esprit
Se former autrement

XML Schema Definition

Module SOA

A.U 2019-2020

cdio
CONCEIVE DESIGN IMPLEMENT OPERATE

Enropean Accreditation of Engineering Programmes EUR-ACE®

Délivré par Cti Commission des Titres d'Ingénieur

CONFÉRENCE DES GRANDES ÉCOLES

Objectifs

- Définir un vocabulaire et une grammaire XML.
- Apprendre le langage XSD

Plan

- L’objectif de XSD
- Types simples en XSD
- Types complexes en XSD
- Héritage
- Contraintes d’intégrité
- Les espaces de noms XML

Introduction

Application « **A** »

<XML>

```
<produit>
    <id> 123</id>
<produit>
```

Application « **B** »

<XML>

```
<product>
    <ID> 123</ID>
    <name>shoes</name>
<product>
```

Objectifs

- But :Définir la structure d'un document XML pour que tous les intermédiaires suivent le même modèle grâce aux schemas XSD (XML Schema Definition).

- Un schéma XML définit:
  - Les éléments permis dans le document
  - Les attributs associés à ces éléments
  - La structure du document et les types de données

Présentation de XSD

- Alternative au DTD
- Recommandations W3C
- Issu de XML

=> Tous les outils (validateurs, parseurs, processeurs, ...) mais également les langages (XSLT, XPath, ...) qui permettent de travailler les documents XML sont utilisables sur des XSD.

DTD (Document Type Definition)

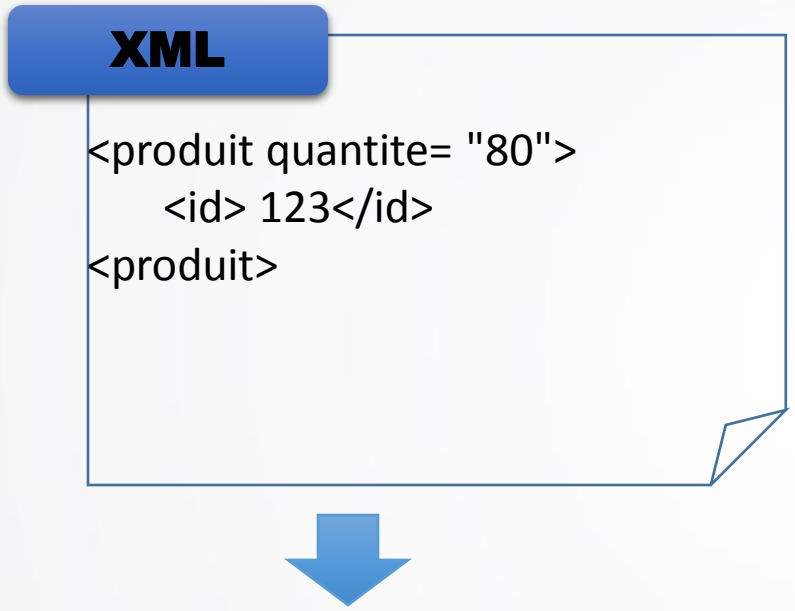
```
<!ELEMENT boutique (telephone*)>
<!ELEMENT telephone (marque, modele)>
<!ELEMENT marque (#PCDATA)>
<!ELEMENT modele (#PCDATA)>
```

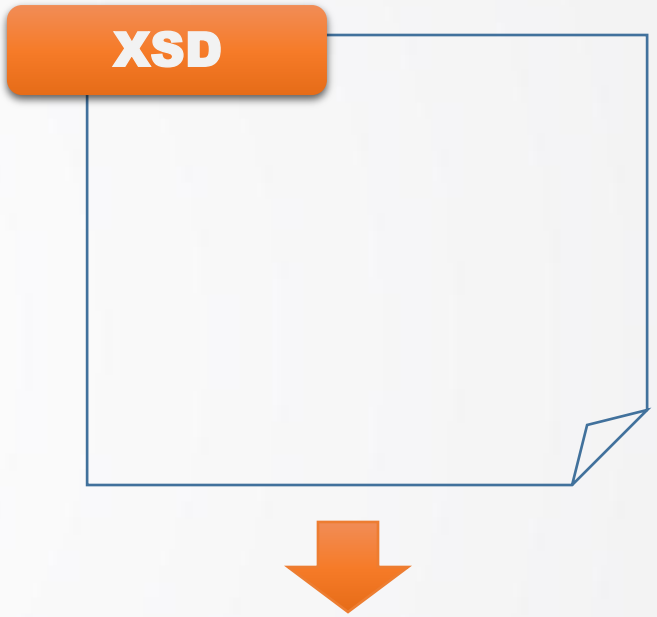
```
<?xml version = "1.0" ?>
<boutique>
 <telephone>
  <marque>Samsung</marque>
  <modele>Galaxy S5</modele>
 </telephone>
 <telephone>
  <marque>Apple</marque>
  <modele>iPhone 6</modele>
 </telephone>
</boutique>
```

