

Le langage XML Schema

<http://www.w3.org/TR/xmlschema-0/>

<http://www.w3.org/TR/xmlschema-1/>

<http://www.w3.org/TR/xmlschema-2/>



Introduction (1)

- Un schéma écrit en **XML Schema** définit une classe de documents
- Les **XML Schémas** améliore les **DTD** :
 - Un XML schéma est un fichier XML :
 - pas un nouveau langage, pas de parseur dédié.
 - Si on sait lire un fichier XML, on peut lire un schéma
 - Propose des types de données utiles (en plus des chaîne de caractères)
 - Booléen, entier, décimal, chaîne, date,...

Introduction (2)



- Un XML schema est composé :
 - **D'un prologue :**
 - indique quels espaces de noms vont être utilisés
 - Espace de noms : pointeur vers un document (pré-) définissant un vocabulaire qui pourra être utilisé dans le document courant
 - **D'un corps :**
 - Liste d'assertions d'éléments, de types simples ou complexes,...

Schémas : Schema, prologue

Syntaxe



- `<?xml version="1.0" encoding="ISO-8859-1"?>`
 - Première assertion d'un document XML
- `<xs:schema`
 - `xmlns:xs="http://www.w3.org/2001/XMLSchema"`
 - `targetNamespace="http://www.lri.fr/~thiam/FCXML"`
 - `xmlns="http://www.lri.fr/~thiam/FCXML"`
 - `elementFormDefault="qualified" version="1.0">`
- Définition de l'espace de nom **xs** qui permet d'utiliser le vocabulaire prédéfini pour XML schema
 - Le vocabulaire XML schema devra être préfixé par **xs** !

Schémas : Schema, prologue

Détails



- **targetNamespace** indique à quel espace de noms appartiennent les éléments définis dans le schéma.
- **xmlns="..."** indique l'espace de noms par défaut
- **elementFormDefault** indique si les éléments doivent
 - être préfixés : **qualified**
 - non préfixés : **unqualified**par leur espace de noms

Schémas : Schema, corps



- Le corps permet de définir des éléments :
 - `<xs:element name="library" />`
 - Un élément peut avoir plusieurs attributs optionnels :
 - **Type="..."** : pour définir des types **simples** ou **complexes**.
 - Des restrictions de cardinalités généralisant celles des DTD :
 - *minOccurs*="x" ou *maxOccurs*="x" avec $x \in \mathbb{N}^* \cup \{\text{unbounded}\}$



Type simple et type complexe

- Un **type complexe** est un type qui peut contenir des **déclarations** d'attributs et d'éléments.
- Un **type simple** ne peut pas contenir de déclarations d'attributs ou d'éléments.



TYPES SIMPLES

Définition



- Un type simple est défini par :
 - un **ensemble de valeurs**,
 - une **représentation lexicale**,
 - un ensemble de **facettes** qui caractérise l'ensemble de valeurs.
- Un type simple peut être :
 - un **type atomique**, un **type liste** ou un **type union**,
 - primitif ou dérivé,
 - prédéfini ou défini par l'utilisateur.

