



République Tunisienne  
Ministère de l'Enseignement Supérieur,  
de la Recherche Scientifique



# Technologies XML

## Chap3 :SCHEMA-XML

**Njeh Maissa**

**Audiences: D-LSI-ADBD**



**XML Schema**

**Année universitaire : 2023/2024**

# 1 Qu'est-ce qu'un schéma XML?

- XML Schéma est un langage de **la structure et du typage des documents XML**
- Le langage XML Schéma est également appelé **XML Schema Definition (XSD)**.
- Un fichier dans lequel est écrit un **Schéma XML** porte l'extension **".xsd"**.

Les documents XML Schéma sont des documents:

- Respectant la syntaxe XML
- Permettant de décrire la structure d'un document XML d'une façon beaucoup plus complexe que les DTD



Les Schémas ont été introduits pour combler les lacune des DTD:

- Syntaxe non XML
- pas de contrainte pour les données textes et attributs.
- manque de types pour les données

Permettre de typer les données :

- Éléments simples et complexes
- Attributs simples...

Permettre de définir des contraintes :

- Existence obligatoire ou optionnelle
- Domaines de valeurs, cardinalités, références

Nombreux types de données prédéfinis (booléens, intervalle de temps, nombres, dates,...)

Possibilité de définir de nouveaux types



La structure globale d'un Schéma est :

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

- Comme tout document XML, un schema xml commence par un **prologue** et un **élément racine**.
- L'élément **racine** est l'élément **<xsd:schema>**
- Un **élément**, dans un schéma, se déclare avec la balise **<xsd:element>**
- Les éléments d'un Schéma XML peuvent être de **type simple ou complexe**
- La déclaration lui donne un **type** qui détermine, d'une part, les attributs **autorises et obligatoires**.

➤ Chaque **élément** est déclaré avec un type qui peut être:

- ❖ soit un type **prédéfini**,
- ❖ soit un **nouveau type** défini dans le schéma.



α soit un type **construction**: c'est-à-dire une description explicite des contenus qu'il autorise.

α soit un **dérivation**: c'est-à-dire modification d'un autre type

- Un schéma est un document XML, on trouve dans son **élément racine** l'**attribut** `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

Indique que les **éléments et les types** de données utilisés dans le schéma proviennent de **l'espace de noms**

<http://www.w3.org/2001/XMLSchema> : spécifie également que les éléments et les types de données qui **proviennent de l'espace de nom**

- Pour **valider votre document XML** à l'aide d'un schéma XML, vous devez ajouter à votre XML ceci:

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

- Bien sûr, votre document XML peut aussi faire **référence à une DTD** dans le prologue.

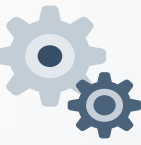
```
<?xml version="1.0" ?>
<!DOCTYPE collection SYSTEM "recettes.dtd">
<collection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="recettes.xsd">
    . . .
</collection>
```

Activer Windows  
Accédez aux paramètres de l'ordinateur  
activer Windows.

On fait **référence** au schéma enregistré localement dans le document XML en utilisant l'attribut **xsi:noNamespaceSchemaLocation= "recettes.xsd"**



## 6 Déclaration d'éléments



La déclaration la plus simple d'un élément prend la forme suivante:

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

Exemples:

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

Utilisation des **attributs default et fixed** de <xsd:element >

```
<xsd:element name= " title" type=" xsd:string" default= " titre par défaut " />  
<xsd:element name= " title" type=" xsd:string " fixed= " titre fixe" />
```

un type prédéfini

## 6 Déclaration d'éléments




```
<xsd:element name= " element" >  
<xsd:simpleType >  
.....  
</ xsd:simpleType >  
</xsd:element >
```

```
<xsd:element name= " element" >  
<xsd:complexType >  
.....  
</ xsd:complexType >  
</xsd:element >
```

**Élément global:** déclaration **xsd:element** est un **enfant** direct de l'**élément** **xsd:schema**

```
<xsd:element name= " title" type=" Title" />  
<xsd:complexType ...>  
.....  
<xsd:element ref= " title" />  
.....  
</ xsd:complexType >
```



nouveau type : type construction



Élément local:

