62-31.3 XML

XML-02-P01-Schemas_XSD

Cédric Benoit

Plan du cours

- Présentation et organisation
- Introduction à XML
- **Schémas XML**
  - Schéma DTD
  - Schéma XSD

Points abordés

- Limitation des DTD
- Apports des schémas
- Premiers pas
- Contraintes d'occurrences
- Déclaration et référencement
- Types simples / Listes / Unions
- Types complexes
- XML ⇔ XSD

Limitation des DTD

- Fichiers DTD pas au format XML
- Ne supporte pas les espaces de noms
- Typage extrêmement limité
- Historiquement plutôt destiné au SGML

Schéma XML (XML Schema)

- Schéma décrit la structure d'un document XML
- Le langage du schéma XML est également appelé XML Schéma Definition (XSD)
- Recommandation du W3C depuis 2001
- D'autres genres de schémas également en plus du DTD et XSD

Apports des schémas (1)

Schémas XML (XSD) permet :

- Le typage des données ce qui permet la gestion de booléens, d'entiers, d'intervalles de temps...
- La création de nouveaux types à partir de types existants
- L'héritage

Apports des schémas (2)

- Le support des espaces de nom
- Les indicateurs d'occurrences des éléments peuvent être tout nombre non négatif
  - Dans une DTD, on est limité à 0, 1 ou un nombre infini d'occurrences pour un élément
- Les schémas sont très facilement concevables par modules
- De définir la structure des éléments (imbrication)

Apports des schémas (3)

Un schéma XML définit:

- Les éléments d'un document
- Les attributs d'un document
- Si un élément est vide ou non
- Le type de données pour un élément ou un attribut
- Les constantes et les valeurs par défaut d'un élément ou d'un attribut

Apports des schémas (4)

Le Schéma XML garantit le transfert des données

- Exemple : format "date"
  - 03-11-2004 peut représenter :
  - le 3 novembre 2004
  - le 11 mars 2004
- <date type="date"> force le format d'une date selon le modèle YYYY-MM-DD

Premiers pas (1)

Comme tout document XML, un schéma XML (XSD) commence par un prologue et a un élément racine.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
    <!-- déclarations d'éléments, d'attributs et de types
    ici -->
</xsd:schema>
```

Premiers pas (2)

L'élément racine est l'élément *xsd:schema*

xmlns:xsd (namespace)

Tout élément d'un schéma doit commencer par le préfixe *xsd*

Ce préfixe est souvent utilisé, mais cela pourrait être aussi un autre préfixe comme *xs, sch,….*

Espace de nommage (namespace)

Chaque auteur peut définir le nom des balises librement

Risque de collision

Idée du namespace: préfixer chaque balise

Définition d'un namespace

xmlsn:<nom> = <uri>

nom = nom du namespace (nom du préfixe)

uri = valeur associée (quelconque, unique)

Espace de nommage (document XML)

```
<subject>Math</subject>

<subject>Thrombosis</subject>

<school:subject>Math</school:subject>

<medical:subject>Thrombosis</medical:subject>
```

Eléments (1)

Un élément dans un schéma, se déclare avec la balise <xsd:element>

```
<xsd:element name="contacts"
             type="typeContacts"/>
<xsd:element name="remarque"
             type="xsd:string"/>
```

Eléments (2)

- Le schéma précédent déclare deux éléments:
  - un élément "contacts"
  - un élément "remarque"
  - Chaque élément est « typé »
  - Le type typeContacts est un type complexe défini par l'utilisateur
  - Le type xsd:string est un type simple

Eléments (3)

Les éléments pouvant contenir des éléments-enfants ou posséder des attributs sont dits de ***type complexe***

```
<branche>
        <feuille genre="carré">valeur</feuille>
        <feuille genre="rond">valeur</feuille>
<branche>
```

Les éléments n'en contenant pas sont dits de ***type simple***

Attributs

Un attribut ne peut être que de type simple (string,….)

Ne peut contenir d'autres éléments ou attributs

Les déclarations d'attributs doivent être placées après les définitions des types complexes, autrement dit, après les éléments xsd:sequence, xsd:choice et xsd:all

Attributs – Exemple (1)

Déclaration d'un attribut *maj* de type *xsd:date*

Type simple qui indique la date de la dernière mise à jour de la liste des contacts

Attributs – Exemple (1)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<!-- déclarations de types ici -->
<xsd:complexType name="typeContacts">
<!-- déclarations du modèle de contenu ici: éléments xsd:sequence,... -->
<xsd:attribute name="maj" type="xsd:date" />
</xsd:complexType>
....
<xsd:element name="contacts" type="typeContacts"/>
<xsd:element name="remarque" type="xsd:string">
...
</xsd:schema>
```

