Structuration de documents

Geoffray Bonnin

Contenu du cours

- Objectif : initiation aux langages d'échange de documents sur le Web
- Notions abordées
  - Bases du XML, DTD et schémas XML
  - Exploration et parcours d'arbres XML (Xpath)
  - Transformation d'arbre, le langage XSLT
  - Bases données XML (Xquery)

Evaluation

- 1 examen sur machine (vers le mois d'avril)
  - 1h30 en salle machine
  - 2 groupes en parallèle
- Evaluation par les pairs (~2 semaines plus tard)
  - 2 groupes en parallèle
  - Distributions à chaque étudiant d'examens rendus par ses pairs
  - Avec une grille d'évaluation
  - But : évaluation de votre capacité à évaluer vos pairs

INTRODUCTION
ET PRINCIPES DE BASE

Rôle du document XML

Besoin initial : échange de données

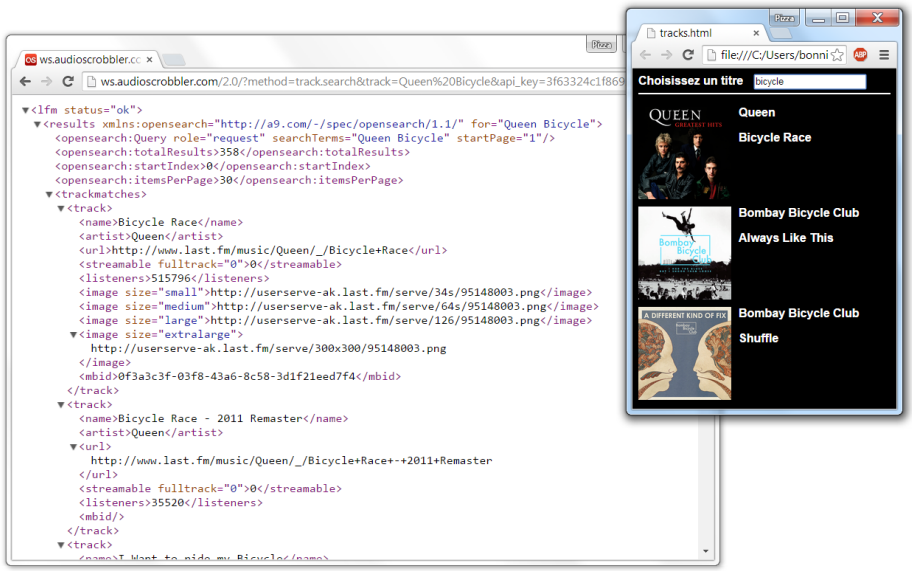
- Débuts d'internet : fichiers texte avec chaque fois un format différent

  `artiste:queen|titre:bicycle|annee:1978`

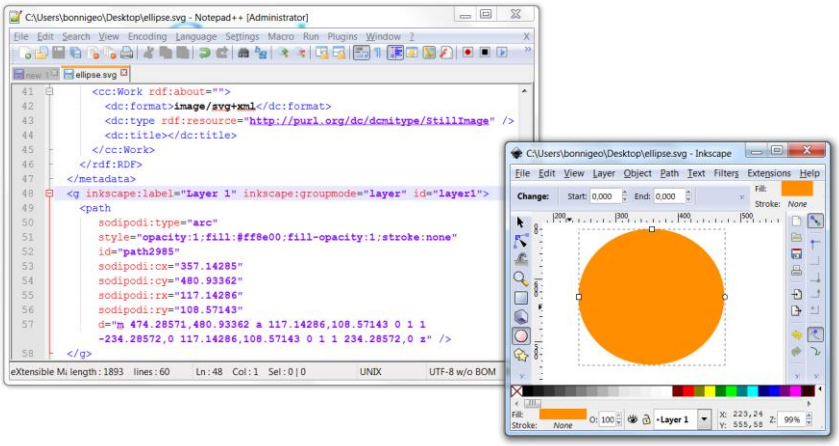
- Besoin de règles communes : naissance de XML

  XML 1.0 sorti en 1998 (20 ans après Bicycle)
  XML 1.1 sorti en 2004 (très peu utilisé)

Exemple d'utilisation de XML



Autre exemple

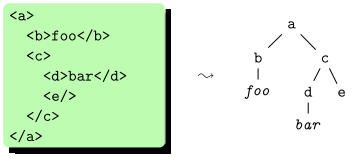


Qu'est-ce que XML ?

