XML: eXtensible Mark-up Language

Natalia Grabar

INSERM UMRS 872, eq. 20 Université René Descartes Paris France;
DIH, HEGP/APHP - 20 rue Leblanc - Paris 15

Mise à niveau, 2009-2010

Plan

1. Intoduction et Initiation à XML
2. Création de documents XML
3. Transformation de documents XML

Biblio:

- *XML, langage et applications*, Alain Michard, Editions Eyrolles, 1999
- *XML in a nutshell*, E.R. Harold & W.S. Means, traduction de T. Broyer, P. Ensarguet, A. Ketterlin, Editions O'REILLY, 2001
- *Comprendre XSLT*, Bernd Amann et Philippe Rigaux, Editions O'REILLY, 2002

Document électronique ou document numérisé

- *Un document est l'ensemble constitué d'un support d'information et des données enregistrées sur celui-ci sous une forme en générale permanente et lisible par l'homme ou par une machine.* (définition ISO)
- Document électronique (image, fichier son, texte)
  objet informatique manipulable par la machine
- Caractéristiques d'un document :
  présentation, structure, contenu, support

Gestion électronique de documents (GED)

Système de GED:

- Système informatisé pour:
  - gestion, classement, stockage, archivage, recherche de documents électroniques
- Système de gestion informatisé du cycle de vie de documents électroniques:
  - depuis sa création jusqu'à sa destruction
  - pour faciliter et optimiser l'accès à l'information qu'il contient et à celle qui le concerne (métadonnées)
- www.archimag.com : Toute l'actualité de la gestion électronique de documents.
- APROGED (Association des professionnels de la GED) : www.aproged.org

Format des documents

- Ouvert *vs.* fermé:
  - Format ouvert: la spécification est publiquement accessible,
  - Format fermé: la spécification est secrète

    généralement, un seul logiciel est capable de l'exploiter
- Normalisation:
  - Normalisation par une institution publique ou internationale (ISO, W3C)
  - Les autres, qui peuvent devenir un standard populaire et éventuellement normalisés par la suite
- Propriétaire:
  - Élaboré par une entreprise, dans un but essentiellement commercial
  - Peut être:
    - ouvert (le format PDF d'Adobe par exemple) s'il est publié
    - fermé (le format *.doc* de Microsoft par exemple)

Format propriétaire

- Incompatibilité verticale:
  - Documents word2000 pas lisibles par Word 6
  - Lecture possible si l'on construit un convertisseur spécial
- Incompatibilité horizontale:
  - D'autres applications doivent construire des convertisseurs pour lire ce type de document

Format normalisé

- Norme internationale garantie par un organisme (ISO, W3C, ...)
- Standard industriel (propriété d'un constructeur)
  engagement publique sur un format et sur sa stabilité
- Le format normalisé repose sur un consensus entre acteurs:
  - utilisateurs, développeurs, constructeurs, ...
- Exemple: le développement du web est devenu possible grâce à l'adoption d'un format commun (HTML)

Différents formats

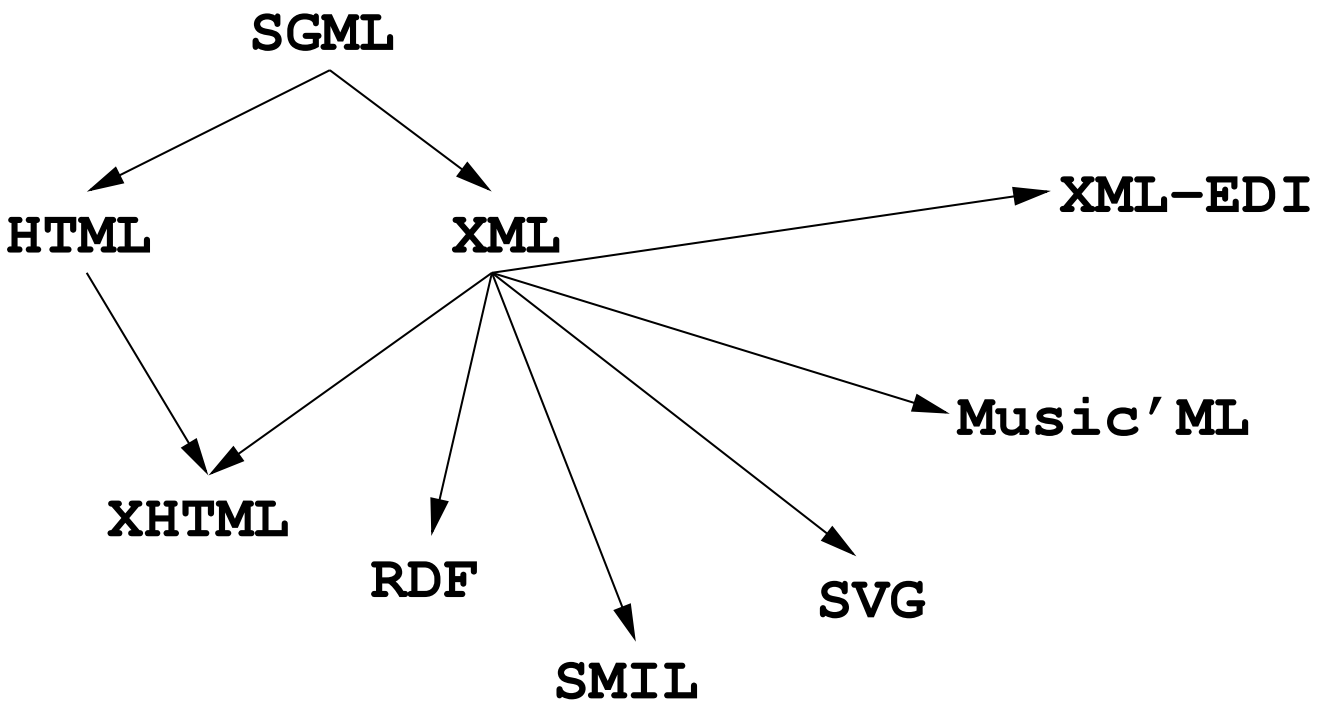
1. Word:
   - Format propriétaire de représentation des documents
     - documentation inaccessible au grand public
   - RTF (Rich Text Format): format d'échange pour documents word
     - en principe, une fidèle représentation du format propriétaire
2. PDF:
   - Format propriétaire de représentation de Adobe
   - Devenu un standard industriel
   - Format de description des documents dans leur présentation exacte de façon portable
3. Latex:
   - Système TEX de composition de documents
   - Fichier source balisé:
     - balises logiques pour la structure des documents
     - balises physiques pour un contrôle sur le résultat obtenu
   - Standard pour plusieurs classes de documents: article, book, report
   - Possibilité de définir de nouvelles balises

Langages à balises

SGML (Standard Generalized Markup Language)

- HTML, XML, etc.
- Langages à balises
- Utilisation des DTD (Document Type Definition): éléments pouvant apparaître dans un type de documents
- Entités, représentation logique des caractères
- Dynamique

Famille SGML



Avantages de SGML

- Traitement d'un nombre croissant de documents électroniques
  - dictionnaires, annuaires, textes de lois, rapports d'activité, ...
- Optimisation du cycle de vie des documents
  - *utilisabilité* et *réutilisabilité*
- Dissociation du contenu et de la représentation physique
  - Document insensible aux changements de version

Inconvénients de SGML

- Jeu de caractères internationaux un peu léger
- Langage complet mais un peu trop complexe
- Lourd à mettre en oeuvre sur le web
- Besoin de logiciels spéciaux pour visualiser du SGML sur Internet
- Accessoire spécifique + la feuille de style du document

HyperText Markup Language (HTML)

- (1990) ... HTML
- Inventé par Tim Berners-Lee (CERN)
- Application “ pauvre ” de SGML au Web
- Langage de description de pages Web
- Principes (à l'encontre de la philosophie SGML)
- Balises de présentation (<I> italique,
 fin de ligne, ...)
- Balises de structure (<P>, ...)
- Nombre de balises limité
  - mais étendu par les éditeurs (balises propriétaires) avant acceptation par le W3C!

Document HTML

- 2 parties principales des documents:
  - en-tête (`HEAD`)
    indications sur le document:
    - titre du document (`TITLE`)
    - mots clés (`KEYWORDS`)
    - langue et jeu de caractères
  - corps (`BODY`):
    - partie visible et informationnelle du document
      texte, images, liens, formulaires, ...

