Chapitre VII

Formalisation - Espaces de nom en XMLSchema

1 Un schéma = un vocabulaire

Un XMLSchema (schema XSD) définit :

- un nouveau vocabulaire
- un ensemble de règles (syntaxe + sémantique) qui précise comment on doit utiliser les mots du vocabulaire (i.e. comment ils s'organisent les uns par rapport aux autres)

un XSD permet de définir un nouveau langage de description d'un système d'information (de la même façon que le diagramme de classe en UML).

Il existe d'autres façons de définir des langages XML (DTD, RelaxNG) ou autre (grammaire BNF).

XML signifie eXtensible Markup Language. Une des signification de l'extensibilité est qu'XML facilite les mélanges de vocabulaires. Cela le rend extrêmement puissant (par exemple : mélange de xhtml et de svg). En contrepartie, il doit supporter un mécanisme universel permettant de conserver la cohérence entre les vocabulaires. On doit à tout moment connaitre le vocabulaire qui est en train d'être utilisé. Par exemple, l'élément `set` représente un ensemble mathématique dans le langage XML `MathML` et fixe la valeur d'un attribut à une durée de temps précise dans le langage XML `SVG`. Or on souhaite pouvoir utiliser les langages `MathML` et `SVG` dans une même page `XHTML`.

Dans ce chapitre, on va voir :

- comment donner un nom à notre nouveau langage
- comment utiliser plusieurs langages au sein d'un même document XML sans entrainer de confusion.

2 Exemple des températures

2.a Exemple XML

On considère l'instance XML de la figure VII.1 sur les relevés des températures à un jour donné.

2.b XMLSchema

On souhaite décrire ce langage XML sous forme d'un XMLSchema en précisant que les valeurs des températures devront être entières et comprises entre -70 et +80 degrés Celsius.

On décide donc d'écrire un type simple `intDeg` dans le schéma de la figure VII.2 pour décrire une valeur possible de température.

```xml
<?xml version="1.0" encoding="utf8" ?>
<température>
    <min>-6</min>
    <max>2</max>
</température>
```

Figure VII.1: Instance XML de relevé de température.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema>
    <element name="température" type="Température"/>
    <complexType name="Température">
        <sequence>
            <element name="min" type="IntDeg"/>
            <element name="max" type="IntDeg"/>
        </sequence>
    </complexType>

    <simpleType name="IntDeg">
        <restriction base="int">
            <minInclusive value="-70"/>
            <maxInclusive value="80"/>
        </restriction>
    </simpleType>
</schema>
```

Figure VII.2: Schema XML de relevé de température pour valider l'instance VII.1.

2.c Espaces de noms

Dans l'exemple précédent, nous avons regroupé par couleurs les éléments de la figure VII.2. En effet, nous pouvons distinguer 2 types d'éléments dans ce schéma:

- les éléments propres au vocabulaire XMLSchema (aussi appelé xsd), comme la racine du document: `schema`
- les éléments propres au vocabulaire que l'on est en train de définir et qui seront les noms des éléments de nos instances, comme `température`.

Espace de noms XMLSchema Défini par le W3C	Espace de noms de notre langage Défini par nous
schema element, sequence complexType, simpleType int	température min, max Température IntDeg

On peut remarquer que dans les 2 vocabulaires, on a des noms d'éléments (`schema`, `element`, `complexType`, `simpleType`, `température`, `min`, `max`) ainsi que des noms de types (`int`, `Température`, `IntDeg`).

3 Définition du vocabulaire et de son nom

3.a Noms de vocabulaires

Nous avons vu dans l'exemple précédent que nous avons affaire à 2 vocabulaires/espaces de noms/langages. Pour les reconnaître, nous devons les nommer.

Par convention, et pour garantir l'extensibilité universelle du XML, on choisi d'utiliser, comme nom de vocabulaire/langage XML des URI. URI signifile *Uniform Resource Identifier*. les URLs (*Uniform Resource Locator*) que vous connaissez pour accéder à un site web via un navigateur sont un sous-ensemble des URI. On a donc souvent des noms de vocabulaires/langages XML qui *ressemblent* à des URLs.

Une URI peut être fictive (si vous tapez une URI nom de vocabulaire XML dans une page web, il n'y a souvent rien au bout), mais fait souvent référence à l'organisme garant du vocabulaire en question. Par exemple, de nombreux langages XML développés par le W3C ont un nom qui commence par `http://www.w3.org`. Par exemple, pour le nom de vocabulaire du xhtml est `http://www.w3.org/1999/xhtml`, le vocabulaire du XMLSchema, aussi appelé XSD a pour nom `http://www.w3.org/2001/XMLSchema`, etc.

Dans ce cours, nous utiliserons souvent des URI qui commencent par `http://www.ujf-grenoble.fr/...`.

3.b En UML

En UML, les espaces de noms sont représentés comme des *packages* autour des éléments correspondants.

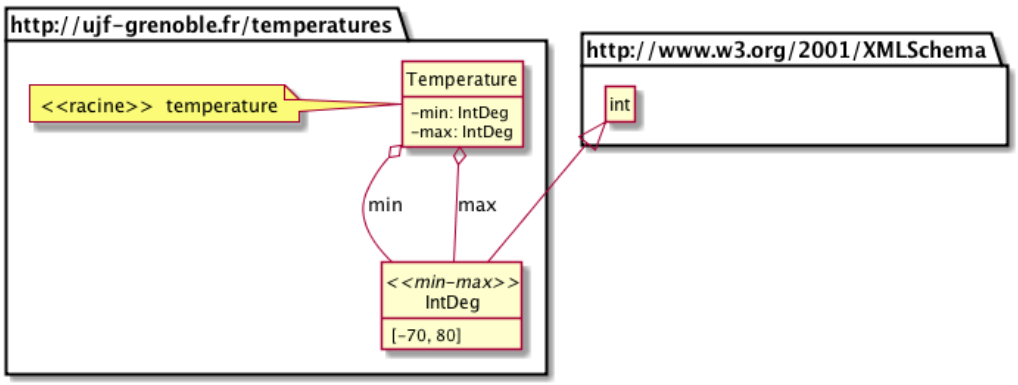

Figure VII.3: Exemple de diagramme UML avec espaces de noms

3.c En XMLSchema

On attribue un nom au vocabulaire, au même endroit où on le définit : dans le schema. Dans l'exemple VII.2, la racine du schéma devient:

```
1 <schema targetNameSpace="nomDuNouveauVocabulaire"     (1)
2         xmlns="nomDuNouveauVocabulaire (ParDéfaut)"   (2)
3         xmlns:xsd="http://www.w3.org/2001/XMLSchema"  (3)
4         elementFormDefault="qualified">               (4)
5  ...
6 </schema>
```

ou

```
1 <schema targetNameSpace="nomDuNouveauVocabulaire"     (1)
2         xmlns:ns="nomDuNouveauVocabulaire"            (2 bis)
3         xmlns:xsd="http://www.w3.org/2001/XMLSchema"  (3)
4         elementFormDefault="qualified">               (4)
5  ...
6 </schema>
```

(1) `targetNamespace` définit le nom du vocabulaire, c'est-à-dire du nom du nouveau langage que l'on est en train de décrire. Les éléments qui seront définis par ce schéma (par exemple `température`, `min`, `max`) appartiennent à cet espace de noms. Dans le cas de l'exemple de la figure VII.2, `nomDuNouveauVocabulaire` deviendrait `http://www.ujf-grenoble.fr/temperatures`.

(2) Nom du langage par défaut, c'est-à-dire du langage auquel appartiendront tous les mots qui ne seront pas préfixés. Le langage par défaut est souvent (mais pas toujours) choisi identique au `targetNamespace`. Dans le cas de l'exemple de la figure VII.2, il s'agirait également de `http://www.ujf-grenoble.fr/temperatures`

(2 bis) Nom du nouveau vocabulaire, ici préfixé. Il est identique au `targetNamespace`. Dans le cas de l'exemple de la figure VII.2, il s'agirait également de `http://www.ujf-grenoble.fr/temperatures`

(3) Définition du préfixe `xsd`. De manière générale, `xmlns` signifie *XML Name Space* et `xmlns:pref` signifie la définition d'un préfixe `pref`. Ici, `xmlns:xsd="http://www.w3.org/2001/XMLSchema"` signifie que tous les mots qui seront préfixés par `xsd:` appartiendront au vocabulaire XMLSchema (i.e. à l'URI `http://www.w3.org/2001/XMLSchema`).

(4) `elementFormDefault` est une directive pour tous les documents instances voulant se conformer à ce schéma. Elle force ne effet les documents instances à qualifier (préfixer) l'espace de nom de tous les éléments de ce vocabulaire.

Remarque : Votre Schema XML doit TOUJOURS contenir au moins :

- 1 `targetNamespace`. `targetNamespace` signifie "voici le nom du vocabulaire que **je définis**"
- 2 `xmlns` (dont 1 seul peut ne pas être préfixé). Il peut y en avoir davantage. `xmlns` signifie "voici le nom du vocabulaire que **j'utilise**"

3.d Retour à l'exemple

Le schéma présenté figure VII.2 devient alors:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsd:schema targetNamespace="http://www.ujf-grenoble.fr/temperatures"
3              xmlns="http://www.ujf-grenoble.fr/temperatures">
4              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5              elementFormDefault="qualified">
6      <xsd:element name="température" type="Température"/>
7      <xsd:complexType name="Température">
8          <xsd:sequence>
9              <xsd:element name="min" type="IntDeg"/>
10             <xsd:element name="max" type="IntDeg"/>
11         </xsd:sequence>
12     </xsd:complexType>
13
14     <xsd:simpleType name="IntDeg">
15         <xsd:restriction base="xsd:int">
16             <xsd:minInclusive value="-70"/>
17             <xsd:maxInclusive value="80"/>
18         </xsd:restriction>
19     </xsd:simpleType>
20 </xsd:schema>
```

