



Technologie XML

Par :
Pr. Imad Zeroual

Faculté des Sciences et Techniques – Errachidia / 2022-2023



Introduction

Informations, Notions.

Langage XML

Introduction, Arbre XML, Règles de syntaxe.

Définition de Type de Document (DTD)

Définition, Éléments, Attributs, Entités.

Schémas XSD

Structure de base, Éléments, Attributs, Indicateurs.

XPATH & XSLT

Langage de requêtage XPath, Feuilles de style XSL.



Schémas XSD

Structure de base, Éléments, Attributs, Indicateurs.

04



‘ Limitations de DTD,’

- ⊕ Les **DTD** ne sont pas au format **XML** : Cela signifie qu'il est nécessaire d'utiliser un outil spécial pour **parser** un tel fichier, et qui est différent de celui utilisé pour l'édition du fichier **XML**.
- ⊕ Les **DTD** ne supportent pas les « **espaces de noms** » : En pratique, cela implique qu'il n'est pas possible, dans un fichier **XML** défini par une **DTD**, d'importer des définitions de balises définies par ailleurs.
- ⊕ Le **typage** des données est extrêmement limité : **#PCDATA** (élément) et **#CDATA** (attribut).



‘ Avantages de XSD,’

- ✓ Il est plus facile de décrire le contenu de document **autorisé** ;
- ✓ Il est plus facile de valider **l'exactitude** des données ;
- ✓ Il est plus facile de définir les **restrictions** sur les données ;
- ✓ Il est plus facile de définir des **formats** de données ;
- ✓ Il est plus facile de **convertir** des données entre différents **types** de données ;
- ✓ Vous n'avez pas besoin d'apprendre un **nouveau** langage que XML ;
- ✓ Vous pouvez utiliser le **même** analyseur XML pour analyser vos fichiers Schéma ;
- ✓ **Réutilisez** votre schéma dans d'autres schémas ;
- ✓ Créez vos **propres** types de données **dérivés** des types standards ;
- ✓ **Référencer** plusieurs schémas dans le **même** document.



‘ Structure de base,

- Comme tout document XML, un Schéma XML commence par un prologue et a un élément racine.

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<xsd:schema ... >
<!-- déclarations d'éléments, d'attributs et de types ici -->
...
</xsd:schema>
```

- L'élément racine est l'élément `xsd:schema` : Cet élément racine est accompagné d'attributs qui précisent le lieu de définition des éléments.
- Tout élément d'un schéma doit commencer par le préfixe `xsd` ou `xs`.



‘ Structure de base,

Attributs de xsd:schema :

- `xmlns:xs = "http://www.w3.org/2001/XMLSchema"`
 - Indique que les **éléments** et les **types** utilisés (schema, element, complexType, sequence, string, boolean, etc.) qui sont définis dans l'espace indiqué entre " " et qu'ils doivent être préfixés par **xs**.

- `xmlns = "http://www.w3schools.com"`
 - Indique que le **nom d'espace** par défaut est celui entre " ".



‘ Structure de base,

Exemple : exo.xsd

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

<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" >
    <!-- déclarations d'éléments, d'attributs et de types ici -->
        <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>
</xsd:schema>
```



‘Structure de base,

Référencement de schéma depuis un document xml :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<livre
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "exo.xsd"
>
  </auteur> Mohamed </auteur>
  </pages> 386 </pages>
</livre>
```



Éléments,

Déclaration :

Un élément, dans un schéma, se déclare avec la balise `<xsd:element>`

Exemple :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" >
    <xsd:element name = "remarque" type = "xsd:string" />
    <xsd:element name = "contacts" type = "typeContacts" />
    <!-- déclarations de types ici -->
</xsd:schema>
```

- Ce schéma déclare deux éléments : **remarque** et **contacts**
- A chaque élément est associé un type via l'attribut **type**
- **remarque** de type **xsd:string**, type simple prédéfini de **XML Schema**
- **contacts** de type **typeContacts**, type complexe défini par l'utilisateur



‘Éléments,

Types de données :

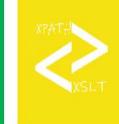
XML Schema permet de spécifier des types de données bien plus finement que le langage DTD.

