XML Schema Definition Language

INTRODUCTION À XML SCHEMA

Document XML valide

- Document bien formé: respecte la syntaxe XML
- Document valide: respecte un modèle, les règles d'un langage de balisage (vocabulaire & grammaire)
  - DTD: Document Type Definition
  - **XML Schema**
- Application XML ou Type de document:

  Langage de balisage qui respecte les règles syntaxiques de XML:

  XHTML, MathML, SVG, RDF, OWL, etc.
- XML: (méta)langage de définition de langages

XML Schema ≥ DTD

- Déclaration des éléments et des attributs
  - Un élément est de type simple ou défini
  - Définition de types d'éléments
  - Définition de types par extension de types existants
  - Hiérarchie de types
- Types de données
  - Types prédéfinis: string, integer, énumération, date, …
  - Précision des types (facettes et motifs)
  - XML Schema fournit un contrôle plus strict qu'une DTD sur les types de données et les motifs de texte
- Syntaxe XML

Recommandation du W3C

http://www.w3.org/XML/Schema#dev

- XML Schema Part 0 : Primer
- XML Schema Part 1 : Structure
- XML Schema Part 2 : Datatypes

Exemple

```
<bibliography>
  <book id='x223'>
      <author>
            <firstname>David</firstname>
                  <lastname>Lodge</lastname>
      </author>
      <title>Small World</title>
      <publisher>Penguin Books</publisher>
      <year>1995</year>
  </book>
  …
</bibliography>
```

Exemple

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="book" type="bookType"/>
 <xs:complexType name='bookType'>
  <xs:sequence>
     <xs:element name='author' type='authorType'/>
     <xs:element name='title' type='xs:string'/>
     <xs:element name='publisher' type='xs:string'/>
     <xs:element name='year' type='xs:integer'/>
  </xs:sequence>
  <xs:attribute name='id' type='xs:string'/>
 </xs:complexType>
 <!-- il manque la définition du type authorType -->
 <!-- il manque la définition du type de bibliography -->
</xs:schema>
```

XML Schema est un langage XML

Les éléments du language XML Schema sont définis dans un namespace :

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

Déclaration des éléments et attributs

```
<xs:element name="book" type="bookType"/>

<xs:complexType name='bookType'>
   …
   <xs:attribute name='id' type=‘xs:string'/>
</xsd:complexType>
```

Définition de type complexe d'élément

```
<xs:element name="book" type="bookType"/>
 <xs:complexType name='bookType'>
            …
</xs:complexType>
```

- L'élément name est déclaré de type bookType
- Le type bookType est défini par un élément complexType

```
<xs:element name="book" >
   <xs:complexType>
            …
    </xs:complexType>
</xs:element>
```

- Le type de l'élément name est anonyme

Définition de type complexe d'élément

```
<xs:element name="book" type="bookType"/>
<xs:complexType name='bookType'>
    <xs:sequence>
        <xs:element name='author' type='authorType'/>
        <!-- autres éléments fils de l'élément book-->
    </xs:sequence>
</xs:complexType>

<xs:element name="book" type="bookType"/>
<xs:element name='author' type='authorType'/>
<xs:complexType name='bookType'>
    <xs:sequence>
        <xs:element ref='author'/>
        <!-- autres éléments fils de l'élément book-->
    </xs:sequence>
</xs:complexType>
```

Définition de type complexe d'élément

- xs:sequence

  Les éléments doivent apparaître 1 fois, dans l'ordre indiqué

```
<xs:complexType name='bookType'>
    <xs:sequence>
        <xs:element name='author' type='author'/>
        <xs:element name='title' type='xs:string'/>
        <xs:element name='publisher' type='xs:string'/>
        <xs:element name='year' type='xs:integer'/>
    </xs:sequence>
    …
</xs:complexType>
```

Définition de type complexe d'élément

- xs:all

  Les éléments doivent apparaître 1 fois, sans ordre