- Extensible Markup Language
  - Métalangage à balises
  - Standard soutenu par le W3C
- Permet
  - de **représenter** des données
  - de manière **arborescente**
- Ne permet pas de « faire » quelque-chose
  - Pas un langage de programmation
  - Pas un protocole de transport réseau
- Avantages
  - Très facile à apprendre
  - Nombreux outils pour l'exploiter

XML – Modèle de données

- XML fournit un encodage pour construire des **arbres**



- Ces arbres ont plusieurs types de **nœuds**
  - **Nœuds élément** (ici **a**, **b**, etc.) : portent un nom et peuvent avoir n'importe quel nombre de fils.
  - **Nœuds texte** (ici `foo`, `bar`) : ont un contenu textuel arbitraire ; ne peuvent pas avoir de fils.

Types de nœuds XML

Au total, il y en a 12, dont :

1. Chaque document XML est encapsulé dans un **nœud document**. Exactement l'un des fils de ce noeud doit être un nœud élément.
2. Les **nœuds élément**. Ils peuvent avoir pour fils d'autres nœuds éléments, des instructions de traitement, des commentaires et des nœuds texte.
3. Ces même nœuds élément peuvent posséder des **nœuds attribut**, qui consistent en un nom et une valeur. Les noms d'attribut doivent être uniques dans un document.
4. Les **nœuds texte**.
5. Les **nœuds espace de nom** (*namespace*).
6. Les **nœuds d'instruction de traitement** :
   `<?target Content may be any string ?>`
7. Les **nœuds commentaire** :
   `<!-- This is a comment -->`
8. Les **nœuds type de document**
   `<!DOCTYPE cours PUBLIC "cours.dtd">`
9. Les **nœuds CDATA** (character data)
   `<![CDATA[<p>Le XML c'est cool</p>]]>`

…

Exemple

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Example from www.w3.org -->
<?xml-stylesheet type='text/xsl'?>
<catalog xmlns='http://www.example.com/catalog'
    xmlns:xlink='http://www.w3.org/1999/xlink'
    xmlns:html='http://www.w3.org/1999/xhtml'>
  <tshirt code='T1534017' sizes='M L XL'
      xlink:href='http://example.com/0,,1655091,00.html'>
    <title>Staind: Been Awhile Tee Black (1-sided)</title>
    <description>
      <html:p>
        Lyrics from the hit song 'It's Been Awhile' are shown in
        white, beneath the large 'Flock &amp;amp; Weld' Staind logo.
      </html:p>
    </description>
    <price currency='EUR'>25.00</price>
  </tshirt>
</catalog>
```

Nœud d'instruction de traitement

Nœud commentaire

Nœuds espace de nom

Nœud élément

Nœud texte

Nœud attribut

Section CDATA