Contraintes d'occurrences

- Un attribut peut avoir un indicateur d'occurrences
- L'élément attribute d'un Schéma XML peut avoir trois attributs optionnels :
  - use: obligatoire, optionnel,…
  - default: valeur par défut
  - fixed: valeur fixe
- Les combinaisons de ces trois attributs permettent de paramétrer ce qui est acceptable ou non dans le fichier XML final

Contraintes d'occurrences – Exemple (1)

- Par exemple, la ligne suivante permet de rendre l'attribut *maj* optionnel, avec une valeur par défaut au 11 octobre 2003 si aucune n'est saisie
- Le format de la date est standardisé au format anglo-saxon année-mois-jour
- Ce format de date permet de classe plus facilement les dates

```
<xsd:attribute name="maj" type="xsd:date"
use="optional" default="2003-10-11"/>
```

Contraintes d'occurrences – Exemple (2)

Il est à noter que la valeur de l'attribut default doit être conforme au type déclaré. Par exemple...

```
<xsd:attribute name="maj"
               type="xsd:date"
               use="optional"
               default="-43"/>
```

... produit une erreur à la validation du schéma.

Texte + attributs

- Un tel élément est de type complexe, car il contient au moins un attribut
- Afin de spécifier qu'il peut contenir également du texte, on utilise l'attribut mixed de l'élément *xsd:complexType*
- Par défaut, mixed="false". Il faut dans ce cas forcer mixed="true"

Texte + attributs - Exemple

```
<xsd:element name="elem">
   <xsd:complexType mixed="true">
      <xsd:attribute name="attr" type="xsd:string"
    use="optional"/>
    </xsd:complexType>
</xsd:element>
```

Déclaration et référencement

- La déclaration d'éléments peut amener à une structure de type « poupée russe »
- Il est recommandé de commencer par déclarer les éléments et attributs de type simple, puis ceux de type complexe
- On peut en effet faire référence, dans une déclaration de type complexe, à un élément de type simple préalablement défini

Exemple – Difficile à maintenir

```
<xsd:element name="livre">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="auteur" type="xsd:string" />
       <xsd:element name="pages" type="xsd:positiveInteger" />
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
```

Exemple – Facile à maintenir

```
<xsd:element name="pages" type="xsd:positiveInteger" />
 <xsd:element name="auteur" type="xsd:string" />
 <xsd:element name="livre">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element ref="auteur" />
       <xsd:element ref="pages" />
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
```