Il distingue notamment deux types :

- Types **simples** prédéfinis de **XML Schema**
- Types **complexes** défini par l'utilisateur



Schémas XSD Éléments



‘Éléments,

Types de données :

- Types simples de XML Schema
 - Ne peuvent comporter ni attributs, ni éléments enfants ;
 - Il existe nombreux types simples prédefinis, mais il est également possible d'en dériver de nouveaux types ;
 - Syntaxe de déclaration :

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

Exemple :

.XML

```
<Nom> Folan </Nom>
<Age> 22 </Age>
<DateNaissance> 14-11-2000 </DateNaissance>
```

.XSD

```
<x:element name = "Nom" type = "xs:string" />
<x:element name = "Age" type = "xs:integer" />
<x:element name = "DateNaissance" type = "xs:date" />
```



‘Éléments,

Types de données :

- Autres types **simples** :

byte	string	time
unsignedByte	long	dateTime
integer	unsignedLong	date
positiveInteger	short	gYear
negativeInteger	unsignedShort	gMonth
nonNegativeInteger	decimal	gYearMonth
nonPositiveInteger	float	gDay
int	double	gMonthDay
unsignedInt	boolean	



“Éléments,”

Types de données :

- Types **simples** : Valeur **par défaut** et valeur **fixée**

Un élément simple peut avoir une valeur par défaut ou une valeur fixée :

- Une valeur **par défaut** est assignée si aucune autre valeur n'est affectée

→ `<xs:element name = "Nom" type = "xs:string" default = "Folan" />`

- Une valeur **fixée** est aussi assignée mais ne peut pas être changée

→ `<xs:element name = "DateNaissance" type = "xs:date" fixed = "14-11-2000" />`



Attributs,

Déclaration :

- Les attributs sont déclarés de types **simples** → `<xs:attribute name = "xxx" type = "yyy" />`
- Seuls les éléments **complexes** peuvent avoir des **attributs** → XML : `<Livre lang = "Fr" />`
- Voici la définition correspondante de type **simple** → `<xs:attribute name = "lang" type = "xs:string" />`
- On peut également affecter une valeur **par défaut** → `<xs:attribute name = "lang" type = "xs:string" default = "En" />`
- On peut également affecter une valeur **fixe** → `<xs:attribute name = "lang" type = "xs:string" fixed = "Fr" />`
- Tous les attributs sont **optionnels** par défaut → `<xs:attribute name = "lang" type = "xs:string" use = "optional" />`
- On peut les rendre **obligatoires** → `<xs:attribute name = "lang" type = "xs:string" use = "required" />`



' Restrictions,

Restriction sur une valeur :

- Les restrictions sont utilisées sur les éléments **XML** pour ne tolérer que des valeurs acceptables ;
- Les restrictions en **XML** sont appelées **facettes**.

Exemple :

La valeur de l'élément « **age** » doit être par exemple entre **0** et **100**.

```
<xs:element name = "age" >  
  <xs:simpleType>  
    </xs:restriction base = "xs:integer" >  
      <xs:minInclusive value = "0" />  
      <xs:maxInclusive value = "100" />  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```



' Restrictions,

Restriction sur un ensemble de valeurs :

- Pour limiter l'étendue d'un élément de XML à un ensemble de valeurs acceptables en utilisant l'énumération ;

Exemple 1 :

Les valeurs de l'élément « voiture » doivent être limités à les valeurs suivantes "Audi", "Nissan" et "Dacia".