DTD vs XSD

DTD	XSD
Nouveau langage → Syntaxe particulière	Langage issu de XML → Syntaxe XML
Types de données limités → PCDATA , CDATA	Types de données plus riches → integer, byte, string, float, …
Nombre d'occurrence très général → *, + et ?	Nombre d'occurrence plus précis [1,100] ,]2,100]
Aucune contrainte sur le contenu des éléments et attributs	Définition des contraintes sur le contenu des éléments/attributs → mot de passe de longueur 8 email contenant le caractère @
	Extensible

Vers un document XML valide





Document XML **bien formé**

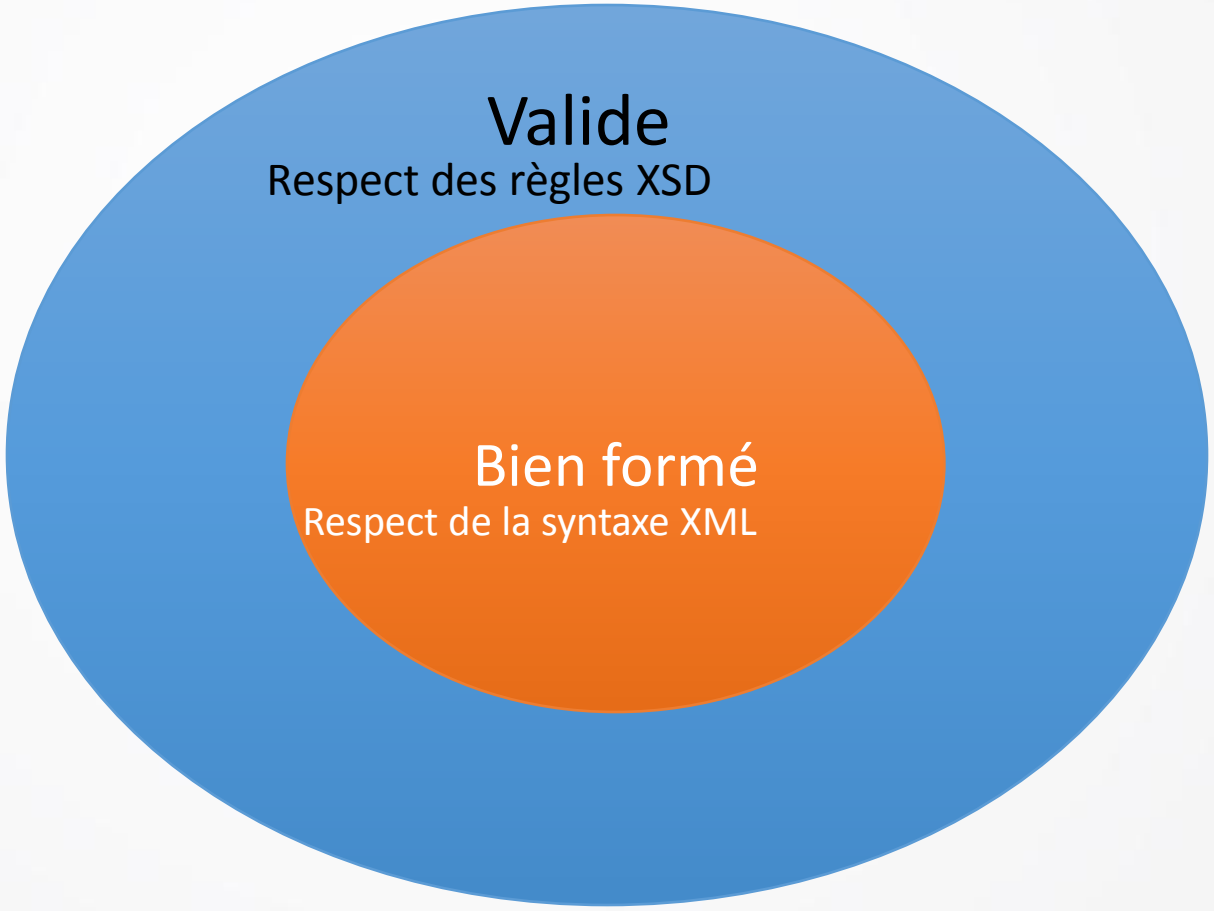
Respect de la syntaxe XML

Document XML **valide**

Respect des règles XSD

Vers un document XML valide

Un document XML est valide si et seulement s'il est bien formé



Structure d'un schéma XML

- Un document schema XML est défini dans un fichier dont l'extension est ***.xsd**
- Comme tout document XML, un schéma XML commence par la prologue XML et a un élément racine
- L'élément **<xs:schema>** est la racine de tout document Schema XML

Fichier XSD

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        …
</xs:schema>
```

Déclaration des éléments

```
<xs:element name="theName" type="theType" />
```

Valeur par défaut

```
<xs:element name="firstName" type="xs:string" default="Mickael" />
```

L'attribut default de l'élément *firstName* précise une valeur au cas où elle serait absente

Valeur fixée

```
<xs:element name="firstName" type="xs:string" fixed="Mickael" />
```

L'attribut fixed de l'élément *firstName* précise une valeur et ne peut être modifiée

Déclaration des attributs

```
<xs:attribute name="theName" type="theType" use="required" />
```

(required ou optional)

Valeur par défaut

```
<xs:attribute name="remark" type="xs:string" default="Remarque à préciser" />
```

L'attribut default de l'attribut *remark* précise une valeur au cas où elle serait absente

Valeur fixée

```
<xs:attribute name="remark" type="xs:string" fixed="Remarque à préciser" />