```
<xs:complexType name='bookType'>
    <xs:all>
        <xs:element name='author' type='author'/>
        <xs:element name='title' type='xs:string'/>
        <xs:element name='publisher' type='xs:string'/>
        <xs:element name='year' type='xs:integer'/>
    </xs:all>
    …
</xsd:complexType>
```

Définition de type complexe d'élément

- xs:choice

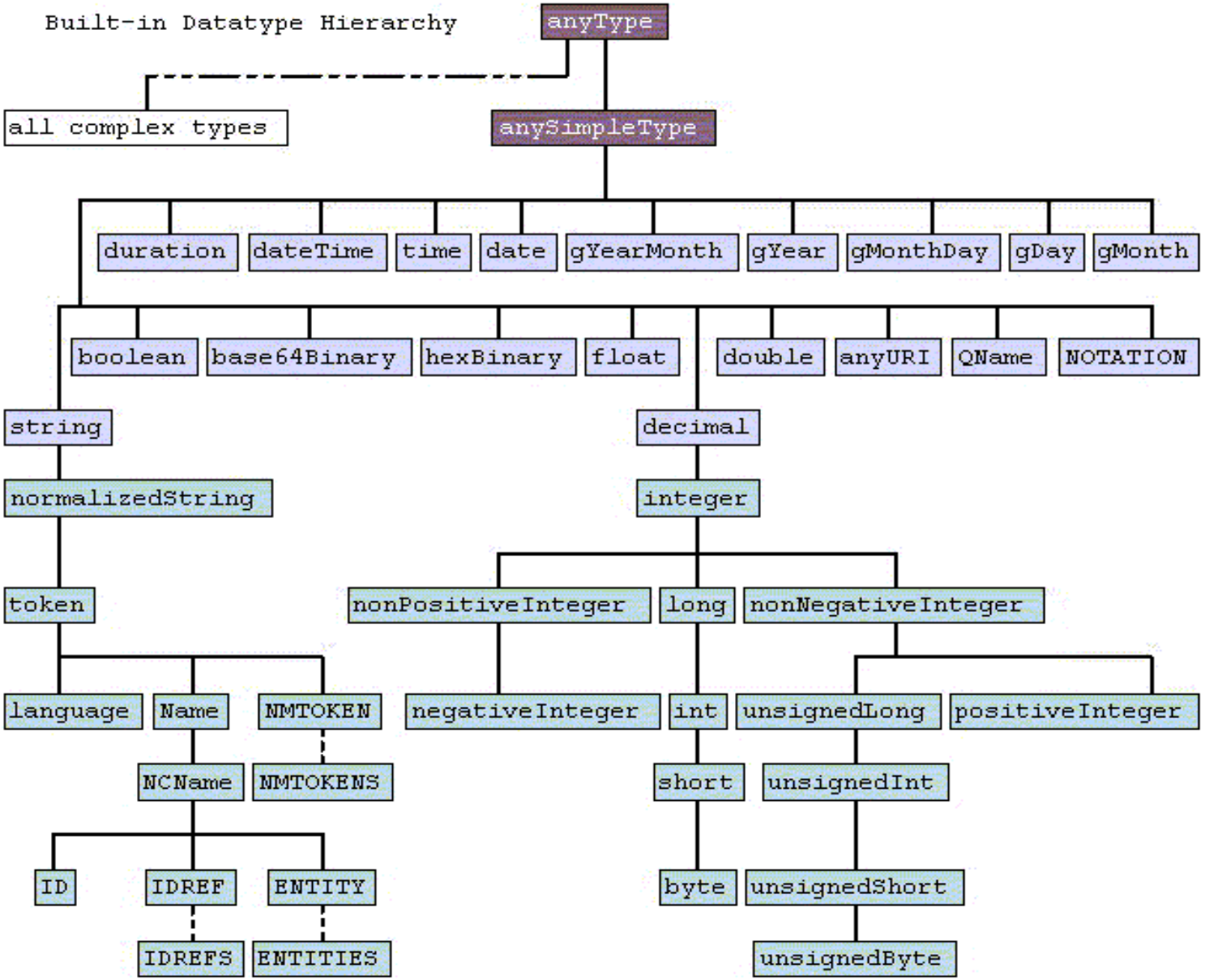
```
<xs:complexType name='bookType'>
    <xs:sequence>
        <xs:element name='author' type='author'/>
        <xs:element name='title' type='xs:string'/>
        <xs:element name='publisher' type='xs:string'/>
        <xs:choice>
            <xs:element name='year' type='xs:gYear'/>
            <xs:element name='date' type='xs:date'/>
        </xs:choice>
    </xs:sequence>
    …
</xs:complexType>
```

Types simples prédéfinis