```
<xs:element name = "voiture" >  
  <xs:simpleType>  
    </xs:restriction base = "xs:string" >  
      <xs:enumeration value = "Audi" />  
      <xs: enumeration value = "Nissan" />  
      <xs: enumeration value = "Dacia" />  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```



' Restrictions,

Restriction sur un ensemble de valeurs :

- Pour limiter l'étendue d'un élément de XML à un ensemble de valeurs acceptables en utilisant l'énumération ;

Exemple 2 :

Dans ce cas le type « Typevoiture » peut être employé par d'autres éléments parce que ce n'est pas une partie de l'élément « voiture ».

```
<xs:element name = "voiture" type = "Typevoiture" />
<xs:simpleType name = "Typevoiture" >
    </xs:restriction base = "xs:string" >
        <xs:enumeration value = "Audi" />
        <xs: enumeration value = "Nissan" />
        <xs: enumeration value = "Dacia" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```



' Restrictions,

Restriction sur une série de valeurs :

- Pour limiter le contenu d'un élément XML afin de définir une série de chiffres ou de lettres pouvant être utilisés, nous utilisons la contrainte « pattern ».

Exemples :

Restriction de l'élément « Lettre » à une seule des lettres minuscules de a à z.

```
<xs:element name = "Lettre" />
<xs:simpleType>
    </xs:restriction base = "xs:string" >
        <xs:pattern value = "[a-z]" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```

Restriction de l'élément « initiales » à deux lettres minuscules ou majuscules.

```
<xs:element name = "initiales" />
<xs:simpleType>
    </xs:restriction base = "xs:string" >
        <xs:pattern value = "[a-zA-Z][a-zA-Z]" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```



' Restrictions,

Restriction sur une série de valeurs :

- Pour limiter le contenu d'un élément XML afin de définir une série de chiffres ou de lettres pouvant être utilisés, nous utilisons la contrainte « pattern ».

Exemples :

Restriction de l'élément « Sexe » à l'un des deux mots suivants : **Femme** ou **Homme**.

```
<xs:element name = "Sexe" />
  <xs:simpleType>
    </xs:restriction base = "xs:string" >
      <xs:pattern value = "Femme | Homme" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restriction de l'élément « Code » à **trois** chiffres en séquence, et chaqu'un doit être compris entre **0** et **9**.

```
<xs:element name = "Code" />
  <xs:simpleType>
    </xs:restriction base = "xs:integer" >
      <xs:pattern value = "[0-9][0-9][0-9]" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```



' Restrictions,

Restriction sur une série de valeurs :

- Pour limiter le contenu d'un élément XML afin de définir une série de chiffres ou de lettres pouvant être utilisés, nous utilisons la contrainte « pattern ».

Exemples :

Restriction de l'élément « Nom » à zéro ou plusieurs occurrences de lettres de a à z.

```
<xs:element name = "Nom" />
<xs:simpleType>
    </xs:restriction base = "xs:string" >
        <xs:pattern value = "([a-zA-Z]+)" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```

Restriction de l'élément « Password » à exactement huit caractères qui doivent être des lettres ou un nombre.

```
<xs:element name = "Password" />
<xs:simpleType>
    </xs:restriction base = "xs:integer" >
        <xs:pattern value = "[a-zA-Z0-9]{8}" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```



' Restrictions,

Restriction sur les espaces blancs :

- Pour spécifier comment les espaces blancs doivent être gérés, nous utilisons la contrainte « `whiteSpace` ».

Exemples :

Restreint le processeur XML pour qu'il ne supprime pas les caractères d'espacement de l'élément « `Adresse` ».

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

Remplacer tous les espaces blancs (sauts de ligne, tabulations, et espaces) de l'élément « `Adresse` » par des espaces.

```
<xs:element name = "Adresse" />
<xs:simpleType>
    </xs:restriction base = "xs:string" >
        <xs:whiteSpace value = "replace" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```



' Restrictions,

Restriction sur les espaces blancs :

- Pour spécifier comment les espaces blancs doivent être gérés, nous utilisons la contrainte « `whiteSpace` ».

