XML: eXtensible Markup Language

Fondements,

Modélisation,

Présentation et

Programmation

Sommaire

- Fondements de la technologie XML
- Modélisation et validation de documents XML
- Présentation des documents XML
- Programmation en utilisant XML

XML: eXtensible Markup Language

Partie 1:

Fondements et notions de base

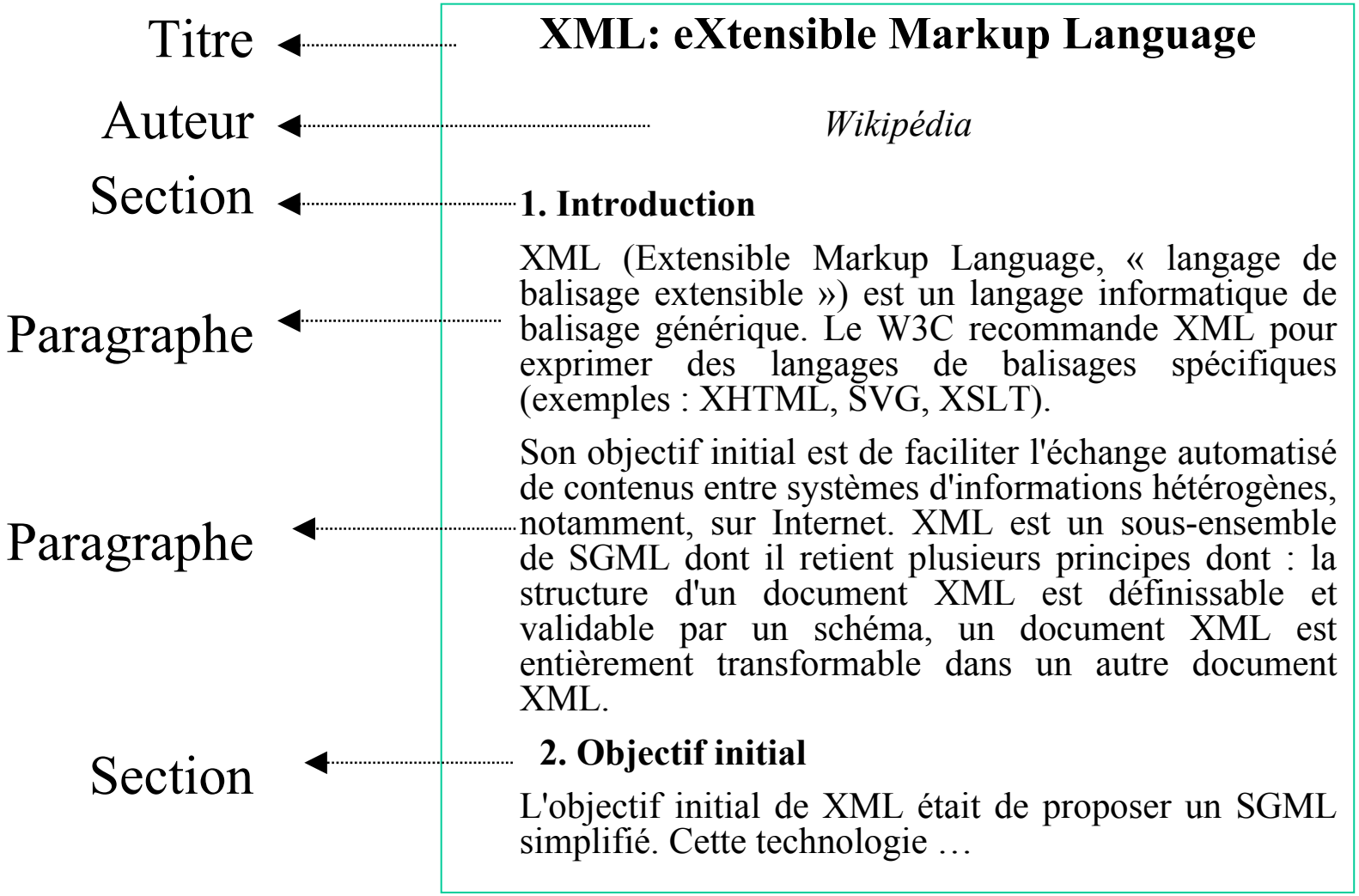
Notion de document électronique

- Définitions
  - Objet qui joue un rôle de médiateur entre les hommes dans le temps et l'espace (échange d'information)
  - Moyen de communication de la pensée, de la connaissance, de l'information et du savoir
  - Un document est un « ensemble formé par un support et une information » (ISO TC-46)
  - Se présente sous la forme de textes, tableaux, dessins, photos, ...
  - Un document a un contenu (structure logique) et un fond (structure physique)

Modèles de document

- Structure logique
  - Décrit le contenu d'un document (information)
  - Par exemple: Chapitre, section, paragraphes, figures, notes…
- Structure physique
  - Décrit la forme et le support du document (formatage)
  - Par exemple: tomes, pages, cadres, pavés, fenêtres
- Caractéristiques communes des 2 structures:
  - Composées ou élémentaires (molécule / atomes)
  - Générique ou spécifique (date / le 29-11-2006)

Exemple d'un document (article)



Exemple d'un document: Structure logique

<Article>

<Titre> XML: eXtensible Markup Language </Titre>

<Auteur>Wikipédia</Auteur>

<Section titre = "Introduction">

<Paragraphe>XML (Extensible Markup Language, « langage de balisage extensible ») est un langage informatique de balisage générique. Le W3C recommande XML pour exprimer des langages de balisages spécifiques (exemples : XHTML, SVG, XSLT). … </Paragraphe>

<Paragraphe> Son objectif initial est de faciliter l'échange automatisé de contenus entre systèmes d'informations hétérogènes, notamment, sur Internet. XML est un sous-ensemble de SGML dont il retient plusieurs principes dont : la structure d'un document XML est définissable et validable par un schéma, un document XML est entièrement transformable dans un autre document XML. </Paragraphe>

</Section>

<Section titre= "Objectif initial"> …

</Section>

</article>

Exemple d'un document: Structure physique

```
<Article>
<Titre police="Times"  taille="24" position="centré" format="gras"/>
<Auteur police="Times"  taille="20" position="centré"
    format="italique"/>
<Section numero="1" police="Times"  taille="18"
     position="centré" format=" gras "/>
<Paragraphe police="Times"  taille="18" position="justifié"/>
</article>
```

World Wide Web Consortium

- W3C - Fondé en 1994
- Consortium industriel international accueilli par différents sites
  - MIT/LCS aux Etats-Unis
  - INRIA en Europe
  - Keio University au Japon
- 448 membres industriels en septembre 2000
- Accroitre le potentiel du WEB
  - Standards et Normes
  - Techniques, langages et architectures pour l'échange de documents sur le WEB

Langages de représentation de documents

- Par ordre chronologique:
  - SGML (Norme ISO 8879 en 1986, révisée en 1988 et 1994)
    - Méta-langage général
  - HTML (Standard W3C depuis 1989)
    - Structure générale figée
  - XML (Standard W3C depuis 1998)
    - Méta-langage simplifié compatible SGML

SGML: présentation

- Une norme internationale :
  - Standard Generalized Markup Language
  - ISO 8879 - 1989
- Un métalangage de balisage de documents
  - lisible par l'être humain et traitable par une machine
  - permet de définir des langages de balisage
- Les documents sont balisés conformément à la grammaire (la DTD)
  - instances de DTD
  - permet un balisage sémantique du fond.
- Implique la notion de validité d'un document

SGML : objectifs

- Séparation du fond et de la forme
  - possibilité de multiples présentations
  - un seul document en SGML
  - plusieurs formats : Postcript, HTML, etc.
- Support de traitements sur le contenu des documents sans prise en compte de la forme
- Proposition d'un cadre défini pour l'expression des modèles documentaires (validité, contrôle)
- Format de stockage et d'échange normalisé

SGML : critiques

- Très lourd et complexe pour la mise en œuvre de documents respectant ce format
- Une grande rigueur est demandée à l'entrée des documents
- Standard complexe et complet pour le traitement des documents
- Liens hypertextes possibles mais complexes

HTML : présentation

- Proposé par le W3C comme format de documents sur le Web
- Langage simple avec des balises standardisées permettant la mise en forme d'un texte.
- Standard reconnu par tous les navigateurs.
- Langage très populaire sur le Web

```
<HTML>
<HEAD>
<TITLE> Exemple </TITLE>
</HEAD>
<BODY>
<H1>Contenu du document</H1>
<A HREF = "http://www.server.fr/Info /dir/test.html"> une référence externe</A>
</BODY>
</HTML>
```

HTML : inconvénients

- Normalisation des différentes balises difficile
  - les constructeurs ont eu tendance à définir leurs propres balises pour répondre à leurs besoins (incompatibilité)
  - HTML est dédié pour un seul type de terminaux
- Mises à jour difficiles
  - restructuration ou remise en forme de l'ensemble des pages du site fastidieux
  - Incapacité d'extension sans "plugins" coté client (formules mathématiques, modélisations de molécules, scènes 3D…)
- Mélange de structures logique et physique
  - données utiles mélangée avec la mise en forme
  - Difficultés à trouver l'information recherchée

SGML et HTML : Résumé

- SGML
  - lange puissant pouvant décrire toute structure
  - Documents difficile à définir
  - Documents difficiles à utiliser
- HTML
  - spécialisation de SGML
  - adapté à la présentation
  - inadapté à l'échange entre programmes

XML : Présentation

- XML= un nouveau langage d'échange basé sur le balisage
- XML= plus simple que SGML
- XML= plus ouvert que HTML
- XML = développé par XML Working Group dirigé par le W3C (depuis 1996)
- XML 1.0 = recommandation officielle du W3C depuis le 10 février 1998

XML: objectifs fixés par le W3C (1)

- 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

XML: objectifs fixés par le W3C (2)

- Les documents XML doivent être lisibles par l'homme
- Les documents XML doivent être 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

Forces de XML

- Séparation de la structure et de la présentation
- Moins confus que HTML
- Plus simple que SGML
- Idéal pour l'échange de données semi-structurées
- Utilisable entre machines hétérogènes

XML: utilités (1)

- XML est un Méta-langage universel pour représenter les données échangées sur le Web qui permet au développeur de délivrer du contenu depuis les applications à d'autres applications ou aux navigateurs
- XML standardise la manière dont l'information est :
  - échangée
  - présentée
  - archivée
  - retrouvée
  - transformée
  - cryptée
  - ...

