XML, DTD

Dan VODISLAV

CY Cergy Paris Université

Licence Informatique L3

Plan

- XML: pourquoi et comment
  - Principes et caractéristiques de base
  - Comparaison avec HTML et les BD relationnelles
- Le format XML
  - Structure des documents XML
  - Documents bien formés et valides
  - Détails de syntaxe
- Schémas XML de base: les DTD
  - Éléments, attributs, entités, notations
- Le monde XML

XML: eXtensible Markup Language

- Langage de description de documents structurés
  - Utilisation de balises (balisage structurel)
- Standard pour l'échange / la publication de données sur le web
- Héritage:
  - HTML: documents publiés sur le web
  - SGML: documentation technique (documents structurés)
    - HTML est une grammaire spécifique de SGML
  - Données structurées: bases de données relationnelles, objet
- XML 1.0: recommandation du W3C (1998)
  - Actuellement XML 1.0 – 5ème révision (2008)
  - XML 1.1 – réduit la dépendance du codage des caractères, reste marginal

Pourquoi XML?

- HTML: documents sur le web
  - Langage *de présentation* pour les documents du web
  - Ensemble de balises et grammaire fixes, mélange d'éléments de structure de document et de mise en page

  → Difficile de déduire la signification du contenu
- Données structurées: bases de données
  - Décrivent *le contenu*, pas la présentation qu'on peut en faire
  - Structure régulière basée sur des types simples: *string, int, boolean*, …

  → Les documents du web sont mal adaptés à cette structuration rigide
    - Texte, structure variable

Conclusion: on a besoin de décrire *le contenu* (indépendamment de *la présentation*), mais à l'aide d'une structuration flexible, adaptée aux documents textuels du web

Exemple

Bibliographie

- G. Gardarin, *XML : des bases de données aux services web*, Dunod, 2003
- S. Abiteboul, N. Polyzotis, *The Data Ring*, CIDR, 2007

HTML

```
<h1>Bibliographie</h1>
<ul><li>G. Gardarin, <i>XML : Des Bases de Données aux Services Web</i>, Dunod, 2003
    <li>S. Abiteboul, N. Polyzotis, <i>The Data Ring</i>, CIDR, 2007
</ul>
```

Base de données relationnelle « Bibliographie »

Auteur	Titre	Éditeur	Conférence	Année
G. Gardarin	XML : des bases de données aux services web	Dunod	NULL	2003
S. Abiteboul	The Data Ring	NULL	CIDR	2007
N. Polyzotis	The Data Ring	NULL	CIDR	2007

Exemple (suite)

- XML

```
<bibliographie>
   <ouvrage année="2003">
      <auteur>G. Gardarin</auteur>
      <titre>XML : Des Bases de Données aux Services Web</titre>
      <éditeur>Dunod</éditeur>
   </ouvrage>
   <ouvrage année="2007">
      <auteur>S. Abiteboul</auteur>
      <auteur>N. Polyzotis</auteur>
      <titre>The Data Ring</titre>
      <conférence>CIDR</conférence>
   </ouvrage>
</bibliographie>
```

XML orienté données et orienté texte

- XML est très flexible → peut représenter à la fois des données très structurées et du texte très peu structuré
- XML orienté données

```
<inventaire>
   <produit code="AZ320">
      <nom>Ordinateur</nom>
      <prix>750</prix>
   </produit>
   <produit code="LM208">
      <nom>Chaise</nom>
      <prix>63</prix>
   </produit>
</inventaire>
```

inventaire

code	nom	prix
AZ320	Ordinateur	750
LM208	Chaise	63

- XML orienté texte

```
<description>Dans la boutique <nom>Le Bureau</nom>, située
  <adresse>25 rue de l'Oise</adresse> on peut acheter tout ce
  dont on a besoin pour son bureau: <produit> ordinateur
  </produit>, <produit> chaise </produit>, etc.
</description>
```


Syntaxe XML

- Un document XML contient:
  - *Un prologue*: présence facultative, mais fortement conseillée
    - Décrit: la version du langage XML, le codage des caractères (par défaut UTF-8), l'existence de déclarations extérieures au document

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

    - Une déclaration de type de document (DTD) - facultatve

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

  - *Un arbre d'éléments*: obligatoire

    ```
    <document>
        <salutation>Bonjour!</salutation>
    </document>
    ```

  - *Des commentaires et des instructions de traitement*: facultatifs

    ```
    <!-- Ceci est un commentaire -->
    <?xml-stylesheet type="text/xsl" href="style.xsl"?>
    ```