```
<xs:element name='year' type='xs:integer'/>
<xs:element name='title' type='xs:string'/>
<xs:attribute name='id' type='xs:string'/>
```

string, integer, decimal, float, Boolean, time, timeInstant, Date, Name, QName, ID, IDREF, NMTOKEN, ENTITY, …

Types simples prédéfinis



Facettes, sur type simple ou complexe

Propriétés qui précisent un type de données:

minOccurs, maxOccurs, length, minLength, maxLength, maxInclusive, minExcusive, maxExclusive, minInclusive, …

```
<xs:element name="date" type="xs:integer"
        minOccurs="0" maxOccurs="1" fixed="2000" default="2000"/>

<xs:ComplexType name="bibliograpyType">
    <xs:sequence>
        <xs:element name="book" type="bookType"
                minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:ComplexType>
```

Dans un élément xs:all, les éléments ne peuvent apparaitre que 0 ou 1 fois

Définition d'un type simple d'élément

Type simple étendu:

- xs:simpleType
- xs:restriction avec attribut base
- facettes

Définition de myInteger, range 10000-99999

```
<xs:simpleType  name="myInteger">
    <xs:restriction  base="xs:integer">
      <xs:minInclusive  value="10000"/>
      <xs:maxInclusive  value="99999"/>
    </xs:restriction>
</xs:simpleType>
```

Définition d’un type simple d’élément

Type simple étendu

Spécification d'un motif à l'aide d‘une expression régulière

Date sous la forme 16-10-1959 :

```
<xs:simpleType name="Date">
   <xs:restriction base="xs:string">
     <xs:pattern value="\d{2}-\d{2}-\d{4}"/>
   </xs:restriction>
</xs:simpleType>
```

Définition d’un type simple d’élément

Type simple énuméré

```
<xs:simpleType name="Region">
   <xs:restriction base="xs:string">
     <xs:enumeration value="Alsace"/>
     <xs:enumeration value="Centre"/>
     <xs:enumeration value="Languedoc-Roussillon"/>
     <xs:enumeration value="PACA"/>
     <!-- etc ... -->
   </xs:restriction>
</xs:simpleType>
```

Définition d’un type simple d’élément

Type liste

```
<xs:simpleType name="ListeRegion">
    <xs:list itemType="Region"/>
</xs:simpleType>

<xs:element name="regions" type="ListeRegion"/>
```

Elément valide de type ListeRegion :

```
<regions>Alsace PACA</regions>
```

Définition d’un type simple d’élément

Union de types

- <xs:simpleType name="zipUnion">
  <xs:union memberTypes="USState listOfMyIntType"/>
  </xs:simpleType>

- <xs:simpleType>
  <xs:union>
  <xs:simpleType>
  <xs:restriction base='nonNegativeInteger'/>
  </xs:simpleType>
  <xs:simpleType>
  <xs:restriction base='string'>
  <xs:enumeration value='unbounded'/>
  </xs:restriction>
  </xs:simpleType>
  </xs:union>
  </xs:simpleType>

Définition d'un type simple d'élément

Un type « simple » avec attribut(s) est vu comme un type complexe de contenu simple

```
<xs:element name="price">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attribute name="currency" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Elément valide : <price currency="EUR">423.46</price>

Déclaration d'élément de contenu vide

- Type d'élément de contenu vide
  (a donc nécessairement au moins un attribut)

```
<xs:element name='vacuum'>
    <xs:complexType>
      <xs:attribute .../>
    </xs:complexType>
</xs:element>

<author firstname="David" lastname="Lodge"/>
```

- Elément de contenu vide

```
<xsd:element name="year" type="xs:integer" nillable="true"/>

<year xsi:nil="true"/>
```

Elément de contenu mixte

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<letter>
  Cher Monsieur <name>Jean Dupont</name>.
  Votre commande  <orderid>1032</orderid>
  sera expédiée le <shipdate>2016-09-19</shipdate>.
</letter>
```

Types quelconques

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>
</xs:element>
```

