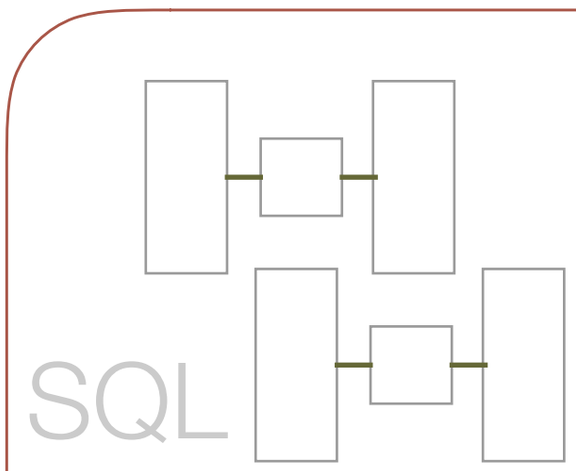
LABD

Master Info M1 2017-2018

Cours 4 : Espaces de noms

Systèmes d'Informations, Documents - Intégration



Etat civil

Système d'Information

Flux standardisés

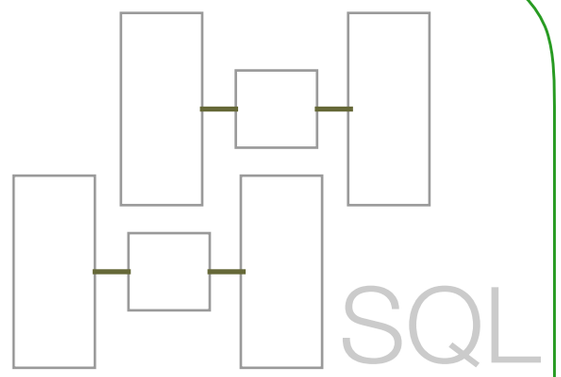
Docu-ments

Docu-ments

Docu-ments

XML JSON ...

Santé



Système d'Information

```
<famille nom="bilasco">
 <membre
   no_secu="1800913245605">
  <role nom="parent" rang="1">
  <age>37</age>
 </membre>
 <membre no_secu="2790943405">
  <role nom="parent" rang="2"/>
  <age>38</age>
 </membre>
</famille>
```

```
<personne nom="marius"
  id="1800913245605">
 <membre type="pied" rang="1">
  <etat>sain</etat>
 </membre>
 <membre type="main" rang="1">
  <etat>sain</etat>
 </membre>
 ...
</personne>
```

```
<famille>
 <membre
   no_secu="1800913245605">
  <role nom="parent" rang="1">
  <age>37</age>
 </membre>
 <membre no_secu="2790943405">
  <role nom="parent" rang="2"/>
  <age>38</age>
 </membre>
</famille>
```

```
<personne
  id="1800913245605">
 <membre type="pied" rang="1">
  <etat>sain</etat>
 </membre>
 <membre type="main" rang="1">
  <etat>sain</etat>
 </membre>
 ...
</personne>
```

```
<famille>
  <membre no_secu="1800913245605">
    <role nom="parent" rang="1">
    <age>37</age>
    <etat_physiologique>
      <membre type="pied" rang="1">
        <etat>sain</etat>
      </membre>
      <membre type="main" rang="1">
        <etat>sain</etat>
      </membre>
    </etat_physiologique>
  </membre>
  ...
</famille>
```

a - Etat civil
b - Santé
c - Intégration

```
<a:famille>
  <a:membre a:no_secu="1800913245605">
    <a:role a:nom="parent" a:rang="1">
    <a:age>37</a:age>
    <c:etat_physiologique>
      <b:membre b:type="pied" b:rang="1">
        <b:etat>sain</b:etat>
      </b:membre>
      <b:membre b:type="main" b:rang="1">
        <b:etat>sain</b:etat>
      </b:membre>
    </c:etat_physiologique>
  </a:membre>
  ...
</a:famille>
```

Motivations

- mélanger différents vocabulaires et éviter les conflits de nom
- modularité, réutilisation
- exporter les définitions d'un schéma

➡ utilisation de noms qualifiés (= préfixés) pour les éléments et les attributs.

