Les schémas XML

Nicolas Singer

Maître de conférence, université Champollion

Pourquoi les schémas XML ?

- Les DTD sont bien adaptés à la définition de la structure de documents textuels (typage faible des éléments terminaux).

- XML a évolué et est utilisé pour représenter tout type de données (dont il faudrait pouvoir définir le type)

  --> **définition d'un langage (en XML) destiné à définir plus précisément les types d'éléments et d'attributs**

Vue générale des schémas à travers un exemple

```
<meteo>
   <obs num="PY1">
        <loc>Paris</loc>
        <date>1998-10-22T15:31:05</date>
        <temp unit="celsius">12.3</temp>
        <hygro>88</hygro>
        <pluvio>6</pluvio>
   </obs>
   <obs num="PY2">
        <loc>Marseille</loc>
        <date>1998-10-22T16:31:05</date>
        <temp unit="celsius">14</temp>
        <hygro>54</hygro>
        <pluvio>3</pluvio>
        <message>Pluviomètre primaire HS</message>
   </obs>
</meteo>
```

DTD correspondant à l'exemple

```
<!ELEMENT meteo (obs)*>
<!ELEMENT obs (loc,date,temp?,hygro?,pluvio?, message?) >
<!ATTLIST obs num ID #REQUIRED>
<!ELEMENT loc (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT temp (#PCDATA)>
<!ATTLIST temp unit (celcius|farenheight|kelvin) 'celcius'>
<!ELEMENT hygro (#PCDATA)>
<!ELEMENT pluvio (#PCDATA)>
<!ELEMENT message (#PCDATA)>
```

- La DTD ne dit rien sur le typage des valeurs de pluviosité (comprises entre 0 et 8), d'hygrométrie (un pourcentage), et de température (une valeur qui sortirait de l'intervalle –50, 50 serait certainement erronée).

Le schéma XML correspondant

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

<xs:element name="meteo" type="meteoType" />

<xs:complexType name="meteoType" >
    <xs:sequence>
        <xs:element name="obs" type="observation"
            minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="observation">
    <xs:sequence>
      <xs:element name="loc" type="xs:string" />
      <xs:element name="date" type="xs:timeInstant" />
      <xs:element name="temp" type="tempType" minOccurs="0" maxOccurs="1" />
      <xs:element name="hygro" type="pourcent" minOccurs="0" maxOccurs="1" />
      <xs:element name="pluvio" type="octal" minOccurs="0" maxOccurs="1" />
      <xs:element name="message" type="xs:string" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="num" type="xs:string" use="required" />
</xs:complexType>
```

Le schéma XML correspondant (suite)

```
<xs:simpleType name="octal">
    <xs:restriction base="xs:non-negative-integer">
        <xs:maxInclusive value="8" />
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="pourcent"
    <xs:restriction base="xs:non-negative-integer">
        <xs:maxInclusive value="100" />
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="tempType" >
    <xs:simpleContent>
        <xs:extension base="xs:decimal">
            <xs:minInclusive value="-50.0" />
            <xs:maxInclusive value="50.0" />
            <xs:attribute name="unit" type="tempUnit" />
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
</xs:schema>
```

Éléments des schémas XML

Un document schéma est essentiellement composé des éléments :

- **schema** : Racine du document
- **element** : permet de décrire un élément
- **attribute** : permet de décrire un attribut
- **complexType** : permet de décrire un type complexe (composé d'éléments et d'attributs)
- **simpleType** : décrit un type simple (dérivé d'un type simple prédéfini)

L'élément "element"

Il permet de définir le nom/type ou référence d'un élément, ainsi que ses occurrences

Attributs possibles:

name : nom de l'élément

ref : référence au nom d'un élément défini ailleurs

type :type de l'élément

minOccurs, maxOccurs : nbr. d'occurrences autorisé de l'élément

default : valeur par défaut du contenu

nullable : indique si l'élément peut être vide

Exemples de l'élément "element"

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

Défini un élement prénom de type texte

```
<xs:element name="adresse" type="typeAdresse" minOccurs="1"
    maxOccurs="2" />
```

Défini un élément "adresse" d'un type défini par ailleurs , pouvant apparaître entre 1 et 2 fois.

```
<xs:element ref="adresse" minOccurs="1" maxOccurs="1" />
```

Cite un élément "adresse" dont le type a été défini lors de sa définition en le rendant obligatoire à l'endroit où il est cité.

L'élément "attribute"

Il permet de définir le nom/type ou référence d'un attribut, ainsi que ses valeurs par défaut

Attributs possibles:

name : nom de l'attribut

ref : référence au nom d'un attribut défini ailleurs

type : type de l'attribut (obligatoirement un type simple)

use : indique si l'attribut est obligatoire (*required*) ou optionnel (*optional*)

default : donne la valeur par défaut de l'attribut si besoin

fixed : donne sa valeur fixe si besoin

Exemples de l'élément "attribute"

```
<xs:attribute name="min" type="xs:positiveInteger"
    use="required" />
```

Défini un attribut "min" obligatoire et de type entier positif non nul.