Structure d'un document HTML

Quelques balises HTML:

- headings (H1, H2, H3, H4, H5, H6)
- sections (BR, P, DIV)
- listes (OL, UL, LI)
- mise en forme (B, I, U)
- tableaux (TABLE, TR, TD)
- hyperliens (A HREF, A NAME)
- entités (caractères accentués, caractères spéciaux)

Entités HTML

(caractères accentués)

car	entité	car	entité
à	à	ù	ù
é	é	û	û
è	è	ü	ü
ê	ê	î	î
ë	ë	ï	ï
ô	ô	œ	œ
ç	ç	æ	æ

Avantages de HTML

- Simplicité: HTML est facile à apprendre et à comprendre
- Portabilité: peu de styles HTML gérés par divers navigateurs
- Largement utilisé:
  - Nombreux outils de création (éditeurs HTML) et de visualisation
  - Possibilité de générer automatiquement des documents
  - Inclusion d'images, de vidéos ou de sons
  - Langages de scripts permettant un comportement dynamique normalisé des pages (HTML 4 au bout de 2 ans d'efforts)

Inconvénients de HTML

- Pas de marquage sémantique d'informations:
  - en fonction de leur signification
- Si l'on souhaite présenter des informations propres à des domaines particuliers (chimie, météorologie, ...)
  - pas de balises <ATOM> ou <CARTE>
  - difficulté d'intégration dans la norme spécifique pour chaque domaine
- HTML est fait pour être affiché dans un navigateur
  - pas pour échanger de l'information entre programmes

Inconvénients de HTML

- Pas de respect d'une sémantique formelle
  - On peut construire un document avec des balises <H2> sans balises <H1>:
    - acceptable en termes de rendu
    - inacceptable en termes de sémantique du langage
  - Faible pouvoir descriptif du contenu du document:
    - on ne sait faire que de la recherche en full-text
    - beaucoup de bruit
    - moteurs de recherche ralentis
- Difficile entretien des liens hypermedias:
  - HTML a été conçu comme si les objets présents sur Internet ne pouvaient pas changer de place
  - `Error 404, document not found`

eXtensible Mark-up Language (XML)

- En 1998 est arrivé ... XML !!!
- Un compromis entre la simplicité de HTML et la complexité de SGML
- Objectifs:
  - lisible: aucune connaissance ne doit théoriquement être nécessaire pour comprendre le contenu d'un document XML
  - autodescriptif et extensible
  - structure arborescente des documents
  - universabilité et portabilité (jeux de caractères Unicode)
  - déployable: peut être distribué par n'importe quel protocole capable de transporter du texte (HTTP, ...)

Avantages de XML

- Intégrabilité: un document XML est utilisable par toute application pourvue d'un parseur dédié
- Extensibilité:
  - édition électronique des documents
  - format commun d'échange d'informations dans n'importe quel domaine d'application:
    - transactions financières (Open Financial eXchange)
    - chimie (Chimical Markup Language)
    - mathématiques (Math ML)

Recommandations XML

- Base d'XML (version XML 1.0), février 1998
- Espaces nominaux [Namespace]
- Méchanismes de liens [XPath, XLink]
- Méchanisme de feuilles de style:
  - CSS, Cascading Style Sheet
  - XSL, eXtensible Stylesheet Language
- Description de graphes complexes [RDF]
  Ressource Description Framework (indexation, RD)
- Création et diffusion de présentations multimédias [SMIL]
- Représentation de dessins vectoriels (animation, interaction)
  Scalable Vector Graphics [SVG]
- Musique, formules mathématiques, etc.

Spécificités du document XML

- Un document XML est composé de:
  - données textuelles
  - balisage
- Un document XML doit être bien formé (document correct):
  - obéir aux règles syntaxiques du langage XML
  - avoir une seule racine: toutes les balises doivent s'imbriquer correctement les unes dans les autres
- Document valide:
  - document bien formé
  - qui obéit à une structure type (DTD, XML scheme)

Structure d'un document XML

1. Un prologue:
   - facultatif mais conseillé
2. Un arbre d'éléments:
   - le contenu propre du document
3. Commentaires et instructions:
   - facultatifs, présents dans le prologue ou dans l'arbre d'éléments

Prologue

```
<?xml version="1.0" encoding='ISO-8859-1' standalone='yes
```

- version du langage XML
- codage de caractères (facultatif):
  - par défaut ISO-10646
  - UTF-16 bits
  - UTF-8 bits

standard ISO	Pays ou Région
UTF8 (Unicode)	Jeu de caractères universel, mondial
ISO-8859-1 (latin1)	Europe occidentale, Amérique latine
ISO-8859-2 (latin2)	Europe centrale et orientale
ISO-8859-3 (latin3)	Europe du sud-est
ISO-8859-4 (latin4)	Scandinavie, Pays Baltes
ISO-8859-5	Cyrillique, Grec
ISO-8859-6	Arabe

- existence de déclarations extérieures au document (standalone vaut 'yes' par défaut)

Prologue

- Le prologue peut contenir:
  - déclaration de type de document:
    indique à quel type de structure particulière le document doit se conformer
  - référence à une DTD externe

```
<!DOCTYPE rapport SYSTEM "rapport.dtd"
[déclarations]>
```

  - de type *rapport*
  - définie dans la ressource *rapport.dtd*
- référence à une DTD interne
  `<!DOCTYPE lettre [déclarations]>`
  - de type *lettre*
  - les déclarations locales entre crochets définissent le type *lettre*

Arbre d'éléments

1. Un document est formé d'une hiérarchie d'éléments
   $\Rightarrow$ un arbre
2. Tout élément fils est complètement inclus dans son père
   $\Rightarrow$ pas de recouvrement entre balises
3. Un seul élément père qui contient tous les autres
   $\Rightarrow$ l'élément racine

Arbre d'éléments

- Un élément d'un document XML
  - une balise d'ouverture
  - le contenu de l'élément
  - une balise de fermeture
    <nom>contenu </nom>
- Un élément vide n'a pas de contenu ni de balise de clôture
  <nom élément/>
- Les noms d'éléments
  - caractères alphanumériques et _, -, ., :
  - pas d'espace ou de fin de ligne
  - 1er caractère alphanumérique ou _
  - ne doit pas commencer par *xml*

Arbre d'éléments

```
<?xml version="1.0" ?>

<annuaire>
   <carte>
        <nom>Dupond</nom>
        <prenom>Jean</prenom>
        <age>54</age>
   </carte>
   <carte>
       <nom>Durand</nom>
       <prenom>Jean</prenom>
       <age>29</age>
   </carte>
</annuaire>
```

Arbre d'éléments

Exercice

- Dessiner l'arbre d'éléments correspondant au documents précédent

Structure logique vs présentation

Une lettre

M. Dupont
3, rue de l'Avenir
99666 NewVille

NewVille, le 31 janvier 2074

Melle Durand
33, avenue de l'Espoir
66999 ParadisTown

Mademoiselle,

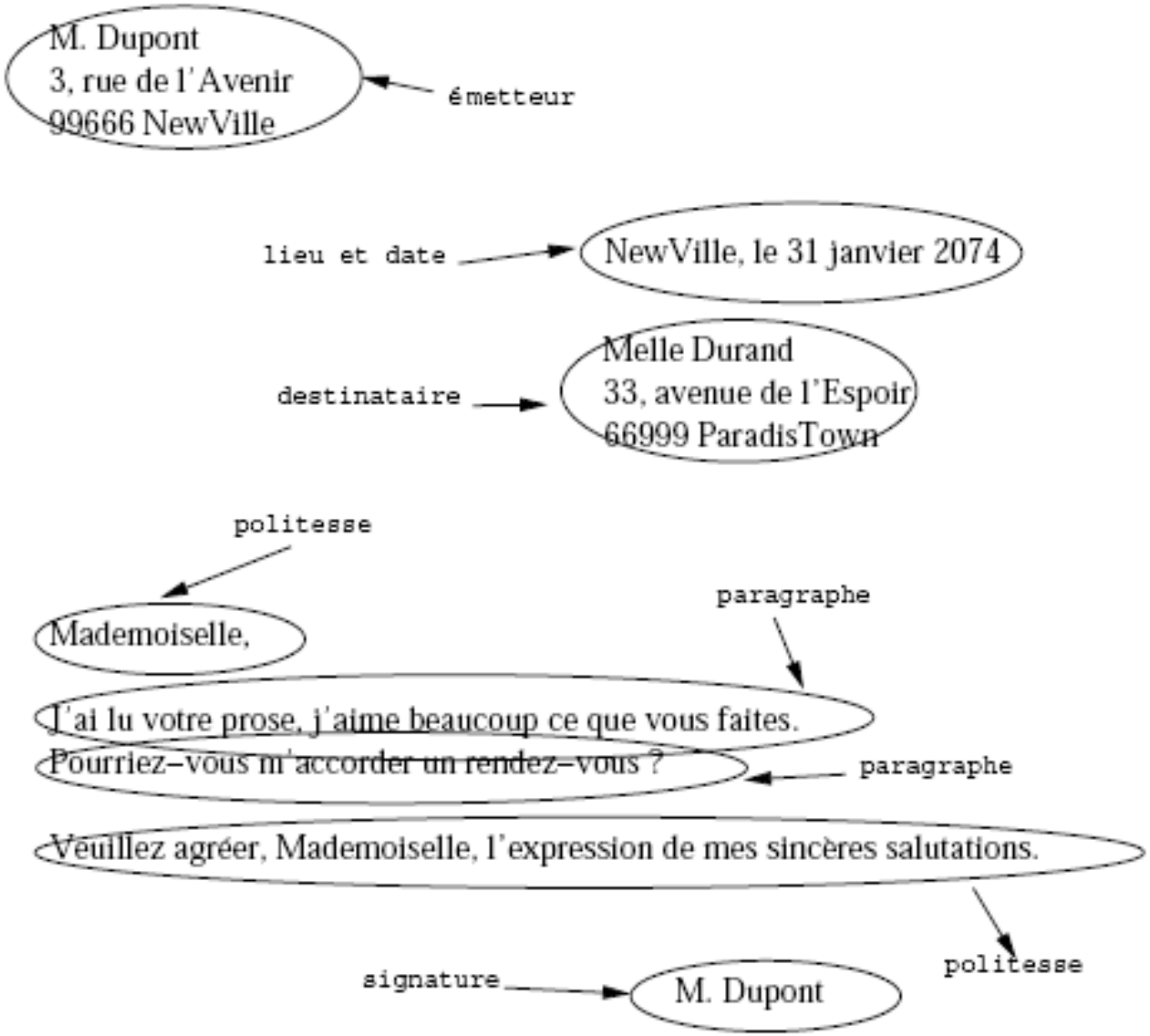
J'ai lu votre prose, j'aime beaucoup ce que vous faites.
Pourriez–vous m'accorder un rendez–vous ?