XML: utilités (2)

- Définir vos propres langages d'échange
  - Commande, facture, bordereau de livraison, etc.
- Modéliser des documents et des messages
  - Modèle logique de données
  - Eléments typés agrégés (DTD, XML Schema)
- Publier des informations
  - Neutre du point de vue format
  - Mise en forme avec des feuilles de style
- Archiver des données
  - Auto-description des archives (recherche d'information)

Concepts de XML

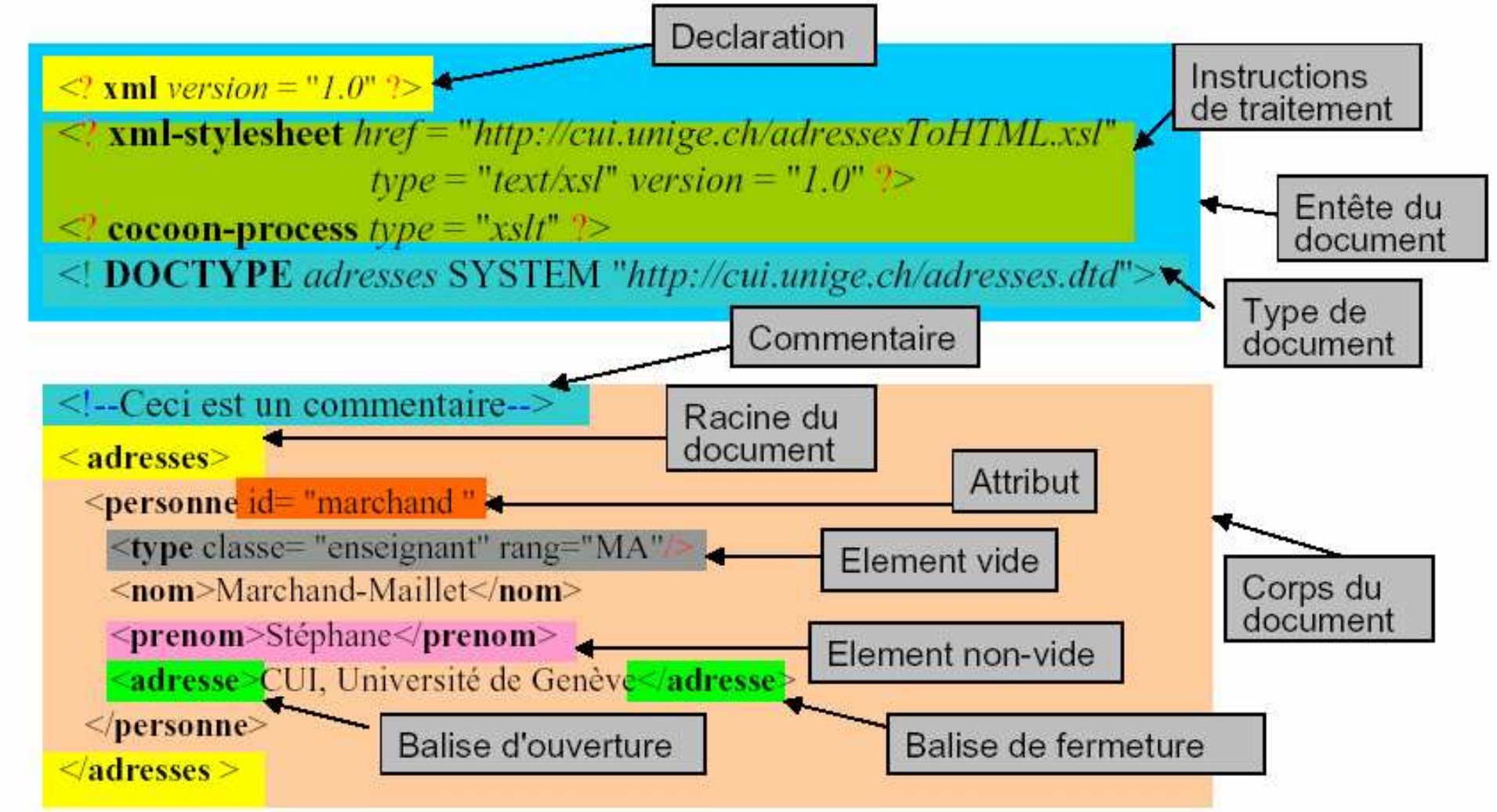
- **Balise (ou tag ou label)**
  - Marque de début et fin permettant de repérer un élément textuel
  - Forme: <balise> de début, </balise> de fin
- **Elément de données**
  - Texte encadré par une balise de début et une de fin
  - Les éléments de données peuvent être imbriquées

```
<producteur>
  <adresse>
      <rue>A. Einstein</rue>
      <ville>Villeurbanne</ville>
  </adresse>
</producteur>
```

- **Attribut**
  - Doublet nom="valeur" qualifiant une balise

```
<producteur no="160017" region="Rhône">
```

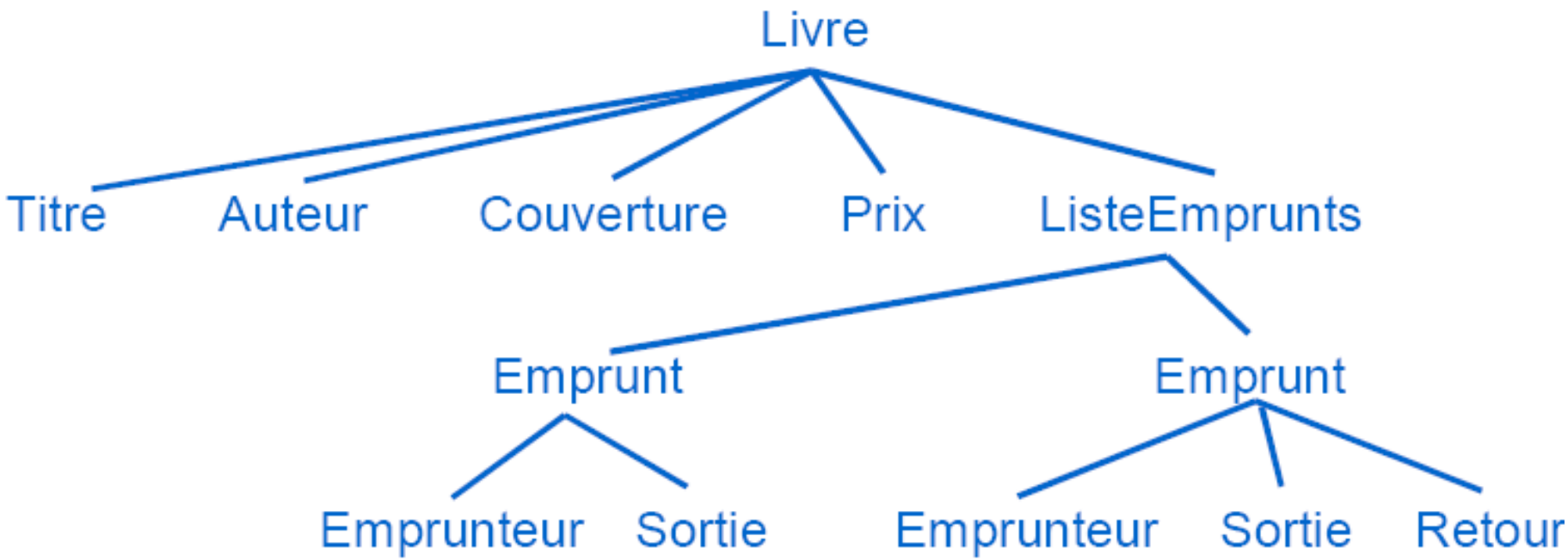
Exemple 1 d'un document XML



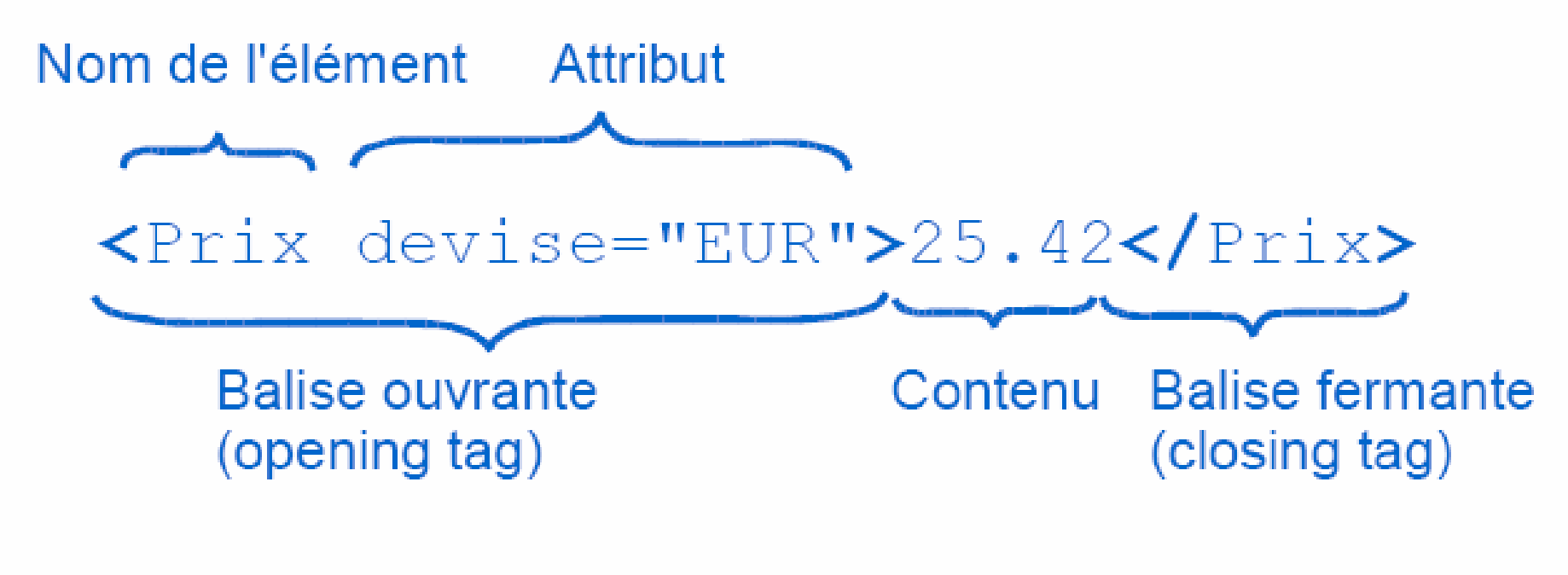
Exemple 2 d'un document XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="bib.css" ?>
<?cocoon-process type="xslt" ?>
<!DOCTYPE Livre SYSTEM "Livre.dtd">

