**

- indent='4pt'**

- end-indent='4pt'**

- font-size='18pt'**

- <xsl:apply-templates/>**

- </fo-display-sequence>**

- </xsl:template**

Les processeurs XSL-FO

- Apache Group : FOP
 - Formating Object Processor
 - Génère du PDF <http://www.apache.org>
- JFOR (Open Source)
 - Génère du RTF <http://www.jfor.org>
- Antenna House
 - XSL Formatter <http://www.antennahouse.com>
- RenderX
 - Génère du PDF <http://www.renderx.com>
- Altova
 - StyleVision http://www.altova.com/products_xsl.html
- XML Mind FO Converter
 - Génère du RTF <http://www.xmlmind.com/foconverter>

XSL Conclusions

- XML = format pour la production de publications échangeables sur le web :
 - production d'une source unique en XML ;
 - XSLT = génération automatique de présentations multiples ;
 - XSL-FO = génération de présentations soignées avec pages maîtres et blocs formatés.

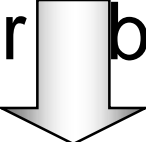
4. Autres spécifications

- XLink = liens externes entre documents
- XPointer = pointeurs internes à un document
- XForms = saisie de formulaires avec XML
- RSS = format pour la syndication de contenus

Les Liens XML

XLink et XPointer

Les Liens dans XML

- Les Liens HTML ont des lacunes
 - pas de saut vers une partie de document sans ancrage nommé
 - + difficile si fichier  ble non accessible en écriture
 - pas d'historique (lien source, ...), liens unidirectionnel
- XLL, eXtensible Linking Language : 2 parties
 - X-Links : liens vers un URI de document

Les Liens dans XML :

Xlink

- Tout élément XML peut devenir lien
 - en ajoutant les attributs
 - xmlns:xlink="http://www.w3.org/1999/xlink"
 - xlink:type type du lien
 - xlink:href adresse URI de la ressource cible
- Exemples de balises XML avec Xlink
 - <FOOTNOTE xlink:type="simple" xlink:href="footnote4.xml">
 - <IMAGE xlink:type="simple" xlink:href="logo.gif">
- Possibilité d'utilisation comme la balise A dans HTML
 - <xlink:type="simple" xlink:href="footnote4.xml">
 <FOOTNOTE> copyright </FOOTNOTE>
 </xlink:simple>
- Plusieurs types de liens
 - simple (id HTML),
 - extended : multi-directionnel (plusieurs sources, plusieurs cibles)

Les Liens dans XML : Xpointer

- Positionnement vers des endroits d'un autre document
 - pas de besoin d'ancrage
 - souvent utilisé avec ID
 - syntaxe complète ou abrégé (avec ID)
ex : accès à l'élément dans seminet ayant comme id "semXml"
`http://emi.ac.ma/emi_p_o/seminet.xml#xpointer(id(semXml))`
en abrégé :
`http://emi.ac.ma/emi_p_o/seminet.xml#semXml`
- Utilisation d'une syntaxe Xpath étendu

Utilisation de Xlien/Xpath

- Avec Id
 - `xlink:href='http://foo.bar.com#xpointer(id("D231"))'`
- Sans Id
 - `Xlink:href="xpointer(/child::chapitre[position()=3]/ child::section[position()=2]/ descendant::figure[position()=1]/child::legend)"`
- Syntaxe de la séquence des enfants
`http://emi.ac.ma/emi_p_o/seminet.xml#/3/2/1`

