eXtensible Mark-up Language

Cours 1 : **Introduction au langage XML (XML, DTD, XML-Schema)**

Novembre 2013

- Version 4.0 -

PRESENTATION DU COURS

- 1 : Introduction au langage XML :
  - *XML : le langage, et quelques dialectes*
  - *DTD : comment définir une grammaire*
- 2 : Transformation de documents XML :
  - *XSLT : langage procédural pour transformer du XML*
  - *Xpath : indexer du XML*
- 3 : APIs XML
  - *langage indépendantes : SAX, DOM*
  - *pour Java : JAXP, JDOM, DOM4J*

Introduction au langage XML
SOMMAIRE DU COURS

- XML, qu'est-ce que c'est ?
- Définition
- Intérêt
- Structure d’un document XML
- Spécifications du langage XML

- Définitions de Types de Document (DTD)

XML, qu'est-ce que c'est ?

- langage de balisage pour la description de documents structurés (eXtensible Markup Language www.w3c.org/XML)
- rôle fondamental pour l'échange de données
- Exemples :
  - SBML : The Systems Biology Markup Language is a language for representing models of biochemical reaction networks http://sbml.org
  - RNAML : a standard syntax to easily express data on RNA sequence and structurehttp://www-lbit.iro.umontreal.ca/rnaml
  - GXL : Graph eXchange Language, is an XML-based standard exchange format for hypergraphs and hierarchical graphs. http://www.gupro.de/GXL
  - CML : Chemical Markup Language, designed to represent molecular information http://cml.sourceforge.net

Premier exemple

extrait d'un document CML pour l'arginine

```
<?xml version="1.0" encoding="UTF-8" ?>
<molecule convention="MDLMol" id="arginine" title="ARGININE"
    xmlns="http://www.xml-cml.org/schema">
   <atomArray>
      <atom id="a1" elementType="C" hydrogenCount="0"
          x2="0.7386" y2="0.1493"/>
      <atom id="a2" elementType="C" hydrogenCount="0"
          x2="-0.3772" y2="-0.6129"/>          ...
   </atomArray>
   <bondArray>
      <bond atomRefs2="a1 a2" order="1"/>
         ...
   </bondArray>
</molecule>
```

Prologue

Balise d'ouverture

Contenu

Element *atomArray*

Balise de fermeture

Element *bond* vide

Attribut

Valeur

XML, qu'est-ce que c'est ?

- balises descriptives (signification des données) plutôt que procédurales (présentation des données)
- libre, indépendant des plateformes logicielles ou matérielles
- XML est extensible: ne contient pas un ensemble fixé de balises
- les documents XML doivent être bien formés suivant une syntaxe définie, et peuvent donc être formellement validés
- XML est particulièrement adapté à l'échange de données et de documents.

XML, qu'est-ce que c'est ?

Parsers et Décodage des documents XML

- L’extraction des données d’un document XML se fait à l'aide d'un outil appelé analyseur syntaxique (en anglais **parser**, ou parseur) qui permet :
  - d'extraire les données d'un document XML (analyse du document ou parsing)
  - éventuellement, de vérifier la validité du document.

DEFINITION 1/2

- eXtensible Markup Language
  - Recommandation (norme) du W3C
  - Spécifiant un langage
  - Constitué d’un ensemble d’éléments appelés balises
  - Utilisable pour créer d’autres langages

- 2 concepts fondamentaux
  - Structure et présentation sont séparés
  - Les balises ne sont pas figées

DEFINITION 2/2

- Conséquences :
  - XML est un format de document
  - XML est un format de données (dialectes)
  - XML est un méta-langage (une famille de langages)

- En simplifié :

« XML est un langage de description de documents structurés » (www.w3c.org/XML).

