

Introduction technique à XML

Code: xml-tech

Originaux

[url: http://tecfa.unige.ch/guides/tie/html/xml-tech/xml-tech.html](http://tecfa.unige.ch/guides/tie/html/xml-tech/xml-tech.html)

[url: http://tecfa.unige.ch/guides/tie/pdf/files/xml-tech.pdf](http://tecfa.unige.ch/guides/tie/pdf/files/xml-tech.pdf)

Auteurs et version

- [Daniel K. Schneider](#) - [Vivian Synteta](#) - [Stephane Lattion](#)
- Version: 2.3 (modifié le 2/10/07 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](#) (IMPORTANT!)

Modules suivants

Module technique suivant: xml-xslt (Transformations XSLT)

Module technique suivant: xml-xslfo (Styles XSLFO)

Module technique suivant: xml-rss (Syndication)

Module technique suivant: xml-ser (server-side XML, à refaire)

Module technique suivant: php-xml (à refaire), voir [php-intro](#) pour le moment

Module technique suivant: java-xml (pour programmeurs, à refaire)

Objectifs

- Initiation au formalisme XML
- Savoir faire de simples grammaires DTD
- Savoir rédiger des textes XML simples (RSS, simple récit, etc.)
- Comprendre la logique de XHTML (par rapport à HTML)

1. Table des matières détaillée

1. Table des matières détaillée	3
2. Le langage XML	5
2.1 Les notions de “well-formed” et de “valid”	6
2.2 Exemples XML de sensibilisation	8
Exemple 2-1:Bonjour en XML	8
Exemple 2-2:Une recette en XML	9
Exemple 2-3:RSS	10
Exemple 2-4:Une grammaire (simple) de récit	11
2.3 Les “espaces de noms” (name spaces)	12
3. Les DTD	13
3.1 Introduction aux DTD de XML	13
3.2 Association d’une DTD avec un fichier XML	14
Exemple 3-1:Hello XML sans DTD	16
Exemple 3-2:Hello XML avec DTD interne	16
Exemple 3-3:Hello XML avec DTD externe	16
Exemple 3-4:Un fichier RSS (DTD externe public)	16
3.3 Déclaration d’éléments (tags)	17
Exemple 3-5:Une DTD pour un simple Address Book	19
Exemple 3-6:Exemple d’un arbre XML valide	20
Exemple 3-7:Exemple d’un arbre XML invalide	20
Exemple 3-8:Une DTD pour une recette	21
3.4 Déclaration d’attributs	22
Exemple 3-9:Une DTD pour un Address Book plus complexe	24
3.5 Attributs vs. Elements	25
3.6 Déclaration d’Entités	26
Exemple 3-10:Exemple DTD avec entités incluses dans une DTD	29
Exemple 3-11:Quelques éléments de XML à titre d’illustration	30
4. Exemples	31
4.1 Exemple “text-centric”	32
Exemple 4-1:Une grammaire pour un simple récit	32
4.2 Exemple plutôt “data-centric” (malgré l’absence d’attributs)	34

Exemple 4-2:DTD pour une grammaire de "pages travaux" 34	
4.3 Exemple très "data-centric"	36
Exemple 4-3:Un fragment SVG (sans la DTD) 36	
5. Utilisation de psgml/Xemacs	37
5.1 Installation et activation	37
5.2 Utilisation d'une DTD privée	37
5.3 Utilisation d'une DTD publique	38
5.4 Retrouver des DTDs publiques	39
5.5 Commandes XEmacs/psgml de base	39
5.6 Quelques messages d'erreur dans Emacs	40
6. Validation	41
6.1 Valideurs en ligne	41
6.2 Valideurs à installer localement	41

2. Le langage XML

- XML = Extensible Markup Language
- XML est un formalisme qui structure certains types de contenus
- Différents formalismes comme les DTD, ou XSD, ou Relax définissent ces applications XML (“langages”, grammaires)

Définition un peu plus formelle:

- XML est un langage pour définir une classe d’objets de données, par exemple:
 - les éléments d’un dessin vectoriel (SVG)
 - les éléments d’une page Web (XHTML)
 - les éléments d’un récit
- Un document XML contient:
 - déclarations, éléments, commentaires, définition de caractères spéciaux et instructions (facultatives) de traitement.
- Un document est un “arbre” (“boites dans des boites”):
 - Chaque document doit avoir une racine et les éléments doivent s’imbriquer proprement (voir -A. ““Well-formed” XML documents (pages correctement formées)” [6]
- Pour la définition formelle de XML, voir la spécification
[url: http://www.w3.org/TR/REC-xml/](http://www.w3.org/TR/REC-xml/) (documentation très technique!)

