XML & DTD

Plan

1. Introduction
2. Caractéristiques
3. Création d'un document XML
4. DTD

Introduction: Langage XML

- XML = eXtensible Markup Language
- XML est issu de la Gestion de Documents (GED)
- Séparation du fond de la forme.
  - Forme = présentation à partir de la structure (style)
  - Fond = structure + données (contenu)
- Multiples précurseurs dont les plus connus :
  - SGML(*Standard Generalized Markup Language*) pour la structuration
  - HTML pour la présentation
  - Approches mélangeant parfois le fond et la forme !

- méta-langage = un langage qui permet de définir d'autres langages : *vocabulaire XML* ou *application XML*

DocBook. Ensemble de définition de balises pour les livres et les articles sur les logiciels et matériels informatiques.

LegalXML. Ensemble de définition de balises pour le domaine juridique.

MathML: Mathematical Markup Language. Ensemble de définition de balises permettant de décrire le contenu et la présentation des formules mathématiques afin de les publier sur le Web.

MusicML: Music Markup Language. Un essai de normalisation pour l'échange de feuilles de musique en XML. Proposé sous forme d'une DTD. Une autre proposition sous le nom XMLMusic existe.

WSDL: Web Services Definition Language. Ensemble de définition de balises de services Web.

…

Galaxie XML

- **XSD : XML Schema Definition.** Le langage de description de format de documeny XML.
- **XSL : Extensible Styling Language.** Le langage d'écriture des feuilles de style et règles de transformation pour XML.
- **XPath : XML Path Language.** Le langage de cheminement dans les arbres XML.
- **XQuery : XML Query Language.** Le langage de requêtes pour XML.
- XAML : Langage d'écritures de composants graphiques

Introduction: A quoi sert XML?

- Echange de données entre des systèmes hétérogènes

Cadre d'une architecture client/serveur

- Couche présentation
- Couche application
- Couche données

Dans ce cadre, XML est un véhicule pour les informations qui livre sur le poste client des données exploitables dans un format de données universel.

Caractéristiques

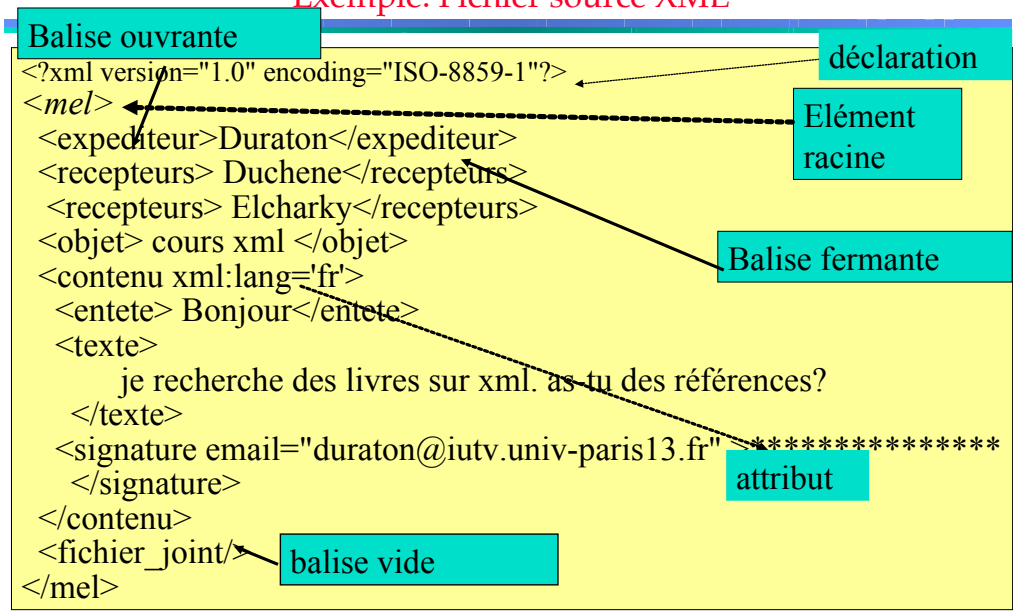
- Objectif principal :
  - Décrire le contenu d'un document sans son affichage
  - Permettre de séparer les données de leur présentation
- Structure générale :
  - Description de la structure logique d'un document à l'aide d'un système de **balises**
  - Les informations manipulées sont décrites sous la forme d'une structure arborescente.