```

L'attribut fixed de l'attribut *remark* précise une valeur et ne peut être modifiée

Les types de données 1/2

- Les attributs sont de types simples
- Les éléments sont de:
  - **Types simples**

    un élément de type simple ne peut comporter ni attributs, ni de sous éléments

  - **Types complexes**

    un élément de type complexe permet d'exprimer des sous éléments et permet également d'y associer des attributs

Les types de données 2/2

Eléments de type simple

```
<film/>
```

```
<film>Gladiator</film>
```

Eléments de type complexe

```
<film type="action " />
```

```
<film type= "action ">
    Gladiator
</film>
```

```
<film>
   <annee>2000</annee>
  <realisateur>R.Scott</realisateur>
</film>
```

```
<film type= "action " >
   <annee>2000</annee>
  <realisateur>R.Scott</realisateur>
</film>
```

XSD

Les types simples 1/8

Les principaux types simples prédéfinis

- xs:int
- xs:boolean
- xs:string
- xs:long
- xs:float
- xs:positiveInteger : 1, 2, …
- xs:negativeInteger : …, -2, -1
- xs:nonNegativeInteger : 0, 1, 2, …
- xs:nonPositiveInteger : …, -2, -1, 0
- xs:unsignedLong : 0, 1, … 18446744073709…

Les types simples: restriction

Les types simples dérivés

- On peut créer de nouveaux types simples en dérivant des types simples existants (prédéfinis ou dérivés)

- Un nouveau type simple peut être défini par **restriction** ou **extension**

Exemple:

- L'age est un entier compris entre **1** et **100**
- L'email est une chaîne de caractères qui doit contenir le caractère **@**

Les types simples: restriction

Les types simples dérivés

- Les restrictions sur les types simples permettent de dériver de nouveaux types à partir de types existants
- Les restrictions passent par l'utilisation des **facettes**
- Une facette permet de définir des contraintes sur le nouveau type à créer

Les types simples: restriction

Les types simples dérivés

- La création de nouveaux types simples est réalisée avec la balise **<xs:simpleType>**

