

XML-Schéma

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

**Pr. Sidi Mohammed Benslimane
École Supérieure en Informatique
08 Mai 1945 – Sidi Bel Abbès –**

s.benslimane@esi-sba.dz



Limites des DTD



- La DTD n'est pas écrite dans une syntaxe XML.
- Pas de typage: les éléments terminaux contiennent que des chaînes de caractères.
- Expression de cardinalités limitée ('?', '*' et '+'). On ne peut pas dire qu'un élément doit apparaître plus de 3 fois mais toujours moins de 7.
- On ne peut pas contraindre la forme de ces contenus (par exemple, entre 5 et 20 caractères, contenant un signe @, ...).

XML Schéma: présentation



- Pour pallier aux limites des DTD, le consortium W3C a recommandé **XML schéma** en Mai 2001, révisée en 2004.
- Le schéma est spécifié en XML (extension du fichier ".xsd").
 - Pas obligé d'apprendre une nouvelle syntaxe pour décrire sa grammaire
- Le schéma XML joue exactement le même rôle que la DTD, mais il est plus riche et complet que la DTD:
 - Permettre de typer les données
 - Éléments 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 d'occurrence des éléments.
- Les types des éléments et des attributs.
- Les valeurs par défaut, le format ou la restriction des valeurs d'un élément ou d'un attribut.

Syntaxe des XML Schemas



- Les schémas sont des documents XML. Ils comportent donc un **prologue**.
- Ils utilisent les espaces 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.

Syntaxe des XML Schemas

- Le **référencement d'un schéma XML** se fait au niveau de l'élément racine du fichier XML grâce à l'utilisation de deux attributs:
 - **L'espace de noms** à travers le préfixe **xmlns**
 - **L'emplacement** qui permet d'indiquer où se situe le fichier contenant le Schéma XML (**SchemaLocation** ou **noNamespaceSchemaLocation**)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<xsd:schema  
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"  
  xsd:SchemaLocation="chemin_vers_fichier.xsd">  
  <!-- déclarations d'éléments, d'attributs et de types ici -->  
</xsd:schema>
```

Déclaration des éléments

- Un élément XML est déclaré par la balise **<xsd:element>**

Syntaxe

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

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

Exemples

<xsd:element name="nom_de_famille" type="xsd:string" />

<xsd:element name="age" type="xsd:positiveInteger"/>

<xsd:element name="date_inscription" type="xsd:date"/>

Déclaration des éléments

□ 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 d'autres attributs optionnels :

□ **Use:** Indicateur de la façon dont l'attribut est utilisé.

Valeurs possibles pour use

- **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.
- **Use= default** : Si l'attribut a une valeur définie il la prend sinon il prend la valeur par défaut.

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"/>
```

Valeur par **défaut** :

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

Autres attributs



L'élément *attribute* de XML Schema peut avoir d'autres attributs optionnels :

- **Id:** doit être du type ID et unique dans le document contenant cet élément.
- `<xsd:attribute name="code" type="xsd:ID" use="required"/>`

Déclaration d'éléments simples



- Un **élément simple** est un élément qui ne peut contenir qu'une **chaîne de caractères**. (il ne peut pas contenir de sous éléments, ni d'attributs)
- Cette chaîne de caractères peut correspondre à :
 - Des types prédéfinis : `xsd:string`, `xsd:decimal`, `xsd:integer`, `xsd:boolean`, `xsd:date`, etc.
 - Des types de données propres à l'utilisateur.

Déclaration d'éléments simples

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="nom" type="xsd:string" />
  <xsd:element name="contacts" type="typeContacts" />
  <!-- déclarations du type simple ici -->
</xsd:schema>
```

Déclare deux éléments : un élément **nom** et un élément **contacts**.

- L'élément **nom** est du type **xsd:string** qui est un type simple prédéfini de XML Schema.
- L'élément **contacts** du type **typeContacts**, qui est un type défini par l'utilisateur.

Déclaration d'éléments complexes

- Un **élément complexe** est un élément qui peut contenir d'autres éléments ou bien des attributs.
- Il existe 5 types d'éléments complexes :
 - 1- Les éléments vides avec attributs.
 - 2- Les éléments qui contiennent uniquement du texte avec des attributs.
 - 3- Les éléments qui contiennent d'autres éléments.
 - 4- Les éléments qui contiennent du texte et d'autres éléments.
 - 5- Les éléments qui contiennent d'autres éléments et des attributs,
- Dans un schéma XML, un élément complexe se déclare en utilisant la balise **xsd:complexType**

Déclaration d'éléments complexes: éléments vides

Exemple: élément vide avec attribut