```
<xs:attribute name="id" type="xs:ID" />
```

Défini un attribut "id" optionnel de type "identifiant unique".

```
<xs:attribute ref="id" default="123" />
```

Cite l'attribut "id" défini ci-dessus en lui donnant comme valeur par défaut "123".

L'élément "complexType"

Il permet de définir un type complexe, autrement dit un type composé d'éléments et/ou d'attributs

Son contenu est la liste des éléments et des attributs autorisés.

Il peut être dérivé d'un autre type par extension.

Seul un élément peut être d'un type complexe (et jamais un attribut).

Attributs possibles:

name : nom du type

mixed : booléen, si *true* élément mélangeant texte et balises

Exemples de l'élément "complexType"

```
<xs:complexType name="adresseType">
<xs:sequence>
<xs:element name="rue" type="xs:string" />
 <xs:element name="numéro" type="xs:positiveInteger"/>
 <xs:element name="ville" type="xs:string" />
 <xs:element name="pays" type="xs:string" />
<xs:attribute name="format" type="xs:string" />
</xs:sequence>
</xs:complexType>
```

Exemples de l'élément "complexType"

L'exemple précédent peut aussi s'écrire :

```
<xs:attribute name="format" type="xs:string" />
<xs:element name="rue" type="xs:string" />
<xs:element name="numéro" type="xs:positiveInteger" />
<xs:element name="ville" type="xs:string" />
<xs:element name="pays" type="xs:string" />

<xs:complexType name="adresseType">
    <xs:sequence>
        <xs:element ref="rue" />
        <xs:element ref="numéro"
        <xs:element ref="ville" />
        <xs:element ref="pays" />
    </xs:sequence>
    <xs:attribute ref="format" />
</xs:complexType>
```

Dérivation d'un "complexType"

Une fois que le type adresse est défini on peut en dériver un autre à partir de celui-ci

```
<xs:complexType name="adresseLongueType">
<xs:complexContent>
    <xs:extension base="adresseType" >
        <xs:element name="email" type="xs:string"
                minOccurs="1" maxOccurs="1" />
    </xs:extension>
</xs:complexContent>
</xs:complexType>
```

L'élément "simpleType"

Il permet de définir un type élémentaire (sans éléments) en général dérivé d'un autre type élémentaire par restriction. Il possède un attribut "name" permettant de spécifier le nom de ce nouveau type.

Son contenu est alors un élément <xs:restriction base="type" /> qui permet de déterminer de quel type simple prédéfini ou pas, dérive le nouveau type simple.

Le contenu de <xs:restriction> est une liste d'éléments appelés "facettes" qui permettent de restreindre le type simple.

Son contenu peut aussi être une énumération de valeurs que peut avoir un élément de ce type.

Un type simple peut s'appliquer aussi bien à un élément qu'à un attribut.

Exemples de l'élément "simpleType"

```
<xs:simpleType name="codePostalType" >
    <xs:restriction base="xs:string>
        <xs:length value="5" />
    </xs:restriction>
</xs:simpleType>
```

Défini un type qui ne pourra prendre pour valeur qu'un chaîne de 5 caractères.

```
<xs:simpleType name="nombreHexadécimalType">
    <xs:restriction  base="xs:string>
        <xs:pattern value="[0-9A-Fa-f]*" />
    </xs:restriction>
</xs:simpleType>
```

Défini un type qui ne pourra être qu'un chaîne dont les caractères forment un nombre hexadécimal

```
<xs:simpleType name="dépType">
    <xs:restriction base="xs:string>
        <xs:enumeration value="Ain"/>
        <xs:enumeration value="Aine"/>
    <xs:restriction>
</xs:simpleType>
```

Défini un type dont les valeurs sont à prendre dans la liste énumérée.

Liste de quelques types simples prédéfinis

Type	**Valeurs possibles ou exemples**
string	Chaîne de caractères
boolean	true | false | 0 | 1
float | double	12.78E-2 (flottant 32bits | 64bits)
decimal	-1.23, 45.000
dateTime	2000-05-31T13:20:45.000+01:00
date	1999-05-31
time	09:30:10.5
gYear	2005
ID | IDREF	Attribut ID et IDREF de XML
integer | nonPositiveInteger | negativeInteger | long | int | short | byte | nonNegativeInteger | positiveInteger	-126789, -1, 0, 1, 1245, etc.
date	1999-05-31
time	13:20:00

Cas particulier : éléments simple avec attributs

Pour déclarer un élément contenant une valeur simple mais possédant des attributs, on en fait un type complexe dérivant d'un type simple

Ex : <poids unite="kg">67</poids>