INTERÊT de XML

- Richesse sémantique
  - Dédié au traitement des données
  - Soutenant une grande variété d’applications
- Facilité de mise en œuvre
  - Simple et lisible
  - Portable et facilement utilisable sur Internet
  - Assurant un développement aisé

SPECIFICATIONS DU LANGAGE XML 1/3

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE exemple SYSTEM "exmple.dtd">

<DEVIS>
   <!-- Entête -->
   <IDENTIFIANT>0401</IDENTIFIANT>
   <LIBELLE>Achats Réseaux</LIBELLE>
   <STATUT>Demande d'achat</STATUT>
   <FOURNISSEUR>Alcatel Commutation</FOURNISSEUR>
   <MONTANT devise="FRF">1452805.30</MONTANT>
   <DATE_LIVRAISON>12/12/2000<DATE_LIVRAISON>
   <!-- Lignes article -->
   <ARTICLE>
      <REFERENCE>ISML43S904</REFERENCE>
      <QUANTITE>280</QUANTITE>
      <PRIX_UNITAIRE devise="FRF">408.00</PRIX UNITAIRE>
      <IMAGE href="media/ISML43S904.gif"/>
   </ARTICLE>
   ...
</DEVIS>
```

Entête

Elément racine

Elément

Commentaire

Attribut

Elément vide

Contenu

SPECIFICATIONS DU LANGAGE XML 2/3

- Contenu d'un document XML :
  - Un prologue ou en-tête (déclarations)
  - Suivit d'un (SEUL) arbre d'éléments (balises)
  - Des commentaires

- Un document XML qui respecte les règles syntaxique est dit **bien formé**. Il est utilisable sans DTD (La grammaire de notre document XML)

- Un document XML bien formé qui respecte sa DTD est dit **valide**. Il est plus facile d'écrire des feuilles de style (XSL) pour les documents valides !

SPECIFICATIONS DU LANGAGE XML 3/3

- Résumé des spécifications :
  - Un document doit commencer par une déclaration XML
  - Toutes les balises avec un contenu doivent être fermées
  - Toutes les balises sans contenu doivent se terminer par les caractères />
  - Le document doit contenir un et un seul élément racine
  - Les balises ne doivent pas se chevaucher
  - Les valeurs d’attributs doivent être entre guillemets
  - La casse doit être respectée pour toutes les occurrences de noms de balise.
- Un document respectant ces critères est dit “bien formé”

XML : le prologue 1/2

• La première déclaration (qui est optionnelle) permet de définir la version et le codage du document.

```
<?xml
   version="1.0"
   encoding="iso-8859-1"
   standalone="yes"          (=> pas besoin de DTD externe)
   ?>
```

• L'encodage est basé sur la norme ISO 10646 (www.unicode.org).

• XML comprend automatiquement l'encodage UTF-8 et UTF-16 (UTF-8 est l'encodage par défaut).

XML : le prologue 2/2

• La référence à la DTD externe doit être placée au début du fichier :

```
<!DOCTYPE nom_d_élément SYSTEM "test.dtd">
```

• On peut enrichier la DTD externe avec des déclarations locales :

```
<!DOCTYPE nom_d_élément SYSTEM "test.dtd"
    [ déclarations ] >
```

• On peut se passer de référence à une DTD externe et définir toutes les balises dans le document XML :

```
<!DOCTYPE nom_d_élément [ déclarations ] >
```

XML : les commentaires

• en XML les commentaires se notent :

```
<!-- texte du commentaire -->
```

• Les contraintes d'utilisation sont

* pas de double tirets dans le texte,
* pas de commentaire dans un élément (l'exemple ci-dessous est incorrect),

```
<produit
  nom="DVD"
  prix='100' <!-- en euros -->
  />
```

* les commentaires sont ignorés (plus ou moins),

XML : les balises (éléments) 1/2

• Forme générale :

<nom_d_élément> contenu </nom_d_élément>

• Les noms sont libres (contrairement à HTML). Ils obéissent à quelques règles:

* 1er caractère { alphabétique, «-», «_» },
* les autres caractères { alphabétique, chiffre, «-», «_», «:» }.
* pas de blanc,
* «xml» au début est interdit (maj./min.).

• La balise de fermeture est obligatoire.

XML : les balises (éléments) 2/2

• Le contenu d'un élément peut être

* vide (<toc></toc> ou <toc/>),
* du texte (sauf «<» et «&») basé sur l'encodage,
* un ou plusieurs éléments complets

<toc> ... </toc>

* une répétition de textes et d'éléments,

<article> Le langage <def>XML</def> contient <liste>
<élément> du texte, </élément>
<élément> des éléments, </élément>
</liste></article>

* Les blancs comptent: <a> X est différent de <a>X.
* Les deux systèmes de codage des ruptures de lignes sont pris en charge.

XML : arbre d'éléments

• Un document XML est un et un seul arbre d'éléments. C'est à dire :

* Pas de chevauchement d'éléments. La notation suivante :