Exemples :

Le processeur `XML` supprime tous les espaces blancs : les sauts de ligne, les tabulations, les espaces, les retours chariot sont `remplacés` par des espaces, les espaces de début et de fin sont `supprimés` et les espaces multiples sont `réduits à un seul espace`.

```
<xs:element name = "Adresse" />
<xs:simpleType>
    </xs:restriction base = "xs:string">
        <xs:whiteSpace value = "collapse" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```



' Restrictions,

Restrictions de longueur :

- Pour limiter la **longueur** d'une valeur dans un élément, nous utilisons les contraintes **length**, **maxLength** et **minLength**.

Exemples :

Restriction de l'élément « Password » à exactement 8 caractères.

```
<xs:element name = "Password" />
<xs:simpleType>
    </xs:restriction base = "xs:string" >
        <xs:length value = "8" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```

Restriction de l'élément « Password » pour avoir au moins 5 caractères et au maximum 8 caractères :

```
<xs:element name = "Password" />
<xs:simpleType>
    </xs:restriction base = "xs:string" >
        <xs:minLength value = "5" />
        <xs:maxLength value = "8" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
```



‘Éléments complexes,’

Tandis que un élément de type **simple** ne peut contenir de sous éléments et des attributs, un **élément** de type **complexe** est un élément **XML** qui contient d'autres **éléments** et/ou **attributs**. Il existe **quatre** types d'éléments complexes :

1. Éléments vides → <Produit id = "342" />

```
<Personne>
  <Nom> Ben Folan </Nom>
  <Prenom> Folan </Prenom>
</Personne>
```

2. Éléments qui ne contiennent que d'autres éléments → <Produit id = "342"> Lait entier </Produit>

3. Éléments qui ne contiennent que du texte → <Faculte id = "102" >

FSTE <Adresse> 509, Boutalamine </Adresse>

</Faculte>

Remarque : Chacun de ces éléments peut également contenir des **attributs** !



‘Éléments complexes,’

Déclaration d'un élément complexe :

Nous pouvons définir un **élément complexe** dans un schéma XML de **trois** manières différentes :

1. L'élément peut être déclaré **directement** en nommant l'élément :

```
<xs:element name = "Etudiant" >  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name = "Nom" type = "xs:string" />  
      <xs:element name = "Prenom" type = "xs:string" />  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

2. L'élément peut avoir un **attribut type** qui fait référence au **nom** du type **complexe** à utiliser :

```
<xs:element name = "Etudiant" type = "Personne" />  
  
<xs:complexType name = "Personne" >  
  <xs:sequence>  
    <xs:element name = "Nom" type = "xs:string" />  
    <xs:element name = "Prenom" type = "xs:string" />  
  </xs:sequence>  
</xs:complexType>
```



‘Éléments complexes,

Déclaration d'un élément complexe :

Nous pouvons définir un **élément complexe** dans un schéma XML de **deux** manières différentes :

3. Un type complexe peut être **basé** sur un type complexe **existant** en ajoutant autres éléments :

Type complexe existant

```
<xs:complexType name = "Personne" >  
    <xs:sequence>  
        <xs:element name = "Nom" type = "xs:string" />  
        <xs:element name = "Prenom" type = "xs:string" />  
    </xs:sequence>  
</xs:complexType>
```

```
<xs:complexType name = "Fonctionnaire" >  
    <xs:complexContent>  
        <xs:extension base = "Personne">  
            <xs:sequence>  
                <xs:element name = "PPR" type = "xs:integer" />  
                <xs:element name = "Grade" type = "xs:string" />  
            </xs:sequence>  
        </xs:extension>  
    </xs:complexContent>  
</xs:complexType>  
<xs:element name = "Etudiant" type = "Personne" />  
<xs:element name = "Enseignant" type = "Fonctionnaire" />
```



‘Éléments complexes,

Déclaration d'un élément vide :

Pour définir un type **sans contenu**, nous devons définir un type qui **autorise** les éléments dans son contenu, mais nous ne déclarons **aucun** élément.

Exemple :

