XSchema

- J. Seinturier (CNRS LSIS umr 6168)
- Université du Sud Toulon Var
- Université Paul Cézanne
- http://www.seinturier.fr

Limitation des DTDs

- Pas au format XML. Nécessité d'utiliser un outil pour manipuler un tel fichier, différent de celui utilisé pour l'édition du document XML.
- le « typage » des données est extrêmement limité.
- Ne supportent pas les espaces de nom: Il n'est pas possible d'importer des définitions de balises définies par ailleurs dans un fichier XML défini par une DTD.

Schemas XML

- Typage des données: permet la gestion de booléens, d'entiers, d'intervalles de temps... Extension de types.
- Indicateurs d'occurrences (cardinalités) des éléments par un nombre. (limité à 0, 1 ou l'infini pour une DTD).
- Support des espaces de nom.
- Héritage: Les éléments peuvent hériter du contenu et des attributs d'un autre élément.

Document XSchema

Un schéma XML est lui-même un **document XML**.

- Contient un **prologue** XML standard
- Déclaration du namespace **xsd** contenant les **préfixes** XSchema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</xsd:schema>
```

Recommandation W3C (Mai 2001)

Eléments

Un schéma XML décrit principalement les **éléments** et les **attributs** de documents XML.

Eléments et attributs représentent la **structure** des documents XML.

Déclaration

Un **élément** est déclaré avec la balise **<xsd:element/>**.

La balise **<xsd:element/>** accepte deux attributs:

- L'attribut **name** qui renseigne le nom de l'élément (nom de balise).
- L'attribut **type** qui renseigne le type de l'élément (ce que la balise peut elle-même contenir).

Exemple de déclaration

Le schéma suivant décrit un document composé de 1 élément texte:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="texte" type="xsd:string" />
</xsd:schema>
```

Elément

Type

Un fichier XML décrit par le schéma:

```
<?xml version="1.0" encoding="UTF-8"?>
<texte>
   Voici un texte !!
</texte>
```

Types

Avantage des schémas: typage des données

Les XML schémas permettent de définir 2 types de données:

- **Type simple**: assimilé à un type de base (entier, chaine de caractère, …)

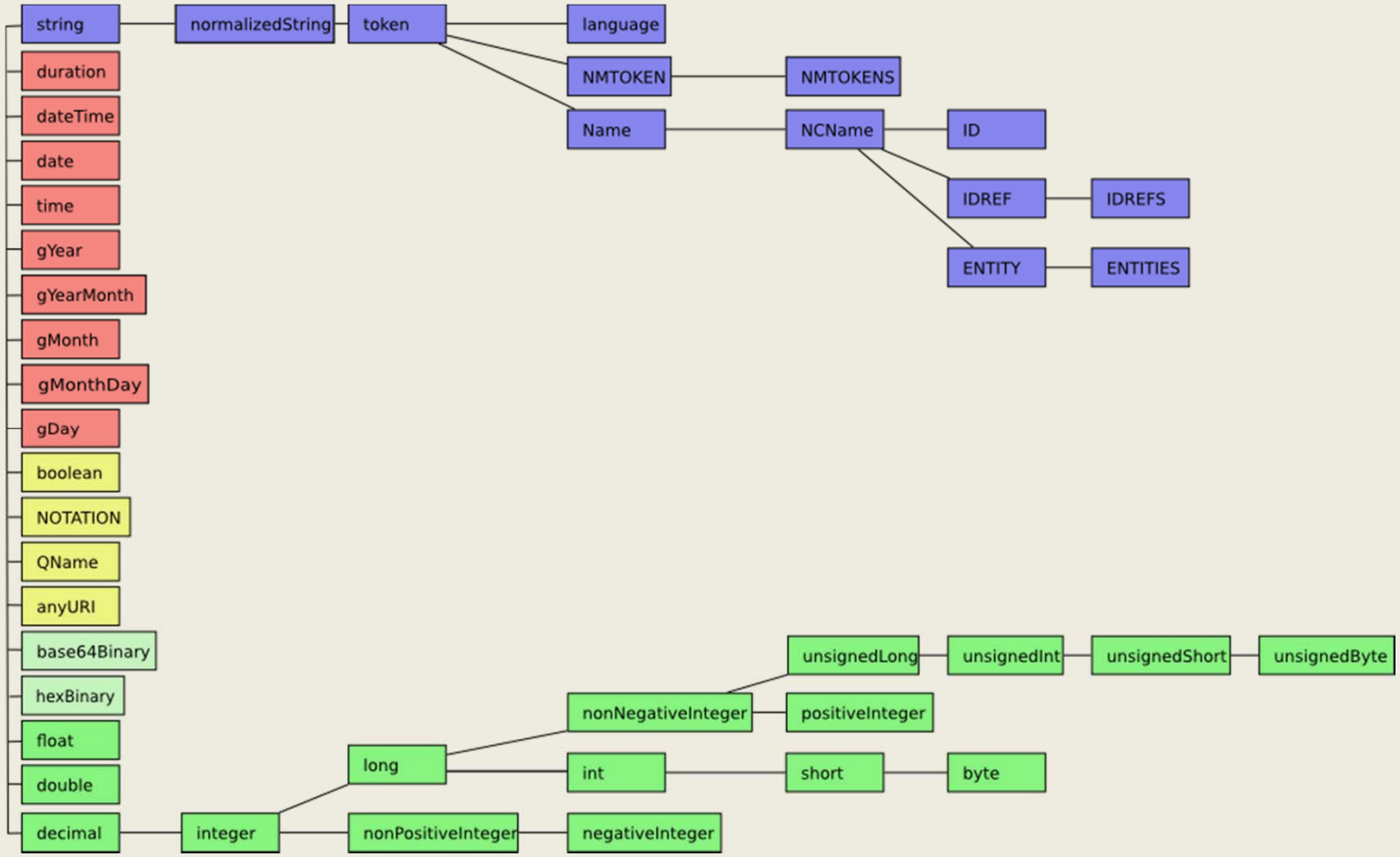
  Un élément de type simple signifie qu'ile **ne peut pas** contenir d'autre élément **ni même d'attribut**.