```
<list> ... <item> ... </list> ... </item>
```

est invalide. Il faut la corriger comme suit

```
<list> ... <item> ... </item> ... </list>
```

* Un document XML est composé d'un seul élément. La notation suivante :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<article> ... </article>
<article> ... </article>
```

est invalide. Il faut la corriger comme suit

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<stock>
  <article> ... </article>
  <article> ... </article>
</stock>
```

XML : les attributs

• Un élément ouvrant peut être enrichi par des couples de la forme attribut1="valeur1" comme dans l'exemple

```
<produit nom="DVD" prix='200'>
```

• La valeur doit être entourée d'apostrophes si elle contient des guillemets, et inversement.

• Le nom des attributs suit les mêmes règles syntaxiques que les noms d'éléments.

• Attributs comme ci-dessus ou sous-éléments ?

```
<produit>
  <nom>DVD</nom>
  <prix>150</prix>
</produit>
```

• L'attribut doit changer l'interprétation des données:

```
<prix monnaie="Euro"> 150 </prix>
```

XML : les attributs réservés

• xml:lang='langue' permet de définir la langue utilisée dans l'élément et tous les sous-éléments.

La langue suit la norme ISO 3166 définie par la RFC 1766 (Request For Comment). Par exemple fr ou en-US ou fr-FR.
• xml:space='preserve' ou xml:space='default' permet de définir l'interprétation des espaces dans l'élément et tous les sous-éléments.
• xml:id='identificateur' permet d'associer une et une seule clef à un élément .
• xml:idref='identificateur' permet de faire référence à une clef.

```
<section id='intro'>
  <titre>introduction à XML</titre>
  ... </section>

<section>
  <p> après la section
  <xref idref='intro'>d'introduction</xref>
  nous allons passer au plat de résistance...
</section>
```

XML : les références d'entités

• Les entités sont des fragments de document XML définis dans la DTD. La référence d'entité se note :

&nom_de_l_entité;

• Il existe des entités prédéfinies :

* & donne &
* < donne <
* > donne >
* " donne "
* ' donne '
* &#nnn; donne le caractère de code décimal nnn,
* &#xnnn; donne le caractère de code héxadécimal nnn,

• Un exemple :

<texte> en HTML, la balise
<p> est très utile !
© L. Tichit
</texte>

XML : section littérale

• Avec les sections littérales Il est possible de stopper l'interprétation des caractères spéciaux. La syntaxe est la suivante :

```
<![CDATA[ texte non soumi à l'analyse ]]>
```

L'exemple précédent devient

```
<texte><![CDATA[en HTML, la balise
<p> est très utile !]]>
&#169; L. Tichit</texte>
```

XML : espaces de noms 1/2

• Un problème apparaît si on mélange deux textes XML dont les éléments ont le même nom. Par exemple

```
<produit>
   <nom>...</nom>
   <desc>...</desc>
</produit>

<fournisseur> <nom>...</nom>
   <desc> <adr>...</adr>
   <tél>...</tél> </desc>
</fournisseur>
```

• Pour régler ce problème on enrichit le nom de l'élément :

```
<dil:produit
  xmlns:dil='http://www.dil.univ-mrs.fr'>
  <dil:nom>...</dil:nom>
  <dil:desc>...</dil:desc>
</dil:produit>
```

XML : espaces de noms 2/2

• Attention, le préfixe n'est qu'une macro. C'est l'espace de nom qui compte. Les deux éléments suivants sont les mêmes:

```
<dil:produit
  xmlns:dil='http://www.dil.univ-mrs.fr'>
  ... </dil:produit>

<lim:produit
  xmlns:lim='http://www.dil.univ-mrs.fr'>
  ... </lim:produit>
```

• Les espaces de noms doivent être utilisés si le document XML rédigé est destiné à être mélangé à d'autres sources.
• On peut fixer l'espace de noms par défaut avec la syntaxe:

```
<produit xmlns='http://www.dil.univ-mrs.fr'>
  <nom>...</nom> <desc>...</desc> </produit>
</produit>
```

cela évite d'utiliser le préfixe.

XML : style XML 1/2

• Il faut coder les éléments qui structurent le texte au niveau typographique

<para>texte du paragraphe</para>

mais éviter

<para><ligne>...</ligne> <ligne>...</ligne></para>

• Il faut éviter le marquage typographique pour le marquage sémantique. Le code

 l'adresse <tt>www.dil...</tt> appartient
au <it>Dept. d'Informatique</it>

devient

<texte> l'adresse <url>www.dil...</url> appartient
au <def>Dept. d'Informatique</def> </texte>

XML : style XML 2/2

• Éviter les commentaires structurés et préférer les méta-informations. Par exemple, le code