1) Utiliser des espaces de noms

2) Créer des espaces de noms

Identification d'un espace de noms

Un espace de noms est identifié par une adresse IRI, (Internationalized Resource Identifier)

- IRI = extension des URI permettant l'utilisation de caractères internationaux (par ex. UTF-8) dans l'adresse elle-même.

  `http://www.exemple.org/clés`

- URI = URN ou URL
  - URN : `urn:isbn-0-395-36341-1`
  - URL : ftp://ftpperso.free.fr/pxml

Remarques

- Le W3C déconseille l'usage d'une IRI relative comme identifiant d'espace de nom
- L'analyse d'un identifiant d'espace de nom tient compte de la casse et ne prend pas en compte la résolution de l'IRI. Tous les exemples suivants représentent des identifiants différents :

```
http://www.Example.org/wine

http://www.example.org/Wine

http://www.example.org/rosé

http://www.example.org/ros%c3%a9

http://www.example.org/ros%c3%A9
```

Finalement en pratique

Identifiant d'espace de nom = URL absolue avec des caractères ASCII.

```
http://www.w3.org/XML/1998/namespace

http://www.w3.org/1999/xhtml

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

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

http://www.w3.org/1999/XSL/Transform

http://xml.insee.fr/schema/

http://fil.univ-lille1.fr/miage-fa-fc
```

Déclaration d'un espace de noms

Le préfixe qui désigne un espace de noms doit avoir été déclaré, grâce à un pseudo attribut qui commence par `xmlns:`

- Les préfixes `xml` et `xmlns` sont réservés.

- La déclaration se fait dans la balise ouvrante d'un élément

- Lorsqu'on déclare un espace de noms, le préfixe est applicable *(mais non appliqué)* dès la balise ouvrante où se fait la déclaration, et pour tout le contenu de cet élément, sauf si le même préfixe est utilisé plus bas pour un autre espace de noms.

- L'utilisation du préfixe pour un élément (ou attribut) indique que cet élément (ou attribut) appartient à l'espace de noms associé au préfixe. (nom qualifié)

- On peut déclarer plusieurs espaces de noms dans une même balise ouvrante.

Déclaration d'un espace de noms (2)

- La déclaration se fait dans la balise ouvrante d’un élément

- Lorsqu’on déclare un espace de noms, le préfixe est applicable *(mais non appliqué)* dès la balise ouvrante où se fait la déclaration, et pour tout le contenu de cet élément.

```
<x xmlns:edi="http://ecom.exple.org/schema">
   ...
</x>
```

Attention : x n’est pas associé à l’espace *edi*, mais l’espace *edi* http://ecom.exple.org/schema est visible jusqu’à la balise fermante. x n’est pas associé à un espace de noms.

Déclaration d'un espace de noms (3)

- La déclaration se fait dans la balise ouvrante d’un élément

- Lorsqu’on déclare un espace de noms, le préfixe est applicable dès la balise ouvrante où se fait la déclaration, et pour tout le contenu de cet élément.

```
<edi:price xmlns:edi="http://ecom.exple.org/sch" units="Euro">
  32.18
</edi:price>
```

Déclaration d'un espace de noms (4)

- La déclaration se fait dans la balise ouvrante d’un élément
- Lorsqu’on déclare un espace de noms, le préfixe est applicable dès la balise ouvrante où se fait la déclaration, et pour tout le contenu de cet élément.

```
<x xmlns:edi="http://ecom.exple.org/schema">
  <lineItem edi:taxClass="exempt">
    Baby food
  </lineItem>
</x>
```

Déclaration d'un espace de noms (5)