```
<Produit id = "342" />
```

.xsd

```
<xsd:element name = "Produit" >  
    <xsd:complexType>  
        <xsd:attribute name = "id" type = "xs:positiveInteger" />  
    </xsd:complexType>  
</xsd:element>
```

Ou

.xsd

```
<xsd:element name = "Produit" type = "ProdType" />  
  
<xsd:complexType name = "ProdType" >  
    <xsd:attribute name = "id" type = "xs:positiveInteger" />  
</xsd:complexType>
```



‘Éléments complexes,

Déclaration d'un élément qui ne contiennent que d'autres éléments :

Pour définir ce type d'élément, nous devons définir un type qui autorise **seulement** des éléments et des attributs.

Exemple : <Personne>

```
<Nom> Ben Folan </Nom>
<Prenom> Folan </Prenom>
</Personne>
```

.xsd

```
<xsd:element name = "Personne" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name = "Nom" type = "xs:string" />
      <xsd:element name = "Prenom" type = "xs:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ou

.xsd

```
<xsd:element name = "Personne" type = "PersInfos" />
<xsd:complexType name = "PersInfos" >
  <xsd:sequence>
    <xsd:element name = "Nom" type = "xs:string" />
    <xsd:element name = "Prenom" type = "xs:string" />
  </xsd:sequence>
</xsd:complexType>
```



‘Éléments complexes,’

Déclaration d'un élément qui ne contiennent que du texte :

Ce type ne contient que du contenu simple (texte et attributs), nous ajoutons un élément `simpleContent` autour du contenu, en définissant une `extension` ou une `restriction` dans l'élément `simpleContent`.

Exemple : <`Produit id = "342"`> Lait entier </`Produit`>

.xsd

```
<xsd:element name = "Produit">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base = "xs:string">
        <xsd:attribut name = "id" type = "xs:integer" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

.xsd

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

<xsd:complexType name = "ProdType">
  <xsd:simpleContent>
    <xsd:extension base = "xs:string">
      <xsd:attribut name = "id" type = "xs:integer" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Ou



‘Éléments complexes,’

Déclaration d'un élément mixte :

Le type complexe mixte peut contenir des attributs, des éléments et du texte. Pour permettre aux données textuelles d'apparaître entre les éléments enfants, l'attribut mixte doit être défini sur "true".

Exemple :

```
<Faculte id = "102">
    FSTE <Adresse> 509, Boutalamine </Adresse>
</Faculte>
```

.xsd

```
<xsd:element name = "Faculte" >
    <xsd:complexType mixed="true" >
        <xsd:sequence>
            <xsd:element name = "id" type = "xs:integer" />
        </xsd:sequence>
    </xsd:complexType>
<xsd:element>
```

.xsd

```
<xsd:element name = "Faculte" type = "FacType" />
    <xsd:complexType name = "FacType" mixed="true" >
        <xsd:sequence>
            <xsd:element name = "id" type = "xs:integer" />
        </xsd:sequence>
    </xsd:complexType>
```

Ou



‘ Indicateurs,

Les indicateurs permettent de **contrôler** l'usage des éléments à l'intérieur du type **complexe**. Il existe **7** types d'indicateurs :

- **Indicateurs d'ordre** : permettent de définir l'ordre des éléments.
 1. All
 2. Choice
 3. Sequence
- **Indicateurs d'occurrence** : pour définir la fréquence à laquelle un élément peut se produire.
 4. maxOccurs
 5. minOccurs
- **Indicateurs de groupes** : pour définir des ensembles d'éléments liés.
 6. Groupname
 7. attributeGroupName



Indicateurs,

Indicateurs d'ordre : all

Cet indicateur spécifie que les éléments enfants peuvent apparaître dans **n'importe quel ordre** et que chaque élément enfant ne doit apparaître qu'**une seule fois**.

Exemple :

.xml

