

XSD

Plan

1. Introduction
2. Type simple
3. Type Complexe
4. Déclaration

<http://www.w3.org/TR/xmlschema11-1/>
(recommandation 05/04/2012)

XML Schema

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

Objectifs des schémas

- Reprendre les acquis des DTD
 - Plus riche et complet que les DTD
- Permettre de typer les données
 - Éléments simples et complexes
 - Attributs simples
- Permettre de définir des contraintes
 - Existence, obligatoire, optionnel
 - Domaines, cardinalités, références
 - Patterns, ...
- S'intégrer à XML
 - Espace de noms

Structure de base

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

Prologue

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Élément racine

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

```
  <!-- déclarations d'éléments, d'attributs et de types ici -->
</xs:schema>
```

Déclaration d'éléments

- Un élément, dans un schéma, se déclare avec la balise `<xs:element>`
 - indiquer le nom de l'élément avec l'attribut `name` ;
 - préciser le nombre d'apparition autorisé pour cet élément à l'aide des attributs `minOccurs` et `maxOccurs` ;
 - définir le type du contenu de l'élément en utilisant l'attribut `type` ;
- Un élément peut être de **type simple** ou de **type complexe**

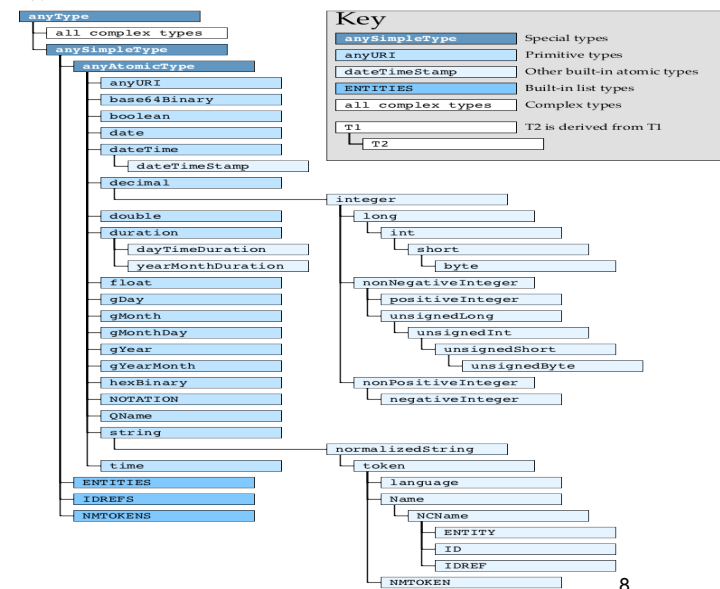
Les types simples (1)

Les **types de données simples** ne peuvent comporter ni attributs, ni éléments enfants.

Les **types simples** incluent les types de base définis dans la bibliothèque standard des XML Schema, les listes et les unions. Mais il est également possible d'en "dériver" de nouveaux.

Les types simples (2)

- Les types prédéfinis
xs:string, xs:integer (voir diapo suivante)
- Les listes
- Les unions



Listes

Liste de valeurs d'un même type simple :

```
<xs:simpleType name="telephone">
  <xs:list itemType="xs:integer" />
</xs:simpleType>
```

XML Schema possède trois types de listes intégrés :
NMTOKENS, ENTITIES et IDREFS.

Union

❓ Les listes et les types simples intégrés ne permettent pas de choisir le type de contenu d'un élément. On peut désirer, par exemple, qu'un type autorise soit un nombre, soit une chaîne de caractères particuliers. Il est possible de le faire à l'aide d'une déclaration d'union.

Exemple

```
<xs:simpleType name="contact">
  <xs:union memberTypes="xs:string telephone" />
</xs:simpleType>
```

Restriction d'un type simple (1)

Cela permet d'ajouter des contraintes à un type de base :

```
<xs:simpleType name="Temperature">  
  <xs:restriction base="xs:integer">  
    <xs:minInclusive value="-15"/>  
    <xs:maxInclusive value="+40"/>  
  </xs:restriction>  
</xs:simpleType>
```

Restriction d'un type simple (2)

Facettes pour chaque type, permet de :

- fixer, restreindre ou augmenter la longueur minimale ou maximale d'un type simple ex. pour xs:string (xs:length, xs:minLength, xs:maxLength)
- énumérer toutes les valeurs possibles d'un type (xs:enumeration)
- prendre en compte des expressions régulières (xs:pattern)
- fixer la valeur minimale ou maximale d'un type
- fixer la précision du type...

Les patterns

- Contraintes sur type simple prédéfini
- Utilisation d'expression régulières
 - Similaires à celles de Perl

Exemple

```
<xs:simpleType name="NumItem">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="\d{3}-[A-Z]{2}"/>  
  </xs:restriction>  
</xs:simpleType>
```

\d{3} = un chiffre répété 3 fois

[A-Z]{2} = une lettre répétée 2 fois

exemple "444-AA"

13

Type complexe: type de contenu ?

- Contenu (simple ou vide) avec attribut(s)
- Contenu mixte
- Contenu complexe

14

Contenu avec attribut

Vide

```
<xs:complexType name="TypeImg">  
  <xs:attribute name="href" type="xs:string" />  
</xs:complexType>
```

Simple

```
<xs:complexType name="TypeTitre">  
  <xs:simpleContent>  
    <xs:extension base="xs:string">  
      <xs:attribute name="lang" type="xs:string" />  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>
```

Contenu Mixte



Afin de spécifier qu'un élément peut contenir également du texte, on utilise l'attribut `mixed` de l'élément `xs:complexType`. Par défaut, `mixed="false"`; il faut dans ce cas forcer `mixed="true"`.

Par exemple :

```
<xs:complexType mixed="true" name="TypeeltMixte">  
  <xs:sequence>  
    <xs:element name="first" type="xs:string"/>  
    <xs:element name="last" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>
```


Contenu complexe : Comment ses éléments apparaissent?

- Connecteur **sequence** <xs:sequence> : les sous-éléments doivent tous apparaître, dans l'ordre ;
- Connecteur de choix **choice** <xs:choice> : seulement un des sous-éléments peut apparaître, au choix ;
- Connecteur **all** <xs:all> : les sous-éléments doivent tous apparaître, mais dans un ordre quelconque

Déclaration des attributs

- À la différence des éléments, un attribut ne peut être que de type simple. Cela signifie que les attributs, comme avec les DTD, ne peuvent contenir d'autres éléments ou attributs.
- Utilisation de la balise *attribute*
- Indiquer le nom avec l'attribut *name*
- Définir le type avec l'attribut *type*
- Préciser le caractère obligatoire (required) ou optionnel (optional) avec l'attribut *use*
- Eventuellement indiquer une valeur par défaut avec l'attribut *default*

Stratégies de déclaration

- En suivant l'arborescence des documents
- A plat avec référence aux éléments déjà définis

Déclaration d'un fichier xsd dans le fichier xml

Fichier mel.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespacesSchemaLocation="mel.xsd">
  </mel>
```


Mieux


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://mondomain.org/namespace/ mel.xsd"
      xmlns:monMel="http://mondomain.org/namespace/">
  </mel>
```

mel.xsd


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

Validation

 Utilisation de la commande xmllint sous linux

 Valider une dtd vis à vis d'un fichier xml

```
xmllint --dtdvalid fich.dtd fich.xml
```

 Valider un fichier xsd vis à vis d'un fichier xml

```
xmllint --schema mel.xsd mel.xml
```

 Il existe des validateurs sur le Web