Modélisation et interopérabilité : Semaine 39, cours 2

Benoît Valiron <benoit.valiron@monoidal.net>

http://inf356.monoidal.net/

Suite (et fin) des DTDs

DTD externe...

style.dtd

```
<!ELEMENT personne (nom, prénom)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
<!ATTLIST personne né CDATA #REQUIRED>
<!ATTLIST personne mort CDATA #IMPLIED>
```

document.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE personne SYSTEM “style.dtd”>
<personne né=”1912”>
  <nom>Turing</nom>
  <prénom>Alan</prénom>
</personne>
```

...DTD interne

document.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE personne [
  <!ELEMENT personne (nom, prénom)>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prénom (#PCDATA)>
  <!ATTLIST personne né CDATA #REQUIRED>
  <!ATTLIST personne mort CDATA #IMPLIED>
]>
<personne né=”1912”>
  <nom>Turing</nom>
  <prénom>Alan</prénom>
</personne>
```

DTD publique

Elle doit se trouver sur internet.
Invocation :

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Mon document</title>
  </head>
  <body>
    <p>Un paragraphe.</p>
  </body>
</html>
```

Règles de DTD

- `<!ELEMENT ...` vu.
- `<!ATTLIST ...` vu.
- `<!ENTITY ...`

ENTITY

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>
```

- Cas particulier : `&#xxx;` signifie le caractère de valeur xxx dans le jeu de caractères unicode

Exemple

```
<?xml version=”1.0”?>
<!DOCTYPE texte [
<!ELEMENT texte (#PCDATA)>
<!ENTITY prof “Benoît Valiron”>
<!ENTITY imag “UFR IMAG”>
<!ENTITY cours “Modélisation et Interopérabilité”>
]>
<texte>
&prof; enseigne le cours &cours; à l&apos;&imag;.
</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>
```

Appel de caractère

- Unicode est un standard international de numérotation de caractère.
- De nombreux caractères ne sont pas accessibles à l'aide du clavier, par exemple les caractères grecs.
- On peut les appeler avec le caractère d'échappement. Celui-ci sera valide si l'encodage utilisé admet une représentation du caractère.

	α	β	γ	δ
Unicode (hexa)	3B1	3B2	3B3	3B4
Unicode (décimal)	945	946	947	948
Appel XML	α	&#946	&#947	&#948