Veuillez agréer, Mademoiselle, l'expression de mes sincères salutations.

M. Dupont

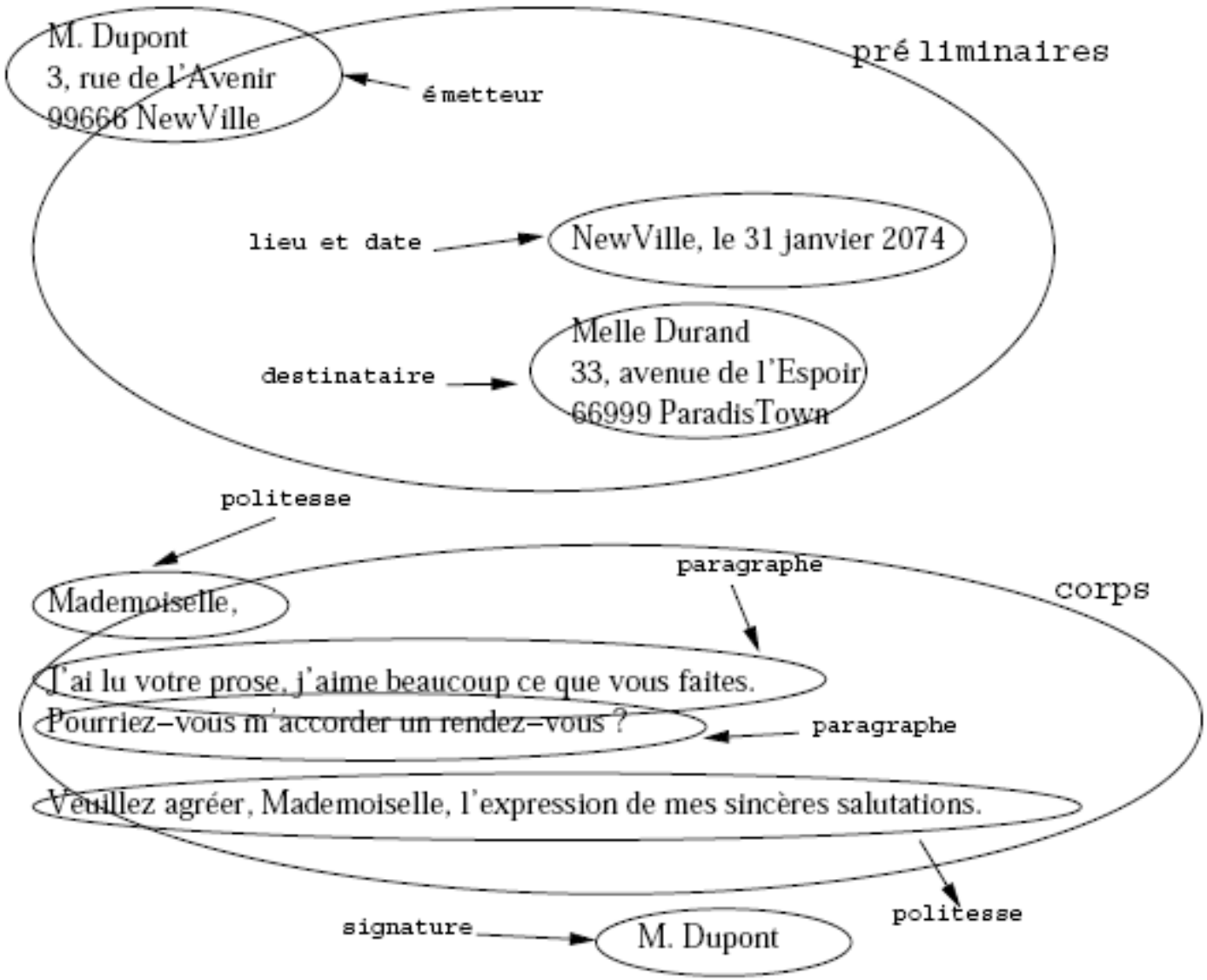
Structure logique vs présentation

Vue étiquetée de la lettre



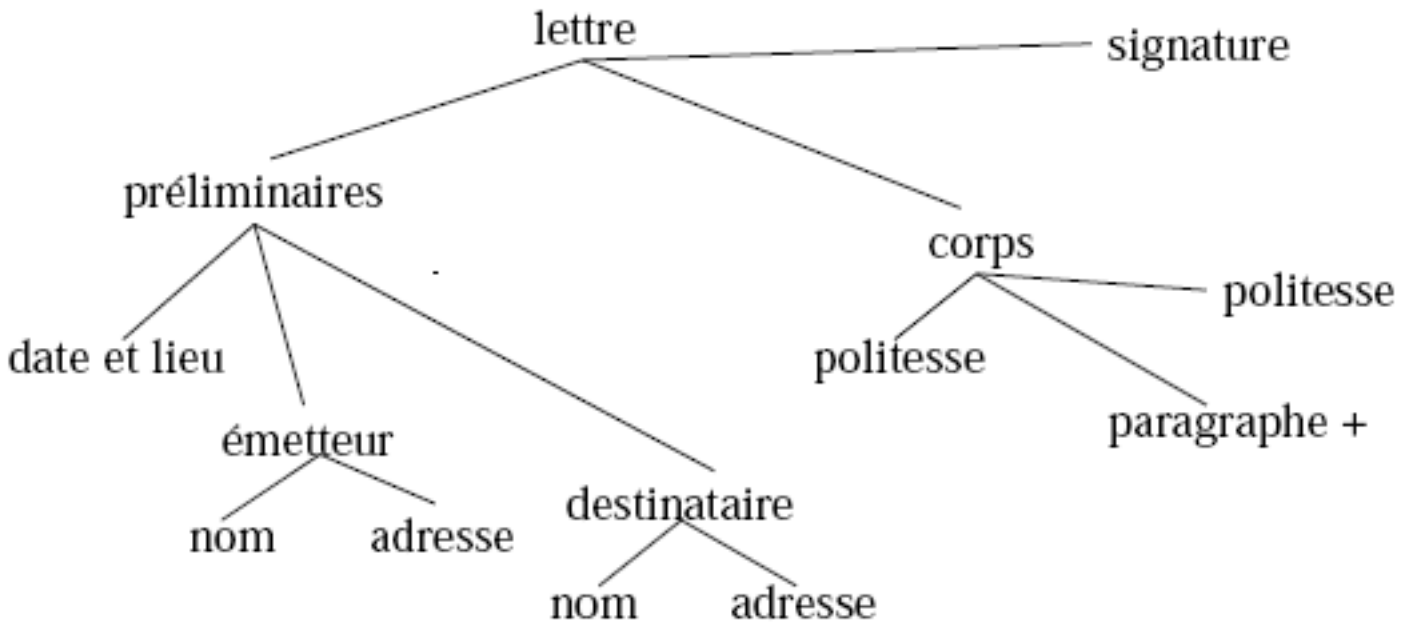
Structure logique vs présentation

Vue logique de la lettre



Structure logique vs présentation

Vue hiérarchique de la lettre



Structure des documents

- DTD ou XML schéma:
  - Définition de la structure des documents XML
  - Définition des éléments nécessaires
  - Relations logiques entre les éléments
  - Validation de la structure des documents
  - Aide pour les éditeurs d'afficher le document XML
- Éléments:
  - Instanciation de la structure

Élément XML

- Chaque élément d'un document XML se compose:
  - balise d'ouveture
  - balise de fermeture, de clôture
  - contenu de l'élément

```
<Paragraphe>
Pourriez-vous m'accorder un rendez-vous ?
</Paragraphe>
```

- Les éléments peuvent avoir des attributs

```
<Paragraphe objet="demande">
Pourriez-vous m'accorder un rendez-vous ?
</Paragraphe>
```

Élément XML

- Un élément doit englober tous les autres éléments du document
- Distinction de la casse

  `<Picture>, <picture>, <PICTURE>`
- Les caractères doivent être remplacés par les entités SGML correspondantes

  `< > & ' "`

  (< > ...)
- Les valeurs d'attributs doivent être placées entre guillemets

  `<Picture src="/images/xml.gif"/>`

Élément XML

- Un élément non vide doit avoir un marqueur d'ouverture et un marqueur de fermeture

  `<Paragraphe> ... </Paragraphe>`

- Un élément vide doit également contenir le slash (/) de fermeture

  `<Picture src="/images/xml.gif"/>`

- Emboîtement des marqueurs

  `<Italic><Bold>XML</Bold></Italic>`

Élément XML

Exercices

1. `<Italic><Bold>XML</Italic></Bold>`
2. `<tel>06 64 31 44 65</Tel>`
3. `<num&rue>34 rue de Bretagne</num&rue>`
4. `<tel>06 64 31 44 65</tel>`
5. `<num&rue>34&36 rue de Bretagne</num&rue>`
6. `<numrue>34 rue de Bretagne</num&rue>`
7. `<Italic><Bold>XML</Bold></Italic>`
8. `<tel>06 64 31 44 65<Tel>`

Élément XML

Exercices

1. <tel type=portable>06 64 31 44 65</tel>
2. <tel type=portable>06 64 31 44 65<tel>
3. <tel type="portable">06 64 31 44 65</tel>
4. <tel type="portable">06 64 31 44 65<tel>
5. <adresse>
<rue>34 rue de Bretagne
<code-postal>74900</Code-postal>
<localite>Montmagny</localite>
<pays>France
</adresse>