Type atomique, type liste ou type union

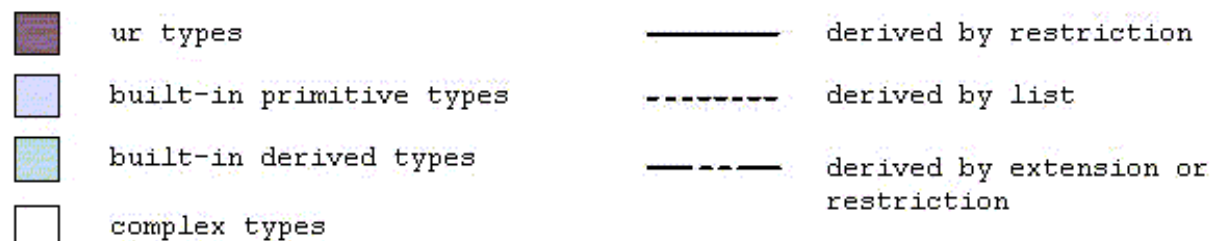
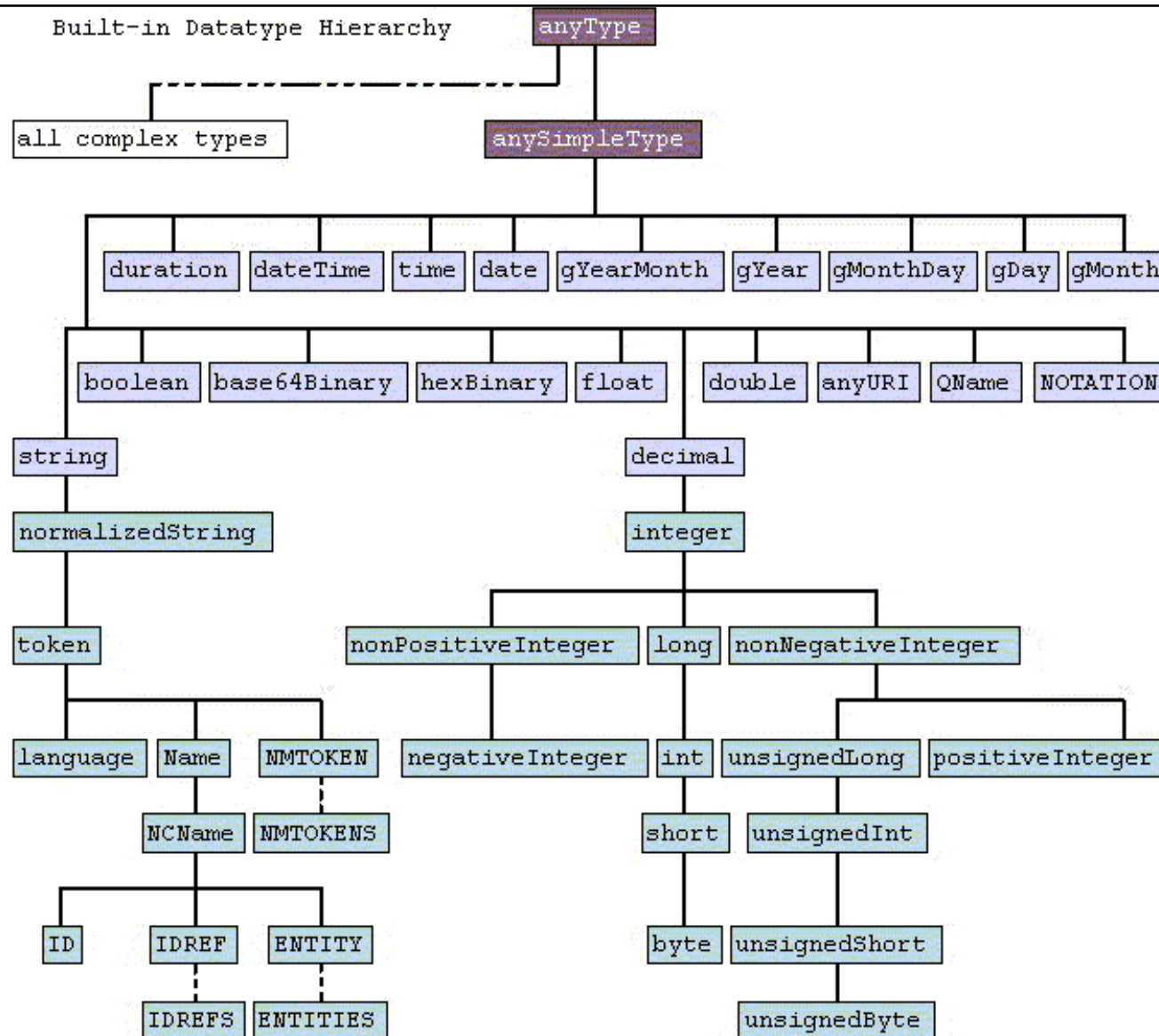


- Un type atomique est un ensemble de valeurs indécomposables dites **valeurs atomiques**.
- Un type liste est un ensemble de séquences de longueur finie de valeurs atomiques.
- Un type union est un ensemble de valeurs atomiques appartenant à plusieurs types atomiques.

