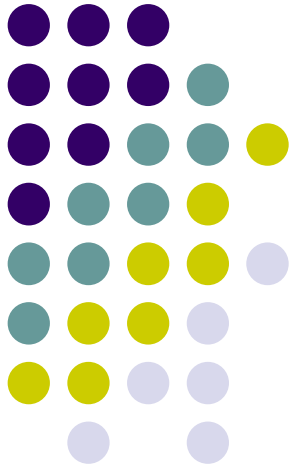
XML-Schéma

Une nouvelle syntaxe pour décrire la grammaire d'un document XML



Insuffisance des DTD

- Syntaxe spécifique (pas XML) qui est difficile à interpréter.
- Trop limitées pour représenter des structures complexes (Pas de notion d'héritage).
- Pas de mécanismes pour permettre l'extension et l'évolution.
- Expression de cardinalités limitée ('?', '*' et '+').
- Pas de types de données à part du texte (#PCDATA).

XML Schéma: présentation

- Pour pallier aux limites des DTD, le consortium W3C a recommandé XML schéma en Mai 2001.
- Un schéma XML d'un document définit:
  - les éléments possibles dans le document
  - les attributs associés à ces éléments
  - la structure du document et les types de données
- Le schéma est spécifié en XML
  - Pas obligé d'apprendre une nouvelle syntaxe pour décrire sa grammaire
- Présente de nombreux avantages
  - structures de données avec types de données
  - extensibilité par héritage et ouverture
  - représentation en arbre
- Les schémas XML jouent donc exactement le même rôle que les DTD dans une application XML, mais avec des possibilités plus grandes de structuration.

Objectifs des schémas

- Reprendre les acquis des DTD
  - Plus riche et complet que les DTD
- Permettre de typer les données
  - Eléments simples et complexes
  - Attributs simples
- Permettre de définir des contraintes
  - Existence, obligatoire, optionnel
  - Domaines, cardinalités, références
  - Patterns, ...

XML Schéma: la structure

XML Schema décrit (en XML) la structure d'un document XML c'est à dire :

- Les éléments qui composent un document.
- Les attributs.
- La hiérarchie entre les éléments.
- L'ordre des sous-éléments.
- Le nombre de sous-éléments.
- Les types des éléments et attributs.
- Les valeurs par défaut, le format ou la restriction des valeurs d'un élément ou d'un attribut.

Les composants primaires

- **Un schéma XML est construit par assemblage de différents composants** (13 sortes de composants rassemblés en différentes catégories).
- **Composants de définition de types**
  - Définition de types **simples (Simple type).**
  - Définition de types **complexes (Complex type).**
- **Composants de déclaration**
  - **Déclaration d'éléments.**
  - **Déclaration d'attributs.**

Syntaxe des XML Schemas

- Les schémas sont des documents XML (d'extension ".xsd") . Ils comportent donc un **prologue**.
- Ils utilisent les espace de nom (namespace). La notion d'espace de nom permet à un document XML quelconque d'utiliser les balises définies dans un schéma donné
- Le préfixe xsd ou xs est utilisé.

```
<?xml version="1.0" encoding="ISO-8859-1">
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!-- contenu -->
</xsd:schema>
```

Cela signifie que l'*espace de nom* auquel ces balises font référence, là où elles sont définies, est un schéma particulier que l'on peut consulter.

Syntaxe des XML Schemas

Référence au schéma

- L'appel à un schéma dans le fichier de données se fera dans l'élément racine.
- On utilise pour ce faire le préfixe xmlns.
- On fait référence au schéma dans le document XML en utilisant l'attribut SchemaLocation

```
<?xml version="1.0" encoding="ISO-8859-1">
<elementRacine
   xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
   xsd:SchemaLocation="lien_relatif_vers_le_schema">
<!-- contenu -->
</elementRacine>
```

Déclaration des éléments

- **Un élément XML est déclaré par la balise ‘element’** de XML schéma qui a de nombreux attributs.