Document bien formé vs document valide

- Document bien formé:
  - document bien formé et conforme à la syntaxe XML
  - ne fait pas obligatoirement référence à une DTD particulière
- Document valide:
  - document bien formé et conforme à la syntaxe XML
  - conforme à une DTD prédéfinie
- Le processeur XML vérifie:
  - si la syntaxe est correcte
  - si la structure du document correspond à la structure donnée dans la DTD

Document bien formé vs document valide

- DTD (Document Type Definition):
  - spécification des règles auxquelles obéissent les éléments et leurs attributs conformes à XML
  - relations logiques qui relient les éléments entre eux
- Document XML:
  - éléments XML pertinents
  - contenu du document

DTD (Document Type Definition)

Exemple

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!-- DTD de document type Livre-->
3 <!ELEMENT OReilly:Livres
    (OReilly:Auteur*, OReilly:Titre,
     OReilly:Prix, OReilly:ISBN?)>
4 <!ATTLIST OReilly:Livres xmlns:OReilly
   CDATA #REQUIRED>
5 <!ELEMENT OReilly:Auteur (#PCDATA)>
6 <!ELEMENT OReilly:Titre (#PCDATA)>
7 <!ELEMENT OReilly:Prix (#PCDATA)>
8 <!ELEMENT OReilly:ISBN (#PCDATA)>
```

DTD (Document Type Definition)

Déclaration XML

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
```

- Dans la balise `<?xml ?>`:
  - `version="..."` version d'XML (version 1.0)
  - `encoding="..."` jeu de caractères (iso latin-1)
  - `version` et `encoding` sont des attributs
  - Les valeurs de ces attributs se trouvent entre les guillemets

DTD (Document Type Definition)

Commentaires XML

```
2 <!-- DTD de document type Livre-->
```

- Commentaires XML:
  - Dans les balises <!-- -->:
  - Le texte des commentaires est ignoré par le processus
  - Ne peuvent pas apparaître:
    - avant la déclaration XML
    - à l'intérieur des marqueurs d'un élément

DTD (Document Type Definition)

Constructeur

```
3 <!ELEMENT OReilly:Livres
    (OReilly:Auteur*, OReilly:Titre,
     OReilly:Prix, OReilly:ISBN?)>
5 <!ELEMENT OReilly:Auteur (#PCDATA)>
6 <!ELEMENT OReilly:Titre (#PCDATA)>
7 <!ELEMENT OReilly:Prix (#PCDATA)>
8 <!ELEMENT OReilly:ISBN (#PCDATA)>
```

- Le constructeur <!ELEMENT> déclare chaque élément du document:
  - OReilly:Livres, OReilly:Titre, ...

DTD (Document Type Definition)

Noms des éléments XML

- Caractères pouvant être utilisés dans les balises:
  - caractères alphanumériques
  - souligné
  - tiret
  - point, virgule
- Caractères ne pouvant pas être utilisés dans les balises:
  - espace
  - fin de ligne

DTD (Document Type Definition)

Noms des éléments XML

```
5 <!ELEMENT OReilly:Auteur (#PCDATA)>
6 <!ELEMENT OReilly:Titre (#PCDATA)>
7 <!ELEMENT OReilly:Prix (#PCDATA)>
8 <!ELEMENT OReilly:ISBN (#PCDATA)>
```

- Chaque nom d'élement peut être divisé en deux parties
  - avant et après :
    - avant: espace de noms `OReilly`
    - après: nom de marqueur
      `Auteur`, `Titre`, `Prix`, `ISBN`

DTD (Document Type Definition)

Noms des éléments XML

- Premiers caractères possibles dans les balises:
  - alphabétiques
  - souligné, virgule, deux points
- Premiers caractères déconseillés dans les balises:
  - tous les autres caractères
  - chaîne `xml`
    indépendamment de la casse
- Pas de nom d'éléments réservés

DTD (Document Type Definition)

Noms des éléments XML

- Caractères ne pouvant pas apparaître entre les balises:
  - < (<)
  - & (&)
  - > (>)
  - ’ (')
  - ” (")
- Ces caractères peuvent être interprétés comme des caractères de marquage

DTD (Document Type Definition)

Noms des éléments XML

- Noms des éléments HTML sont en majuscules
- Noms des éléments XML sont en minuscules
- Lorsque un nom est composé de plusieurs mots:
  - on utilise un trait d'union - `copyright-information`
  - on commence chaque mot par une majuscule `CopyrightInformation`

DTD (Document Type Definition)

Syntaxe de <!ELEMENT>

```
3 <!ELEMENT OReilly:Livres
    (OReilly:Auteur*, OReilly:Titre,
     OReilly:Prix, OReilly:ISBN?)>
5 <!ELEMENT OReilly:Auteur (#PCDATA)>
...
```

- <!ELEMENT déclaration d'un élément
- OReilly:Livres nom de l'élément
- entre parentèses: contenu de l'élément
- > fermeture de la balise de l'élément

DTD (Document Type Definition)

Syntaxe de <!ELEMENT>

```
3 <!ELEMENT OReilly:Livres
    (OReilly:Auteur*, OReilly:Titre,
     OReilly:Prix, OReilly:ISBN?)>
```

- L'ordre des éléments entre les parenthèses spécifie l'ordre de leur apparition dans le document
- La virgule est le connecteur des éléments

DTD (Document Type Definition)

Syntaxe de `<!ELEMENT>`

- Opérateurs de quantification:
  - `OReilly:Titre`
    - présence obligatoire
    - ne peut apparaître qu'une seule fois
  - `OReilly:ISBN?`
    - peut apparaître 0 ou 1 fois
  - `OReilly:Auteur*`
    - peut apparaître 0 ou n fois
  - `OReilly:Nom+`
    - doit apparaître au moins 1 fois

DTD (Document Type Definition)

Syntaxe de <!ELEMENT>

- Alternance des éléments

```
<!ELEMENT OReilly:Auteur
   (OReilly:Prenom*, (OReilly:Nom+|OReilly:Titre+)>
<!ELEMENT OReilly:Prix
   (OReilly:PrixNet?, OReilly:PrixBrut?)>
```

- L'alternance est indiquée par la barre verticale |
- Un des deux éléments:
  - `OReilly:Nom+` ou bien `OReilly:Titre+`

  doit apparaître dans le document

DTD (Document Type Definition)

Syntaxe de <!ELEMENT>

```
5 <!ELEMENT OReilly:Auteur (#PCDATA)>
6 <!ELEMENT OReilly:Titre (#PCDATA)>
7 <!ELEMENT OReilly:Prix (#PCDATA)>
8 <!ELEMENT OReilly:ISBN (#PCDATA)>
```

- `(#PCDATA)`
  indique que le contenu de cet élément doit être textuel

DTD (Document Type Definition)

Syntaxe de <!ELEMENT>

- #PCDATA, contenu textuel
  - est utilisé le plus souvent pour des éléments feuille, qui n'ont pas d'enfants
- EMPTY, élément vide
  - un élément vide n'a jamais d'enfant
- ANY, n'importe quel autre élément de la DTD
  - on ne l'utilise que pendant le développement d'une DTD
- le ou les éléments entre les parenthèses
  - spécifient la descendance d'un élément

DTD (Document Type Definition)

<!ELEMENT> et hiérarchie des éléments

```
3 <!ELEMENT OReilly:Livres
    (OReilly:Auteur*, OReilly:Titre,
     OReilly:Prix, OReilly:ISBN?)>
```

- Le premier élément déclaré est l'élément racine
  `<OReilly:Livres>`
- Les parenthèses introduisent:
  - les éléments enfants de l'élément racine
  - ou de tout élément parent

  `<OReilly:Auteur>, <OReilly:Titre>,`
  `<OReilly:Prix>, <OReilly:ISBN>`

DTD (Document Type Definition)

<!ELEMENT> et hiérarchie des éléments

```
3 <!ELEMENT OReilly:Livres
    (OReilly:Auteur*, OReilly:Titre,
     OReilly:Prix, OReilly:ISBN?)>
```

- Chacun de ces éléments enfant peut à son tour être parent d'autres éléments:

```
<!ELEMENT OReilly:Auteur
   (OReilly:Prenom*, OReilly:Nom+)>
<!ELEMENT OReilly:Prix
   (OReilly:PrixNet?, OReilly:PrixBrut?)>
```

DTD (Document Type Definition)

Hiérarchie des éléments

- Déterminer l'élément racine de la DTD
  - selon les applications, l'élément racine peut varier
- Déterminer les éléments fils
- Déterminer le type et le nombre d'apparition de chaque élément
- Prévoir un minimum de flexibilité
- Vérifier sur des exemples, adapter si nécessaire

Attributs des éléments XML

Déclaration des attributs

```
4 <!ATTLIST OReilly:Livres xmlns:OReilly
   CDATA #REQUIRED>
  <!ATTLIST OReilly:Prix monnaie
   CDATA #REQUIRED>
```

- La déclaration des attributs des éléments est faite avec la balise `<!ATTLIST ...>`

```
<!ATTLIST nom-element nom-attribut type-attribut
defaut>
```

Attributs des éléments XML

- Les éléments peuvent avoir des attributs
  - propriétés, informations complémentaires associées aux éléments
- Les attributs servent pour:
  - affiner
  - modifier le comportement d'un élément
- Les attributs ont un nom et une valeur:
  - Les noms d'attributs doivent respecter les mêmes règles que celles des noms d'éléments
  - Les valeurs d'attributs sont indiquées entre les guillemets

```
<Prix monnaie="Euro">10,50</Prix>
```