```
<xsd:complexType name="personnelInfo">  
<xsd:attribute name="nom" type="xsd:string" />  
<xsd:attribute name="prenom" type="xsd:string" />  
</xsd:complexType>  
<xsd:element name="personne" type="personnelInfo" />
```

```
<xsd:element name="personne">  
<xsd:complexType>  
<xsd:attribute name="nom" type="xsd:string" />  
<xsd:attribute name="prenom" type="xsd:string" />  
</xsd:complexType>  
<xsd:element/>
```

```
<personne nom="mohamed" prenom="Ali" />
```


Déclaration d'éléments complexes:

Élément ne contenant que du texte avec un (des) attribut(s)

- 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"**.

Exemple:

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

```
<voiture marque="Renault"> Clio </voiture>
```

Types complexes



- Il existe 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>
<xsd:element name="Personne" type="typePersonne"/>
```

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>
```

```
<xsd:element name="Personne" type="typePersonne"/>
```

Types complexes: All

- Les éléments listés doivent être tous présents au plus une fois.
- Ils 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:gYear"/>
```

```
    <xsd:element name="éditeur" type="xsd:string"/>
```

```
  </xsd:all>
```

```
</xsd:complexType>
```

```
<xsd:element name="Personne" type="LivreType"/>
```

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:attribute name="ISBN" type="xsd:string"/>`
 `</xsd:complexType>`
 `</xsd:element>`
- *... Peut être déclaré comme suit...*
- `<xsd:element name="pages" type="xsd:positiveInteger" />`
 `<xsd:element name="auteur" type="xsd:string"/>`
- `<xsd:attribute name="ISBN" type="xsd:string"/>`
 `<xsd:element name="/livre">`
 `<xsd:complexType>`
 `<xsd:sequence>`
 `<xsd:element ref="auteur" />`
 `<xsd:element ref="pages" />`
 `</xsd:sequence>`
- `<xsd:attribute ref="ISBN"/>`
 `</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:

Exemple

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

```
<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>
```

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

Indicateurs d'occurrences

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

```
<xsd:element name="elem">
  <xsd:complexType>
    <xsd:sequence minOccurs="1" maxOccurs="unbounded">
      <xsd:element ref="elem1"/>
      <xsd:element ref="elem2"/>
      <xsd:element ref="elem3"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<!ELEMENT elem (elem1, elem2, elem3)+>
```

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).