Syntaxe

```
<xsd:element name="nomElement" type="xsd:nomType"/>
```

- Les deux principaux attributs sont:
  - **name** : Le nom de l’élément (de la balise associée).
  - **type** : Le type qui peut être simple ou complexe.

Exemples

```
<xsd:element name="age" type="xsd:decimal"/>
<xsd:element name="nom"  type="xsd:string" />
```

Déclaration des éléments

- **Un élément XML** possède d'autres attributs.

- Déclaration d'une **valeur par défaut** :

<xs:element name="code_postal" type="xs:string" **default**="22000"/>

- Déclaration d'une **valeur figée** :
- <xs:element name="pays" type="xs:string" **fixed**="Algerie"/>

Déclaration des attributs

- **Un attribut** est une valeur nommée et typée associée à un élément.
- **Le type** d'un attribut défini en XML schéma est obligatoirement simple.

Syntaxe

<xsd:attribute **name**="nomAttribut" **type**="xsd:nomType"/>

Exemples

<xsd:attribute name= "Date_creation" type="xsd:date"/>

Autres attributs

- **L'élément attribute de XML Schema peut avoir deux attributs optionnels** : **use** et **value**.
- **On peut ainsi définir des contraintes** de présence et de valeur.
- Selon ces deux attributs, la valeur peut:

  être obligatoire ou non

  être définie ou non par défaut.

Exemple:

```
<xsd:attribute name= "Date_peremption" type="xsd:date"
   use="default" value= "2005-12-31"/>
```

Valeurs possibles pour use

- **Use= default** : Si l'attribut à une valeur définie il la prend sinon il prend la valeur par défaut.
- **Use = required** : L'attribut doit apparaître et prendre la valeur fixée si elle est définie.
- **Use = optional** : L'attribut peut apparaître et prendre une valeur quelconque.
- **Use= fixed** : La valeur de l'attribut est obligatoirement la valeur définie.

Exemples :

Rendre un **attribut obligatoire** :

```
<xs:attribute name="email" type="xs:string" use="required"/>
```

Rendre un **attribut facultatif** :

```
<xs:attribute name="Tel" type="xs:string" use="optional"/>
```

Types complexes

- **Déclarés au moyen de l'élément**

  <xsd:complexType name="...

- **Ils peuvent contenir d'autres éléments**, des attributs.

- **Trois façons de composer des éléments** dans un type complexe: sequence, choice, all.

Types complexes: Sequence

- Un type **sequence** est défini par une suite de sous-éléments qui doivent être présents dans l'ordre donné.

```
<xsd:complexType name="typePersonne">
    <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="email" type="xsd:string"/>
        <xsd:element name="téléphone" type="numéroDeTéléphone"/>
    </xsd:sequence>
</xsd:complexType>
```

Types complexes: Choice

⌘ Un seul des éléments listés doit être présent.

```
<xsd:complexType name="typePersonne">
    <xsd: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="email" type="xsd:string"/>
        </xsd:choice>
        <xsd:element name="téléphone" type="numéroDeTéléphone"/>
    </xsd:sequence>
</xsd:complexType>
```

Types complexes: All

- Les éléments listés doivent être tous présents au plus une fois.
- Il peuvent apparaître dans n'importe quel ordre.

```
<xsd:complexType name=" LivreType " >
    <xsd:all>
        <xsd:element name="titre" type="xsd:string"/>
        <xsd:element name="auteur" type="xsd:string"/>
        <xsd:element name="année" type="xsd:string"/>
        <xsd:element name="éditeur" type="xsd:string"/>
    </xsd:all>
</xsd:complexType>
```

Déclaration et référencement

- 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.

```
<!-- Définition globale de l'élément title -->
<xsd:element name="title" type=" xsd:string"/>
...
<!-- Définition d'un type -->
<xsd:complexType ... >
    ...
    <!-- Utilisation de l'élément title -->
    <xsd:element ref="title"/>
    ...
</xsd:complexType>
```

Déclaration et référencement

```
<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>
```

... Peut être déclaré comme suit...

```
<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>
```

Indicateurs d'occurrences

- Les attributs **minOccurs** et **maxOccurs**, indiquent respectivement les nombres minimal et maximal de fois où un élément peut apparaître.
- Ils sont l'équivalent des opérateurs ?, * et + des DTD.
- Ils peuvent apparaître comme attribut des éléments xsd:element, xsd:sequence, xsd:choice et xsd:all.
- L'attribut minOccurs prend un entier comme valeur.
- L'attribut maxOccurs prend un entier ou la chaîne **unbounded** comme valeur pour indiquer qu'il n'y a pas de nombre maximal.
- La valeur par défaut de ces deux attributs est la valeur 1.