Définition d'un type par spécialisation

xs:extension peut aussi s'utiliser avec un type complexe

```
<xs:complexType name='eBookType'>
    <xs:complexContent>
        <xs:extension base='bookType'>
            <xs:sequence>
                <xs:element name='URI' type='uriReference'/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Définition d’un type par spécialisation

```
<xs:element name="item" type="bookType"/>
```

Elément item de type eBookType valide :

```
<item xsi:type="eBookType">
    <author> ... </author>
    ...
    <date> … </date>
    <URI> ... </URI>
</s:item>
```

Définition d'un type par spécialisation

- Types abstraits

  <xs:complexType name="Vehicle" **abstract="true"**/>

  – Nécessité d'utiliser des éléments de types dérivés dans le document XML (munis de l'attribut xsi:type)

- Limiter la dérivation de types dans un schéma XML

  <xs:complexType name="Address" **final**="restriction|extension|#all">

- Limiter l'utilisation d'un type dérivé à la place d'un type de base dans un document XML

  <xs:complexType name="Address" **block**='restriction|extension|#all'>

Contraintes d’intégrité

- Exemple

```
<purchaseReport
     xmlns="http://www.example.com/Report"
     period="P3M" periodEnding="1999-12-31">
  <regions>
    <zip code="95819">
      <part number="872-AA" quantity="1"/>
      <part number="926-AA" quantity="1"/>
      <part number="833-AA" quantity="1"/>
      <part number="455-BX" quantity="1"/>
    </zip>
    <zip code="63143">
      <part number="455-BX" quantity="4"/>
    </zip>
  </regions>
  <parts>
    <part number="872-AA">Lawnmower</part>
    <part number="926-AA">Baby Monitor</part>
    <part number="833-AA">Lapis Necklace</part>
    <part number="455-BX">Sturdy Shelves</part>
  </parts>
</purchaseReport>
```

Contraintes d'intégrité

- Unicité de la valeur d'un attribut

La valeur de l'attribut code des éléments purchaseReport /regions/zip est unique :

```
<xs:element name="purchaseReport">
   <xs:complexType> … </xs:complexType>
   <xs:unique name="dummy1">
    <xs:selector xpath='regions/zip'/>
    <xs:field xpath='@code'/>
   </xs:unique>
   …
</xs:element>
```

Contraintes d'intégrité

- Exemple

```
<purchaseReport
     xmlns="http://www.example.com/Report"
     period="P3M" periodEnding="1999-12-31">
  <regions>
    <zip code="95819">
      <part number="872-AA" quantity="1"/>
      <part number="926-AA" quantity="1"/>
      <part number="833-AA" quantity="1"/>
      <part number="455-BX" quantity="1"/>
    </zip>
    <zip code="63143">
      <part number="455-BX" quantity="4"/>
    </zip>
  </regions>
  <parts>
    <part number="872-AA">Lawnmower</part>
    <part number="926-AA">Baby Monitor</part>
    <part number="833-AA">Lapis Necklace</part>
    <part number="455-BX">Sturdy Shelves</part>
  </parts>
</purchaseReport>
```

Contraintes d'intégrité

- Clé et référence

La valeur de l'attribut code des éléments purchaseReport /regions/zip est unique :

```
<xs:element name="purchaseReport">
   <xs:complexType> … </xs:complexType>
   ….
   <xs:key name="pNumKey">
    <xs:selector xpath="r:parts/r:part"/>
    <xs:field xpath="@number"/>
   </xs:key>

   <xs:keyref name="dummy2" refer="r:pNumKey">
    <xs:selector xpath="r:regions/r:zip/r:part"/>
    <xs:field xpath="@number"/>
   </xs:keyref>    …
</xs:element>
```

Contraintes d'intégrité

- Exemple