<Livre>
  <Titre>Les réseaux</Titre>
  <Auteur>A. Tanenbaum</Auteur>
  <Couverture imgsrc="/imgs/res-tan.jpg"/>
  <Prix devise="EUR">25.42</Prix>
  <ListeEmprunts>
    <Emprunt>
      <Emprunteur>François Duchemin</Emprunteur>
      <Sortie>25/09/2000</Sortie>
      <Retour>02/10/2000</Retour>
    </Emprunt>
    <Emprunt>
      <Emprunteur>Hervé Delarue</Emprunteur>
      <Sortie>05/10/2000</Sortie>
    </Emprunt>
  </ListeEmprunts>
</Livre>
```

Structure Arborescente d'un document XML



La base d'un document XML: l'élément



- Quand utiliser les attributs?
  - Valeur unique de type simple (information monovaluée)
- Quand utiliser les éléments?
  - Valeur de type complexe (énumérations, possède des propriétés)

Espaces de noms

- Comment mixer des tags (ou balises) issus de différents langages?
- Un espace de noms est caractérisé par un préfixe

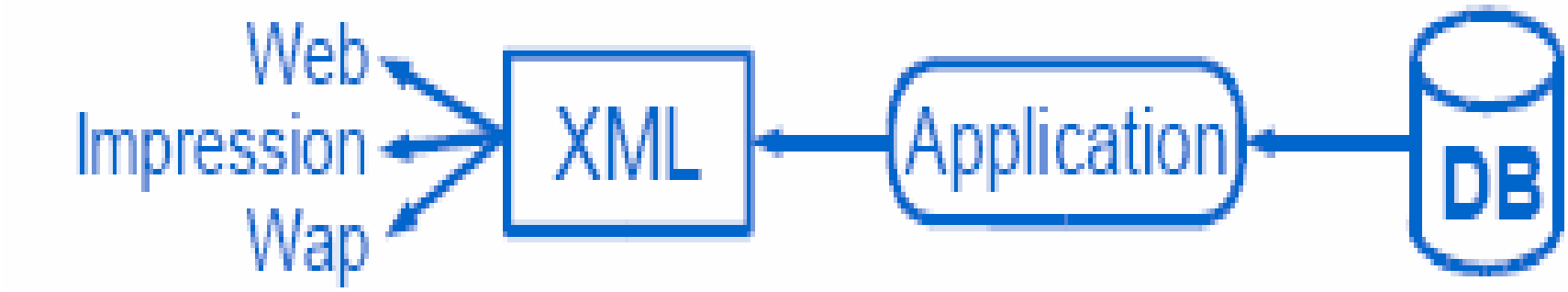
Le préfixe permet de retrouver le langage

```
<!-- définition des espaces de noms ->
<tag xmlns:mic="http://www.michelin.com/2001/Guide",
 xmlns:pj="http://www.pagejaune.com/2001/Annuaire">
<!– utilisation des espaces de noms ->
<mic:Adresse><ville>Villeurbanne</ville><zip>69100</zip></mic:Adresse>
<pj:Adresse>Villeurbanne 69 Rhône</pj:Adresse>
```

- Mécanisme intéressant pour l'intégration de contenus

XML: contextes d'utilisation

- Architectures N-tiers

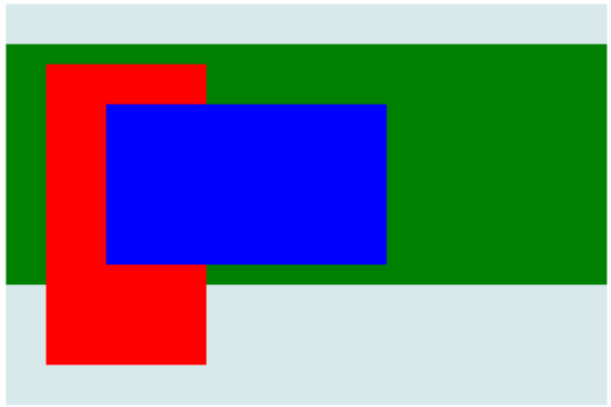


  - Production de documents XML à partir de services (couche métier)
  - Transformation de ces documents pour des formats d'affichage, d'impression, de transfert..

XML: contextes d'utilisation

- Stockage de données
  - Format spécifiques (exemple: SVG)

```
<?xml version="1.0" encoding="utf-8"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:ev="http://www.w3.org/2001/xml-events"
   version="1.1"
   baseProfile="full"
   x="0"
   y="0"
   width="300"
   height="200"
   id="svg2">
  <title>Rectangles</title>
  <defs
     id="defs4" />
  <g
     id="layer1">
    <rect
       width="300"
       height="120"
       x="0"
       y="20"
       fill="green"
       id="rect1306" />
    <rect
       width="80"
       height="150"
       x="20"
       y="30"
       fill="red"
       id="rect1308" />
    <rect
       width="140"
       height="80"
       x="50"
       y="50"
       fill="blue"
       id="rect1310" />
  </g>
</svg>
```



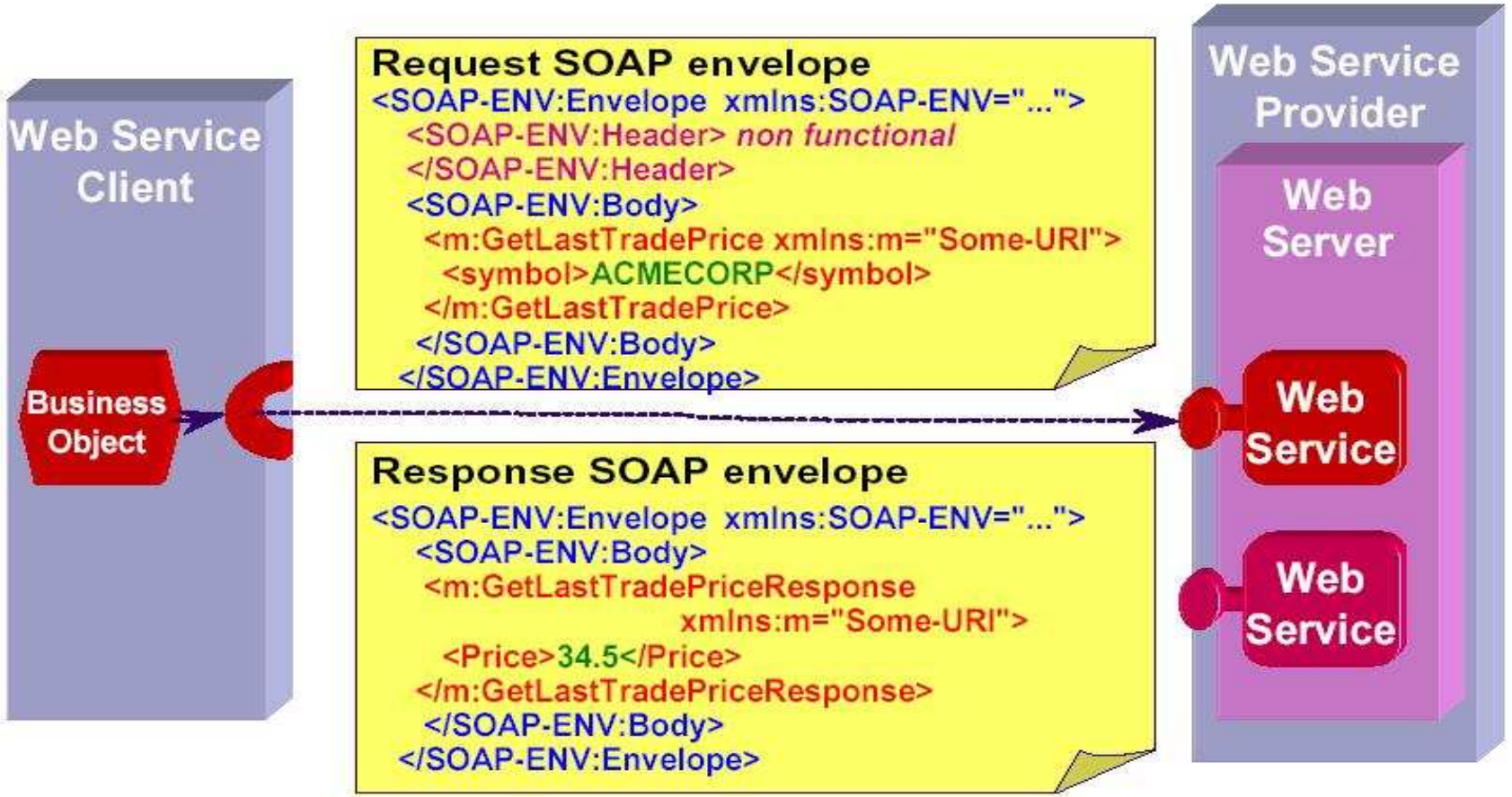
XML: contextes d'utilisation

- Stockage de données
  - Fichiers de configuration (exemple: JBOSS)

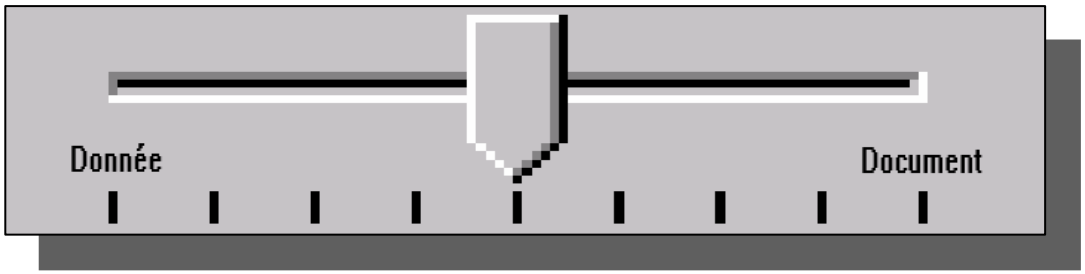
```
<?xml version="1.0" encoding = "ISO-8859-1"?>
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN'
'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>

