

XML (Extensible Markup Language)

Nous abordons ici XML qui se présente comme le langage universel d'échange d'informations.

1 Introduction

Xml est, comme html, un langage de balisage permettant de décrire les données et leur structure. Contrairement à html, les balises ne sont pas prédéfinies ce qui justifie le X (eXtensible).

Html est conçu pour présenter l'information, Xml est essentiellement utilisé pour permettre de structurer et échanger de l'information (éventuellement de la stocker) sous forme de fichier 'texte' (xml constitue une sorte d'«esperanto» pour les données). Xml permet ainsi de communiquer et de valider de l'information en étant indépendant d'un environnement spécifique ce qui l'amène à devenir un outil indispensable au net et au partage d'informations entre des milieux hétérogènes.

Un premier exemple

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<classe>
<cours>Application distribuées</cours>
<etudiants>
<etudiant>
<no>23456</no><nom>Dupont</nom><prenom>Marc</prenom>
</etudiant>
<etudiant>
<no>24235</no><nom>Durand</nom><prenom>Françoise</prenom>
</etudiant>
</etudiants>
</classe>
```

La première ligne stipule la version xml utilisée ainsi que le type d'encodage.

Les différentes balises, librement choisies – mais les plus parlantes possible –, doivent s'imbriquer parfaitement et toute balise ouverte doit être fermée.

Les balises sont sensibles à la casse.

Tous les éléments du texte doivent être repris dans l'élément « racine » (ici classe).

Un document xml présente un arbre dont les nœuds sont des éléments qui eux-mêmes peuvent porter du texte, des éléments, des attributs, ...

Nous voyons sur cet exemple que ce document xml se décrit lui-même.

Les éléments peuvent être répétés et leur ordre est significatif :

```
<etudiant>
<no>23456</no><nom>Dupont</nom><prenom>Marc</prenom>
<mails>
<mail>mdupont@heb.be</mail>
<mail>marc_dupont@hotmail.com</mail>
</mails>
</etudiant>
```

Les éléments peuvent être «qualifiés» au travers d'attributs ayant un nom et une valeur. La valeur doit être fournie entre ' ou ".

```
<cours année="2">Application distribuées</cours>
```

2 Les notions constitutives de xml

Éléments et Attributs

Un élément est un nœud d'un arbre et peut être répété. L'ordre des éléments est significatif.

Un attribut est une caractéristique d'un élément et ne peut pas être répété. L'ordre des attributs est non significatif.

Il n'est pas aisé de percevoir quand il est préférable de définir des attributs plutôt que des éléments.

Les caractéristiques imposant d'utiliser un élément peuvent être :

- Les données reprises sont destinées à être publiées
- Si il est probable que l'évolution amènera à structurer cette information (descendant, ...)
- Si cette donnée est à utiliser plusieurs fois
- Si l'ordre des éléments de même type est significatif

Caractéristique imposant d'utiliser un attribut : pour faire référence à un autre élément :

```
<étudiant opt="G">...
<option acro="G">Gestion ...
```

Caractéristique incitant à utiliser un attribut : spécification d'un statut, usage, ... d'un objet plutôt que d'une propriété (ceci est très flou)

