XML Schema

Code: xml-schema

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/xml-schema/xml-schema.html**

url: **http://tecfa.unige.ch/guides/tie/pdf/files/xml-schema.pdf**

Auteurs et version

- Daniel K. Schneider

- Version: 0.5 (modifié le 29/10/09)

Prérequis

Module technique précédent: **xml-tech**

Abstract

- Petite introduction à XML Schema

Objectifs

- Edition de fichiers XML avec un schéma XSD
- Traduction de DTDs vers XSD avec un outil

- Modification de types de données dans un XSD
- Création de grammaires XSD simples (!)

Ressources

- The W3C XML Schema primer: **http://www.w3.org/TR/xmlschema-0/**
- Roger Costello's extensive XML Schema tutorial: **http://www.xfront.com/**

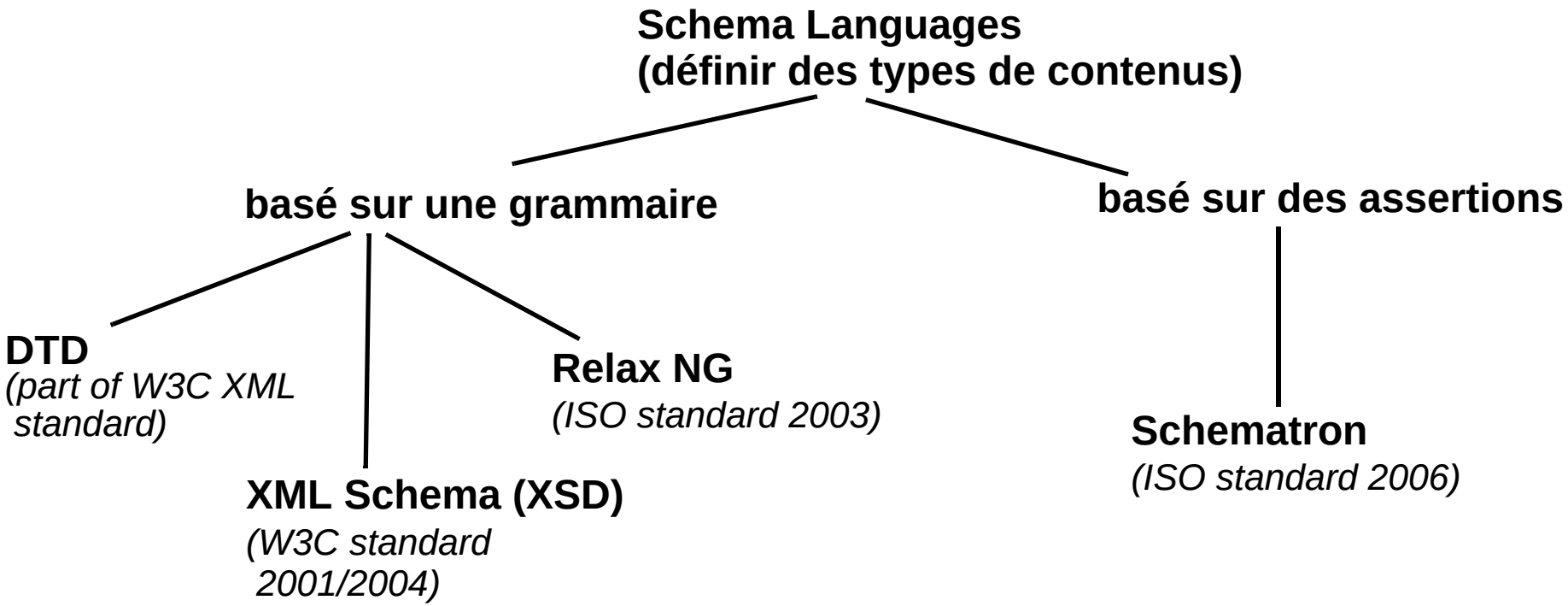
Exemples

- Vous retrouverez les fichiers exemple de ce texte ici:
- **http://tecfa.unige.ch/guides/xml/examples/xsd-examples/**

1. Table des matières détaillée

2. Introduction

2.1 Types de grammaires XML (Schema languages)



- Basé sur un grammaire:
  - Eléments autorisés dans un document XML, leur ordre, occurrences, etc...
  - Contenu et type de données pour chaque élément et attribut.
- Basé sur des assertions:
  - Assertions sur la nature des relations entre éléments and attributs dans un document XML.

