

Slide 0

XML : vision générale

Stéphane Bortzmeyer

<bortzmeyer@nic.fr>

2 Mars 2004

Slide 0

Ce document est distribué sous les termes de la GNU Free Documentation License

<http://www.gnu.org/licenses/licenses.html#FDL>. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

Slide 1

XML se répand partout

Utilisations "texte" :

- documentation (DocBook, RFC 2629),
- format de fichiers (OpenOffice).

Utilisations "données" :

- fichiers de configuration (ZoneCheck, plug-in de qualification),
- protocoles pour les registres (EPP, IRIS),
- protocole pour "middleware" (XML-RPC),

Tous les projets de logiciel libre font maintenant leur documentation en XML (sauf Python).
Dans le monde IETF, voir le RFC 3470.

Mais tout le monde n'est pas d'accord : <http://bitmover.com/pipermail/lmbench-users/2003-November/000076.html> ou <http://xmlsucks.org/> ou encore <http://c2.com/cgi/wiki?XmlSucks>.

Ironie : XML était prévu pour le Web et normalisé par le W3C mais est peu utilisé pour le Web.

Les bases de XML

XML est un méta-langage (les vrais langages ou **vocabulaires** sont Docbook, FO, SVG, TEI, ...).

Slide 2

XML est un cadre pour faire des formats de fichier.

Pas un langage de programmation.

Le modèle de données est hiérarchique.

XML était un sous-ensemble de SGML (mais c'est moins vrai désormais).

À cause de cette notion de vocabulaire, il est difficile de dire “je connais XML” si on ne connaît que DocBook.

Slide 3

La soupe de sigles XML

XML, XSLT, XSL-FO, Xpath, Xlinks, Xpointer, W3C Schema, Relax NG, DTD, Xquery, SVG, XHTML, SOAP, RSS, RDF, ...

Slide 4

La syntaxe

Dans un document XML, il y a des **éléments** (qui ont parfois des **attributs**) emboîtés.

Les éléments peuvent être vides, contenir d'autres éléments ou bien du texte.

Les noms des éléments sont définis par le vocabulaire.

```
<domain holder='AR41-NIC'>  
  <name>example.fr</name>  
  <active/>  
</domain>
```

Sans la connaissance du vocabulaire, un document XML ne sert pas à grand'chose (sauf vocabulaire trivial).

C'est pour cela que ce n'est pas parce qu'une application produit du XML qu'elle est ouverte.

GFDL : "Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD..."

Les *namespaces*

Si on veut mélanger des éléments de plusieurs vocabulaires.

```
<afnic:domain holder='AR41-NIC'>
  <afnic:name>example.fr</afnic:name>
  <afnic:active/>
  <inpi:mark final='1'>123456789</inpi:mark>
</afnic:domain>
```

Slide 5

Les préfixes des espaces de noms (ici, "afnic" et "inpi") sont définis par un URI (RFC 3305). C'est l'URI qui compte, **pas** le préfixe :

```
<afnic:domain holder='AR41-NIC'
  xmlns:afnic="http://www.afnic.fr/Confiance" >
  <afnic:name>example.fr</afnic:name>
  <afnic:active/>
  <inpi:mark final='1' xmlns:inpi="urn:INPI">123456789</inpi:mark>
</afnic:domain>
```

Les *namespaces* rendent la validation difficile.

Conformité

Les niveaux de conformité d'un document XML :

1. **bien formé** : syntaxiquement correct, les marques fermantes correspondent aux entrantes. Ne dépend pas du vocabulaire.
2. **valide** : obéit à un vocabulaire précis.
3. **correct par rapport aux règles "métier"** : pour l'instant, ce niveau n'est pas formellement traité par XML.

Ce document n'est pas bien formé :

```
<debut>  
<milieu>  
</debut>
```

Un document XML **doit** être bien formé. XML, en raison de la catastrophe d'HTML, rompt avec le principe "Être libéral dans ce qu'on reçoit, ..."

Slide 6

Slide 7

Validité

Ce document DocBook est bien formé mais n'est pas valide.

```
<article>
  <para>Title should come first.</para>
  <title>Bad example</title>
</article>
```

On peut utiliser un éditeur qui guide pour ne produire que des documents valides (Emacs+psgml, Emacs+nxml, ...).