Attributs des éléments XML

Syntaxe des attributs

```
4 <!ATTLIST OReilly:Livres xmlns:OReilly
   CDATA #REQUIRED>
  <!ATTLIST OReilly:Prix monnaie
   CDATA #REQUIRED>
```

- Dans la balise `<!ATTLIST ...>`
  - nom de l'élément `OReilly:Livres` `OReilly:Prix`
  - nom de l'attribut attendu `xmlns:OReilly`, `monnaie`
  - type de l'attribut `CDATA`
  - nature de l'attribut `#REQUIRED`
    - valeur par défaut, nature optionnelle
- Noms des attributs:
  - Doivent respecter les mêmes règles que celles de la création des noms des éléments

Attributs des éléments XML

Valeurs par défaut

```
<!ATTLIST boite longueur CDATA "25">
<!ATTLIST boite largeur CDATA "20">
```

- Déclaration des valeurs par défaut des attributs

Attributs des éléments XML

Modificateurs des valeurs

```
<!ATTLIST OReilly:Livres xmlns:OReilly CDATA #REQUIRED>
<!ATTLIST OReilly:Prix monnaie CDATA #REQUIRED>
```

- `#REQUIRED` la valeur doit être spécifiée dans le document
- `#IMPLIED` la valeur d'attribut peut rester non spécifiée:
  - `<!ATTLIST personne marital (celibat|marie|divorce|veuf) #IMPLIED>`
  - `<!ATTLIST image semblables IDREFS #IMPLIED>`
- `#FIXED` la valeur est spécifiée dans la DTD:
  - elle est fixe
  - elle ne peut pas être modifiée dans le document
    - `<!ATTLIST date annee CDATA #FIXED "2009">`
    - `<!ATTLIST personne compagnie CDATA #FIXED "Paris13">`

Attributs des éléments XML

Types d'attributs

- CDATA contenu textuel des attributs:
  - `<!ATTLIST boite longueur CDATA "25">`
  - `<!ATTLIST boite largeur CDATA "20">`
  - `<!ATTLIST date annee CDATA #FIXED "2002">`
  - `<!ATTLIST personne compagnie CDATA #FIXED "Paris13">`

Attributs des éléments XML

Types d'attributs

- ID identifiant unique d'éléments
- IDREF
  - valeur d'un attribut de type ID unique utilisé ailleurs dans le document
  - création d'un lien interne d'un document
- IDREFS, liste d'IDREF, séparés par des espaces

Attributs des éléments XML

Exemples

```
<!ELEMENT secteur (employe*)>
<!ELEMENT employe (#PCDATA)>
<!ATTLIST employe empid ID #REQUIRED>
<!ATTLIST employe chef IDREF #IMPLIED>
```

```
<secteur>
  <employe empid="e134">Jack Russell</employe>
  <employe empid="e135">Samuel Tessen</employe>
  <employe empid="e136" chef="e134">Terry White</employe>
  <employe empid="e137" chef="e135">Steve Macon</employe>
</secteur>
```

Attributs des éléments XML

Exemples

```
<!ATTLIST boite longueur CDATA "25">
<!ATTLIST boite largeur CDATA "20">
<!ATTLIST cadre visible (true|false) "true">
<!ATTLIST personne marital
  (celibat|marie|divorce|veuf) #IMPLIED>
<!ATTLIST image fichier CDATA #REQUIRED>
<!ATTLIST image semblables IDREFS #IMPLIED>
<!ATTLIST date annee CDATA #FIXED "2002">
<!ATTLIST personne nom CDATA #REQUIRED>
<!ATTLIST personne email CDATA #REQUIRED>
<!ATTLIST personne compagnie CDATA #FIXED "Paris 13">
```

Attributs des éléments XML

Types d'attributs

- MTOKEN un nom XML légal sans espace
  - peut être utile pour déclaration de signes d'un langage
- MTOKENS liste de TOKEN séparés par des espaces
- enumerated une série de valeurs dont une peut être choisie:

```
<!ATTLIST cadre visible (true|false) "true">
<!ATTLIST personne marital
   (celibat|marie|divorce|veuf) #IMPLIED>
```

Attributs des éléments XML

Regroupement d'attributs

```
<!ELEMENT OReilly:Prix (#PCDATA)>
<!ATTLIST OReilly:Prix PrixNet CDATA #REQUIRED>
<!ATTLIST OReilly:Prix PrixBrut CDATA #IMPLIED>
<!ATTLIST OReilly:Prix Monnaie CDATA #FIXED "euro">

 <!ATTLIST OReilly:Prix
   PrixNet CDATA #REQUIRED
   PrixBrut CDATA #IMPLIED
   Monnaie CDATA #FIXED "euro">
```

Attributs des éléments XML

Entités internes

- ENTITY correspond à une entité interne déclarée dans la DTD
- ENTITIES liste d'ENTITY séparées par des espaces

```
<!ENTITY copyright "&#xA9;">
<!ENTITY ft "France Telecom">
<!ENTITY pcdata "(#PCDATA)">
<!ENTITY image1 SYSTEM
  "http://www.biomath.jussieu.fr/photo.jif" NDATA GIF89a>
<!ELEMENT auteur &pcdata;>
```

Attributs des éléments XML

Entités internes

```
<!ENTITY pcdata "(#PCDATA)">
<!ELEMENT auteur &pcdata;>

<!ENTITY nom-entité "valeur-entité">
<!ENTITY nom-entité SYSTEM "une URL">
<!ENTITY nom-entité PUBLIC "un lien public">
```

- Référence à une entité paramètre
  - forcément située dans la DTD
  - &nom;
- Il est possible d'utiliser une référence à une entité paramètre dans la déclaration de valeur d'une autre entité, paramètre ou générale

Attributs des éléments XML

Entités internes

```
<!-- dans la DTD -->
<!ENTITY pub "&#xc9;dition Eyrolles.">
<!ENTITY rights "Tous droits réservés">
<!ENTITY book "Michard A. XML langage et
applications, &#xA9 1998 &pub; &rights;">

<!-- dans le document -->
<p>&book;</p>
```

- Michard A. XML langage et applications, © 1998 Éditions Eyrolles. Tous droits réservés

Attributs des éléments XML

Éléments vides

- Un élément vide ne comporte pas de texte entre les marqueurs d'ouverture et de fermeture
- Les éléments vides sont utilisés pour
  - ajouter un contenu non littéral au document
  - fournir une information supplémentaire à l'application qui analyse le code XML

```
<Picture src="/images/xml.gif"></Picture>
<Picture src="/images/xml.gif"/>
```

- Fusionnement des marqueurs d'ouverture et de fermeture

Attributs des éléments XML

Éléments vides

```
<!ELEMENT image EMPTY>
<!ATTLIST image src ENTITY #REQUIRED>
<!ENTITY imagechapitre SYSTEM "imageChap.jpg" NDATA "jpg">

<image src="imagechapitre">
```

DTD *vs.* XML Schema

Objectifs de XML Schema

- Structure logique générique d'une classe de documents
- Structure possible de chaque instance de document
  - Éléments que peut contenir un document
  - Attributs autorisés pour chaque élément et leurs valeurs par défaut
  - Types de contenu possible pour chaque élément
  - Ordre d'apparition et le nombre d'occurrences des sous-éléments
- Type (datatype) de chaque élément et chaque attribut

DTD *vs.* XML Schema

Limitations de DTD

- Syntaxe différente: inconsistence
  - Document (instance) XML est écrit dans une syntaxe
  - DTD dans une autre syntaxe
- Types de données limités
  - Les DTDs supportent un seul type de données `PCDATA`
  - Impossible d'introduire un élément précis
    - entier entre 0 et 100
- Possibilités de réutilisations très limitées (uniquement le mécanisme des `ENTITY`)
- Les constructeurs (, barre verticale) et les indicateurs d'occurences , +, ? ne permettent pas la définition précise du nombre d'occurences autorisées d'un élément.