<ejb-jar>
   <enterprise-beans>
     <session>
            <ejb-name>LegoBlocksEJB</ejb-name>
            <home>legoBean.LegoBlocksSessionHome</home>
            <remote>legoBean.LegoBlocksSession</remote>
            <ejb-class>legoBean.LegoBlocksSessionBean</ejb-class>
            <session-type>Stateful</session-type>
            <transaction-type>Container</transaction-type>
     </session>
   </enterprise-beans>
</ejb-jar>
```

XML: contextes d'utilisation

- Echange de données
  - Échanges Business to Business, services web…



Bases de données vs XML



- **Approche « Donnée »**
  - **Structuration forte et simple**
  - **Compatibilité SGBDR existants**
  - **Mise à jour en place**
  - **Intégrité sémantique**
  - **Indexation exacte**
  - **Adapté au transactionnel et décisionnel**
  - **Performances attendues « moyenne » à « forte » pour une volumétrie « moyenne »**
- **Approche « Document »**
  - **Structuration faible et complexe**
  - **Systèmes documentaires spécialisés**
  - **Gestion de versions**
  - **Recherche textuelle**
  - **Indexation approchée**
  - **Accès type moteur de recherche**
  - **Performances attendues « moyenne » pour une volumétrie « forte »**

Equivalences BD / XML

- Une table dans une BD peut correspondre à un document XML (elle correspond en réel à une classe d'objets)

  <class name="person">

- Une ligne dans la table décrit une instance d'un objet. Elle correspond à un élément XML

  <instance id="Person_001">

- Chaque colonne de la ligne correspond à une propriété de l'objet

  <property name="age">22</property>

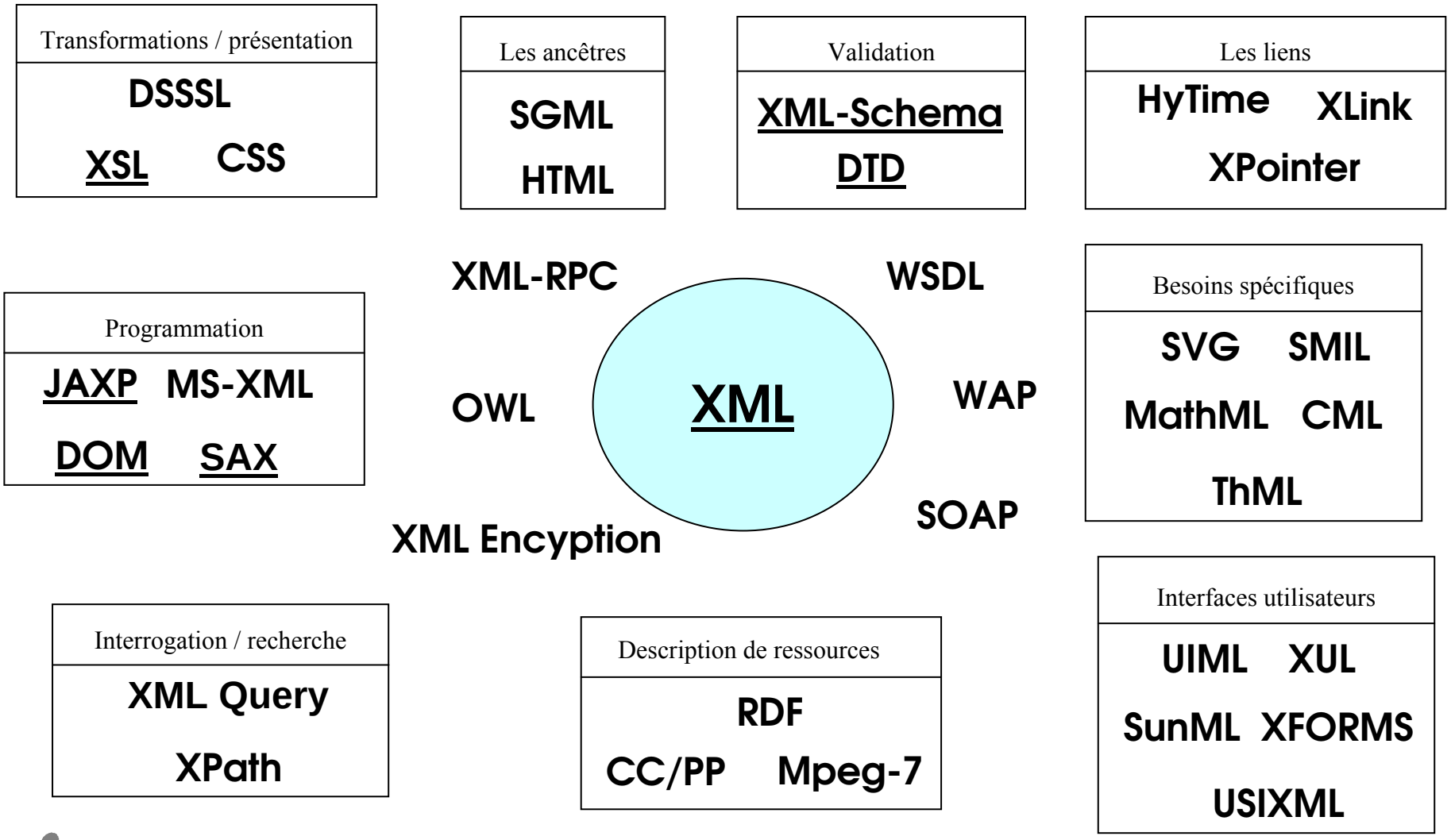
→Un autre contexte d'utilisation de XML:

- Mapping relationnel/objet
- Fusion de sources de données hétérogènes

L’état actuel de XML

- Un standard d'échange
  - Lisible : texte balisé avec marquage
  - Clair : séparation du fond et de la forme
  - Extensible : supporte les évolutions applicatives
  - Sécurisé : pare-feu, encryption, signature
- Développé par le W3C
  - Pour le Web (Internet, Intranet)
  - S'étend à l'entreprise et ses partenaires
- Supporté par les grands constructeurs
  - IBM, Microsoft .net, SUN, BEA, etc.
  - Des outils génériques et ouverts

Langages **XML** : une **nébuleuse** en expansion



XML: Exercice (structuration de données)

- **Donnez une version structurée du document suivant:**

Une bouteille d'eau Cristaline de 150 cl contient par litre 71 mg d'ions positifs calcium, et 5,5 mg d'ions positifs magnésium. On y trouve également des ions négatifs comme des chlorures à 20 mg par litre et des nitrates avec 1 mg par litre. Elle est recueillie à **St-Cyr la Source**, dans le département du Loiret. Son code barre est 3274080005003 et son pH est de 7,45. Comme la bouteille est sale, quelques autres matériaux comme du fer s'y trouvent en suspension.

Une seconde bouteille d'eau Cristaline a été, elle, recueillie à la source d'**Aurèle** dans les Alpes Maritimes. La concentration en ions calcium est de 98 mg/l, et en ions magnésium de 4 mg/l. Il y a 3,6 mg/l d'ions chlorure et 2 mg/l de nitrates, pour un pH de 7,4. Le code barre de cette bouteille de 50 cl est 3268840001008.

Une bouteille de même contenance est de marque Volvic, et a été puisée à... **Volvic**, bien connu pour ses sources donnant un pH neutre de 7. Elle comprend 11,5 mg/l d'ions calcium, 8,0 mg/l d'ions magnésium, 13,5 mg/l d'ions chlorures et 6,3 mg/l d'ions nitrates. Elle contient également des particules de silice. Son code barre est 3057640117008.

PS : Volvic est dans le Puy-de-Dôme...

XML: Exercice (structuration de données)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <eau>
- <bouteille>
  <marque>Cristaline</marque>
- <composition>
  <ion type="positif">calcium 71mg/l</ion>
  <ion type="positif">magnésium 5,5mg/l</ion>
  <ion type="negatif">chlorure 20mg/l</ion>
  <ion type="negatif">nitrate 1mg/l</ion>
  <autre type="metal">fer</autre>
  </composition>
- <source>
  <ville>St-Cyr la Source</ville>
  <departement>Loiret</departement>
  </source>
  <code_barre>3274080005003</code_barre>
  <contenance unit="cl">150</contenance>
  <ph>7,45</ph>
  </bouteille>
<bouteille>
 <marque>Cristaline</marque>
- <composition>
 <ion type="positif">calcium 98mg/l</ion>
 …
 </composition>
- - <source>
 <ville>Aurèle</ville>
 <departement>Alpes Maritimes</departement>
 </source>
 <code_barre>3268840001008</code_barre>
 <contenance unit="cl">50</contenance>
 <ph>7,4</ph>
 </bouteille>
- <bouteille>
 <marque>Volvic</marque>
- <composition>
 …
 </composition>
- <source>
 …
 </source>
  </bouteille>
 </eau>
```

XML: eXtensible Markup Language

Partie 2:

Modélisation des documents: DTD et Schéma

Introduction

- Un document XML peut être associé à :
  - une DTD ou un schéma pour décrire les balises
  - Une feuille de style pour présenter les données
- Une DTD ou/et un schéma permettent de définir son propre langage basé sur XML
  - Vocabulaire (balises)
  - Grammaire (imbrications)

  ➔ Dialecte ou Jargon

Validité des documents

- Document bien formé (Well Formed document)
  - Guillemets (ou apostrophes) obligatoires autour des valeurs

    <adresse id='2' type='domicile'/>
  - Les éléments vides utilisent une notation spécifique

    <image src='image3.gif'/> <image src='image3.gif'></image>
  - Les balises doivent être correctement imbriquées

    * NON ! *
  - Le document a une seule racine
  - Un attribut est unique dans son élément
- Document valide (Valid document)
  - bien formé + conforme à la DTD ou au schéma qui lui est associé

Exercice: documents bien formés?

• QUESTION: Est-ce que les documents suivants sont bien formés ?

```
<?xml version="1.0" encoding="iso-8859-1"?>
<titre>Ma vie</titre>
<sous-titre>par Christian Rémillard</sous-titre>
<para>Il était une fois...</para>
```

• REPONSE 1:

```
<comité>
  <nom>Comité pour la révision des métadonnées</nom>
  <membres nom="Louis Dantin" nom="Charles Gil" nom="E. L. Massicotte"/>
</comité>
```

• REPONSE 2:

```
<comité>
  <nom>Comité pour la révision des métadonnées</nom>
  <membres>
    <membre nom="Louis Dantin"/>
    <membre nom="Charles Gil"/>
    <membre nom="E. L. Massicotte"/>
  </membres>
</comité>
```

• REPONSE 3:

DTD

- Permet de définir le «vocabulaire» et la structure qui seront utilisés dans le document XML

- Grammaire du langage dont les phrases sont des documents XML (instances)

- Peut être mise dans un fichier (DTD externe) et être appelée dans le document XML

Déclaration d'élément simple

- <! ELEMENT *balise* (*définition*) >
  - Le paramètre *définition* représente soit un type de donnée prédéfini, soit un élément de données composé, constitué lui même d'éléments
  - Types prédéfinis
    - ANY : L'élément peut contenir tout type de donnée
    - EMPTY : L'élément ne contient pas de données spécifiques
    - #PCDATA : L'élément doit contenir une chaîne de caractère
  - Exemple
    - <! ELEMENT Nom (#PCDATA)>
    - <Nom>Victor Hugo</Nom>

Déclaration d'élément composé

- Définit une séquence ou un choix d'éléments
- Syntaxe spécifique avec opérateurs de composition d'éléments :

  <! ELEMENT balise (*composition*) >

Opérateur	Signification	Exemple
+	L'élément doit être présent au minimum une fois	A+
*	L'élément peut être présent plusieurs fois (ou aucune)	A*
?	L'élément peut être optionnellement présent	A?
|	L'élément A **ou** B peuvent être présents (pas les deux)	A|B
,	L'élément A doit être présent et suivi de l'élément B	A,B
()	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

Exemple d'élément composé

- *<!ELEMENT personne (nom, prenom+, tel?, adresse) >*

  <!ELEMENT nom (#PCDATA) >

  <!ELEMENT prenom (#PCDATA) >

  <!ELEMENT tel(#PCDATA) >

  <!ELEMENT email (#PCDATA) >

  <!ELEMENT adresse (ANY) >

- *<personne>*

  <nom>Hugo</nom>

  <prenom>Victor</prenom>

  <prenom>Charles</prenom>

  <tel>01120243</tel>

  <adresse><rue></rue><ville>Paris</ville></adresse>

  </personne>

Déclaration d'attributs

- <! ATTLIST *balise* *Attribut* *Type* *Mode* >
  - *balise* spécifie l'élément auquel est attaché l'attribut
  - *Attribut* est le nom de l'attribut déclaré
  - *Type* définit le type de donnée de l'attribut choisi parmi:
    - CDATA
      - Chaînes de caractères entre guillemets ("aa") non analysées
    - Enumération
      - Liste de valeurs séparées par |
      - <! ATTLIST *balise Attribut* (Valeur1 | Valeur2 | ...) >
    - ID et IDREF
      - Clé et référence à clé
  - *Mode* précise le caractère obligatoire ou non de l'attribut
    - #REQUIRED, #IMPLIED ou #FIXED

