Modélisation et interopérabilité : Technologie XML

Benoît Valiron <benoit.valiron@monoidal.net>

http://inf356.monoidal.net/

Le problème

Manipuler des données textuelles à travers

- Diverses applications (communication)
- Homme-machine (lisibilité)
- Divers groupes de projet
- Un temps potentiellement long (support pour 10 ans ?)
- Un peu mieux que ASCII brut...

Encodage de données

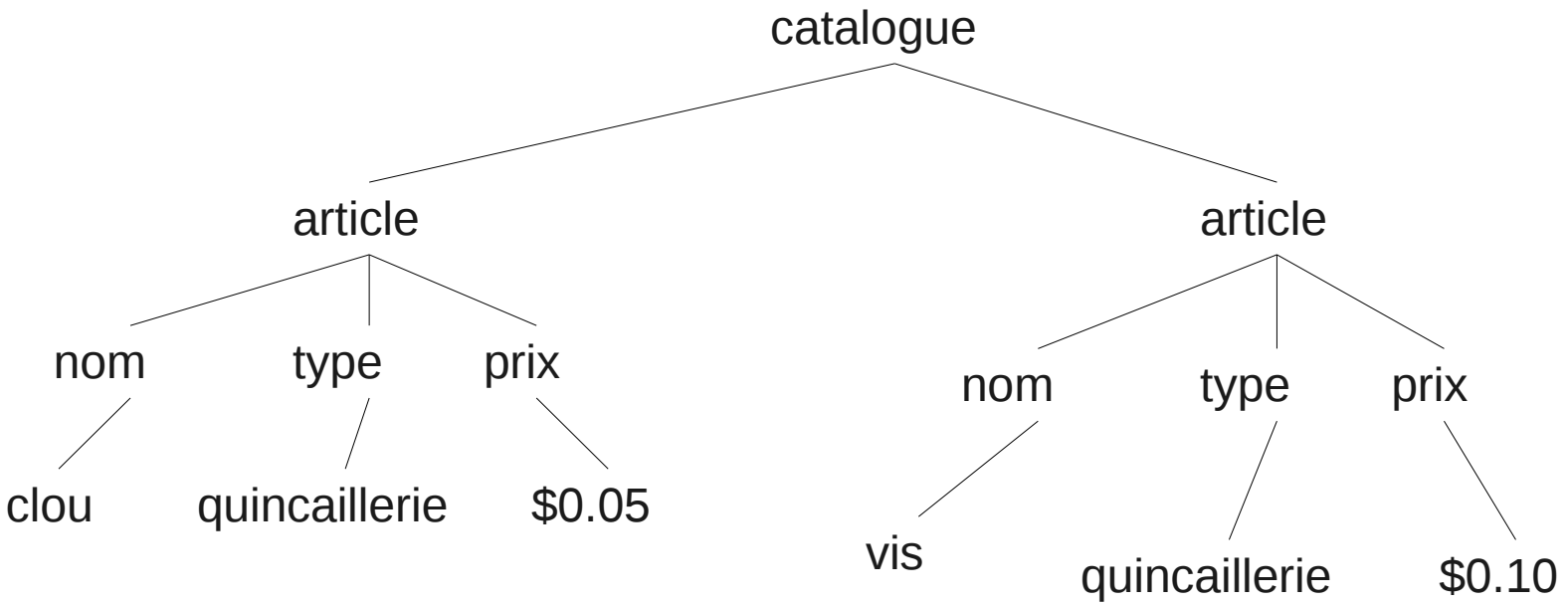
Comment faire ?

En tableau (CVS par exemple) :

ID	Article	Quantité	Prix
#0001	Clou	84	$0.05
#0002	Vis	10	$0.10
#0003	Marteau	2	$10.00

Peu flexible... Et des cafetières ?