```
<xs:complexType name="typePoids">
    <xs:simpleContent>
      <xs:extension base="xs:positiveInteger">
        <xs:attribute name="unite" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

Liste des facettes utilisables pour restreindre un type

Facettes portant sur le temps

- period et duration : prennent pour valeur une durée

Facettes portant sur des types ordonnées

- maxInclusive, minInclusive, maxExclusive, minExclusive : permettent de définir des bornes
- precision et scale : permettent de préciser un nombre de chiffres dans le nombre (précision) et après la virgule (scale)

Facettes portant sur les types ayant une longueur (de la famille des chaînes de caractères)

- lenght, minLength, maxLength: indique la longueur imposée, min ou max d'une valeur

La facette pattern restreint la valeur à être compatible avec l'expression régulières spécifiée

Groupes d'éléments

On peut donner un nom à des groupes d'éléments pour pouvoir y faire référence plus tard (par exemple dans un type complexe)

```
<xs:group name="monGroupe">
    <xs:element ref="adresse" /> minOccurs="1" maxOccurs="5" />
    <xs:element name="nom" type="xs:string" minOccurs=1
    maxOccurs="1" />
</xs:group>

<xs:complexType name="TypeBidon">
     <xs:sequence>
        <xs:element ref="autreElement" />
        <xs:group ref="monGroupe" minOccurs="0" maxOccurs="2" />
     </xs:sequence>
</xs:complexType>
```

Question: Dans une instance d'un élément de type "TypeBidon", combien de fois pourra apparaître un élément "adresse" ?

Groupes d'éléments

On peut également regrouper des éléments dans des listes de choix. Deux constructions sont possibles :

- Construction choice

```
<xs:choice>
    <xs:element …. />
    <xs:element …. />
    <xs:element …./>
    ….
</xs:choice>
```

Un seul des éléments listés peut apparaître dans une instance d'élément ayant ce type.

- Construction all

```
<xs:all>
    <xs:element …. />
    <xs:element …. />
    <xs:element …./>
    ….
</xs:all>
```

Chaque élément peut apparaître zéro ou une fois dans une instance, et ce dans n'importe quel ordre

Notons que ces listes de choix peuvent apparaître encapsulées dans un élément *group* ou directement dans un élément *complexType.*On peut aussi leur donner un nom (avec l'attribut *name* pour pouvoir y faire référence plus tard.

Groupes d'attributs

On peut donner un nom à un ensemble d'attributs pour pouvoir y faire référence plus tard

```
<xs:attributeGroup name="attributsAdresse">
    <xs:attribute name="codePostal" type="xs:string" />
    <xs:attribute name="email" type="xs:string" />
    <xs:attribute name="format" type="xs:string" />
</xs:attributeGroup>

<xs:complexType name="adresseType">
    <xs:sequence>
    <xs:element … />
    <xs:element … />
   </xs:sequence>
    <xs:attributeGroup ref="attributsAdresse" />
</xs:complexType>
```

Ainsi on défini tout les attributs que devra porter une instance de "adresseType" dans un seul groupe.

Déclaration de type anonymes

Un élément (ou un attribut) peut directement définir son type dans son contenu.

Ainsi

```
<xs:element name="adresse" type="adresseType" />
<xs:complexType name="adresseType">
      <xs:sequence>
          <xs:element name="rue" type="xs:string" />
          <xs:element name="ville" type="xs:string" />
     </xs:sequence>
      <xs:attribute name="format" type="xs:string" />
</xs:complexType>
```

Peut aussi s'écrire

```
<xs:element name="adresse" >
<xs:complexType>
    <xs:sequence>
       <xs:element name="rue" type="xs:string" />
       <xs:element name="ville" type="xs:string" />
     <xs:sequence>
     <xs:attribute name="format" type="xs:string" />
</xs:complexType>
</xs:element>
```

Lien du XML vers le schéma

Un XML référence son schéma en mettant les attributs suivants sur sa racine :

```
<?xml version="1.0"?>

<racine
        xmlns="http://www.mondomaine.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mondomaine.com schema.xsd">

    ....

</racine>
```

Où **http://www.mondomaine.com** représente le domaine associé à votre XML
Et **schema.xsd** est le nom du document contenant le schéma XML

Références

XML Schema Tutorial

- http://www.w3schools.com/schema/