Exemple

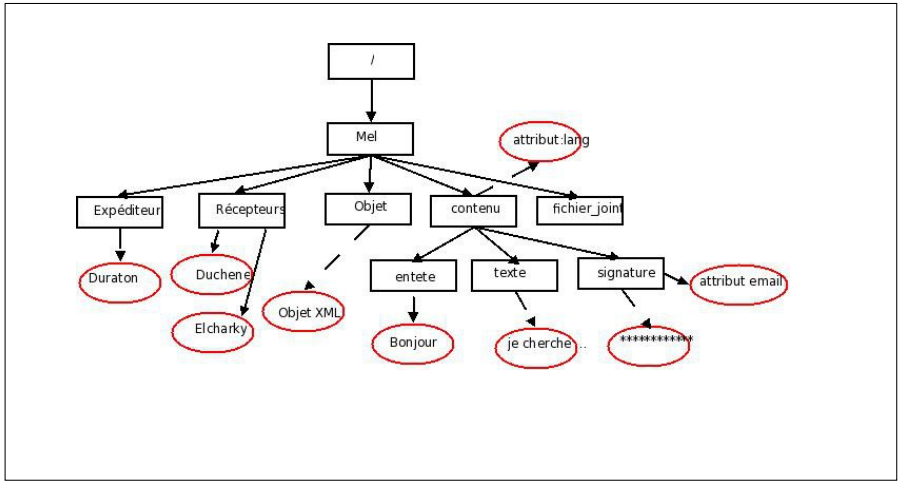
Un document décrivant un message : un message est constitué

- d'un expéditeur,
- d'un ou plusieurs récepteurs,
- d'un objet,
- éventuellement de fichiers joints
- d'un contenu, le contenu peut contenir
  - éventuellement une entête,
  - un texte,
  - une signature.

Exemple: Fichier source XML



Exemple : Représentation en arbre



Composition d'un fichier XML

- Un fichier XML est composé :
  - d'un prologue,
  - d'un élément racine
  - et d'un arbre constitué d'éléments imbriqués les uns dans les autres (ayant une relation parent-enfant) et d'éléments adjacents.

Prologue

- Le prologue est constitué d'au moins une déclaration XML et éventuellement d'une déclaration de DTD
- Déclaration XML

Exemple

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

Il peut contenir le codage des caractères

- ISO-8859-1
- UTF-8

Il peut indiquer si un autre fichier doit être chargé

- standalone: yes/no *par défaut (yes)*

Eléments

Un élément

- est déterminé par un **nom :** un nom d'élément doit commencer par une lettre ou un souligné, il peut contenir des chiffres, des lettres, des traits d'union, des points, des double-points ou soulignés.
- est composé par un couple de balises :
  - une balise ouvrante : < nom_element>
  - une balise fermante : </nom_element>

Le contenu peut être :

- non vide : contient du texte (tous les caractères sont acceptés sauf le"&" et le caractère plus petit que (<)) ou d'autres éléments.
- Vide : exemple <HR> </HR> ou <HR/>

exemple : <Nom> Duraton </Nom>

```
<Nom>
    <Nom_patronymique> Duraton</Nom_patronymique>
     <Nom_marital> Duduche </ Nom_marital>
</Nom>
```

Attributs (1)

- Un élément présente des caractéristiques appelées **attributs**
- Syntaxe des attributs

nom = valeur

Exemple:

<*signature* email="duraton@iutv.univ-paris13.fr" >

- Un élément peut contenir plusieurs attributs séparés par un blanc.
- Les valeurs des attributs doivent être entourées de guillemets simples (') ou doubles("). Si la valeur contient un type de guillemet, utilisez l'autre type comme délimitateur.

Exemple:

<choix test = 'msg="salut" '>

Attributs (2)

Les attributs réservés

xml:lang indique le langage utilisé (fr pour le français)

xml:space= "default|preserve" (un espace blanc à l'intérieur d'un élément est significatif)

xml:link signale à un processeur Xlink qu'un élément particulier est un lien

Commentaire

- **les commentaires:** entre <!-- et --> éventuellement sur plusieurs lignes.