```
<xs:simpleType name="newType" >
        ...
</xs:simpleType>
```

- La restriction est exprimée avec la balise **<xs:restriction>**

```
<xs:simpleType name="newType" >
        <xs:restriction base="type" >
                ...
        </xs:restriction>
</xs:simpleType>
```

Nouveau Type simple

Type simple de départ

Les types simples: restriction

Les principales facettes

- **Facette length**

```
<xs:element name=" password" type ="passwordType" />
<xs:simpleType name ="passwordType" >
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
</xs:simpleType>
```

- **Facette minLength, maxLength**

```
<xs:element name="password"  type="passwordType" />
<xs:simpleType name =" passwordType" >
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
</xs:simpleType>
```

Les types simples: restriction

Les principales facettes

- **Facette minLInclusive, minExclusive, maxInclusive, maxExclusive**

```
<xs:element name=" age" type ="ageType" />
<xs:simpleType name="ageType">
   <xs:restriction base="xs:int">
      <xs:minInclusive value="1" />
      <xs:maxInclusive value="100" />
   </xs:restriction>
</xs:simpleType>
```

Les types simples: restriction

Les principales facettes

- **Facette enumeration**

```
<xs:element name=" sexe" type ="sexeType" />
<xs:simpleType name="sexeType" >
  <xs:restriction base="xs:string">
    <xs:enumeration value="homme" />
    <xs:enumeration value="femme" />
    <xs:enumeration value="indéterminé" />
  </xs:restriction>
</xs:simpleType>
```

Trois valeurs sont autorisées

Les types simples: restriction

Les principales facettes

- **Facette pattern**

```
<xs:element name="email" type ="emailType" />
<xs:simpleType name="emailType">
  <xs:restriction base="xs:string">
    <xs:pattern value= " [a-z]*@[a-z]* " />
  </xs:restriction>
</xs:simpleType>
```

Toutes les chaînes de caractères de type *emailType* doivent respecter ce pattern

[a-z]*	0 ou plusieurs lettre(s)
([a-z][A-Z])+	1 ou plusieurs paires de lettres min et maj sToP, Stop,STOP,stop
male|female	Liste de choix
[a-zA-Z0-9]{8}	8 caractères (chiffre, lettre min, lettre maj)

Les types complexes 1/8

- Un élément **de type complexe** peut contenir d'autres **éléments et / ou des attributs**

- **Quatre** combinaisons d'éléments complexes sont à distinguer
  - **Eléments vides** qui ne contiennent que **des attributs**
  - **Eléments de type simple** qui contiennent **des attributs**
  - **Eléments** qui peuvent contenir **des sous éléments**
  - **Eléments** qui contiennent **des attributs** et **des sous éléments**

- La création d'un éléments de type complexe est réalisée avec la balise **<xs:complexType>**

```
<xs:complexType name="newType" >
        ...
</xs:complexType>
```

Les types complexes 2/8

Eléments vides qui ne contiennent que des attributs

```
<child remark="He's too much" old="3" sexe="homme"/>
```

```
<xs:element name="child"  type ="childType" />

<xs:complexType name ="childType" >
    <xs:attribute name="remark" type="xs:string" use="required" />
    <xs:attribute name="old" type= "xs:int" />
    <xs:attribute name="sexe" type="xs:string" />
</xs:complexType>
```

Les types complexes 3/8

Eléments qui peuvent contenir des sous éléments

```
<person>
  <name>…</name>
  <firstName>…</firstName>
  <old>…</old>
  <email>…</email>
</person>
```

sequence exprime que les sous éléments doivent apparaître **dans l'ordre** spécifié

```
<xs:element name="person" type="personType "  />
<xs:complexType name="personType " >
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="firstName" type="xs:string" />
      <xs:element name="old" type="ageType" />
      <xs:element name="email" type="emailType" />
   </xs:sequence>
</xs:complexType>
```

XSD

esprit Se former autrement

Les types complexes 4/8

Eléments qui peuvent contenir des sous éléments

Indicateurs d'ordre

- XSD permet d'exprimer trois sortes d'indicateurs d'ordre:
  - sequence
  - all
  - choice

- **all** tous les sous éléments peuvent apparaître dans **n'importe quel ordre**