```
<purchaseReport
    xmlns="http://www.example.com/Report"
    period="P3M" periodEnding="1999-12-31">
 <regions>
  <zip code="95819">
   <part number="872-AA" quantity="1"/>
   <part number="926-AA" quantity="1"/>
   <part number="833-AA" quantity="1"/>
   <part number="455-BX" quantity="1"/>
  </zip>
  <zip code="63143">
   <part number="455-BX" quantity="4"/>
  </zip>
 </regions>
 <parts>
  <part number="872-AA">Lawnmower</part>
  <part number="926-AA">Baby Monitor</part>
  <part number="833-AA">Lapis Necklace</part>
  <part number="455-BX">Sturdy Shelves</part>
 </parts>
</purchaseReport>
```

Documentation

```
<xs:annotation>
    <xs:documentation>
        Ceci est un commentaire
    </xs:documentation>
</xs:annotation>
```

Peut apparaître en tête de schéma ou dans tout autre élément

Associer un schéma à un document

1. Document XML sans déclaration de namespace
   Schéma XML avec noms d'éléments et d'attributs déclarés sans (target) namespace

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

Associer un schéma à un document

2. Document XML avec déclaration de namespace
   Schéma XML avec un target namespace et, par défaut, les noms d'éléments et d'attributs déclarés localement non qualifiés

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:po="http://www.example.com/PO1"
    targetNamespace="http://www.example.com/PO1">
 <element name="purchaseOrder" type="po:PurchaseOrderType"/>
 <element name="comment" type=« xs:string"/>
 <complexType name="PurchaseOrderType">
  <sequence>
   <element name="shipTo" type="po:USAddress"/>
   <element name="billTo" type="po:USAddress"/>
   <element ref="po:comment" minOccurs="0"/>
  </sequence>
 </complexType>
 <complexType name="USAddress"> <!-- etc. --> </complexType>
 <!-- etc. -->
</schema>
```

Associer un schéma à un document

2. Document XML avec déclaration de namespace
   Schéma XML avec un target namespace et, par défaut, les noms d'éléments et d'attributs déclarés localement non qualifiés

```
<?xml version="1.0"?>
<apo:purchaseOrder
    xmlns:apo="http://www.example.com/PO1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.example.com/PO1  po.xsd"
    orderDate="1999-10-20">
 <shipTo country="US">
  <name>Alice Smith</name>
  <street>123 Maple Street</street>
  <!-- etc. -->
 </shipTo>
 <billTo country="US">
  <!-- etc. -->
 </billTo>
 <apo:comment>Hurry, my lawn is going wild<!/apo:comment>
 <!-- etc. -->
</apo:purchaseOrder>
```

Associer un schéma à un document

3. Document XML avec déclaration de namespace
   Schéma XML avec un target namespace et noms d'éléments déclarés localement qualifiés

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:po="http://www.example.com/PO1"
    targetNamespace=http://www.example.com/PO1
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <element name="purchaseOrder" type="po:PurchaseOrderType"/>
  <element name="comment" type=« xs:string"/>
  <complexType name="PurchaseOrderType">
    <!-- etc. -->
  </complexType>
  <!-- etc. -->
</schema>
```

Par défaut, éléments et attributs sont unqualified et la déclaration est inutile

Associer un schéma à un document

3. Document XML avec déclaration de namespace
   Schéma XML avec un target namespace et noms d'éléments déclarés localement qualifiés

```
<?xml version="1.0"?>
<apo:purchaseOrder
     xmlns:apo="http://www.example.com/PO1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.example.com/PO1  po.xsd"
     orderDate="1999-10-20">
  <apo:shipTo country="US">
   <apo:name>Alice Smith</apo:name>
   <apo:street>123 Maple Street</apo:street>
   <!-- etc. -->
  </apo:shipTo>
  <!-- etc. -->
</apo:purchaseOrder>
```

Associer un schéma à un document

3. Document XML avec déclaration de **namespace par défaut**
   Schéma XML avec un target namespace et noms d'éléments déclarés localement qualifiés

```
<?xml version="1.0"?>
<purchaseOrder
    xmlns="http://www.example.com/PO1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.example.com/PO1  po.xsd"
    orderDate="1999-10-20">
  <shipTo country="US">
   <name>Alice Smith</name>
   <street>123 Maple Street</street>
   <!-- etc. -->
  </shipTo>
  <!-- etc. -->
</purchaseOrder>
```