```
<Personne>
    <Nom> Ben Folan </Nom>
    <Prenom> Folan </Prenom>
</Personne>
<Personne>
    <Prenom> Hayan </Prenom>
    <Nom> Ben Bayan </Nom>
</Personne>
```

.xsd

```
<xs:element name = "Personne" >
    <xs:complexType>
        <xs:all>
            <xs:element name = "Nom" type = "xs:string" />
            <xs:element name = "Prenom" type = "xs:string" />
        </xs:all>
    </xs:complexType>
</xs:element>
```



Indicateurs,

Indicateurs d'ordre : Choice

Cet indicateur spécifie qu'un élément enfant ou un autre peut apparaître.

Exemple :

.xml

```
<Personne>
    <Age>18</Age>
</Personne>
<Personne>
    <DateNaissance>14-11-2000</DateNaissance>
</Personne>
```

.xsd

```
<xs:element name = "Personne" >
    <xs:complexType>
        <xs:choice>
            <xs:element name = "Age" type = "xs:integer" />
            <xs:element name = "DateNaissance" type = "xs:date" />
        </xs:choice>
    </xs:complexType>
</xs:element>
```



Indicateurs,

Indicateurs d'ordre : sequence

Cet indicateur spécifie que les éléments enfants doivent apparaître dans un ordre spécifique.

Exemple :

.xml

```
<Personne>
    <Nom> Ben Folan </Nom>
    <Prenom> Folan </Prenom>
</Personne>
<Personne>
    <Nom> Ben Bayan </Nom>
    <Prenom> Hayan </Prenom>
</Personne>
```

.xsd

```
<x:element name = "Personne" >
    <x:complexType>
        <x:sequence>
            <x:element name = "Nom" type = "xs:string" />
            <x:element name = "Prenom" type = "xs:string" />
        </x:sequence>
    </x:complexType>
</x:element>
```



Indicateurs,

Indicateurs d'occurrence : maxOccurs

Cet indicateur spécifie le nombre maximal de fois qu'un élément peut se produire.

Exemple :

.xml

```
<Personne>
    <Nom> Folan Ben Folan </Nom>
    <Enfant> Mohamed </Enfant>
    <Enfant> Ali </Enfant>
    <Enfant> Omar </Enfant>
    <Enfant> Fatima </Enfant>
    <Enfant> Meryam </Enfant>
</Personne>
```

.xsd

```
<xsd:element name = "Personne" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name = "NomComplet" type = "xs:string" />
            <xsd:element name = "Enfant" type = "xs:string" maxOccurs = "10" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Remarque : Pour permettre à un élément d'apparaître un nombre illimité de fois, utilisez l'instruction `maxOccurs="unbounded"`.



Indicateurs,

Indicateurs d'occurrence : minOccurs

Cet indicateur spécifie le nombre minimal de fois qu'un élément peut se produire.

Exemple :

.xml

```
<Personne>
    <Nom> Folan Ben Folan </Nom>
    <Enfant> Mohamed </Enfant>
    <Enfant> Ali </Enfant>
    <Enfant> Omar </Enfant>
    <Enfant> Fatima </Enfant>
    <Enfant> Meryam </Enfant>
</Personne>
```

.xsd

```
<xs:element name = "Personne" >
    <xs:complexType>
        <xs:sequence>
            <xs:element name = "NomComplet" type = "xs:string" />
            <xs:element name = "Enfant" type = "xs:string" maxOccurs = "10" />
            <xs:element name = "Voiture" type = "xs:string" minOccurs = "0" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Remarque : Pour tous les indicateurs d'Ordre et de Groupe, la valeur par défaut pour maxOccurs et minOccurs est 1.



‘ Indicateurs,

Indicateurs de groupes : Groupname

Cet indicateur doit contenir un élément **all**, **choice** ou **sequence**. Le groupe d'éléments défini peut y être référencé dans une autre déclaration d'élément.

Exemple :

```
<xs:group name = "NomComplet" >
    <xs:sequence>
        <xs:element name = "Nom" type = "xs:string" />
        <xs:element name = "Prenom" type = "xs:string" />
    </xs:sequence>
</xs:group>
<xs:element name = "Personne" >
    <xs:complexType>
        <xs:sequence>
            <xs:group ref = "NomComplet" />
            <xs:element name = "DateNaissance" type = "xs:date" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```



Indicateurs,

Indicateurs de groupes : AttributeGroup

Cet indicateur regroupe les attributs liés entre eux afin de l'utiliser dans une déclaration d'élément.