```
<personne> <!-- mise à jour le 10/11/00 -->
  <nom>...</nom>
  ...
```

doit être évité au profit de

```
<personne>
  <comment> mis à jour le 10/11/00 </comment>
  <nom>...</nom>
  ...
```

• Soigner le choix entre attribut et sous-élément.

Attention: seuls les éléments sont exploitables par des feuilles de style CSS. Les navigateurs ne peuvent pas afficher la valeur des attributs.

Validation d’un document XML : DTD (Document Type Definition) 1/3

- Un document XML avec une syntaxe correcte est dit **bien formé**
- C'est la garantie que n'importe quelle application peut lire sans problème un document XML
- On peut vérifier également la conformité d'un document XML par rapport à une structure prédéfinie, on dit alors qu'un document est **valide**
- Une DTD
  - fournit de l'information sur les données d’un document XML
  - permet de déclarer de nouvelles balises et de spécifier des contraintes sur celles-ci
  - permet à une application de savoir quel document XML produire et quoi lire
  - permet de connaître ce qui est supporté (interopérabilité)
  - c’est une grammaire qui décrit les éléments et les attributs acceptés dans un document XML respectant cette DTD

DEFINITIONS DE TYPES DE DOCUMENT 2/3

- Résumé des spécifications :
  - Une DTD (grammaire) permet de déclarer :
    - *un type d'élément,*
    - *une liste d'attribut d'un élément,*
    - *une entité*
  - Chaque balise du langage doit faire l'objet d'une et d'une seule déclaration
- Un document XML est dit "valide" s'il possède une DTD et si sa syntaxe est conforme aux règles de la DTD
  - Un document "valide" est obligatoirement "bien formé"

DEFINITIONS DE TYPES DE DOCUMENT 3/3

- Structure d'une DTD :

```
<!ELEMENT DEVIS (IDENTIFIANT, STATUT, MONTANT, ARTICLE+)>
<!ELEMENT IDENTIFIANT (#PCDATA)>
<!ELEMENT STATUT (#PCDATA)>
<!ELEMENT MONTANT (#PCDATA )>
<!ATTLIST MONTANT devise CDATA #REQUIRED>
<!ELEMENT ARTICLE (REFERENCE, QUANTITE, PRIX_UNITAIRE, IMAGE+)>
<!ELEMENT REFERENCE (#PCDATA)>
<!ELEMENT QUANTITE (#PCDATA)>
<!ELEMENT PRIX_UNITAIRE (#PCDATA)>
<!ATTLIST PRIX_UNITAIRE devise CDATA #REQUIRED>
<!ELEMENT IMAGE EMPTY>
<!ATTLIST IMAGE href CDATA #REQUIRED>
```

Déclaration d'un élément dont le contenu est une suite d'éléments

Déclaration d'un élément dont le contenu est une suite de caractères

Déclaration d'un élément vide

Déclaration de liste d'attributs

DTD : définition d'éléments

- Définition d'éléments : <!ELEMENT SELECT (OPTGROUP|OPTION)+ >

elt?	elt est optionel
elt+	elt apparaît au moins une fois
elt*	elt apparaît entre 0 et n fois
elt1 | elt2	elt1 ou elt2
elt1, elt2	elt2 suit elt1
(elt1,elt2)+	elt1 suivi de elt2 apparaît au moins une fois
#PCDATA	données de type texte dans l'encodage courant *parsable character data*
ANY	n'importe quoi
EMPTY	vide

DTD : définition d'attributs 1/2

- Définition d'attributs : *<!ATTLIST elt definition>*

  definition → donnée par un nom, un type (éventuel) et une valeur par défaut

- Nature des attributs : optionnels, obligatoires, valeur déterminée
  - optionnel sans valeur par défaut

    <!ATTLIST personne att1 CDATA #IMPLIED>
  - optionnel avec valeur par défaut

    <!ATTLIST personne att1 "bidule">
  - obligatoire

    <!ATTLIST personne att1 CDATA #REQUIRED>
  - fixe

    <!ATTLIST personne att1 CDATA #FIXED "bidule">

- Exemple :

  <!ATTLIST personne id ID #REQUIRED>

  <!ATTLIST personne att1 CDATA #IMPLIED att2 CDATA #IMPLIED>

DTD : définition d'attributs 2/2

- Types d'attributs :
  - données caractères : CDATA
  - enumération : (oui | non | peut-être)
  - ID : identifiant pour l'élément (doit être unique dans le document)
  - IDREF, IDREFS : référence à un ID de ce document (resp. plusieurs références séparées par des espaces)
  - ENTITY, ENTITIES : la valeur de l'attribut doit être le nom d'une entité déclarée dans la DTD (resp. un ensemble d'entités séparées par des espaces)

XML & DTD : exemple 1 1/2

- Le fichier essai.xml :

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no" ?>
<!DOCTYPE stock SYSTEM "essai.dtd">
<stock>
  <produit>
    <nom> Livre </nom>
    <prix monnaie="Francs"> 50 </prix>
    <comment> Un article très recherché </comment>
  </produit>
  <produit>
    <nom> CD </nom><prix monnaie="Euros"> 23 </prix>
  </produit>
</stock>
```