2.2 Comparaison de schémas basés sur une grammaire

Caractéristiques	DTD	XML Schema (XSD)	Relax NG
Adoption	large	Applications data-centric	R&D, mais aussi qq. standards
Complexité structurale	moyenne	puissant (e.g. sets, element occurrence constraints)	puissant
Types de données	peu (10, surtout des valeurs d'attributs)	puissant (44 + plus types dérivables)	puissant
Complexité globale	basse	élevée	moyenne
Formalisme en XML	non	oui	oui (notation courte aussi)
Association avec un document XML	DOCTYPE declaration	Namespace declaration	Pas de solution standard
Support navigateur	IE (Firefox non)	non	non
File suffix	*.dtd	*.xsd	*.rng / *.rnc
Entités	oui	non (xinclude)	non

2.3 Ressources

- XML Schema (aussi appelé XSD pour "XML Schema Definition") est difficile
- Une bonne façon d'apprendre XSD est de traduire une DTD avec un outil
- Voir 7. "Traduire des DTDs vers XSD" [31]

W3C websites:

url: **http://www.w3.org/XML/Schema (W3C Overview Page)**

url: **http://www.w3.org/TR/xmlschema-0/ The W3C XML Schema primer**

Specifications:

url: **http://www.w3.org/TR/xmlschema-1/ XML Schema Part 1: Structures Second Edition 2004**

url: **http://www.w3.org/TR/xmlschema-2/ XML Schema Part 2: Datatypes Second Edition 2004**

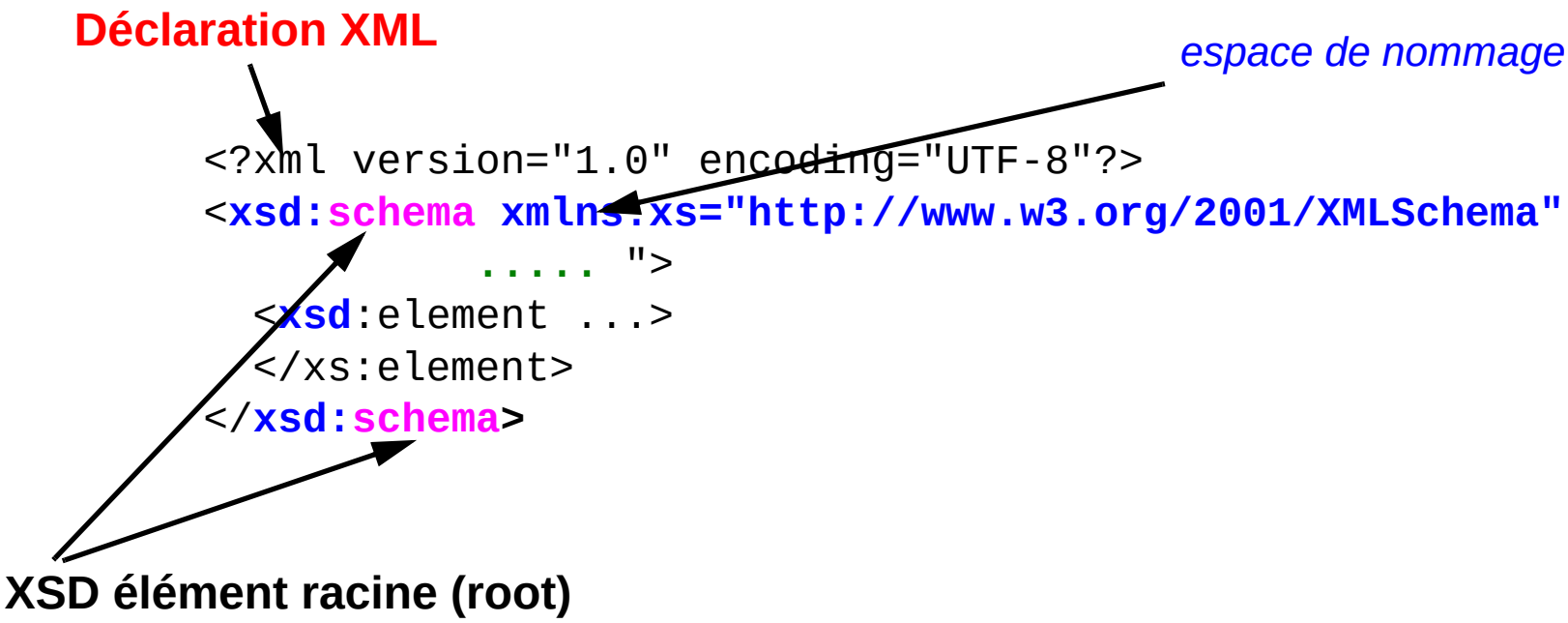
Outils:

- Exchanger XML Editor gère XML Schema
  - Support pour l'édition XML
  - Validation d'une fichier XSD
  - Validation d'un XML avec des XSD
  - Traduction DTD/XSD/Relax NG

3. XSD - structure du fichier et espaces de nom

3.1 Structure et espace de nom d'un fichier XSD

- Comme tout fichier XML, XSD doit commencer par une déclaration XML
- Racine d'un XSD : <**schema> ... </schema>**
- L'élément **schema** possède des attributs (voir plus loin)
- XSD utilise des espaces de noms pour distinguer éléments appartenant à XSD (le langage) et les éléments et attributs définis par un schéma donné (même principe que XSLT).



Espaces de noms (namespaces) et préfixes

- On peut **soit** définir un préfixe pour les éléments XSD **soit** pour vos éléments
  - Voir solution 1 et 2 ci-dessous
- Vous pouvez aussi choisir si vos éléments XML auront un namespace

A. Solution 1: Donner un namespace au code XSD

- Souvent on utilise le préfixe **`xs:`** pour le code XSD
  - Parfois **`xsd:`** cela n'a pas d'importance
- **`elementFormDefault="qualified"`** veut dire que vos balises n'auront pas de namespace

Example 3-1: XSD définition pour une simple recette de cuisine

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Simple recipe Schema -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">
  <xs:element name="list">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="recipe"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

- Cette solution est préférable en règle générale (mais voir plus loin comment associer une XSD à un fichier XML: il faut encore ajouter des attributs)

B. Solution 2: Donner un namespace au code du schéma

- Les éléments définis pour votre schéma ont un préfixe dans la définition.
  - vous devez donc définir votre namespace (substituer "yourdomain.org/namespace")
- On déclare que XML Schema a le namespace par défaut, c.a.d. les éléments XSD ne seront pas préfixés.

Example 3-2: XSD définition pour une simple recette

```
<schema
  xmlns='http://www.w3.org/2000/10/XMLSchema'
  targetNamespace='http://yourdomain.org/namespace/'
  xmlns:t='http://yourdomain.org/namespace/'>

 <element name='t:list'>
  <complexType>
   <sequence>
    <element ref='t:recipe' maxOccurs='unbounded'/>
   </sequence>
  </complexType>
 </element>
```

3.2 Validation

- Un document XML décrit par un XSD est appelé **instance document**.
- Dans XML Exchanger, clic sur l'icone validation, ensuite sélectionner le fichier XSD

A. Association d'un XSD avec un fichier XML, Solution 1

- Il faut déclarer le namespace `xsi:` (**XMLSchema-instance)**
- L'attribut ***xsi:noNamespaceSchemaLocation*** définit l'URL de votre XSD
- Attention: il faut utiliser cela tel quel !!!
- Je déconseille cette solution (voir la solution 2)

XML file (http://tecfa.unige.ch/guides/xml/examples/xsd-examples/recipe-no-ns.xml)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<list
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="recipe-no-ns.xsd">
  <recipe> ....
</list>
```

XSD file (http://tecfa.unige.ch/guides/xml/examples/xsd-examples/recipe-no-ns.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">
  <xs:element name="list">
```

B. Association d'un XSD avec XML, Solution 2

Solution à retenir: L'idée est que chaque fragment XML fait tjrs partie d'un namespace:

1. Les fichiers XML et XSD doivent inclure une **namespace declaration pour un domaine**

   Le fichier XML doit inclure en plus:

2. une déclaration pour le **XMLSchema-instance namespace**
3. un attribut **xsi:schemaLocation** qui dit ou trouver XSD
   - Cet attribut peut contenir plusieurs pairs "namespace-URL"

Example 3-3: XML pour une recette avec une XSD associée (file recipe.xml)

XML file (http://tecfa.unige.ch/guides/xml/examples/xsd-examples/recipe.xml)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<list
  xmlns="http://myrecipes.org/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://myrecipes.org/ recipe.xsd" >
  <recipe>
    <meta> .....</meta>
    ......
  </recipe>
</list>
```

En gros: Il faut substituer qc. pour le rose et le rouge ci-dessus. Faites attention à la syntaxe de l'attribut schemaLocation: "`Namespace_url XSD_URL NameSpace_URL XSD_URL` ..."

XSD file (http://tecfa.unige.ch/guides/xml/examples/xsd-examples/recipe.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Simple recipe Schema -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://myrecipes.org/"
           xmlns="http://myrecipes.org/"
           elementFormDefault="qualified">

   ....
</xs:schema>
```

- La XSD définit un namespace pour vos balises.
- Il faut substituer http://myrecipes.org/" par un URL de votre choix, mais de préférence un URL sur lequel vous avez le contrôle (par exemple votre home page).

