XSchema : les schémas XML

Espace de noms : `http://www.w3.org/2001/XMLSchema`
Préfixe usuel : `xs`

Recommandations

- XML schema part 0 : Primer 02/05/01
- XML schema part 1 : Structures 02/05/01
- XML schema part 2 : Datatypes 02/05/01

Schémas pour données semi structurées

- Pourquoi un schéma ?
  - partage et échange de données
  - optimisation des traitements sur les données
- Schéma
  - description des contraintes de structure
- Langage de schémas
  - méta formalisme pour décrire les schémas

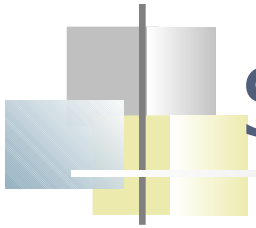
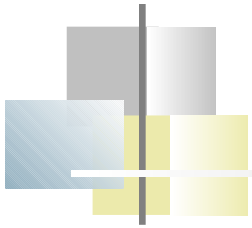


Schéma XML

- Un schéma définit
  - comme une DTD une classe de documents
    - Les éléments
    - Leurs attributs
    - La structure du document
  - contrairement à une DTD
    - des types simples et complexes de données
    - des contraintes sur les valeurs
    - Et bien d’autres choses…réutilisation, extensibilité
- Une DTD a une syntaxe propriétaire
- Un schéma est spécifié en XML

Exemple : Document bien formé

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<listeCD>
  <cd>
    <nom>100 Masterpieces, vol. 5</nom>
    <piste>
      <num>1</num>
      <titre>Fidelio, ouverture</titre>
      <interprete>Budapest Symphony Orchestra</interprete>
      <compositeur>Beethoven</compositeur>
      <duree>6:20</duree>
    </piste>
    <piste>
      ...>
    </piste>
  </cd>
  <cd>  ... </cd>
</listeCD>
```



Exemple : DTD

```
<!ELEMENT listeCD (cd)*>
<!ELEMENT cd (nom, (piste)*)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT piste (num, titre, interprete+, compositeur, duree)>
<!ELEMENT num (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT interprete (#PCDATA)>
<!ELEMENT compositeur (#PCDATA)>
<!ELEMENT duree (#PCDATA)>
```

<!ELEMENT listeCD (cd)*>

Le schéma de l'exemple

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element xs:name="LISTECD" >
  <xs:complexType>
   <xs:sequence>
     <xs:element xs:name="CD"
                          xs:minOccurs="0" xs:maxOccurs="unbounded" >
       <xs:complexType>
        ...
       </xs:complexType>
     </xs:element>
   </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

```
<!ELEMENT cd (nom, (piste)*)>
<!ELEMENT piste (num, titre, interprete+, compositeur, duree)
```

Exemple suite (le modèle de contenu de cd)

```
<xs:element xs:name="NOM" xs:type="xs:string" xs:minOccurs="1" />
<xs:element xs:name="PISTE" xs:minOccurs="0" xs:maxOccurs="unbounded" >
  <xs:complexType>
   <xs:sequence>
      <xs:element xs:name="NUM" xs:type="xs:positiveInteger" xs:minOccurs="1" />
      <xs:element xs:name="TITRE" xs:type="xs:string" xs:minOccurs="1" />
      <xs:element xs:name="INTERPRETE" xs:type="xs:string"
                  xs:minOccurs="1"     xs:maxOccurs="unbounded" />
      <xs:element xs:name="COMPOSITEUR" xs:type="xs:string" xs:minOccurs="1" />
      <xs:element xs:name="DUREE" xs:type="xs:timeDuration"
   xs:minOccurs="1" />
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

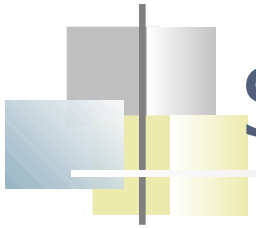


Schéma XML

- Un Schéma XML est un document XML
  - Son élément racine
    - Espace de noms - Spécification de schéma

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

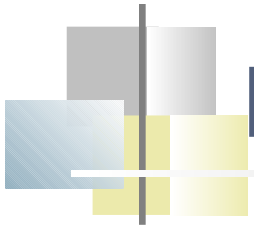
  - Chaque élément est ensuite décrit précisément

