Les bases de XML

Plan

1. Origines : Web et documents
2. Bref historique
3. Principaux concepts
4. Syntaxe de balisage
5. Utilisation de XML

Formats de documents

Formats liés à un processeur : TeX, PostScript, PDF, .doc

- Le format contient des instructions pour le processeur
- Seul le processeur (ou un clone) peut traiter le document
- Traitements limités aux fonctions du processeur

Langage descriptif

- Décrit le document pour lui−même
- Indépendant des processeurs
- Toutes sortes de traitements possibles

XML est un langage de balisage descriptif

Origines de XML : SGML

Objectif initial de XML : publier sur le Web des données SGML (Standard Generalized Markup Language −− ISO 8879)

SGML :

- Langage de représentation de données et documents structurés
- Langage de balisage descriptif

Problèmes de SGML :

- Complexe, difficile à apprendre
- Utilisé uniquement par les professionnels de la documentation
- Difficile à utiliser sur le Web

Le Web est un système d'information en réseau constitué d'agents (programme agissant pour une personne, une entité ou un processus) qui échangent de l'information.

Les trois piliers de l'architecture du Web :

Identification
: Des *ressources* identifiées par un schéma d'adressage universel et unique, les URIs [RFC2396]

Représentation
: Les ressources sont représentées à l'aide d'un ensemble non−exclusif de formats

Interaction
: Les ressources sont échangées selon des protocoles comme HTTP, FTP, SMTP ; MIME [RFC2046] joue un rôle clé

XML participe à la représentation des ressources sur le Web

Histoire

- SGML publié par l'ISO en 1986 −− ISO 8879
- Nombreuses propositions de simplifications de SGML
- Documentation HTML de 1992 : "The hypertext markup language is an SGML format" −− TimBL
- Nombreuses extensions propriétaires de HTML, commentaires structurés
- Création du W3C SGML ERB en juillet 1996, devenu XML WG en juillet 1997 (J. Bosak, chair)
- Création du SGML WG en septembre 1996, devenu XML IG en juillet 1997
- Téléconférences, listes de diffusion, réunions, drafts
- Bases de XML stabilisées en novembre 1996, les détails en avril 1997
- Microsoft annonce CDF en mars 1997
- Recommandation le 10 février 1998 : XML 1.0
- Seconde édition : 6 octobre 2000
- Version 1.1 : recommandation candidate

XML = Extensible Markup Language

Les noms auxquels vous avez échappé : **MGML**(Minimal Generalized Markup Language),**SIMPL**(Simple Internet Markup Protocol) − voir aussi Happy Birthday, XML!

Objectifs de XML

Un sous−ensemble de SGML (un profil d'application) optimisé pour le Web :

1. XML doit être facilement utilisable sur le Web
2. XML doit supporter une grande variété d'applications
3. XML doit être compatible avec SGML
4. Il doit être facile d'écrire des programmes qui traitent des documents XML
5. Le nombre d'options doit être réduit au minimum, idéalement à zéro
6. Les documents XML doivent être lisibles et raisonnablement clairs
7. La conception de XML doit être menée rapidement

8. La description de XML doit être formelle et concise
9. Les documents XML doivent être faciles à créer
10. La concision du balisage XML est d'une importance minime

D'après XML 1.0 Rec

Structure, contenu et présentation

Trois aspects dans les documents :

- Le contenu
- La structure logique
- La présentation

XML permet de représenter les **contenus textuels** et la **structure logique**

- Les autres contenus sont des ressources externes (photos, vidéo, sons…)
- La présentation est décrite par des moyens complémentaires (CSS, XSL)
- La présentation peut changer, indépendamment des contenus et de la structure

Structure et éléments

Un document est essentiellement représenté en XML comme une structure logique arborescente

Les éléments sont les constituants « logiques » du document :