2.1 Les notions de “well-formed” et de “valid”

A. “Well-formed” XML documents (pages correctement formées)

- Le document commence par une déclaration XML (l'attribut version est obligatoire)

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

- possibilité de choisir un encodage (le défaut est utf-8):

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

- Structure hiérarchique:

- begin-tags (balises d'ouverture) et end-tags (balises de fermeture) doivent correspondre
- pas de croisements de type `<i>......</i> `
- Case sensitivity, donc "LI" n'est pas égal à "li" par exemple

- Balise de type "EMPTY" (balises sans balises de fermeture ...):

- Ces balises sans contenu utilisent la syntaxe XML "auto-fermante" (par ex.. `
`)

- Les valeurs d'attributs sont entre guillemets (quoted):

- (par ex.. ``)

- 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 `<`; `&`; `>`; `"`; `'`; à la place de: `<`, `&`, `>`, `"`, `'`

- C'est également valable pour les URLs !!

```
http://truc.unige.ch/programme?bla&machin
```

devient

```
http://truc.unige.ch/programme?bla&amp;machin
```

B. “Valid XML documents”

- Un document “valide” doit être:
 - “well-formed” (bien formé, formé correctement)
 - être associé à une DTD (ou une autre grammaire)
 - et être conforme à cette DTD (ou un autre type de grammaire)

C. DTD (Document Type Definition)

Une DTD est:

- Une grammaire (ou jeu de règles) qui :
 - définit le jeux de balises utilisables ainsi que leurs attributs,
 - et qui spécifie leur possible imbrication (relation).
 - La DTD peut être référencée par un URI ou incluse directement dans le document XML
- Ils existent d'autres types de grammaires comme XML Schema (XSD), Relax NG, etc.
 - Leur puissance sémantique est plus élevée (c.à.d que l'on peut exprimer plus de contraintes)
 - Relax NG offre le meilleur rapport puissance/facilité.
 - DTD est la plus répandue
 - XML Schema est utilisé par ex. pour formaliser des langages "webservices", par ex. SOAP
 - XML Schema et Relax ne seront pas abordés dans ce document.
 - Pas mal d'experts XML détestent XML Schema et militent pour Relax NG ...

2.2 Exemples XML de sensibilisation

Exemple 2-1: Bonjour en XML

[url: http://tecfa.unige.ch/guides/xml/examples/simple/](http://tecfa.unige.ch/guides/xml/examples/simple/)

Données XML

- On a un document de type <page>

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

La DTD (à titre d'indication) qui validerait ce document pourrait être:

- Voir 3. "Les DTD" [13] 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/98/41/index1a_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>
```

- Cet exemple montre quelques éléments d'un vocabulaire pour recettes
- Pour la DTD, voir exemple 3-8 "Une DTD pour une recette" [21]

Exemple 2-3: RSS

url: <http://tecfa.unige.ch/tecfa/teaching/staf10/rss/example/>

- Le(s) standard(s) RSS permettent de parler titre et résumés de nouvelles entre portails et weblogs. Ce mécanisme peut aussi être utilisé "manuellement".

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

<!DOCTYPE rss SYSTEM "rss-0.91.dtd">
<rss version="0.91">
  <channel>
    <title>Vivians island</title>
    <description>Project-Based e-Learning</description>
    <link>http://tecfa.unige.ch/perso/vivian/</link>
    <item>
      <title>Vive les poissons1</title>
      <description>blabla bla bla ...</description>
      <link>http://tecfa.unige.ch/perso/vivian/</link>
    </item>
    <item>
      <title>Vive les poissons3</title>
      <description>blabla bla bla ...</description>
      <link>http://tecfa.unige.ch/perso/vivian/</link>
    </item>
  </channel>
</rss>
```