Deux éléments locaux peuvent avoir **le même nom avec des types différents**

```
<xsd:element name= " strings" />
```

```
  <xsd:complexType>
```

```
    <xsd:sequence>
```

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

```
    </xsd:sequence>
```

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

1<sup>er</sup> élément test

```
</xsd:element>
```

```
<xsd:element name= " integers" />
```

```
  <xsd:complexType>
```

```
    <xsd:sequence>
```

```
      <xsd:element name= " test" type=" xsd:integer" />
```

```
    </xsd:sequence>
```

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

2ème élément test

```
</xsd:element>
```

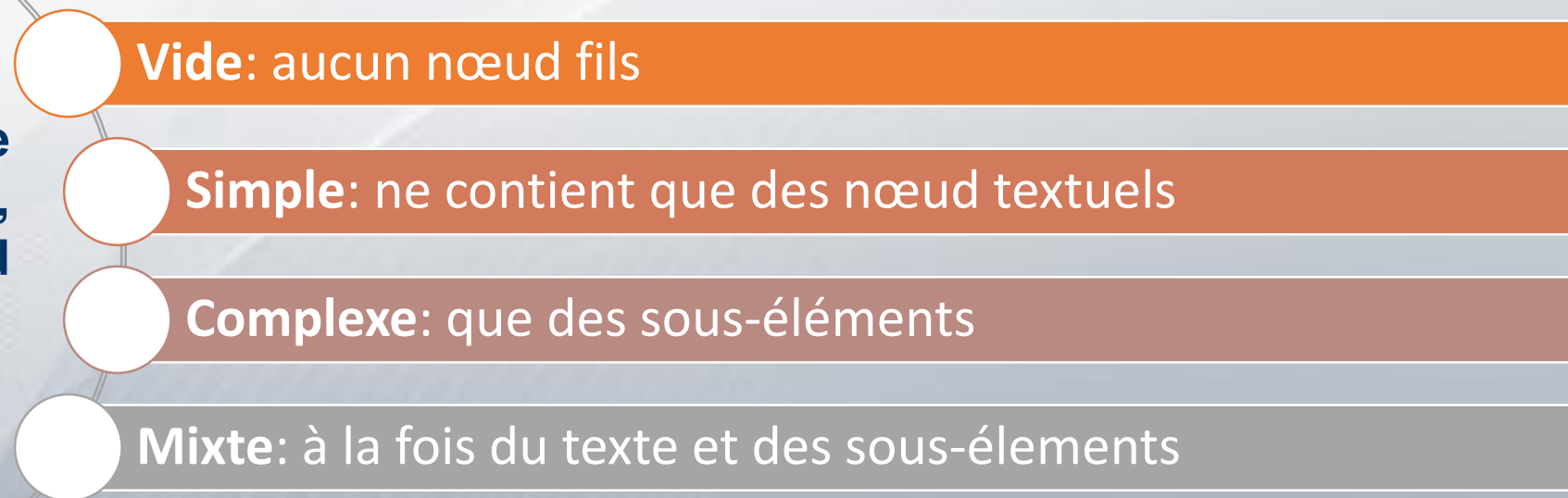
## Élément simple

- Le contenu d'un nœud textuel ou d'un attribut

## Élément complexe

- Utilisés pour décrire les autres formes de contenu

**Différents modèles de contenu pour un élément, selon la nature de ses nœud fils, sont autorisés:**



Xsd:anyType

```
graph LR; anyType[Xsd:anyType] --- simpleType[xsd:simpleType]; anyType --- complexType[xsd:complexType]; simpleType --- atomicType[xsd:atomicType]; simpleType --- list[xsd:list]; simpleType --- union[Xsd:union]; simpleType --- extention1[xsd:extention]; simpleType --- restriction1[xsd:restriction]; complexType --- sequence[xsd:sequence]; complexType --- choice[xsd:choice]; complexType --- all[xsd:all]; complexType --- extention2[xsd:extention]; complexType --- restriction2[xsd:restriction];
```

xsd:simpleType

xsd:atomicType

xsd:list  
Xsd:union

xsd:extention  
xsd:restriction

xsd:complexType

xsd:sequence  
xsd:choice  
xsd:all

xsd:extention  
xsd:restriction

## note.xml

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
</note>
```

## note.dtd

```
<!ELEMENT note (to, from,
heading)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
```

## note.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.xxxx.com"
xmlns="https://www.xxxx.com">
|
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xsd:string"/>
        <xs:element name="from" type="xsd:string"/>
        <xs:element name="heading" type="xsd:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xsd:schema>
```