Exemple :

```
<xs:attributeGroup name = "LivreAttrib" >
    <xs:sequence>
        <xs:attribut name = "id" type = "xs:integer" />
        <xs:attribut name = "lang" type = "xs:string" />
        <xs:attribut name = "edition" type = "xs:integer" />
    </xs:sequence>
</xs:attributeGroup>
<xs:element name = "Livre" >
    <xs:complexType>
        <xs:sequence>
            <xs:attributeGroup ref = "LivreAttrib" >
                <xs:element name = "Pages" type = "xs:integer" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
```



TD-3,

Exercice 1 :

Rédiger un schéma **xsd** pour valider ce fichier **xml** :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<Examen
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="examen.xsd" code="coursXML"
>
  <Titre> Outils et documents XML </Titre>
  <Date mois = "sep" annee = "2008" />
  <Questions>
    <question>
      <Partie/>
    </question>
    <question>
      <Partie/>
    </question>
    <question>
      <Partie/>
    </question>
    <question>
      <Partie/>
    </question>
  </Questions>
</Examen>
```



TD-3,

Correction :

Le schéma **xsd** pour valider ce fichier **xml** :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<Examen
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="examen.xsd" code="coursXML"
>
  <Titre> Outils et documents XML </Titre>
  <Date mois = "sep" annee = "2008" />
  <Questions>
    <question>
      <Partie/>
    </question>
    <question>
      <Partie/>
    </question>
    <question>
      <Partie/>
    </question>
    <question>
      <Partie/>
    </question>
  </Questions>
</Examen>
```

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
  elementFormDefault = "qualified" >
  <xs:element name = "Examen" >
    <xs:complexType>
      <xs:attribut name = "code" type = "xs:string" use = "required" />
      <xs:sequence>
        <xs:element name = "Titre" type = "xs:string" />
        <xs:element name = "Date" >
          <xs:complexType>
            <xs:attribut name = "mois" type = "xs:string" />
            <xs:attribut name = "annee" type = "xs:integer" />
          </xs:complexType>
        </xs:element>
        <xs:element name = "Questions" >
          <xs:complexType>
            <xs:sequence>
              <xs:element name = "question" maxOccurs="unbounded" >
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name = "Partie" type = "xs:string" />
                  <xs:sequence>
                  </xs:complexType>
                </xs:element>
                <xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:element>
    </xs:sequence>
  </xs:element>
</xsd:schema>
```



TD-3,

Exercice 2 :

Rédiger un schéma **xsd** pour valider ce fichier **xml** :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<Livre
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="livre.xsd" titre="Mon livre"
>
  <Auteurs>
    <auteur nom = "nom1" prenom = "prenom1" />
    <auteur nom = "nom1" prenom = "prenom1" />
  </Auteurs>
  <Sections>
    <section titre = "section1" >
      <Chapitre titre = "chapitre1" >
        <Paragraphe> Premier paragraphe </Paragraphe>
        <Paragraphe> Deuxième paragraphe </Paragraphe>
      </Chapitre >
    </section>
  </Sections>
</Livre>
```



TD-3,

Correction :