On peut vérifier a posteriori (réception de XML par le réseau : **sécurité**).

OpenReg ne semble pas valider les éléments EPP...

Slide 8

Décrire les règles

Décrire un langage au dessus de XML (DTD, W3C schémas et Relax NG).

On peut aussi le faire informellement.

C'est un travail de **modélisation** (donc pas purement mécanique).

Slide 9

Document Type Definition

Une DTD décrit la composition des éléments.

```
<!ELEMENT domain (fqdn, nameservers)>
<!ATTLIST domain holder IDREF #REQUIRED>
<!ELEMENT nameserver (name, ipaddress?)>
<!ELEMENT nameservers (nameserver)+>
<!ELEMENT fqdn (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

Les DTD sont hérités de SGML.

Leur syntaxe est non-XML.

Aucun typage possible (cf. règle LDH dans l'élément **name** ci-dessus).

Beaucoup d'outils pour DTD.

Slide 10

Un document conforme à cette DTD

```
<domain holder="PL1">
  <fqdn>foobar.example</fqdn>
  <nameservers>
    <nameserver>
      <name>ns1.example.com</name>
    </nameserver>
    <nameserver>
      <name>ns2.example.com</name>
    </nameserver>
  </nameservers>
</domain>
```

ipaddress est optionnelle.
Insister sur l'absence de typage (entiers positifs, etc).

Slide 11

Identifiants

Pour trouver la DTD d'un document XML, on utilise les identifiants :

- publics,
- système.

```
<!DOCTYPE zone PUBLIC "-//Generic NIC//DTD DNS Registry V1.0"  
    "http://www.gennic.net/registry.dtd">
```

Les catalogues permettent de trouver un nom de fichier à partir d'un identifiant.

Slide 12

Véifier avec une DTD

```
nsgmls -s -wxml /usr/share/sgml/declaration/xml.dcl example.xml  
xmllint --valid --noout example.xml
```

Le premier outil, partie du paquetage sp, traite SGML. Le second, partie du paquetage libxml2, ne traite que XML.

W3C schémas

Un des langages de schémas mais pas le seul.

Normalisé par le W3C.

Syntaxe XML.

Comme une DTD, un W3C schéma décrit la composition des éléments.

Très riche, notamment pour le typage.

Sépare la description de la structure et celle du contenu.

Utilisé dans plusieurs protocoles Internet comme EPP ou IRIS.

Peu d'outils (trop complexe ?).

Slide 13

Slide 14

Exemple de W3C schema, EPP

```
<element name="epp" type="epp:eppType"/>
...
<complexType name="greetingType">
  <sequence>
    <element name="svID" type="epp:sIDType"/>
    <element name="svDate" type="dateTime"/>
    <element name="svcMenu" type="epp:svcMenuType"/>
  </sequence>
</complexType>
```

Slide 15

Xerces

Une vérification d'un schéma avec Xerces (validateur d'Apache) :

```
% perl DOMPrint.pl -v=always -n -s personal-schema.xml > /dev/null
Error in eval: ERROR:
FILE:      /home/bortzmeyer/tmp/personal-schema.xml
LINE:      9
COLUMN:    12
MESSAGE:   Not enough elements to match content model :
           '(name,email*,url*,link,link,link?)'
at DOMPrint.pl line 131
at DOMPrint.pl line 132
```

Slide 16

Relax NG

Un autre langage de schéma, très populaire dans le monde du logiciel libre.

Normalisé dans le cadre d'Oasis.

Syntaxe XML ou "compacte".

Relax décrit des motifs, auxquels un document valide doit se conformer (cf. expressions rationnelles).

Moins riche pour le typage que les W3C schémas.

On peut traduire du Relax en DTD ou en W3C schémas.

Déjà des outils.

Slide 17

Exemple de schéma Relax NG

```
<element name="addressBook" xmlns="http://relaxng.org/ns/structure/1.0">
  <zeroOrMore>
    <element name="card">
      <element name="name">
        <text/>
      </element>
      <element name="email">
        <text/>
      </element>
    </element>
  </zeroOrMore>
</element>
```

Slide 18

Exemple Relax NG en syntaxe compacte