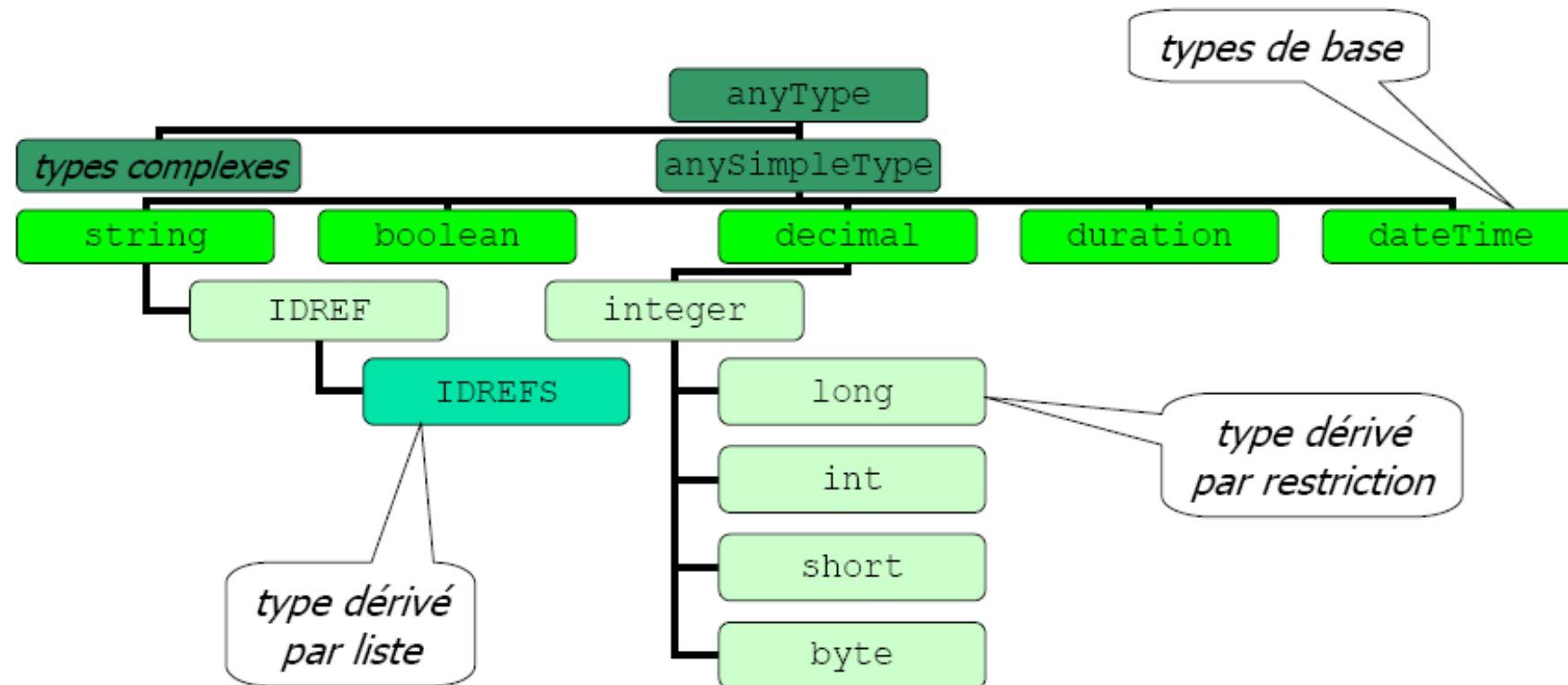
Types prédéfinis ou définis par l'utilisateur



- Un type **prédéfini** peut être primitif ou dérivé.
- Un type peut être **défini** par dérivation de types existants.
- Un type défini peut être **anonyme** ou **nommé**.