DTD *vs.* XML Schema

Avantages de XML-Schema

- XML-Schema offre
  - Plusieurs types de données
  - Avec possibilité de créer ses propres types de données
- Ecrits selon la même syntaxe (XML) que celle des instances de documents
  - Pas besoin d'apprendre un autre language
  - On peut utiliser les éditeurs XML pour éditer les schémas XML
  - Possibilité d'utiliser les parseurs XML pour parser les Schema
  - On peut aussi transformer les Schema avec XSLT

DTD *vs.* XML Schema

Avantages de XML-Schema

- Les Schemas XML sont Orientés-Objet (Extensibles)
  - On peut les étendre ou restreindre un type (on peut dériver une définition d'un type sur la base d'un autre)
- Les Schémas XML sont extensibles comme le sont les documents XML, ce qui permet de:
  - Réutiliser un schéma dans d'autres schémas
  - Créer son propre type de données sur la base d'un type standard
  - Référencer plusieurs schémas dans un même document

DTD *vs.* XML Schema

Avantages de XML-Schema

- Il existe des convertisseurs
  - DTD $\Rightarrow$ XML schema
  - XML schema $\Rightarrow$ DTD

Exemple d'un document XML

```
1 <?xml version="1.0" encoding="iso-8859-1"
    standalone="no"?>
2 <!DOCTYPE OReilly:Livres SYSTEM "livres.dtd">
3 <!-- Ici commencent les données XML -->
4 <OReilly:Livres xmlns:OReilly="http://oreilly.com/">
5 <OReilly:Titre> XML - precis &amp; concis
  </OReilly:Titre>
6 <OReilly:Prix>10,50</OReilly:Prix>
7 </OReilly:Livres>
```

Déclaration XML

```
1 <?xml version="1.0" encoding="iso-8859-1"
    standalone="no"?>
```

- Déclaration XML
  - définit un document comme document XML
- Dans la balise `<?xml ?>`
- Version d'XML `version 1.0` (obligatoire)
- Jeu de caractères `iso latin-1` (optionnel)
  - codage par défaut UTF-8 ou UTF-16
  - UNICODE utilise UTF-16 codage de caractères sur 16 octets

Exemple d'un document XML

Déclaration XML

```
1 <?xml version="1.0" encoding="iso-8859-1"
    standalone="no"?>
```

- Déclaration XML
  - définit un document comme document XML
- `standalone`: appel d'une DTD externe
  `yes`, `no`:
  - si `no`, la DTD se trouve dans le document distant
  - si `yes`, la DTD se trouve dans le même document
- Une DTD externe est conseillée

Exemple d'un document XML

Appel d'une DTD

```
2 <!DOCTYPE OReilly:Livres SYSTEM "livres.dtd">
```

- Identification d'une DTD pour le document XML:
  - Élément racine du document `OReilly:Livres`
    - élément le plus enveloppant auquel s'applique la DTD
  - Fichier DTD valide pour ce document:
    - la DTD du document se trouve dans un fichier local séparé `SYSTEM`
    - nom du fichier DTD `livres.dtd`

Exemple d'un document XML

Appel d'une DTD

```
2 <!DOCTYPE OReilly:Livres SYSTEM "livres.dtd">
```

- `SYSTEM` l'identificateur est un URI pointant sur une DTD
  - souvent ce sont des URL:
    - `<!DOCTYPE OReilly:Livres SYSTEM "http://www.oreilly.com/dtd/livres.dtd">`
  - ou des chemins d'accès des fichiers:
    - `<!DOCTYPE OReilly:Livres SYSTEM "livres.dtd">`
- Syntaxe
- L'éditeur charge la DTD à partir de l'URI

Exemple d'un document XML

Appel d'une DTD

```
2 <!DOCTYPE OReilly:Livres SYSTEM "livres.dtd">
```

- PUBLIC pointe sur la copie locale d'un identificateur système:
  - `<!DOCTYPE OReilly:Livres PUBLIC "-//iut//exemple de la dtd//fr" "http://www.oreilly.com/dtd/exemple.dtd">`
- Syntaxe
- Les identificateurs publics sont utilisés pour gérer les copies locales des DTD

Exemple d'un document XML

Appel d'une DTD

```
<!DOCTYPE OReilly:Livres PUBLIC
  "-//iut//exemple de la dtd//fr"
  "http://www.oreilly.com/dtd/livres.dtd">
```

- Syntaxe d'une référence à d'une DTD publique:
  - Entre les barres obliques doubles
  - + ou -
    - + si l'organisme est enregistré auprès de l'ISO
    - - si l'ogranisme n'est pas enregistré
- Propriétaire de la DTD
- Description de la DTD (espaces autorisés)
- Langue utilisée

Exemple d'un document XML

Déclaration d'une DTD interne

```
<?xml version="1.0" encoding="iso-8859-1"
  standalone="yes"?>
<!DOCTYPE OReilly:Livres [
  <!ELEMENT OReilly:Livres
    (OReilly:Auteur*, OReilly:Titre,
     OReilly:Prix, OReilly:ISBN?)>
  <!ATTLIST OReilly:Livres xmlns:OReilly
    CDATA #REQUIRED>
  ...
]>
...
```

- La DTD est placée entre les crochets dans la déclaration du type de document `<!DOCTYPE >`

Document XML

Sections `CDATA`

- Possibilité d'ajouter du texte libre
- Non traité par le parseur XML
- Dans la balise `<[CDATA[...]]>`

Document XML

Sections CDATA

```
<?xml version="1.0" encoding="iso-8859-1"
  standalone="no"?> ...
<!DOCTYPE OReilly:Livres SYSTEM "livres.dtd">
<exemple>
  <[CDATA[
   <?xml version="1.0" encoding="iso-8859-1"
     standalone="no"?>
   <!DOCTYPE exemple SYSTEM "livres.dtd">
   <!-- Ici commencent les donne'es XML -->
   <OReilly:Titre> XML - precis &amp; concis
     </OReilly:Titre>
   <OReilly:Prix>10,50</OReilly:Prix>
   </OReilly:Livres>
  ]]>
</exemple>
```

Espace de noms XML

Domaine de noms

- addition aux spécifications XML
- recommandé, mais pas obligatoire
- assure l'unicité parmi les éléments XML
- déclaré par l'utilisation de l'attribut `xmlns:XX="YY"`:
  - `XX` nom de l'ID (identifiant) de l'espace de noms
  - `YY` valeur de l'ID
    - identifiant unique, différent de tous les autres espaces de noms
    - utilisation d'une URL est recommandée

```
<OReilly:Livres xmlns:OReilly="http://oreilly.com/">
```

Espace de noms XML

- Déclaration de plusieurs espaces de noms dans un document

```
<OReilly:Livres xmlns:OReilly="http://oreilly.com/">
<xsl:stylesheet xmlns:xsl="http://www.w3.org/">
```

- Une déclaration d'espace de noms peut apparaître comme attribut de tout élément
- L'espace de noms doit rester confinée entre les marqueurs d'ouverture et de fermeture

Espace de noms XML

- Déclaration de plusieurs espaces de noms dans un élément

```
<OReilly:Livres xmlns:OReilly="http://oreilly.com/"
  xmlns:xsl="http://www.w3.org/">
</OReilly:Livres>
```

Espace de noms XML

Portée des espaces de noms

- Si on ne spécifie pas de nom après le préfixe `xmlns`:
  - l'espace de noms affecté est une valeur par défaut
  - appliqué à tous les éléments qui n'utilisent pas un préfixe d'espace de noms

Espace de noms XML

```
<Livres xmlns:"http://oreilly.com/"
   xmlns:Songline="http://www.songline.org/">
 <Livre>
  <Titre>XML - precis &amp;amp: concis</Titre>
  <ISBN>2-84177-104-0</ISBN>
 </Livre>
 <Songline:CD>18231</Songline:CD>
</Livres>
```

- L'espace de noms par défaut `http://oreilly.com/` s'applique à tous les éléments `Livres`, `Livre`, `Titre`, `ISBN`,
- sauf un `Songline:CD` qui possède son propre demaine de noms.

Espace de noms XML

- Affectation d'une chaîne vide à l'espace de noms
- pour garantir qu'il n'y ait aucun espace de noms par défaut `Prix, Numero`

```
<Livres:OReilly xmlns=" "
   xmlns:OReilly="http://oreilly.com/">
 <Livre:OReilly>
  <Titre:OReilly>XML - precis &amp;: concis </Titre:OReilly>
  <ISBN:OReilly>2-84177-104-0</ISBN:OReilly>
  <Prix>10,50</Prix>
  <Numero>254</Numero>
 </Livre:OReilly>
</Livres:OReilly>
```

Outils XML

- Navigateurs XML
  - affichage des documents
    - ressemblent à un navigateur HTML
    - sont destinés à afficher les documents XML
    - Internet Explorer à partir de la version 4
    - Netscape à partir de la version 5
    - d'autres navigateurs (Opéra)
- Éditeurs XML
  - création des documents XML
    - blocknote, XMLSpy (avec assistance)
- Produits expérimentaux

Xopus, un éditeur XML

- WYSIWYG XML éditeur
- Respect de la structure XML conforme au XML Schema
- Éditeur structuré: séparation du contenu et du style
- Transformation de format: en ligne, sur l'écran, imprimé, ...
- Pré-validation:
  - Pré-validation des format
  - Respect des XML Schema (XSD)
- Web based:
  - Fonctionne dans le navigateur: Internet Explorer et Firefox
- WYSIWYG user interface
  - XSL Stylesheet support
  - Class leading CSS Stylesheet support, with support for floats and positioning
- Change tracking
- undo/redo, copy-paste
- Spell checker

http://xopus.com/xopus-web-based-wysiwyg-xml-editor.html

Outils XML

- Parseurs XML
  - Analyse lexicale des documents XML
    - reconnaissance d'unités lexicales
  - Analyse syntaxique
    - reconnaissance des structures
    - détection d'erreurs
- Feuilles de style
  - réalisation de présentations d'un document
- Outils spécialisés (types de documents, domaines)
- Produits expérimentaux

Domaines d'application d'XML

- Publication, échanger des données
  - Édition
  - Maintenance de gros sites Web
  - Création de documents HTML
  - Échange d'information entre les entreprises
    - formats standards
  - Bases de données
    - importation et exportation de données
  - Commerce électronique
  - Applications scientifiques
  - etc.