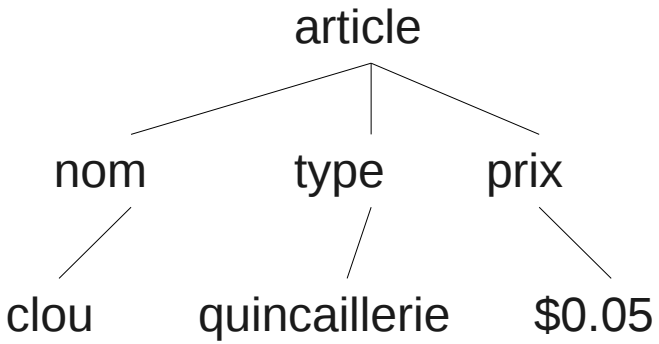
Encodage en arbre



Ici, on peut facilement rajouter un nouveau type, de nouveaux éléments, ralonger l'arborescence... sans casser la logique de la base de données.

Correspondance textuelle

Aux noeuds correspondent des balises




```
<article>

<nom>
clou
</nom>

<type>
quincaillerie
</type>

<prix>
$0.05
</prix>

</article>
```

Balise ouvrante

Noeud de texte

Balise fermante

Historique : SGML

- Problème : Conservation de données textuelles à travers le temps, l'espace et les personnes
- SGML “Standard Generalized Markup Language”
  - 1970, chez IBM. Standard ISO-8879 en 1986
  - Par Charles Goldfarb, Ed Mosher, Ray Lorie
  - Pour écrire de la documentation (plusieurs milliers de pages) : Séparer le fond de la forme
  - Utilisation par l'administration US, les militaires, l'aéronautique, mais aussi l'Europe avec le CERN...

Historique : HTML

- HTML “HyperText Markup Language”
  - Application SGML, pour les documents hypertextes
  - ~ 1990
  - Ensemble restreint de balises
- Évolution par les éditeurs de navigateurs
  - Ajout de balises, ajout de feuilles de styles
- 1994 : création du W3C, qui orchestre et synthétise les changements

Le sujet du cours : XML

- XML “eXtensible Markup Language”
  - 1996 : le W3C établit groupe de travail sur une simplification de SGML
  - Jon Bosak, Tim Bray, C. M. Sperberg-McQueen, James Clark, ...
  - 1998 : recommandation XML 1.0
  - Garde la flexibilité de SGML mais en plus léger : le texte ISO-8879 fait 600 pages, la spécification XML moins d'une centaine.
  - Apporte de nombreux outils: Feuilles de style variées, XSLT, XPath, Xlink, XInclude, XSL-FO, ...

Cette U.E.

Horaires :

- Cours (11) : jeudi, 8h-9h30
- TDs (11) : mardi, 13h30-15h (G1) et 11h30-13h (G2)
- TPs (10) : mercredi, 15h15-16h45 (G1) et 17h-18h30 (G2)

Évaluation :

- Contrôle continu : un projet à rendre à la fin de l'année
- Examen final : comme d'habitude

Bibliographie :

- *XML en concentré*, E. R. Harold et W. S. Means, O'Reilly
- *XML*, G. Chagnon et F. Nolot, Collection Synthex, Pearson Education.
- http://www.w3c.org/TR

Plan du cours

1 – Base : XML et DTDs

2 – Espaces de noms

3 – Exemples : XHTML, flux RSS, SVG, MathML, RDF et Dublin Core.

4 – Formats de texte : DocBook, OpenOffice, XSL-FO.