➡ L'élément **note** est un type **complexe** car il contient **d'autres éléments**. Les autres éléments (**to**, **from**, **head**) sont des types **simples** car ils **ne contiennent pas d'autres éléments**.

## Une référence à une DTD

```
<?xml version="1.0"?>

<!DOCTYPE note SYSTEM
"http://www.xxxx.com/xml/note.dtd">

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
</note>
```

**note.xml**

## Une référence à un schéma XML

```
<?xml version="1.0"?>

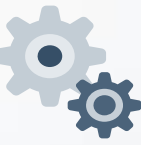
<note
  xmlns="https://www.xxx.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://www.xxxx.com/xml note.xsd">

  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
</note>
```

l'attribut **schemaLocation** : a deux valeurs, séparées par un espace.

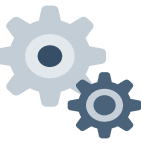
La première valeur **est l'espace de noms à utiliser.**

La deuxième valeur **est l'emplacement du schéma XML à utiliser**



**XML Schema contient de nombreux types de données intégrés. Les types les plus courants sont :**

- ✓ **xs:chaîne**
- ✓ **xs:décimal**
- ✓ **xs:entier**
- ✓ **xs:booléen**
- ✓ **xs:date**
- ✓ **xs:heure**



## **1-Déclaration d'éléments simples:**

Un élément **simple** est un élément XML qui ne peut contenir que du **texte**. Il ne peut contenir **aucun autre élément ou attribut**.



## 8 Déclaration d'éléments simples



- Exemple1: **note.xsd**

- L'élément **note** est du type **xsd:float** qui est un type simple prédéfini de XML Schéma.

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="note" type="xsd:float" />
</xsd:schema>
```

- On fait **référence** au schéma enregistré localement dans le document XML en utilisant l'attribut **xsi:noNamespaceSchemaLocation**

**note.xml**

```
<?xml version="1.0"?>
<note xmlns:xsi="
  http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="note.xsd">
  16.5
</note>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="contacts" type="typeContacts" />
  <xsd:element name="remarque" type="xsd:string" />
  <!-- déclarations de types simples ou complexes ici -->
</xsd:schema>
```

- Déclare deux éléments : un élément **contacts** et un élément **remarque**.
  - L'élément **contacts** du type **typeContact** , qui est un type complexe défini par l'utilisateur.
  - L'élément **remarque** est du type **xsd:string** , qui est un type simple prédéfini de xml schema

## 8 Déclaration d'éléments simples

```
<xsd:element name="strings" />
  <xsd:simpleType>
    <xsd:restriction base=" " ">
      <xsd:element name="test" type="xsd:string" />
    </xsd:sequence>
  </xsd:simpleType>
</xsd:element>
```

**Exemple:** élément vide sans attribut

- Document XML Schéma: élément sans attributs

**<xsd:element name='vide' type='xsd:anytype'/>**

- Document XML:

**<vide/>**

- Un **élément complexe** est un élément qui peut contenir d'autres éléments ou bien des attributs.
- Il existe 4 types d'éléments complexes :

*Les éléments vides avec attributs*

*Les éléments qui contiennent d'autres éléments*

*Les éléments (avec des attributs) qui contiennent uniquement du texte*

*Les éléments qui contiennent du texte et d'autres éléments*

# Déclaration d'éléments complexes

Dans un schéma XML, un élément complexe se déclare en utilisant la balise **xsd:complexType**