Exercice

- Faire l'arbre et la DTD des références bibliographiques de Medline
  - exemple de fichier est fourni

Web sémantique

- Le Web sémantique: ensemble de technologies pour
  - rendre le contenu des ressources Web accessible et utilisable par les programmes et agents logiciels
  - grâce à un système de métadonnées formelles

Web sémantique

Description des métadonnées

- RDF (Resource Description Framework)
  - langage qui définit un cadre général pour la standardisation des métadonnées des ressources Web
- Sur la base de RDF:
  - FOAF pour la description des relations entre personnes
  - RDFS, OWL pour la description de la structure des vocabulaires
  - SKOS, ODP pour la description des relations

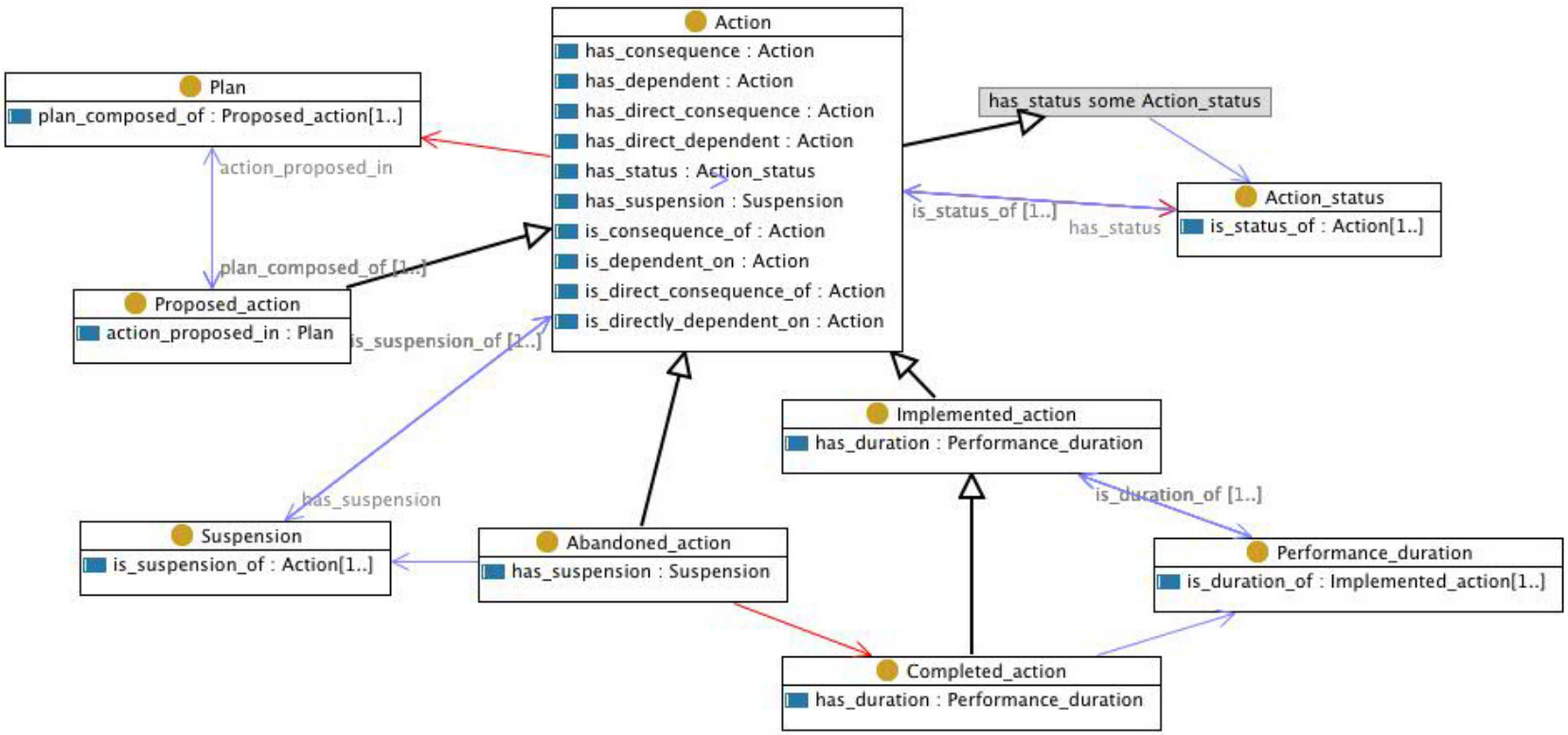
Web sémantique

ODP (Ontology Design Patterns)

- ODP sont des solutions (modèles) pour la création et maintenance des ontologies
- Ils permettent de créer des ontologies riches et rigoreuses avec peu d'effort
  - Content ODPs
  - Reengineering ODPs
  - Alignment ODPs
  - Logical ODPs
  - Architectural ODPs
  - Lexico-Syntactic ODPs

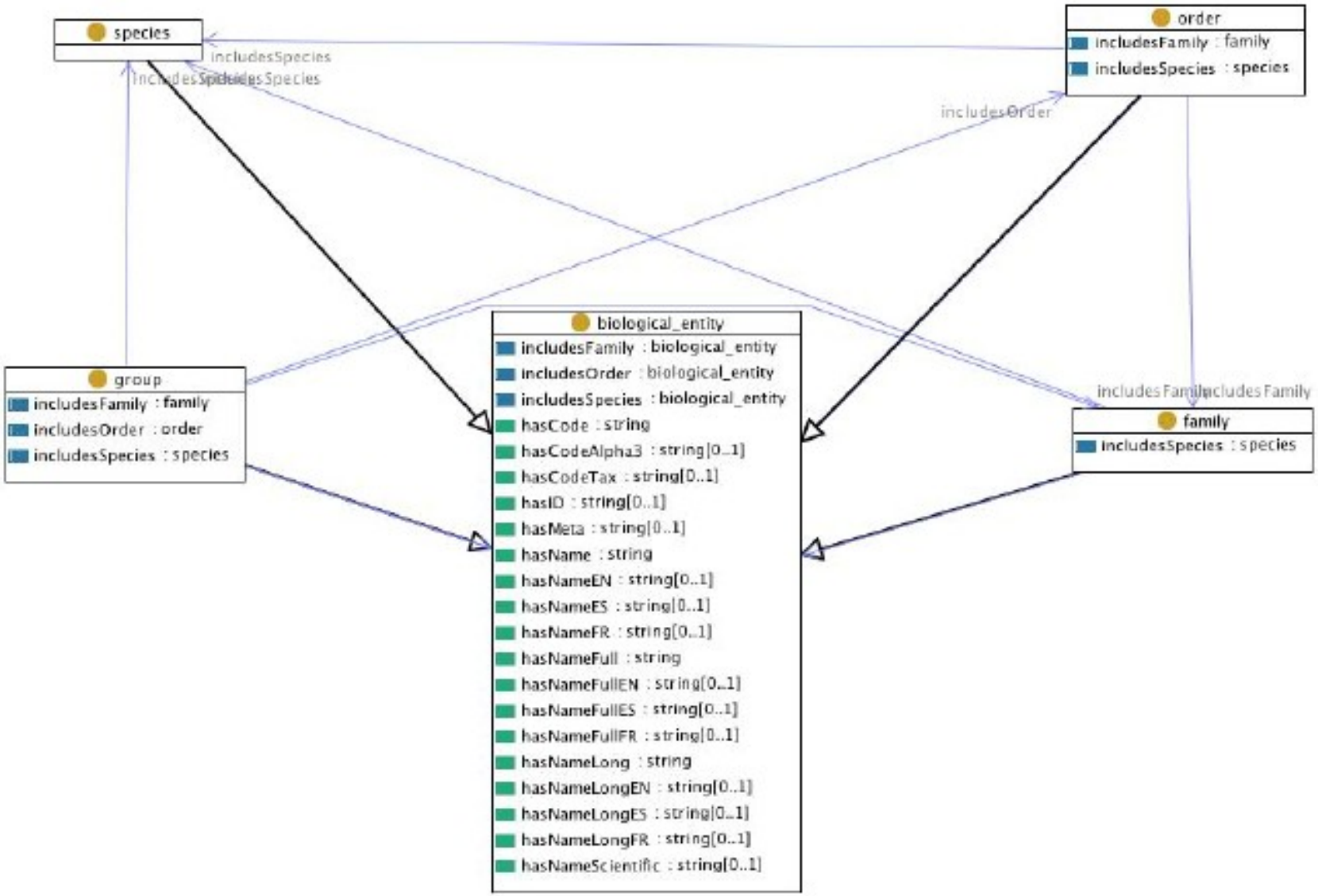
Web sémantique

Content ODPs



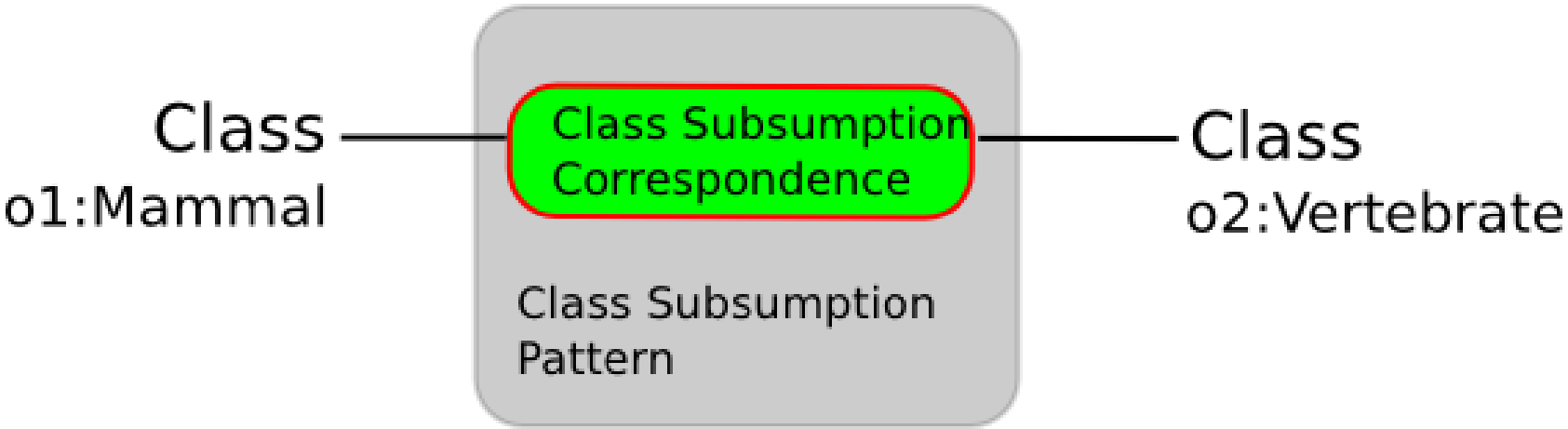
Web sémantique

Content ODPs



Web sémantique

Alignment ODPs



- Problème: La classe d'une ontology est une sous-classe dans une autre ontology, sans propriétés précises
- Solution: ODP établit une correspondence unidirectionnelle de la classe plus specifique vers une classe plus large

