TP 3 : Validation des documents XML -Schéma

Objectif : valider un document XML

Lab. 1 : la validation par schéma :

Rappel : Un fichier XML est dit « valide » lorsqu' 'il satisfait à l'ensemble des contraintes imposées par son grammaire de validation (DTD, Schéma)

Un schéma W3C (que l'on nommera par la suite schéma) est une grammaire définie dans un **formalisme XML**.

On peut considérer les schémas comme remplaçant les DTD.

Quelques caractéristiques des schémas :

- types de base riches et extensibles ;
- réutilisation par importation et héritage ;
- davantage de souplesse dans les cardinalités ;

1. La racine d'un schéma :

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

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

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

L'élément racine est l'élément xsd:schema.

2. Déclaration des éléments :

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

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

<xsd:element name="Personne" type="typePersonne"></xsd:element>
<xsd:element name="nom" type="xsd:string"></xsd:element>
<!-- déclarations de types ici -->
</xsd:schema>
```

Le schéma précédent déclare deux éléments : un élément Personne et un élément nom.

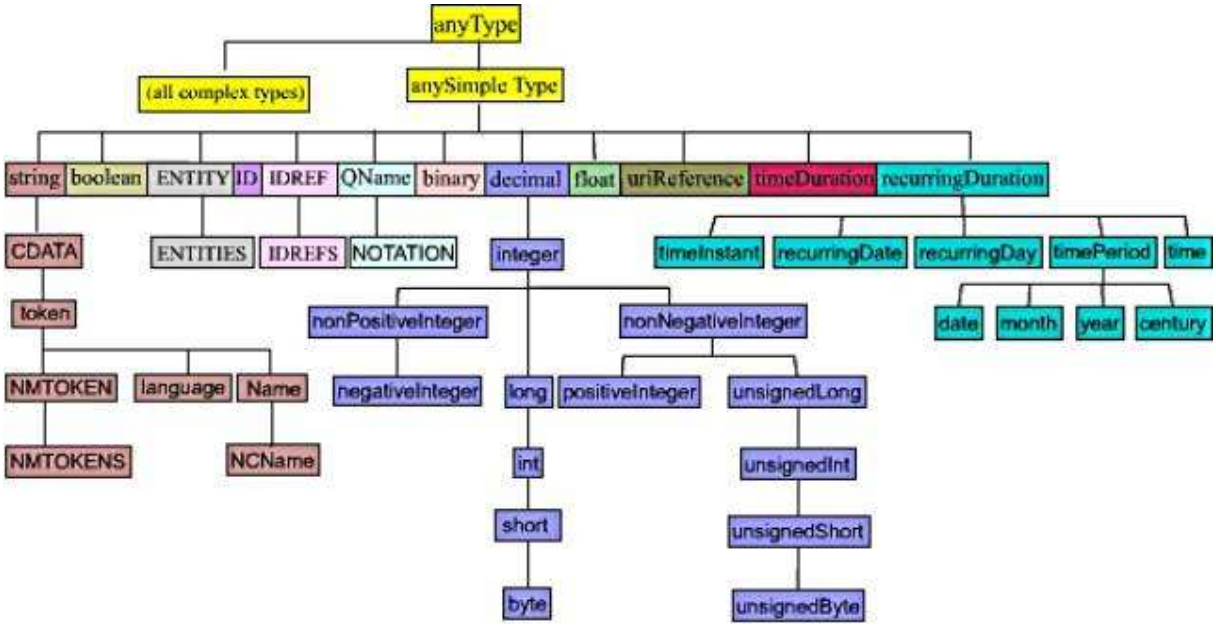
Chaque élément est "typé" -c'est-à-dire qu'il doit respecter un certain format de données. L'élément Personne est ainsi du type typePersonne, qui est un type complexe défini par l'utilisateur. L'élément nom quant à lui est du type xsd:string qui est un type simple prédéfini de XML Schema.

Chaque élément déclaré est associé à un type de données via l'attribut type**. Les éléments pouvant contenir des élément-enfants ou posséder des attributs** sont dits de type *complexe*, tandis que les éléments n'en contenant pas sont dits de type *simple*.

a. Les types simples :

Les types de données simples ne peuvent comporter ni attributs, ni éléments enfants. Il en existe de nombreux prédéfinis.

Nombreux sont les types prédéfinis dans la bibliothèque de types intégrés de XML Schéma. La figure suivante en donne la hiérarchie, et leur liste détaillée figure sur le site du W3C.



Exemple :

On pourra envisager, par exemple, dans un schéma décrivant un bon de commande, la déclaration d'un élement quantite.

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

Qui force la valeur de l'élément à un être un entier positif. Un bon de commande XML suivant ce schéma, ayant une commande spécifiant quantite="-3" sera alors automatiquement refusé par le système.

b. Les types complexes :

Un élément de type simple ne peut contenir de sous-élément. Il est nécessaire pour cela de le déclarer de type "complexe". On peut alors déclarer, des séquences d'éléments, des types de choix ou des contraintes d'occurrences.

Dans un schéma, le type complexe nécessite toujours une balise complexType. Le type peut être global et donc associé à un nom ou bien être local à un élément.

Quelques exemples :

