3. XML Schéma

- Un schéma d'un document définit:
  - les éléments possibles dans le document
  - les attributs associés à ces éléments
  - la structure du document
  - les types de données
- Le schéma est spécifié en XML
  - pas de nouveau langage
  - balisage de déclaration
  - espace de nom spécifique xsd: ou xs:
- Présente de nombreux avantages
  - structures de données avec types de données
  - extensibilité par héritage et ouverture
  - analysable à partir d'un parseur XML standard

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, ...
- S'intégrer à la galaxie XML
  - Espace de noms
  - Infoset (structure d'arbre logique)

Le modèle des schémas

- Déclaration des éléments et attributs
  - Nom
  - Typage similaire à l'objet
- Spécification de types simples
  - Grande variété de types
- Génération de types complexes
  - Séquence (Sequence)
  - Choix (Choice)
  - Tas (All)

Les types simples (1)

- Chaines de caractères
  - string
  - normalizedString
  - token
- Binaires
  - byte
  - unsignedByte
  - base64Binary
  - hexBinary
- integer
  - -126789, -1, 0, 1, 126789
- positiveInteger
  - 1, 126789
- negativeInteger
  - -126789, -1
- nonNegativeInteger
  - 0, 1, 126789
- nonPositiveInteger
  - -126789, -1, 0
- int
  - -1, 126789675
- unsignedInt
  - 0, 1267896754

Les types simples (2)

- long
  - -1, 12678967543233
- unsignedLong
  - 0, 12678967543233
- short
  - -1, 12678
- unsignedShort
  - 0, 12678
- decimal
  - -1.23, 0, 123.4, 1000.00
- float
  - -INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
- double
  - -INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
- boolean
  - true, false 1, 0
- time
  - 13:20:00.000, 13:20:00.000-05:00
- dateTime
  - 1999-05-31T13:20:00.000-05:00
- duration
  - P1Y2M3DT10H30M12.3S
- date
  - 1999-05-31
- gMonth
  - --05--
- gYear
  - 1999

Les types simples (3)

- gYearMonth
  - 1999-02
- gDay
  - ---31
- gMonthDay
  - --05-31
- Name
  - shipTo
- QName
  - po:USAddress
- NCName
  - USAddress
- anyURI
  - http://www.example.com/,
  - http://www.example.com/doc.html#ID5
- language
  - en-GB, en-US, fr
- ID
  - "A212"
- IDREF
  - "A212"
- IDREFS
  - "A212" "B213"
- ENTITY
- ENTITIES
- NOTATION
- NMTOKEN, NMTOKENS
  - US
  - Brésil Canada Mexique

Types de données

- Simple : type primitif (chaîne, numérique et date)
  - **<xsd:element name="immat" type="xsd:string"/>**
- Dérivé : construit en ajoutant des contraintes (de toute sorte,
  - expressions régulières permises
  - Restriction (sous-type)
  - Union (alternative entre plusieurs types)
  - Liste (ensemble de types semblables)
- Complexes (contenu d'un élément) : construit par composition d'éléments
  - <sequence> : collection ordonnée d'éléments typés
  - <all> : collection non ordonnée d'éléments typés
  - <choice>: choix entre éléments typés

Le document XSD

- L'extension du fichier est .xsd
- L'élément racine est <schema>
- Le *namespace* suivant est obligatoire (préfixe xsd ou xs)
  - **xmlns:xsd="http://www.w3.org/2001/XMLSchema"**
  - Source des éléments et des types de données

Référencer un xsd

- Un document xml fait référence à un schémas par
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
- Dans le cas d'un schéma local (cas d'une DTD SYSTEM) on utilise l'attribut
  xsi:noNamespaceSchemaLocation
- Si non
  xsi:schemaLocation="http://ensak.usms.ac.ma/fichier.xsd"
- Exemple
  - **<?xml version="1.0"?>**
    - **<rootElement**
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="fichier.xsd">
      …
      </rootElement>