5, 6 – Validation avancée : Relax-NG

7 – Xpath

8, 9 – XSLT

10 – Modèle de document : DOM

11 – Parseurs XML : SAX et StAX

1 – Métalangage XML

Un document XML

```
<personne né=”1912” mort = '1954'>
  <!-- Ceci est un document xml -->
  <nom>
    <prénom>Alan</prénom>
    <nom_famille>Turing</nom_famille>
  </nom>
  <profession>informaticien</profession>
  <profession>mathématicien</profession>
  <profession>cryptographe</profession>
</personne>
```

Noeuds de type élément

- Un élément s'ouvre et se ferme par une balise:

```
<prénom>Alan</prénom>
```

- Les balises doivent être bien balancées:

```
<italique><gras>Mal formé</italique></gras>
```

- Les éléments sans fils peuvent prendre deux écritures:

```
<profession></profession>
ou
<profession />
```

Élément racine

- C'est le **seul** élément sans parent.
- Document non valide :

```
<nom>
  <prénom>Alan</prénom>
  <nom_famille>Turing</nom_famille>
</nom>
<nom>
  <prénom>Alonzo</prénom>
  <nom_famille>Church</nom_famille>
</nom>
```

Contenu mixte

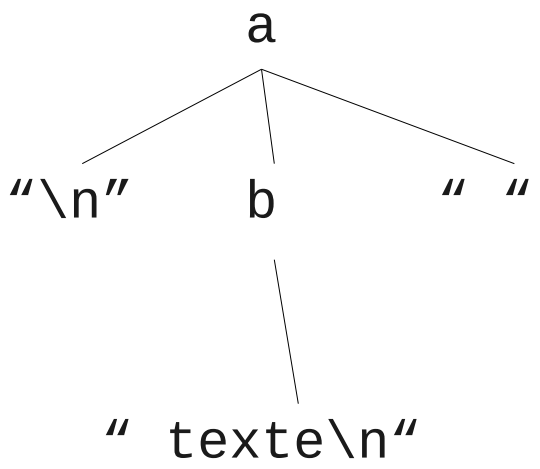
- Un élément peut contenir :
  - Des sous-éléments
  - Des données textuelles
  - Un mélange des deux (à la HTML)

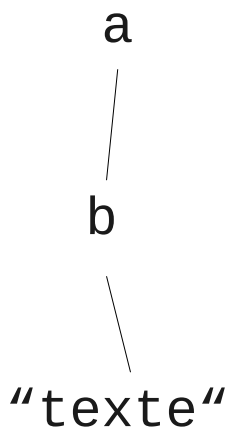
```
<paragraph>
  <prénom>Alan</prénom> <nom_famille>Turing</nom_famille>
  est un <profession>informaticien</profession> à
  l'origine de la fameuse <emphase>Machine de
  Turing</emphase>
</paragraph>
```

Gestion des espaces

- Espaces, tabulations et sauts de ligne sont considérés ensemble.
- En général, ils sont ignorés par les applications :

```
<a>
<b> texte
</b> </a>
```





Attributs

- Les éléments XML peuvent avoir des attributs
  - Pair nom – valeur (chaine de caractères)
  - “table d'association”

```
<personne naissance=”23-06-1912” mort = '07-06-1954'>
  Alan Turing
</personne>
```

  - Espaces ou pas
  - “ ou '

Sous-éléments ou attributs ?

```
<personne>
  <nom prénom=”Alan” nom_famille=”Turing” />
  <profession value=”informaticien” />
  <profession value=”mathématicien”/>
  <profession value=”cryptographe”/>
</personne>
```

- Attributs ~ métadonnée (ID, date...)
- Attributs uniques pour un élément donné
- Attributs : seulement du texte

Nom XML

- Nom de balise et nom d'attribut :
  lettres, chiffres, - _ .
- C'est tout ! (Ni espace, ni autre ponctuation)
- De plus, il ne peuvent pas commencer par
  chiffres - .
- Le “:” est réservé pour les espaces de noms
- La chaine “xml” en début de nom est réservé
- Casse : <AAA> est distinct de <aaa>

Contre-exemples

- `<o'clock>5</o'clock>`
- `<m/j/a>17-09-2009</m/j/a>`
- `<nom famille>Turing</nom famille>`
- `<4-nombres>0123</4-nombres>`

Exemples

- `<o_clock>5</o_clock>`
- `<m-j-a>17-09-2009</m-j-a>`
- `<nomfamille>Turing</nomfamille>`
- `<_4-nombres>0123</_4-nombres>`

Appel d'entité

- Problème : Comment avoir “<” dans un élément texte ou dans la valeur d'un attribut ?
- Solution : avec un caractère échappé.