```
<xs:complexType name= "personType">
  <xs:all>
      <xs:element name="name" type="xs:string" />
      <xs:element name="firstName" type="xs:string" />
      <xs:element name="old" type="ageType" />
      <xs:element name="email" type="emailType" />
  </xs:all>
</xs:complexType>
```

```
<person>
  <name>…</name>
  <email>… </email>
  <old>…</old>
  <firstName>…</firstName>
</person>
```

Les types complexes 5/8

Eléments qui peuvent contenir des sous éléments

Indicateurs d'ordre

- **choice** exprime qu'**un seul** élément parmi tous les sous éléments peut apparaître

```
<xs:complexType name= "personType">
  <xs:choice>
      <xs:element name="name" type="xs:string" />
      <xs:element name="firstName" type="xs:string" />
      <xs:element name="old" type="ageType" />
      <xs:element name="email" type= "emailType" />
  </xs:choice>
</xs:complexType>
```

```
<person>
   <name>…</name>
</person>
```

Les types complexes 6/8

Eléments qui peuvent contenir des sous éléments

Indicateurs d'occurence

- maxOccurs : précise le nombre d'occurrence maximum
- minOccurs : précise le nombre d'occurrence minimum

- Si les valeurs de maxOccurs ou minOccurs ne sont pas explicitement précisées, la valeur par **défaut est de 1**

- Pour définir une valeur infinie, fixer la valeur à **unbounded**

```
<xs:complexType name= "personType">
 <xs:sequence>
     <xs:element name="name" type="xs:string" />
     <xs:element name="firstName" type="xs:string" />
     <xs:element name="old" type="ageType" minOccurs="0" />
     <xs:element name="email" type= " emailType"  minOccurs="2"
                                                  maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

old est un élément optionnel

Les types complexes 7/8

Eléments qui peuvent contenir des sous éléments

- Même si on a un seul élément fils, on doit utiliser un indicateur d'ordre

```
<xs:complexType name= "personType">
 <xs:sequence>
      <xs:element name="name" type="xs:string" />
</xs:sequence>
</xs:complexType>
```

Les types complexes 8/8

Eléments qui peuvent contenir des sous éléments et des attributs

```
<xs:complexType name= "personType">
  <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="firstName" type="xs:string" />
      <xs:element name="old" type="ageType" />
      <xs:element name="email" type= "emailType" />
 </xs:sequence>
 <xs:attribute name="id" type="xs:int" />
</xs:complexType>
```

```
<person id ="139" >
   <name>…</name>
   <firstName>…</firstName>
   <old>…</old>
   <email>…</email>
</person>
```

L'héritage en XSD

Héritage d'un élément simple

- Possibilité de définir un nouveau type sur la base d'un type simple existant
- Utilisation de la balise **<xs:simpleContent>**

<poids>67</poids>

extension

<poids unite="kg" >67</poids>

```
<xs:element name ="poids"  type ="poidsType" />
<xs:complexType name ="poidsType" >
    <xs:simpleContent>
      <xs:extension base="xs:integer">
         <xs:attribute name="unite" type="xs:string" />
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

L'héritage en XSD

Héritage d'un élément complexe

- Possibilité de définir un nouveau type complexe sur la base d'un type complexe existant

Type complexe existant

```
<adress>
    <receiver></receiver>
    <street></street>
    <city></city>
</address>
```

Nouveau type complexe

```
<adressUS>
    <receiver></receiver>
    <street></street>
    <city></city>
    <state></state>
    <zip></zip>
</adressUS>
```

L'héritage en XSD

Héritage d'un élément complexe

- Utilisation de la balise **<xs:complexContent>**

Le type *addressType* définit trois sous éléments en séquence

```
<xs:complexType name="addressType">
 <xs:sequence>
   <xs:element name="receiver" type="xs:string" />
   <xs:element name="street" type="xs:string" />
   <xs:element name="city" type="xs:string" />
 </xs:sequence>
</xs:complexType>
```

Le type *usAddressType* propose une extension avec deux nouveaux sous éléments en séquence