Schéma XML

- Déclaration
  - des éléments `xs:name, xs:type, xs:minoccurs, xs:maxoccurs`
  - des attributs `xs:name, xs:type, xs:use, xs:default, xs:fixed`

- Spécification de type simple ou complexe
  - Définir le contenu d'un élément
  - types simples : type de base, extensible par contrainte

    `string, boolean, decimal, integer, nonpositiveinteger, date, ID, IDREF, month, timeduration ...`
  - types complexes : agrégation d'éléments typés
    - `Sequence` séquence d'éléments typés
    - `All` collection non ordonnée d'éléments typés
    - `Choice` alternative entre éléments typés



Définition d'un type simple

- Un type simple ne peut pas contenir de déclarations d'éléments ou d'attributs

<xs:element xs:name="*name***" xs:type="***type***"/>**

- Il peut être
  - Prédéfini
  - Dérivé de types existants
  - Anonyme / nommé

```
<xs:element xs:name="date" xs:type="xs:date"/>
<date>1999-05-21</date>
```

Les types simples (1)

- string
  - Confirm this is electric
- normalizedString
  - Confirm this is electric
- token
  - Confirm this is electric
- byte
  - -1, 126
- unsignedByte
  - 0, 126
- base64Binary
  - GpM7
- hexBinary
  - 0FB7
- integer
  - -126789, -1, 0, 1, 126789
- positiveInteger
  - 1, 126789
- negativeInteger
  - -126789, -1
- nonNegativeInteger
  - 0, 1, 126789
- nonPositiveInteger
  - -126789, -1, 0
- int
  - -1, 126789675
- unsignedInt
  - 0, 1267896754

Les types simples (2)

- long
  - -1, 12678967543233
- unsignedLong
  - 0, 12678967543233
- short
  - -1, 12678
- unsignedShort
  - 0, 12678
- decimal
  - -1.23, 0, 123.4, 1000.00
- float
  - -INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
- double
  - -INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
- boolean
  - true, false 1, 0
- time
  - 13:20:00.000, 13:20:00.000-05:00
- dateTime
  - 1999-05-31T13:20:00.000-05:00
- duration
  - P1Y2M3DT10H30M12.3S
- date
  - 1999-05-31
- gMonth
  - --05--
- gYear
  - 1999

Les types simples (3)

- gYearMonth
  - 1999-02
- gDay
  - ---31
- gMonthDay
  - --05-31
- Name
  - shipTo
- QName
  - po:USAddress
- NCName
  - USAddress
- anyURI
  - http://www.example.com/,
  - http://www.example.com/doc.html#ID5
- language
  - en-GB, en-US, fr
- ID
  - "A212"
- IDREF
  - "A212"
- IDREFS
  - "A212" "B213"
- ENTITY
- ENTITIES
- NOTATION
- NMTOKEN, NMTOKENS
  - US
  - Brésil Canada Mexique

Types simples - Contraintes

→ <quantité>1</quantité>

Type simple anonyme dérivé par restriction

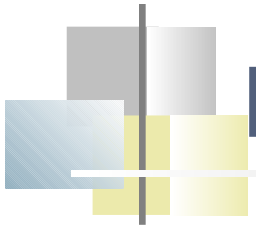
```
<xs:element xs:name="quantité">
   <xs:simpleType>
         <xs:restriction xs:base="xs:positiveinteger">
            <xs:maxexclusive xs:value="100"/>
         </xs:restriction>
   </xs:simpleType>
</xs:element>
```

→ <item *num='926-AA'*>...</item>

Type simple défini par utilisation d'expressions régulières

Définition du type nommé SKU

```
<xs:attribute xs:name="num" xs:type="SKU"/>
<xs:simpleType xs:name="SKU">
       <xs:restriction xs:base="xs:string">
           <xs:pattern xs:value="\d{3}-[A-Z]{2}"/>
       </xs:restriction>
</xs:simpleType>
```



Définition d’un type complexe

- Un type complexe définit des compositions d’éléments et/ou d’attributs
  - À contenu simple
    - Applicables aux éléments qui ont un contenu purement textuel mais avec attributs
    - Les types simples tels quels ne sont donc pas applicables
  - À contenu complexe
- Aucun type complexe n’est prédéfini

Définition d'un type complexe

```
<xs:element xs:name="name">
      <xs:complexType>
         modèle_de_contenu
      </xs:complexType>
</xs:element>
```

```
<xs:complexType xs:name="TypeName">
         modèle_de_contenu
</xs:complexType>

<xs:element xs:name="name" xs:type="TypeName"/>
```