Arbre d'éléments

- Un document est formé d'une hiérarchie (arbre) d'éléments
  - L'arbre a un élément racine unique
  - Le contenu d'un élément est délimité par des balises
  - Tout élément fils est inclus dans son père : <a> n'est pas correct
- Un élément est de la forme: **`<nom attr='valeur'> contenu </nom>`**
  - `<nom>` : *balise d'ouverture*
  - `</nom>` : *balise de fermeture,* obligatoire (sauf pour les éléments vides `<nom/>`)
    - nom formé de lettres, chiffres, '_', '-', '.', ':' (signification spéciale)
    - commence par une lettre ou par '_' et ne commence pas par les caractères « xml »
  - `contenu` : contenu d'un élément
    - vide, texte, autres éléments, imbrication de texte et d 'autres éléments
    - instructions de traitement, commentaires
  - `attr='valeur'` : *ensemble* éventuellement vide d'*attributs*
    - la valeur doit être délimitée par des apostrophes ou des guillemets

```
<livre langue='fr' editeur="O'Reilly"/>
```


Sections CDATA

- Le contenu texte (#PCDATA) peut contenir des caractères réservés (par exemple '<' ou '>')
  - On peut les « protéger » en utilisant des sections CDATA
- Exemple

```
<program>
  if(i<5) return i;
</program>
```

→ incorrect

```
<program>
  <![CDATA[if(i<5) return i;]]>
</program>
```

→ correct

Documents bien formés et documents valides

- Document XML *bien formé* : document correct *sans DTD*
  - le prologue ne contient pas de déclaration de type de document (DTD)
  - contient un arbre d'éléments correct

```
<?xml version="1.0" standalone="yes" ?>
<document>
  <salutation>Bonjour!</salutation>
</document>
```

- Document XML *valide* : document correct *avec DTD*
  - son prologue contient une déclaration de type de document (DTD)
  - son arbre d'éléments respecte la structure définie par la déclaration de type

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE document [
  <!ELEMENT document (salutation)>
  <!ELEMENT salutation (#PCDATA)>
]>
<document>
  <salutation>Bonjour!</salutation>
</document>
```


Formes sérialisée et arborescente

- Forme sérialisée d'un document/élément
  - Chaîne de caractères (texte) incluant balises et contenu textuel

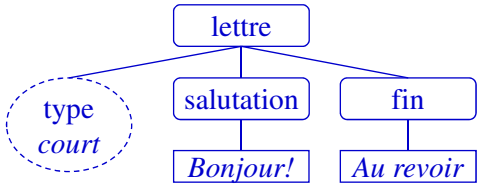
  Exemple

```
<lettre type='court'><salutation>Bonjour!</salutation><fin>Au
revoir</fin></lettre>
```

  ou avec un peu de mise en forme

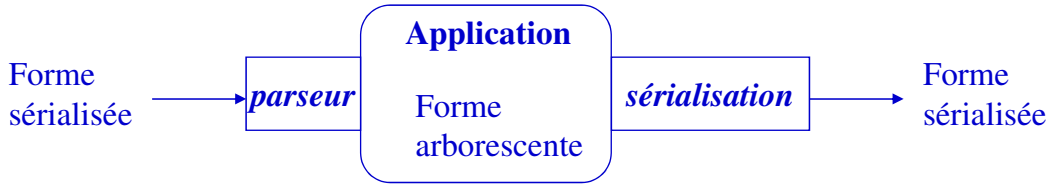
```
<lettre type='court'>
   <salutation>Bonjour!</salutation>
   <fin>Au revoir</fin>
</lettre>
```

- Forme arborescente
  - Utilisée par les applications, modèle DOM (W3C)



Utilisation des formes sérialisée et arborescente

- Les documents XML : forme sérialisée
- Les applications : transformation en forme arborescente
  - Plus simple à manipuler, parcourir, transformer



Les DTD

- DTD = grammaire pour la structure des documents
  - Facultative, interne ou externe au document
  - Contient des déclarations pour les *éléments*, *attributs*, *entités*, *notations* utilisés
- Avantages de l'utilisation de DTD
  - Partage d'une même structure entre plusieurs documents, structures « standard » pour une communauté
  - Vérification stricte et automatisable de la correction des documents

Document

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE document SYSTEM "accueil.dtd">
<document type='exemple'>
  <salutation> Bonjour! </salutation>
</document>
```