```
<auteur/> : pas un type complexe
<auteur>Mr Dupond</auteur> : pas un type complexe
<auteur id="10"/> : type complexe
<auteur id="10">Mr Dupond</auteur> : type complexe
<auteur><titre>Mr</titre><nom>Dupond</nom></auteur> : type complexe
```

- **La séquence :**

La séquence caractérise des éléments fils présents dans un ordre donné.

Exemple :

```
<xs:element name="Personne">
<xs:complexType>
<xs:sequence>
<xs:element name="nom" type="xs:string"/>
<xs:element name="prenom" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Dans cet exemple, l'élément plan contient nécessairement 2 éléments fils de contenu simple auteurs et chapitres.

Nous aurions pu également l'écrire avec un type global (par exemple PersonneType) :

```
<xsd:element name="Personne" type="typePersonne">
…
<xs:complexType name="typePeronne">
<xs:sequence>
<xs:element name="nom" type="xs:string"/>
<xs:element name="prenom" type="xs:string"/>
</xs:sequence>
</xs:complexType>
```

- **Le choix:**

Le choix, comme son nom l'indique, étend les possibilités de constituer des documents XML valides en proposant des alternatives d'éléments.

Exemple :

```
<xs:element name="Personne">
<xs:complexType>
<xs:choice>
<xs:element name="nom" type="xs:string"/>
<xs:element name="prenom" type="xs:string"/>
```

```
</xs:choice>
</xs:complexType>
</xs:element>
```

Ce qui autorisera un élément nom ou bien un élément prénom dans l'élément Personne.

- **Tout :**

Le connecteur all est propre au schéma et caractérise un ensemble d'éléments de présence obligatoire mais sans contrainte sur l'ordre (toutes les permutations sont donc valides).

Exemple :

```
<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>
```

Dans cet exemple l'élément Personne contiendra les éléments nom et prenom dans n'importe quel ordre.

- **Les cardinalités**

Tout comme dans les DTD, les cardinalités sont possibles sur les éléments.

Ces cardinalités sont positionnées par les attributs minOccurs et maxOccurs. L'infini est caractérisé par la chaîne unbounded.

Pour faire le parallèle avec les DTD, nous retrouvons l'équivalent des opérateurs ?, + et * avec :

? : minOccurs="0" maxOccurs="1" ;
+ : minOccurs="1" maxOccurs="unbounded" ;
* : minOccurs="0" maxOccurs="unbounded".

Lorsqu'on ne précise rien, minOccurs et maxOccurs ont la valeur 1.

Exemple sur un élément :

```
<xs:element name="plan">
<xs:complexType>
<xs:sequence>
<xs:element name="auteur" type="xs:string"
maxOccurs="unbounded"/>
<xs:element name="chapitre" type="xs:string"
minOccurs="2" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Dans cet exemple, l'élément plan contient un élément auteur suivi d'au moins 2 éléments chapitre.

Exemple sur un connecteur :

```
<xs:element name="plan">
<xs:complexType>
<xs:sequence minOccurs="2" maxOccurs="3">
<xs:element name="auteur" type="xs:string"/>
<xs:element name="chapitre" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Avec cet exemple, nous contrôlons que l'élément plan contient entre 2 et 3 suites d'éléments auteur et chapitre.

3. Déclaration des attributs :

A 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. De plus, les déclarations d'attributs doivent être placées *après* les définitions des types complexes, autrement dit, après les éléments <xsd:sequence>, <xsd:choice> et <xsd:all>.

Exemple :

```
<xs:element name="personne"
<xs:complexType>
…
<xs:attribute name="nom" type="xs:string"/>
</xs:complexType>
</xs:element>
```

Dans cet exemple, nous associons à l'élément personne l'attribut nom. L'attribut nom est local à l'élément personne ; il ne peut pas être employé dans un autre élément.