```
<xs:complexType name="usAddressType">
 <xs:complexContent>
   <xs:extension base="addressType">
     <xs:sequence>
           <xs:element name="state" type="xs:string" />
           <xs:element name="zip" type="xs:string" />
    </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

esprit Se former autrement

Espace de noms XML 1/6

Problème

```
<employe>
  <id>E0000001</id>
  <nom>Smith</nom>
  <prenom>John </prenom>
</employe>
```

```
<departement>
   <id>D001</id>
   <nom>Marketing</nom>
</departement>
```

Fusion des 2 documents

```
<entreprise>
 <departement>
   <id>D001</id>
   <nom>Marketing</nom>
   <employe>
     <id>E0000001</id>
     <nom>Smith</nom>
     <prenom>John </prenom>
   </employe>
 </departement>
</entreprise>
```

Confusion sur le sens des éléments id et nom

Espace de noms XML 2/6

Objectif Distinguer les éléments et les attributs de différentes documents XML qui ont le même nom

Solution Utiliser les espaces de noms XML

```
<emp:employe>
  <emp:id>E0000001</emp:id>
  <emp:nom>Smith</emp:nom>
  <emp:prenom>John </emp:prenom>
</emp:employe>
```

```
<dep:departement>
   <dep:id>D001</dep:id>
   <dep:nom>Marketing</dep:nom>
</dep:departement>
```

Fusion des 2 documents

```
<dep:departement>
   <dep:id>D001</dep:id>
   <dep:nom>Marketing</dep:nom>
   <emp:employe>
      <emp:id>E0000001</emp:id>
      <emp:nom>Smith</emp:nom>
      <emp:prenom>John </emp:prenom>
  </emp:employe>
</dep:departement>
```

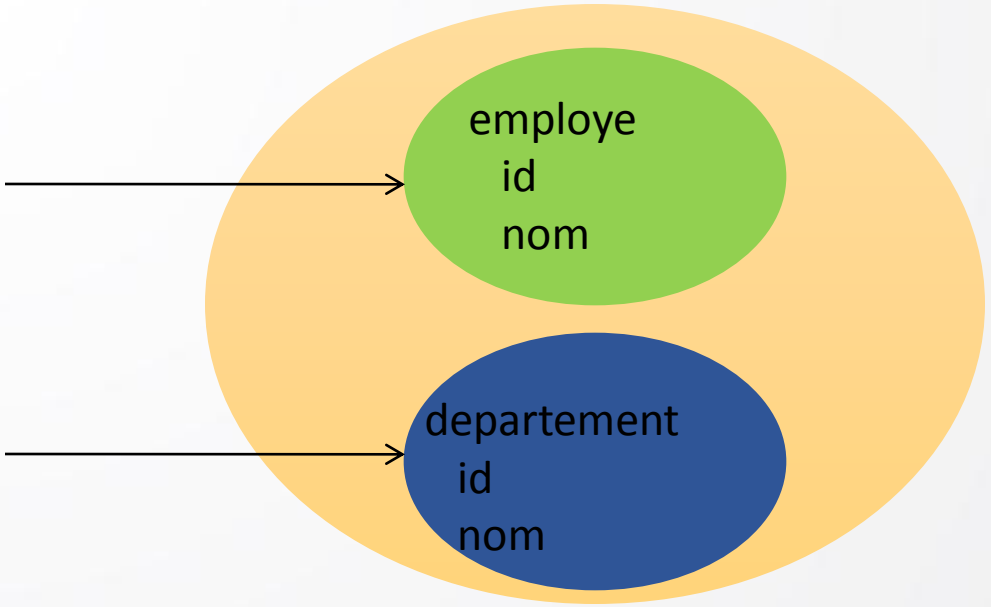
Espace de noms XML 3/6

Déclaration des espaces de noms

- Un espace de nom associe un préfixe à un URI
- L'URI (Uniform Resource Identifier) sert à identifier un espace de noms
- Le préfixe est une chaîne utilisée pour référencer l'espace de nom dans un fichier XML.