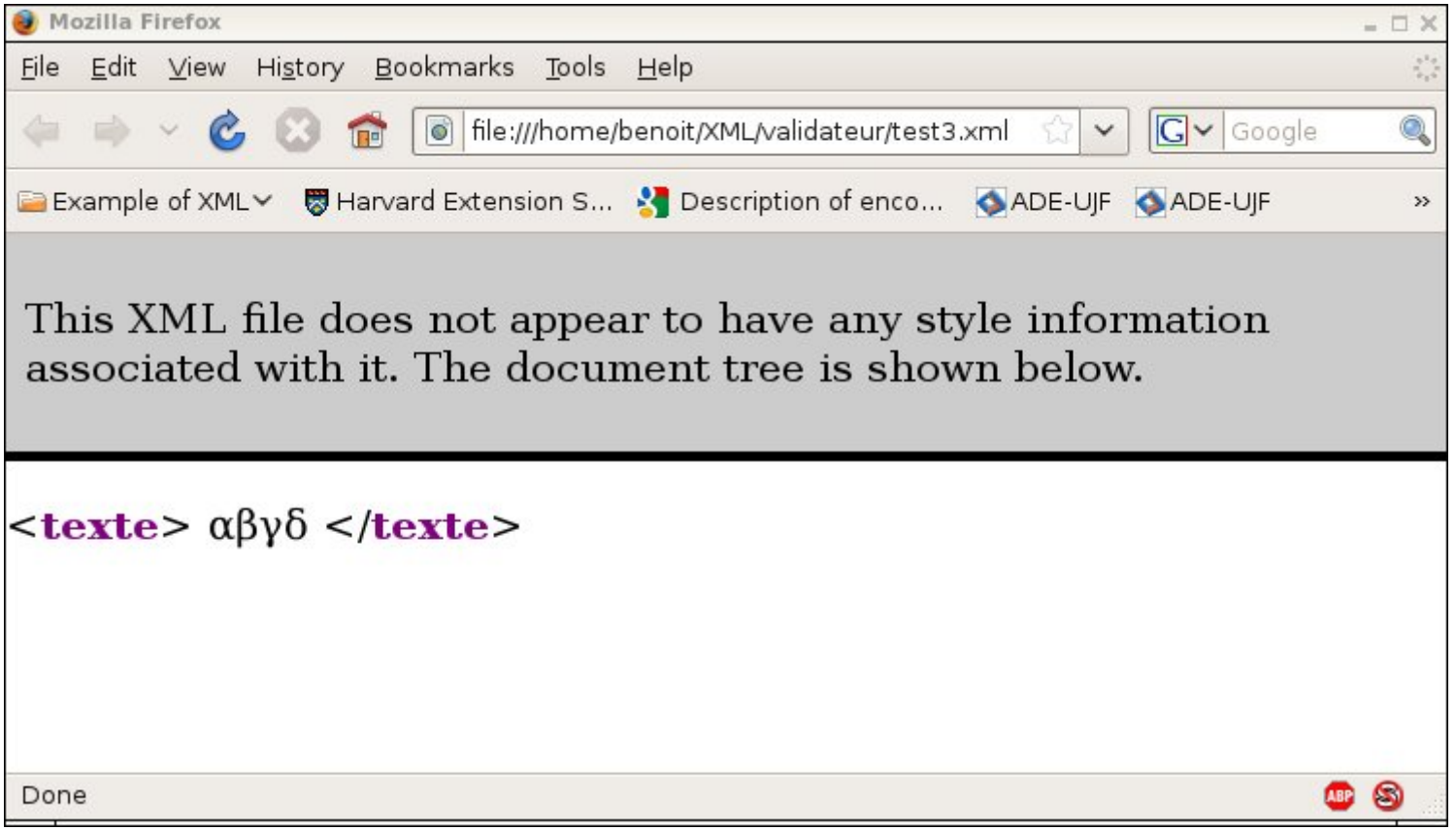
Exemple

```
<?xml version=”1.0” encoding=”utf-8”?>
<texte>
&#945;&#946;&#947;&#948;
</texte>
```

Ou encore

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE texte [
<!ELEMENT texte (#PCDATA)>
<!ENTITY alpha “&#945;”>
<!ENTITY beta “&#946;”>
<!ENTITY gamma “&#947;”>
<!ENTITY delta “&#948;”>
]>
<texte>
&alpha;&beta;&gamma;&delta;
</texte>
```

Résultat dans firefox



Contre-exemple (1/2)

```
<?xml version=”1.0”?>
<!DOCTYPE plan [
<!ELEMENT plan ((x,y,z), (x,y,z), (x,y,z))>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT z (#PCDATA)>
]>
<plan>
<x>0</x><y>0</y><z>0</z>
<x>0</x><y>1</y><z>0</z>
<x>0</x><y>0</y><z>1</z>
</plan>
```

Contre-exemple (2/2)

```
<?xml version=”1.0”?>
<!DOCTYPE plan [
<!ENTITY pt “(x, y, z)”>
<!ELEMENT plan (&pt;, &pt;, &pt;)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT z (#PCDATA)>
]>
<plan>
<x>0</x><y>0</y><z>0</z>
<x>0</x><y>1</y><z>0</z>
<x>0</x><y>0</y><z>1</z>
</plan>
```

NON-VALIDE !

ENTITY (suite)

- Entités paramètre : utilisable uniquement dans la DTD

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

- Pour des raccourcis. Exemple:

```
<!ENTITY % liste “(a | b | c | d) #REQUIRED”>
<!ELEMENT alphabet EMPTY>
<!ATTLIST alphabet lettre %liste;>
```

- Définition et utilisation en DTD externe
- Redéfinition possible en DTD interne

Exemple

```
<?xml version=”1.0”?>
<!DOCTYPE plan [
<!ENTITY % pt “(x, y, z)”>
<!ELEMENT plan (%pt;, %pt;, %pt;)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT z (#PCDATA)>
]>
<plan>
<x>0</x><y>0</y><z>0</z>
<x>0</x><y>1</y><z>0</z>
<x>0</x><y>0</y><z>1</z>
</plan>
```

VALIDE !

Exemple

tic.dtd

```
<!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>
```

Exemple

fichier.dtd