XML & DTD : exemple 1 2/2

- Le fichier essai.dtd (Document Type Definition):

```
<!ELEMENT stock    (produit+)>
<!ELEMENT produit  (nom,prix,comment?)>
<!ELEMENT nom      (#PCDATA)>
<!ELEMENT prix     (#PCDATA)>
<!ATTLIST prix monnaie (Euros|Francs) #IMPLIED>
<!ELEMENT comment (#PCDATA)>
```

DTD : définition d'entités

- Définition d'entités : pour d'associer un nom à un contenu (alias)

```
<!ENTITY BBSG "Master de Bioinformatique, Biochimie ...">
<!ENTITY Logo SYSTEM "/usr/images/logo.png">
```

Référence :

```
étudiant en &BBSG;
<img src="&Logo;">
```

- Entités paramétrées :

```
<!ENTITY % message "#PCDATA | article">
<!ENTITY % statut "statut (public|privé) 'public'">
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
```

Référence :

```
<!ELEMENT elt1 (%message;)*>
<!ELEMENT article  (#PCDATA)>
<!ATTLIST  article date CDATA #IMPLIED>
<!ATTLIST  article &statut;>
<!ELEMENT entete (%heading;)>
```

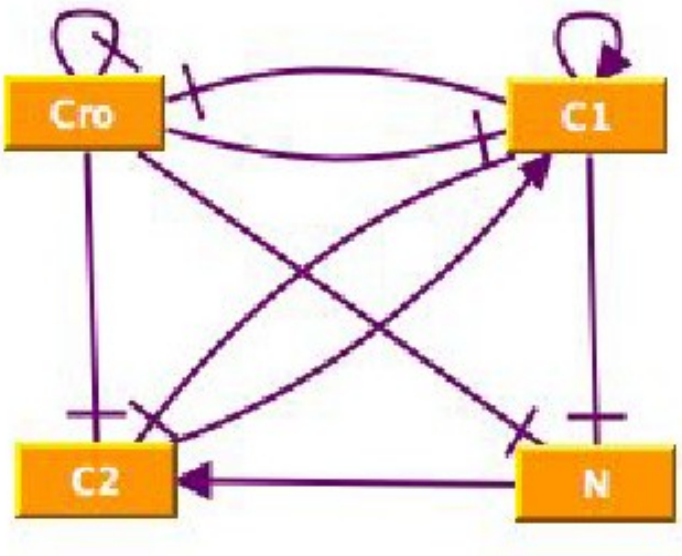
XML & DTD : exemple 2

```
<!-- fichier doc1.dtd -->
<!ELEMENT liste_profs (prof)*>
<!ELEMENT prof (nom,mail)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT mail (#PCDATA)>
<!ENTITY adr "crfb.univ-mrs.fr">
<!ENTITY moi "tichit">
<!ENTITY prof-dept-info SYSTEM "prof-dept-info.xml">
<!ENTITY prof-dept-bio SYSTEM "prof-dept-bio.xml">
```

```
<!-- fichier fichier1.xml -->
<?xml version="1.0"encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE liste_profs SYSTEM "doc1.dtd">
<liste_profs>
&prof-dept-info;  &prof-dept-bio;  contact :<mail>&moi;@&adr;</mail>
</liste_profs>
```