```
<xsd:element name="personne">  
  <xsd:complexType>  
    .....  
  </xsd:complexType>  
</xsd:element>
```

## 1- Déclaration d'éléments complexes: éléments vides

- Les **éléments vides** ne contiennent pas de texte, ni d'autres éléments
- Un **élément vide** peut contenir des **attributs**
- La définition d'un **attribut** se fait grâce à l'élément **xsd:attribute**

### Exemple 1 : éléments vides

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

## Exemple 2 : élément vide avec attributs

```

```

Un schema XML validant :

```
<xsd:element name="img">  
  <xsd:complexType>  
    <xsd:attribute name="src" type="xsd:anyURI" use="required"/>  
  </xsd:complexType>  
</xsd:element>
```

Acti

### Remarque:

Un attribut est **optionnel** par défaut. Ou : **use='optional'**

Pour que l'attribut soit **obligatoire** : **use="required"**

Pour que l'attribut soit **interdit** : **use="prohibited"**



# 13 Déclaration d'éléments complexes

## Exemple 3 :élément vide avec attributs

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

- Nom est un attribut **obligatoire**
- Prenom est un attribut **optionnel**
- Les attributs peuvent avoir une valeur par **défaut** OU **une valeur fixe** spécifiée.

.....

```
<xsd:attribute name=" nom " type="xsd:string" default="EN"/>
<xsd:attribute name=" prenom " type="xsd:string" fixed=" FR"/>
```

.....

# 14 Déclaration d'éléments complexes

## 2- Description des éléments ayant des sous-éléments

### 2.1- Une séquence ordonnée d'éléments:

- Dans la balise **xsd:complexType**, la liste des sous-éléments est décrite à l'intérieur d'une balise **xsd:sequence**
- La balise **xsd:sequence** définit une liste ordonnée de sous éléments.
- A l'intérieur de la balise **xsd:sequence**, chaque sous-élément est décrit par une balise **<xsd:element>**

#### Exemple :auteur.xsd

```
<xsd:element name=" auteur" >
<xsd:complexType >
  <xsd:sequence >
    <xsd:element name=" nom " type="xsd:string" />
    <xsd:element name=" prenom " type="xsd:string" />
  </xsd:sequence >
</xsd:complexType > </xsd:element >
```

# 15 Déclaration d'éléments complexes

## 2.2- Une alternative d'éléments: xsd:choice

Exemple : soit l'adresse d'une personne ou son adresse électronique

```
<xsd:complexType name="typePersonne">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string" />
    <xsd:element name="prenom" 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>
  </xsd:sequence>
  <xsd:element name="telephone" type="numeroDeTelephone" />
</xsd:complexType>
```

DTD :


```
<!ELEMENT typePersonne (nom,prenom,dateDeNaissance,(adresse | adresseElectronique),telephone)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>.....
```

# 16 Déclaration d'éléments complexes

## 2.2- Une alternative d'éléments: xsd:choice

Exemple 2: l'équivalent de :<!ELEMENT Librairie(Livre|Magazine|BD)\*>

```
<xsd:element name=" Librairie" >
<xsd:complexType >
<xsd:choice minOccurs= " 0 " maxOccurs= " unbounded " >
<xsd:element name=" Livre " type="xsd:string" />
<xsd:element name=" Magazine " type="xsd:string" />
<xsd:element name=" BD " type="xsd:string" />
</ xsd:choice>
</xsd:complexType > </xsd:element >
```



- L'indicateur <maxOccurs> spécifie le **nombre maximum de fois** qu'un élément peut apparaître
- L'indicateur <minOccurs> spécifie le **nombre minimum de fois** qu'un élément peut apparaître
- Pour permettre à un élément d'apparaître un **nombre illimité de fois**, utilisez l'instruction **maxOccurs="unbounded"**

# 17 Déclaration d'éléments complexes

## 2.3- Une séquence non ordonnée d'éléments: xsd:all

Exemple : l'ordre n'est pas important du nom et du prénom.

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

# 18 Déclaration d'éléments complexes

## 3- Les éléments ayant que du texte et des attributs

### Exemple 1: Élément à contenu simple