Exemple d'attributs

```
<! ATTLIST personne
            num ID,
            age CDATA,
            genre (Masculin | Feminin ) >

<!ELEMENT auteur (#PCDATA) >
<!ELEMENT editeur (#PCDATA) >
<!ATTLIST auteur
            genre (Masculin | Feminin ) #REQUIRED
            ville CDATA #IMPLIED>
<!ATTLIST editeur
             ville CDATA #FIXED "Paris">
```

Exemple de DTD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- DTD Livre : Livre.dtd -->
<!-- Caractérise un livre et son historique d'emprunts -->

<!ELEMENT Livre (Titre, Auteur, Couverture, Prix, ListeEmprunts)>
<!ELEMENT Titre (#PCDATA) >
<!ELEMENT Auteur (#PCDATA) >

<!ELEMENT Couverture EMPTY>
<!ATTLIST Couverture imgsrc CDATA #REQUIRED>

<!ELEMENT Prix (#PCDATA)>
<!ATTLIST Prix devise CDATA #REQUIRED>

<!ELEMENT ListeEmprunts (Emprunt*)>

<!ELEMENT Emprunt (Emprunteur,Sortie,Retour?)>
<!ELEMENT Emprunteur (#PCDATA)>
<!ELEMENT Sortie (#PCDATA)>
<!ELEMENT Retour (#PCDATA)>
```

Exemple de DTD interne

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT(PERSONNE*)>
<!ELEMENT PERSONNE (#PCDATA)>
<!ATTLIST PERSONNE PNUM ID #REQUIRED>
<!ATTLIST PERSONNE MERE IDREF #IMPLIED>
<!ATTLIST PERSONNE PERE  IDREF #IMPLIED> ]>

<DOCUMENT>
    <PERSONNE PNUM = "P1">Marie</PERSONNE>
    <PERSONNE PNUM = "P2">Jean</PERSONNE>
    <PERSONNE PNUM = "P3" MERE="P1" PERE="P2">Pierre</PERSONNE>
    <PERSONNE PNUM = "P4" MERE="P1" PERE="P2">Julie</PERSONNE>
</DOCUMENT>
```

DTD externe

- Modèle pour plusieurs documents
  - partage des balises, attributs et structures

- Définition locale ou externe

  <!DOCTYPE doc SYSTEM "doc.dtd">

  <!DOCTYPE doc PUBLIC "www.myweb.com/doc.dtd">

- Exemple de document

  <?xml version="1.0" standalone="no"?>

  <!DOCTYPE modeles SYSTEM "modeles.dtd">

DTD : Entité paramètre

- Permet la définition d'un groupe d'éléments sous un nom (macro) *<!ENTITY %nom "definition">*

- Réutilisable dans une DTD par simple appel *%nom;*

- Exemple :

  <!ENTITY %genres "(homme | femme)">

  <!ATTLIST auteur genre %genres; #REQUIRED>

- Peuvent être externes :

  <!ENTITY %mpeg SYSTEM "http://www.mpeg.com">

DTD : Entité générale

- Permet la définition d'un texte sous un nom

  <!ENTITY nom "texte">

- Réutilisable dans un document par simple appel *&nom;*
- Exemple

```
<?xml version="1.0"?>
<!DOCTYPE exemple [
    <!ELEMENT exemple (#PCDATA, important)>
    <!ELEMENT important (#PCDATA)>
    <!ENTITY cie "Les Vignerons Réunis">
    <!ENTITY imp "<important>Attention!</important>"> ]>
<exemple> &cie; &imp; </exemple>
```

Quelques règles d'écriture

- Modularité
  - définir dans des entités séparées les parties réutilisables
- Précédence
  - Regrouper les déclarations d'entités en tête
- Abstraction
  - Utiliser des entités pour les modèles de contenus
- Spécificité
  - Éviter les DTD trop générales
- Simplicité
  - Découper les DTD trop complexes

DTD: Exercice (Catalogue de films)

- On se propose de définir un format XML de stockage d'un catalogue de films sur DVD.
- Le catalogue comprend un ensemble de fiches de films
- Chaque fiche comprend:
  - Un numéro unique
  - Le titre du film
  - Un ou plusieurs réalisateurs
  - Un ou plusieurs éditeurs
  - Les acteurs principaux
  - Le genre du film (comédie, horreur, action…) (en option)
  - Un commentaire optionnel qui présente brièvement l'histoire du film
  - Un lien éventuel vers le site du film

DTD: Exercice (Catalogue de films)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--premier niveau: cataloque -->
<!ELEMENT cataloguedvd (fiche)*>
<!--deuxième niveau: fiche -->
<!ELEMENT fiche (titre, technique,commentaire?,internet?)>
<!ATTLIST fiche genre (Horreur|Action|comédie|inconnu) #IMPLIED
                  numero CDATA #REQUIRED>
<!--troisième niveau: sous éléments de 'fiche' -->
<!ELEMENT titre (#PCDATA)>
<!ELEMENT commentaire (#PCDATA)>
<!ELEMENT internet (#PCDATA)>
<!ELEMENT technique (realisateur+,editeur+,acteur*)>
<!--quatrième niveau: sous éléments de 'technique' -->
<!ELEMENT realisateur (#PCDATA)>
<!ELEMENT editeur (#PCDATA)>
<!ELEMENT acteur (#PCDATA)>
```

Insuffisance des DTD

- Pas de types de données à part du texte (#PCDATA)
- Expression de cardinalités limitée (‘?’, ‘*’ et ‘+’)
- Syntaxe spécifique (pas XML)
  - difficile à interpréter
  - difficile à traduire en schéma objets
- Propositions de compléments
  - XML-schema du W3C

XML Schéma

- Un schéma d'un document définit:
  - les éléments possibles dans le document
  - les attributs associés à ces éléments
  - la structure du document et les types de données
- Le schéma est spécifié en XML
  - pas de nouveau langage
  - balisage de déclaration
  - utilise un espace de nom xs: (ou xsd:)
- Présente de nombreux avantages
  - types de données personnalisés
  - extensibilité par héritage et ouverture
  - analysable par un parseur XML standard

Objectifs des schémas

- Reprendre les acquis des DTD
  - Plus riche et complet que les DTD
- Permettre de typer les données
  - Eléments simples et complexes
  - Attributs simples
- Permettre de définir des contraintes
  - Existence obligatoire ou optionnelle
  - Domaines de valeurs, cardinalités, références
  - Patterns, ...
- S'intégrer à la nébuleuse XML
  - Espace de noms
  - Structure d'arbre logique

Les types XML

- La base d'un schéma XML: l'élément

<xs:element name="..." type="..."/>

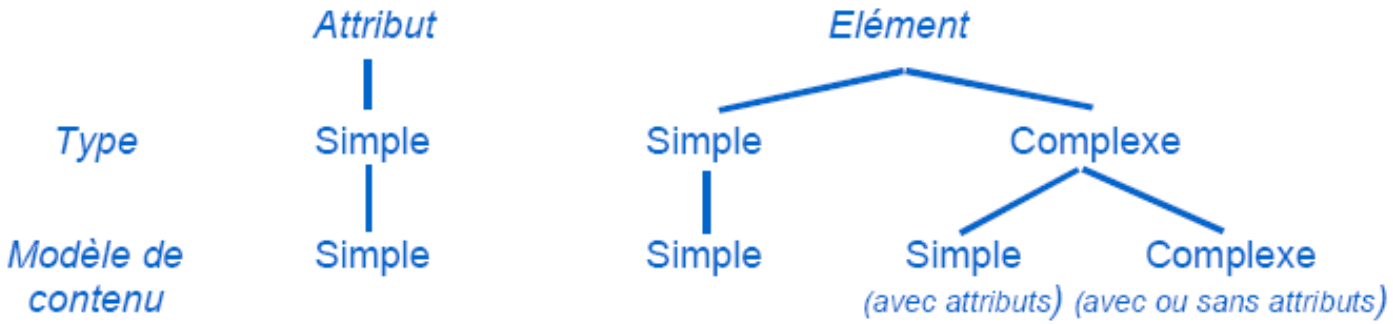
- Un élément peut avoir un type:
  - Simple si sa valeur a un type prédéfini en XML-SCHEMA (xs:string, xs:int, xs:decimal, xs:double…) ou une extension de ces types
  - Complexe s'il contient des sous éléments ou s'il comporte un attribut
    - xs:all tous les éléments doivent exister (peu importe l'ordre)
    - xs:choice un des éléments doit exister
    - xs:sequence tous les éléments doivent exister dans l'ordre spécifié

Modèles de contenu et types

- Quatre catégories de modèles de contenu

Contenu	**Vide**	**Simple**	**Complexe**	**Mixte**
Eléments	*non*	*non*	*oui*	*oui*
Texte	*non*	*oui*	*non*	*oui*

- Deux catégories de types de données: simple et complexe



Les types simples

- string
  Confirm this is electric
- byte
  -1, 126
- integer
  -126789, -1, 0, 1, 126789
- positiveInteger
  1, 126789
- negativeInteger
  -126789, -1
- hexBinary
  0FB7
- int
  -1, 126789675
- unsignedInt
  0, 1267896754
- boolean
  true, false 1, 0
- date
  1999-05-31
- ID
  "A212"
- IDREF
  "A212"
- IDREFS
  "A212" "B213"
- anyURI
  http://www.example.com/e1.html#5
- language
  en-GB, en-US, fr
- dateTime
  1999-05-31T13:20:00.000-05:00
- Et beaucoup d'autres
  Short, long, float

Les types complexes

- Définition d'objets complexes
  - <sequence> : collection ordonnée d'éléments typés
  - <all> : collection non ordonnée d'éléments typés
  - <choice>: choix entre éléments typés
- Exemple

```
<xs:complexType name="AddresseFR">
  <xs:sequence>
        <xs:element name="nom" type="xs:string"/>
        <xs:element name="rue" type="xs:string"/>
        <xs:element name="ville" type="xs:string"/>
        <xs:element name="codep" type="xs:decimal"/>
   </xs:sequence>
  <xs:attribute name="pays" type="xs:NMTOKEN" fixed="FR"/>
</xs:complexType>
```

Héritage de types

- Définition de sous-types par héritage
  - Par extension : ajout d'informations
  - Par restriction : ajout de contraintes
- Possibilité de contraindre la dérivation

- Exemple d'extension :