- La déclaration se fait dans la balise ouvrante d’un élément

- Lorsqu’on déclare un espace de noms, le préfixe est applicable dès la balise ouvrante où se fait la déclaration, et pour tout le contenu de cet élément, sauf si le même préfixe est utilisé plus bas pour un autre espace de noms.

```
<x xmlns:edi="http://ecom.exple.org/sch1">
  <edi:a>
     <edi:b xmlns:edi="http://ecom.exple.org/sch2">
        <edi:a>
           <edi:c/>
        </edi:a>
     </edi:b>
     <edi:b/>
  </edi:a>
</x>
```

Déclaration d'un espace de noms (6)

- La déclaration se fait dans la balise ouvrante d’un élément

- Lorsqu’on déclare un espace de noms, le préfixe est applicable dès la balise ouvrante où se fait la déclaration, et pour tout le contenu de cet élément.

```
<x xmlns:edi="http://ecom.exple.org/schema">
   ...
</x>
```

```
<edi:d></edi:d>
```

X déclaration invalide, l’espace du nom *edi* *http://*ecom.exple.org/schema n’est plus visible

Déclaration d'un espace de noms (7)

On peut déclarer plusieurs espaces de noms dans une même balise ouvrante.

```
<bk:book   xmlns:bk="urn:loc.gov:books"
           xmlns:isbn="urn:ISBN:0-395-36341-6">
  <bk:title>Cheaper by the Dozen</bk:title>
  <isbn:number>1568491379</isbn:number>
</bk:book>
```

Espace de noms par défaut

- Si on utilise l'attribut `xmlns` (sans `:`), on définit alors un espace de noms par défaut, pour lequel il n'existe pas de préfixe associé. L'espace de nom par défaut ne s'applique pas aux attributs.

```
<?xml version="1.1"?>
<!-- elements are in the HTML namespace, by default -->
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>Frobnostication</title></head>
  <body>
    <p>
      Moved to <a href="http://frob.example.com">here</a>.
    </p>
  </body>
</html>
```

Espace de noms par défaut (2)

- Si on utilise l'attribut `xmlns` (sans `:`), on définit alors un espace de noms par défaut, pour lequel il n'existe pas de préfixe associé.

```
<?xml version="1.1"?>
<!-- initially, the default namespace is "books" -->
<book xmlns="urn:loc.gov:books"
      xmlns:isbn="urn:ISBN:0-395-36341-6">
  <title>Cheaper by the Dozen </title>
  <isbn:number>1568491379</isbn:number>
  <notes>
  <!-- make HTML the default namespace for some commentary -->
    <p xmlns="http://www.w3.org/1999/xhtml">
      This is a <i>funny</i> book!
    </p>
  </notes>
</book>
```

Espace de noms par défaut (3)

- Il faut, en général, réserver l'espace de noms par défaut à l'espace de noms le plus utilisé.

- Tant que l'espace de noms par défaut n'a pas été spécifié, les éléments dont le nom n'est pas qualifié ne font partie d'aucun espace de noms. Leur propriété espace de noms n'a pas de valeur.

- Il est possible de revenir à l'espace de noms par défaut non spécifié en affectant la chaîne vide à l'attribut `xmlns`.

- Les attributs peuvent également avoir des noms qualifiés formés d'un préfixe et d'un nom local. Ils font alors partie de l'espace de noms auquel est associé le préfixe.

- Les attributs dont le nom n'est pas qualifié ne font jamais partie de l'espace de noms par défaut. Cette règle s'applique que l'espace de noms par défaut soit spécifié ou non.