Exemple 2-4: Une grammaire (simple) de récit

[url: http://tecfa.unige.ch/guides/xml/examples/recit/](http://tecfa.unige.ch/guides/xml/examples/recit/)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="recit.css" type="text/css" ?>
<!DOCTYPE RECIT SYSTEM "recit.dtd">

<RECIT xmlns:xlink="http://www.w3.org/1999/xlink">
  <Titre>Le garçon webmestre</Titre>
  <Contexte>Il était une fois un garçon ni joli ni moche, ni bête ni intelligent,
ni drôle ni ennuyeux.
  </Contexte>
  <Probleme>Il était assez heureux dans sa vie de webmaster pour l'office de la promotion
des technologies anciennes. Toutefois, il lui manquait une amie.
  </Probleme>
  <But>Il fallait que cela change ! </But>
  <FIL>
    <EPISODE>
      <SousBut>Un jour il s'est dit qu'il doit agir coûte que coûte.</SousBut>
      <TENTATIVE>
        <Action>Il s'est rendu au pub du coin. Arrivé au bar il voit un ancien camarade de
classe accompagné de deux filles. Ils commencent à discuter et quand il raconta qu'il
était WebMaster, une des filles lui demande ce qu'il faisait. Alors il expliqua fièrement
qu'il faisait des pages HTML avec Frontier. Et avec un élan de courage il demanda à la
fille s'il pouvait lui offrir un verre.</Action>
      </TENTATIVE>
    ..... </RECIT>
```

2.3 Les “espaces de noms” (name spaces)

- Il est possible dans un même document de mélanger plusieurs grammaires (si l'application le permet), par exemple: XHTML + SVG + MathML + XLink.
- Pour éviter qu'il y ait confusion entre différentes balises on doit définir un “namespace” pour chaque grammaire ajoutée à l'endroit où on l'utilise
- Un namespace est identifié par un URL qui est *juste un nom* (l'URL peut être sans contenu)

Exemple d'utilisation de SVG

- `xmlns="..."` indique que *toutes* les balises suivantes sont du domaine SVG

```
<svg xmlns="http://www.w3.org/2000/svg">  
  <rect x="50" y="50" rx="5" ry="5" width="200" height="100" ....
```

- `xmlns:svg="..."` veut dire que toutes les balises préfixées par `svg:` sont du domaine SVG

```
<html xmlns:svg="http://www.w3.org/2000/svg">  
  <svg:rect x="50" y="50" rx="5" ry="5" width="200" height="100" ....
```

Exemple d'utilisation de XLink (qui est une grammaire pour définir des liens):

```
<RECIT xmlns:xlink="http://www.w3.org/1999/xlink">  
<INFOS>  
  <Date>30 octobre 2003 - </Date><Auteur>DKS - </Auteur>  
  <A xlink:href="http://jigsaw.w3.org/css-validator/check/referer"  
    xlink:type="simple">CSS Validator</A>  
</INFOS>
```

- `xmlns:xlink="http://www.w3.org/1999/xlink">` veut dire que toutes les balises qui commencent par `xlink:` font appel au standard XLink (identifié par un URL)

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) ou RELAX NG

Une DTD est une grammaire qui définit:

1. les balises (tags) possibles et leurs attributs
2. L'imbrication des balises à l'intérieur d'autres balises
3. Quels balises et attributs sont à option et lesquels sont obligatoires

Chaque balise XML est défini une seule fois comme un élément dans la DTD

- Exemple illustratif:

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

3.2 Association d'une DTD avec un fichier XML

A. Déclaration d'une DTD dans un fichier XML

Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE hello SYSTEM "hello.dtd">
```

Il existe 4 façons d'utiliser une DTD

1. On ne déclare pas de DTD (dans ce cas le fichier est juste "bien formé")
2. On déclare la DTD et on y ajoute les définitions dans le même fichier (DTD interne)
 - On parle dans ce cas d'un XML "standalone" (le fichier XML se suffit à lui-même)
3. On déclare la DTD en tant que DTD "privée", la DTD se trouve quelque part dans votre système ou sur Internet
 - répandu pour les DTDs "faites maison"
4. On déclare une DTD "public", c.a.d. on utilise un nom officiel pour la DTD.
 - cela présuppose que votre éditeur et votre client connaissent cette DTD
 - répandu pour les DTDs connues comme XHTML, SVG, MathML, etc.