```
<bio>
  <links>
    <link rel="original" href="http://www.last.fm/music/Bobby+McFerrin/+wiki"/>
  </links>
  <published>Fri, 19 Aug 2011 01:06:30 +0000</published>
  <summary>
    <![CDATA[
      Bobby McFerrin (born New York City, March 11, 1950) is a <a href="http://www.last.fm/tag/jazz" class="bbcode_tag"
      rel="tag">jazz</a>-influenced a cappella <a href="http://www.last.fm/tag/vocal" class="bbcode_tag"
      rel="tag">vocal</a> performer and conductor. A ten-time Grammy Award winner, he is one of the world's best known
      vocal innovators and improvisers. His song &quot;<a
      href="http://www.last.fm/music/Bobby+McFerrin/_/Don't+Worry%2C+Be+Happy">Don't Worry, Be Happy</a>&quot; (featured
      in the 1988 movie Cocktail, and the 2005 movie Jarhead) was a #1 U.S. pop hit in 1988. He has also worked in
      collaboration with instrumental performers including pianist <a href="http://www.last.fm/music/Chick+Corea"
      class="bbcode_artist">Chick Corea</a> and cellist <a href="http://www.last.fm/music/Yo-Yo+Ma"
      class="bbcode_artist">Yo-Yo Ma</a>. This collaboration has established him as an ambassador of both the <a
      href="http://www.last.fm/tag/classical" class="bbcode_tag" rel="tag">classical</a> and <a
      href="http://www.last.fm/tag/jazz" class="bbcode_tag" rel="tag">jazz</a> worlds. <a
      href="http://www.last.fm/music/Bobby+McFerrin">Read more about Bobby McFerrin on Last.fm</a>.
    ]]>
  </summary>
  <content>
    <![CDATA[
      Bobby McFerrin (born New York City, March 11, 1950) is a <a href="http://www.last.fm/tag/jazz" class="bbcode_tag"
      rel="tag">jazz</a>-influenced a cappella <a href="http://www.last.fm/tag/vocal" class="bbcode_tag"
      rel="tag">vocal</a> performer and conductor. A ten-time Grammy Award winner, he is one of the world's best known
      vocal innovators and improvisers. His song &quot;<a
      href="http://www.last.fm/music/Bobby+McFerrin/_/Don't+Worry%2C+Be+Happy">Don't Worry, Be Happy</a>&quot; (featured
      in the 1988 movie Cocktail, and the 2005 movie Jarhead) was a #1 U.S. pop hit in 1988. He has also worked in
      collaboration with instrumental performers including pianist <a href="http://www.last.fm/music/Chick+Corea"
      class="bbcode_artist">Chick Corea</a> and cellist <a href="http://www.last.fm/music/Yo-Yo+Ma"
      class="bbcode_artist">Yo-Yo Ma</a>. This collaboration has established him as an ambassador of both the <a
      href="http://www.last.fm/tag/classical" class="bbcode_tag" rel="tag">classical</a> and <a
      href="http://www.last.fm/tag/jazz" class="bbcode_tag" rel="tag">jazz</a> worlds. <a
      href="http://www.last.fm/music/Bobby+McFerrin">Read more about Bobby McFerrin on Last.fm</a>. User-contributed text
      is available under the Creative Commons By-SA License and may also be available under the GNU FDL.
```

Déclaration XML

- Instruction de traitement particulière

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
```

- Attributs
  - Version : en général « 1.0 »
  - Encodage : « UTF-8 » (ou autre)
  - Standalone :
    - « yes » : pas de DTD (cf. suite du cours)
    - « no » : DTD externe

Nœuds élément ou attribut ?

```
<personne guitare1="papalardo"  guitare2="gibson"/>
```

ou

```
<personne>
  <guitare>papalardo</guitare>
  <guitare>gibson</guitare>
</personne>
```

?

- Si valeur peu répétée → pas d'importance
- Sinon → nœud élément

Contraintes de XML

- Documents biens formés
  - À chaque balise de début doit correspondre une balise de fin.
  - Les éléments peuvent être imbriqués, mais ils ne doivent pas se recouvrir.
  - Il ne doit y avoir qu'un seul élément racine.
  - Les valeurs des attributs doivent être entre guillemets.
  - Un élément ne doit pas avoir deux attributs avec le même nom.
  - Pas de commentaire et instruction de traitement à l'intérieur de balises.
  - Pas de caractère '<' ou '&' non échappé dans les nœuds texte ou attribut.
  - Le nom d'un élément ne peut commencer par un chiffre.
  - …
- Pour vérifier
  - Le plus simple : ouvrir le document dans un navigateur
  - Plus sophistiqué : utiliser un parseur XML (Xerces, MSXML, Expat, libxml…)