Exemple (1a)

```
<xs:element xs:name="Livre">
  <xs:complexType>
    <xs:sequence>
        <xs:element xs:name="titre" xs:type="xs:string"/>
        <xs:element xs:name="auteur" xs:type="typeAuteur"
                    xs:minoccurs="1"  xs:maxoccurs="unbounded"/>
        <xs:element xs:name="année" xs:type="gYear"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<Livre>
 <titre>montitre</titre>
 <auteur>monauteur1</auteur>
 <auteur>monauteur2</auteur>
 <année>2008</année>
</Livre>
```

Exemple (1b)

```
<xs:complexType xs:name="typeLivre">
    <xs:sequence>
    <xs:element xs:name="titre" xs:type="xs:string"/>
      <xs:element xs:name="auteur" xs:type="typeAuteur"
      xs:minoccurs="1" xs:maxoccurs="unbounded"/>
      <xs:element xs:name="année" xs:type="gYear"/>
    </xs:sequence>
    <xs:attribute xs:name="isbn" xs:type="xs:string"/>
</xs:complexType>

<xs:element xs:name="Livre" xs:type="typeLivre">

<Livre isbn="1…9">
 <titre>montitre</titre>
 <auteur>monauteur1</auteur>
 <auteur>monauteur2</auteur>
 <année>2008</année>
</Livre>
```

```
<xs:element xs:name="Livre" xs:type="typeLivre">
```

```
<Livre isbn="1…9">
 <titre>montitre</titre>
 <auteur>
          <nom>nom1</nom>
          <prenom>prenom1</prenom>
 </auteur>
 <auteur>
          <nom>nom2</nom>
          <prenom>prenom2</prenom>
 </auteur>
 <année>2008</année>
</Livre>
```

Exemple (1c)

Types simples

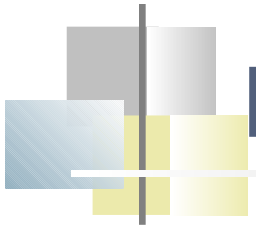
```
<xs:element xs:name="titre" xs:type="xs:string"/>

<xs:element xs:name="année" xs:type="gYear"/>
```

Types complexes

```
<xs:complexType xs:name="typeAuteur">
   <xs:sequence>
     <xs:element xs:name="nom" xs:type= "xs:string"/>
     <xs:element xs:name="prenom" xs:type= "xs:string"/>
  </xs:sequence>
 </xs:complexType>

<xs:complexType xs:name="typeLivre">
    <xs:sequence>
      <xs:element xs:ref="titre"/>
      <xs:element xs:name="auteur" xs:type="typeAuteur"
                  xs:minoccurs="1" xs:maxoccurs="unbounded"/>
      <xs:element xs:ref="année"/>
   </xs:sequence>
   <xs:attribute xs:name="isbn" xs:type="xs:string"/>
 </xs:complexType>
```

Exemple (2)

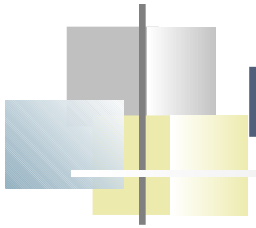
```
<xs:complexType xs:name="xs:typeLivre">
   <xs:sequence>
      <xs:element xs:name="titre" xs:type="string"/>
      <xs:element xs:name="auteur" xs:type="typeAuteur"
                  xs:minoccurs="1" xs:maxoccurs="unbounded"/>
      <xs:choice>
         <xs:sequence>
              <xs:element xs:name="editeur" xs:type="xs:string"/>
              <xs:element xs:name="edition"   xs:type="xs:integer"/>
         </xs:sequence>
         <xs:element xs:name="année" xs:type="gYear"/>
      </xs:choice>
   </xs:sequence>
   <xs:attribute xs:name="ISBN" xs:type="string"/>
</xs:complexType>
```



Exemple suite (2)

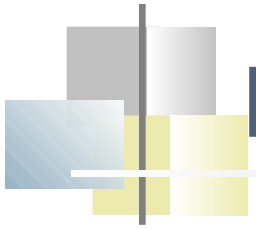
```
<Livre isbn="1…9">
  <titre>montitre</titre>
  <auteur>monauteur1</auteur>
  <auteur>monauteur2</auteur>
  <année>2008</année>
</Livre>

<Livre isbn="1…9">
  <titre>montitre</titre>
  <auteur>monauteur1</auteur>
  <auteur>monauteur2</auteur>
  <editeur>monediteur</editeur>
  <edition>1</edition>
</Livre>
```

Chacun de ces éléments peut apparaître une fois ou pas du tout
L'ordre des éléments n'a pas d'importance (cela n'a pas d'équivalent dans une DTD).