Example 3-4: IMS Content Packaging 1.1.4 et IMS/LOM Metadata

url: **C.f. http://tecfa.unige.ch/guides/tie/html/pedago-normes/pedago-normes.html**

Le fichier XML utilisera deux espaces de nommage

```
<manifest
  xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
  xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  identifier="MANIFEST-1"
  xsi:schemaLocation=
   "http://www.imsglobal.org/xsd/imscp_v1p
    imscp_v1p1.xsd
    http://www.imsglobal.org/xsd/imsmd_v1p2
    imsmd_v1p2p2.xsd">

 <metadata>
    <imsmd:lom> ......
   </imsmd:lom>
 </metadata>
 <organizations default="learning_sequence_1">
.....
```

- Cet exemple montre comment utiliser deux espaces de nom pour deux XSD
- imscp_v1p1 est le namespace par défaut (sans préfixe)
- imsmd_v1p1 est le namespace pour les métadonnées.

Extrait du fichier ims_cp_rootv1p1.xsd

```
<xsd:schema
     xmlns = "http://www.imsglobal.org/xsd/imscp_v1p1"
     targetNamespace = "http://www.imsglobal.org/xsd/imscp_v1p1"
     xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
     version = "IMS CP 1.1.4"
     elementFormDefault = "qualified">
```

3.3 Définition d'éléments

- Une structure XML est hiérarchique ...

<xs:element>

- Définit un élément

  Exemple d'un simple élément sans enfant et sans attributs:

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

Définition d'éléments enfants

- On peut les définir de deux façons:
  - soit avec un élément enfant complexType
  - soit avec un attribut "type" qui se réfère à une définition complexType

<xs:complexType> (1)

- enfant de xs:element

```
<xs:element name="recipe">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="meta"/>
        <xs:element minOccurs="0" ref="recipe_author"/>
        <xs:element ref="recipe_name"/>
        <xs:element ref="ingredients"/>
        <xs:element ref="directions"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

<xs:complexType> (2)

- Alternativement, on déclare un complex type seul et ensuite on l'utilise dans des déclarations d'éléments.

  url: **http://tecfa.unige.ch/guides/xml/examples/xsd-examples/recipe2.xsd**

- Référence à un type CSD

```
<xs:element name="recipe" type="recipe_contents" />

<xs:complexType name="recipe_contents">
    <xs:sequence>
      <xs:element ref="meta"/>
      <xs:element minOccurs="0" ref="recipe_author"/>
      <xs:element ref="recipe_name"/>
      <xs:element ref="meal"/>
      <xs:element ref="ingredients"/>
      <xs:element ref="directions"/>
    </xs:sequence>
 </xs:complexType>
```

4. Types de données

Définir ce que éléments ou attributs peuvent contenir. Ces types sont dans le même namespace que les balises XSD.

4.1 Types simples

Exemples:

Simple Type	Exemples (séparés par des virgules)	Explication
string	Confirm this is electric	A text string
base64Binary	GpM7	Base86 encoded binary data
hexBinary	0FB7	HEX encoded binary data
integer	...-1, 0, 1, ...	
positiveInteger	1, 2, ...	
negativeInteger	... -2, -1	
nonNegativeInteger	0, 1, 2, ...	
long	-9223372036854775808, ... -1, 0, 1, ... 9223372036854775807	
decimal	-1.23, 0, 123.4, 1000.00	
float	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN	

Simple Type	Exemples (séparés par des virgules)	Explication
boolean	true, false, 1, 0	
duration	P1Y2M3DT10H30M12.3S	1 year, 2 months, 3 days, 10 hours, 30 minutes, and 12.3 seconds
dataTime	1999-05-31T13:20:00.000-05:00	May 31st 1999 at 1.20pm Eastern Standard Time
date	1999-05-31	
time	13:20:00.000, 13:20:00.000-05:00	
gYear	1999	
Name	shipTo	XML 1.0 Name type
QName	po:USAddress	XML Namespace QName
anyURI	http://www.example.com/	
language	en-GB, en-US, fr	valid values for xml:lang as defined in XML 1.0

Exemple d’un élément:

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

Exemple d’un attribut (voir plus loin):