Conventions de nommage

- Employer des minuscules pour les attributs et les éléments
- Eviter les accents dans les noms d'attribut et d'élément
- Noms composés de plusieurs mots : '-', '_' ou CamelCase

```
<value-of/>
<value_of/>
<valueOf/>
```

Espaces de nom

- Mélange de documents XML → conflits de noms

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

- Utilisation d'un préfixe

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Espaces de nom

- Préfixe : raccourci vers un espace de nom

```
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

- Espace de nom par défaut

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

VALIDATION DES DOCUMENTS XML

Qu'est-ce que la validation ?

- Vérification de la conformité de la **grammaire**
  - Renforce la qualité des données échangées
  - Respect des règles parfois très important (XHTML)
- Plusieurs formes de grammaire
  - DTD
  - Schémas XML
  - Autres (RelaxNG)

Que sont les DTD ?

- Document Type Definition
- Forme de grammaire ancienne
- Avantage : rapide à écrire
- Inconvénients
  - Langage différent de XML (uniquement pour du XML)
  - Typage de données limité (que du `#PCDATA`)
  - Gestion des espaces de nom difficile (intégration de préfixes)
- Interne ou externe au document XML

DTD interne et externe

- DTD interne

```
<!DOCTYPE cours [
  ...
]>
<cours>
...
</cours>
```

- DTD externe

```
<!DOCTYPE cours SYSTEM "cours.dtd">
<cours>
...
</cours>
```

Définition d'un élément

```
<!ELEMENT nom_d_element DEF_CONTENU>
```

DEF_CONTENU

- `EMPTY` : élément sans contenu (peut avoir des attributs)
- `ANY` : peut contenir n'importe quel autre élément de la DTD
- `(#PCDATA)` : élément texte (parsable characer data)
- `(nom_d_element)` : référence vers d'autres éléments de la DTD
- Ensemble d'éléments séparés par '|', '*', '+', '?', etc.

Exemples

```
<!ELEMENT personne (nom_prenom | nom)>
<!ELEMENT nom_prenom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
```

Permet d'écrire

```
<personne>
  <nom_prenom>Jimi Hendrix</nom_prenom>
</personne>
```

ou

```
<personne>
  <nom>Jimi Hendrix</nom>
</personne>
```

Exemples

```
<!ELEMENT personne (prenom,nom)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
```

Permet d'écrire

```
<personne>
  <prenom>Jimi</prenom>
  <nom>Hendrix</nom>
</personne>
```

Exemples

Opérateurs de quantification

- * : 0 à n fois
- + : 1 à n fois
- ? : 0 ou 1 fois

```
<!ELEMENT plan (introduction?,chapitre+,conclusion?>

<!ELEMENT chapitre (auteur*,paragraphe+)>

<!ELEMENT livre (auteur+,chapitre+)+>
```

Définition d'un attribut

```
<!ATTLIST nom_d_element
  nom TYPE OBLIGATION VALEUR_PAR_DEFAUT
  nom TYPE OBLIGATION VALEUR_PAR_DEFAUT
  …
>
```

- TYPE
  - CDATA
  - ID : identifiant unique (combinaison de chiffres et de lettres)
  - IDREF : une référence vers un autre identifiant
  - IDREFS : une liste de références vers d'autres identifiants
  - NMTOKEN : un mot
  - NMTOKENS : une liste de mots
  - Énumération de valeurs séparée par le caractère '|'
- OBLIGATION
  - #REQUIRED : attribut obligatoire
  - #IMPLIED : attribut optionnel
  - #FIXED : attribut toujours présent avec une valeur fixe (rare)

Exemples