Lieu de la déclaration

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

B. Syntaxe de la déclaration

- Chaque déclaration de la DTD commence par:

```
<!DOCTYPE
```

- ... et fini par:

```
>
```

- La racine de l'arbre XML (ici: <hello>) doit être indiquée après <!DOCTYPE
- Syntaxe pour définir une DTD interne (seulement !)
 - La DTD sera insérée entre [...]

```
<!DOCTYPE hello [  
    <!ELEMENT hello (#PCDATA)>  
    ]>
```

- Syntaxe pour définir une DTD privée externe:

- La DTD est dans l'URL indiqué après le mot clef "SYSTEM".

```
<!DOCTYPE hello SYSTEM "hello.dtd">
```

- Important: Ne pas répéter la déclaration de la DTD dans le fichier *.dtd !!
- Dit autrement: La déclaration de la DTD se fait dans le fichier XML et PAS dans le fichier *.dtd. Ce fichier ne définit que les règles ...

C. Définition de la racine de l'arbre

- Le mot "hello" après le mot clef DOCTYPE indique que "hello" est l'élément racine de l'arbre XML.

```
<!DOCTYPE hello SYSTEM "hello.dtd">
```

- Important: la racine de l'arbre XML doit être définie comme ELEMENT dans la DTD
 - normalement on la met au début, mais ce n'est pas une obligation

Quelques exemples:

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>
```

3.3 Déclaration d'éléments (tags)

Rappel du principe:

- On doit définir **chaque élément** que l'on pense utiliser dans les documents
- On doit indiquer comment les éléments s'imbriquent
- Chaque élément ne se définit qu'une seule fois !!

Exemple de sensibilisation:

```
<!ELEMENT name (family,given)>  
<!ELEMENT family (#PCDATA)>
```

veut dire que:

- l'élément "name" a deux éléments enfants: family et given
- l'élément "family" ne contient que du texte (pas d'autres éléments)

En utilisant un "schéma":

```
name      ==> family + given  
family    ==> "texte"
```

Syntaxe de la définition d'un élément:

Syntaxe: `<!ELEMENT nom_balise spécification_contenu>`

La spécification du contenu d'un élément contient:

- soit une combinaison d'autres éléments,
- soit une combinaison d'autres éléments plus des éléments spéciaux #PCDATA, ANY (contenu mixte)
- soit l'élément #PCDATA, ou ANY
- soit EMPTY

Règles de combinaison: On peut combiner selon les règles ci-dessous:

A et B = tags	Explication specification_contenu	Exemples
A?	A (un seul) est une option, (donc: A ou rien)	<!ELEMENT person (name, email?)
A+	Il faut un ou plusieurs A	<!ELEMENT person (name, email+)
A*	A est une option, il faut zéro, un ou plusieurs A	<!ELEMENT person (name, email*)
A B	Il faut A ou B, mais pas les deux	<!ELEMENT person (email fax)
A , B	Il faut A, suivi de B (dans l'ordre)	<!ELEMENT person (name , email?)
(A, B) +	Les parenthèses regroupent. Ici: un ou plusieurs (A suivi de B)	<!ELEMENT liste (name, email)+

Éléments spéciaux

Élément spéciaux	Explication specification_contenu	Exemples
#PCDATA	"Parsed Character Data" Données (non-interprétées par XML) dans le langage d'encodage courant.	<!ELEMENT email (#PCDATA)>
ANY	Mot clé qui indique que tous les éléments sont autorisés (déconseillé)	<!ELEMENT person ANY>
EMPTY	Balise auto-fermante comme dans 	<!ELEMENT br EMPTY>

Restriction sur les identificateurs (noms)

- chaque identificateur doit commencer par une lettre ou '_'
- ensuite lettres, chiffres, '_', '-', '.', ':'

Comprendre/lire une DTD:

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

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