Hiérarchie des types prédéfinis : ZOOM



Facettes des types de base :

Fondamentales



<u>Datatype</u>	ordered	bounded	cardinality	numeric
<u>string</u>			X	
<u>boolean</u>			X	
<u>float</u>	X	X	X	X
<u>double</u>	X	X	X	X
<u>decimal</u>	X			X
<u>duration</u>	X		X	
<u>dateTime</u>	X		X	

Facettes des types de base : contraignantes ou non-fondamentales



	string	boolean	decimal	float	duration	dateTime
length	x				x	x
minLength	x					
maxLength	x					
pattern	x	x	x	x	x	x
enumeration	x		x	x	x	x
whiteSpace	x	x	x	x	x	x
minInclusive			x	x	x	x
minExclusive			x	x	x	x
maxInclusive			x	x	x	x
maxExclusives			x	x	x	x
totalDigits			x			
fractionDigits			x			

Types prédéfinis temporels



duration	1 an, 2 mois, 3 jours, 10 heures, 30 minutes : P1Y2M3DT10H30M
dateTime	13h20, le 31 mai 1999 (UTC) : 1999-05-31T13:20 13h20, le 31 mai 1999 (UTC + 5h) : 1999-05-31T13:20:00-05:00
time	13h20 du jour courant (UTC) : 13:20
date	31 mai 1999 : 1999-05-31
gYear	1999 : 1999
gYearMonth	mai 1999 : 1999-05
gMonth	mois de mai de l'année courante : --05--
gMonthDay	31 mai de l'année courante : --05-31
gDay	31 du mois courant : --31

Types de base :

string, boolean, float et decimal



string	Ensemble des séquences de caractères de longueur finie	
boolean	Ensemble de valeurs = {true, false, 1, 0}	
float	Ensemble de valeurs = $m \times 2^e$ où $ m \leq 2^{24}$ et $-149 \leq e \leq 104$ + zéro positif et négatif infinité positive et négative pas un nombre	-1E4 1267.43233E12 12.78e-2 12 0 -0 INF -INF NaN
decimal	Nombre décimaux de précision arbitraire. Ensemble de valeurs = $i \times 10^n$ où i et n sont des entiers et $n \geq 0$	-1.23 12678967.543233 +100000.00 210

Types prédéfinis dérivés de decimal: un extrait



integer	Dérivé du type decimal par fractionDigits = 0	-1 0 12678967543233 +100000
long	Dérivé du type integer par minInclusive = -9223372036854775808 et maxInclusive = 9223372036854775807	-1 0 12678967543233 +100000
int	Dérivé du type integer par minInclusive = -2147483648 et maxInclusive = 2147483647	-1 0 126789675 +100000
short	Dérivé du type int par minInclusive = -32768 et maxInclusive = 32767	-1 0 12678 +10000
byte	Dérivé du type short par minInclusive = -128 et maxInclusive = 127	-1 0 126 +100

Types dérivés par **restriction**



- Intervalle
- Énumération
- motif



Intervalle : Exemple

- Ensemble des altitudes des points de la Terre (le plus haut sommet a une altitude de 8850m).

```
<xsd:simpleType name="altitude">  
  <xsd:restriction base="xsd:integer">  
    <xsd:minInclusive value="0"/>  
    <xsd:maxInclusive value="8850"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Enumération : Exemple 1



- Liste des noms des pays africains :

```
<xsd:simpleType name="nom-pays-af">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Algérie"/>
    <xsd:enumeration value="Sénégal"/>
    <xsd:enumeration value="Congo Brazzaville"/>
    <xsd:enumeration value="Gambie"/>
    <xsd:enumeration value="Mauritanie"/>
    ...
  </xsd:restriction>
</xsd:simpleType>
```



Enumération : Exemple 2

- Liste des codes des pays :

```
<xsd:simpleType name="code-pays-af">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AL"/>
    <xsd:enumeration value="SN"/>
    <xsd:enumeration value="CGB"/>
    <xsd:enumeration value="GB"/>
    <xsd:enumeration value="MAU"/>
    ...
  </xsd:restriction>
</xsd:simpleType>
```

Motif : Exemple 1



- Différents motifs de ISBN à 10 chiffres

```
<xsd:simpleType name="ISBN">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="\d-\d{2}-\d{6}-[\dX]" />  
    <xsd:pattern value="\d-\d{3}-\d{5}-[\dX]" />  
    <xsd:pattern value="\d-\d{4}-\d{4}-[\dX]" />  
    <xsd:pattern value="\d-\d{5}-\d{3}-[\dX]" />  
  </xsd:restriction>  
</xsd:simpleType>
```



Motif : Exemple 2

- Différentes formes d'un nom XML :

