Introduction technique à XML

Code: xml-tech

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/xml-tech/xml-tech.html**
url: **http://tecfa.unige.ch/guides/tie/pdf/files/xml-tech.pdf**

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 1.4 (modifié le 2/12/02 par DKS)

Prérequis

- Internet et WWW de base, HTML de base, concepts XML-DOM

Module technique précédent: internet
Module technique précédent: www-tech
Module technique précédent: html-intro
Module technique précédent: xml-dom

Modules suivants

Module technique suivant: xml-xslt (XSLT, surtout utilisé pour server-side XML)
Module technique suivant: xml-ser (server-side XML)
Module technique suivant: php-xml (à faire), voir php-intro pour le moment
Module technique suivant: java-xml (pour programmeurs)

Objectifs

- Avoir une idée de XML
- Savoir faire de simples grammaires DTD
- Savoir rédiger des textes comme **http://tecfa.unige.ch/tecfa/teaching/formcont/webmaster2001/formcont.sxml.text**

1. Table des matières détaillée

2. Le langage XML

- XML = Extensible Markup Language
- Création/utilisation de balises (tags) et attributs selon vos besoins
- Syntaxe qui ressemble à celle de HTML
- Une application XML se définit par le formalisme DTD

Définition un peu plus formelle:

- XML est un langage pour définir une classe de "data-objects".
- Structure physique:
  - Un document XML est une collection d'entités (entities)
  - Une entité peut référer à d'autres entités pour inclusion
  - Chaque document commence par une racine ("root" ou "document entity")
- Structure logique, un document XML contient:
  - declarations, éléments, commentaires, définition de caractères spéciaux et instructions de traitement.
- Structure physique et logique doivent s'imbriquer proprement (voir -A. ""Well-formed" XML documents (arbres)" [5]
- XML (comme d'autres grammaires en informatique) est décrit sous format EBNF (Extended Backus-Naur Form).
- Voir: the Annotated XML Specification, **http://www.xml.com/axml/axml.html**

2.1 Les notions de "well-formed" et "valid" d’un document XML

A. "Well-formed" XML documents (arbres)

- Le document commence par une déclaration XML (version obligatoire),
  ```
  <?xml version="1.0"?>
  ```
  - possibilité de choisir un encodage (le défaut est utf-8):
  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  ```
- Structure hiérachique:
  - begin-tags and end-tags doivent correspondre
  - pas de croisements de type <i>......</i> ....
  - Case sensitivity
- Tags "EMPTY" (sans end-tag ...):
  - Les tags "EMPTY" utilisent la syntaxe XML (e.g.
)
- Les valeurs d’attributs sont quotés:
  - (e.g. <a href="http://tecfa.unige.ch:8080/xml.html">)
- Un seul élément racine (root):
  - L’élément root ne peut apparaître qu’une fois
  - Le root ne doit pas apparaître dans un autre élément (comme <html>)
- Caractères spéciaux: <, &, >,", ’
  - Utilisez < & > &aquot; ' à la place dans un texte !

B. "Valid XML documents"

- Un document "valid" doit être:
  - "well-formed",
  - posséder un DTD (ou une autre grammaire)
  - et être conforme au DTD (ou à la grammaire en général)

C. DTD (Document Type Definition)

- Un DTD est:
  - Une grammaire (ou jeu de règles) qui définit les tags utilisés et leurs attributs, et qui spécifie leur possible imbrication (relation).
  - et il peut être référencé par un URI ou inclus dans le document XML
- Ils existent bien d’autres grammaires comme XSchema (qui peut-être va remplacer les DTD), Relax, etc.