```
<xsd:element name=" devise" >  
<xsd:complexType >  
  <xsd:simpleContent>  
    <xsd:extension base= " float" >  
      <xsd:attribute name=" unite " type="xsd:string" />  
    </xsd:extension >  
  </xsd:simpleContent>  
</xsd:complexType >  
</xsd:element >
```

Une extension est une notion qui permet d'ajouter des informations a un type existant



L'élément **devise** peut prendre des valeurs réelles et un attribut **unite**.  
Il s'agit d'un contenu simple (pas de sous éléments), extension du type float.

Exemple: **<devise unite='dinars'>97.5</devise>**



### 3- Les éléments ayant que du texte et des attributs

#### Exemple 1: Element à contenu simple

```
<xsd:element name=" personne" >  
<xsd:complexType >  
  <xsd:simpleContent>  
    <xsd:extention base= "  xsd:string " >  
      <xsd:attribute name="  pays "  type="xsd:string" />  
    </xsd:extention >  
  </ xsd:simpleContent>  
</xsd:complexType >  
</xsd:element >
```

L'élément *xsd:simpleContent* indique que le nouvel élément ne contient pas de sous-éléments.

Dans cet exemple, l'élément *personne* contient du texte et un attribut nommé *pays*.  
Exemple: `<personne pays="maroc">ala</personne>`



#### 4- Les éléments qui contiennent du texte et d'autres éléments (élément mixte)

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

Exemple XML correspondant :

```
<personne>  
  Je suis <nom>Rachad</nom> <prenom>Ali</prenom>.  
</personne>
```

## L'élément mixte : Exemple

```
<xs:element name="letter" type="lettertype"/>
<xs:complexType name="lettertype" mixed="true">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="orderid" type="xs:positiveInteger"/>
    <xs:element name="shipdate" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

Exemple XML correspondant :

```
<letter>
  Dear Mr. <name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

## 21 Déclaration d'éléments quelconques

- Comme pour les DTD, il peut y avoir des éléments quelconques:

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

- Il est possible de faire de même pour les attributs:

```
<xsd:element name="personne" type="personnelInfo" />  
<xsd:complexType name="personnelInfo">  
  <xsd:anyAttribute />  
</xsd:complexType>
```

- L'attribut **type** indique le type de l'élément.
- Les schémas XML permettent de définir différents types de données pour les éléments du schéma. Les plus communs sont :



- La cardinalité des éléments: **minOccurs**, **maxOccurs**.
  - **maxOccurs="unbounded"** : .....
- Ces deux attributs ont une valeur égale 1 par défaut.

Il est recommandé de commencer par déclarer:

- 1- les éléments et attributs de type simple,
- 2- puis ceux de type complexe.

On peut faire référence dans une déclaration de type complexe à un élément de type simple préalablement défini.

```
<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>
```



**<!-- - declaration des elements simples -->**

**<xsd:element name=" auteur" type="xsd:string" />**

**<xsd:element name=" pages" type="xsd:positiveInteger" />**

**<!-- - declaration des elements complexe -->**

**<xsd:element name=" livre">**

**<xsd:complexType >**

**<xsd:sequence >**

**<xsd:element ref=" auteur " />**

**<xsd:attribute ref=" pages " />**

**</xsd:sequence >**

**</xsd:complexType >**

**</xsd:element >**

## Création d'un type complexe à partir de types simples

Soit un *élément contenant une valeur de type simple*, et possédant un attribut, comme:

```
<poids unite=" kg">60</poids>
```

Un tel *élément ne peut pas être déclaré de type simple, car il* contient un attribut.  
On va *dériver par extension un type complexe typePoids à partir* du type simple **positiveInteger**