```
<xsd:simpleType name="Identifiant">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[:_A-Za-z][-._:_0-9A-Za-z]*"/>  
  </xsd:restriction>  
</xsd:simpleType>
```




Types liste

- L'opérateur **xsd:list** définit un nouveau type simple
- valeurs sont les listes de valeurs du type simple donné par l'attribut **itemType**.
- pas de listes générales (# langages de programmation)
- Uniquement listes de valeurs séparées par des espaces
- Souvent utilisées comme valeurs d'attributs.
- Pas de caractères d'espacement dans type simple donné par **itemType**

Exemple 1



- Liste de pays :

```
<xsd:simpleType name="liste-pays">  
  <xsd:list itemType="nom-pays-af"/>  
</xsd:simpleType>
```

- Contenu conforme à ce type :

```
<pays-africains>  
  Sénégal Gambie Mauritanie  
</pays-africains>
```

Exemple 2



- Type pour liste d'entiers

```
<xsd:simpleType name="IntList">  
  <xsd:list itemType="xsd:integer"/>  
</xsd:simpleType>
```

- Type pour liste de 5 entiers

```
<xsd:simpleType name="IntList5">  
  <xsd:restriction base="IntList">  
    <xsd:length value="5"/>  
  </xsd:restriction>  
</xsd:simpleType>
```



Types union

- L'opérateur **xsd:union** définit un nouveau type simple dont les valeurs sont celles des types listés dans l'attribut **memberTypes**.

Exemple 1



- Ensemble des noms ou des codes des pays :

```
<xsd:simpleType name="nom-ou-code-pays">  
  <xsd:union memberTypes="nom-pays-af code-pays-af"/>  
</xsd:simpleType>
```

- Éléments dont le contenu est conforme à ce type :

```
<pays>AL</pays>  
<pays>Algerie</pays>
```

Exemple 2



- Définition du type **Unbounded**

```
<xsd:simpleType name="Unbounded">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="unbounded"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

- Définition du type **IntegerOrUnbounded**

```
<xsd:simpleType name="IntegerOrUnbounded">  
  <xsd:union memberTypes="xsd:nonNegativeInteger  
                        Unbounded"/>  
</xsd:simpleType>
```



TYPES COMPLEXES

Type complexe



- Un type complexe définit des compositions d'éléments et/ou d'attributs :

```
<xsd:complexType ...>
```

Suite de déclarations d'attribut, d'élément ou de composition d'éléments

```
</xsd:complexType>
```
- Une composition d'éléments est réalisée à l'aide de 3 constructeurs:
 - **sequence** : séquence d'éléments typés,
 - **choice** : un élément parmi une liste d'éléments possibles,
 - **all** : tas d'éléments typés.



Déclaration d'un élément

- Un élément dans la définition d'un type complexe est spécifié sous la la forme d'un élément ***xsd:element*** dont les attributs peuvent être :
 - ***name*** : nom de l'élément,
 - ***type*** : type de l'élément,
 - ***ref*** : référence à un nom d'élément déclaré au niveau global (**réutilisation**),
 - ***minOccurs*** : nombre minimum d'occurrences de l'élément (1 par défaut),
 - ***maxOccurs*** : nombre maximum d'occurrences de l'élément (1 par défaut),
 - ...
 - le contenu est la définition d'un type.



Déclaration d'un attribut

- Un attribut dans la définition d'un type complexe est spécifié sous la forme d'un élément **xsd:attribute** dont les attributs peuvent être :
 - *name* : nom de l'attribut,
 - *ref* : référence à un nom d'attribut déclaré au niveau global (**réutilisation**),
 - *type* : type des valeurs de l'attribut,
 - *use* : required, optional (par défaut), ...
 - *default* : valeur par défaut,
 - *fixed* : valeur fixée,
 - ...
 - le contenu peut être : un type simple.