```
<!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 qu'on utilisera vraisemblablement ?
 - pour que le document soit valide il faut combien d'éléments "person" au moins ?
 - est-ce une DTD interne ou externe? (question piège)

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:

- quelles sont les erreurs? (il y en a trois)

Exemple 3-8: Une DTD pour une recette

- Pour un exemple XML, voir l'exemple 2-2 "Une recette en XML" [9]

```
<!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'une DTD

- L'élément recette contient combien d'éléments ?
- Lesquels 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>`

- Vous avez un contrôle limité sur le type de contenus autorisés pour un attribut

	Types d'attributs (TypeAttribut)
ID	Permet de définir un identificateur unique pour un élément du document. Donc chaque id doit être différent ! Exemple d'usage: faire des tables de matière.
IDREF	Doit correspondre à un attribut "ID" dans un des éléments du document. Voir ci-dessus.
IDREFS	Doit correspondre à 1 ou plusieurs ID attributs (séparés par des blancs)
(A B C ..)	Liste énumérée de valeurs d'attributs à choix. Vous permet de contrôler un petit peu le contenu que les utilisateurs peuvent rentrer.
CDATA	"Character Data" - Contenu arbitraire, mais normalisé: espaces et fin de lignes convertis en un seul espace !
NMTOKEN	L'utilisateur peut rentrer un seul mot

	Explication de TypeDef
#IMPLIED	Attribut à option (l'utilisateur peut l'utiliser)
#REQUIRED	Attribut obligatoire (l'utilisateur doit rentrer une valeur)
#FIXED Value	Attribut avec valeur fixe (la valeur est déjà fixée dans la DTD)

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
  ident      ID          #REQUIRED
  gender     male|female) #IMPLIED
  nom        CDATA       #REQUIRED
  prenom     CDATA       #REQUIRED
  relation   (brother|sister) #REQUIRED >

<!ATTLIST portable
  proprio   IDREF        #REQUIRED >
```

Exemple 3-9: Une DTD pour un Address Book plus complexe

[contenu du 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 d'un XML valide par rapport aux règles dans ab.dtd:

```
<!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
 - `<compagnon nom="lisa" genre="giraffe">` fait référence à `<animal cat="giraffe">`
- 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.
- Lors de l'utilisation d'un document, les entités sont remplacés par le contenu référencé

Seulement 5 entités sont prédéfinies (toutes pour les signes spéciaux)

- on vous rappelle à cette occasion qu'il faut toujours utiliser ces entités (même dans les URL qu'on utilise avec XHTML !)

Entity	Signe
&amp;	&
&lt;	<
&gt;	>
&quot;	"
&apos;	'

Les autres entités doivent être définies par l'auteur de la DTD

- soit dans la DTD elle-même
- soit par une déclaration qui renvoie à un contenu défini ailleurs (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">
<!ENTITY pm "Patrick Mendelsohn">
<!ENTITY acirc "Â">
<!ENTITY espace " ">
<!ENTITY copyright "©">
<!ENTITY explication SYSTEM "project1a.xml">
```

Référence à une entité générale (simple substitution)

`<para> ± sort du château, s'``<para>`
va donner:

`<para> Patrick Mendelsohn sort du chÂteau</para>`

Inclusion d'un contenu d'un autre fichier:

```
<para> blabla
<citation> &explication; </citation>
.... </para>
```

va donner:

```
<para> blabla
<citation> .... tout le contenu du fichier project1a.xml... </citation>
.... </para>
```

B. Entités "paramétriques"

- Aident à fabriquer des DTD complexes
- 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: Les listes d'attributs ci-dessous contient tous les attributs définis dans l'entité %stamp;

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

- on évite donc de taper plusieurs fois les mêmes déclarations.

Exemple 3-10: Exemple DTD avec entités incluses dans une DTD

url: <http://tecfa.unige.ch/lib/xml/dtd/ePBL11/>

- Il faut déclarer la DTD à inclure comme une ENTITIY externe et ensuite l'inclure.