2.2 Exemples

Exemple 2-1: Bonjour en XML

url: http://tecfa.unige.ch/guides/xml/cocoon/simple/

Données XML

- On a un document de type <page>

```
<page>
 <title>Hello Cocoon friend</title>
 <content>
      Here is some content :)
 </content>
 <comment>
      Written by DKS/Tecfa, adapted from S.M./the Cocoon samples
 </comment>
</page>
```

Le DTD (à titre d’indication)

- Voir 3. “Les DTD” [9] pour comprendre

```
<!ELEMENT page  (title, content, comment?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```


Exemple 2-2: Une recette en XML

Source: Introduction to XML by Jay Greenspan,

http://www.hotwired.com/webmonkey/98/41index1a_page5.html?tw=html

```
<?xml version="1.0"?>
<list>
 <recipe>
   <author>Carol Schmidt</author>
   <recipe_name>Chocolate Chip Bars</recipe_name>
   <meal>Dinner
     <course>Dessert</course>
   </meal>
   <ingredients>
     <item>2/3 C butter</item>      <item>2 C brown sugar</item>
     <item>1 tsp vanilla</item>     <item>1 3/4 C unsifted all-purpose flour</item>
     <item>1 1/2 tsp baking powder</item>
     <item>1/2 tsp salt</item>      <item>3 eggs</item>
     <item>1/2 C chopped nuts</item>
     <item>2 cups (12-oz pkg.) semi-sweet choc. chips</item>
   </ingredients>
   <directions>
 Preheat oven to 350 degrees. Melt butter; combine with brown sugar and vanilla in large mixing
bowl.Set aside to cool.  Combine flour, baking powder, and salt; set aside. Add eggs to cooled
sugar mixture; beat well.  Stir in reserved dry  ingredients, nuts, and chips.
 Spread in greased 13-by-9-inch pan. Bake for 25 to 30 minutes until  golden brown; cool.  Cut
into squares.
   </directions>
 </recipe>
</list>
```

- Cette exemple montre quelques éléments d’un vocabulaire pour recettes
- Pour le DTD, voir exemple 3-8 “Un DTD pour une recette” [16]

3. Les DTD

3.1 Introduction aux DTD de XML

- DTD = Document Type Definition
- Note: Il existe d'autres formalismes pour définir une grammaire, par ex.:
  - XSD (XML Schema Definition)
- il s'agit d'une grammaire qui definit:
  - les tags possibles et leurs attributs
  - quels tags sont autorisés à l'intérieur d'autres tags (leur imbrication)
  - quels tags et attributs sont à option et quels sont obligatoires
- Chaque "tag" XML est définit comme un élément dans le DTD
  - Exemple illustratif: <!ELEMENT title (#PCDATA)>

Note: HTML avec XML ?

- HTML est une application SGML, indéfinissable avec XML !
- La version XML de HTML s'appelle XHTML (et elle est "wellformed")
  - légèrement différente de HTML 4.0
  - toutes les balises doivent être fermées
  - il est conseillé d'avoir une déclaration XML au début
  - une déclaration XHTML est obligatoire

3.2 Le langage pour définir une grammaire DTD

A. Déclaration d'un DTD dans un fichier XML

Il existe 4 façons d'utiliser un DTD

1. On déclare pas de DTD (dans ce cas le fichier est juste "bien formé"
2. On déclare le DTD et on y ajouter les défintions dans le fichier (DTD interne)
3. On déclare le DTD en tant que DTD "privé", le DTD se trouve quelque part dans votre système ou sur Internet
4. On délare un DTD "public", c.a.d. on utilise un nom officiel pour le DTD. Cela présuppose que votre éditeur et votre client conaissent ce DTD.

Lieu de la déclaration

- Le DTD est déclaré entre la déclaration de XML et le document lui-même.
- La déclaration de XML et celle du DTD font parti du prologue
  - (qui peut contenir d'autres éléments comme les processing instructions)
- Attention: l'encodage du DTD doit correspondre à celui des fichiers XML !

Exemple 3-1: Hello XML sans DTD

```
<?xml version="1.0" standalone="yes"?>
<hello> Hello XML et hello cher lecteur ! </hello>
```

Exemple 3-2: Hello XML avec DTD interne

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE hello [
   <!ELEMENT hello (#PCDATA)>
    ]>
<hello> Hello XML et hello chère lectrice ! </hello>
```