Éléments à contenu **simple**



Élément à contenu simple : exemple



- L'élément :

```
<pays>Gambie</pays>
```

- peut être déclaré :

```
<xsd:element name="pays" type="mns:nom-pays-af"/>
```

Séquence d'éléments : type anonyme



- L'élément :

```
<livre>  
    <titre>Programmer en XML</titre>  
    <année>2004</année>  
</livre>
```
- peut être déclaré :

```
<xsd:element name="livre">  
    <xsd:complexType>  
        <xsd:sequence>  
            <xsd:element name="titre" type="xsd:string"/>  
            <xsd:element name="année" type="xsd:gYear"/>  
        </xsd:sequence>  
    </xsd:complexType>  
</xsd:element>
```

Eléments et attributs : type nommé



- L'élément :

```
<livre edition="2">  
    <titre>Programmer en XML</titre>  
    <année>2004</année>  
</livre>
```
- peut être déclaré :

```
<xsd:element name="livre">  
    <xsd:complexType name="livre">  
        <xsd:sequence>  
            <xsd:element name="titre" type="xsd:string"/>  
            <xsd:element name="année" type="xsd:gYear"/>  
        </xsd:sequence>  
        <xsd:attribute name="édition" type="xsd:positiveInteger"/>  
    </xsd:complexType>  
</xsd:element>
```



Éléments à contenu **mixte**

- *pur* lorsqu'il ne contient que des éléments qui, eux-mêmes, peuvent à leur tour contenir du texte et/ou des éléments.
- *mixte* lorsqu'il contient du texte autre que des caractères d'espacement en dehors de ses enfants.
- attribut *mixed* de l'élément **xsd:complexType** contient la valeur true.

Exemple



- L'élément :

```
<titre>
```

La logique <sigle>FOL</sigle> est peu expressive

```
</titre>
```

- peut être déclaré :

```
<xsd:element name="titre">
```

```
  <xsd:complexType mixed="true">
```

```
    <xsd:sequence>
```

```
      <xsd:element name="sigle" type="xsd:string"/>
```

```
    </xsd:sequence>
```

```
  </xsd:complexType>
```

```
</xsd:element>
```


Elément à contenu vide



- L'élément :

```
<prix monnaie="euros" valeur="25.5"/>
```

- peut être déclaré :

```
<xsd:element name="prix">  
  <xsd:complexType>  
    <xsd:attribute name="monnaie" type="xsd:string"/>  
    <xsd:attribute name="valeur" type="xsd:decimal"/>  
  </xsd:complexType>  
</xsd:element>
```

Type nommé



- On peut nommer le type définissant un prix en euros ou en dollars :

```
<xsd:complexType name="prix-international">  
  <xsd:attribute name="monnaie" type="xsd:string"/>  
  <xsd:attribute name="valeur" type="xsd:decimal"/>  
</xsd:complexType>
```

- et y faire référence dans la déclaration de l'élément prix :

```
<xsd:element name="prix" type="prix-international"/>
```

Élément à contenu alternatif



- Les éléments :

```
<prix><euros>25.5</euros></prix>
```

```
<prix><dollars>28.75</dollars></prix>
```

- peuvent être déclarés :

```
<xsd:element name="prix">
```

```
  <xsd:complexType>
```

```
    <xsd:choice>
```

```
      <xsd:element name="euros" type="xsd:decimal"/>
```

```
      <xsd:element name="dollars" type="xsd:decimal"/>
```

```
    </xsd:choice>
```

```
  </xsd:complexType>
```

```
</xsd:element>
```

Opérateur d'ensemble : Élément contenu non ordonné



- Les éléments doivent apparaître une fois dans un ordre quelconque
- Pas d'équivalent en DTD

```
<xsd:element name="book">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="title"          type="xsd:string"/>
      <xsd:element name="author"         type="xsd:string"/>
      <xsd:element name="year"           type="xsd:string"/>
      <xsd:element name="publisher" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

Exemple



```
<?xml version="1.0" encoding="iso-8859-1"?>
<bibliography>
  <book key="Michard01" lang="fr">
    <author>Alain Michard</author>
    <title>XML langage et applications</title>
    <publisher>Eyrolles</publisher>
    <year>2001</year>
  </book>
  <book key="Zeldman03" lang="en">
    <title>Designing with web standards</title>
    <author>Jeffrey Zeldman</author>
    <year>2003</year>
    <publisher>New Riders</publisher>
  </book>
  ...
</bibliography>
```

Groupe d'éléments



- La séquence d'éléments :

```
<prenom>Jean</prenom><nom>Dupont</nom>
```

- peut être déclarée :