```
<xsd:element name=" poids" >  
<xsd:complexType name=" typePoids" >  
<xsd:simpleContent >  
<xsd:attribut name=" unite " type="xsd:positiveInteger " use=Required />  
</xsd:simpleContent >  
</xsd:complexType >  
</xsd:element >
```



L'élément *xsd:simpleContent* indique que le nouvel élément ne contient pas de sous-éléments.



- Les types simples et complexes permettent déjà de faire plus de choses que les déclarations dans le langage DTD.
- Il est possible de raffiner leur déclaration de telle manière qu'ils soient une **restriction ou une extention** d'un type déjà existant en vue de préciser un peu plus leur forme

- La dérivation par **restriction** permet de créer de **nouveaux types simples** en imposant des contraintes à des types simple prédéfinis se XML Schéma.
- Pour cela, on utilise des **facettes**, qui sont des **contraintes** appliquées à un type simple particulier.
- Une ***facette permet de placer une contrainte sur l'ensemble des*** valeurs que peut prendre un type de base.

- Par exemple, on peut créer un **type simple**, appelé **monEntier**, limité aux valeurs comprises entre 0 et 99 inclus.
- On dérive ce type par restriction à partir du type simple prédéfini **nonNegativeInteger**, en utilisant la facette: **maxExclusive**.

Exemple:

```
<xsd:simpleType name=" monEntier " >  
<xsd:restriction base =" xsd:nonNegativeInteger " >  
<xsd: maxExclusive value=" 100 " />  
</xsd:restriction >  
</xsd:simpleType>
```

Le type **prédéfini** string a six caractéristiques ou **facettes modifiables** :

**enumeration** : énumération de valeurs possibles.

**length** : longueur de la chaîne.

**minLength** : longueur minimale.

**maxLength** : longueur maximale.

**pattern** : permet de préciser des caractéristiques grâce à des expressions régulières.

**whitespace** : permet la gestion des espaces.

## 30 La dérivation par restriction du type

- **Exemple 1:**

Enumérer les valeurs possibles des noms des jours de la semaine (enumeration)

```
<xsd:attribute name="jour" type="typeJourSemaine" use="required" />
<xsd:simpleType name="typeJourSemaine">
  <xsd:restriction base="xsd:string">
    <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:enumeration value="dimanche" />
  </xsd:restriction>
</xsd:simpleType>
```

- **Exemple 2:**

limiter la longueur d'une chaîne

```
<xsd:simpleType name="typeMotLangueFrancaise">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="20" />
  </xsd:restriction>
</xsd:simpleType>
```

- **Exemple 3:** Utilisation des expressions régulières

```
<xsd:simpleType name="typeAdresseElectronique">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="(.)+@(.)+"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Dans cet exemple, **(.)+** signifie que l'on peut mettre n'importe quel caractère au moins une fois et qu'entre les deux chaînes doit apparaître **le caractère @**



- **Exemple 4: *réduction du nombre de chiffres***

Un numéro ISBN est un référent international pour une publication.

Il s'agit d'un numéro à 10 ou 13 chiffres. On peut le déclarer ainsi :

```
<xsd:simpleType name="typeISBN">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[0-9]{10}" />  
  </xsd:restriction>  
</xsd:simpleType>
```



- Le numéro ISBN est de la forme :  
d-ddddd-ddd-d ou d- ddd-ddddd-d ou d-dd-dddddd-d, où 'd' est un chiffre.  
Le type sera de la forme :

```
<xsd:simpleType name="typeISBN">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>  
    <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>  
    <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

- **Exemple 5:** La valeur de l'âge ne peut être inférieure à 0 ou supérieure à 120:

```
<xsd:simpleType name=" typeAge " >  
  <xsd:restriction base =" xsd:nonNegativeInteger " >  
    <xsd: maxExclusive value=" 120 " />  
  </xsd:restriction >  
</xsd:simpleType>
```

- **Exemple 6:** La contrainte **whiteSpace** est définie sur "**preserve**", ce qui signifie que le processeur XML **ne supprimera pas** les caractères de l'espace blanc:

```
<xsd:simpleType name="age">  
  <xsd:restriction base="xsd:string">  
    <xsd:whiteSpace value="preserve"/>  
  </xs:restriction>  
</xs:simpleType>
```

- **Exemple 7:** La contrainte d'espace blanc est définie sur "**replace**", ce qui signifie que le processeur XML **REEMPLACERA** tous les caractères d'espace blanc (retour de ligne, tabulation, espaces et retours de chariot) avec des espaces:

```
<xsd:simpleType name="age">  
  <xsd:restriction base="xsd:string">  
    <xsd:whiteSpace value="replace"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

