haute école neuchâtel berne jura arc gestion neuchâtel delémont

Hes·so Haute Ecole Spécialisée de Suisse occidentale

Certifié ISO 9001

62-31.3 XML

XML-02-P06-Derivations

Cédric Benoit

Plan du cours

- Présentation et organisation
- Introduction à XML
- Schémas XML
- **Types de données**
  - Types prédéfinis
  - Création de types simples
  - Patterns
  - Création de types complexes
  - **Création de types dérivés**

Points abordés

- Introduction
- Types simples
  - Dérivation par Restriction
  - Dérivation par Liste
  - Dérivation par Union
- Remarques et conseils
- Types complexes

Introduction

- Création de nouveaux types
- Pour les types simple Trois mécanismes pour définir un nouveau type en "dérivant" un type existant
  - Restriction : contraintes supplémentaires
  - List : liste de valeurs autorisées
  - Union : liste de valeurs d'autres types

- Types complexes: 4 classes de types complexes
  - SimpleContent
  - ComplexContent
  - MixedContent
  - EmptyContent

Types simples - Restrictions multiples (1)

Il est possible de définir un nouveau type par restriction en se basant sur un type déjà défini par restriction.

Si la nouvelle restriction affecte une nouvelle facette, elle est simplement ajoutée à l'ensemble de base, comme **string**, etc. (on fait une union des restrictions).

Si la nouvelle restriction affecte une ancienne facette, les choses se compliquent.

Types simples - Restrictions multiples (2)

Pour les facettes suivantes :

enumeration, fractionDigits, maxExclusive, maxInclusive, maxLength, minExclusive, minInclusive, minLength et **totalDigits**.

Il est interdit d'étendre le domaine de validité

Types simples - Restrictions multiples (3)

Exemple avec une restriction se basant sur un type prédéfini.

```
<xs:simpleType name="TypeMinInclusive1020">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="10"/>
        <xs:maxInclusive value="20"/>
    </xs:restriction>
</xs:simpleType>
```

Types simples - Restrictions multiples (4)

En reprenant le type simple personnalisé **TypeMinInclusive1020**, on désire changer son domaine de validité (10 à 20 → 0 à 9) comme montré ci-dessous, **ce n'est pas possible**.

```
<xs:simpleType name="TypeMinInclusive09">
    <xs:restriction base="TypeMinInclusive1020">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="9"/>
    </xs:restriction>
</xs:simpleType>
```

Types simples - Restrictions multiples (5)

En reprenant le type simple personnalisé **TypeMinInclusive1020**, on désire restreindre son domaine de validité (10 à 20 → 11 à 16), comme montré ci-dessous, **c'est possible**.

```
<xs:simpleType name="TypeMinInclusive09">
    <xs:restriction base="TypeMinInclusive1020">
        <xs:minInclusive value="11"/>
        <xs:maxInclusive value="16"/>
    </xs:restriction>
</xs:simpleType>
```

Types simples - Restrictions multiples (6)

Pour la facette **length**, il est impossible de la changer.

C'est-à-dire qu'il est interdit de redéfinir la taille d'un type personnalisé par restriction. Exemple :

```
<xs:simpleType name="Type30Car">
    <xs:restriction base="xs:string">
        <xs:length=«30"/>
    </xs:restriction>
</xs:simpleType>
```

Types simples - Restrictions multiples (7)

En reprenant le type simple personnalisé **Type30Car**, on désire changer son domaine de validité (30 → 15) comme montré ci-dessous, **ce n'est pas possible**.

```
<xs:simpleType name="Type15Car">
        <xs:restriction base=" Type30Car ">
                <xs:length="15"/>
        </xs:restriction>
</xs:simpleType>
```

Types simples - Restrictions multiples (8)

Il est possible de bloquer les restrictions liées aux facettes. Toutes les facettes (sauf **enumeration** et **pattern**) acceptent un attribut (fixed="true") qui empêche toute modification de restriction par dérivation.

```
<xs:simpleType name="TypeMinInclusive1020">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="10" fixed="true"/>
        <xs:maxInclusive value="20" fixed="true"/>
    </xs:restriction>
</xs:simpleType>
```

Dérivation par Liste (1)

La dérivation **Liste** est un mécanisme qui permet de créer une liste de type atomique séparée par un espace blanc.

Chaque élément de la liste doit être du même type.

Ce mécanisme est utilisé pour maintenir une compatibilité avec SGML.