```
<xs:complexType name="AdressePays">
  <xs:complexContent>
    <xs:extension base="Adresse">
      <xs:sequence>
        <xs:element name="pays«  type="string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Les patterns

- Contraintes sur type simple prédéfini
- Utilisation d'expression régulières
  - Similaires à celles de Perl
- Exemple de restriction

```
<xs:simpleType name="num5">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{5}"/>
    </xs:restriction>
</xs:simpleType>
```

- **Autres facettes de restriction**

  - xs:mininclusive, xs:maxexclusive, xs:enumeration, xs:length…

Réutilisation de types

- **Type simple avec extension**

```
<xs:simpleType name="num5">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{5}"/>
    </xs:restriction>
</xs:simpleType>
```

- **Type complexe (séquence)**

```
<xs:element name="livre">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Titre" type="xs:string"/>
            <xs:element name="Auteur" type="xs:string"/>
            <xs:element name="ISBN" type="num5"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Les occurences

- Une bibliothèque contient au moins un livre

```
<xs:element name="biblio">
   <xs:complexType>
    <xs:sequence>
    <xs:element ref="livre" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
</xs:element>
```

Les attributs 1/2

- Les éléments à contenu simple avec attributs

```
<titre id="RF525">La bible de XML</titre>
```

```
<xs:complexType name="titleWithID">
   <xs:simpleContent>
    <xs:extension base="xs:string">
        <xs:attribute name="id" type="xs:ID"/>
    </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

Les attributs 2/2

- Les éléments à contenu complexe avec attributs

```
<traduction langue="allemand" dateTraduction="2003-12-01">
    <traducteur>Michael</traducteur>
</traduction>
<xs:element name="traduction" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
          <xs:element name="traducteur" type="xs:string" minOccurs="1"
    maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="langue" use="required" type="xs:string"/>
     <xs:attribute name="dateTraduction" use="optional" type="xs:date"/>
    </xs:complexType>
</xs:element>
```

Groupage d'éléments

```
<xs:group name="TitreAuteurISBN">
    <xs:sequence>
     <xs:element name="Titre" type="xs:string"/>
     <xs:element name="Auteur" type="xs:string"/>
     <xs:element name="ISBN" type="num5"/>
    </xs:sequence>
</xs:group>

<xs:element name="livre">
    <xs:complexType>
     <xs:sequence>
          <xs:group ref="TitreAuteurISBN"/>
          <xs:element ref="traduction"/>
     </xs:sequence>
    </xs:complexType>
</xs:element>
```

Référence à un Schéma XML

- Référence sans espace de noms

```
<Livre xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="Livre.xsd">
  <Titre>Les réseaux</Titre>
    ...
```

- Référence avec espace de noms

```
<bib:Livre xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:bib="http://www.cpe.fr/ns/bib"
           xsi:schemaLocation="http://www.cpe.fr/ns/bib Livre.xsd">
  <bib:Titre>Les réseaux</bib:Titre>
```

Bilan DTD et Schéma

- Les DTD définissent la grammaire des documents
- Elles sont de plus en plus souvent remplacées par des schémas
- Le standard XML-Schéma est un peu complexe
  - Une solution intermédiaire entre les DTD et les Schémas
  - http://www.relaxng.org
  - RELAX est en cours de standardisation ISO

XML: eXtensible Markup Language

Partie 3:

Présentation et transformations de documents

Formatage de document XML

- Un document et sa DTD ou schéma ne donnent pas d'indications sur sa représentation.
  - Une description supplémentaire est nécessaire:

DSSSL

Document Style Semantics and Specification language

Norme ISO
Liée à SGML
Dérivé de Scheme

```
(element para
  (make paragraph
    (process-children)))
(element emph
  (make sequence
    font-posture: 'italic
     (process-children)))
```

CSS

Cascading Style Sheets

Standard W³C

```
para {
  display: block
}
emph {
  display: inline;
  font-style: italic; }
```

XSL

eXtensible Stylesheet Language

Standard W³C
Lié à XML
Implémenté en XML

```
<xsl:template match="para">
  <fo:block>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="emph">
  <fo:sequence
        font-style="italic">
    <xsl:apply-templates/>
  </fo-sequence></xsl:template>
```

Cascading Style Sheets (CSS)

- Recommandation W3C (CSS1: 1996, CSS2: 1998)
- S'applique à HTML et XML
- Support approximatif par les navigateurs
  - Voir ***http://www.richinstyle.com/bugs/table.html***
  - Meilleurs support dans Mozilla et IE6
- Principes
  - Cascade
  - Correspondance d'éléments (sélecteurs)

CSS: Syntaxe

- Attachement d'une feuille de style à un document
  - <xml-stylesheet type="text/css" href="livre.css"?>
- Syntaxe générale
  - Sélecteur {propriété: valeur; propriété: valeur;…}
- Sélecteur
  - Voir ***http://www.w3.org/TR/REC-CSS2/selector.html***

el	Les éléments el
el1 el2	Les éléments el2 descendants de el1
el [att=foo]	Les éléments el contenant un attribut att de valeur foo
el:hover	Un élément el lorsqu'il est survolé par la souris
el:first-line	La première ligne d'un élément el

CSS: propriétés

- Mesure des longueurs
  - Mesure relatives: em, ex, %
    - font-size: 1.2em; (1.2 * la taille de l'élément parent)
    - line-height: 3ex; (3 * la taille d'une minuscule)
  - Mesures absolues: px, pt, mm, cm
- Couleurs
  - Prédéfinis: black, blue, green, maroon, yellow,…
  - Numériques: #rrvvbb, rgb (n, n, n)
- Exemple

  texte { font-size: 1.2em; line-height: 3ex; }

CSS: propriétés usuelles

Disposition	display	block inline list-item...
	position	relative absolute fixed
	top, right, bottom, left	
	float	left right none
Couleurs et fonds	color	
	background-color	*couleur* transparent
	background-image	
	background-repeat	no-repeat repeat repeat-x repeat-y
Polices	font-family	*police* sans-serif cursive ...
	font-size	*taille* x-small small medium large...
	font-weight	normal bold bolder lighter
	font-style	normal oblique italic
Texte	text-indent	*longueur* %
	text-align	left right center justify
	text-decoration	underline overline blink
Espacement	margin	
	padding	
Bordures	border	
	border-style	dotted dashed solid double...
Listes	list-style-type	
	list-style-image	

CSS: Exemple (extrait)

Propriété

Valeur

sélecteur

instruction

```
Livre { font-family: Helvetica }
Titre { display: block;
        font-size: 2em;
        text-align: center; }
Auteur { display: block;
         text-align: center;
         font-style: italic }
ListeEmprunts { display: block;
                counter-reset: emprunt; }

Emprunt { display: block;
          margin-top: 10pt;
          margin-bottom: 10pt;}
Emprunt:before { content: "Emprunt " counter(emprunt) " ";
                 counter-increment: emprunt; }

Emprunteur { display: block;
             margin-left: 10pt;
             font-weight: bold; }
Emprunteur:before { content: "Emprunteur: " }
```

Faiblesses de CSS

- CSS a été initialement prévu pou la présentation des documents HTML

→Même défauts que HTML

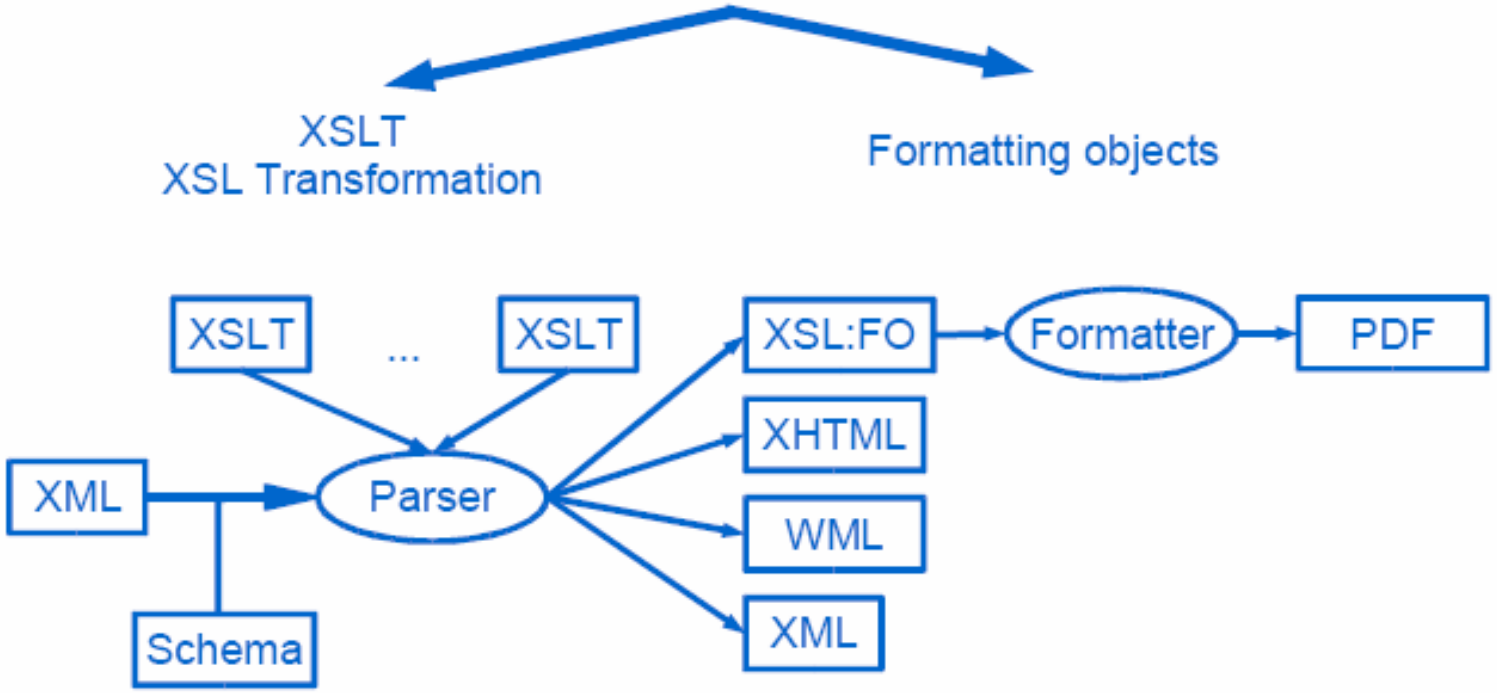
- Syntaxe non modifiable et non extensible
- Syntaxe difficile à normaliser
- Difficultés pour trouver des éléments

XPath: adressage dans un document XML

- Identification de nœuds dans un document XML
- Base du langage de transformation XSLT et des pointeurs XLink
- Syntaxe: chemin dans l'arborescence du document
  - sélecteur/sélecteur/sélecteur…
  - La syntaxe générale d'un sélecteur: **élément[prédicat]**
- Exemples:
  - **image [@imgsrc=joconde.jpg]**
  - **/livre/listeEmprunts/emprunt[2]/sortie**

XSL: eXtensible style sheet Language

- **Description du formatage à appliquer à un document XML**
- **Composé de deux sous-ensembles**



XSLT: XSL transformation

- **Langage de transformation d'un arbre XML dans un autre**
  - **Restructuration**
  - **Génération (indexes, tables, …)**
- **Recommandation W3C depuis 1999**
  - **http://www.w3.orgTR/1999/REC-xslt-19991116**
- **Transformation possible côté client ou côté serveur**
  - **Côté client**
    - **Support intégré dans IE (après mise à jour de MSXML dans IE5)**
    - **En utilisant un formateur XML externe via javascript (ActiveX)**
  - **Côté serveur**
    - **En invoquant un formateur par CGI, PHP ou ASP**
    - **En invoquant un formateur programmé (ex: Saxon, Xalan)**
    - **En utilisant un cadre de publication XML (ex: Cocoon)**

XSLT: principes généraux

- **Une feuille XSLT est un document XML**
  - **<xsl-stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform>**

    …

    </xsl-stylesheet>

- **Ensemble de règles**

  Sélection d'éléments et description de rendu à produire en sortie

Règles XSL: sélection

- **Syntaxe XPath par rapport au nœud courant**

- **Utilisation de l'élément** ***<xsl: template match=selecteur>***

  <xsl: template match="para">… </xsl: template>
  <xsl: template match="book[index] ">… </xsl: template>
  <xsl: template match="book[@title='Moby Dick'] "/chapter>…

- **Règles nommées**

  <xsl: call-template name="format-date">

  . . .