Exemples:

<!-- ========== -->

<!-- _____-->

attention

incorrect <!---------->

Les entités

- Une entité est un raccourci prédéfini
- La syntaxe est *caractère* **&** suivi d'un identificateur suivi **;**
- Les entités servent à écrire un caractère impossible ou à simplifier l'écriture d'une chaîne de caractères:
  - l'insertion d'un caractère impossible à saisir comme le caractère "<" décrit par l'entité **<**
  - Exemple : ´**Ex ≥ 0, f(y) < z**

<equation>∀ x ≥ 0, f(y) < z </equation>

- Une entité définit un contenu qui sera inséré dans le document

Les instructions de traitement

- Une instruction de traitement ou PI est une balise contenue dans le document XML mais grammaticalement non prise en compte.
- Elles permettent aux développeurs de placer des informations spécifiques pour une application à l'intérieur du document.
- Une instruction de traitement (PI) contient 2 parties :
  - la cible
  - les données
- syntaxe <? *cible donnée* ?>

La cible est un mot-clé qu'un processeur XML utilise pour déterminer si les données lui sont destinées ou non

Exemple

<?xml-stylesheet type="text/xsl" href="biblio.xsl"?>

l'application est xml-stylesheet, le processeur de feuille de style du XML. Cette instruction de traitement est utilisée par les navigateurs pour la mise en forme du document.

Les sections CDATA

- Elles permettent de définir des chaînes de caractères qui ne sont pas interprétées par l'analyseur.
- Une section CDATA commence par un délimiteur de 9 caractères: **<![CDATA[** et se termine par délimiteur de 3 caractères **]]>**.
- Le contenu peut être quelconque. Il n'est pas interprété par l'analyseur

exemple:

<para> on peut écrire <![CDATA[if (&x < &y)]]> </para>

Les espaces de noms

- **Définition:**

Un espace de noms permet de qualifier de façon unique les éléments et attributs.

Exemple: définition d'une balise fraise

Espace de noms "outillage" <outillage:fraise>

Espace de noms "nourriture" <nourriture:fraise>

Un espace de noms doit être déclaré pour être utilisé

- Syntaxe

un élément est décrit par *nom_de_domaine:nom_element*

un nom de domaine ou espace de nom est défini dans le document par l'attribut xmlns: *nom_de_domaine* = "URI"

ex: <xs:schema **xmlns**:xs="http://www.w3.org/2001/XMLSchema">

URI = Uniform Resource Identifier - adresse sur le web

Création de documents XML

Définition:

Un document XML est bien formé lorsqu'il est syntaxiquement correct.

- Contient une déclaration XML
- Contient un élément racine contenant tous les autres
- Toute balise ouverte doit être fermée
- La casse doit être respectée (<Debut> et <debut> sont distinctes)
- Il ne peut pas y avoir de balises imbriquées ou croisées

Création de documents XML

Définition

Un document XML est valide s'il est bien formé et s'il respecte les règles définies dans la grammaire associée au document XML

On peut décrire une grammaire d'un document XML avec une DTD ou avec XML Schema

Une grammaire n'est pas obligatoire mais recommandée

DTD=Document Type Definition

Définition:

Une DTD est composée d'une série d'expressions qui définissent la structure logique de documents XML.

- Permet de définir le «vocabulaire» et la structure qui sont utilisés dans le document XML
- Grammaire du langage dont les phrases sont des documents XML (instances)

Une DTD peut être externe ou interne

interne: La DTD et le document XML se trouvent dans le même fichier

Une DTD externe s'applique à plusieurs documents. Elle est définie dans un fichier distinct.