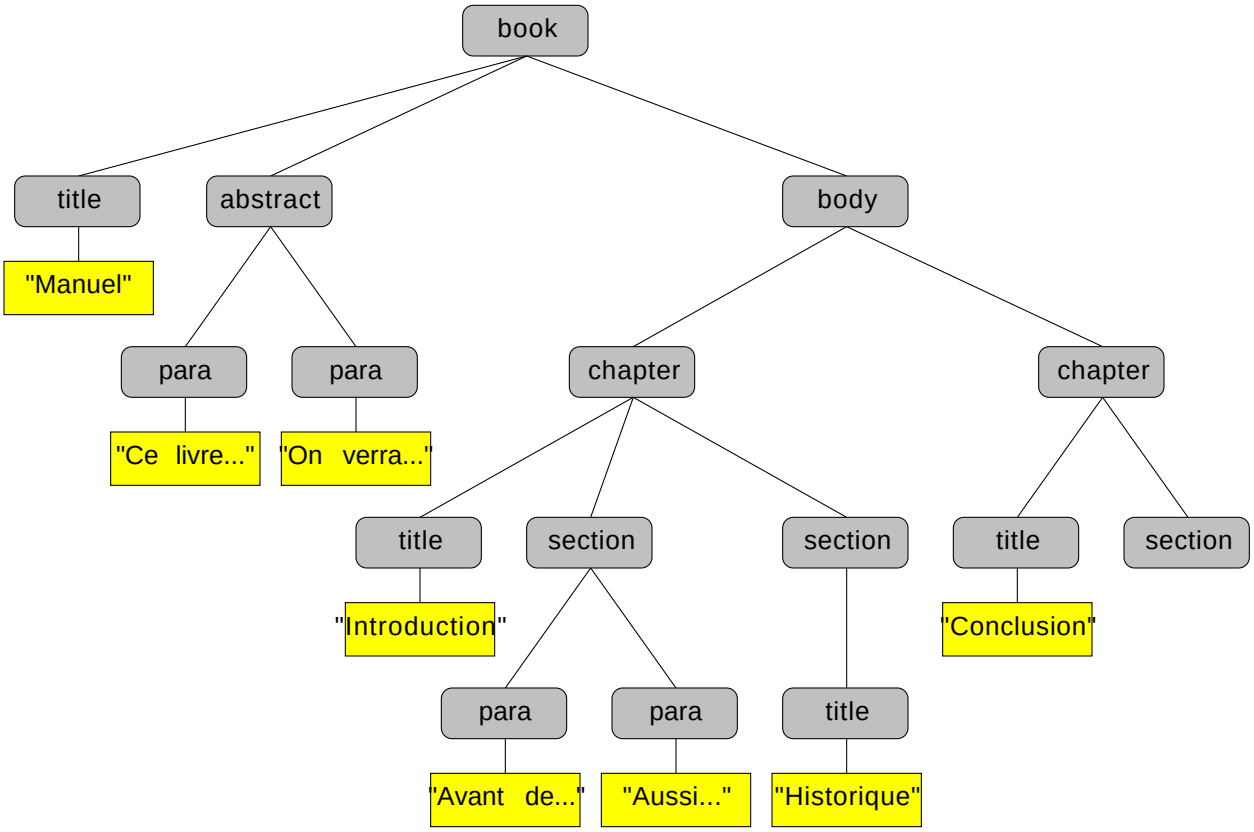
- Manuel, Titre, Auteur, Résumé, Chapitre, Section, Paragraphe, Note, Exemple, etc.
- Pièce, Personnage, Acte, Scène, Réplique, etc.
- LivreCuisine, Plat, Recette, Ingrédient, Temps, Préparation, Étape, etc.
- etc.

Les éléments ne sont pas prédéfinis, mais choisis en fonction du type de document à représenter

Les éléments terminaux de la structure sont des éléments vides ou des chaînes de caractères

L'ensemble de la structure est ordonné

Exemple



Attributs

Les attributs donnent des précisions sur les éléments et leur contenu

- Exemples : langue, statut, sous−type, etc.

Les attributs ne sont pas prédéfinis, mais choisis en fonction du type de document à représenter

- Exceptions : `xml:lang`, `xml:space`, `xmlns`

Un attribut a un nom et une valeur

Un élément peut avoir zéro, un ou plusieurs attributs

Les attributs d'un même élément n'ont pas d'ordre

Exemple

Identificateurs et références

Deux types d'attributs permettent des relations non hiérarchiques dans la structure du document :

- ID : identificateur unique dans le document (ex : `label`)
- IDREF : référence à un élément portant un attribut de type ID (ex : `item`)

Exemple

```
<Para>...voir <RefBib item="bib1"/> en fin d'ouvrage...</Para>
...
```

```
<Bilbio>
   <BibItem label="bib1">
      <Auteur> J. Dupont </Auteur><Titre>...</Titre>
   </BibItem>
   ...
</Biblio>
```

Langage de balisage

Le contenu est structurés en éléments qualifiés par des attributs avec des valeurs

Chaque élément est représenté par une paire de balises (*tags*) et son contenu :

```
<chapitre>...</chapitre>
```

Les balises ouvrantes portent les attributs :

```
<chapitre version="provisoire" date="16/06/03">
```

L'imbrication et l'ordre des éléments reflètent la structure :

```
<ol xml:lang="fr">
   <li>Des balises décrivent la structure</li>
   <li>Structure arborescente</li>
</ol>
```

Contraintes syntaxiques

Tout élément doit avoir une balise ouvrante et une balise fermante :

```
<aaa>...</aaa>
```

Raccourci d'écriture pour les éléments vides :

```
<img src="image.png"></img>  ou  <img src="image.png"/>
```

Les paires de balises ouvrantes/fermantes doivent être imbriquées :