```
<xsd:group name="prenom-nom">  
  <xsd:sequence>  
    <xsd:element name="prenom" type="xsd:string"/>  
    <xsd:element name="nom" type="xsd:string"/>  
  </xsd:sequence>  
</xsd:group>
```

- (On peut aussi grouper des attributs)

Utilisation des groupes d'éléments



- Pour déclarer des éléments décrivant des personnes décrites par leur nom et leurs prénoms, on peut définir le type suivant :

```
<xsd:complexType name="personne">  
  <xsd:group ref="prenom-nom"/>  
</xsd:complexType>
```

Dérivation d'un type complexe par extension



- A partir du type personne, on peut définir le type chercheur en ajoutant au prénom et au nom, le laboratoire et le pays :

```
<xsd:complexType name="chercheur">
  <xsd:complexContent>
    <xsd:extension base="personne">
      <xsd:sequence>
        <xsd:element name="laboratoire" type="xsd:string"/>
        <xsd:element ref="pays">
      </xsd:sequence>
    </xsd:extension>
  </complexContent>
</xsd:complexType>
```


Schéma XML



- Un schéma XML est une suite de définitions de types et de déclarations d'attributs.



Exemple du document ...

```
<livres> ...  
  <livre edition="1">  
    <titre>Le langage XML</titre>  
    <auteur>  
      <prenom>Jean</prenom><nom>Dupont</nom>  
      <laboratoire>LRI</laboratoire><pays>France</pays>  
    </auteur>  
    <auteur>  
      <prenom>Pierre</prenom><nom>Durand</nom>  
      <laboratoire>LRI</laboratoire><pays>France</pays>  
    </auteur>  
    <année>2004</année>  
    <prix monnaie="euros" valeur="25.5"/>  
  </livre>...  
</livres>
```

DTD



```
<!ELEMENT livres (livre*)>
<!ELEMENT livre (titre, auteur+, année, prix)>
<!ATTLIST livre edition CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (prenom, nom, laboratoire, pays)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT laboratoire (#PCDATA)>
<!ELEMENT année (#PCDATA)>
<!ELEMENT prix EMPTY>
<!ATTLIST prix monnaie CDATA #REQUIRED valeur CDATA #REQUIRED>
<!ELEMENT pays (#PCDATA)>
```

Schéma XML : exemple



```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:group name="prenom-nom">
    <xsd:sequence>
      <xsd:element name="prenom" type="xsd:string"/>
      <xsd:element name="nom" type="xsd:string"/>
    </xsd:sequence>
  </xsd:group>
  <xsd:complexType name="personne">
    <xsd:group ref="prenom-nom"/>
  </xsd:complexType>
  ...
```

Schéma XML : exemple



```
...  
<xsd:simpleType name="nom-pays">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="Algerie"/>  
    <xsd:enumeration value="Australie"/>  
    <xsd:enumeration value="Belgique"/>  
    <xsd:enumeration value="France"/>  
    ...  
  </xsd:restriction>  
</xsd:simpleType>  
  
<xsd:element name="pays" type="nom-pays"/>  
...
```

Schéma XML : exemple



```
<xsd:complexType name="chercheur">
  <xsd:complexContent>
    <xsd:extension base="personne">
      <xsd:sequence>
        <xsd:element name="laboratoire" type="xsd:string"/>
        <xsd:element ref="pays"/>
      </xsd:sequence>
    </xsd:extension>
  </complexContent>
</xsd:complexType>

<xsd:complexType name="prix-international">
  <xsd:attribute name="monnaie" type="xsd:string"/>
  <xsd:attribute name="valeur" type="xsd:decimal"/>
</xsd:complexType> ...
```

Schéma XML : exemple



```
...
<xsd:element name="livre">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="titre" type="xsd:string"/>
      <xsd:element name="auteur" type="chercheur"
        maxOccurs="unbounded"/>
      <xsd:element name="année" type="xsd:gYear"/>
      <xsd:element name="prix"
        type="prix-international"/>
    </xsd:sequence>
    <xsd:attribute name="edition" type="xsd:positiveInteger"/>
  </xsd:complexType>
</xsd:element>
...
```

Schéma XML : exemple



```
...  
<xsd:element name="livres">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element ref="livre" maxOccurs="unbounded"/>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>  
</xsd:schema>
```


END