Exercice

```
<?xml version="1.0" encoding="utf-8"?>

<exercice xmlns:pre="http://A">
  <pre:niveau xmlns:pre="http://D" xmlns="http://A">
     <out:garage xmlns:out="http://C">
        <pre:alcove/>
        <entree xmlns:pre="http://E">
           <pre:cuisine/>
        </entree>
     </out:garage>
  </pre:niveau>
  <def xmlns="http://B">
     <pre:figue>
        <levain/>
     </pre:figue>
  </def>
</exercice>
```

exercice -> X
niveau -> D
garage -> C
alcove -> D
entree -> A
cuisine -> E
def -> B
figue -> A
levain -> B

2) Créer des espaces de noms

Exporter un espace de noms

attribut `targetNamespace`

- Pour exporter (créer) un espace de nom dans un schéma, on utilise l'attribut `targetNamespace` de la balise xsd:schema

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
      "http://www.w3.org/2001/XMLSchema
          http://www.w3.org/2001/XMLSchema.xsd"
  targetNamespace="http://fil.univ-lille1.fr/miage-fa-fc">
   <!-- description de la miage fa fc -->
   ...
```

Exporter un espace de noms (2)

```
<xsd:schema
    …
    targetNamespace="http://fil.univ-lille1.fr/miage-fa-fc">
  <!-- description de la miage fa fc –>
  <xsd:element name="creneaux" …/>
    <xsd:element name="creneau" …/>
    <xsd:element name="trimestre" …/>
    <xsd:element name="jour" …/>

    …
```

Les éléments creneaux, creneau, trimestre, jour … sont définis dans l'espace de nom http://fil.univ-lille1.fr/miage-fa-fc

Exporter un espace de noms (3)

```
<?xml version="1.0" encoding="utf-8"?>
<miage-fa-fc
  xmlns="http://fil.univ-lille1.fr/miage-fa-fc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://fil.univ-lille1.fr/miage-fa-fc
http://www.fil.univ-lille1.fr/FORMATIONS/MIAGE-FC-FA/schemas/
miage-fa-fc.xsd"
    annee="2011">
  <creneaux>
    <creneau>
      <trimestre>T1</trimestre>
        <jour>lundi</jour>
        <de>09:00:00</de>
        <a>12:00:00</a>
        <salle>M5-A2</salle>
        ...
```

Espaces de noms et schémas

attribut `elementFormDefault`

Quand on exporte un espace de noms,

- Les déclarations globales appartiennent à l'espace de nom
- Les déclarations locales n'appartiennent pas à l'espace de nom, sauf si on ajoute l'attribut `elementFormDefault="qualified"`
- on dispose aussi de l'attribut `attributeFormDefault`

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="http://www.w3.org/2001/XMLSchema
                        http://www.w3.org/2001/XMLSchema.xsd"
    xmlns="http://fil.univ-lille1.fr/miage-fa-fc"
    targetNamespace="http://fil.univ-lille1.fr/miage-fa-fc"
    elementFormDefault="qualified">
  <!-- description de la miage fa fc -->
```

Espaces de noms et schémas (2)

```
<xsd:schema …
   targetNamespace="http://fil.univ-lille1.fr/miage-fa-fc">
  <!-- description de la miage fa fc –>
  <xsd:element name="creneaux" …/>
  <xsd:element name="creneau" …/>
  …
```

creneau - visibilité globale, associé à l'espace de noms cible

```
<xsd:schema …
   targetNamespace="http://fil.univ-lille1.fr/miage-fa-fc">
  <xsd:element name="creneaux">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="creneau" .../>
      <xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

creneau - visibilité locale, pas d'espace de noms associé

Espaces de noms et schémas (3)

```
<xsd:schema …
   targetNamespace="http://fil.univ-lille1.fr/miage-fa-fc">
  <xsd:element name="creneaux">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="creneau" .../>
      <xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

creneau - visibilité locale, pas d'espace de noms associé

```
<xsd:schema …
   targetNamespace="http://fil.univ-lille1.fr/miage-fa-fc"
   elementFormDefault="qualified">
  <xsd:element name="creneaux">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="creneau" .../>
      <xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

creneau - visibilité locale, associé à l'espace de noms

Espaces de noms et schémas (4)

```
<xsd:schema …
  targetNamespace="fil.fr/miage-fa-fc">
 <xsd:element name="creneaux">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="creneau" .../>
    <xsd:sequence>
  </xsd:complexType>
 …
</xsd:element>
```

creneau
- pas d'espace de noms associé