- **Type complexe**: type lui-même composé d'autres éléments (ou **contenant des attributs**)

  Permet de définir des séquences d'éléments, des ensembles, des cardinalités.

Types simples

Un ensemble de types simples prédéfinis (W3C):



Types simples: exemple

Défini un élément de type simple date

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name= "date" type="xsd:date" />
</xsd:schema>
```

Document XML **valide** selon le schéma précédent:

```
<?xml version="1.0" encoding="UTF-8"?>
<date> 2002-09-24 </date>
```

Document XML **non valide** selon le schéma précédent (*date non standard*):

```
<?xml version="1.0" encoding="UTF-8"?>
<date> 21 Aout 2004 </date>
```

Types simples prédéfinis (1/2)

String : "Du texte"
normalizedString : "Du texte encore"
token : "du texte toujours"
language : en-GB, en-US, fr
ID : "A212" , "B213"
IDREF : "A212"
IDREFS : "A212 B213"
ENTITY, ENTITIES , NOTATION , NMTOKEN, NMTOKENS

base64Binary : GpM7
hexBinary: 0FB7

byte : -1, 126
unsignedByte: 0, 126
positiveInteger : 1, 126789
negativeInteger : -126789, -1
nonNegativeInteger : 0, 1, 126789
nonPositiveInteger : -126789, -1, 0
int : -1, 126789675
unsignedInt : 0, 1267896754
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
integer : -126789, -1, 0, 1, 126789

Types simples prédéfinis (2/2)

boolean : true, false 1, 0
anyURI : http://www.example.com/,
http://www.example.com/doc.html#ID5
NOTATION : voir W3C
QNAME : x:p, xsd:element

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
gYearMonth : 1999-02
gDay : ---31
gMonthDay : --05-31

Restriction de type

Un **type simple** peut être restreint afin de former un **nouveau** type simple (*restriction / facets*).

Une restriction de type est décrite grâce à la balise **<xs:restriction/>**.

Syntaxe de restriction

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:simpleType name="ageType">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="120"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Nom du nouveau type

Type original à restreindre

Liste d'**éléments** restrictions

Contraintes générales / textuelles

enumeration: Défini une liste de valeurs possibles.

length: Nombre **exact** de caractères ou d'item de liste autorisés (≥ 0).

maxLength: Nombre **maximum** de caractères ou d'item de liste autorisés (≥ 0).

minLength: Nombre **minimum** de caractères ou d'item de liste autorisés (≥ 0).

whiteSpace: Spécification du traitement des espaces.

Expression régulière

pattern: Défini la séquence de caractères acceptables en fonction d'une expression régulière.

Exemples de contraintes (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="car">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Audi"/>
    <xsd:enumeration value="Golf"/>
    <xsd:enumeration value="BMW"/>
    <xsd:whiteSpace value="collapse"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
</xsd:schema>
```