XML, Schéma et DTD

```
<?xml version="1.0"?>
<mail> <to>Ahmed</to>
<from>Mohamed</from>
<subject>Rendez-vous</subject>
<body>N'oublie pas le week-end!</body>
</mail>
```

```
<!ELEMENT mail (to, from, subject, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

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

<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema" >

<xs:element name="mail">
<xs:complexType>

<xs:sequence>

<xs:element name="to" type="xs:string"/>
<xs:element name="from"
type="xs:string"/> <xs:element name="
subject" type="xs:string"/> <xs:element
name="body" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>
```

Elément simple

- Il ne peut contenir que du texte
  - Ne peut avoir d'attributs
  - Ne peut être vide
  - Extensible par des contraintes

<xs:element name="name" type="type" />

  - **name** est le nom de l'élément
  - **<xsd:element name="contact" type="xsd:string" />**
  - Les Types les plus utlisés sont : xs:boolean xs:integer xs:date xs:string xs:decimal xs:time
  - D'autres attributs sont possibles
    - **default="default value"**
    - **fixed="value"**
  - Le nombre minimum et maximum d'occurrences est défini par
    - **minOccurs**
    - **maxOccurs** : valeur maximale **unbounded**

Elément complexe

- Un élément complexe est défini par
  - **<xs:element name="*name*">**
    <xs:complexType>
    ... *informations*
    </xs:complexType>
    </xs:element>
- Exemple
  - **<xs:element name="personne">**
    <xs:complexType>
    <xs:sequence>
    <xs:choice>
    <xs:element name="firstName" type="xs:string" />
    <xs:element name="lastName" type="xs:string" />
    </xs:choice>
    </xs:sequence>
    </xs:complexType>
    </xs:element
- <xs:sequence> : les éléments doivent apparaître dans l'ordre
- <xs:choice> : c'set le (ou |) d'une DTD
- <xs:all> : les éléments peuvent apparaître dans n'importe quel ordre (maxOccurs 1)

Les attributs

- Ils sont généralement déclarés comme les éléments simples
  - **<xs:attribute name="*name*" type="*type*" />**
- Un attribut peut avoir trois attributs optionnels : use, default et fixed
- Contraintes d'occurrence

<xsd:attribute name="maj" type="xsd:date" use="optional" default="2022-1-1"/>

  - **use**="optional" | ="required" | ="prohibited"
- Utilisations des attributs (élément vide avec attribut)
  - **<xsd:element name="parution">**
    <xsd:complexType>

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

    </xsd:complexType>
    </xsd:element>

Référencer un élément

- Une fois on définit un élément avec name="..." on peut le référencer par ref="..."

```
<xsd:element name="auteur" type="xsd:string" />
<xsd:attribut name="ISBN" type="xsd:string" />
   <xsd:element name="livre">
     <xsd:complexType>
       <xsd:attribut ref="ISBN" />
          <xsd:sequence>
           <xsd:element ref="auteur" />
            <xsd:element name="titre" type="xsd:string"/>
       </xsd:sequence>
     </xsd:complexType>
   </xsd:element>
```

Restriction sur les types

- La syntaxe générale de mise en œuvre

```
<xs:simpleType  name="name">  (ou xs:complexType)
   <xs:restriction base="type">
        ... Les restrictions ...
    </xs:restriction>
</xs:element>
```

- Exemple

```
<xs:simpleType  name="age">
     <xs:restriction base="xs:integer">
          <xs:minInclusive value="0">
          <xs:maxInclusive value="140">
     </xs:restriction>
</xs:element>
```

Types de restrictions sur les nombres

- Restrictions sur les chiffres
  - minInclusive -- nombre ≥ à la valeur donnée
  - minExclusive -- nombre > à la valeur donnée
  - maxInclusive – nombre ≤ à la valeur donnée
  - maxExclusive – nombre < à la valeur donnée
  - totalDigits – nb de chiffres
  - fractionDigits – nb chiffres après virgule

Types de restrictions sur les châines

- Restrictions sur les Strings
  - length – le nombre de caractères construisant la chaîne
  - minLength -- le nombre de caractères minimum
  - maxLength -- le nombre de caractères maximum
  - pattern – Expression régulière à vérifier par la chaîne
  - whiteSpace – conditions sur l'espace
    - value="preserve" préserver les espaces
    - value="collapse" remplacer tous les espaces par un simple espace