```
<x:creneaux
   xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
    xmlns:x="http://fil.fr/miage-fa-fc" ...>
    <creneau>…</creneau>
</x:creneaux>
```

```
<creneaux
   xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
    xmlns="http://fil.fr/miage-fa-fc" ...>
    <creneau>…</creneau>
</creneaux>
```

X

espace de nom par défaut pour l'élément creneau

```
<xsd:schema …
  targetNamespace="fil.fr/miage-fa-fc">
 <xsd:element name="creneaux">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="creneau" .../>
    <xsd:sequence>
  </xsd:complexType>
 …
</xsd:element>
```

creneau
- espace de noms associé

```
<x:creneaux
   xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
    xmlns:x="http://fil.fr/miage-fa-fc" ...>
    <x:creneau>…</x:creneau>
</x:creneaux>
```

```
<creneaux
   xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
    xmlns="http://fil.fr/miage-fa-fc" ...>
    <creneau>…</creneau>
</creneaux>
```

Espaces de noms et schémas (5)

```
<xsd:schema …
  targetNamespace="fil.fr/miage-fa-fc">
 <xsd:element name="creneaux">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="creneau" .../>
    <xsd:sequence>
  </xsd:complexType>
 …
</xsd:element>
```

creneau
- pas d'espace de noms associé

```
<x:creneaux
  xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
   xmlns:x="http://fil.fr/miage-fa-fc" ...>
   <creneau>…</creneau>
</x:creneaux>
```

```
<creneaux
  xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
   xmlns="http://fil.fr/miage-fa-fc" ...>
   <creneau xmlns="">…</creneau>
</creneaux>
```

OK

reinitialisation de l'espace de nom par défaut

```
<xsd:schema …
  targetNamespace="fil.fr/miage-fa-fc">
 <xsd:element name="creneaux">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="creneau" .../>
    <xsd:sequence>
  </xsd:complexType>
 …
</xsd:element>
```

creneau
- espace de noms associé

```
<x:creneaux
  xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
   xmlns:x="http://fil.fr/miage-fa-fc" ...>
   <x:creneau>…</x:creneau>
</x:creneaux>
```

```
<creneaux
  xsi:schemaLocation=
  "http://fil.fr/miage-fa-fc miage-fa-fc.xsd"
   xmlns="http://fil.fr/miage-fa-fc" ...>
   <creneau>…</creneau>
</creneaux>
```

Inclusions de schémas

On peut assembler plusieurs composants de schémas (définitions de types, déclarations d'éléments, ...), provenant de plusieurs documents.

- élément `include` qui permet d'inclure les définitions provenant d'autres schémas mais pas de plusieurs espaces de noms.

- Les schémas inclus doivent avoir
  1. soit le même espace de noms cible que le document qui les inclut
  2. soit pas d'espace de noms, dans ce cas, c'est l'espace de noms du schéma qui inclut tous les autres qui est pris en compte.

Exemple d'inclusion sans espace de noms cible

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="emp.xsd"/>
  <xsd:include schemaLocation="dept.xsd"/>

  <xsd:element name="ent">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="employes"/>
        <xsd:element ref="departements"/>
      </xsd:sequence>
    </xsd:complexType>
  ... les clefs étrangères ...
  </xsd:element>
</xsd:schema>
```

- `dept.xsd` et `emp.xsd` sont des fichiers dans le même répertoire.
- `dept.xsd` (resp. `emp.xsd`) contient les déclarations de l'élément `departements`(resp. `employes`) et de tous ses sous-éléments.