Restriction du type prédéfini **string**

Liste de 3 valeurs: *Audi, Golf, BMW*

Ignore les caractères blancs

```
<?xml version="1.0" encoding="UTF-8"?>
<car>Audi</car>
```

Document **valide**

```
<?xml version="1.0" encoding="UTF-8"?>
<car>Citroen</car>
```

Document **non valide**

(Valeur non listée)

Exemples de contraintes (2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="password">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:minLength value="5"/>
    <xsd:maxLength value="8"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
</xsd:schema>
```

→ Entre 5 et 8 caractères alphanumériques

```
<?xml version="1.0" encoding="UTF-8"?>
<password >edfrtER</password>
```

Document **valide**

```
<?xml version="1.0" encoding="UTF-8"?>
<password >  edfrtER  </password>
```

Document **non valide**
(Espaces non gérés)

Exemples de contraintes (3)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="ref">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:pattern value="([a-z][A-Z]\d)+"/>
    <xsd:length value="6"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
</xsd:schema>
```

Répétition du triplet (*Lettre minuscule, lettre majuscule, chiffre*)

6 digits exactement

```
<?xml version="1.0" encoding="UTF-8"?>
<ref>zA1tB5</ref>
```

Document **valide**

```
<?xml version="1.0" encoding="UTF-8"?>
<ref>zA1tB5xY7</ref>
```

Document **non valide**
(trop de caractères)

Contraintes numériques

fractionDigits: Spécifie le nombre maximal de décimales après la virgule.

maxExclusive: Spécifie strictement la borne supérieure de la valeur.

maxInclusive: Spécifie la valeur maximale de la donnée numérique.

minExclusive: Spécifie strictement la borne inférieure de la valeur.

minInclusive: Spécifie la valeur minimale de la donnée numérique.

totalDigits: Spécifie le nombre de digits autorisés.

Exemples de contraintes (4)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="boundedfloat">
  <xsd:simpleType>
   <xsd:restriction base="xsd:float">
    <xsd:maxExclusive value="25.0"/>
    <xsd:minInclusive value="5.0"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
</xsd:schema>
```

→ Nombre réel de l'intervalle [5.0; 25.0[

```
<?xml version="1.0" encoding="UTF-8"?>
<boundedfloat>5.0</boundedfloat>
```

Document **valide**

```
<?xml version="1.0" encoding="UTF-8"?>
<boundedfloat>4.3</boundedfloat>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<boundedfloat>25</boundedfloat>
```

Document **non valide** *(hors bornes)*

Types complexes

Type lui-même composé **d'autres éléments** ou **contenant des attributs**

4 sortes de types complexes:

- **Eléments vides** (*Empty*).
- Eléments composés **uniquement** d'autres **éléments**.
- Eléments composés **uniquement de texte**.
- Eléments composés **à la fois** de **texte** et **d'éléments** (*Mixte*).

Un type complexe est déclaré grâce à la balise **<xs:complexType>**

Sortes de types complexes

Eléments vides (empty):

```
<emptyel value="5"/>
```

Eléments composés **uniquement** d'autres **éléments**:

```
<?xml version="1.0" encoding="UTF-8"?>
<complex>
  <el>hello</el>
  <bal>it's me</bal>
</complex>
```

Eléments composés **uniquement de texte**:

```
<el> blah blah </el>
```

Eléments composés **à la fois** de **texte** et **d'éléments**:

```
<?xml version="1.0" encoding="UTF-8"?>
<mixte>
  Du texte
  <el>Un élément !</el>
</mixte>
```

Elément vide

Pas de mot clé réservé

Ne peut contenir **que des attributs**

Définition d'élément vide

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="empty">
      <xsd:complexType></xsd:complexType>
   </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<empty />
```

Document **valide**

```
<?xml version="1.0" encoding="UTF-8"?>
<empty>1</empty>
```

Document **non valide**

Elément composés d'éléments

Peut contenir **uniquement** d'autres **éléments**

Définis à l'aide d'**indicateurs** (***indicators***)