Web sémantique

Lexico-Syntactic ODPs

Properties of mammals are hair, sweat glands, milk, and giving live birth.

```
Property/characteristic/attribute/features of NP<class>
 be [PARA] [(NP<property>,)* and] NP<property>
```

Metals are lustrous, malleable and good conductors of heat and electricity.

```
NP<class> be [(AP<property>,)*] and AP<property>
```

Web sémantique

SKOS (Simple Knowledge Organization System)

- Spécifications et standards pour l'utilisation des ressources sémantiques (thesaurus, classifications, ...)
- Relations
  - sub-property
  - hasParent, hasMother
  - hasTopConcept
  - broader
  - broadMatch, closeMatch, exactMatch
  - historyNote
  - example
  - ...

Web sémantique

- Langages et technologies du Web sémantique:
  - parfois présentés comme des outils de représentation des connaissances
  - adaptés à l'environnement Web
  - permettant de transformer automatiquement:
    - les données en information
    - les informations en savoir

XSL (eXtensible Stylesheet Language)

- Instructions pour l'affichage des documents sur l'écran
  - ensemble de marqueurs utilisés pour appliquer des règles de formatage aux éléments d'un document XML
  - une feuille de style est un document XML
    - respect du langage XML

XLS *vs.* CSS

- XSL est l'une des plus complexes spécifications d'XML
  - XSLT (Extensible Stylesheet Language Transformation)
  - XSL-FO (Formatting Objects)
- XSL permet de transformer la structure du document DTD
  - arborescence du document HTML
- Fusion de documents différents en un seul
- Création de plusieurs documents à partir d'un seul

DTD d'un document

```
<?xml version="1.0" encoding="iso-8859-1"?>
 <!-- DTD de document type -->
 <!ELEMENT OReilly:Catalogue (OReilly:Livres+)>
 <!ELEMENT OReilly:Catalogue Prologue>
 <!ELEMENT OReilly:Livres
   (OReilly:Auteur*, OReilly:Titre,
    OReilly:Prix, OReilly:ISBN?)>
 <!ATTLIST OReilly:Livres xmlns:OReilly CDATA #REQUIRED>
 <!ELEMENT OReilly:Auteur (#PCDATA)>
 <!ELEMENT OReilly:Titre (#PCDATA)>
 <!ELEMENT OReilly:Prix (#PCDATA)>
 <!ATTLIST OReilly:Prix monnaie (Dollars|Euro) "Euro">
 <!ELEMENT OReilly:ISBN (#PCDATA)>
```

Document XML

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<?xml:stylesheet type="text/xsl" href="oreilly.xsl"?>
<!DOCTYPE OReilly:Catalogue SYSTEM "oreilly.dtd">
<!-- Ici commencent les donne'es XML -->
 <Prologue> Quelques livres</Prologue>
 <OReilly:Catalogue xmlns:OReilly="http://oreilly.com/">
  <OReilly:Livres>
   <OReilly:Titre> C++ In a Nutshell</OReilly:Titre>
   <OReilly:Auteur>Ray Lischner</OReilly:Auteur>
   <OReilly:Prix monnaie="Dollars">39,95</OReilly:Prix>
  </OReilly:Livres>
  <OReilly:Livres>
   <OReilly:Titre> Net &amp;amp; XML</OReilly:Titre>
   <OReilly:Auteur>Niel M. Bornstein</OReilly:Auteur>
   <OReilly:Prix monnaie="Dollars">39,95</OReilly:Prix>
  </OReilly:Livres>
</OReilly:Catalogue>
```

Document XML

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<?xml:stylesheet type="text/xsl" href="oreilly.xsl"?>
<!DOCTYPE OReilly:Catalogue SYSTEM "oreilly.dtd">
```

- Dans le document XML
  - faire un lien avec la feuille de style

Feuille de style XSL

```
1 <?xml version="1.0"?>
2 <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:OReilly="http://www.oreilly.com/">
3 <xsl:output method="html" encoding="iso-8859-1"/>
4 <xsl:template match="/">
5 <html>
6 <xsl:apply-templates/>
7 </html>
8 </xsl:template>
9 </xsl:stylesheet>
```

Feuille de style XSL

```
1 <?xml version="1.0"?>
2 <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:OReilly="http://www.oreilly.com/">
```

- Déclaration XML
  - avec ses attributs
- Déclaration que le document est une feuille de style XSL
- Ouverture de la feuille de style
  - avec ses attributs

Feuille de style XSL

```
3 <xsl:output method="html" encoding="iso-8859-1"/>
```

- indications sur le fichier de sortie
  - fichier HTML
  - encodé avec iso-8859-1

Feuille de style XSL

```
4 <xsl:template match="/">
5 <html>
6 <xsl:apply-templates/>
7 </html>
8 </xsl:template>
```

- affichage des valeurs des éléments

Feuille de style XSL

```
4 <xsl:template match="/">
5 <html>
6 <xsl:apply-templates/>
7 </html>
8 </xsl:template>
```

- `<xsl:template>`
  - positionnement au niveau d'un élément XML
- `<xsl:template match="/">`
  - positionnement au niveau de l'élément racine " /" est l'élément racine
- `</xsl:template>`
  - fermeture de la balise

Feuille de style XSL

```
5 <html>
6 <xsl:apply-templates/>
7 </html>
```

- impression des balises <html> </html>
- <xsl:apply-templates/>
  - insertion du contenu des éléments

Feuille de style XSL

```
9 </xsl:stylesheet>
```

- fermeture de la feuille de style

Feuille de style XSL

Affichage

```
<html>
<body>
Quelques livres

C++ In a Nutshell Ray Lischner 39,95

Net &amp;amp; XML Niel M. Bornstein 39,95

</body>
</html>
```

Feuille de style XSL

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="PMID" select='inconnu'/>
<xsl:output method="text" encoding="UTF-8" indent="yes"/>
 <xsl:template match="MedlineCitation">
 Title (<xsl:value-of select="PMID"/>):
  <xsl:value-of select=".//ArticleTitle"/><xsl:text>
  </xsl:text>
  Abstract (<xsl:value-of select="PMID"/>):
   <xsl:value-of select=".//AbstractText"/><xsl:text>
  </xsl:text>
 </xsl:template>
<xsl:template match="text()"/>
</xsl:stylesheet>
```

Feuille de style XSL

```
<xsl:output method="text" encoding="UTF-8" indent="yes"/>
```

- Élément `<xsl:output method="text">`:
  - Précise le format de sortie
  - Valeurs autorisées:
    - xml pour toutes les applications XML (valeur pas défaut)
    - html pour HTML; dans ce cas le processeur peut corriger certaines balises
    - text pour du texte quelconque

Feuille de style XSL

- Le processeur XSL parcourt l'arbre et applique les règles de transformations vérifiées (à condition vraie) aux noeuds sélectionnés.
- Les règles sont spécifiées via les éléments `template`
- A ne pas perdre de vue:
  - Règles par défaut
  - Priorités d'application des règles
- 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 `<condtition>` then `<production>`
- Codées en XML avec espace de nom xsl

Feuille de style XSL

<xsl:template ...>

```
<xsl:template match="MedlineCitation">
...
</xsl:template>
```

- Définition des règles par <xsl:template ...>
- Attributs
  - match condition de sélection des noeuds sur lesquels la règle s'applique (XPath)
  - name nom de la règle, pour invocation explicite (en conjonction avec <call-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

Feuille de style XSL

- `<xsl:template>` définir une règle et son contexte
- `<xsl:apply-templates />` appliquer les transformations aux enfants du noeud courant
- `<xsl:value-of select ... />` extrait la valeur d'un élément sélectionné à partir du noeud courant
- `<xsl:for-each>` définir un traitement itératif
- `<xsl:element>` générer un élément
- `<xsl:attribute>` générer un attribut
- `<xsl:if test=condition>` définir un traitement conditionnel

Feuille de style XSL

```
Title (<xsl:value-of select="PMID"/>):
 <xsl:value-of select=".//ArticleTitle"/><xsl:text>
 </xsl:text>
Abstract (<xsl:value-of select="PMID"/>):
 <xsl:value-of select=".//AbstractText"/><xsl:text>
 </xsl:text>
```

- Positionnement au niveau de l'élément `ArticleTitle` et `AbstractText`
- Récupération de son contenu
- Impression

Exercices

- Modifier le fichier XML de Medline
- `xsltproc style.xsl file.xml`