```
element addressBook {  
  element card {  
    element name { text },  
    element email { text }  
  }*  
}
```

Slide 19

Quel outil choisir ?

Ne pas oublier que Trang peut traduire de n'importe quel langage de schéma en presque n'importe quel autre.

Ou bien utiliser un fichier XML d'exemple comme définition du vocabulaire (Examplotron).

Slide 20

XSLT

XSLT est un langage de transformation de documents XML :

- en documents XML d'un autre vocabulaire,
- ou en documents texte (BIND, LaTeX, *roff, etc).

Normalisé par le W3C, écrit en XML.

Insister : XSLT n'est pas une feuille de style et ne connaît rien à la présentation.

Slide 21

Exemple XSLT

Traduction de bibliographie RFC 2629 en bibliographie utilisant la DTD du NIC générique :

```
<xsl:template match="reference">
  <xsl:variable name="id">
    <xsl:value-of select="substring-after(@anchor,'RFC')"/>
  </xsl:variable>
  <rfc><xsl:attribute name="num">
    <xsl:value-of select="$id"/></xsl:attribute>
    <xsl:apply-templates select="front/author"/>
    <xsl:text>#10;</xsl:text>
    <xsl:apply-templates select="front/title"/>
    <xsl:text>#10;</xsl:text>
    <xsl:apply-templates select="front/date"/>
    <xsl:text>#10;</xsl:text>
  </rfc>
```

select prend comme argument une expression Xpath.

Slide 22

Autre exemple XSLT

Traduction d'un document XML d'un registre en fichier de zone BIND :

```
<xsl:template match="/zone/domain">
  <xsl:for-each select="nameservers/nameserver">
    <xsl:value-of select="../../fqdn"/>. IN NS <xsl:value-of select="name"/>.
    <xsl:variable name="name"><xsl:value-of select='name'/></xsl:variable>
    <!-- Add the glue records, if necessary -->
    <xsl:if test="substring($name, 2 + string-length($name) -
      string-length('$tld?')) = $tld">
      <xsl:text>#10;</xsl:text>
      <xsl:value-of select="$name"/>. IN A <xsl:value-of select="ipaddress"/>
      </xsl:if>
      <xsl:text>#10;</xsl:text>
    </xsl:for-each>
    <xsl:text>#10;</xsl:text>
  </xsl:template>
```

Slide 23

XSLT en pratique

Implémentations : cela a pris du temps mais il en existe désormais beaucoup d'excellentes en logiciel libre.

xsftproc (libxml2), xalan, sablotron, 4suite, ...

Un exemple un peu tordu. La FAQ du service de surveillance des TLD (écrit en Mason) du NIC générique est écrite en XML (DTD QAML). Un processeur XSLT est appelé par Mason pour afficher la FAQ (rassurez-vous, il y a un cache). Cocoon permet de faire encore pire.

Slide 24

Usine à gaz avec XSLT

```
<% $stylesheet->output_string($results) %>

<%init>
my $results = $stylesheet->transform($source);
</%init>

<%once>
use XML::LibXSLT;
use XML::LibXML;

my $prefix = "/var/www/www.generic-nic.net/dyn/mon";

my $parser = XML::LibXML->new();
my $xslt = XML::LibXSLT->new();

my $source = $parser->parse_file("$prefix/faq.xml");
my $style_doc = $parser->parse_file("$prefix/faq.xsl");
```

Slide 25