exemple 3 : GINML

- Une DTD définit la grammaire des fichiers contenant les modèles de graphes de régulation ou les graphes de transitions d'états produits par GINsim.




```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "http://gin.univ-mrs.fr/GINsim/GINML_2_1.dtd">
<gxl xmlns:xlink="http://www.w3.org/1999/xlink">
    <graph id="phage4" class="regulatory" nodeorder="C1 Cro C2 N">
        <node id="N" basevalue="1" maxvalue="1">
        </node>
        <node id="Cro" basevalue="3" maxvalue="3">
                <parameter idActiveInteractions="Cro_Cro_0" val="2"/>
        </node>
        <node id="C1" basevalue="2" maxvalue="2">
                <parameter idActiveInteractions="C2_C1_0" val="2"/>
                <parameter idActiveInteractions="C1_C1_0" val="2"/>
                <parameter idActiveInteractions="C1_C1_0 C2_C1_0" val=
        </node>
        <node id="C2" basevalue="0" maxvalue="1">
                <parameter idActiveInteractions="N_C2_0" val="1"/>
        </node>
        <edge id="N_C2_0" from="N" to="C2" minvalue="1" maxvalue="1" sign="positive">
        </edge>
< edge id="Cro_Cro_0" from="Cro" to="Cro" minvalue="3" maxvalue="3" sign="negative">
< /edge>
< edge id="C1_Cro_0" from="C1" to="Cro" minvalue="2" maxvalue="2" sign="negative">
< /edge>
< edge id="C1_C1_0" from="C1" to="C1" minvalue="2" maxvalue="2" sign="positive">
< /edge>
        <edge id="C2_C1_0" from="C2" to="C1" minvalue="1" maxvalue="1" sign="positive">
        </edge>
    </graph>
</gxl>
```

INCONVENIENT DES DTD

- Inconvénients des DTD :
  - Une DTD est non extensible (ce n’est pas un document XML).
  - Une DTD ne permet pas de typer les données
  - Une DTD ne peut prendre en compte qu’un seul espace de nom (namespace).

XML-SCHEMA 1/3

- En réponse aux lacunes des DTD, une alternative a été proposée comme recommandation : il s'agit de XML-Schema
- Cette nouvelle norme achève de faire d'XML un format pivot…
- La version 1.1 de XML Schema (datée de mai 2001) se compose de 3 normes :
  - XML Schema tome 0 : Introduction
  - XML Schema tome 1 : Structures
  - XML Schema tome 2 : Types de données

XML-SCHEMA 2/3

- Les documents XML-Schema sont des documents :
  - respectant la syntaxe XML,
  - permettant de décrire la structure d'un document XML d'une façon beaucoup plus complète que les DTD.
- XML-Schema permet en effet de :
  - spécifier la typologie des données que va contenir le document XML décrit par le XML-Schema,
  - gérer une quarantaine de types de données simples,
  - gérer des types complexes,
  - gérer les occurrences des données.

XML-SCHEMA 3/3

- Exemple de document XML-Schema :

Le document XML :

```
<entree>
  <nom>Harry Cover</nom>
  <telephone>0102030405</telephone>
</entree>
```

Le document XML-Schema correspondant :

```
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="entree">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nom" type="xsd:string"
                   minoccurs="1" maxoccurs="1"/>
      <xsd:element name="telephone" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Entete

Définition de la balise complexe *entree*

Définition de la balise String *nom*

Définition de la balise de type Décimal *telephone*

Constituants d'un XML-SCHEMA 1/8

- Déclaration de l'entête :
  - L'élément <xsd:schema> permet de déclarer un document XML-Xchema.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
            targetNamespace="http://www.annuaire.org"
            xmlns="http://www.annuaire.org"
            elementFormDefault="qualified"/>
```

L'attribut targetNamespace permet de préciser l'espace de nommage de ce type de documents.

L'attribut elementFormDefault précise si les documents XML respectant cette grammaire doivent référer à cet espace de nommage.