Types de données

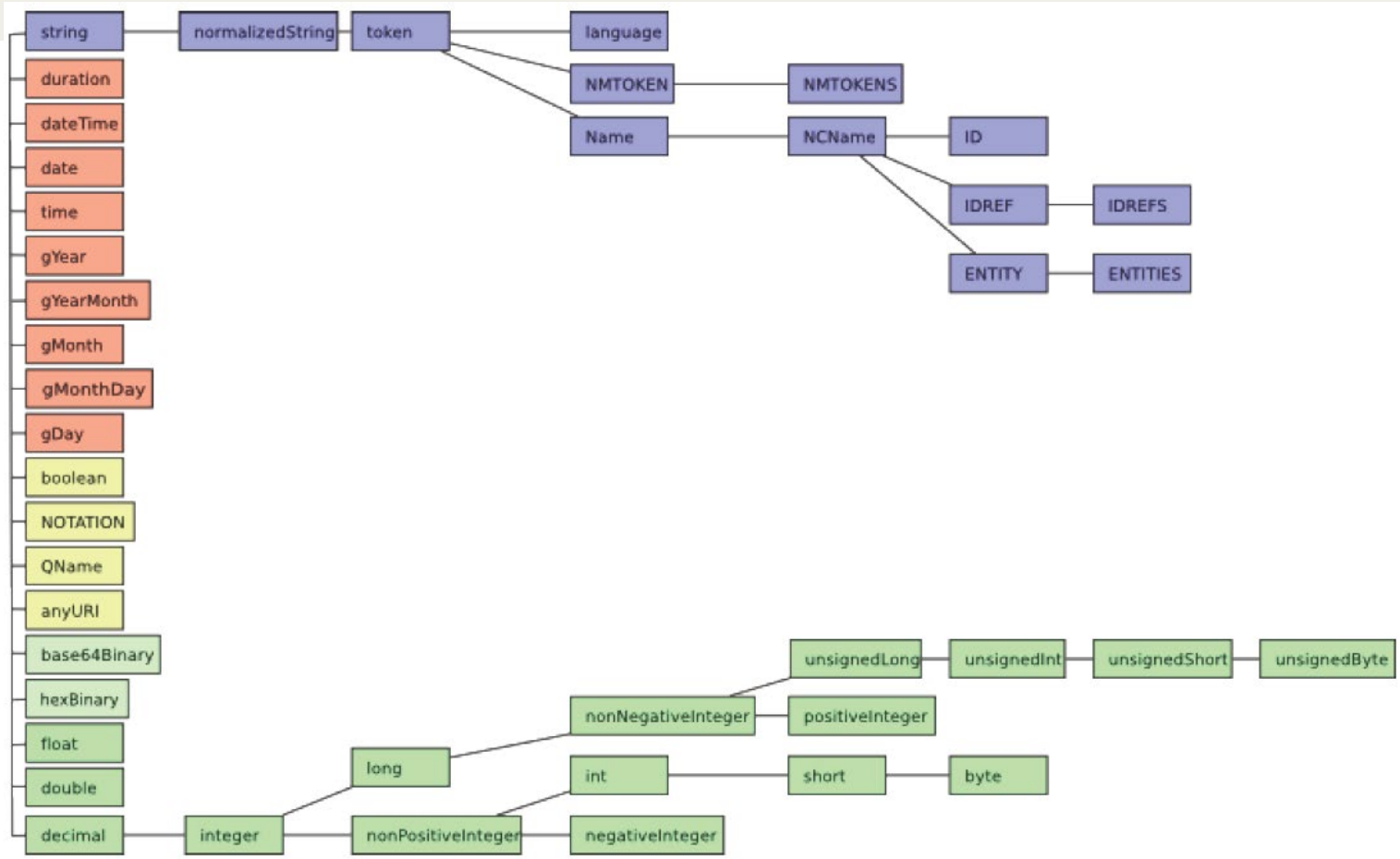
Types simples: contient des données → feuille de l'arbre

Types complexes: contient des données enrichies ou des structures d'éléments → branches ou tronc de l'arbre

Types simples

- Les types de données simples ne peuvent comporter
  - ni attributs
  - ni éléments enfants
- Il existe de nombreux types prédéfinis, mais il est également possible d'en "dériver" de nouveaux (type utilisateur)
- Il est possible de déclarer des "listes" de types

Types prédéfinis



Type prédéfinis - Exemple

```
<xsd:attribute
    name="quantite" type="xsd:positiveInteger"
    use="default"
    value="1"/>
```

Type listes

Les types listes sont des suites de types simples.

Il est également possible de créer une liste personnalisée, par « dérivation » de types existants

Type listes - Exemple

Par exemple:

```
<xsd:simpleType name="numeroDeTelephone">
        <xsd:list itemType="xsd:unsignedByte"/>
</xsd:simpleType>
<xsd:element name="telephone"
                type="numeroDeTelephone"/>
```

Un élément conforme à cette déclaration serait

```
<telephone>01 44 27 60 11</telephone>
```

Unions

- Les types listes et les types simples intégrés ne permettent pas de choisir le type de contenu d'un élément
- On peut désirer, par exemple, qu'un type autorise soit un nombre, soit une chaîne de caractères particuliers
- Il est possible de le faire à l'aide d'une déclaration d'union.