```
<xsd:attributeGroup name = "est_visible">
    <xsd:attribute name = "est_visible" type = "xsd:boolean"/>
</xsd:attributeGroup>
```

4.2 Types simples définis par l'utilisateur

Example 4-1: Exemple "list":

XSD:

```
<xsd:element name="listOfMyInt" type="listOfMyIntType"/>
<xsd:simpleType name="listOfMyIntType">
    <xsd:list itemType="xsd:integer"/>
</xsd:simpleType>
```

XML:

```
<listOfMyInt>20003 15037 95977 95945</listOfMyInt>
```

Example 4-2: "restriction" sur un mot à choisir

XSD:

```
<xsd:element name="theorie" type="liste_theories"/>
<xsd:simpleType name="liste_theories">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="constructivisme"/>
        <xsd:enumeration value="behavioriste"/>
        <xsd:enumeration value="cognitiviste"/>
    </xsd:restriction>
</xsd:simpleType>
```

XML:

```
<theorie>constructivisme</theorie>
```

Example 4-3: Restriction sur un nombre

- Le type est défini comme enfant ici.

XSD:

```
<xs:element name="age">

 <xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="120"/>
  </xs:restriction>
 </xs:simpleType>

</xs:element>
```

XML:

```
<age>100</age>
```

5. Organisation d'éléments

- Ici on montre qqs. design patterns pour définir des contraintes structurelles...

5.1 Références vs. insertion directe

- On conseille de définir tous les éléments à plat et ensuite y référer

Eléments dans éléments (déconseillé)

```
<xs:element name="meta">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="author" type="xs:string"/>
        <xs:element name="version" type="xs:string"/>
        <xs:element name="date" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

Définition d'éléments avec une référence (mieux)

- Voir Example 5-1: "Une liste d'enfants ordonnées" [23]

```
<xs:sequence>
        <xs:element ref="author"/>
        .....
</xs:sequence>
```

5.2 Séquences

- Les attributs minOccurs and maxOccurs définissent le nombre min/max. d'un enfant.

Example 5-1: Une liste d'enfants ordonnées

```
<xs:element name="meta">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="author"/>
      <xs:element ref="date"/>
      <xs:element ref="version"/>
    </xs:sequence>
  </xs:complexType>

<xs:element name="version" type="xs:string"/>
<xs:element name="date" type="xs:string"/>
<xs:element name="author" type="xs:string"/>
```

Example 5-2: Une liste avec un ou plusieurs éléments

```
<xs:element name="list">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="recipe"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example 5-3: Une liste avec un élément email à option et répétable

```
<xs:element name="person">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="name"/>
       <xs:element minOccurs="0" maxOccurs="unbounded" ref="email"/>
       <xs:element ref="link"/>
     </xs:sequence>
     <xs:attributeGroup ref="attlist.person"/>
   </xs:complexType>
 </xs:element>
```

5.3 Choix

Example 5-4: Choix d'éléments à option et répétable

```
<xs:element name="INFOS">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="date"/>
      <xs:element ref="author"/>
      <xs:element ref="a"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Example 5-5: Choix obligatoire entre éléments

```
<xs:element name="ATTEMPT">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="action"/>
      <xs:element ref="EPISODE"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

5.4 Mixed contents

```
<xs:element name="para">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="strong"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="strong" type="xs:string"/>
```

5.5 Empty elements (éléments vides)

- Définir un élément "vide" consiste à définir un élément sans enfants

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

- Cela s'applique aussi aux "complex elements":
  - Voir Example 6-1: "Groupes d'attributs (file family.xsd)" [28]

6. Attributs

- Les déclarations d'attributs sont compliquées en XSD, c.a.d. on définit un complexType.
- Le paramètre ***use***: optional, prohibited or required
  - défaut est "optional"

Voici quelques examples

```
<xs:element name="Name">
 <xs:complexType>
   <xs:attribute name="lang" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