DTD

```
<!-- fichier accueil.dtd. Exemple de DTD simple -->

<!-- Définition de l'élément racine -->
<!ELEMENT document (salutation)>
<!-- Définition de l'attribut type pour l'élément document -->
<!ATTLIST document type CDATA #IMPLIED>

<!-- Un élément salutation ne contient que du texte -->
<!ELEMENT salutation (#PCDATA)>
```

Déclaration d'éléments

`<!ELEMENT nom modèle>`

- `ELEMENT` est un mot-clé (en majuscules !)
- `nom` est un nom valide d'élément
- `modèle` est le *modèle de contenu* de cet élément

- Modèles de contenu
  - *éléments* : l'élément est composé d'autres éléments (fils)
  - *données* : l'élément contient du texte
  - *mixte* : mélange de texte et de sous-éléments
  - *libre* : contenu quelconque bien formé
  - *vide* : pas de contenu, seuls les attributs sont importants

Modèles de contenu pour les éléments

- Sous-éléments : plusieurs façons de les combiner
  - séquence : `<!ELEMENT chapitre (titre,intro,section)>`
    - Remarque: *l'ordre* des éléments est important
  - alternative :

```
<!ELEMENT chapitre (titre,intro,(section|sections))>
```

  - indicateurs d'occurrence: ***** (0-n), **+** (1-n), **?** (0-1)

```
<!ELEMENT chapitre (titre,intro?,section+)>
<!ELEMENT section (titre-section,texte-section)>
<!ELEMENT texte-section (p|f)*>
```

- Données : texte

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

- Mixte : une seule façon de mélanger texte et sous-éléments est acceptée

```
<!ELEMENT p (#PCDATA|em|exposant|indice|renvoi)*>
```

Modèles de contenu pour les éléments (suite)

- Libre : contenu bien formé, mais sans restrictions

```
<!ELEMENT p ANY>
```

- Vide : pas de contenu, seuls les attributs sont importants

```
<!ELEMENT p EMPTY>
```

  - Exemple

```
<!ELEMENT p (#PCDATA|bibref)* >
<!ELEMENT bibref EMPTY>
<!ATTLIST bibref ref IDREF #REQUIRED>
```

  - utilisation:

```
<p> consulter <bibref ref='REF-19980210'/> </p>
```


Déclaration d'attributs

```
<!ATTLIST nom-élém  nom-attr type-attr décl-défaut
                    nom-attr type-attr décl-défaut …
>
```

  - Pour un élément donné on décrit la liste de ses attributs
  - Chaque attribut: un nom, un type et une valeur par défaut
  - Remarque: *l'ordre des attributs n'est pas important*

Ex.

```
<!ELEMENT ex (#PCDATA)>
<!ATTLIST ex xml:lang NMTOKEN #IMPLIED
             cible ID #REQUIRED
             nb (1 | 2 | 3) '1'
             propriétaire CDATA #FIXED 'moi' >
```

- Valeur par défaut d'un attribut
  - La valeur en question
  - `#REQUIRED` : attribut obligatoire, valeur à être précisée dans le document
  - `#IMPLIED` : attribut facultatif, valeur à être précisée dans le document
  - `#FIXED` (suivi de la valeur) : valeur de l'attribut fixée pour tout élément instance

Attributs de type ID et IDREF

- Permettent de créer des renvois à l'intérieur d'un document
  - ID: identifie l'élément référencé, IDREF: crée le renvoi
  - Transforment la structure d'arbre du document en *graphe*
- Exemple

DTD

```
<!ELEMENT personne (nom, …)>
<!ATTLIST personne num ID #REQUIRED>
<!ELEMENT livre (titre, auteur+, …)>
<!ELEMENT auteur EMPTY>
<!ATTLIST auteur ref IDREF #REQUIRED>
```

Document

```
<personne num='p1'><nom>Gardarin</nom></personne>
<personne num='p2'><nom>Valduriez</nom></personne>
<livre>
    <titre>Bases de données</titre>
    <auteur ref='p1'/>
    <auteur ref='p2'/>
</livre>
```

- Remarque: les renvois ne sont pas typés, rien ne garantit qu'on référence une personne
  - En fait *ref='p1'* renvoie vers n'importe quel élément qui a un attribut ID de valeur 'p1'

Question: attribut ou sous-élément ?