Exemple d'instance du schéma précédent

```
<?xml version="1.0" encoding="utf-8"?>
<ent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="entreprise.xsd">
  <employes>
    <emp num="1">
      <nom>toto</nom><prenom>jules</prenom>
      <salaire>3452</salaire>
      <dept>informatique</dept>
    </emp>
  </employes>
  <departements>
    <dept>
      <nom>informatique</nom>
      <contact>Mme Machin 45-76-77-09-54</contact>
      <chef>1</chef>
    </dept>
  </departements>
</ent>
```

Exemple d'inclusion avec espace de noms cible

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.lifl.fr/~yroos/schema"
        targetNamespace="http://www.lifl.fr/~yroos/schema"
        elementFormDefault="qualified"
>
 <xsd:include schemaLocation="emp.xsd"/>
 <xsd:include schemaLocation="dept.xsd"/>

  <xsd:element name="ent">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="employes"/>
        <xsd:element ref="departements"/>
      </xsd:sequence>
    </xsd:complexType>
    ... et les clefs étrangères ...
  </xsd:element>
</xsd:schema>
```

emp.xsd -> http://www.lifl.fr/~yroos/schema
dept.xsd -> http://www.lifl.fr/~yroos/schema
entreprise.xsd ->http://www.lifl.fr/~yroos/schema

Exemple d'inclusion avec espace de noms cible

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:sch="http://www.lifl.fr/~yroos/schema"
        targetNamespace="http://www.lifl.fr/~yroos/schema"
        elementFormDefault="qualified"
>
  <xsd:include schemaLocation="dept.xsd"/>
  <xsd:include schemaLocation="emp.xsd"/>

  <xsd:element name="ent">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="sch:employes"/>
        <xsd:element ref="sch:departements"/>
      </xsd:sequence>
    </xsd:complexType>
    ... et les clefs étrangères ...
  </xsd:element>
</xsd:schema>
```

emp.xsd -> http://www.lifl.fr/~yroos/schema
dept.xsd -> http://www.lifl.fr/~yroos/schema
entreprise.xsd ->http://www.lifl.fr/~yroos/schema

Instance du schéma précédent

```
<?xml version="1.0" encoding="utf-8"?>
<ent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.lifl.fr/~yroos/schema"
       xsi:schemaLocation="http://www.lifl.fr/~yroos/schema
       http://saxo.lifl.fr/~yroos/schema/entreprise.xsd">
  <employes>
    <emp num="1">
      <nom>toto</nom> <prenom>jules</prenom>
      <salaire>3452</salaire> <dept>informatique</dept>
    </emp>
  </employes>
  <departements>
    <dept>
      <nom>informatique</nom>
      <contact>Mme Machin 45-76-77-09-54</contact>
      <chef>1</chef>
    </dept>
  </departements>
</ent>
```

emp.xsd -> http://www.lifl.fr/~yroos/schema
dept.xsd -> http://www.lifl.fr/~yroos/schema
entreprise.xsd ->http://www.lifl.fr/~yroos/schema

Importation de schémas

- Un schéma est associé à un espace de noms cible

- L'élément `import` permet de faire référence à des composants d'un schéma qui appartient à un autre espace de noms que le schéma dans lequel on fait référence à ces composants.

- Dans l'exemple qui suit, on utilise un composant du schéma de XHTML pour notre propre schéma.

Exemple d'importation de schémas

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.lifl.fr/~yroos/schema"
      xmlns:art="http://www.lifl.fr/~yroos/schema"
      xmlns:html="http://www.w3.org/1999/xhtml"
      elementFormDefault="qualified"
>

  <xsd:import namespace="http://www.w3.org/1999/xhtml"
  schemaLocation=
   "http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd"/>
...
```

Exemple d'importation de schémas

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.lifl.fr/~yroos/schema"
      xmlns:art="http://www.lifl.fr/~yroos/schema"
      xmlns:html="http://www.w3.org/1999/xhtml"
      elementFormDefault="qualified"
>
...
 <xsd:element name="dansRevue">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="auteur" maxOccurs="unbounded"
            type="xsd:string"/>
       <xsd:element name="revue" type="xsd:string"/>
       <xsd:element name="titre" type="xsd:string"/>
       <xsd:element minOccurs="0" name="resume"
        type="html:Block"/>
     </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Exemple d'instance de ce schéma