DTD interne: la dtd est définie dans le fichier xml

Déclaration d'une DTD

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

 <!DOCTYPE racine  [sous-ensemble interne]>
<racine>
……le document xml
</racine>
```

DTD externe: la dtd est définie dans un fichier séparé

Déclaration d'une DTD externe:
<!DOCTYPE *racine* SYSTEM "url ">
url : donne l'adresse du fichier qui contient la dtd
Cette balise est à inclure dans le fichier xml validé par cette DTD

Déclaration d'élément simple

- <! ELEMENT *balise* (*définition*) >
  - Le paramètre *définition* représente soit un type de donnée prédéfini, soit un élément de données composé, constitué lui même d'éléments
  - Types prédéfinis
    - ANY : L'élément peut contenir tout type de donnée
    - EMPTY : L'élément ne contient pas de données spécifiques
    - #PCDATA : L'élément doit contenir une chaîne de caractère
  - Exemple

<! ELEMENT Expediteur (#PCDATA)>
<*expediteur*>Duraton</*expediteur*>

Déclaration d'élément composé

- Définit une séquence ou un choix d'éléments
- Syntaxe spécifique avec opérateurs de composition d'éléments :
  - <! ELEMENT balise (*composition*) >

Opérateur	Signification	Exemple
+	L'élément doit être présent au minimum une fois	A+
*	L'élément peut être présent plusieurs fois (ou aucune)	A*
?	L'élément peut être optionnellement présent	A?
|	L'élément A **ou** B peuvent être présents (pas les deux)	A|B
,	L'élément A doit être présent et suivi de l'élément B	A,B
()	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

Exemple d'élément composé

<!ELEMENT contenu (entete?,texte, signature)>
<!ELEMENT entete (#PCDATA)>
<!ELEMENT texte (#PCDATA)>
<!ELEMENT signature (#PCDATA)>

Exemple XML correspondant

```
<contenu xml:lang='fr'>
   <entete> Bonjour</entete>
   <texte>je recherche des livres sur xml. as-tu des références?
   </texte>
   <signature email="duraton@iutv.univ-paris13.fr" >
       **************</signature>
</contenu>
```

DTD: Déclaration d'un attribut

<!ATTLIST *nom-élément nom_attribut type_attribut Mode*>

Exemple:

<!ATTLIST signature nom CDATA #IMPLIED telephone CDATA #IMPLIED email CDATA #REQUIRED>

type d'attributs

type_attribut définit le type de donnée de l'attribut choisi:

- CDATA
  - Chaînes de caractères entre guillemets ("aa") non analysées
- Enumération
  - Liste de valeurs séparées par |
  - <! ATTLIST *balise Attribut* (Valeur1 | Valeur2 | ...) >
- ID et IDREF
  - Clé et référence à clé

Mode précise le caractère obligatoire ou non de l'attribut

#REQUIRED, #IMPLIED ou #FIXED….

DTD: Mode

Déclaration de défaut

- une valeur par défaut
- #REQUIRED attribut est obligatoirement présent
- #IMPLIED attribut est facultatif
- #FIXED le domaine de valeurs possibles est inclus dans la déclaration. Prend la 1ère valeur par défaut

Attribut de type NMTOKEN- NMTOKENS

NMTOKENS:Les attributs de type NMTOKEN ne peuvent contenir que des lettres, des chiffres, un point [.] , un tiret [-], un trait de soulignement [_] et un deux-points [:].

```
<!ELEMENT attributes (#PCDATA)>
<!ATTLIST attributes
    aaa CDATA #IMPLIED
    bbb NMTOKEN #REQUIRED
    ccc NMTOKENS #REQUIRED>
<attributes aaa="#d1" bbb="a1:12" ccc=" 3.4 div    -4"/>
```


Type énuméré

- Permet de limiter la liste de valeurs possibles pour un attribut. L'attribut est défini alors comme étant de type énuméré. Donner une autre valeur dans le fichier XML provoque une erreur.

Exemple de déclaration d'une liste de choix d'attributs :

```
<!ELEMENT img EMPTY>
  <!ATTLIST img format (BMP | GIF | JPEG) "JPEG">
```

La valeur par défaut est JPEG

Type ID

Ce type sert à indiquer que l'attribut en question peut servir d'identifiant dans le fichier XML. Deux éléments ne pourront pas posséder le même attribut possédant la même valeur.

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ensELT [<!ELEMENT ensELT (elt1*)>
<!ELEMENT elt1 (#PCDATA)>
  <!ATTLIST elt1 attr ID #IMPLIED>]>
  <ensELT>
<elt1 attr="machin"></elt1>
<elt1 attr="truc"></elt1>
<elt1 attr="machin"></elt1>
</ensELT>
```

Test

```
> dtdvalid testId.xml
file: testId.xml:E:8:1: ID "machin" already defined [cvc-id.2]
```