Exemple 3-3: Hello XML avec DTD externe

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE hello SYSTEM "hello.dtd">
<hello> Hello XèMèLè et hello cher lectrice ! </hello>
```

Exemple 3-4: Un fichier RSS (DTD externe public)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN"
 "http://my.netscape.com/publish/formats/rss-0.91.dtd">
<rss version="0.91">
<channel> ...... </channel>
</rss>
```


B. Déclaration du DTD (interne ou externe) dans le fichier XML

- Chaque déclaration du DTD commence par:
  ```
  <!DOCTYPE
  ```
- ... et fini par:
  ```
  >
  ```
- La racine de l'arbre XML (ici: `<hello>`) doit être indiquée après <!DOCTYPE
- Syntaxe pour définir un DTD interne (seulement !)
  - Le DTD sera inséré entre [...]
    ```
    <!DOCTYPE hello [
        <!ELEMENT hello (#PCDATA)>
                    ]>
    ```
- Syntaxe pour définir un DTD privé externe:
  - Le DTD est dans l'URL indiqué après le mot clef "SYSTEM".
  ```
  <!DOCTYPE hello SYSTEM "hello.dtd">
  ```
  - Important: Ne pas répeter la déclaration du DTD dans le fichier *.dtd !!

C. Définition du DTD

- Important: la racine de l'arbre XML doit être définie comme ELEMENT dans le DTD (normalement au début)
- Pour le reste: voir les sections suivantes

3.3 Déclaration d'éléments (tags)

Eléments DTD simples

```
Syntaxe: <!ELEMENT nom_du_tag spécification_contenu>
```

La spécification du contenu d'un élément contient soit une combinaison d'autres éléments, soit les éléments spéciaux #PCDATA, ANY, EMPTY (voir les exemples ci-après).

On peut combiner selon les règles ci-dessous:

A et B = tags	Explication specification_contenu
A?	A (un seul) est une option, (match A ou rien)
A+	Il faut un ou plusieurs A
A*	A est une option, il faut zero, un ou plusiers A
A | B	Il faut A ou B, mais pas les deux
A , B	Il faut A, suivi de B (dans l'ordre)
(A, B) +	Les parenthèses regroupent. Ici: un ou plusieurs (A suivi de B)

Eléments spéciaux

Elément spéciaux	Explication specification_contenu
#PCDATA	"Parsed Character Data" Données (non-interprétés par XML) dans le langage d'encodage courant.
ANY	Mot clé qui indique que tous les éléments sont autorisés (déconseillé)
EMPTY	Tag sans "closing" comme

Restriction sur les identificateurs (noms)

- doit commencer par une lettre ou '_'
- ensuite lettres, chiffres, '_' , '-', '.', ':'
  - J'ai constaté que certains outils digèrent très mal le "_" (à éviter donc !)

Comprendre/lire un DTD:

- pas trop difficile quand on a l'habitude
- Il faut partir de l'élément "root" (indiqué dans le DOCTYPE)
- Ensuite voir comment est définit chaque sous-élément, et ainsi de suite.

Exemple 3-5: Un DTD pour un simple Address Book

```
<!DOCTYPE addressBook [
 <!ELEMENT addressBook (person)+>
 <!ELEMENT person (name,email*)>
 <!ELEMENT name (family,given)>
 <!ELEMENT family (#PCDATA)>
 <!ELEMENT given (#PCDATA)>
 <!ELEMENT email (#PCDATA)>
]>
```

- Questions:
  - quelle est la racine?
  - c'est un DTD interne ou externe?