- Il est possible avec les XML Schema de décrire des listes d'éléments:  
Prenons l'exemple XML du loto :

```
<loto>  
  <boule rang="1 ">19</boule>  
  <boule rang="2 ">42</boule>  
  <boule rang="3 ">24</boule>  
  <boule rang="4 ">10</boule>  
  <boule rang="5 ">6</boule>  
  <boule rang="6 ">35</boule>  
</loto>
```

peut aussi s'écrire : **<loto>19 42 24 1 0 6 35</loto>**

Ce qui se définit en XML Schema par :

```
<xsd:simpleType name="loto">  
  <xsd:list itemtype="xsd:positiveInteger"/>  
</xs:simpleType>
```

- Pour être plus précis, on va restreindre les nombres de 1 à 49 et limiter la liste à 6 éléments :

```
<xsd:simpleType name="chiffreloto">
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:maxInclusive value="49"/>
    <xsd:minInclusive value="1"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="listeloto">
  <xsd:list itemType=" chiffreloto ">
</xsd:simpleType>
<xsd:simpleType name="loto">
  <xsd:restriction base=" listeloto ">
    <xsd:length value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

## Union de types simples

- Le mot clé union permet de faire l'union entre des types simples.

```
<xsd:simpleType name="chiffreloto">  
  <xsd:restriction base="xsd:positiveInteger">  
    <xsd:maxInclusive value="49"/>  
  </xsd:restriction>  
</xsd:simpleType>  
<xsd:simpleType name="typeloto">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="premier tirage"/>  
    <xsd:enumeration value="deuxieme tirage"/>  
  </xsd:restriction>  
</xsd:simpleType>  
<xsd:simpleType name="loto">  
  <xsd:union memberTypes="chiffreloto typeloto"/>  
</xsd:simpleType>
```

- Les éléments suivants sont alors des "instances" valides de cette déclaration :

```
<loto>premier tirage</loto>
```

```
<loto>19632587416</loto>
```

Voici une DTD: éléments sans attributs

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

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



```
<?xml version="1.0" ?>
<Librairie xmlns="http://www.librairie.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.librairie.org librairie.xsd">
  <Livre>
    <Titre>Le guide du routard Galactique</Titre>
    <Auteur>Douglas Adams</Auteur>
    <Date>1979</Date>
    <ISBN>0345391802</ISBN>
    <Editeur>Ballantine Books</Editeur>
  </Livre>
</Librairie>
```

```
<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"/>
            <xsd:element name="Date" type="xsd:string"/>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Editeur" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

## 2ieme correction

```
</xml version="1.0">  
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
  <xsd:element name="Librairie">  
    <xsd:complexType>  
      <xsd:sequence>  
        <xsd:element ref="Livre" minOccurs="1" maxOccurs="unbounded"/>  
      </xsd:sequence>  
    </xsd:complexType>  
  </xsd:element>  
  <xsd:element name="Livre">  
    <xsd:complexType>  
      <xsd:sequence>  
        <xsd:element ref="Titre"/>  
        <xsd:element ref="Auteur"/>  
        <xsd:element ref="Date" />  
        <xsd:element ref="ISBN" />  
        <xsd:element ref="Editeur"/>  
      </xsd:sequence>  
    </xsd:complexType>  
  </xsd:element>  
  <xsd:element name="Titre" type="xsd:string"/>  
  <xsd:element name="Auteur" type="xsd:string"/>  
  <xsd:element name="Date" type="xsd:string"/>  
  <xsd:element name="ISBN" type="xsd:string"/>  
  <xsd:element name="Editeur" type="xsd:string"/>  
</xsd:schema>
```



Version équivalente à la première, mais plus compacte.

```
<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"/>
            <xsd:element name="Date" type="xsd:string"/>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Editeur" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Version équivalente avec des types personnalisés :

```
<xsd:element name="Librairie">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Livre" type="typeLivre" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:complexType name="typeLivre">
  <xsd:sequence>
    <xsd:element name="Titre" type="xsd:string"/>
    <xsd:element name="Auteur" type="xsd:string"/>
    <xsd:element name="Date" type="xsd:string"/>
    <xsd:element name="ISBN" type="xsd:string"/>
    <xsd:element name="Editeur" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```