Exemple de ID et IDREF

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT(PERSONNE*)>
<!ELEMENT PERSONNE (#PCDATA)>
<!ATTLIST PERSONNE PNUM ID #REQUIRED>
<!ATTLIST PERSONNE MERE IDREF #IMPLIED>
<!ATTLIST PERSONNE PERE  IDREF #IMPLIED>
]>
<DOCUMENT>
<PERSONNE PNUM = "P1">Marie</PERSONNE>
<PERSONNE PNUM = "P2">Jean</PERSONNE>
<PERSONNE PNUM = "P3" MERE="P1" PERE="P2">Pierre</PERSONNE>
<PERSONNE PNUM = "P4" MERE="P1" PERE="P2">Julie</PERSONNE>
</DOCUMENT>
```

Attribut de type IDREFS

<!ATTLIST element_name attribute_name **IDREFS** default_value>

Exemple :

<!ATTLIST individual individual_id ID #REQUIRED parent_id **IDREFS** #IMPLIED>

```
<individuals>
  <individual individual_id="e10001" parent_id="e10002 e10003">
    <first_name>Bart</first_name>
    <last_name>Simpson</last_name>
  </individual>
  ....
</individuals>
```


DTD: les entités

Définition

Une entité est une référence à quelque chose, soit un nom de variable (entité interne), soit un alias (entité externe)

plusieurs types d'entités

Syntaxe

syntaxe: <!ENTITY *NomEntité valeur*>

utilisation: *&NomEntité;*

On peut faire référence à un objet externe, image, autre fichier xml ..

Les entités internes caractères

Elle servent à donner un nom facilement lisible à des caractères qui ne sont pas représentables dans l'alphabet utilisé, ou qui ne sont pas disponibles au clavier.

Exemples tirés de la DTD du langage HTML 4.01 :

<!ENTITY nbsp " ">

<!ENTITY eacute "é">

Exemples: caractères du langage XML

& : & < : < > : > " : " ‘ : '

Les entités de caractères définies dans une DTD peuvent être utilisées dans un document XML référençant cette DTD à l'aide de la notation &NomEntité;.

Les entités internes

- Exemple :

Une entité déclarée dans une DTD

ex: <!ENTITY deg "° " >

utilisation: il fait 25 ° C

<!ENTITY Mr " Monsieur ">

<!ENTITY Mme "Madame">

<Personne>

  &Mr;

</Personne>

DTD: exemple

contenu du fichier mel.dtd

```
<!ELEMENT mel (expediteur,
   recepteurs+,objet?,contenu,fichier_joint*)>
<!ELEMENT expediteur ( #PCDATA)>
<!ELEMENT recepteurs (#PCDATA)>
<!ELEMENT objet (#PCDATA)>
<!ELEMENT contenu (entete?,texte, signature)>
<!ATTLIST contenu xml:lang NMTOKENS #REQUIRED>
<!ELEMENT entete (#PCDATA)>
<!ELEMENT texte (#PCDATA)>
<!ELEMENT signature (#PCDATA)>
<!ATTLIST signature nom CDATA #IMPLIED telephone CDATA
   #IMPLIED email CDATA #REQUIRED>
<!ELEMENT fichier_joint (#PCDATA)>
```


Exemple

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE mel SYSTEM "mel.dtd">
<mel>
  <expediteur>Duraton</expediteur>
  <recepteurs> Duchene</recepteurs>
  <recepteurs> Dutruc</recepteurs>
  <objet> cours xml </objet>
  <contenu xml:lang='fr'>
    <entete> Bonjour</entete>
    <texte>
      je recherche des livres sur xml. as-tu des références?
    </texte>
    <signature email="duraton@iutv.univ-paris13.fr" >
     ***************
    </signature>
  </contenu>
  <fichier_joint/>
</mel>
```

- Une DTD externe peut être complétée par des déclarations internes (Exemple)

```
<?xml version="1.0" encoding='ISO-8859-1' standalone='no' ?>
<!DOCTYPE exemple SYSTEM "exemple.dtd"> [
<!-- déclarations internes -->
<!ATTLIST exemple lang (fr | en | it) #REQUIRED >
<!ENTITY FR "en français" >
<!-- fin des déclarations internes -->
]>
<!-- début de l'instance -->
<exemple lang='fr'> texte &FR; de l'instance de document
correspondant à la DTD définie ci-dessus
</exemple>
```

- le fichier DTD (Exemple : exemple.dtd)

```
<!-- début de la DTD -->
<!ELEMENT exemple (#PCDATA)>
<!-- fin de la DTD -->
```


Limitations des DTD

- Les DTD ne sont pas au format XML.
- Les DTD ne supportent pas les "espaces de nom"
- Le "typage" des données est extrêmement limité.