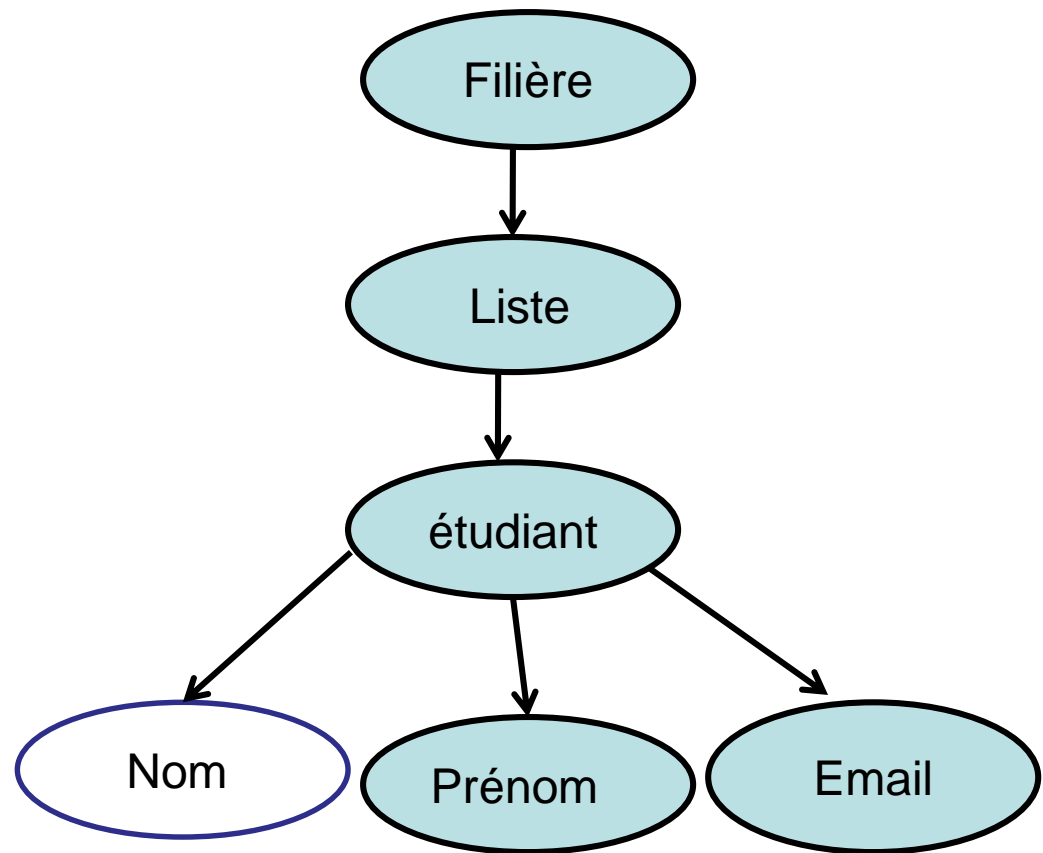
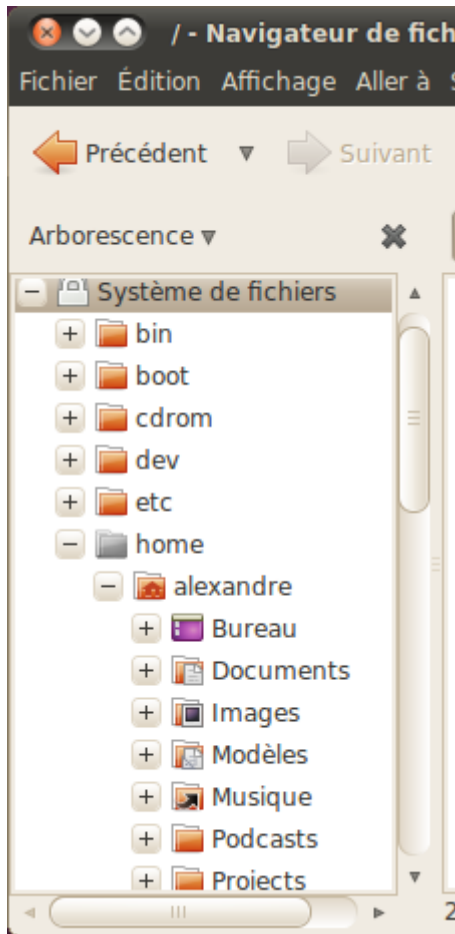
xpath

- C'est un langage de requêtes
- Le langage XPATH offre un moyen d'identifier un ensemble de nœuds dans un document XML.
- Le résultat d'une expression XPath peut être :
un ensemble de nœuds (ou un nœud seul)
Un contenu

xpath

- XPath permet d'effectuer des requêtes dans un document XML. XPath est aux documents XML, ce qu'est SQL aux bases de données.
- XPath est utilisé par XSLT pour faire de la transformation de documents XML.

Expressions de chemins



Expression XPath qui retourne 1 nœud **Filière/Liste/étudiant/Nom**

Exemples

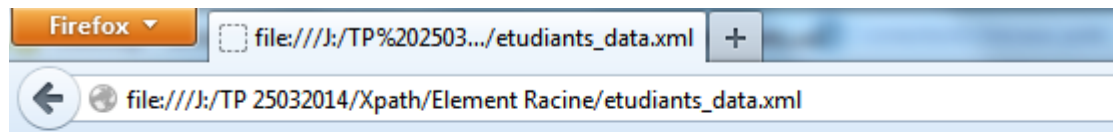
- Sélectionner l'élément racine de votre document XML

```
<xsl:template match="/">  ← <liste> </liste>
  <xsl:value-of select="." />
</xsl:template>
```

Maaroufi
Manal
M_maaroufi@gmail.com
N816 Residence alexandrie, Avenue Mohamed5 B.P.20000 Fes, Maroc.

Count()

- le nombre d'étudiants inscrits dans le document XML ;



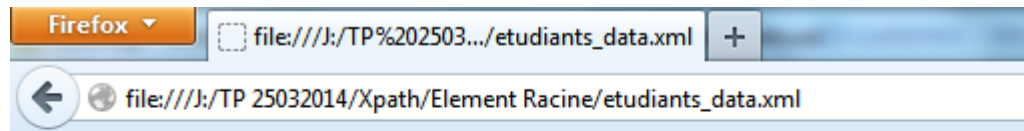
le nombre des étudiants inscrits est: 19

```
<xsl:template match="/">
<html>
<h1> le nombre des étudiants inscrits est:
<xsl:value-of select="count(//etudiant)"/>
</h1>
</html>
</xsl:template>
```

Position

- Afficher le nom de l'étudiant numéro 2 de la liste.

```
<xsl:value-of select="//etudiant[position()=2]/nom />  
<xsl:value-of select="//etudiant[2]/nom"/>
```



l'étudiant numéro 2 est: Maaroufi

Contenu des attributs

- Afficher uniquement le GSM de l'étudiant 19
- `<xsl:value-of
select="//etudiant[position()=19]/telephone[@type='GSM`