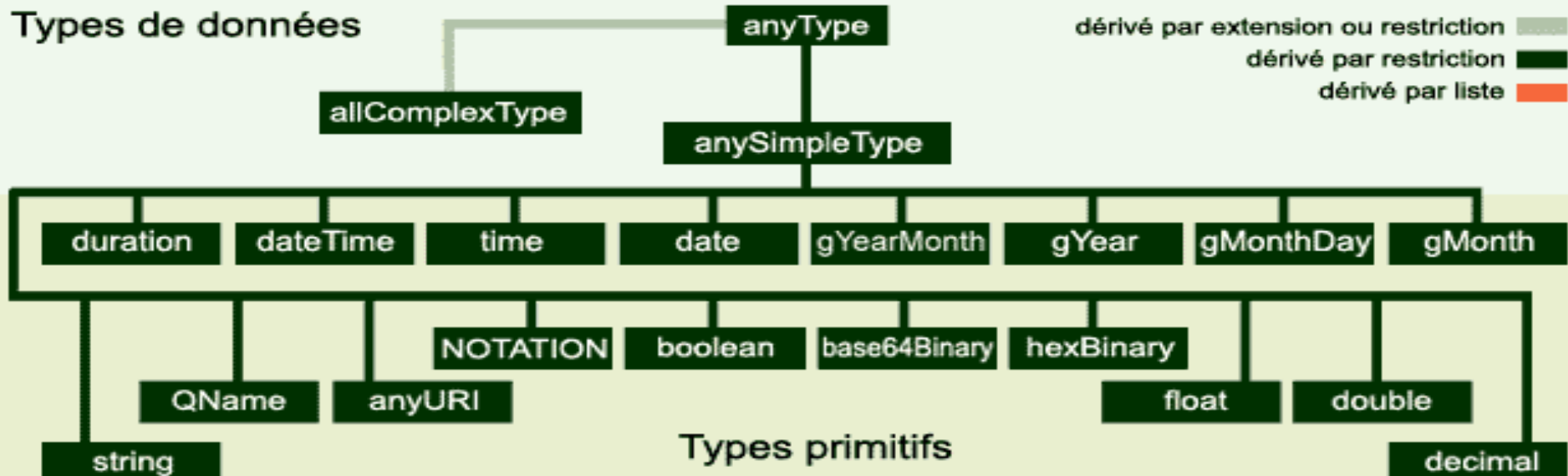
Hiérarchie des types prédéfinis

Types de données

dérivé par extension ou restriction

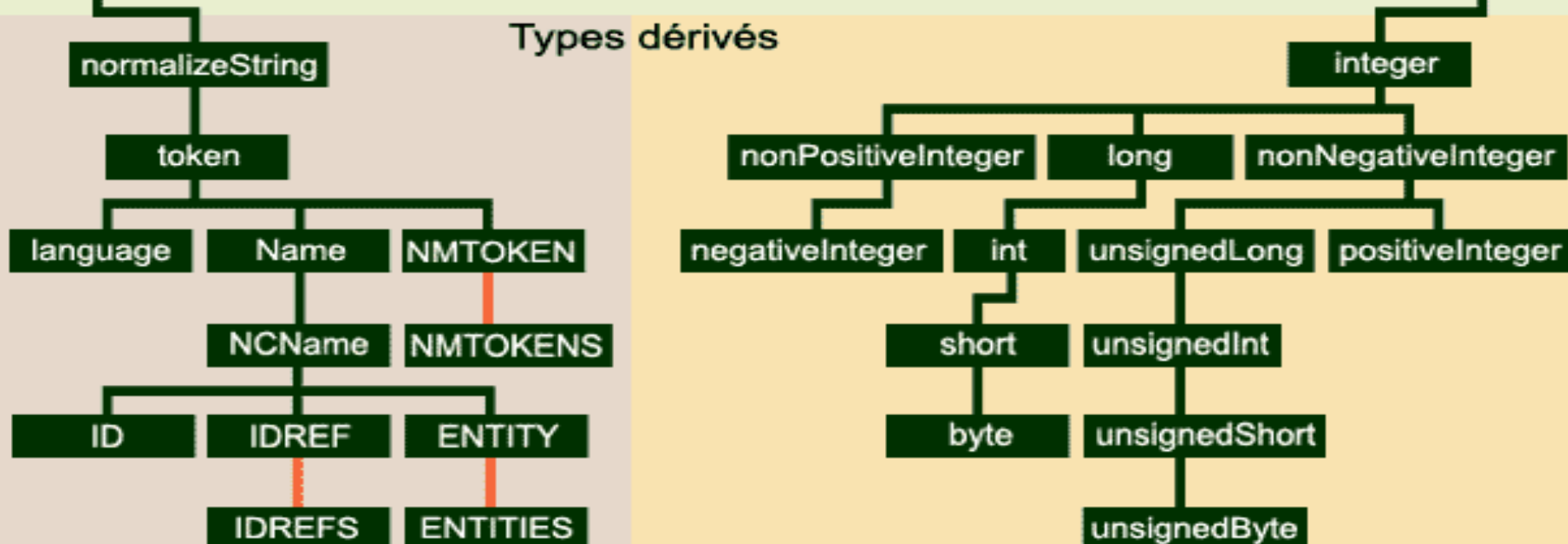
dérivé par restriction

dérivé par liste



Types primitifs

Types dérivés



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	<u>http://www.esi-sba.dz</u>
□ Etc.	

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

□ **Les types complexes** sont dérivés par

- Restriction:
 - Nombre d'occurrences
 - Intervalle de valeurs possibles.
- Extension:
 - Ajout d'éléments à un type complexe.

1- Dérivation par restriction (Type Simple)

- 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.
 - Ect.

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>  
  
<xsd:element name="jour" type="JourDeSemaine"/>
```

Exemple de minInclusive/ maxExclusive

- Contrôler que l'élément "age" est de type entier et qu'il est compris entre 18 et 65.