Exemple 3-6: Exemple d'un arbre XML valide

```
<addressBook>
  <person>
    <name>  <family>Wallace</family> <given>Bob</given> </name>
    <email>bwallace@megacorp.com</email>
  </person>

  <person>
    <name>  <family>Tuttle</family> <given>Claire</given> </name>
    <email>ctuttle@megacorp.com</email>
  </person>
</addressBook>
```

Exemple 3-7: Exemple d'un arbre XML invalide

```
<addressBook>
    <address>Derrière le Salève</address>
    <person>
    <name>
      <family>Schneider</family> <firstName>Nina</firstName>
    </name>
    <email>nina@dks.com</email>
  </person>
  <name>
   <family> Muller </family> </name>
</addressBook>
```

- Question:
  - quels sont les erreurs? (y en a trois :)

Exemple 3-8: Un DTD pour une recette

- Pour un exemple XML voir l'exemple 2-2 "Une recette en XML" [8]

```
<!DOCTYPE list [
 <!ELEMENT list (recipe+)>
 <!ELEMENT recipe (author, recipe_name, meal, ingredients, directions)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT recipe_name (#PCDATA)>
 <!ELEMENT meal (#PCDATA)>
 <!ELEMENT ingredients (item+)>
 <!ELEMENT item (#PCDATA)>
 <!ELEMENT directions (#PCDATA)>
]>
```

- Voir: **http://tecfa.unige.ch/guides/xml/examples/simple-dtd/choco-chip.xml**

Exercice 1: Lecture d'un DTD

- L'élément recette contient combien d'éléments ?
- Lequels de ces éléments sont optionels ?

3.4 Déclaration d'attributs

Attributs DTD simples (Voir la spec pour une définition "clean" !!):

```
Syntaxe: <!ATTLIST target_tag attr_nom TypeAttribut TypeDef Defaut>
```

	Explication de TypeAttribut
ID	Attribut unique dans le document
IDREF	Doit correspondré à un ID attribut dans un des éléments
IDREFS	Doit correspondre à 1 ou plusieurs ID attributs (séparés par des blancs)
(A|B|C|..)	Liste énumérée de valeurs d'attributs à choix
CDATA	"Character Data" - Contenu arbitraire, mais normalisé: espaces et fin de lignes convertis en un seul espace !
NMTOKEN	Un seul Mot

	Explication de TypeDef
#IMPLIED	Attribut à option
#REQUIRED	Attribut nécessaire
#FIXED Value	Attribut avec valeur fixe

Illustrations:

```
<!ATTLIST person prenom CDATA#REQUIRED
<!ATTLIST person gender (male|female) #IMPLIED>
 <!ATTLIST form method CDATA #FIXED "POST">
```



```
 <!ATTLIST list type (bullets|ordered) "ordered">
 <!ATTLIST sibling type (brother|sister) #REQUIRED>
<!ATTLIST person id ID #REQUIRED>
```

Attributs DTD multiples:

```
Syntaxe: <!ATTLIST   target_tag
            attr1_nom   TypeAttribut   TypeDef   Defaut
            attr2_nom   TypeAttribut   TypeDef   Defaut
            ...
        >
```

Illustrations:

```
<!ATTLIST person
    gender      (male|female)    #IMPLIED
    nom         CDATA            #REQUIRED
    prenom      CDATA            #REQUIRED
    relation    (brother|sister) #REQUIRED
>
```


Exemple 3-9: Un DTD pour un moins simple Address Book

[fichier ab.dtd]

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name,email*)>
<!ATTLIST person id ID #REQUIRED>
<!ATTLIST person gender (male|female) #IMPLIED>
<!ELEMENT name (#PCDATA|family|given)*>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT link EMPTY>
<!ATTLIST link manager IDREF #IMPLIED subordinates IDREFS #IMPLIED>
```

Exemple:

```
<!DOCTYPE addressBook SYSTEM "ab.dtd">
<addressBook>
  <person id="B.WALLACE" gender="male">
    <name>
      <family>Wallace</family> <given>Bob</given>
    </name>
    <email>bwallace@megacorp.com</email>
    <link manager="C.TUTTLE"/>
  </person>
  <person id="C.TUTTLE" gender="female">
    <name>
      <family>Tuttle</family> <given>Claire</given>
    </name>
    <email>ctuttle@megacorp.com</email>
    <link subordinates="B.WALLACE"/>
  </person>
</addressBook>
```


3.5 Attributs vs. Elements

- Il s'agit ici une grande FAQ sans réponse précise
- Ci-dessous quelques réflexions à pondérer ....

Il faut plutôt utiliser un élément

- lorsque l'ordre est important (l'ordre des attributs est au hasard)
- lorsqu'on veut réutiliser un élément plusieurs fois (avec le même parent)
- lorsqu'on veut (dans le futur) avoir des descendants / une structure interne
- pour représenter un type de données (objet) plutôt que son usage, autrement dit: une "chose" est un élément et ses propriétés sont des "attributs".
- lorsque XML sert comme markup pour un texte à publier (tout ce que le lecteur devra voir se trouvera dans un élément, tout ce qui est "meta" dans attributs)

Il faut plutôt utiliser un attribut

- lorsqu'on désire faire référence à un autre élément (parent="giraffe") et (cat="giraffe") dans l'élément référencé
- pour indiquer l'usage/type/etc. d'un élément comme dans: <address usage="prof"> ... </address>
- lorsque vous voulez imposer des valeurs par défaut dans le DTD
- lorsque vous voulez un type de données (pas grand chose dans le DTD)

3.6 Déclaration d'Entités

- Une "entity" est un bout d'information stocké quelque part.
- Seulement 5 entités sont prédéfinis (toutes pour les signes spéciaux):

Entity	Signe
&	&
<	<
>	>
"	"
'	'

- Les autres entités doivent être définies par l'auteur du DTD
  - soit dans le même fichier, soit à l'extérieur (comme pour les DTD)

	Explications
SYSTEM	Le contenu de l'entité est accessible par un URI
.....	......

A. Entités générales

```
Syntaxe: <!ENTITY   nom_du_tag   "contenu">
```

Illustrations:

```
<!ENTITY tecfaUnit "Unité de technologies de formation et apprentissage">
<!ENTITY tecfaDesc SYSTEM "http://tecfa.unige.ch/../tecfa_description.xml">
```

Référence à une entité générale (inclusion)

- servent partout.
- Exemples d'entités déclarées:

```
<!ENTITY pm "Patrick Mendelsohn">
<!ENTITY acirc  "&#194;">
<!ENTITY espace " ">
<!ENTITY copyright "&#xA9;">
<!ENTITY explication SYSTEM "project1a.xml">
```

Ensuite, on fait une référence du style "&nom_paramètre;":

```
<para> &pm; sort du ch&acirc;teau <para>
va donner:
<para> Patrick Mendelsohn sort du ch&acirc;teau,
s'arrête devant le panneau et lit:&espace;
<citation> &explication </citation>
</para>
```


B. Entités "paramétriques"

- Servent à faire des DTD
- Exemple:

```
<!ENTITY % stamp '
  id ID #IMPLIED
  creation-day NMTOKEN #IMPLIED
  .......
  mod-by NMTOKEN #IMPLIED
  version NMTOKEN #IMPLIED
  status (draft|final|obsolete) #IMPLIED
  approval (ok|not-ok|so-so) #IMPLIED
  main-author CDATA #IMPLIED
'
>
```

Usage: on fait une référence du style "%nom_paramètre;":

```
<!ELEMENT main-goal (title, content, (after-thoughts)?, (teacher-
comments)?)>
<!ATTLIST main %stamp; >
```


C. Annexe: La définition de XML

- XML (comm d'autres grammaires en informatique) est décrit sous format EBNF (Extended Backus-Naur Form)
  - Comprendre EBNF est nécessaire pour bien comprendre la spécification de XML (mais ce n'est pas une nécessité absolue)
  - Voir: the Annotated XML Specification, **http://www.xml.com/axml/axml.html**

Exemple 3-10: Quelques éléments de XML à titre d'illustration