```
<!ENTITY % elt “a”>
<!ELEMENT racine (%elt;*)>
<!ELEMENT %elt; EMPTY>
```

fichier.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE racine SYSTEM “fichier.dtd”>
<racine><a/><a/></racine>
```

fichier2.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<!DOCTYPE racine SYSTEM “fichier.dtd” [
  <!ENTITY % elt “b”>
]>
<racine><b/><b/><b/></racine>
```

Espace de noms
(Namespace)

Objectifs

- Distinguer les éléments et attributs issus de divers vocabulaires et pouvant partager le même nom. Exemple : `<set>`
  - Dans MathML : ensemble mathématique.
  - Dans SVG : fixe la valeur d'un attribut à une durée de temps précise
- Grouper tous les éléments XML appartenant à une même application XML pour que les logiciels puissent les reconnaître facilement.

Exemple

Un catalogue de tableaux:

```
<?xml version=”1.0” encoding=”utf-8”?>
<catalog>
 <painting>
  <title>Souvenir d'un jardin à Eden</title>
  <artist>Van Gogh</artist>
  <date>1888</date>
  <description>Deux femmes regardent sur la gauche ;
     une troisième travaille au jardin.</description>
 </painting>
 <painting>
  <title>La balançoire</title>
  <artist>Renoir</artist>
  <date>1876</date>
  <description>Jeune fille sur une balançoire.</description>
 </painting>
 ...
</catalog>
```

- Pour indexer le catalogue par un moteur de recherche, il est possible de lui ajouter des méta données en formats RDF et Dublin Core:

```
<RDF>
 <Description about=”http://mon-catalogue”>
  <title>Tableaux impressionistes</title>
  <creator>Benoît</creator>
  <date>1888</date>
  <description>Liste de tableaux célèbres</description>
  <date>2009</date>
 </Description>
</RDF>
```

- Reste à intégrer cet arbre au document, et un indexeur automatique pourra référencer le catalogue.

```
<?xml version=”1.0” encoding=”utf-8”?>
<catalog>
 <RDF>
  <Description about=”http://mon-catalogue”>
   <title>Tableaux impressionistes</title>
   <creator>Benoît</creator> <date>1888</date>
   <description>Liste de tableaux célèbres</description>
   <date>2009</date>
  </Description>
 </RDF>
 <painting>
  <title>Souvenir d'un jardin à Eden</title>
  <artist>Van Gogh</artist>
  <date>1888</date>
  <description>Deux femmes regardent sur la gauche ;
     une troisième travaille au jardin.</description>
 </painting>
 <painting>
  <title>La balançoire</title>
  <artist>Renoir</artist>
  <date>1876</date>
  <description>Jeune fille sur une balançoire.</description>
 </painting>
 ...
</catalog>
```

Solution : les espaces de nom

- On associe un espace de nom à un élément ou à un attribut comme suit:
  - `prefix:element`
  - `prefix:attribut`
- Le préfixe est associé à un espace de nom qui est sous la forme d'une URI (en général une adresse internet qui pointe sur la définition). On l'appelle nom qualifié.
- L'association se fait avec

```
<prefix:element xmlns:prefix=”URI”>
```

- La portée de la déclaration est l'élément où elle apparait et tous ces descendants.

Noms qualifiés classiques

- Espace de nom XHTML :
  http://www.w3.org/1999/xhtml
- Espace de nom MathML :
  http://www.w3.org/1998/Math/MathML
- Espace de nom SVG :
  http://www.w3.org/2000/svg
- Espace de nom Dublin Core :
  http://purl.org/dc

Exemple

```
<?xml version=”1.0”?>
<catalog>
 <rdf:RDF xmlns:rdf=”http://www.w3.org/TR/REC-rdf-syntax#”>
  <rdf:Description
          about=”http://mon-catalogue”
          xmlns:dc=”http://purl.org/dc”>
   <dc:title>Tableaux impressionistes</dc:title>
   <dc:creator>Benoît</dc:creator>
   <dc:date>1888</dc:date>
   <dc:description>Liste de tableaux célèbres
      </dc:description>
   <dc:date>2009</dc:date>
  </rdf:Description>
 </rdf:RDF>
 <painting>
  <title>Souvenir d'un jardin à Eden</title>
  <artist>Van Gogh</artist>
  <date>1888</date>
  <description>Deux femmes regardent sur la gauche ;
     une troisième travaille au jardin.</description>
 </painting>
 ...
</catalog>
```

- La DTD correspondante pourra utiliser une règle différente pour `dc:description` et pour `description`
- Les éléments sans préfixes ne sont dans aucun espace de nom.
- Un attribut sans préfixe (comme `about`) ne sont dans aucun espace de nom, même si ils appartiennent à un élément avec espace de nom.
- Certains préfixes sont canoniques, comme

  `dc, rdf, svg`.

Espace de nom par défaut

- On peut aussi utiliser `xmlns` seul :