Voici montré avec un exemple:

- Dans le cours staf-18 les étudiants uilisent la DTD "ePLpaper11.dtd" pour rédiger leur papier. Cette DTD inclut la DTD ibtwsh6.dtd et en utilise une partie:

```
<! ENTITY % foreign-dtd SYSTEM "ibtwsh6_ePBL.dtd" >
%foreign-dtd;
```

- ibtwsh6_ePBL.dtd est un sous-ensemble de XHTML qu'on utilise ici pour permettre la mise en forme à l'intérieur d'éléments XML "sémantiques". %foreign-dtd; va l'importer.
- Cette DTD a plusieurs "racines" définies.
 - %vert.model et %struct.model correspondent à une série de balises
 - Si on décide d'utiliser un modèle "très ouvert":

```
<!ELEMENT introduction %vert.model;>
```

- A Tecfa, on préfère %struct.model:

```
<!ELEMENT introduction %struct.model;>
```

```
<!ELEMENT conclusion %struct.model;>
```

Activité:

- Essayez de voir ce que font vert.model et struct.model en examinant le fichier ibtwsh6_ePBL.dtd.

C. Annexe: La définition de XML

- XML (comme beaucoup 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)

Exemple 3-11: 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. Exemples

Souvent, on distingue entre DTDs "text-" ou "data-centric":

A. Les DTD "text-centric"

- sont destinées à produire des textes selon certaines normes sémantiques ou "typographiques".
- Idéalement, les utilisateurs entrent le texte avec un éditeur XML (semi-)wysiwyg
- Une DTD "text-centric" utilise très peu d'attributs. En règle générale, tout le contenu à afficher se trouve dans des éléments. Marche bien avec CSS ou XSLT
- Exemples:
 - La grammaire "sémantique" du récit ci-dessous, RSS
 - DocBook, XHTML, etc pour des normes "typographiques"

B. Les DTD "data-centric"

- sont surtout destinées au traitement par des machines, souvent présence de beaucoup d'attributs
- Edition avec un éditeur d'abres XML (ou un outil spécialisé), affichage après traitement XSLT "lourd" ou avec un programme spécialisé
- Exemples:
 - SVG, TopicMaps, OWL, RDF, etc.

4.1 Exemple "text-centric"

Exemple 4-1: Une grammaire pour un simple récit

[url: http://tecfa.unige.ch/guides/xml/examples/recit/](http://tecfa.unige.ch/guides/xml/examples/recit/)

```
<!ELEMENT RECIT (Titre, Contexte, Probleme, But, FIL, Morale, INFOS)>
<!ATTLIST RECIT xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">

<!ELEMENT FIL (EPISODE+)>
<!ELEMENT EPISODE (SousBut, TENTATIVE+, Resultat) >
<!ELEMENT TENTATIVE (Action | EPISODE) >
<!ELEMENT INFOS ( ( Date | Auteur | A )* ) >
<!ELEMENT Titre (#PCDATA) >
<!ELEMENT Contexte (#PCDATA) >
<!ELEMENT Probleme (#PCDATA) >
<!ELEMENT But (#PCDATA) >
<!ELEMENT SousBut (#PCDATA) >
<!ELEMENT Resultat (#PCDATA) >
<!ELEMENT Morale (#PCDATA) >
<!ELEMENT Action (#PCDATA) >
<!ELEMENT Date (#PCDATA) >
<!ELEMENT Auteur (#PCDATA) >
<!ELEMENT A (#PCDATA)>
<!ATTLIST A xlink:href CDATA #REQUIRED xlink:type CDATA #FIXED "simple"
```

Inspiration: Thorndyke, P.W., "Cognitive structures in comprehension and memory of narrative discourse", *Cognitive Psychology* 9 (1977) 77-110. (version simplifiée ici !)