7 Indicateurs

3 indicateurs d'**ordre** (*Order indicators*)

2 indicateurs de **cardinalité** (*Occurrence indicators*)

2 indicateurs de **groupement** (*Group indicators*)

Indicateurs d'ordre

Spécifient **l'ordre** dans lequel les éléments **doivent apparaitre**

Brique de base de la déclaration d'éléments

3 Indicateurs

All (*Tous*): Les éléments cités doivent **tous** apparaitre **sans ordre précis**

Choice (*Un parmi*): Un seul des éléments cités doit apparaitre

Sequence (*Liste*): Les éléments cités doivent **tous** apparaitre **dans l'ordre de déclaration**

All (Tous)

Les éléments cités doivent **tous** apparaitre **sans ordre précis**

Déclaration xml: **<xsd:all>**

Exemple (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="person">
  <xsd:complexType>
   <xsd:all>
    <xsd:element name="firstname" type="xsd:string"/>
    <xsd:element name="lastname" type="xsd:string"/>
   </xsd:all>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Exemple (2/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="person">
  <xsd:complexType>
   <xsd:all>
    <xsd:element name="firstname" type="xsd:string"/>
    <xsd:element name="lastname" type="xsd:string"/>
   </xsd:all>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person >
   <firstname>Howard</firstname>
   <lastname>Bulot</lastname>
</person>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person >
   <lastname>Bulot</lastname>
   <firstname>Howard</firstname>
</person>
```

Documents valides:
tous les éléments apparaissent.

```
<?xml version="1.0" encoding="UTF-8"?>
<person >
   <firstname>Howard</firstname>
</person>
```

Document invalide: tous les éléments doivent apparaitre.

Choice (Un parmi)

Un seul des éléments cités doit apparaitre

Déclaration xml: **<xsd:choice>**

Exemple (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="menu">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element name="entree" type="xsd:string"/>
    <xsd:element name="dessert" type="xsd:string"/>
   </xsd:choice>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Exemple (2/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
   <entree>Salade</entree>
</menu>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<menu >
   <dessert>Pomme</dessert>
</menu>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="menu">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element name="entree" type="xsd:string"/>
    <xsd:element name="dessert" type="xsd:string"/>
   </xsd:choice>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Documents valides:
Un seul élément apparait à la fois.

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
   <entree>Jambon sec</entree>
   <dessert>Gateau</dessert>
</menu>
```

Document invalide: deux éléments apparaissent.

Sequence (Liste)

Les éléments doivent **tous** apparaitre **dans l'ordre de leur déclaration**

Déclaration xml: **<xsd:sequence>**

Exemple (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="person">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="firstname" type="xsd:string"/>
    <xsd:element name="lastname" type="xsd:string"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Exemple (2/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="person">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="firstname" type="xsd:string"/>
    <xsd:element name="lastname" type="xsd:string"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person >
   <firstname>Howard</firstname>
   <lastname>Bulot</lastname>
</person>
```

Document valide:
tous les éléments apparaissent dans l'ordre.

Document invalide: tous les éléments doivent apparaitre.

```
<?xml version="1.0" encoding="UTF-8"?>
<person >
   <firstname>Howard</firstname>
</person>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person >
   <lastname>Bulot</lastname>
   <firstname>Howard</firstname>
</person>
```

Document invalide: l'ordre doit être respecté.

Indicateurs de cardinalité

Spécifient **le nombre d'occurrence** d'un même élément

Combinaison avec les indicateurs d'ordres

2 Indicateurs

minOccurs (*nombre minimal d'apparition*): Les éléments cités doivent apparaitre **au moins ce nombre de fois**

maxOccurs (*nombre maximal d'apparition*): Les éléments cités ne **doivent pas** apparaitre **plus que cette cardinalité**

minOccurs

Nombre **minimal** d'apparition d'un élément

Déclaration xml: **attribut minOccurs** de la balise **<xsd:element>**

Exemple (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="parent">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="child" type="xsd:string" minOccurs="1"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Exemple (2/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="parent">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="child" type="xsd:string" minOccurs="1"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<parent>
   <name>Alfred</name>
   <child>Jean</child>
</parent>
```

Document valide:
child apparait au moins une fois.

```
<?xml version="1.0" encoding="UTF-8"?>
<parent>
   <name>Alfred</name>
</parent>
```