Unions - Exemple

```
<xsd:simpleType
  name="numéroDeTelephoneMnemoTechnique">
  <xsd:union memberTypes="xsd:string
  numéroDeTéléphone"/>
</xsd:simpleType>

<telephone>18</telephone>
<telephone>Pompiers</telephone>
```

Types complexes

- Un élément de type simple ne peut pas contenir de sous-élément
- Il est nécessaire pour cela de le déclarer de type complexe.
- On peut alors déclarer **des séquences d'éléments**, **des types de choix** ou **des contraintes d'occurrences**

Séquence d'élément

- On utilise pour ce faire l'élément ***xsd:sequence***, qui reproduit l'opérateur "," du langage DTD.

Séquence d'élément - Exemple

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

est équivalent à une déclaration d'élément, dans une DTD, où apparaîtrait (nom, prénom, dateDeNaissance,adresse, adresseElectronique, téléphone)

Choix d'élément

- On peut vouloir modifier la déclaration de type précédente en stipulant qu'on doive indiquer soit l'adresse d'une personne, soit son adresse électronique
- Pour cela, il suffit d'utiliser un élément *xsd:choice* qui a les mêmes effets que l'opérateur **|** dans une DTD

Choix d'élément - Exemple

```
<xsd:complexType name="typePersonne">
    <sequence>
        <xsd:element name="nom" type="xsd:string" />
        <xsd:element name="prénom" type="xsd:string" />
<xsd:element name="dateDeNaissance" type="xsd:date" />
<xsd:choice>
  <xsd:element name="adresse" type="xsd:string" />
  <xsd:element name="adresseElectronique" type="xsd:string" />
</xsd:choice>
    </sequence>
    <xsd:element name="téléphone" type="numéroDeTéléphone" />
</xsd:complexType>
```

Elément all (1)

- Cet élément est une nouveauté par rapport aux DTD
- Il indique que les éléments enfants peuvent apparaître une fois ou pas du tout (équivalent de l'opérateur ? dans une DTD)
- Mais dans n'importe quel ordre

Elément all (2)

- Cet élément ***xsd:all*** doit être un enfant direct de l'élément ***xsd:complexType*** (comme pour ***xsd:sequence*** et ***xsd:choice***)

- Ce mécanisme peut être reproduit aussi avec les indicateurs d'occurrences

Elément all - Exemple

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

Indicateurs d'occurrences (1)

- Dans une DTD, un indicateur d'occurrence ne peut prendre que les valeurs 0, 1 ou l'infini (?, + ou *)
- On peut forcer un élément *elem* à être présent 378 fois, mais il faut pour cela écrire (elem, elem..., elem, elem) 378 fois
- XML Schema permet de déclarer directement ce genre occurrence, car tout nombre entier non négatif peut être utilisé

Indicateurs d'occurrences (2)

- Les attributs utiles sont *minOccurs* et *maxOccurs*, qui indiquent respectivement les nombres minimal et maximal de fois où un élément peut apparaître

- Pour déclarer qu'un élément peut être présent un nombre illimité de fois, on utilise la valeur *unbounded*

Indicateurs d'occurrences (3)

Dans une DTD	Valeur de minOccurs	Valeur de maxOccurs
*	0	unbounded
+	1 (pas nécessaire, valeur par défaut)	unbounded
?	0	1 (pas nécessaire, valeur par défaut)
rien	1 (pas nécessaire, valeur par défaut)	1 (pas nécessaire, valeur par défaut)
impossible	nombre entier n quelconque	nombre entier m quelconque supérieur ou égal à n

Indicateurs d'occurrences (1) – Exemple

```
<xs:attribute name="categorie" type="xs:string" default="Adulte"/>
<xs:attribute name="unitePrix" type="xs:string" default="CHF"/>
...
<xs:element name="Prix">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:double">
                <xs:attribute ref="categorie"/>
                <xs:attribute ref="unitePrix"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
...
```

Indicateurs d'occurrences (2) – Exemple

```
...
<xs:element name="GrillePrix">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="Prix" minOccurs="1"
    maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="id" use="optional"/>
    </xs:complexType>
</xs:element>
```

Lien xml <-> xsd

- Dans le cas d'une référence locale, correspondant à une XSD de type SYSTEM, on fait référence au schéma dans le document XML en utilisant l'attribut *noNamespaceSchemaLocation*

```
<biblio
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="biblio10.xsd">
```

Exemple (fichier .xsd)

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

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Exemple (fichier .xml)

```
<?xml version="1.0"?>
<note
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Merci pour votre attention !