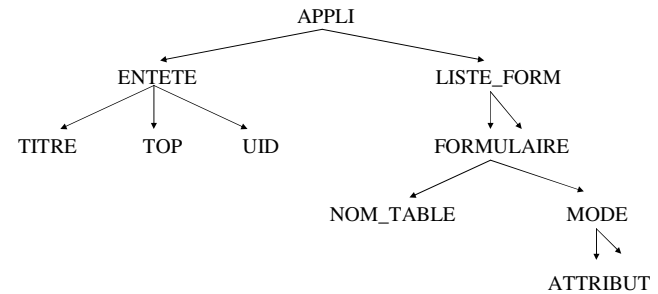


XML : la norme

Document XML, DTD et schémas XML

Document structuré (représentation arborescente)



Document structuré en XML

```
<?xml version="1.0"?>
<APPLI>
  <ENTETE>
    <UID>citcom/citcom@MICA</UID>
    <TITRE>MonAppli</TITRE>
    <TOP>[body bgcolor=#dddfaf][hr]</TOP>
  </ENTETE>
  ...
</APPLI>
```

Document XML (2)

```
<LISTE_FORM>
  <FORMULAIRE>
    <NOM_TABLE>VINS</NOM_TABLE>
    <MODE valeur="QBE"/>
  </FORMULAIRE>
  <FORMULAIRE>
    <NOM_TABLE>PRODUCTEURS</NOM_TABLE>
    <MODE valeur="INS"/>
  </FORMULAIRE>
  <FORMULAIRE>
    <NOM_TABLE>VINS</NOM_TABLE>
    <MODE valeur="NOR">
      <ATTRIBUT>ANNEE</ATTRIBUT>
      <ATTRIBUT>CRU</ATTRIBUT>
    </MODE>
  </FORMULAIRE>
</LISTE_FORM>
```

Représentation d'un document en XML

5

XML comme méta-langage de balisage

- Même principe de balisage qu'en HTML (texte est entouré de balises ou éléments)
- À la différence de HTML l'ensemble des balises utilisables est ouvert (sauf si présence d'une DTD ou d'un schéma XML)
- Permet de définir le niveau de finesse que l'on souhaite (il n'y a pas une seule façon de baliser un document)
- Pas d'association implicite de caractéristiques de présentation à une balise

6

Documents bien-formés

- Un document est bien-formé si son balisage est correct (on retrouve HTML, la vérification en plus)
- Tout document XML doit être bien-formé si on veut qu'il soit manipulable par un programme (c'est la propriété minimale)
- Balisage correct :
 - Tout balise ouverte doit être fermée ET
 - Tout balise ouverte dans un contexte doit être fermée dans ce même contexte

7

Exemple de document mal formé

```
<FORMULAIRE>
  <NOM_TABLE>VINS</NOM_TABLE>
  <MODE valeur="QBE "> 1
</FORMULAIRE>
<FORMULAIRE>
  <NOM_TABLE>PRODUCTEURS
  <MODE valeur="INS"/>
</FORMULAIRE>
  </NOM_TABLE> 2
```

1 : balise MODE non fermée

2 : balise NOM_TABLE fermée hors de son contexte (doit être dans le contexte d'un FORMULAIRE)

8

Attributs

- nom=valeur
- Valeur entre "
- un élément ne peut avoir deux attributs de même nom

```
<MODE valeur="NOR">...</MODE>
```

9

Attributs vs éléments

- Les attributs ne sont pas faits pour représenter de la structure
- les éléments sont plus faciles à manipuler (voir XSLT par exemple)
- les éléments se prêtent mieux à l'évolution

10

Typage d'un document XML

DTD et schéma XML

11

Typage

- Un document XML peut être typé ou non
- Typage permet de définir l'ensemble des balises utilisables dans un document ainsi que les contraintes entre balises
- Deux langages sont possibles :
 - DTD : vient du monde SGML, répandu mais assez restrictif
 - Schéma XML : nouveau et très (trop?) puissant

12

Document valide

- Un document est valide s'il est bien-formé et qu'il est conforme à son type (DTD ou schéma XML)
- Dans tous les cas, le type n'est pas obligatoire pour manipuler un document XML (c'est un plus)

13

Notion de DTD

14

Déclaration d'un élément

- Chaque balise doit être déclarée dans une déclaration `<!ELEMENT>`
- Une déclaration `<!ELEMENT>` donne le nom et le "type" du contenu de l'élément
- Le "type" est défini via des expressions régulières simples

15

Spécifications du type

- ANY : libre
- #PCDATA : données textuelles
- EMPTY : élément vide
- Séquences : *, +
- Choix : ?, |

16

Déclarations d'attributs dans une DTD

- **Cet élément de structure :**

```
<MODE valeur="NOR">
  <ATTRIBUT>ANNEE</ATTRIBUT>
  <ATTRIBUT>CRU</ATTRIBUT>
</MODE>
```

- **se déclare comme suit :**

```
<!ELEMENT MODE (ATTRIBUT*)>
<!ATTLIST MODE valeur (NOR, INS, QBE) "NOR" #REQUIRED>
```

17

DTD du document

```
<!ELEMENT APPLI ( ENTETE, LISTE_FORM ) >
<!ELEMENT ENTETE ( UID, TITRE, TOP ) >
<!ELEMENT TITRE ( #PCDATA ) >
<!ELEMENT TOP ( #PCDATA ) >
<!ELEMENT UID ( #PCDATA ) >
<!ELEMENT LISTE_FORM ( FORMULAIRE+ ) >
<!ELEMENT FORMULAIRE ( NOM_TABLE, MODE ) >
<!ELEMENT MODE ( ATTRIBUT* ) >
<!ATTLIST MODE valeur (NOR, INS, QBE) #REQUIRED "NOR">
<!ELEMENT ATTRIBUT ( #PCDATA ) >
<!ELEMENT NOM_TABLE ( #PCDATA ) >
```