```
<personne statut="prof" >...
```

Entité

Une entité est une référence à une information. Seules 5 entités sont prédéfinies :

D'autres entités peuvent être définies. Nous le verrons plus loin.

&	&
<	<
>	>
"	"
'	'

Données

Il s'agit du texte proprement dit.

Commentaires

Les commentaires se placent entre <!-- et --> .

3 Les documents «bien-formés» (well-formed)

Un document Xml respectant les règles ébauchées ci-dessus sera dit «bien-formé». Pour vérifier que notre texte est bien-formé, nous pouvons le soumettre à un «parser» (analyseur syntaxique).

Pour manipuler les documents xml nous utiliserons les outils gratuits de Altova¹. Ces outils sont téléchargeables à l'adresse: <http://www.altova.com/altovaxml.html> et leur documentation à l'adresse: <http://www.altova.com/documents/AltovaXML.pdf>.

Pour assurer la vérification du fait qu'un document xml soit bien formé, il faut exécuter la commande suivante :

AltovaXML -wellformed MonFichier.xml

4 Les documents valides

Nous avons vu qu'un document xml respectant la syntaxe du langage est dit « bien-formé ».

Lorsque nous recevons un document xml nous pouvons contrôler que la syntaxe est correcte mais il est aussi important de pouvoir vérifier qu'un certain nombre de contraintes d'intégrité des données représentées soient respectées. La description de ces contraintes peut se faire suivant deux voies : spécification d'un modèle sous forme de **DTD (Document Type Definition)** ou de **schéma (XSD)**. Dans le cadre de ce cours nous nous contenterons d'aborder les DTD. Les schémas permettent de réaliser des descriptions plus fines mais leur apprentissage demanderait trop de temps.

Un document xml sera **valide** par rapport à un modèle lorsqu'il respectera l'ensemble des règles et propriétés définies dans le modèle.

DTD (Document Type Definition)

Un DTD peut être interne ou externe. S'il est interne, repris à l'intérieur même du document xml, le document portera sa propre description. S'il est externe, il reprendra une description sur laquelle un groupe s'est mis d'accord et il permet à chacun d'évaluer la validité du document par rapport à la référence commune.

Éléments

Les éléments se déclarent comme suit

<!ELEMENT *nom spécification* >

<!ELEMENT monElement EMPTY > pour un élément vide (cf
 en html)

<!ELEMENT monElement (#PCDATA) > pour un élément non répétitif sans descendant

<!ELEMENT monElement (elementFils1, elementFils2, ...) > pour n éléments distincts

<!ELEMENT monElement (element1| element2) > pour un élément d'un des deux types fournis

les multiplicités :

+	1..n
	1..1
*	0..n
?	0..1

¹ Il existe aussi XMLStarlet : <http://xmlstar.sourceforge.net/> disponible à la fois sous Linux et Windows.

Exemple

Le dtd correspondant à l'exemple présenté ci-dessus serait donc :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT classe (cours, etudiants)>
<!ELEMENT etudiants (etudiant)*>
<!ELEMENT cours (#PCDATA)>
<!ELEMENT etudiant (no,nom,prenom,mails?)>
<!ELEMENT mails (mail*)>
<!ELEMENT mail (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
```

Validation

Pour une validation via un dtd externe, nous pouvons entrer la commande :

```
AltovaXML -validate MonFichier.xml -dtd MaDtdExterne.dtd
```

Ou ajouter la ligne `<!DOCTYPE classe SYSTEM " MaDtdExterne.dtd ">` dans le fichier xml et lancer la commande `AltovaXML -validate MonFichier.xml`

Pour une validation via un dtd interne :

```
AltovaXML -validate MonFichier.xml
```

Où nous aurions :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE classe [
<!ELEMENT classe (cours, etudiants)>
<!ELEMENT etudiants (etudiant)*>
<!ELEMENT cours (#PCDATA)>
<!ELEMENT etudiant (no,nom,prenom)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
]>
<classe>
<cours>Application distribuées</cours>
<etudiants>
<etudiant>
<no>23456</no><nom>Dupont</nom><prenom>Marc</prenom>
...
```

Exemples-Exercices :

Modifier le fichier xml et son dtd pour qu'à chaque étudiant soit associé au moins un prénom.

Modifier les données et le modèle pour que le moyen de locomotion privilégié de l'étudiant pour arriver à l'institut soit repris : voiture (donnez la marque), vélo (rien), tram(n° de ligne), train(gare bruxelloise), autre(rien).

Attributs

Les déclarations d'attributs sont plus complexes :

```
<!ATTLIST nom-élément nom-attribut type-attribut valeur-defaut>
```

type-attribut peut avoir, entre autres, les valeurs suivantes :

CDATA	Donnée 'caractères'
(en1 en2 ..)	Énumération des valeurs possibles
ID	La valeur est un id unique
IDREF	La valeur est une référence à un id d'un autre élément
IDREFS	La valeur est une liste de références
ENTITY	La valeur est une entité
ENTITIES	La valeur est une liste d'entités

Valeur-defaut peut présenter

value	La valeur par défaut
#REQUIRED	La valeur est obligatoire
#IMPLIED	La valeur facultative
#FIXED value	La valeur est immuable

Remarque : les attributs de type ID doivent obligatoirement comporter un caractère non numérique !

Exemples : cf. http://www.w3schools.com/dtd/dtd_attributes.asp

Exemples-Exercices :

Modifier les documents xml et dtd pour que chaque étudiant soit associé les attributs 'section' (obligatoire parmi G I et R) et annee (obligatoire parmi 1, 2 et 3).

Placer l'info de l'élément no comme attribut d'identification de l'étudiant (attention, placez une lettre devant le n°, E par exemple) et retirer l'élément 'no'.

Entités

Les entités définissent des raccourcis. Elles peuvent être internes ou externes.

Entité interne

Dans le dtd :