Document invalide: child doit apparaitre au moins une fois.

maxOccurs

Nombre **maximal** d'apparition d'un élément

Déclaration xml: **attribut maxOccurs** de la balise **<xsd:element>**

Exemple (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="child">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="parent" type="xsd:string" maxOccurs="2"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Exemple (2/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<child>
   <name>Jean</name>
   <parent>Alfred</parent>
   <parent>Zoe</parent>
</child>
```

Document valide:
parent apparait deux fois.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="child">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
  <xsd:element name="parent" type="xsd:string" maxOccurs="2"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<child>
   <name>Jean</name>
   <parent>Alfred</parent>
   <parent>Zoe</parent>
   <parent>Dino</parent>
</child>
```

Document invalide: parent doit apparaitre au plus deux fois.

Type anonyme

Permet l'utilisation d'éléments **non décrits** par le schéma

Déclaration xml: **<xsd:any>**

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

Document valide:
Ajout de l'élément children.

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
 <person>
  <firstname>Hege</firstname>
  <lastname>Refsnes</lastname>
  <children>
   <childname>Cecilie</childname>
  </children>
 </person>
 <person>
  <firstname>Stale</firstname>
  <lastname>Refsnes</lastname>
 </person>
</persons>
```

Attributs

Un **attribut** est déclaré avec la balise **<xsd:attribute/>**.

La balise **<xsd:attribute/>** accepte **cinq** attributs:

- L'attribut **name** qui renseigne le nom de l'attribut.
- L'attribut **type** qui renseigne le type de l'attribut (un attribut ne peut bien sur qu'être de **type simple**).
- L'attribut **default** qui donne une valeur par défaut à l'attribut.
- L'attribut **fixed** qui donne à l'attribut a une valeur constante.
- L'attribut **use** qui indique si l'attribut est optionnel ou non (''required'').

Un **attribut** ne peut être déclaré que dans un **type complexe**

La **déclaration** des attributs dans un type complexe se fait **en dernier**

Exemple (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xs:element name="person">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="desc" type="xs:string"/>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" />
   <xs:attribute name= "age" type="xs:integer" use="required"/>
  </xs:complexType>
 </xs:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
 <person name="Holmes" age="46">
  <desc>Detective</desc>
 </person>
</persons>
```

Document valide

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
 <person name="Bulot">
  <desc>Detective</desc>
 </person>
</persons>
```

Document non valide: manque l'attribut age

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
 <person name="Bulot" age="46 ans">
  <desc>Detective</desc>
 </person>
</persons>
```

Document non valide: mauvais type de l'attribut age

Exemple (2/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xs:element name="person">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="desc" type="xs:string"/>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" fixed="Bob"/>
   <xs:attribute name= "age" type="xs:integer" default="0"/>
  </xs:complexType>
 </xs:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
 <person age="46">
  <desc>Detective</desc>
 </person>
</persons>
```

Document valide

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
 <person name="Bulot" age="46 ans">
  <desc>Detective</desc>
 </person>
</persons>
```

Document non valide: age ne peut être redéfini

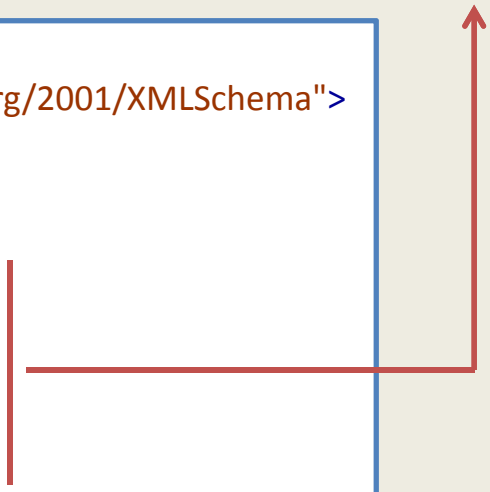
Contraintes sur les attributs

Un **attribut** peut être restreint afin de **contraindre** sa **valeur** (*restriction / facets*).

Une restriction de valeur est décrite grâce à la balise **<xs:restriction/>**.