18

Document XML avec référence à la DTD

```
<?xml version="1.0"?>
<!DOCTYPE APPLI SYSTEM "GenAppli.dtd">
<APPLI>
  <ENTETE> ...
</ENTETE>
<LISTE_FORM> ...
</LISTE_FORM>
</APPLI>
```

- DTD externe : cas standard (permet de partager une DTD entre plusieurs documents)
- Mais la DTD peut être inclus dans le document (DTD interne)

19

Domaines nominaux Namespaces

- **Besoin :** désigner de manière non ambiguë des éléments (par exemple fn peut être foot-notes ou fonction)
- **Principe :** document standard défini des noms d'éléments (MathML p.e). Ce document est désigné de manière unique par une URI
- tout nom d'élément doit être préfixé par l'URI qui le définit

20

Des DTD aux schémas XML

- DTD et schémas spécifient :
 - la *structure* des documents
 - "cet élément contient ces éléments, qui contiennent ces autres éléments, etc"
 - le *type* de chaque élément/attribut
 - "cet élément est un entier dans l'intervalle 0 à 12,000" (les DTDs ne savent pas exprimer ce genre de chose)

21

Pourquoi des schémas XML

Inconvénients des DTDs

- syntaxe non XML
 - une syntaxe pour les documents XML, une autre pour les DTD --> incohérence
- support limité des types de données
 - Pas d'intervalle sur les entiers par exemple, pas d'extensibilité, d'héritage, ...
 - Besoin d'un système de type plus riche, permettant de définir des schémas "à la BD"
 - DTD supporte 10 types; schéma XML supporte 44+ types

22

Principes des Schémas XML

Schéma XML sont une extension significative des DTDs:

- Types étendus
 - 44+ versus 10
 - extensible (on peut définir ses propres types)
- même syntaxe que les documents XML
- support de concepts objets
 - définition d'un type par extension ou restriction d'un autre
- support des ensembles
- notion de "clé"

23

Un exemple simple

- Transformer GenAppli.dtd en un schéma XML équivalent
 - conversion simple, i.e., UID, TITRE, ATTRIBUT, et NOM_TABLE vont rester des strings comme dans la DTD
 - on pourrait modifier le schéma XML en utilisant des constructions de type plus puissantes

24

DTD du document

```

<!ELEMENT APPLI ( ENTETE, LISTE_FORM ) >
<!ELEMENT ENTETE ( UID, TITRE ) >
<!ELEMENT TITRE ( #PCDATA ) >
<!ELEMENT UID ( #PCDATA ) >
<!ELEMENT LISTE_FORM ( FORMULAIRE+ ) >
<!ELEMENT FORMULAIRE ( NOM_TABLE, MODE ) >
<!ELEMENT MODE ( ATTRIBUT* ) >
<!ATTLIST MODE valeur (NOR, INS, QBE) #REQUIRED "NOR">
<!ELEMENT ATTRIBUT ( #PCDATA ) >
<!ELEMENT NOM_TABLE ( #PCDATA ) >

```

25

```

<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://e-charle.fr/TNS"
  xmlns="http://e-charle.fr/TNS"
  elementFormDefault="qualified">
  <xsd:element name="APPLI">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="ENTETE" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="LISTE_FORM" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="ENTETE">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="UID" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="TITRE" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="LISTE_FORM">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="FORMULAIRE" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

```

26

xsd = Xml-Schema Definition

```

<xsd:element name="FORMULAIRE">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="NOM_TABLE" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="MODE" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="MODE">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:attribute ref="valeur" type="typeVmode" use="required" value="NOR"/>
      <xsd:element ref="ATTRIBUT" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:simpleType name="typeVmode" base="xsd:string">
  <xsd:enumeration value="NOR"/>
  <xsd:enumeration value="INS"/>
  <xsd:enumeration value="QBE"/>
</xsd:simpleType>
<xsd:element name="TITRE" type="xsd:string"/>
<xsd:element name="UID" type="xsd:string"/>
<xsd:element name="NOM_TABLE" type="xsd:string"/>
<xsd:element name="ATTRIBUT" type="xsd:string"/>
</xsd:schema>

```

27

Référencer un schéma dans un document XML

```

<?xml version="1.0"?>
<APPLI xmlns="http://e-charle.fr/TNS" ①
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://e-charle.fr/TNS/genappli.xsd" ②
  <ENTETE><TITRE>Mon titre</TITRE>
  <UID>toto@titi</UID></ENTETE>
  <LISTE_FORM>
    <FORMULAIRE><NOM_TABLE>ESSAI</NOM_TABLE>
    <MODE valeur="NOR"><ATTRIBUT>A1</ATTRIBUT></MODE>
  </FORMULAIRE>
  </LISTE_FORM>
  ...
</APPLI>

```

1. Définit le namespace par défaut.
2. Indique au valideur de schéma quelle est la définition de 1

28

Synthèse schéma XML

- Permet de définir des modèles de documents beaucoup plus riches (on n'en a vu qu'une petite partie)
- la proposition est très complète, voire complexe
- peut être utilisé également pour des méta-données (à la place de RDF)
- commence à être supporté dans les outils

29

Bilan

- XML semble un bon compromis entre la richesse de SGML et la simplicité de HTML
- Un document XML sans type associé est (presque) un document HTML
- Un document XML avec type associé est (presque) un document SGML
- Le typage via les DTD devrait disparaître au profit des schémas XML plus puissants et plus XML
- Attention cependant, car les schémas XML sont très complexes

30