URI: http://employe.com
Préfixe: emp

URI: http://departement.com
Préfixe: dep



Espace de noms XML 4/6

Déclaration des espaces de noms

- La déclaration de l'espace de noms se fait au moyen de l'attribut **xmlns**

```
<element xmlns:prefix="URI">
```

```
<emp:employe xmlns:emp="http://emloye.com">
  <emp:id>E0000001</emp:id>
  <emp:nom>Smith</emp:nom>
  <emp:prenom>John </emp:prenom>
</emp:employe>
```

```
<dep:departement xmlns:dep="http://departement.com">
  <dep:id>D001</dep:id>
  <dep:nom>Marketing</dep:nom>
</dep:departement>
```

Espace de noms XML 5/6

Déclaration des espaces de noms

```
<emp:employe xmlns:emp="http://emloye.com">
  <emp:id>E0000001</emp:id>
  <emp:nom>Smith</emp:nom>
  <emp:prenom>John </emp:prenom>
</emp:employe>
```

```
<dep:departement xmlns:dep="http://departement.com">
  <dep:id>D001</dep:id>
  <dep:nom>Marketing</dep:nom>
</dep:departement>
```

Fusion des 2 documents

```
<entreprise xmlns:dep="http://departement.com"
            xmlns:emp="http://emloye.com" >
 <dep:departement>
  <dep:id>D001</dep:id>
  <dep:nom>Marketing</dep:nom>
  <emp:employe>
    <emp:id>E0000001</emp:id>
    <emp:nom>Smith</emp:nom>
    <emp:prenom>John </emp:prenom>
  </emp:employe>
 </dep:departement>
</entreprise>
```

esprit Se former autrement

Espace de noms XML 6/6

Déclaration des espaces de noms

Espace de nom pour XSL

```
<?xml version="1.0"?>
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
                …
</xsl:stylesheet >
```

Espace de nom pour XSD

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
                …
</xs:schema>
```

Espace de noms XSD 1/4

XML Schema Namespace

http://www.w3.org/2001/XMLSchema

schema
element
complexType
string
integer
boolean

Les éléments et les types appartenant au XML Schema Namespace sont utilisés pour écrire un document XSD

Document XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    xmlns="http://www. departement.org"
                    targetNamespace="http://www. departement.org">
  <xs:element name="departement" type="depType"/>
  <xs:complexType name="depType">
    <xs:sequence>
                <xs:element name="id" type="xs:integer"/>
                <xs:element name="nom" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Les nouveaux types et élément déclarés dans le document XSD appartiennent à un nouveau espace de nom: c'est le tragetNamespace

Target Namespace

http://www.departement.org

departement
id
nom
departemenType

Le targetNamesapce est utilisé par le fichier XML pour la validation

Document XML