Exemple d'un document (il manque une partie!)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="recit.css" type="text/css" ?>
<!DOCTYPE RECIT SYSTEM "recit.dtd">

<RECIT xmlns:xlink="http://www.w3.org/1999/xlink">
  <Titre>Le garçon webmestre</Titre>
  <Contexte>Il était une fois un garçon ni joli ni moche, ni bête ni intelligent, ni drôle ni ennuyeux.
</Contexte>
  <Probleme>Il était assez heureux dans sa vie de webmaster pour l'office de la promotion des technologies
anciennes. Toutefois, il lui manquait une amie.
</Probleme>
  <But>Il fallait que cela change !</But>
  <FIL>
    <EPISODE>
      <SousBut>Un jour il s'est dit qu'il doit agir coûte que coûte.</SousBut>
      <TENTATIVE> <Action>Il s'est rendu au pub du coin. Arrivé au bar il voit un ancien camarade de classe
accompagné de deux filles. Ils commencent à discuter et quand il raconta qu'il était WebMaster, une des
filles lui demande ce qu'il faisait. Alors il expliqua fièrement qu'il faisait des pages HTML avec Frontier.
Et avec un élan de courage il demanda à la fille s'il pouvait lui offrir un verre.</Action> </TENTATIVE>
      <Resultat>La fille lui répondit: "Non merci". Un peu désespéré le garçon rentra chez lui.</Resultat>
    </EPISODE>
    .....
  </FIL>
  <Morale> La morale de l'histoire est que HTML ne suffit plus. </Morale>
  <INFOS> <Date>30 octobre 2003 - </Date><Auteur>DKS - </Auteur>
  <A xlink:href="http://jigsaw.w3.org/css-validator/check/referer" xlink:type="simple">CSS Validator</A>
</INFOS>

</RECIT>
```

4.2 Exemple plutôt "data-centric" (malgré l'absence d'attributs)

Exemple 4-2: DTD pour une grammaire de "pages travaux"

[url: http://tecfa.unige.ch/tecfa/teaching/staf14/files/workpage/workpage.html](http://tecfa.unige.ch/tecfa/teaching/staf14/files/workpage/workpage.html)

```
<!-- ***** Students ***** -->
<!ELEMENT student ( personal-data, courses )>

<!-- ***** Personal-data ***** -->
<!ELEMENT personal-data (first-name, family-name, homepage-url, email, promotion, unix-
login, moo-login)>

<!ELEMENT first-name      (#PCDATA)>
<!ELEMENT family-name    (#PCDATA)>
<!ELEMENT homepage-url   (#PCDATA)>
<!ELEMENT email          (#PCDATA)>
<!ELEMENT promotion      (#PCDATA)>
<!ELEMENT unix-login     (#PCDATA)>
<!ELEMENT moo-login      (#PCDATA)>

<!-- ***** Exercise(s) ***** -->
<!ELEMENT courses ((course)*)>
<!ELEMENT course (title, (url)?, (instructor)?, (exercise)+)>
<!ELEMENT exercise (exercise-number,
                    title,
                    sections
                    )>
<!ELEMENT exercise-number (#PCDATA)>

<!ELEMENT sections ((section)+)>
```

```
<!ELEMENT section (deposit-date,
                    title,
                    url,
                    description,
                    (status)?,
                    (participants)?,
                    (comments)?
                    )>

<!ELEMENT deposit-date (#PCDATA)>
<!ELEMENT url           (#PCDATA)>
<!ELEMENT status        (#PCDATA)>
<!ELEMENT participants  (#PCDATA)>
<!ELEMENT comments      (#PCDATA | p | ol)*>

<!-- ***** Common Items ***** -->
<!ELEMENT title (#PCDATA)>
<!ELEMENT description (#PCDATA | p | ol)*>
<!ELEMENT ol (li)+>
<!ELEMENT li (#PCDATA)>
<!ELEMENT p (#PCDATA)>
```

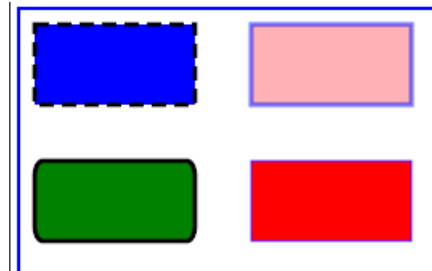
Note: Les contenus sont affichés avec XSLT.

4.3 Exemple très "data-centric"

Exemple 4-3: Un fragment SVG (sans la DTD)

[url: http://tecfa.unige.ch/guides/tie/code/svg-intro/shapes/rectangles1.svg](http://tecfa.unige.ch/guides/tie/code/svg-intro/shapes/rectangles1.svg)