```
[1] document ::= prolog element Misc
[2] element  ::= EmptyElemTag | STag content ETag [WFC: Element Type Match]
                 [VC: Element Valid]
```

4. Utilisation de psgml/Xemacs

- Note: il existe une logique assez similaire pour les "vrais" éditeurs XML. Si XEmacs ne convient pas, il faut absolument utiliser un outil qui sait "utiliser" un DTD pour vous aider à rédiger comme Xerlin (gratuit), XML Spy, XML Writer, etc.).

4.1 Installation et activation

- psgml est un "module" XEmacs pour faire du SGML et XML
- Le module psgml est normalement compris dans une installation complète
- Il se charge automatiquement pour les fichiers *.xml
  - Si ce n'est pas le cas, taper: ALT-x xml-mode
    (pour les fichier *.html qui contiennent XHTML par exemple, note: sur notre serveur les fichiers *.xhtml sont servis comme "XML" (pas lisibles pour les anciens navigateurs)

4.2 Utilisation d'un DTD privé

- copier le DTD dans le même répertoire
- et utiliser la déclaration de type suivante:
  `<!DOCTYPE hello SYSTEM "hello.dtd">`

4.3 Utilisation d'un DTD public

- Avantage: le client sait interpréter une déclaration publique et Xemacs aussi

Xemacs connaît déjà certains DTD,

- vous pouvez vérifier en ouvrant un fichier *.xml en question
- ou en regardant dans le fichier CATALOG

Insertion d'un nouveau DTD dans le système:

1. Il faut localiser le fichier CATALOG, il se trouver à un endroit comme celui-ci:
   `c:\Program Files\XEmacs\xemacs-packages\etc\psgml-dtds\CATLOG`
2. Il faut ajouter le type de lignes suivantes dans le fichier CATALOG:
   - Exemple Windows:

```
PUBLIC "-//TopicMaps.Org//DTD XML Topic Map (XTM) 1.0//EN" xtm1.dtd
```

   - Exemple Unix:

```
PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN" /web/lib/xml/dtd/rss-
0.91.dtd
PUBLIC "-//TECFA//DTD Stepbystep 0.3//EN" /web/lib/xml/dtd/stepbystep03.dtd
PUBLIC "-//W3C//DTD SVG 1.0//EN" /web/lib/xml/dtd/svg10.dtd
```

   - La syntaxe est simple:

```
PUBLIC "nom du DTD" chemin-local-du-fichier.dtd
```

   On vous conseille de mettre les DTD soit dans un répertoire spécial, soit au même endroit que CATALOG.

4.4 Retrouver des DTDs publiques

- Le site **http://www.w3.org/**
- Les organisations responsables des différents DTD
- Regarder dans un fichier XML en question ("View Source")
- Note: pour retrouver le nom public d'un DTD on conseille de trouver un exemple, ensuite copier la déclaration.

Note: A Tecfa, on met certains DTD ici:
url: **http://tecfa.unige.ch/lib/xml/dtd/**

4.5 Commandes XEmacs/psgml de base

ALT-X xml-mode
met l'éditeur en mode XML

DTD->Parse DTD
relit le DTD (utile si vous faites des changements dans le DTD et/ou si vous déclarez le DTD plus tard)

Move->Next trouble spot
affiche la prochaine erreur (à partir de la position du curseur)
Note: il s'agit d'une validation "light", XEmacs ne trouve pas tout.

4.6 Quelques messages d'erreur

p end-tag implied by p start-tag

p end-tag implied by h1 start-tag

- Veux dire que Xemacs à trouvé une balise <p> (ou <h1>) alors qu'il aurait du trouver une balise </p>. Autrement dit, on a oublié de fermer un paragraphe.

head element can't end here, need one of (title base)

ul element can't end here, need one of (li)

- Veux dire qu'il manque un élément dans <head> ... </head>, en occurence soit "title", soit "base". Pareil pour "ul" et "li" ...

Out of context bla tag

- Veut dire qu'il ne peut pas avoir un <bla> ... </bla> à cet endroit.