```
<departement>
  <id>D001</id>
  <nom>Marketing</nom>
</departement>
```

XSD

Espace de noms XSD 2/4

Espaces de noms XSD

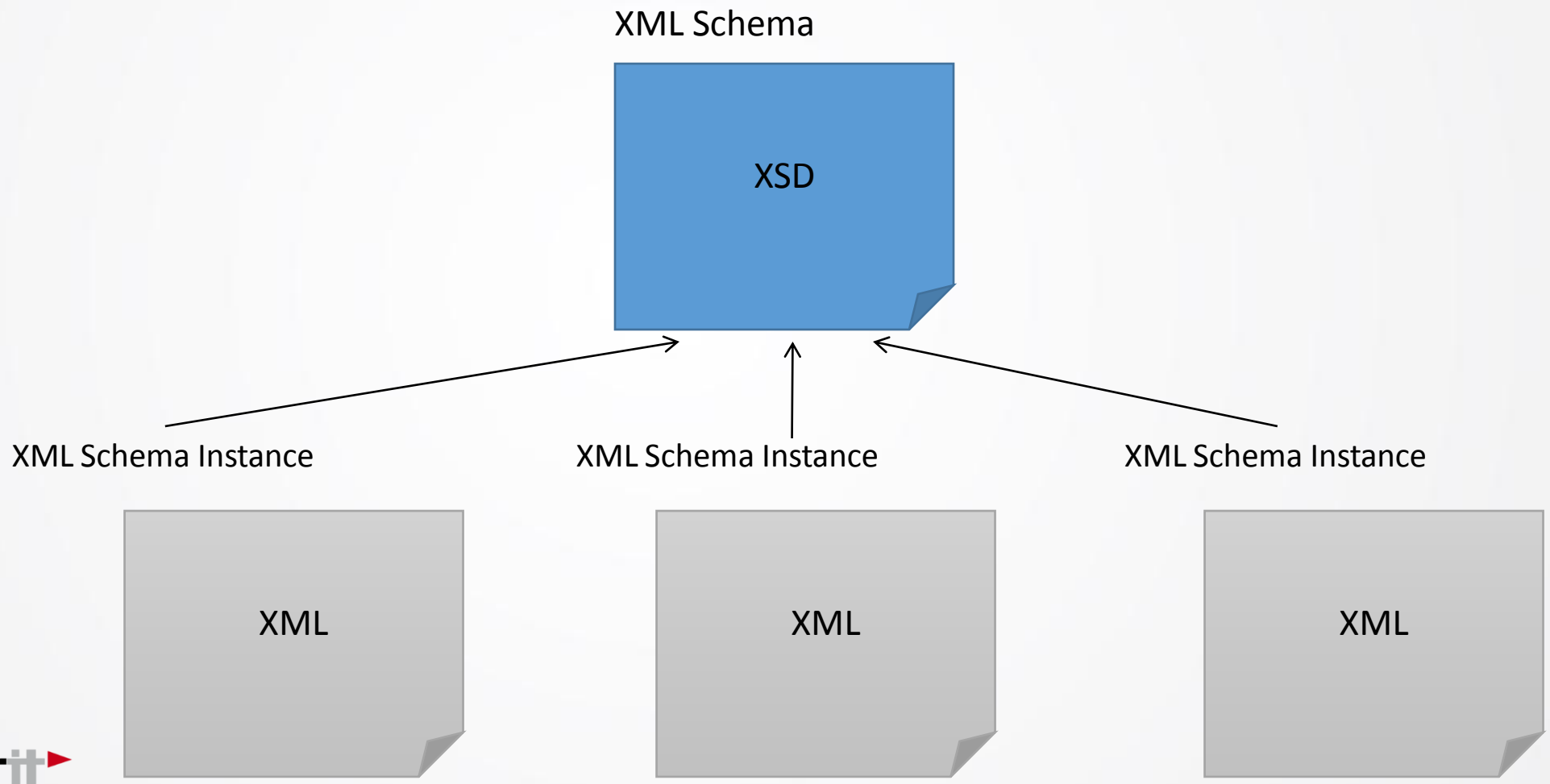
Fichier XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:dep="http://www.departement.org"
           targetNamespace="http://www.departement.org" >
        <xs:complexType name="depType" >
            <xs:sequence>
                <xs:element name="id" type="xs:integer"/>
                <xs:element name="nom" type="xs:string"/>
        </xs:sequence>
        </xs:complexType>
        <xs:element name="departement" type=" dep:depType "/>
</xs:schema>
```

xmlns:xs	▪espace de nommage des éléments et types XSD
xmlns:dep	▪espace de nommage des nouveaux types définis par le programmeur
targetNamespace	▪espace de nommage du schema XSD cible ▪C'est l'espace de noms qui sera référé par le fichier XML

Espace de noms XSD 3/4

Association d'un fichier XML à un fichier XSD



Espace de noms XSD 4/4

Association d'un fichier XML à un fichier XSD

Fichier XML

```
<departement  xmlns="http://www.departement.org"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://www.departement.org departement.xsd">
```

xmlns	▪Espace de noms des éléments utilisés dans le fichier XML
xmlns:xsi	▪Il s'agit d'une instance de schema XSD
xsi:schemaLocation	▪Localiser le document XSD

En résumé

- XSD est un langage permettant la définition de la structure d'un document XML

- XSD offre une richesse de types
  - ✓ Types simples, types complexes
  - ✓ Restriction, extension

- L'association d'un fichier XML à un fichier XSD passe par l'utilisation des espaces de noms.

Références

- http://www.liafa.jussieu.fr/~carton/Enseignement/XML/Cours/Schemas/
- http://www.grappa.univ-lille3.fr/~torre/Enseignement/Cours/XML/xmlschema.php
- http://www.teluq.ca/inf6450/mod1/chapitre4.xml