Le schéma **xsd** pour valider ce fichier **xml** :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<Livre
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="livre.xsd" titre="Mon livre">
  <Auteurs>
    <auteur nom = "nom1" prenom = "prenom1" />
    <auteur nom = "nom1" prenom = "prenom1" />
  </Auteurs>
  <Sections>
    <section titre = "section1" >
      <Chapitre titre = "chapitre1" >
        <Paragraphe> Premier paragraphe </Paragraphe>
        <Paragraphe> Deuxième paragraphe </Paragraphe>
      </Chapitre >
    </section>
  </Sections>
</Livre>
```

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" elementFormDefault = "qualified" >
  <xs:element name = "Livre" >
    <xs:complexType>
      <xs:attribut name = "titre" type = "xs:string" use = "required" />
      <xs:sequence>
        <xs:element name = "Auteurs" >
          <xs:complexType>
            <xs:sequence>
              <xs:element name = "auteur" maxOccurs="unbounded" >
                <xs:complexType>
                  <xs:attribut name = "nom" type = "xs:string" />
                  <xs:attribut name = "prenom" type = "xs:string" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xsd:schema>
```



TD-3,

Correction :

Le schéma **xsd** pour valider ce fichier **xml** :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<Livre
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="livre.xsd" titre="Mon livre">

  <Auteurs>
    <auteur nom = "nom1" prenom = "prenom1" />
    <auteur nom = "nom1" prenom = "prenom1" />
  </Auteurs>
  <Sections>
    <section titre = "section1" >
      <Chapitre titre = "chapitre1" >
        <Paragraphe> Premier paragraphe </Paragraphe>
        <Paragraphe> Deuxième paragraphe </Paragraphe>
      </Chapitre >
    </section>
  </Sections>
</Livre>
```

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

  <xs:element name = "Livre" >
    <xs:complexType>
      <xs:attribut name = "titre" type = "xs:string" use = "required" />
      <xs:sequence>
        <xs:element name = "Auteurs" >
          <xs:complexType>
            <xs:sequence>
              <xs:element name = "Sections" />
            </xs:sequence>
            <xs:complexType>
              <xs:sequence>
                <xs:element name = "section" maxOccurs="unbounded" >
                  <xs:complexType>
                    <xs:attribut name = "titre" type = "xs:string" use = "required" />
                    <xs:sequence>
                      <xs:element name = "Chapitre" maxOccurs="unbounded" >
                        <xs:complexType>
                          <xs:attribut name = "titre" type = "xs:string" use = "required" />
                          <xs:sequence>
                            <xs:element name = "Paragraphe" type = "xs:string" maxOccurs="unbounded" />
                            <xs:sequence>
                              </xs:complexType>
                            </xs:sequence>
                          </xs:element>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xsd:schema>
```



TD-3,

Correction :

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<Livre
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="livre.xsd" titre="Mon livre">

  <Auteurs>
    <auteur nom = "nom1" prenom = "prenom1" />
    <auteur nom = "nom1" prenom = "prenom1" />
  </Auteurs>
  <Sections>
    <section titre = "section1">
      <Chapitre titre = "chapitre1" >
        <Paragraphe> Premier paragraphe </Paragraphe>
        <Paragraphe> Deuxième paragraphe </Paragraphe>
      </Chapitre >
    </section>
  </Sections>
</Livre>
```

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" elementFormDefault = "qualified" >
  <xsd:element name = "Livre" >
    <xsd:complexType>
      <xsd:attribut name = "titre" type = "xs:string" use = "required" />
      <xsd:sequence>
        <xsd:element name = "Auteurs" >
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name = "auteur" maxOccurs = "unbounded" >
                <xsd:complexType>
                  <xsd:attribut name = "nom" type = "xs:string" />
                  <xsd:attribut name = "prenom" type = "xs:string" />
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name = "Sections" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name = "section" maxOccurs = "unbounded" >
          <xsd:complexType>
            <xsd:attribut name = "titre" type = "xs:string" use = "required" />
            <xsd:sequence>
              <xsd:element name = "Chapitre" maxOccurs = "unbounded" >
                <xsd:complexType>
                  <xsd:attribut name = "titre" type = "xs:string" use = "required" />
                  <xsd:sequence>
                    <xsd:element name = "Paragraphae" type = "xs:string" maxOccurs = "unbounded" />
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```