

XML

Damien Nouvel



Plan

1. Structuration de données
2. Définition de la structure des documents
3. Transformations XML

Sémantique des données

- ▶ Structuration HTML
 - Mise en forme de pages web
 - Transition vers la sémantique
 - ⇒ Les données ont une structure différente de leur présentation
 - ▶ **XML** : eXtensible Markup Language
 - Utilisation de **balises** (SGML)
 - Format d'échanges de données
 - HTML est un format SGML / XML
- ⇒ Quelles structures pour quelles données ?

Exemple : Linguistique

- ▶ *Ceci est un énoncé!*

```
<phrase>
  <pronom> Ceci </pronom>
  <verbe> est </verbe>
  <grouphenominal>
    <determinant> un </determinant>
    <nomcommun> énoncé </nomcommun>
  </grouphenominal>
  <ponctuation> ! </ponctuation>
</phrase>
```

Exemple : voiture

- ▶ Description d'une voiture

```
<voiture>
  <pneus>Michelin 135 neige</pneus>
  <moteur type="diesel">1.6 TDI</moteur>
  <habitacle>
    <volant forme="rond">
      <klaxon />
    </volant>
    <sieges confort="cuir" nombre="4">top classe</sieges>
  </habitacle>
  <parechoc>abimé mais tient encore</parechoc>
</voiture>
```

Structuration

- ▶ Structuration des données
 - Arbre (graphe : nœuds et arcs)
 - Sémantique des noms des nœuds
 - Sémantique des noms d'attributs
 - ⇒ Définition d'une structure possible
 - ⇒ On ne tient pas compte de la présentation
- ▶ Bases de données
 - Traditionnellement : **tables**
 - ⇒ Pour le stockage, pas pour l'échange

Plan

1. Structuration de données
2. Définition de la structure des documents
3. Transformations XML

Structuration

- ▶ **Vocabulaire** et **structure** par domaine / application
 - Pas n'importe quelle balises / attributs (HTML)
 - Pas n'importe quelles imbrications
- ▶ Décrire une voiture
 - ⇒ Le volant doit être dans l'habitacle
 - ⇒ Un moteur peut-être de type diesel ou essence (pas d'ergol)
 - ⇒ ...
- ⇒ Vérification de la **validité** d'un document
- ⇒ Définir la structure du document ...en **XML**!
- ⇒ Possibilité d'utiliser des **espaces de noms** (xmlns)

DTD et XML Schema

- ▶ **DTD** : Document Type Definition (ISO, fichiers .dtd)
- ▶ XML Schema (W3C, fichiers .xsd)
 - Structure générale

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ...
</xs:schema>
```

- Éléments du schema
 - Éléments simples : balises sans attributs ni enfants
 - ⇒ Nom de balise et type de contenu
 - ⇒ **Types : chaîne, nombres, date, ...**
 - Éléments complexes : balises avec attributs ou enfants
 - ⇒ Contient d'autres définitions
 - ⇒ Peut tenir compte de l'ordre (**sequence**)

Formats de données populaires

- ▶ Quelques formats populaires
 - RDF (Resource Description Framework) : web sémantique
 - OWL (Ontology Web Language) : web sémantique
 - RSS (Rich Site Summary), Atom : syndication de sites
 - Dublin Core, MODS : Bibliographie
 - BiblioML : Bibliographie (BNF)
 - TEI (Text Encoding Initiative) : applications TAL
 - NIF (NLP Interchange Format) : applications TAL
 - MODS : bibliographie (Bibliothèque du Congrès, USA)
 - METS : Collection de fichiers
 - GML (Geographic Markup Language) : données géographiques
 - EbXML : commerce électronique
 - XBRL : Données comptables
 - XMI (XML Metadata Interchange) : applications / logiciels

Plan

1. Structuration de données
2. Définition de la structure des documents
3. Transformations XML

Du format de données au format de présentation

- ▶ Des données XML vers la présentation HTML

⇒ Transformations

- Balises HTML mélangées aux balises de transformation
 - ⇒ Chaque transformation réussie affiche un texte / du HTML
 - ⇒ **Sélection** et **itération** sur les nœuds
- ▶ Format du fichier

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

Les expressions XPath

- ▶ Sélectionner des nœuds dans l'**arbre** XML
 - Chemins (à l'image du CSS et des URLs)
 - **Absolus** : /noeud1/noeud2
 - **Relatifs** : noeud1/noeud2
 - **Partout** : //noeud1
 - **Attributs** : noeud1@attribut
 - Prédicats (critères)
 - Utilisation de crochets : /noeud1[condition]
 - Valeurs de nœuds ou d'attributs
 - Numéro, nombre, ordre des enfants
 - Opérateurs
 - Concaténation
 - Caractères (majuscules, minuscules)
 - ...

⇒ Mécanisme complexe mais puissant

Appliquer un squelette (template)

- Syntaxe

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

⇒ Sélectionne un nœud dans l'arbre selon l'expression

- Applique les transformations XSLT

Itérer sur les données (for-each)

- Syntaxe