Dérivation par Liste (2)

Exemple : liste de valeurs d'entiers

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

Dérivation par Liste – Facettes (1)

- La dérivation **Liste** acceptent les facettes suivantes: **length, minLength, maxLength, enumeration, whiteSpace, minInclusive, maxInclusive, minExclusive** et **maxExclusive.**
- L'unité de mesure est le nombre d'éléments dans la liste**.**
- Pour appliquer des facettes à un type dérivé par liste il faut procéder en deux temps: 1. Définition de la liste, 2. Application des facettes (attention s**impleType** n'accepte qu'une méthode de dérivation)

Dérivation par Liste – Facettes (2)

Exemple : liste valeurs d'entiers limité à une valeur maximale de 100.

```
<xs:simpleType name = "TypeListeEntier100">
    <xs:list>
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:maxInclusive value="100"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:list>
</xs:simpleType>
```

Dérivation par Liste – Facettes (3)

Exemple : une liste qui supporte une suite de 10 mots maximum séparés par un espace blanc.

Phase 1 :

```
<xs:simpleType name = "TypeListeMots">
        <xs:list itemType = "xs:string"/>
</xs:simpleType>
```

Dérivation par Liste – Facettes (4)

Phase 2:

```
<xs:simpleType name = "TypeListeMots10">
    <xs:list>
        <xs:simpleType>
            <xs:restriction base="TypeListeMots">
                <xs:maxLength value="10"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:list>
</xs:simpleType>
```

Dérivation par Union

Ce mécanisme permet de "fusionner" plusieurs espaces de définition.

Les deux seules facettes autorisées sont : **pattern** et **enumeration**.

Attention : **enumeration** est interdit avec le type prédéfini **boolean**.

Dérivation par Union – Exemple (1)

Exemple : définir un type simple personnalisé permettant de saisir soit une date, soit un nombre entier.

```
<xs:simpleType name = "TypeEntierOuDate">
    <xs:union itemType = "xs:integer xs:date"/>
</xs:simpleType>
```

Dérivation par Union – Exemple (2)

Exemple : définir un type simple personnalisé permettant de saisir soit un nombre entier, soit un texte. Syntaxe 1 :

```
<xs:simpleType name = "TypeTexteOuEntier">
        <xs:union>
                <xs:simpleType>
                        <xs:restriction base="xs:integer">
                        </xs:restriction>
                </xs:simpleType>
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="indéfini"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:union>
</xs:simpleType>
```

Dérivation par Union – Exemple (3)

Exemple : définir un type simple personnalisé permettant de saisir soit un nombre entier, soit un texte. Syntaxe 2 :

```
<xs:simpleType name = "TypeTexteOuEntier">
        <xs:union itemType = "xs:integer">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="indéfini"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:union>
</xs:simpleType>
```

Remarques et conseils (1)

L'ordre des dérivations est important.

Les facettes changent de signification lorsqu'elles sont appliquées à des listes ou à des unions.

Définir les types de base avec restrictions en premier et ensuite définir les listes et les unions.

ComplexContent – Dérivation (1)

Les types complexes peuvent avoir une dérivation par **extension**, ce qui permet d'ajouter des éléments et des attributs.

La dérivation des types complexes ne fonctionne correctement qu'avec ***sequence***.

ComplexContent – Dérivation (2)

Exemple d'un type complexe pour la saisie de données liées à une personne de bas qui se nomme **typeBasePersonne**.

```
<xs:complexType name="typeBasePersonne">
        <xs:sequence>
                <xs:element ref="Nom"/>
                <xs:element ref="Prenom"/>
                <xs:element ref="DateNaissance"/>
                </xs:sequence>
        <xs:attribute ref="idPers" use="required"/>
</xs:complexType>
```

ComplexContent – Dérivation (3)

Création type complexe **typeAuteur** en s'appuyant sur le type **typePersonne** avec l'ajout de l'élément *decede*.

```
<xs:complexType name="typeAuteur">
        <xs:complexContent>
                <xs:extension base="typeBasePersonne">
                        <xs:sequence>
                                <xs:element ref="decede" minOccurs="0"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

ComplexContent – Dérivation (5)

Dérivation par **restriction**.

Permet d'utiliser un type de base et de restreindre les éléments et les attributs possibles.

ComplexContent – Dérivation (7)

Exemple d'un type complexe pour la saisie de données liées à une personne de bas qui se nomme **typePersonne**.

```
<xs:complexType name="typePersonne">
        <xs:sequence>
                <xs:element ref="Nom"/>
                <xs:element ref="Prenom"/>
                <xs:element ref="DateNaissance"/>
                <xs:element ref="decede" minOccurs="0"/>
                <xs:element ref="qualification" minOccurs="0"/>
                </xs:sequence>
        <xs:attribute ref="idPers" use="required"/>
</xs:complexType>
```

ComplexContent – Dérivation (7)

En reprenant le type complexe **typePersonne**, il y a la Suppression de l'élément *decede.*

```
<xs:complexType name="typePersonneQualifie">
        <xs:complexContent>
            <xs:restriction base="typePersonne">
              <xs:sequence>
                    <xs:element ref="Nom"/>
                    <xs:element ref="Prenom"/>
                    <xs:element ref="DateNaissance"/>
                    <xs:element ref="qualification" minOccurs="0"/>
              </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
</xs:complexType>
```

Merci pour votre attention !