```
<!ATTLIST chapitre
  titre CDATA #REQUIRED
  auteur CDATA #IMPLIED>

<!ATTLIST crayon
  couleur (rouge|vert|bleu) "bleu">

<!ATTLIST objet devise CDATA #FIXED "Franc">
```

Définition d'une entité

- Sorte de variable à utiliser dans le document XML (ou dans la DTD)
- Trois types
  - Interne
    - Côté DTD
```
<!ENTITY nom "Jimi">
```
    - Côté XML
```
<guitariste>&nom;</guitariste>
```
  - Externe
    - Côté DTD
```
<!ENTITY papalardo SYSTEM "papalardo.xml">
```
    - Côté XML
```
<guitare>&papalardo;</guitare>
```
  - Paramétrique
    - Côté DTD
```
<!ENTITY % listeGuitares
"guitare (papalardo|gibson) #REQUIRED">
```
    - Côté DTD aussi
```
<!ATTLIST nom %listeGuitares; >
```

Schémas XML

- Issu des insuffisances des DTD
- Apports des schémas XML :
  - Tout en XML
  - Grand nombre de types de données de base
  - Possibilité de créer de nouveaux types
  - Notion d'héritage
  - …

Définition de schéma W3C – aperçu


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

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Référence à un schéma XML - aperçu

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

<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Espace de nom par défaut

Instance d'espace de nom du schéma

Espace de nom à utiliser et schéma à utiliser pour cet espace de nom

Eléments simples

- Ne contiennent que du texte
  - Ne contiennent pas d'autre élément
  - Ne contiennent pas d'attribut
- Syntaxe :

```
<xs:element name="xxx" type="yyy"/>
```

- Types les plus communs

```
xs:string    xs:decimal
xs:integer   xs:boolean
xs:date      xs:time
```

- Valeurs par défaut et fixe

```
<xs:element name="xxx" type="yyy" default="hoevels"/>
<xs:element name="xxx" type="yyy" fixed="leibinger"/>
```

Exemple d'éléments simples

- Définition

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

- Utilisation

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Attributs

- Attributs et types
  - Déclarés comme étant de type simple
  - Si un élément a un attribut → l'élément est dit de type complexe
- Syntaxe similaire à celle des éléments simples :

```
<xs:attribute name="xxx" type="yyy"/>
```

- Valeurs par défaut et fixe :
  également similaire à la syntaxe des éléments simples
- Attributs optionnels ou requis

```
<xs:attribute name="xxx" type="yyy" use="required"/>
```

Eléments complexes

- Peuvent contenir
  - D'autres éléments
  - Des attributs
  - Du texte
- Exemple

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

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Définir un élément complexe

1. Méthode directe

```
<xs:element name="guitarist">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<guitarist>
  <firstname>Jimi</firstname>
  <lastname>Hendrix</lastname>
</guitarist>
```

2. Méthode indirecte

```
<xs:element name="guitarist" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Autre exemple : contenu vide avec attributs

```
<personne guitare="papalardo" />
```

```
<xs:element name="personne" type="personneType"/>

<xs:complexType name="personneType">
  <xs:attribute name="guitare" type="xs:string"/>
</xs:complexType>
```

Autre exemple : contenu mixte

```
<guitarist>
  Salut <firstname>Jimi</firstname> <lastname>Hendrix</lastname> comment ça va ?
</guitarist>
```

```
<xs:element name="guitarist" type="personinfo"/>

<xs:complexType name="personinfo" mixed="true">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Connecteurs

Trois types de connecteurs

- Séquence (`<xs:sequence>`) :
  Les éléments doivent suivre l'ordre indiqué
- Choix (`<xs:choice>`) :
  Donne le choix entre plusieurs possibilités d'élément
- Tout (`<xs:all>`) :
  Tous les éléments sont obligatoires mais l'ordre n'importe pas

Cardinalités

Indication des occurrences minimum (`minOccurs`) et maximum (`maxOccurs`)

```
<xs:element name="guitarist" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="middlename" type="xs:string" minOccurs="0"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Autre valeur possible : `"unbounded"`