Exemple (3)

```
<xs:complexType>
  <xs:all>
    <xs:element xs:name="nom" xs:type="xs:string" />
    <xs:element xs:name="prénom" xs:type="xs:string" />
    <xs:element xs:name="dateDeNaissance" xs:type="xs:date" />
    <xs:element xs:name="adresse" xs:type="xs:string" />
    <xs:element xs:name="adresseElectronique" xs:type="xs:string" />
    <xs:element xs:name="téléphone" xs:type="numéroDeTéléphone" />
  </xs:all>
</xs:complexType>
```



Exemple (4) – contenu mixte

```
<xs:complexType  xs:mixed="true"  xs:name="typeExemple">
   <xs:sequence>
      <xs:element xs:name="sigle" xs:type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:element xs:name="exemple" xs:type="typeExemple">

<exemple>Le <sigle>XML</sigle> est connu de tous</exemple>
```

Exemple (5) – contenu vide

```
<xs:complexType  xs:name="Duréetype">
   <xs:attribute xs:name="durée" xs:type="xs:decimal"/>
  <xs:attribute  xs:name="unité" xs:type="xs:string"/>
</xs:complexType>

<xs:element xs:name="DuréeItinéraire" xs:type="Duréetype">

<DuréeItinéraire durée="3" unité="heure"/>
```

Héritage de types

- par restriction : ajout de contraintes
- par extension : ajout d'informations

```
<xs:complexType xs:name="Adresse" >
  <xs:sequence>
       <xs:element xs:name="rue"  xs:type="string"/>
       …
     </xs:sequence>
</xs:complexType>
```

```
<xs:complexType xs:name="AdressePays" >
  <xs:complexContent>
    <xs:extension base="Adresse"/>
      <xs:sequence>
        <xs:element xs:name="Pays" xs:type= "xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!ELEMENT listeCD (cd)*>
<!ELEMENT cd (nom, (piste)*)>
<!ELEMENT piste (num, titre, interprete+, compositeur, duree)>
```

Référencer dans les déclarations d'éléments les types propres définis

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType xs:name= "TypeListeCD" >
     <xs:sequence>
      <xs:element xs:ref="CD" xs:minOccurs="0" xs:maxOccurs="unbounded" >
     </xs:sequence>
  </xs:complexType>
</xs:schema>

<xs:element xs:name="LISTECD" xs:type= "TypeListeCD" >
<xs:element xs:name="CD" xs:type= "TypeCD" >
<xs:element xs:name="Piste" xs:type= "TypePiste" >
```

Exercice

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">

   <xsd:element name="commande" type="CommandeType"/>
   <xsd:element name="commentaire" type="xsd:string"/>

   <xsd:complexType name="CommandeType">
        <xsd:sequence>
           <xsd:element name="livrer" type="Adresse"/>
           <xsd:element name="facturer" type="Adresse"/>
           <xsd:element ref="commentaire" minOccurs="0"/>
           <xsd:element name="produits" type="ProduitType"/>
        </xsd:sequence>
        <xsd:attribute name="date_com" type="xsd:date"/>
     </xsd:complexType>
```

Exercice suite

```
<xsd:complexType name="ProduitType">
  <xsd:sequence>
    <xsd:element name="produit" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="nom_prod" type="xsd:string"/>
          <xsd:element name="quantite">
            <xsd:simpleType>
              <xsd:restriction base="xsd:positiveInteger">
                <xsd:maxExclusive value="100"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="prix" type="xsd:decimal"/>
          <xsd:element ref="commentaire" minOccurs="0"/>
          <xsd:element name="date_livraison" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="num_prod" type="xsd:positiveInteger" use="required"/>
      </xsd:complexType>
```