Figure VII.4: Schema XML de relevé de température pour valider l'instance VII.1 avec espaces de noms.

3.e Cas où le vocabulaire par défaut n'est pas le vocabulaire défini

On a vu plus haut que le vocabulaire défini par défaut, c'est-à-dire sans préfixe (par `xmlns="nomDuVocabulaire"` était souvent le vocabulaire du `targetNamespace`. Mais ce n'est pas obligatoire. Par exemple, on peut définir XMLSchema comme vocabulaire par défaut. En réalité, cette situation est souvent plus pratique car cela évite d'avoir à écrire le préfixe associé au vocabulaire du schéma de partout. Comme il ne peut y avoir qu'un seul vocabulaire par défaut, on doit alors préfixer le vocabulaire `temperature`. Ceci donnerait par exemple le schéma de la figure VII.5:

On peut remarquer dans ce schéma:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <schema targetNamespace="http://www.ujf-grenoble.fr/temperatures"
3              xmlns:tmp="http://www.ujf-grenoble.fr/temperatures">
4              xmlns="http://www.w3.org/2001/XMLSchema"
5              elementFormDefault="qualified">
6      <element name="température" type="tmp:Température"/>
7      <complexType name="Température">
8          <sequence>
9              <element name="min" type="tmp:IntDeg"/>
10             <element name="max" type="tmp:IntDeg"/>
11         </sequence>
12     </complexType>
13
14     <simpleType name="IntDeg">
15         <restriction base="int">
16             <minInclusive value="-70"/>
17             <maxInclusive value="80"/>
18         </restriction>
19     </simpleType>
20 </schema>
```

Figure VII.5: Schema XML de relevé de température pour valider l'instance VII.1 avec espaces de noms.

- les éléments du vocabulaire `http://www.w3.org/2001/XMLSchema` (`schema`, `element`, `complexType`, etc.) appartiennent au vocabulaire par défaut et ne sont donc plus préfixés par `xsd:`.
- de même le type `int` ligne 15 appartient au vocabulaire par défaut et n'est donc plus préfixé par `xsd:`.
- la définition du nom des types (complexes ou simples) du vocabulaire en cours de définition n'a pas besoin d'être préfixé. En effet, lignes 7 et 14, on sait que l'on est en train de définir des types du vocabulaire en cours de définition et par conséquent, `Température` ligne 7 et `IntDeg` ligne 14 appartiennent forcément au `targetNamespace` et donc au vocabulaire `http://www.ujf-grenoble.fr/temperature`
- de même, le nom des éléments qui sont en train d'être définis (lignes 6, 9 et 10) n'ont pas besoin d'être préfixés car ils appartiennent obligatoirement au `targetNamespace`.

3.f Peut-il n'y avoir aucun vocabulaire par défaut dans un XMLSchema?

Oui, bien sûr ! On peut décider de tout préfixer. Mais comme dit précédemment, il est souvent astucieux de poser le vocabulaire XMLSchema comme vocabulaire par défaut et préfixer le vocabulaire que l'on définit.

4 Contraindre un document XML à un schema

On souhaite ici contraindre un document XML à un schéma, c'est-à-dire lier le document XML à un schéma de façon à ce que sa validité par rapport au schéma puisse être testée automatiquement.