- correct : `<aaa>...<bbb>...</bbb>...</aaa>`
- faux : `<i>...<b>...</i>...</b>`

Un document possède une racine et une seule

Tous les attributs doivent avoir une valeur

Majuscules et minuscules sont différents

Caractères et entités

Certains caractères du contenu peuvent être ambigus : `< > &`

- *Character reference*, le code du caractère en décimal ou en hexa : `&#945;` ou `&#x3b1;`
- *Character entity*, un caractère désigné par son nom : `&alpha;`
  Ce nom doit être défini
  Entités prédéfinies : `amp`, `lt`, `gt`, `apos`, `quot`

Un fragment de texte entier peut être échappé dans une section CDATA

Exemple : dans `<![CDATA[<greeting>Hello!</greeting>]]>`, les chaînes `<greeting>` et `</greeting>` sont considérées comme du contenu, par comme des balises

Internationalisation

- Le contenu et le balisage peuvent être exprimés dans n'importe quelle écriture
- Support d'Unicode (ISO/IEC 10646)
- D'autres codes peuvent être utilisés
- La langue du texte peut être indiquée : attribut `xml:lang`
- Plusieurs langues dans le même document

Exemple

Commentaires et PIs

Des commentaires peuvent être insérés, mais pas dans les balises :

```
<!-- declarations for <head> & <body> -->
```

Les *Processing instructions (PIs)* permettent de passer des instructions aux applications :

```
<?NomAppli  paramètres?>
```

Exemple : lien vers les feuilles de style

```
<?xml-stylesheet  type="text/xsl"  href="pmathml.xsl"?>
```

Espaces et sauts de ligne

Pour rendre le code XML lisible, on le formate souvent avec des sauts de ligne et des espaces.

Les sauts de ligne sont équivalents à des espaces

Les espaces sont interprétés par l'application

Le document peut contenir des attributs `xml:space` (valeur `default` ou `preserve`)

Documents bien formés

Un document XML est bien formé (*well-formed*), s'il respecte les règles de la syntaxe XML

Un processeur XML doit s'arrêter lorsqu'il rencontre une erreur

XML -- Exemples

```
<?xml  version="1.0"?>
  <memo  xml:lang="fr">
     <to>J. Smith</to>
     <from>P. Brown</from>
      <date>20/01/2000</date>
     <body>
        <p>meeting confirmed
        <em>tomorrow</em> 10am
       </p>
     </body>
  </memo>
```

```
<?xml  version="1.0"?>
  <mfrac>
     <mn> 1 </mn>
     <mrow>
        <mn> 2 </mn>
        <mo> &InvisibleTimes; </mo>
        <mi> n </mi>
     </mrow>
  </mfrac>
```

```
<?xml  version="1.0"?>
  <customer>
     <name>...</name>
     <order>...</order>
     <order>...</order>
  </customer>
```

```
<?xml  version="1.0"?>
  <par>
     <video  id="a"  src="movie1"/>
     <seq>
        <audio  src="audio1"/>
        <audio  begin="5s"  src="audio2"/>
     </seq>
```

```
                                    </par>
```

Niveaux de structuration

Jusqu'où faut−il détailler la structure?

```
<auteur> Jean Dupont </auteur>
```

```
<auteur><prénom> Jean </prénom><nom> Dupont </nom></auteur>
```

Attribut `d` de l'élément `path` de SVG : structure dans l'attribut

Un triangle : `<path d="M 100 100 L 300 100 L 200 300 z"/>`

Éléments et attributs

XHTML : éléments `<div>` `<p>` `<span>` et attribut `class`

```
<div class="section">
    <p class="titre−sect">Introduction</p>
    <div class="corps−sect">
       <p class="para"><span class="acronyme">XML</span> est un...</p>
       <p class="note">Notons que...</p>
    </div>
</div>
```

ou

```
<section>
    <titre−sect>Introduction</titre−sect>
    <corps−sect>
       <para><acronyme>XML</acronyme> est un...</para>
       <note>Notons que...</note>
    </corps−sect>
</section>
```

Attributs et contenu

On peut « cacher » le contenu dans les attributs :

```
<auteur prénom="Jean" nom="Dupont"/>
```

au lieu de

```
<auteur>
    <prénom> Jean </prénom>
    <nom> Dupont </nom>
</auteur>
```

Parsers XML

Le composant de base d'une application XML est un parser.

Un parser analyse le source XML pour une application qui traite la structure et le contenu du document

Deux grands types de parser XML

- **événements** : callback appelé pour chaque information de structure
  début balise, attribut, fin élément, etc.
  adapté aux gros documents et au streaming
- **arbre** : la structure du document est construite en mémoire
  cette structure peut être accédée par un API standard (DOM)

Limites de XML

- XML n’est qu’une syntaxe
- XML ne porte aucune sémantique
- Description de structures uniquement
- Pas de types

Avantages de XML

- Productivité
- Réutilisabilité
- Perennité
- Intégrité
- Partage
- Portabilité