Même chose, mais en plus long:

```
<xs:element name="Name">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
     <xs:attribute name="lang" type="xs:string" use="required"/>
   </xs:extension
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

Groupes d'attributs

- Utilisé pour déclarer des attributs plus complexes
- Les groupes sont réutilisables, c.a.d c'est l'équivalent entités paramétriques pour les DTD.

Example 6-1: Groupes d'attributs (file family.xsd)

url: http://tecfa.unige.ch/guides/xml/examples/xsd-examples/family.xsd

```
<xs:element name="person">
    <xs:complexType>
      <xs:attributeGroup ref="attlist.person"/>
    </xs:complexType>
</xs:element>
```

La définition ci-dessus se réfère au groupe d'attributs défini ci-dessous:

```
<xs:attributeGroup name="attlist.person">
    <xs:attribute name="name" use="required"/>
    <xs:attribute name="gender">
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="male"/>
          <xs:enumeration value="female"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>

<!-- cont. à la page suivante ... -->
```

```
    <xs:attribute name="type" default="mother">
       <xs:simpleType>
         <xs:restriction base="xs:token">
           <xs:enumeration value="mother"/>
           <xs:enumeration value="father"/>
           <xs:enumeration value="boy"/>
           <xs:enumeration value="girl"/>
         </xs:restriction>
       </xs:simpleType>
     </xs:attribute>
     <xs:attribute name="id" use="required" type="xs:ID"/>
</xs:attributeGroup>
```

Voici un fragment xml valide:

url: **http://tecfa.unige.ch/guides/xml/examples/xsd-examples/family.xml**

```
<family>
  <person name="Joe Miller" gender="male" type="father" id="I123456789"/>
  <person name="Josette Miller" type="girl" id="I123456987"/>
</family>
```

6.1 Contraintes sur les valeurs

- Voici un exemple

Example 6-2: Contraintes sur l'age

```
<xs:element name="age">

<xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="18"/>
    <xs:maxInclusive value="120"/>
  </xs:restriction>
</xs:simpleType>

</xs:element>
```

7. Traduire des DTDs vers XSD

- Ci-dessous qqs. chablons de traduction
- La plupart des éditeurs ont un traducteur
  - le résultat n'est pas forcément très beau ...
  - dans Exchanger XML Editor: Menu Schema -> Convert Schema

7.1 Définition d'éléments

Exemples de http://www.w3.org/2000/04/schema_hack/

DTD	XML Schema

`<!ELEMENT ROOT (A,B) >`	`<element name="ROOT">`
	`<complexType content="elementOnly">`
	`<element ref="t:A">`
	`<element ref="t:B">`
	`</complexType>`
	`<element>`

`<!ELEMENT ROOT (A|B) >`	`<element name="ROOT">`
	`<complexType content="elementOnly">`
	`<choice>`
	`<element ref="t:A">`
	`<element ref="t:B">`
	`</choice>`
	`</complexType>`
	`<element>`

DTD	XML Schema
<!ELEMENT ROOT (A|(B,C)) >	<element name="ROOT"> <complexType content="elementOnly"> <choice> <element ref="t:A"> <sequence> <element ref="t:B"> <element ref="t:C"> </sequence> </choice> </complexType> <element>
<!ELEMENT ROOT (A?,B+,C*) >	<element name="ROOT"> <complexType content="elementOnly"> <element ref="t:A" minOccurs="0"> <element ref="t:B" maxOccurs="unbounded"> <element ref="t:C" minOccurs="0" maxOccurs="unbounded"> </complexType> <element>

7.2 Définition d'attributs

DTD	XML Schema
<!ATTLIST ROOT a CDATA #REQUIRED>	<element name="ROOT"> <complexType content="elementOnly"> <attribute name="a" type="string" use="required"/> </complexType> </element>
<!ATTLIST ROOT a CDATA #IMPLIED>	<element name="ROOT"> <complexType content="elementOnly"> <attribute name="a" type="string" use="optional"/> </complexType> </element>
<!ATTLIST ROOT a (x|y|z)#REQUIRED;>	<element name="ROOT"> <complexType content="elementOnly"> <attribute name="a"> <simpleType base="string"> <enumeration value="x"/> <enumeration value="y"/> <enumeration value="z"/> </simpleType> </attribute> </complexType> </element>

DTD	XML Schema
<!ATTLIST ROOT a CDATA #FIXED "x">	<element name="ROOT"> <complexType content="elementOnly"> <attribute name="a" type="string" use="fixed" value="x"/> </complexType> </element>

7.3 Exemples

Voir: http://tecfa.unige.ch/guides/xml/examples/xsd-examples/