Les contraintes possibles sont les mêmes que pour un type simple.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xs:element name="person">
  <xs:complexType>
   <xsd:attribute name="age">
    <xsd:simpleType>
     <xsd:restriction base="xsd:integer">
      <xsd:maxExclusive value="150"/>
      <xsd:minInclusive value="0"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
  </xs:complexType>
 </xs:element>
</xsd:schema>
```



Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xs:element name="person">
  <xs:complexType>
   <xsd:attribute name="age">
    <xsd:simpleType>
     <xsd:restriction base="xsd:integer">
      <xsd:maxExclusive value="150"/>
      <xsd:minInclusive value="0"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
  </xs:complexType>
 </xs:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person age="230"/>
```

Document non valide: age ne peut avoir de valeur supérieure à 150

```
<?xml version="1.0" encoding="UTF-8"?>
<person age="35"/>
```

Document valide

Héritage

XSchema rend possible l'**héritage** entre types simples ou complexes

2 types d'héritage

- Héritage par restriction (vu p. 12)
- Héritage par extension: **<xsd:extension>**

Attribut base: spécifie le **type** à étendre

Restriction: Ajout de contraintes

Extension: Ajout d'informations

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="size">
 <xs:restriction base="xs:string">
  <xs:enumeration value="small" />
  <xs:enumeration value="medium" />
  <xs:enumeration value="large" />
 </xs:restriction>
</xs:simpleType>
<xs:complexType name="jeans">
 <xs:simpleContent>
  <xs:extension base="size">
   <xs:attribute name="sex">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="male" />
      <xs:enumeration value="female" />
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
</xs:schema>
```

Clés (key)

Permet de **référencer** des éléments de manière **unique**

Déclaration: **<xsd:key>**

Une clé peut être un **attribut** ou un **élément:**

- **Unique selon sa portée**
- **Non nul (ou non vide)**
- **Toujours présent**

L'élément **<xsd:key>** doit obligatoirement contenir dans l'ordre:

- **Un et seul** sélecteur de portée (**<xsd:selector>**)
- **Au moins un** indicateur de champ (**<xsd:field>**)

Sélecteur de portée (selector)

Indique dans quelle **partie du document** la clé doit être unique

Défini grâce à une expression **XPath**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="service">
  <xs:complexType>
   <xs:sequence maxOccurs="unbounded">
    <xs:element name="employe">
     <xs:complexType>
      <xs:attribute name="nom" type="xs:string"/>
      <xs:attribute name="id" type="xs:token"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="cle">
   <xs:selector xpath="employe"/>
   <xs:field xpath="@id"/>
  </xs:key>
 </xs:element>
</xs:schema>
```

L'unicité de la clé s'applique aux employés

Identification des champs (field)

Indique quels attributs ou éléments forment la clé

Défini grâce à une expression **XPath**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="service">
  <xs:complexType>
   <xs:sequence maxOccurs="unbounded">
    <xs:element name="employe">
     <xs:complexType>
      <xs:attribute name="nom" type="xs:string"/>
      <xs:attribute name="id" type="xs:token"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="cle">
   <xs:selector xpath="employe"/>
   <xs:field xpath="@id"/>
  </xs:key>
 </xs:element>
</xs:schema>
```

L'attribut id de l'élément personne doit être unique

Références (keyref)

Permet de **référencer** des éléments ayant une clé

Déclaration: **<xsd:keyref>**

- Attribut **refer** spécifie la **clé** (key) à **référencer**

L'élément **<xsd:keyref>** doit obligatoirement contenir dans l'ordre:

- **Un et seul** sélecteur de portée (**<xsd:selector>**)
- **Au moins un** indicateur de champ (**<xsd:field>**)

```
<xs:key name="chef" refer="cle">
  <xs:selector xpath="employe"/>
  <xs:field xpath="@chef"/>
</xs:key>
```

Exemple

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="service">
  <xs:complexType>
   <xs:sequence maxOccurs="unbounded">
    <xs:element name="employe">
     <xs:complexType>
      <xs:attribute name="nom" type="xs:string"/>
      <xs:attribute name="id" type="xs:token"/>
      <xs:attribute name="chef" type="xs:token"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>

  <xs:key name="cle">
   <xs:selector xpath="employe"/>
   <xs:field xpath="@id"/>
  </xs:key>

  <xs:keyref name="cleref" refer="cle">
   <xs:selector xpath="employe"/>
   <xs:field xpath="@chef"/>
  </xs:keyref>
 </xs:element>

</xs:schema>
```