&alias;

<	>	&	"	'
<	>	&	“	'

```
<test>(x &lt; y) &amp; (x &gt; z)</test>
```

Section CDATA

- Si beaucoup de symbole à échapper, cela peut devenir pénible. On peut utiliser la notation

```
<![CDATA[ texte ]]>
```

- “texte” sera traité comme du texte
- Pas de “]]>” dans une section CDATA...

```
<texte><![CDATA[<html><head>
  <title>TITRE</title></head>
  <body>Hello World</body></html>]]>
  </texte>

<texte>&lt;html&gt;&lt;head&gt;&lt;title&gt;
  TITRE&lt;/title&gt;&lt/head&gt;&lt;body&gt;
  Hello World&lt;/body&gt;&lt;/html&gt;</texte>
```

Commentaires

- Des noeuds de commentaires peuvent être inclus. Ils ne sont pas interpreté par les applications.

```
<!-- commentaire -->
```

- Le double trait d'union “--” ne doit pas apparaitre dans le commentaire.
- Les commentaires ne doivent pas apparaître dans les balises

Instructions de traitement

Notation :

```
<?instruction … ?>
```

Exemples :

```
<?php … ?>
<?xml-stylesheet … ?>
```

Ne font pas parties de l'arbre XML.

Déclaration XML

- Un document peut (mais n'est pas obligé) commencer par une déclaration XML.
- Ressemble à une instruction de traitement :

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

- Ordre des arguments important.

Encodage de caractères

- Un caractère comme É peut avoir plusieurs encodage, voir ne pas pouvoir être encodé.
- É :

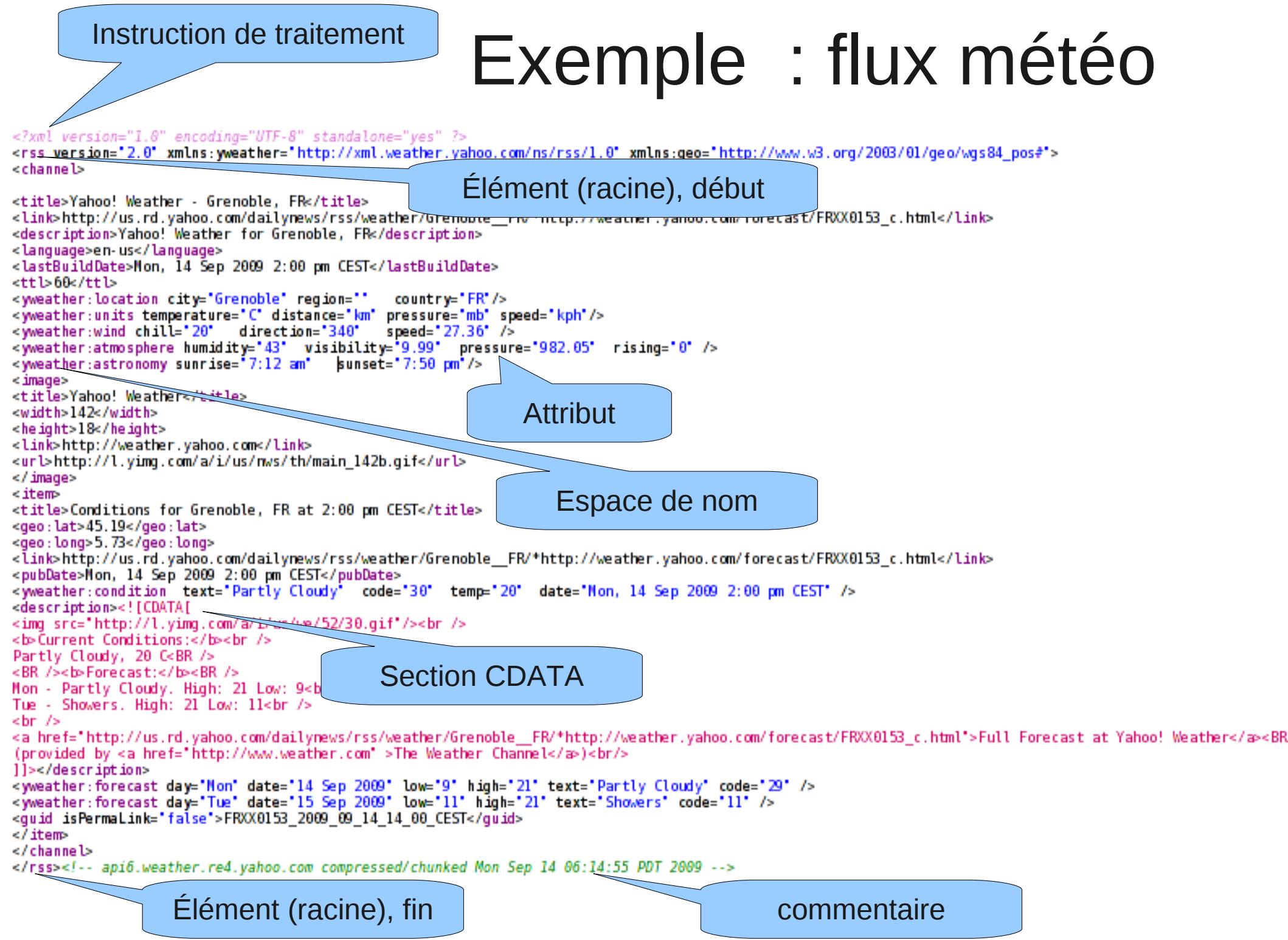
ASCII	latin-1	utf-8
impossible	11101001	11000011 Ou 10101001