```
<?xml version="1.0" standalone="no"?>
<svg width="270" height="170" xmlns="http://www.w3.org/2000/svg">
  <rect x="5" y="5" width="265" height="165"
        style="fill:none;stroke:blue;stroke-width:2" />
  <rect x="15" y="15" width="100" height="50" fill="blue"
        stroke="black" stroke-width="3" stroke-dasharray="9 5"/>
  <rect x="15" y="100" width="100" height="50"
        fill="green" stroke="black" stroke-width="3" rx="5" ry="10"/>
  <rect x="150" y="15" width="100" height="50" fill="red"
        stroke="blue" stroke-opacity="0.5" fill-opacity="0.3" stroke-width="3"/>
  <rect x="150" y="100" width="100" height="50"
        style="fill:red;stroke:blue;stroke-width:1"/>
</svg>
```



5. 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 Exchanger lite, Xerlin ou Merlot (gratuits), Oxygen, XML Spy, XML Writer, etc.).

5.1 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))

5.2 Utilisation d'une DTD privée

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

5.3 Utilisation d'une DTD publique

- Avantage: si le client (Mozilla, Xemacs, etc.) sait interpréter une déclaration, on n'est pas obligé de renvoyer à un fichier DTD

Xemacs connaît déjà certaines DTD,

- vous pouvez vérifier en regardant dans le fichier CATALOG

Insertion d'une nouvelle DTD dans le système:

1. Il faut localiser le fichier CATALOG, il se trouve 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 simple:

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

5.4 Retrouver des DTDs publiques

- Le site <http://www.w3.org/>
- Les organisations responsables des différentes DTD
- Regarder dans un fichier XML en question (“View Source”)
- Note: pour retrouver le nom public d’une DTD on conseille de trouver un exemple, puis de copier la déclaration.

Note: A Tecfa, on met certaines DTD ici:

[url: http://tecfa.unige.ch/lib/xml/dtd/](http://tecfa.unige.ch/lib/xml/dtd/)

5.5 Commandes XEmacs/psgml de base

ALT-X xml-mode

met l’éditeur en mode XML

DTD->Parse DTD

relit la DTD (utile si vous faites des changements dans la DTD et/ou si vous déclarez la 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.

5.6 Quelques messages d'erreur dans Emacs

p end-tag implied by p start-tag

p end-tag implied by h1 start-tag

- Veut dire que Xemacs est tombé sur une balise <p> (ou <h1>) à un endroit, 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)

- Veut dire qu'il manque un élément dans <head> ... </head>, en l'occurrence soit "title", soit "base". Pareil pour "ul" et "li" ...
- Autrement dit, certaines balises exigent qu'il y ait des balises imbriquées et ces balises manquent.

Out of context bla tag

- Veut dire qu'il ne peut pas avoir un <bla> ... </bla> à cet endroit.
- Autrement dit, on a pas le droit d'utiliser une balise n'importe où.

6. Validation

- La plupart des éditeurs XML ont un valideur intégré
 - Chercher dans les menus (souvent "Edit")
 - Editeur "freeware" conseillé: XML Exchanger Lite
 - Editeur "freeware" d'arbre (si vous aimez cela) conseillé: XMLMind standard edition
 - Editeur "freeware" WYSIWYG basé sur CSS: Morphon

[url: http://edutechwiki.unige.ch/en/XML_editor](http://edutechwiki.unige.ch/en/XML_editor) (résumé d'une évaluation de plusieurs éditeurs)

6.1 Valideurs en ligne

- La plupart des valideurs en ligne n'analysent que la "well-formedness"

[url: http://validator.w3.org/](http://validator.w3.org/) (principalement destiné à valider les vocabulaires du W3C)

6.2 Valideurs à installer localement

- xmlTester
 - nécessite Java (à installer d'abord)
 - à TECFA: installé dans winapp; cliquer sur w:\bin\XML_Tester.bat