Indicateurs d'occurrences

- L'utilisation des attributs minOccurs et maxOccurs est illustrée par l'équivalent en schéma de ce fragment de DTD

- <!ELEMENT elem (elem1, elem2?, elem3*) >

```
<xsd:element name="elem">
    <xsd:complexType>
      <xsd:sequence>
       <xsd:element ref="elem1"/>
       <xsd:element ref="elem2" minOccurs="0"/>
       <xsd:element ref="elem3" minOccurs="0"  maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Equivalences DTD - XML Schéma

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

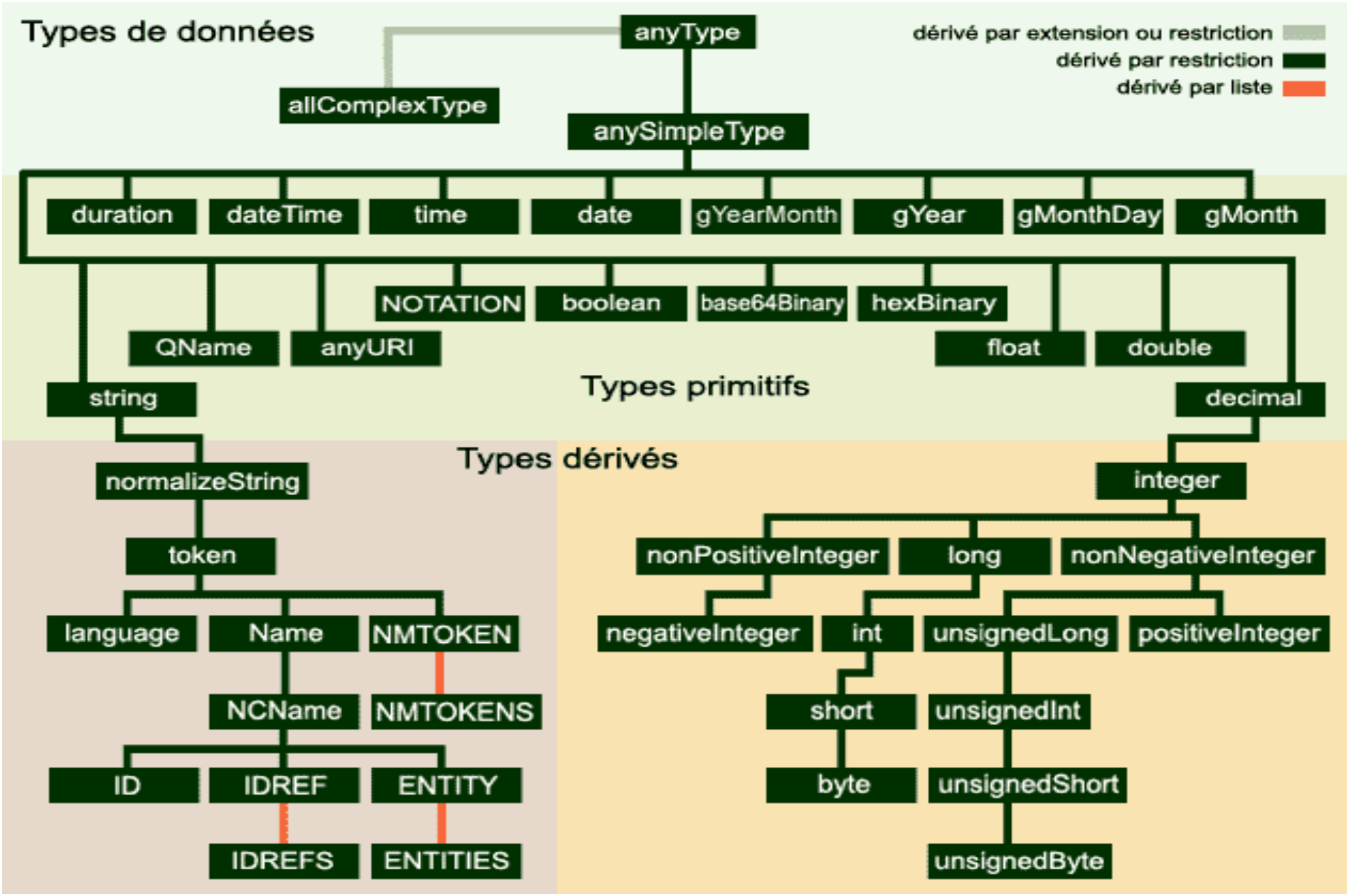
Les types de données XML schéma

Objectifs de la définition des types

- **Fournir des types primitifs** analogues à ceux qui existent en pascal ou en Java.

- **Permettre de créer des types de données usagers** dérivés de types existants en contraignant certaines propriétés (domaine, précision, longueur, format).

Hiérarchie des types prédéfinis



Types simples (prédéfinis)

- Un élément est de **type simple** s'il **ne contient ni d'autres éléments ni d'attributs**.
- Les types simples sont prédéfinis ou dérivés de types prédéfinis.
- **Types simples prédéfinis**

Type	Forme lexicale
String	**Bonjour**
boolean	{**true**, **false, 1, 0**}
float	**2345E3**
decimal	**808.1**
dateTime	**2010-11-24T10:20:00-05:00.**
binary	**0100**
uriReference	**http://www.univ-sba.dz**

Dérivation de nouveaux types

C’est l’action de définir un type en partant de la définition d'un ou de plusieurs types.

- les types simples peuvent être dérivés par
  - restriction (ajout de nouvelles contraintes)
  - union (union de l'espace lexicaux des types membres)
  - liste
- Les types complexes sont dérivés par
  - restriction
  - extension

1- Dérivation par restriction

- La dérivation par **restriction** restreint l'ensemble des valeurs d'un type pré existant.
- La restriction est définie par des contraintes de **facettes** du type de base:
  - lenght : la longueur d'une donnée.
  - minLenght: la longueur minimum.
  - maxLenght: la longueur maximum.
  - pattern: défini par une expression régulière.
  - enumeration: un ensemble discret de valeurs.
  - maxInclusive: une valeur max comprise.
  - maxExclusive: une valeur max exclue.
  - minInclusive: une valeur min comprise.
  - maxInclusive: une valeur min exclue.
  - ...

Exemple d'une énumération