- Par défaut, XML est en UTF-8.
- L'encodage est changé par l'attribut “encoding”

Document bien formé

- À chaque balise de début correspond une balise de fin
- Les éléments peuvent être imbriqué mais pas se recouvrir
- Un seul élément racine
- Valeurs d'attribut entre guillemets
- Un élément ne peut avoir 2 attributs de même nom
- Les commentaires ne doivent pas apparaître dans les balises
- Les caractères & et < sont échappés
- ...

Exemple : flux météo




```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rss version="2.0" xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0" xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
<channel>

<title>Yahoo! Weather - Grenoble, FR</title>
<link>http://us.rd.yahoo.com/dailynews/rss/weather/Grenoble__FR/*http://weather.yahoo.com/forecast/FRXX0153_c.html</link>
<description>Yahoo! Weather for Grenoble, FR</description>
<language>en-us</language>
<lastBuildDate>Mon, 14 Sep 2009 2:00 pm CEST</lastBuildDate>
<ttl>60</ttl>
<yweather:location city="Grenoble" region="" country="FR"/>
<yweather:units temperature="C" distance="km" pressure="mb" speed="kph"/>
<yweather:wind chill="20" direction="340" speed="27.36" />
<yweather:atmosphere humidity="43" visibility="9.99" pressure="982.05" rising="0" />
<yweather:astronomy sunrise="7:12 am" sunset="7:50 pm"/>
<image>
<title>Yahoo! Weather</title>
<width>142</width>
<height>18</height>
<link>http://weather.yahoo.com</link>
<url>http://l.yimg.com/a/i/us/nws/th/main_142b.gif</url>
</image>
<item>
<title>Conditions for Grenoble, FR at 2:00 pm CEST</title>
<geo:lat>45.19</geo:lat>
<geo:long>5.73</geo:long>
<link>http://us.rd.yahoo.com/dailynews/rss/weather/Grenoble__FR/*http://weather.yahoo.com/forecast/FRXX0153_c.html</link>
<pubDate>Mon, 14 Sep 2009 2:00 pm CEST</pubDate>
<yweather:condition text="Partly Cloudy" code="30" temp="20" date="Mon, 14 Sep 2009 2:00 pm CEST" />
<description><![CDATA[
<img src="http://l.yimg.com/a/i/us/we/52/30.gif"/><br />
<b>Current Conditions:</b><br />
Partly Cloudy, 20 C<BR />
<BR /><b>Forecast:</b><BR />
Mon - Partly Cloudy. High: 21 Low: 9<b
Tue - Showers. High: 21 Low: 11<br />
<br />
<a href="http://us.rd.yahoo.com/dailynews/rss/weather/Grenoble__FR/*http://weather.yahoo.com/forecast/FRXX0153_c.html">Full Forecast at Yahoo! Weather</a><BR
(provided by <a href="http://www.weather.com" >The Weather Channel</a>)<br/>
]]></description>
<yweather:forecast day="Mon" date="14 Sep 2009" low="9" high="21" text="Partly Cloudy" code="29" />
<yweather:forecast day="Tue" date="15 Sep 2009" low="11" high="21" text="Showers" code="11" />
<guid isPermaLink="false">FRXX0153_2009_09_14_14_00_CEST</guid>
</item>
</channel>
</rss><!-- api6.weather.re4.yahoo.com compressed/chunked Mon Sep 14 06:14:55 PDT 2009 -->
```