La présence d'un attribut peut être définie par l'attribut use, qui peut prendre les valeurs suivantes :

- prohibited : interdire l'usage d'un attribut par dérivation d'un type complexe.
- optional : l'attribut n'est pas obligatoirement renseigné (employé par défaut).
- required : l'attribut est obligatoire.

Deux autres attributs, default et fixed, servent à définir respectivement une valeur par défaut, si l'attribut est optionnel, et une valeur obligatoire.

Voici un exemple :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="a">
<xs:complexType>
<xs:attribute name="t1" use="required" type="xs:int"/>
<xs:attribute name="t2" use="optional" type="xs:string" default="valeur »
```

```
<xs:attribute name="t3" use="required" type="xs:token" fixed="autre"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

Remarque :

L'élément vide n'a pas de contenu ; sa définition est donc sans typage. On l'écrit, par exemple :

```
<xs:element name="br"/>
```

La balise br ne peut donc s'exprimer que sous cette forme
.

Un contenu mixte sert à faire cohabiter du texte et des éléments. On positionne un attribut mixed dans un type complexe pour obtenir l'effet recherché.

Un exemple :

```
<xs:complexType name="personType" mixed="true">
<xs:sequence>
<xs:element name="nom" type="xs:string"/>
<xs:element name="prenom" type="xs:string"/>
</xs:sequence>
</xs:complexType>
<xs:element name="employe" type="personType"/>
```

On pourrait donc, par exemple, écrire ce document XML :
<employe>Bonjour <nom>MrDupont</nom>,<prenom>Jean</prenom> de Paris</employe>

Lab. 3 : Mise en pratique

Exercice 1 :

Soit un document XML contenant un nombre indéterminé d'éléments sous la forme :

```
<contact titre="..." techno="...">
<nom>...</nom>
<prenom>...</prenom>
<telephone> ...</telephone>
<email>...</email>
<email>...</email>
...
</contact>
```

L'élément telephone et l'attribut techno sont en option. Les textes seront des chaînes simples xs:string.
Vous utiliserez les types complexes numerosType et contactType pour construire un schéma nommé annuaire.xsd.

Exercice 2 :

Rédiger le schéma XML pour le document livre.xml vu dans le TP1

Exercice 3 :

Rédiger le schéma XML pour le document annuaire.xml vu dans le TP1

Exercice 4:

1. Rédiger un Schéma XML pour une bibliographie. Cette bibliographie Contient des livres et des articles.
    a. Les informations nécessaires pour un livre (élément livre) sont :

    - Son titre général (élément titre) ;
    - Les noms des auteurs (éléments auteur) ;
    - Ses tomes (élément tomes) et pour chaque tome (éléments tome), leur nombre de pages (élément pages) ;
    - Des informations générales sur son édition (élément infosEdition) comme par exemple le nom de l'éditeur (élément editeur), le lieu d'édition (élément lieuEdition), le lieu d'impression (élément lieuImpression), son numéro ISBN (élément ISBN) ;

    b. Les informations nécessaires pour un article (élément article) sont :

    - Son titre (élément titre) ;
    - Les noms des auteurs (éléments auteur) ;
    - Ses références de publication (élément infosPublication) : nom du journal (élément nomJournal), numéro des pages (élément pages), année de publication (élément anneePublication) et numéro du journal (élément numéroJournal)

    c. On réservera aussi un champ optionnel, pour chaque livre et chaque article, pour un avis (élément avis) personnel.
2. Tester ce Schéma XML avec un fichier XML que l'on écrira et validera.

Exercice 5 :

1. Modifier le Schéma précédent. On ne déclarera, pour le moment, que des types de chaînes de caractères.

    a. Ajouter un attribut optionnel soustitre à l'élément titre
    b. Faisant l'élément tome un élément vide
    c. Ajouter a ce dernier un attribut requis nbPages et un attribut optionnel sousTitre ;
    d. Faisant de l'élément nomJournal un attribut de l'élément infosPublication et en lui donnant comme valeur par défaut Feuille de Chou ;

2. Utiliser ce Schéma pour créer un fichier XML valide.

Exercice 6 :

Écrire un schéma pour des documents XML - examen.

- Un examen contient un code de cours, un titre et une date composée du mois et de l'année.
- Ces éléments sont suivis par une liste de questions.
- Un examen a de 5 à 6 questions
- chaque question a une ou plusieurs parties.
- Une partie est un mélange de texte et d'autres parties.
- La valeur de mois doit être une chaîne de caractère valide.

Donnez une instance de document valide par rapport à ce schéma.