```
<svg xmlns=”http://www.w3.org/2000/svg”
   width=”12cm” height=”10cm”>
 <ellipse rx=”110” ry=”130”/>
 <rect x=”4cm” y=”5cm” width=”3cm” height=”4cm”/>
</svg>
```

- Tous les éléments descendants de `<svg>` seront dans l'espace de nom.
- Mais les attributs : `width, height, ...` sont sans espace de nom.

Règle

- Pour connaitre l'espace de nom d'un élément :
  - Si préfixe, on cherche la définition en remontant vers les ancêtres
    - Si pas de préfixe : erreur !
  - Si pas de préfixe, on recherche la définition du préfixe par défaut en remontant vers les ancêtres.
    - Si pas de définition : pas d'espace de nom.
- Pour connaitre l'espace de nom d'un attribut :
  - Si préfixe, on cherche la déf inition en remontant vers les ancêtres
    - Si pas de préfixe : erreur !
  - Si pas de préfixe, pas d'espace de nom.

```
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
<title>Mon fichier HTML</title>
</head>
<body>
 <h1>Une section</h1>
 <p>Un paragraphe</p>
 <hr />
</body>
</html>
```

Est équivalent à

```
<html:html xmlns:html=”http://www.w3.org/1999/xhtml”>
<html:head>
<html:title>Mon fichier HTML</html:title>
</html:head>
<html:body>
 <html:h1>Une section</html:h1>
 <html:p>Un paragraphe</html:p>
 <html:hr />
</html:body>
</html:html>
```

```
<book xmlns="urn:loc.gov:books"
      xmlns:isbn="urn:ISBN:0-395-36341-6">
<title>Cheaper by the Dozen</title>
<isbn:number>1568491379</isbn:number>
<notes>
 <p xmlns="http://www.w3.org/1999/xhtml">
   This is a <i>funny</i> book!
 </p>
</notes>
</book>
```

Est équivalent à

```
<bk:book xmlns:bk="urn:loc.gov:books"
         xmlns:isbn="urn:ISBN:0-395-36341-6">
<bk:title>Cheaper by the Dozen</bk:title>
<isbn:number>1568491379</isbn:number>
<bk:notes>
 <html:p xmlns:html="http://www.w3.org/1999/xhtml">
   This is a <html:i>funny</html:i> book!
 </html:p>
</bk:notes>
</bk:book>
```

```
<?xml version=”utf-8”?>
<html xmlns=”http://www.w3.org/1999/xhtml”
      xmlns:xlink=”http://www.w3.org/1999/xlink”>
<head><title>Trois espaces de nom</title></head>
<body>
 <h1 align=”center”>Un dessin</h1>
 <svg xmlns=”http://www.w3.org/2000/svg”
    width=”12cm” height=”10cm”>
  <ellipse rx=”110” ry=”130”/>
  <rect x=”4cm” y=”5cm” width=”3cm” height=”4cm”/>
 </svg>
<p xlink:type=”simple” xlink:href=”eclipse.html”>
Pour en savoir plus sur les éclipses.
</p>
<p xlink:type=”simple” xlink:href=”rect.html”>
Pour en savoir plus sur les rectangles.
</p>
<hr />
<p>Dernière modification le 13 mai 2002.</p>
</body>
</html>
```

Espaces de nom et DTDs

- Les espaces de nom sont indépendant des DTDs.
- En particulier, l'attribut `xmlns:xxx` doit être déclaré.
- Ils ne modifient en rien la syntaxe de la DTD:

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

- Le nom de l'élément dans la règle doit correspondre exactement au préfixe choisi: une fois la DTD établie, le préfixe n'est plus modifiable.
- On peut jouer avec les entités paramètres si c'est vraiment nécéssaire.