Validation DTD

Pourquoi valider ?

- Un fichier est écrit et lu par plusieurs personnes ou applications
  - L'émetteur : s'assurer d'être compris
  - Le récepteur : s'assurer de pouvoir comprendre le fichier
- Règle : Strict quand on publie, Laxiste quand on lit ! Les navigateur ne vérifient pas les DTDs
- Documentation formelle : nouvel acteur peut communiquer.

Association à un document XML

- Externe :
```
<!DOCTYPE racine PUBLIC place1 place2 >
<!DOCTYPE racine SYSTEM place >
```
- Interne :
```
<!DOCTYPE racine [ ..... ]>
```
- Mixte :
```
<!DOCTYPE racine PUBLIC p1 p2 [ .... ]>
<!DOCTYPE racine SYSTEM place [ .... ]>
```
- Les DTDs internes sont prioritaires

Exemple

document.xml :

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE personne SYSTEM “personne.dtd”>
<personne>
  <prénom>Alan</prénom>
  <nom_famille>Turing</nom_famille>
</personne>
```

personne.dtd :

```
<?xml version=”1.0” encoding=”utf-8”?>
<!ELEMENT personne (prénom, nom_famille) >
<!ELEMENT prénom (#PCDATA) >
<!ELEMENT nom_famille (#PCDATA) >
```

Règles de bases

- Portent sur les éléments et les attributs
- Format:
  `<!mot-clé paramètre1 paramètre2 … >`
- Mots clés: `ELEMENT ATTLIST ENTITY (NOTATION)`
  - `ELEMENT` : éléments XML
  - `ATTLIST` : attribut XML
  - `ENTITY` : caractères spéciaux et macro texte

ELEMENT

- Format :

```
<!ELEMENT nom ANY>
<!ELEMENT nom EMPTY>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT nom (modèle_de_contenu)>
```

- Modèle de contenu (sans texte) est construit à l'aide de la syntaxe:

```
(a, b, ...)        (a | b | ...)
(a*)    (a+)   (a?)
```

- Contenu mixte: `(#PCDATA | a | b | ...)*`

Exemple

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE personnes [
<!ELEMENT personnes (personne*) >
<!ELEMENT personne (
        (prénom, initial?, nom_famille)
      | nom_famille
) >
<!ELEMENT prénom (#PCDATA) >
<!ELEMENT initial (#PCDATA) >
<!ELEMENT nom_famille (#PCDATA) >
]>
<personnes>
...
</personnes>
```