- On a le choix de représenter les composantes d'un élément:
  - Par des attributs:
    `<livre titre="XML pour les nuls">…</livre>`
  - Par des sous-éléments:
    `<livre><titre>XML pour les nuls</titre>…</livre>`
- Quand utiliser des sous-éléments?
  - Si le contenu est complexe (composé de plusieurs parties)
  - S'il y a plusieurs instances de la composante
  - Si l'ordre des composantes est important
  - Si les espaces dans le contenu sont importants (ex. programme, vers)
- L'avantage des attributs: meilleure lisibilité
- Une règle sémantique:
  - Éléments ~ données
  - Attributs ~ metadonnées

Déclaration d'entités

- Entité: raccourci, macro
  - Définie par un *nom d'entité* et une *valeur*
  - On utilise une entité de nom A en écrivant &A;
    - Effet: remplacer dans le document &A; avec la valeur de A
- Types d'entités: prédéfinies, internes, externes
- Entités prédéfinies: caractères réservés en XML ou absents sur le clavier
  - lt (<), gt (>), quot ("), amp (&), apos (')
    - Ex: pour dire "i < 5" on écrit "i < 5"
  - #code-unicode (caractère spécifié par son code Unicode)
    - Ex: le caractère de code hexa 00A9 est écrit ©

Entités internes et externes

- Entités internes : valeur définie explicitement dans la DTD
  - Définies dans la DTD: **`<!ENTITY nom-entité "valeur">`**
- Entités externes XML : valeur définie dans un fichier externe
  - Référencées par URL (mot-clé SYSTEM)
  - Doivent être des documents bien formés

```
<?xml version='1.0' ?>
<!DOCTYPE bouquin [
  <!ENTITY chapitre1 SYSTEM "chap1.xml">
  <!ENTITY chapitre2 SYSTEM "chap2.xml">
  <!ENTITY auteur "Toto">
]>
<bouquin>
  <titre> Les joies de XML </titre>
  <auteur> &auteur; </auteur>
  <intro> Il était une fois … </intro>
  &chapitre1;
  &chapitre2;
</bouquin>
```

Entités paramètre

- Utilisées dans la DTD, pas dans le document
  - Entités: utilisées dans les documents, appel par *&nom;*
  - Entités paramètre: utilisées dans les DTD, appel par *%nom;*
  - Définition: `<!ENTITY % nom "valeur">` ou `<!ENTITY % nom SYSTEM url>`
- Exemple
  - Dans la DTD

```
<!ENTITY % genres (policier | aventures)>
<!ENTITY book "Le dahlia noir">
<!ELEMENT titre (#PCDATA)>
<!ATTLIST titre genre %genres; #REQUIRED>
```

  - Dans le document

```
<titre genre='policier'> &book; </titre>
```

produit

```
<titre genre='policier'> Le dahlia noir </titre>
```


Limitations des DTD

- Peu de types de contenu et d'attributs
  - Essentiellement du texte

  → Vérifications limitées sur la validité du contenu
- Conclusion
  - Les DTD ne sont pas suffisantes pour l'échange de données structurées dans les applications (commerce électronique, intégration de données, …)
  - Mais… elles sont encore très utilisées pour des applications simples
- Autre inconvénient: les DTD ne sont pas en format XML
- Standard pour les schémas XML plus avancés → XML Schema
  - Format XML
  - Une large palette de types (entiers, réels, dates, booléens, etc.)
  - Contraintes d'intégrité: notions de clé, unicité
  - Espaces de noms
  - Héritage de types
  - …

Le monde XML

- Dialectes XML: DTD spécifique pour une utilisation précise
  - *RSS, Atom* : pour les fils d'actualités, blogs, podcasts
  - *RDF, OWL* : pour des annotations sémantiques, ontologies
  - *WML* : sites web pour téléphone mobile (protocole WAP)
  - *SVG* : graphique 2D animée
  - *MathML* : formules mathématiques
  - *SMIL* : présentations multimédia
  - ***XHTML*** : HTML qui respecte les règles XML
  - …
- Standards XML
  - *DOM* : représentation arborescente des documents XML
  - *SAX* : API pour XML vu comme une séquence de « tokens »
  - *XML Schema*: schémas XML plus évolués
  - *XPath, XQuery* : langages d'interrogation pour XML
  - *XSL* (XSLT, XSL-FO) : feuilles de style pour XML
  - *XLink* : liens en XML
  - *Services Web* : communication entre machines basée sur XML
  - …

Le monde XML (suite)

- Outils
  - Éditeurs XML, éditeurs de schémas
  - Visualiseurs XML
  - Validateurs de schéma
  - Bases de données (XPath, XQuery, XQuery Update)
  - Processeurs XSL
  - API de programmation
  - Gestionnaires de services web
  - …

Bibliographie spécifique

- Le site W3C
  `http://www.w3.org/XML/Core`
- A. Michard, *XML - Langage et applications*, Eyrolles