Exercice suite

```
        </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Schéma XML : valeurs uniques et clés

- Certains objets - éléments, attributs ou n-uplets formés d'éléments et/ou d'attributs - doivent avoir une valeur unique
  - Clé : les constructeurs `unique` et `key`
    - sélection à l'intérieur d'une certaine portée (d'un ensemble de nœuds)
      - d'un objet dont la valeur doit être unique
      - d'un ensemble d'objets dont les valeurs doivent être uniques
    - `Unique`
      - Déclaration de champ à valeur unique, s'il existe, dans une portée
    - `Key`
      - Déclaration de champ à valeur unique dans une portée; il doit exister
  - Clé étrangère : constructeur `keyref`

Exemple tiré du livre d'Alain Michard

Exemple complet

```
<?xml version="1.0"?>
<meteo>
   <obs num='1'>
        <loc>Paris-Montsouris</loc>
        <date>2001-10-22T15:31:05</date>
   </obs>
   <obs num='5'>
        <loc>Pic-du-midi</loc>
        <date>2001-10-22T15:33:05</date>
   </obs>
   <!-- autres observations -->

   <resultat>
        <mesure observation='1'>…</mesure>
        <mesure observation='5'>…</mesure>
   </resultat>

</meteo>
```

Clé

`field` doit obligatoirement identifier un attribut ou un élément de type simple

`field` et `selector` contiennent des expressions XPath (un sous-ensemble)

```
<meteo>
  <obs num='1'>
    <loc>…</loc>
    <date>…</date>…
  </obs>
   …
</meteo>
```

```
<xs:key xs:name="maclé" >
    <xs:selector xs:xpath="obs"/>
    <xs:field xs:xpath="@num"/>
</xs:key>
```

Déclaration de champ à valeur unique (`field`) dans une portée (`selector`), l'objet doit exister

Clé étrangère

```
<resultat>
  <mesure observation='1'>
  …
  </mesure>
  …
</resultat>
```

Déclaration de champ référençant un champ à valeur unique (`field`) dans une portée (`selector`)

```
<xs:keyref xs:name="cleref" xs:refer="maclé" >
    <xs:selector xs:xpath="mesure"/>
    <xs:field xs:xpath="@observation"/>
</xs:keyref>
```

Exemple tiré du livre d'Alain Michard

Exemple complet

```
<?xml version="1.0"?>
<meteo>
   <obs num='1'>
        <loc>Paris-Montsouris</loc>
        <date>2001-10-22T15:31:05</date>
   </obs>
   <obs num='5'>
        <loc>Pic-du-midi</loc>
        <date>2001-10-22T15:33:05</date>
   </obs>
   <!-- autres observations -->

   <resultat>
        <mesure observation='1'>…</mesure>
        <mesure observation='5'>…</mesure>
   </resultat>

</meteo>
```

La position de la déclaration des contraintes (à l'intérieur de la définition de l'élément `meteo`) donne le contexte à partir duquel le contrôle sera effectué

Exemple complet (fin)

```
<xs:complexType xs:name="meteo">
 <xs:sequence>
    <xs:element xs:name="obs"    xs:type="xs:type_obs"
                xs:minOccurs="1" xs:maxOccurs="unbounded" />
    <xs:element xs:name="resultat"  xs:type="xs:type_resultat"
                                    xs:minOccurs="1"  />
 </xs:sequence>
 <xs:key xs:name="maclé">
      <xs:selector xs:xpath="obs"/>
       <xs:field xs:xpath="@num"/>
  </xs:key>
  <xs:keyref xs:name="cleref" xs:refer="maclé">
      <xs:selector xs:xpath="mesure"/>
      <xs:field xs:xpath="@observation"/>
    </xs:keyref>
</xs:complexType>
```

Exemple complet (suite)

```
<xs:complexType xs:name="xs:type-obs">
  <xs:sequence>
     <xs:element xs:name="loc" xs:type="xs:string"/>
     <xs:element xs:name="date" xs:type="xs:date"/>
  </xs:sequence>
  <xs:attribute xs:name="num" xs:type="xs:positiveInteger"
                            xs:use="required"/>
</xs:complexType>

<xs:complexType xs:name="xs:type-resultat">
  <xs:sequence>
     <xs:element xs:name="mesure" xs:type="xs:string"
                                  xs:maxoccurs="unbounded"/>
     <xs:attribute xs:name="observation" xs:type="xs:positiveInteger"
                                         xs:use="required"/>
  </xs:sequence>
</xs:complexType>
```

Référencer un schéma dans un document XML

- Le fichier XML

```
<exemple
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="biblio2.xsd">
  ...
</exemple>
```

- Le fichier contenant le schéma

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   ...
</xs:schema>
```