```
<xsd:simpleType name="IntervalAge">  
  <xsd:restriction base="xsd:integer">  
    <xsd:minInclusive value="18"/>  
    <xsd:maxExclusive value="66"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

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

Exemple de maxLength/minLength

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

```
<xsd:simpleType name="longName">  
  <xsd:restriction base="xsd:string">  
    <xsd:maxLength value="12"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Exemple de patterns

- Définit un motif auquel doit correspondre la chaîne de caractères

```
<xsd:simpleType name="codeZip">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[0-9]{5}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Exemple de patterns

- Définit un motif auquel doit correspondre la chaîne de caractères

```
<xsd:simpleType name="password">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[a-zA-Z0-9]{8}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Cas des types entiers



- S'applique à
 - xs:integer,
 - xs:byte,
 - xs:int,
 - xs:long,
 - xs:negativeInteger,
 - xs:nonNegativeInteger,
 - xs:nonPositiveInteger,
 - xs:positiveInteger,
 - etc.

Exemple de totalDigits



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

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


Exemple de fractionDigits



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

Cette facette spécifie le nombre de chiffres après la virgule.

2 - Dérivation par union (Type Simple)



- 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>
```

3 - Dérivation par extension (Type Complexe)

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>
```

4 - Dérivation par restriction (Type Complexe)

```
<xsd:complexType name="Librairie">  
  <xsd:sequence>  
    <xsd:element name="Livre" type="xsd:string" maxOccurs="unbounded"/>  
  </xsd:sequence>  
</xsd:complexType>
```

```
<xsd:complexType name="petiteLibrairie">  
  <xsd:complexContent>  
    <xsd:restriction base="Librairie">  
      <xsd:sequence>  
        <xsd:element name="Livre" type="xsd:string" maxOccurs="1000" />  
      </xsd:sequence>  
    </xsd:restriction>  
  </xsd:complexContent>  
</xsd:complexType>
```



Sommaire

Sommaire de déclaration d'éléments

1. Élément avec contenu Simple.

A) Déclaration d'élément utilisant un type prédéfini :

```
<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/>

Exercice d'application

Voici une DTD: éléments sans attributs

```
<!ELEMENT Librairie (Livre)+>  
<!ELEMENT Livre (Titre, Auteur+, Annee, Editeur)>  
<!ELEMENT Titre(#PCDATA)>  
<!ELEMENT Auteur(#PCDATA)>  
<!ELEMENT Annee(#PCDATA)>  
<!ELEMENT Editeur(#PCDATA)>
```

- 1- Donner un exemple de document valide.
- 2- Donner le document XML Schema équivalent.

Exercice d'application:

Exemple de document valide

```
<!ELEMENT Librairie (Livre)+>
<!ELEMENT Livre (Titre, Auteur+, Annee, Editeur)>
<!ELEMENT Titre(#PCDATA)>
<!ELEMENT Auteur(#PCDATA)>
<!ELEMENT Annee(#PCDATA)>
<!ELEMENT Editeur(#PCDATA)>
```

```
<?xml version="1.0" ?>
<Librairie
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsd:schemaLocation="http://www.librairie.org/librairie.xsd">
  <Livre>
    <Titre>Schémas XML</Titre>
    <Auteur> Jean-Jacques Thomasson</Auteur>
    <Annee>2011</Annee>
    <Editeur>Eyrolles</Editeur>
  </Livre>
</Librairie>
```

Exercice d'application:

Document XML Schéma équivalent

```
<!ELEMENT Librairie (Livre)+>
<!ELEMENT Livre (Titre, Auteur+, Annee, Editeur)>
<!ELEMENT Titre(#PCDATA)>
<!ELEMENT Auteur(#PCDATA)>
<!ELEMENT Annee(#PCDATA)>
<!ELEMENT Editeur(#PCDATA)>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Librairie">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Livre" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Titre" type="xsd:string"/>
              <xsd:element name="Auteur"
                type="xsd:string" maxOccurs="unbounded"/>
              <xsd:element name="Annee" type="xsd:gYear"/>
              <xsd:element name="Editeur" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Exercice d'application:

Document XML Schéma équivalent

```
<!ELEMENT Librairie (Livre)+>
<!ELEMENT Livre (Titre, Auteur+, Annee, Editeur)>
<!ELEMENT Titre(#PCDATA)>
<!ELEMENT Auteur(#PCDATA)>
<!ELEMENT Annee(#PCDATA)>
<!ELEMENT Editeur(#PCDATA)>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="TypeLib">
    <xsd:sequence>
      <xsd:element name="Livre" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Titre" type="xsd:string"/>
            <xsd:element name="Auteur"
                          type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Annee" type="xsd:gYear"/>
            <xsd:element name="Editeur" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="Librairie" type="TypeLib"/>
</xsd:schema>
```