```
my $stylesheet = $xslt->parse_stylesheet($style_doc);  
</%once>
```

Slide 26

Attaque par programme

XSLT n'est pas obligatoire pour faire des transformations.

On peut aussi utiliser un programme en *name your favorite language here*.

Plusieurs API :

- celle de l'analyseur expat, bâtie sur des événements (début d'élément, fin d'élément, ...),
- SAX (Simple API for XML), également bâtie sur des événements mais pas les mêmes,
- DOM (Document Object Model), bâti sur un modèle hiérarchique du document.

Insister : avec le langage X et DOM, on peut faire tout ce que fait XSLT.
Rappeler que le modèle de données de XML est hiérarchique.

Slide 29

Créer du XML

On peut créer du XML :

- avec vi,
- avec un programme qui fait du `printf("<foobar>..."`,
- avec un programme qui s'appuie sur DOM ou sur une autre API qui guide la génération de XML,
- avec un éditeur qui aide (mode psgml d'emacs ou nxml - mode Relax de Clark),

Avec l'approche printf, il faut échapper les caractères spéciaux. Et il faut faire attention à produire du XML valide.

Slide 30

Quelques langages (ou vocabulaires)

Des exemples pris au hasard...

Slide 31

FO (Formatting Objects)

Un langage de description de page.

Normalisé W3C.

Concurrent de PDF ou de Postscript.

Pas encore d'imprimantes FO :-)

On utilise souvent XSLT pour produire du FO à partir de Docbook ou de TEI.

```
<fo:block space-before.optimum="1em">
Since we are using the XML version of DocBook, here is how to
call <fo:simple-link internal-destination="jade">jade</fo:simple-link>
to translate
<fo:inline-sequence
font-family="monospace">myfile.db</fo:inline-sequence>
to TeX:</fo:block>
```

Pour faire du PDF à partir de FO : FOP.

Slide 32

SVG, Scalable Vector Graphics

Un langage de description d'images.

Normalisé W3C.

Concurrent de Flash ou de FIG.

```
<rect x="0" y="0" width="1" height="1" style="stroke:none;fill:white;"/>
<polygon points="1.4998,1.0002 586.5,1.0002 586.5,325 1.4998,325 "
          style="stroke:#a5dbff;stroke-width:0;fill:#a5dbff;"/>
```

Mozilla supporte SVG.

Slide 33

DocBook

Une vocabulaire très riche et très complexe pour les documentations techniques.

Normalisé Oasis.

Utilisé par quasiment tous les projets de logiciel libre.

Documentations :

```
<para><replaceable>nameservers</replaceable> is the exhaustive
list of nameservers that are primary or secondary for the zone,
they are separated by a semicolon, [...] a comma-separated list of
IP addresses. For example:
<literal>ns1.toto.fr=192.168.1.5,192.168.1.6;ns2.toto.fr</literal>
</para>
```

Articles :

```
<para>Le domaine est donc assez miné, et le
vocabulaire complexe (<xref linkend="unicode.glossary"/>).</para>
```

Slide 34

Désigner des éléments : Xpath

Syntaxe non-XML pour désigner une partie d'un document XML.
Une sorte de grep *XML-aware*.

`/foo/bar` : tout fils nommé `bar` d'un élément `foo` de premier niveau.

`//contact[@handle="SB1"]` : tout élément (quel que soit son niveau) `contact` qui a un attribut `handle` dont la valeur est `SB1`.

Serveur whois du NIC générique, dorsal XML :

```
# Use XPath in the DOM tree to find the domain
result = Evaluate("/zone/domain[fqdn=\"\" + \
    string.lower(domain) + "\"]",
    xml_dom_object.documentElement)
```

On trouve une bibliothèque Xpath dans toutes les bonnes bibliothèques XML.

Slide 35

Liens : Xlinks

Liens simples, analogues à ceux de HTML.

Ou liens riches, bi-directionnels ou vers un ensemble, pas juste un point.

Un *namespace* et des attributs standards.

Aucun navigateur Web ne les accepte ?

```
<report xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:type="simple"
    xlink:href="http://www.free.fr/~example/report.txt">
    ...
```

Xpointers

Xpath + URI.

Permettent de pointer vers l'intérieur d'un document, sans modifier ce dernier.

Pointer vers le troisième paragraphe du document doc.xml :

```
http://exa.fr/foutoir/doc.xml#xpath(/para[position()=3])
```

Peuvent s'utiliser dans des Xlinks.

Slide 36

Les avocats de l'IPR vont encore trouver un moyen d'interdire Xpath ! (Liens "profonds".)

Slide 37

Encodage XML: les services Web

Pas grand'chose à dire : XML ne sert que pour l'encodage.

Le programmeur ne voit pas de XML.