```
<xsl:for-each select="expression">  
  ...  
</xsl:for-each>
```

⇒ Répète un traitement sur des nœuds

- Contenu exécuté pour chaque nœud sélectionné
- L'expression est un chemin XPath

Afficher des valeurs (`value-of`)

- Syntaxe

```
<xsl:value-of select="expression" />
```

⇒ Récupère une donnée du XML

- La valeur résultante peut-être
 - Le contenu d'un nœud (dont nœuds internes)
 - La valeur d'un attribut (`@attribut`)

Ordonner les données (sort)

- Syntaxe

```
<xsl:sort select="expression" order="ascending"  
  data-type="text" lang="fr" />
```

⇒ Ordonne des nœuds sélectionnés

- À l'intérieur de `for-each`
- Selon la donnée
 - `data-type="text"` : lexicographie (langue, majuscules)
 - `data-type="number"` : numérique

Tester les données (**if**)

- ▶ Syntaxe

```
<xsl:if test="expr1=expr2">  
  ...  
</xsl:if>
```

⇒ Exécute le contenu si le test est vrai

- ▶ Tests basé sur des expressions
- ▶ Plusieurs tests possibles
 - = : égalité des expressions évaluées
 - < : test numérique si $\text{expr1} < \text{expr2}$
 - > : test numérique si $\text{expr1} > \text{expr2}$
- ▶ Peut souvent être fait à l'aide des prédicats

Élément HTML (element)

- Syntaxe

```
<xsl:element name="monelement">  
  ...  
</xsl:element>
```

⇒ Insère un élément HTML

- Permet d'y ajouter des attributs
- Par exemple :

```
<xsl:element name="p">  
  Mon paragraphe  
</xsl:element>
```

Génère :

```
<p>Mon paragraphe</p>
```

Attribut HTML (attribute)

- Syntaxe

```
<xsl:attribute name="monattribut">  
  ...  
</xsl:attribute>
```

⇒ Insère un attribut dans un élément HTML

- Par exemple :

```
<xsl:element name="a">  
  <xsl:attribute name="href">  
    http://www.inalco.fr  
  </xsl:attribute>  
  Lien vers l'Inalco  
</xsl:element>
```

Génère :

```
<a href="http://www.inalco.fr">Lien vers l'INALCO</a>
```

Variable XSL (attribute)

- ▶ Syntaxe

```
<xsl:variable name="mvariable" select="expression">  
  ...  
</xsl:variable>
```

⇒ Enregistre une variable à utiliser plus tard

⇒ La variable peut ensuite être utilisée avec \$mvariable

- ▶ Par exemple :

```
<xsl:variable name="puissance" select="expression">  
  <xsl:value-of select="moteur/puissance" />  
</xsl:variable>  
...  
<xsl:value-of select="$puissance" />
```