Les patterns

- Contraintes sur type simple prédéfini
- Utilisation d'expression régulières
  - Similaires à celles de Perl
- Exemples
  - **<xsd:simpleType name="CIN">**
    <xsd:restriction base="xsd:string">
    <xsd:pattern value="\[A-Z]{2}d{8}"/>
    </xsd:restriction>
    </xsd:simpleType>

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

Enumération

- Une énumeration fixe l'ensemble des valeurs prise par l'attribut value
- Exemple:
  - <xs:simpleType name="Saison">
    <xs:restriction base="xs:string">
    <xs:enumeration value="Printemps"/>
    <xs:enumeration value="Eté"/>
    <xs:enumeration value="Automne"/>
    <xs:enumeration value="Hiver"/>
    </xs:restriction>
    </xs:simpleType>
    </xs:element>

Autres types d'élément

- Élément texte avec attributs
  - <xsd:element name="parution">
    <xsd:complexType>
    <xsd:simpleContent>
    <xsd:extension base="xsd:string">
    <xsd:attribute name="date" type="xsd:date" />
    </xsd:extension>
    </xsd:simpleContent>
    </xsd:complexType>
    </xsd:element>
- Vous avez aussi la possibilité d'utiliser une restriction

Élément a contenu mixte

- Il peut contenir des éléments des attributs et du texte
  - Il est déclaré par l'attribut **mixed="true"**

```
<xs:element name="auteur">
    <xs:complexType mixed="true">
        <xs:all>
            <xs:element name="nom" type="xs:string"/>
            <xs:element name="né" type="xs:date"/>
        </xs:all>
    </xs:complexType>
</xs:element>
```

Héritage ou extension

- On peut étendre un type complexe de base
-
```
<xs:complexType  name="newType">
   <xs:complexContent>
      <xs:extension  base="autreType">
         ……
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```
- Exemple :

```
<xs:complexType name="AdressePays">
  <xs:complexContent>
    <xs:extension base="Adresse">
      <xs:sequence>
        <xs:element name="pays" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Portée d'un élément

- Un élément déclaré au début d'un schémas <schema> a une portée globale
- Les éléments déclarés à l'intérieur d'un type complexe sont locales
- ```
<xs:element name="personne">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="firstName" type="xs:string" />
            <xs:element name="lastName" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```
  - firstName et lastName sont déclarés localement
- Pour utiliser le type personne
  - <xs:element name="élève" type="personne"/>
  - <xs:element name="professeur" type="personne"/>
```

Réutilisation de déclarations

- Possibilité de référencer un élément plus global
  - <element ref="Nom" /> (ci-dessus)
  - Importe l'élément et son type
- Possibilité d'inclure les types associés à un espace de noms
  - <import nameSpace = "http:// ..."
  - schemaLocation = "http:// ..." />
- Possibilité d'étendre un schéma
  - <redefine schemaLocation="http:// ..."/>
  - .... Extensions ...
  - </redefine>

Quelques règles d'écriture

- Modularité
  - définir dans des types séparés les parties réutilisables
- Précédence
  - Regrouper les déclarations de types en tête
- Abstraction
  - Utiliser des entités ou types pour les modèles de contenus
- Spécificité
  - Éviter les types trop généraux
- Simplicité
  - Découper les schémas ou DTD trop complexes

4. Les outils de développement

- IDE = Integrated Developement Environment
- Des éditeurs
  - De texte XML, parfois avec structure séparée
  - De schéma XML, avec interface graphique
  - De règles XSL, avec moteur de transformation
  - De requêtes XQuery, avec moteur sur document
- Des interfaces XML
  - Aux fichiers
  - Aux bases de données
  - Aux applications
- Le plus fameux : XML Spy

Quelques outils de travail

Editeur	Outil	Support
Tibco	Turbo XML	DTD, XSL, XQuery Schéma
Altova	XMLSpy	DTD, Schéma XSL, XQuery
SyncRO Ltd.	Oxygen	DTD, Schéma XSL, XQuery
Data Junction	XML Junction	Schéma
Insight Soft.	XMLMate	DTD, Schéma, XSL, XPath
XML Mind	XMLMind Editor	DTD, Schéma, XSL, XPath