```
<racine xmlns="nomDuVocabulaireParDéfaut"                          (1)
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      (2)
        xsi:schemaLocation="NomDuVocabulaireUtilisé schema.xsd"> (3)
 ...
</racine>
```

(1) `xmlns="nomDuVocabulaireParDéfaut"` signifie ici dans l'instance, la même chose que dans le schéma, c'est-à-dire qu'il désigne le vocabulaire par défaut, celui auquel appartiennent les éléments non préfixés. Il s'agit en général du vocabulaire défini dans le schéma.

(2) `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` définit le préfixe `xsi` pour désigner le langage XMLSchema-Instance (note, c'est un langage différent de XMLSchema). Ce préfixe sert en fait juste pour la ligne d'après étant donné que l'attribut `schemaLocation` de la racine n'a pas été défini dans notre vocabulaire mais appartient à XMLSchema-Instance

(3) L'attribut `schemaLocation` (appartement au langage XMLSchema-Instance) permet de spécifier le fichier `xsd` dans lequel est défini le vocabulaire `NomDuVocabulaireUtilisé`. On peut noter que le nom du vocabulaire et le nom du fichier sont dans les même guillemets de valeur de l'attribut `schemaLocation` séparés par un espace. Le nom du fichier est en réalité un chemin relatif pour accéder au schéma depuis l'instance. Par exemple, si le schéma était situé dans le répertoire parent de celui dans lequel l'instance était situé, on aurait: `xsi:schemaLocation="NomDuVocabulaireUtilisé ../schema.xsd"`.

4.a Retour à l'exemple

Supposons que le schéma de la figure VII.4 (ou de la figure VII.5 qui est équivalent) est situé dans le même répertoire que l'instance `temperatures.xml` sous le nom `temperatures.xsd`.

On aurait alors le document suivant:

```
<?xml version="1.0" encoding="UTF-8"?>
<temperature xmlns="http://ujf-grenoble.fr/temperatures"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://ujf-grenoble.fr/temperatures
                 temperatures.xsd">
    <min>-6</min>
    <max>2</max>
</temperature>
```

Ce document va pouvoir être validé automatiquement par rapport au schéma du fichier `temperatures.xsd` par NetBeans par exemple.

4.b Cas où le vocabulaire par défaut n'est pas le vocabulaire utilisé

On peut souhaiter ne pas utiliser de vocabulaire par défaut. A ce moment, on utilise un préfixe en utilisant `xmlns:pref="NomDuVocabulairePréfixé"` pour définir le préfixe `pref`.

On obtient alors cet exemple d'instance XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<tmp:temperature xmlns:tmp="http://ujf-grenoble.fr/temperatures"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://ujf-grenoble.fr/temperatures
                 temperatures.xsd">
    <tmp:min>-6</tmp:min>
    <tmp:max>2</tmp:max>
</tmp:temperature>
```

4.c Peut-il n'y avoir aucun vocabulaire par défaut dans l'instance de document XML ?

Oui, encore une fois, bien sûr ! On peut décider de tout préfixer. Cette stratégie est même ABSOLUMENT OBLIGATOIRE lorsqu'on écrit des feuilles de transformation XSLT : il FAUT préfixer le vocabulaire XSLT **et** tous les autres vocabulaires utilisés (ceux du ou des documents XML sur lesquels s'appliquent la transformation).

5 A qui appartiennent les éléments/attributs ?

Voici à présent quelques règles pour identifier les espaces de noms (vocabulaires) des éléments:

- Les éléments sans préfixe sont dans le vocabulaire par défaut (défini par `xmlns="NomDuVocabulaireParDéfaut")`.
- Si aucun vocabulaire par défaut n'est défini, les éléments sans préfixe sont sans vocabulaire (sans espace de nom ou *namespace*).
- Les attributs sans préfixe sont **sans espace de nom**, même s'ils appartiennent à une balise qui a un espace de nom.

Nous verrons que certains préfixes sont canoniques (c'est-à-dire qu'ils sont *réservés* à certains vocabulaires) comme `xsd`, `rdf`, `svg`, etc.

5.a Règles pour connaître l'espace de noms d'un élément

- Si l'élément a un préfixe,
  - On cherche la définition du préfixe en remontant vers les ancêtres de l'élément.
  - S'il n'y a pas de définition du préfixe dans les ancêtres, c'est qu'il y a une **erreur**.
- Si l'élément n'a pas de préfixe,
  - On cherche la définition du vocabulaire par défaut en remontant vers les ancêtres de l'élément.
  - S'il n'y a pas de définition du vocabulaire par défaut, l'élément n'a **pas d'espace de nom**.

5.b Règles pour connaître l'espace de noms d'un attribut

- Si l'attribut a un préfixe
  - On cherche la définition du préfixe en remontant vers les ancêtres
  - Si le préfixe n'est pas défini, il y a une **erreur**
- Si l'attribut n'a pas de préfixe, **il n'a pas d'espace de nom**.