ATTLIST

- Format :

```
<!ATTLIST elt attribut type #REQUIRED>
<!ATTLIST elt attribut type #IMPLIED>
<!ATTLIST elt attribut type #FIXED valeur>
<!ATTLIST elt attribut type valeur>
```

  - `REQUIRED :` Attribut obligatoire
  - `IMPLIED :` optionel
  - `FIXED valeur :` optionel, mais fixé si présent
  - `Valeur :` optionel, avec valeur par défaut

Les types d'attributs

- Dix types d'attributs existent dans XML:
  - `CDATA` : texte quelconque
  - `ENUMERATION` : une liste d'unités lexicales
  - `ID` : nom XML unique dans le document
  - `IDREF` : référence à un attribut de type ID
  - `IDREFS` : “id1 id2 id3 ...”
  - `NMTOKEN, NMTOKENS, ENTITY, ENTITIES, NOTATION.`

Exemple : CDATA et IDs

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE organigramme [
<!ELEMENT organigramme (personne*,projet*)>
<!ELEMENT personne EMPTY>
<!ELEMENT projet EMPTY>
<!ATTLIST personne nom CDATA #REQUIRED>
<!ATTLIST personne id ID #REQUIRED>
<!ATTLIST projet membres IDREFS #REQUIRED
                  nom CDATA #IMPLIED “Boire du café”>
]>
<organigramme>
  <personne id=”1” nom=”Bob”/>
  <personne id=”2” nom=”Marie”/>
  <personne id=”3” nom=”Alfred”/>
  <projet membres=”1 2” nom=”repeindre le couloir” />
  <projet membres=”1 2 3” />
  <projet membres=”3” nom=”Truc urgent” />
</organigramme>
```

Enumérations

- Une énumération est sous la forme

```
<!ATTLIST date jour (Lundi | Mardi |
Mercredi | Jeudi | Vendredi | Samedi |
Dimanche) #REQUIRED>
```

- Les noms dans l'énumération sont des “unités lexicales nominales XML” : des lettres, des chiffres, _ - . : sans restrictions.
- Exemple:

```
<!ATTLIST element att
    (.bashrc | 123 | _.-ee | 3-4 | xml)
    #IMPLIED>
```

ENTITY

- Deux types d'entités majeurs :
  - Les entités générales
  - Les entités paramètres
- Entités générales : pour faire des abréviations

```
<!ENTITY moi “Benoît Valiron”>
```

- Utilisés comme `&apos; < &lq; & &quot;`

```
<texte>Je m&apos;appelle &moi</texte>
```

Exemple

```
<?xml version=”1.0”?>
<!DOCTYPE texte [
<!ELEMENT texte (#PCDATA)>
<!ENTITY b1 “Tic ”>
<!ENTITY b2 “&b1 &b1”>
<!ENTITY b3 “&b2 &b2”>
<!ENTITY b4 “&b3 &b3”>
]>
<texte>&b4</texte>
```

ENTITY (suite)

- Entités paramètre : utilisable uniquement dans la DTD

```
<!ENTITY % nom_entité texte_remplacement>
```

- Pour des raccourcis. Exemple:

```
<!ENTITY % liste “(a | b | c | d | e | f | g |
h | i | j | k | l | m | n | o | p | q | r | s |
t | u | v | w | x | y | z) #REQUIRED”>
<!ELEMENT alphabet EMPTY>
<!ATTLIST alphabet lettre %liste>
```

- Pour changer une DTD en interne

Exemple

tic.dtd

```
<?xml version=”1.0” encoding=”utf-8”?>
<!ENTITY % tic '“Tic ”'>
<!ELEMENT texte (#PCDATA)>
<!ENTITY b1 %tic>
<!ENTITY b2 “&b1 &b1”>
<!ENTITY b3 “&b2 &b2”>
<!ENTITY b4 “&b3 &b3”>
```

fichier.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE texte SYSTEM “tic.dtd ” [
<!ENTITY % tic “Tac ”>
]>
<texte>&b4</texte>
```