```
<!ENTITY institut "ESI : Ecole supérieure d'informatique">
```

Dans le xml :

```
<enseignement>&institut;</enseignement>
```

Entité externe

Dans le dtd :

```
<!ENTITY institut SYSTEM "http://www.esi.be/dtd/entite.dtd">
```

Dans le xml :

```
<enseignement>&institut;</enseignement>
```

Exemples-Exercices :

Modifier les documents précédents pour spécifier la valeur de l'élément cours au travers d'une entité : prévoir dans le dtd les entités adi2, alg2, bdg2.

5 Affichage Xml à l'aide d'un navigateur

Les navigateurs permettent d'afficher des documents xml sous forme arborescente (attention comme pour tout traitement de document xml, le traitement s'arrête dès qu'une erreur est rencontrée contrairement au traitement d'un document html).

Il est aussi possible d'adjoindre à un document xml un fichier css qui donnera des spécifications d'affichage.

Exemple

Dans l'exemple de départ, nous pouvons ajouter la ligne :

```
<?xml-stylesheet type="text/css" href="exemple.css"?>
```

et écrire un css :

```
<style type="text/css">
classe { }
cours { display: block; font-size: 200%;height: 1cm; text-align: center; }
etudiant { display: block; background-color: #F5F5DC; margin-bottom: 20px; }
nom { font-size: 16pt ; font-family: arial; font-weight: bold; color: Maroon; padding-left: 10px; }
prenom { font-size: 11pt ; font-family: arial; font-weight: bold; color: Maroon; padding-left: 10px; }
mails { display: block; margin-left: 2cm; }
mail { display: list-item; }
</style>
```

6 Exercices

Modifiez le xml de l'exemple et le dtd associé pour (travaillez étape par étape)

- Ajouter une info indiquant si un étudiant est doubleur ou non (Attribut ou entité ?)
- Modifiez le document de telle sorte que le même cours puisse être dispensé à plusieurs classes et que l'on connaisse le nom du professeur.
- Ajouter l'institut dans lequel est dispensé ce cours au travers d'une entité

Le fichier catalogue.xml vous étant fourni, en vous basant sur votre connaissance du schéma de eVente vu en atelier logiciel, proposez un dtd adapté. De plus, prévoyez un css de mise en forme.

7 Espaces de nom

Si l'on fusionne des documents xml d'origine différentes on risque d'être confronté à des homonymies. Ce problème peut être pallié en qualifiant les éléments au travers de la notion d'«espace de nom».

Exemple: Supposons que nous ayons dans un document xml :

```
<livre>
<titre>jkglug</titre>
<auteur>khlkuhl</auteur><auteur>ujggk</auteur>
<isbn>...</isbn>
</livre>
```

et dans un autre

```
<livre>
  <date>...</date><commande>....</commande>
  <camion>...</camion>
</livre>
```

Il est évident que si ces deux extraits cohabitent dans un même document nous seront confrontés à une ambiguïté. Nous pourrions lever ce problème de la manière suivante :

```
<b:livre xmlns:b="http://www.malibrairie.com/definition">
<b:titre>jkglug</b:titre>
<b:auteur>khlkuhl</b:auteur><b:auteur>ujggk</b:auteur>
<b:isbn>...</b:isbn>
</b:livre>
```

ou encore

```
<livre xmlns="http://www.malibrairie.com/definition">
<titre>jkglug</titre>
<auteur>khlkuhl</auteur><auteur>ujggk</auteur>
<isbn>...</isbn>
</livre>
```

8 Références

Parmi les nombreuses références disponibles nous vous proposons toujours :

<http://www.w3schools.com/xml> et <http://www.w3schools.com/dtd>