  <xsl:template name="format-date">…</xsl:template>

Règles XSL: production

- **Répétition**

```
<xsl: template match="ListeEmprunt">
        <table>
            <tr><th>Emprunteur</th>
            <th>sortie<th><th>Retour</th> </tr>
                <xsl:for-each select="emprunt">
                    <tr><td><xsl:value-of select="emprunteur"/></td>…</tr>
                </xsl:for-each>
        </table>
</xsl: template>
```

- **Conditions**

```
<xsl: if  test="condition">… </xsl: if >

 <xsl: choose>
                <xsl: when   test="condition" >… </xsl: when>
</xsl: choose>
```

Exemple XSLT(1/4):
Document XML d'origine

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="bib.css" ?>
<?cocoon-process type="xslt" ?>
<!DOCTYPE Livre SYSTEM "Livre.dtd">

<Livre>
  <Titre>Les réseaux</Titre>
  <Auteur>A. Tanenbaum</Auteur>
  <Couverture imgsrc="/imgs/res-tan.jpg"/>
  <Prix devise="EUR">25.42</Prix>
  <ListeEmprunts>
    <Emprunt>
      <Emprunteur>François Duchemin</Emprunteur>
      <Sortie>25/09/2000</Sortie>
      <Retour>02/10/2000</Retour>
    </Emprunt>
    <Emprunt>
      <Emprunteur>Hervé Delarue</Emprunteur>
      <Sortie>05/10/2000</Sortie>
    </Emprunt>
  </ListeEmprunts>
</Livre>
```

Exemple XSLT(2/4):
feuille de style…

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://http://www.w3.org/TR/REC-html40"
                version='1.0'>

<xsl:output method="html" encoding="ISO-8859-1"/>

<xsl:template match="/">
  <html>
    <head>
      <title>Fiche Emprunts</title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="Livre">
  <h1><xsl:value-of select="Titre"/></h1>
  <h2><xsl:value-of select="Auteur"/></h2>
  <xsl:apply-templates select="ListeEmprunts"/>
</xsl:template>
```

Exemple XSLT(2/4):
feuille de style (suite)

```
<xsl:template match="ListeEmprunts">
  <table>
    <tr>
      <th>Emprunteur</th>
      <th>Sortie</th>
      <th>Retour</th>
    </tr>
    <xsl:apply-templates select="Emprunt"/>
  </table>
</xsl:template>

<xsl:template match="Emprunt">
  <tr>
    <td><xsl:value-of select="Emprunteur"/></td>
    <td><xsl:value-of select="Sortie"/></td>
    <td><xsl:value-of select="Retour"/></td>
  </tr>
</xsl:template>

</xsl:stylesheet>
```

Exemple XSLT(2/4):
document produit

```
<html xmlns="http://http://www.w3.org/TR/REC-html40">
  <head>
    <title>Fiche Emprunts</title>
  </head>
  <body>
    <h1>Les réseaux</h1>
    <h2>A. Tanenbaum</h2>
    <table>
      <tr>
        <th>Emprunteur</th>
        <th>Sortie</th>
        <th>Retour</th>
      </tr>
      <tr>
        <td>François Duchemin</td>
        <td>2000-09-25</td>
        <td>2000-10-02</td>
      </tr>
      <tr>
        <td>Hervé Delarue</td>
        <td>2000-10-05</td>
        <td/>
      </tr>
    </table> </body> </html>
```

XML: eXtensible Markup Language

Partie 4:

Programmation en utilisant XML

XML et langages de programmation

- Comment résoudre le problème de la distance entre XML et les programmes objets?

- Les services sont souvent programmés en langage objet: Java, C++, C#, VB, PHP ...

- XML
  - les messages XML doivent être traduits en objets
  - analyseurs XML ou parseurs (terme technique utilisé)
  - différents niveau d'interfaces :
    - Flux d'événements (SAX)
    - Traduction en objet génériques (DOM)

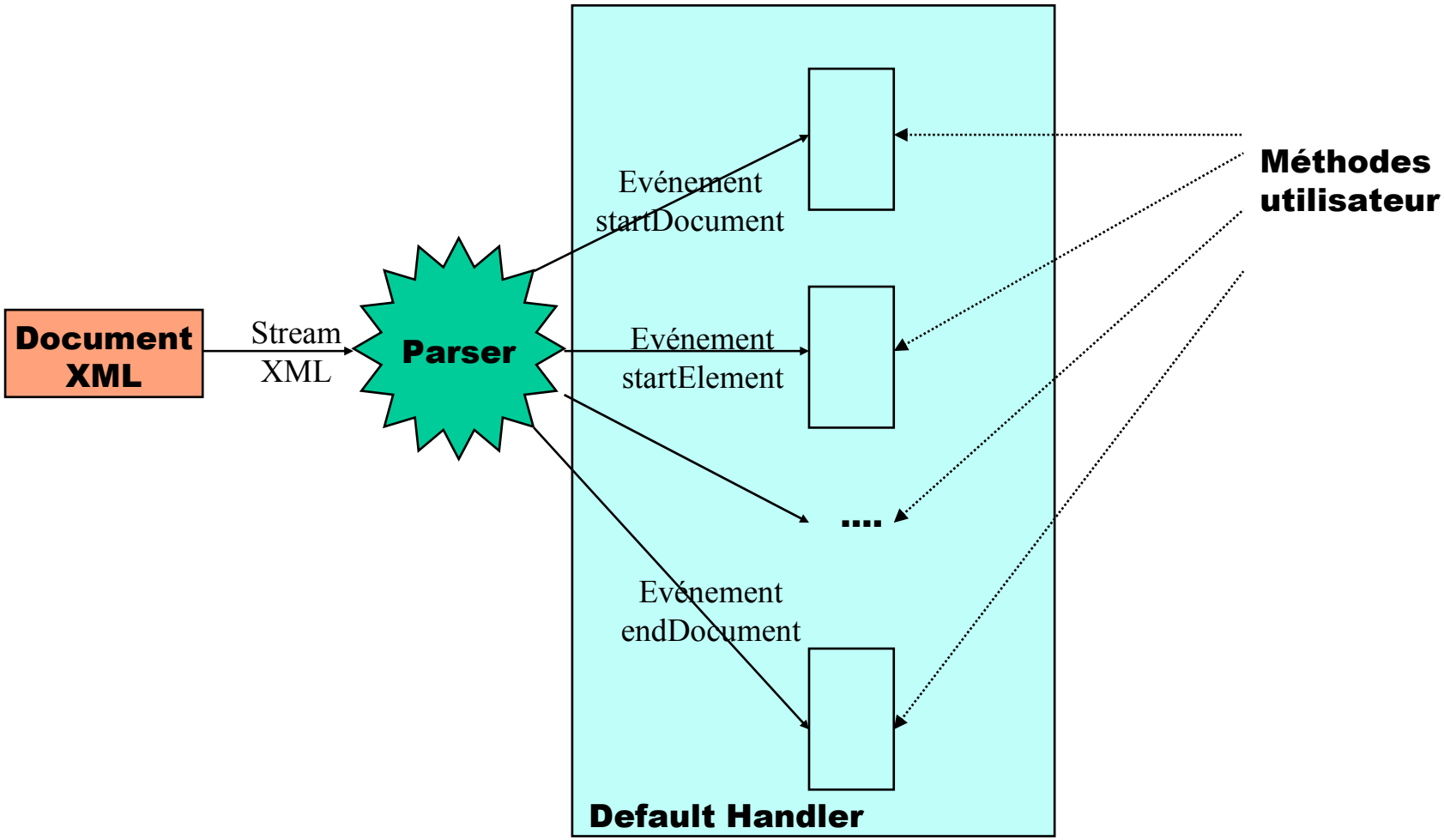
Les parseurs XML

- SAX (Simple API for XML)
  - Développement communautaire
  - http://sax.sourceforge.net
- DOM (Document Object Model)
  - Recommandation W3C
  - http://www.w3.org/DOM/
- JAXP (Java API for XML Processing)
  - Développement communautaire SUN
  - http://java.sun.com/xml/jaxp
  - Couche d'abstraction de SAX, DOM et XSLT indépendante de l'implémentation
  - XML et JAVA = un couple parfait ☺

SAX: Simple API for XML

- SAX (Simple API for XML)
  - modèle simplifié d'événements
  - développé par un groupe indépendant.
- Types d'événement :
  - début et fin de document ;
  - début et fin d'éléments ;
  - attributs, texte, … .
- Nombreux parseurs publics
  - XP de James Clark, Aelfred, Saxon
  - MSXML3 de Microsoft
  - Xerces de Apache
  - JAXP de SUN

SAX: Principe de fonctionnement



Les méthodes essentielles de DefaultHandler

- XMLReader
  - setContentHandler
  - setErrrorHandler
  - parse
- ContentHandler
  - startDocument
  - endDocument
  - startElement
  - endElement
  - characters
- ErrorHandler
  - fatalError
  - error
  - warning
- EntityResolver
  - resolveEntity

Default Handler

- Principales méthodes:

`void startDocument()`	Receive notification of the beginning of a document.
`void endDocument()`	Receive notification of the end of a document.
`void startElement(String namespaceURI, String localName, String qName, Attributes atts)`	Receive notification of the beginning of an element.
`void endElement(String namespaceURI, String localName, String qName)`	Receive notification of the end of an element.
`void characters(char[] ch, int start, int length)`	Receive notification of character data.