```
<xsd:simpleType name="JourDeSemaine">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="dimanche" />
        <xsd:enumeration value="lundi" />
        <xsd:enumeration value="mardi" />
        <xsd:enumeration value="mercredi" />
        <xsd:enumeration value="jeudi" />
        <xsd:enumeration value="vendredi" />
        <xsd:enumeration value="samedi" />
    </xsd:restriction>
</xsd:simpleType>
```

Exemple de minInclusive/ maxExclusive

- Contrôler que l'élément "age" est de type entier et qu'il est compris entre 0 et 120.

```
<xsd:simpleType name="IntervalAge">
        <xsd:restriction base="xs:integer">
            <xsd:minInclusive value="0"/>
            <xds:maxInclusive value="120"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:element name="age"  type ="IntervalAge"/>
```

Exemple de minLength

- Permet de définir la longueur maximum (minimum) mesurée en nombre de caractères

```
<xs:simpleType name="longName">
    <xs:restriction base="xs:string">
        <xs:minLength value="6"/>
    </xs:restriction>
</xs:simpleType>
```

Exemple de patterns

- Définit un motif auquel doit correspondre la chaîne de caractères

```
<xs:simpleType name="Nom_alpha">
    <xs:restriction base="xs:string">
        <xs:pattern value="([A-Z]([a-z]*) ?)+"/>
    </xs:restriction>
</xs:simpleType>
```

Cas des types entiers

- S'applique à
  - xs:byte, xs:int, xs:integer, xs:long,
  - xs:negativeInteger, xs:nonNegativeInteger,
  - xs:nonPositiveInteger, xs:positiveInteger,
  - xs:short, xs:unsignedByte, xs:unsignedInt,
  - xs:unsignedLong, xs:unsignedShort

Exemple de totalDigits

```
<xs:simpleType name="MaxChiffre">
  <xs:restriction base="xs:integer">
    <xs:totalDigits value="5"/>
  </xs:restriction>
</xs:simpleType>
```

Cette facette n'autorise que les entiers ayant au max 5 chiffres

Exemple de fractionDigits

```
<xs:simpleType name=« ApresVirgule">
 <xs:restriction base="xs:decimal">
   <xs:fractionDigits value="2"/>
 </xs:restriction>
</xs:simpleType>
```

Cette facette spécifie le nombre de chiffres après la virgule.

2 - Dérivation par extension

Dériver un nouveau type par **extension** consiste à ajouter à un type existant des sous-éléments ou des attributs.

```
<xsd:complexType name=" LivreType " >
    <xsd:all>
        <xsd:element name="titre" type="xsd:string"/>
        <xsd:element name="auteur" type="xsd:string"/>
        <xsd:element name="année" type="xsd:string"/>
        <xsd:element name="éditeur" type="xsd:string"/>
    </xsd:all>
</xsd:complexType>
```

⌘ **Exemple: eLivreType** ajoute un élément URI à un type LivreType

```
<xsd:complexType name=" eLivreType " >
    <xsd:complexContent>
        <xsd:extension base='LivreType'>
            <xsd:sequence>
                <xsd:element name='URI' type='uriReference'/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

3 - Dérivation par union

- Pour créer un nouveau type on effectue l'union ensembliste de toutes les valeurs possibles de différents types existants.

- Exemple:

```
<xsd:simpleType name="maDate">
    <xsd:union memberTypes="string date"/>
</xsd:simpleType>
```

Sommaire

Sommaire de déclaration d'éléments

1. Élément avec contenu Simple.

A) Déclaration d'élément utilisant un type prédéfinit :

```
<xsd:element name="numEtudiant" type="xsd:positiveInteger"/>
```

B) Déclaration d'élément utilisant un simpleType:

```
<xsd:simpleType name="shapes">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="triangle"/>
      <xsd:enumeration value="rectangle"/>
      <xsd:enumeration value="square"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="geometry" type="shapes"/>
```

définition (d'un type simple)

déclaration

C) Une formulation alternative de l'exemple précédant

```
<xsd:element name="geometry">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="triangle"/>
         <xsd:enumeration value="rectangle"/>
         <xsd:enumeration value="square"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

Sommaire de déclaration d'éléments

2. Élément contenant des éléments fils.

A) Définir un élément enfant **inline**:

```
<xsd:element name="Person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="FirstName" type="xsd:string"/>
      <xsd:element name="Surname" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

B) Une formulation alternative de l'exemple précédant est de créer un **complexType** ensuite de l'utiliser

```
<xsd:complexType name="PersonType">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="FirstName" type="xsd:string"/>
      <xsd:element name="Surname" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="Person" type="PersonType"/>
```

Webographie

- ‘XML Schema Part 0: Primer’ W3C
  http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/
- ‘XML Schema Part 1: Structures’ W3C
  http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
- ‘XML Schema Part 2: Datatypes’ W3C
  http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
- XML Schema Requirements http://www.w3.org/TR/1999/NOTE-xml-schema-req-19990215 (Février 1999)
- Introduction aux schémas http://www.w3schools.com/schema
- Les schémas XML Gregory Chazalon, Joséphine Lemoine http://site.voila.fr/xmlschema
- W3C XML Schema, Eric Van Der Vlist, http://www.xml.com/pub/a/2000/11/29/schemas/