Constituants d'un XML-SCHEMA 2/8

- Déclaration des types de données :
  - Il est possible de déclarer un type de données
    - *soit dans la déclaration d'un élément (local)*
    - *soit hors de la déclaration de l'élément (global)*
  - XML-Schema permet d'utiliser des données :
    - *de type prédéfini (string, int…)*
    - *de type complexe*
    - *dont le type est une restriction de type*
    - *dont le type est une extension de type*

Constituants d'un XML-SCHEMA 3/8

- Déclaration des types de données :
  - **Types prédéfinis** :
    - *byte, unsignedByte, hexBinary, integer, positiveInteger, negativeInteger, int, unsignedInt, long, unsignedLong, short, unsignedShort, decimal, float, double...*
    - *string, NormalizedString, token*
    - *boolean, anyURI, language*
    - *time, dateTime, duration, date, gMonth, gYear, gYearMonth, gDay, gMonthDay*
    - *ID, IDREF, IDREFS, ENTITY, ENTITIES, NOTATIN, NMTOKEN, NMTOKENS*

  Exemple : <xsd:element name="comment" type="xsd:string"/>

Constituants d'un XML-SCHEMA 4/8

- Déclaration des types de données :
  - **Types complexes** :

    Exemple : le type de données *TypeAdresse* se compose de 6 éléments *Numero, Rue1, Rue2, Ville, CP et Pays* :

```
<xsd:complexType name="TypeAdresse">
  <xsd:sequence>
   <xsd:element name="Numero"  type="xsd:positiveInteger"/>
   <xsd:element name="Rue1"    type="xsd:string"/>
   <xsd:element name="Rue2"    type="xsd:string"/>
   <xsd:element name="Ville"   type="xsd:string"/>
   <xsd:element name="CP"      type="xsd:decimal"/>
   <xsd:element name="Pays"    type="xsd: NMTOKEN" fixed="France"/>
</xsd:sequence>
<xsd:element name="adresse" type="TypeAdresse"/>
```

Constituants d'un XML-SCHEMA 5/8

- Déclaration des types de données :
  - **Restriction de type existant** :

    Exemple : le type de données string comprend 6 attributs optionnels : pattern, enumeration, length, minlength, maxlength, whitespace. Si on désire représenter un choix Oui/Non (restriction sur l'attribut enumeration) :

```
<xsd:simpleType name="choixOuiNon">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="oui"/>
    <xsd:enumeration value="non"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="choix" type="choixOuiNon"/>
```

Constituants d'un XML-SCHEMA 6/8

- Déclaration des types de données :
  - **Extension / dérivation de type existant** :

    Exemple : si l'on souhaite créer un type Personne contenant en plus du nom et du prénom, un élément de type Adresse (extension du type Adresse vu précédemment) :

```
<xsd:complexType name="Personne">
  <xsd:complexContent>
    <xsd:extension base="adresse">
      <xsd:sequence>
        <xsd:element name="Nom" type="xsd:string"/>
        <xsd:element name="Prenom" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Constituants d'un XML-SCHEMA 7/8

- Déclaration des éléments :

1. Définition d'un élément dont le type est déjà déclaré :

```
<xsd:complexType name="entree">
   ...
</xsd:complexType>
<xsd:element name="entree" type="entree">
```

2. Définition d'un élément contenant la définition du type :

```
<xsd:element name="entree">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nom" type="xsd:string"/>
      <xsd:element name="telephone" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Constituants d'un XML-SCHEMA 8/8

- Déclaration des attributs :

Exemple de l'élément montant dans un document XML :

<montant devise="EURO" horsTaxe="true"/>.

L'attribut devise est optionnel et a comme valeur par défaut «EURO».
L'attribut horsTaxe est obligatoire et a comme valeur par défaut «true».

Modélisation en XML-Schema :

```
<xsd:element name="element">
  <xsd:complexType>
    <xsd:attribute name="devise" type="xsd:string"
                   use="implied" value="EURO"/>
    <xsd:attribute name="horsTaxe" type="xsd:boolean"
                   use="required" value="true"/>
  </xsd:complexType>
</xsd:element>
```

Conclusion : DTD versus XML-SCHEMA

- La DTD permet de définir facilement et rapidement des grammaires simples.
- XML-Schema permet de définir de manière plus formelle et complète une grammaire mais c'est au prix d'une complexité accrue.
- Un document XML-Schema respecte la syntaxe XML.
- Un document XML-Schema est généralement plus volumineux et plus difficile à lire qu'une DTD (pour un opérateur humain).