Exemple SAX


<personne>
<nom>
Cover
</nom>
<prenom>
Harry
</prenom>
<adresse>
<rue>
Stendhal
</rue>
<ville>
Paris
</ville>
</adresse>
</personne>
startDocument ()
startElement (personne)
startElement (nom)
characters (Cover)
endElement (nom)
startElement (prenom)
characters (Harry)
endElement (prenom)
startElement (adresse)
startElement (rue)
characters (Stendhal)
endElement (rue)
startElement (ville)
characters (Paris)
endElement (ville)
endElement (adresse)
endElement (personne)
endDocument ()

SAX: exemple d'utilisation (Xerces)

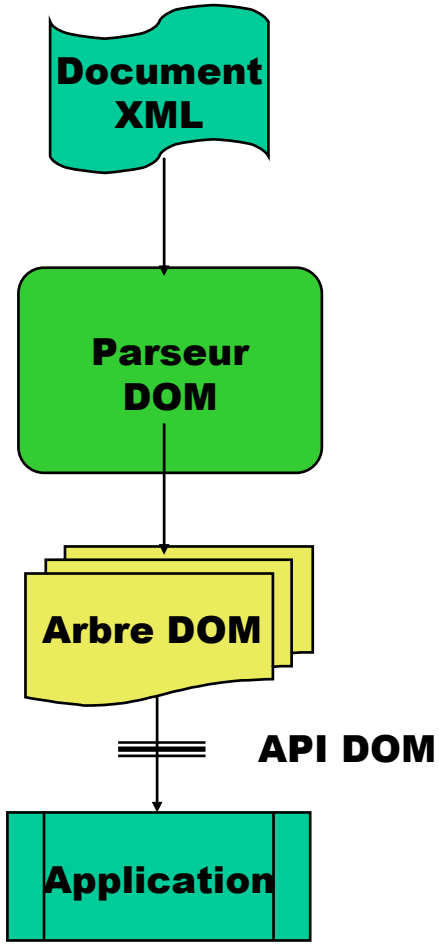
```
import org.apache.xerces.parsers.SAXParser;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

public class MySAXHandler extends DefaultHandler {
  public void startDocument () {
    System.out.println("Start document");
  }
  public void endDocument () {
    System.out.println("End document");
  }
  public static void main(String[] argv) throws Exception {
    DefaultHandler handler = new MySAXHandler();
    XMLReader reader = new SAXParser();
    reader.setContentHandler(handler);
    reader.parse(argv[0]);
  }
}
```

DOM: Document Object Model

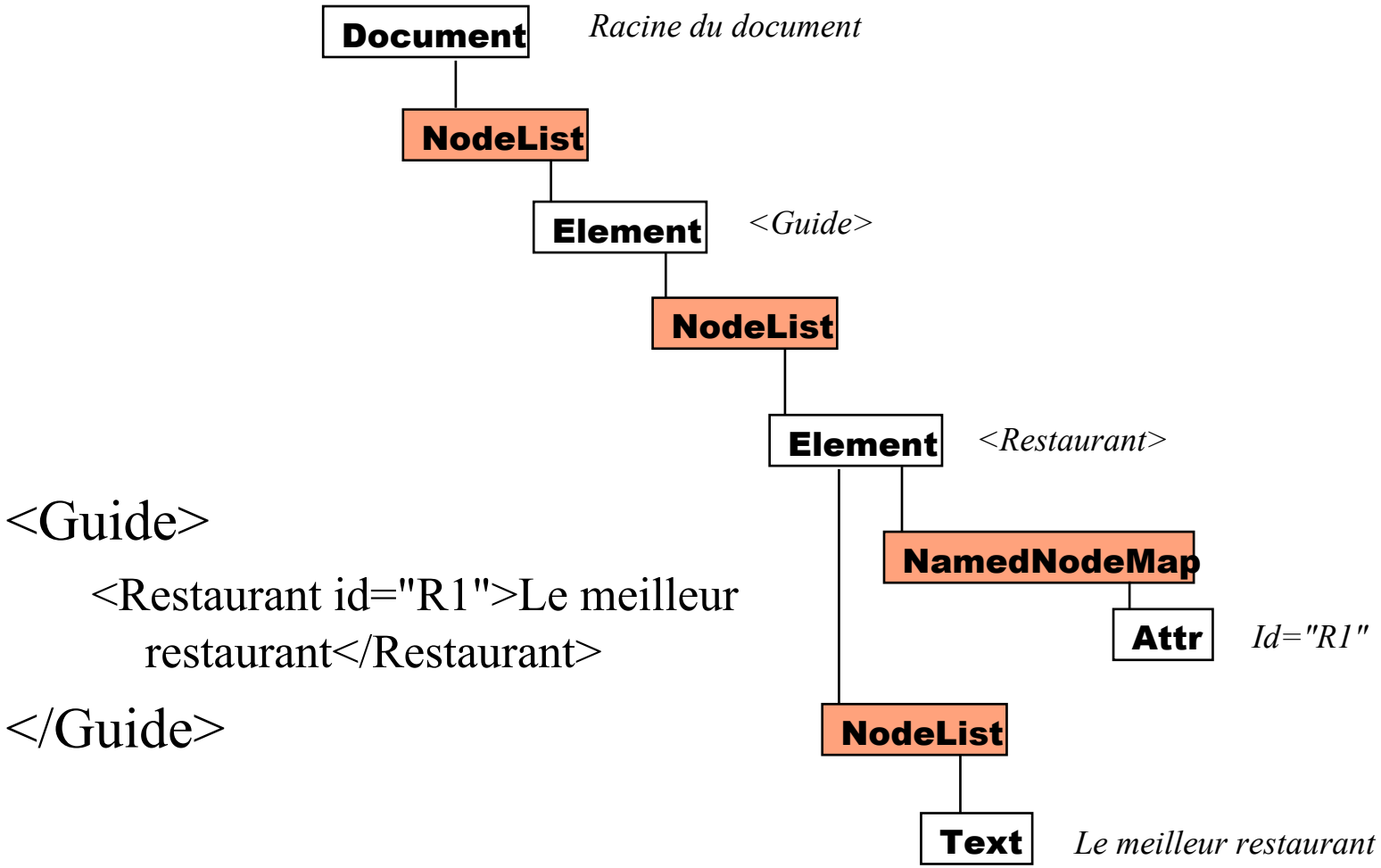
- Standard W3C fait pour HTML et XML
- Structure d'objets pour représenter un document
  - Résultat d'un "parser"
  - Arbre d'objets reliés entre eux
- Interface objet pour naviguer dans un document
  - Orientée objet
  - Peut être utilisée en:
    - Java, C++
    - C#, VB
    - Python, PHP

Principaux parseurs DOM

Xerces	Apache (Java, C++)
MSXML	Microsoft
JDOM	Jason Hunter (libre)
JAXP J2EE	Sun, ...



Exemple d'arbre DOM




```
<Guide>
    <Restaurant id="R1">Le meilleur
      restaurant</Restaurant>
</Guide>
```

Lecture d'un document XML avec DOM

```
import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.Document;

public class DOMDemo {
   ...
   public static void main(String[] argv) throws Exception {
      DOMParser parser = new DOMParser();
      parser.parse(argv[0]);
      Document doc = parser.getDocument();
      ...
   }
}
```

Org.w3c.dom.Document

- Interface qui représente une arborescence XML

`Attr createAttribute(String name)`	Crée un attribut
`Element createElement(String name)`	Crée un élément
`Element createElementNS(String namespaceURI, String qualifiedName)`	Crée un élément dans un espace nominal
`Text createTextNode(String data)`	Crée un noeud textuel
`Element getDocumentElement()`	Récupère l'élément racine
`NodeList getElementsByTagName(String name)`	Renvoie une liste de tous les éléments de nom donné.

Org.w3c.dom.Element

- Interface représentant un élément XML

String	**getAttribute**(String name) **Retrieves an attribute value by name.**
Attr	**getAttributeNode**(String name) **Retrieves an attribute node by name.**
NodeList	**getElementsByTagName**(String name) **Returns a NodeList of all descendant Elements with a given tag name, in the order in which they are encountered in a preorder traversal of this Element tree.**
String	**getTagName**() **The name of the element.**
void	**removeAttribute**(String name) **Removes an attribute by name.**
void	**setAttribute**(String name, String value) **Adds a new attribute.**

Org.w3c.dom.Attr

- Interface représentant un attribut XML

String	**getName**() Returns the name of this attribute.
Element	**getOwnerElement**() The Element node this attribute is attached to or null if this attribute is not in use.
boolean	**getSpecified**() If this attribute was explicitly given a value in the original document, this is true; otherwise, it is false.
String	**getValue**() On retrieval, the value of the attribute is returned as a string.
void	**setValue**(String value) On retrieval, the value of the attribute is returned as a string.

Exemple d'utilisation de DOM

```
Public class ExempleDOM
public static main (String argc[]) throws IOException, DOMExcetion
{XMLDocument xmlDoc = new XmlDocument();
// creation des nœuds
ElementNode nom = (ElementNode) xmlDoc.createElement("nom");
ElementNode prenom = (ElementNode) xmlDoc.createElement("prenom");
ElementNode nomfam = (ElementNode) xmlDoc.createElement("nomfam");
// creation de l'arbre
xmlDoc.appendChild(nom);
nom.appenChild(prenom);
prenom.appendChild(xmlDoc.createTextNode("Jean");
nom.appenChild(nomfam);
nomfam.appendChild(xmlDoc.createTextNode("Dupont");
// positionnement d'un attribut
nom.setAttribute("ville", "Paris");
// sortie
System.exit(0); } }
```

Document:

```
<nom ville ="Paris">
    <prenom> Jean </prenom>
    <nomfa> Dupont </nomfa>
</nom>
```

SAX et DOM

SAX

☺ Rapide

☺ Peu gourmand en mémoire

☹ Accès séquentiel

☹ Unidirectionnel au document

DOM

☺ Accès aléatoire au document

☺ Vue hiérarchique du document

☺ Permet de construire/modifier un document

☹ Gourmand en mémoire

Outils XML

- Editeurs
  - XML SPY…
- Browsers
  - Mozilla…
- Convertisseurs
  - Xalan…
- Parseurs
  - SAX, JDOM…
- Serveurs XML
  - Cocoon (cadre de publication XML)…

Pour finir

- QUESTION:
  - Qu'est ce qu'on peut faire avec XML?

- REPONSE:
  - Imagination is more important than knowledge

    « Albert Einstein »

Références

- Elliotte Rusty Harold, XML bible, 2nd Edition, Wiley, 2001, 1206 pages.
- Alain michard, XML: langage et applications, Eyrolles, 2000
- Support de cours en ligne de George Gardarin, http://perso.orange.fr/georges.gardarin/Cours2003.htm
- T. Bray et.al. (ed.). *Extensible Markup Language* (XML) 1.0. World Wide Web Consortium (W3C), February 1998. http://www.w3.org/TR/REC-xml
- B. McLaughlin. Java and XML. *OReilly and Associates*, Sebastopol, 2000