```
<?xml version="1.0" encoding="utf-8"?>
<bibliographie xmlns:xsi=
      "http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.lifl.fr/~yroos/schema"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xsi:schemaLocation=
       "http://www.lifl.fr/~yroos/schema articles.xsd"
>
  <dansRevue>
    <auteur>Tryphon Tournesol</auteur>
    <revue>Revue de Physique</revue>
    <titre>Ma machine à voyager dans le temps</titre>
    <resume>
       <html:div>
          et patati <html:br/> et patata
       </html:div>
     </resume>
  </dansRevue>
</bibliographie>
```

En conclusion

Bonne pratique

L'utilisation des espaces de noms peut parfois être un peu compliquée.

Quand un espace de nom est très clairement majoritaire dans le document XML (que ce soit une instance ou un schéma), on le définit comme espace de nom par défaut.

Sinon une bonne pratique est de lier (dans un premier temps) tous les espaces de noms à des préfixes et de ne définir un espace de nom par défaut que lorsqu'il y en a un qui se détache (c.a.d. quand on tape majoritairement toujours le même préfixe)

Compléments sur XPath

Compléments sur XPath

- if
- expressions quantifiées
  - some, every
- for
- composabilité
  et transparence référentielle
- typage
- namespaces

```
<famille id="MARTIN">
   <femme id="1">
      <prenom>Juliette</prenom>
      <age>63</age>
      <poids>58</poids>
   </femme>
   <homme id="2">
      <prenom>Romeo</prenom>
      <age>65</age>
      <poids>97</poids>
   </homme>
   <homme id="3">
      <prenom>Max</prenom>
      <age>25</age>
      <poids>73</poids>
      <pere>2</pere>
      <mere>1</mere>
   </homme>
   <femme id="4">
      <prenom>Marie</prenom>
      <age>18</age>
      <poids>54</poids>
      <pere>2</pere>
      <mere>1</mere>
   </femme>
   <homme id="5">
      <prenom>Paul</prenom>
      <age>5</age>
      <poids>10</poids>
      <pere>3</pere>
   </homme>
</famille>
```

Instruction conditionnelle `if`

```
"if" "(" Expr ")" "then" Expr "else" Expr
```

Exemple :

```
if (count(//femme) < count(//homme)) then
  //homme/prenom
else
  //femme/prenom
```

```
<prenom>Romeo</prenom>
<prenom>Max</prenom>
<prenom>Paul</prenom>
```

```
<famille id="MARTIN">
   <femme id="1">
      <prenom>Juliette</prenom>
      <age>63</age>
      <poids>58</poids>
   </femme>
   <homme id="2">
      <prenom>Romeo</prenom>
      <age>65</age>
      <poids>97</poids>
   </homme>
   <homme id="3">
      <prenom>Max</prenom>
      <age>25</age>
      <poids>73</poids>
      <pere>2</pere>
      <mere>1</mere>
   </homme>
   <femme id="4">
      <prenom>Marie</prenom>
      <age>18</age>
      <poids>54</poids>
      <pere>2</pere>
      <mere>1</mere>
   </femme>
   <homme id="5">
      <prenom>Paul</prenom>
      <age>5</age>
      <poids>10</poids>
      <pere>3</pere>
   </homme>
</famille>
```

Expression quantifiée

C’est une expression booléenne dont la syntaxe est :

```
("some" | "every") "$"VarName

"in" Expr ("," "$"VarName "in" Expr)*

  "satisfies" Expr
```

Exemple :

```
some $x in //femme, $y in //homme

  satisfies $x/age + $y/age = 88

every $x in //femme, $y in //homme

  satisfies $x/age + $y/age = 88
```

La première expression vaut vrai, la seconde vaut faux.

```
<famille id="MARTIN">
   <femme id="1">
      <prenom>Juliette</prenom>
      <age>63</age>
      <poids>58</poids>
   </femme>
   <homme id="2">
      <prenom>Romeo</prenom>
      <age>65</age>
      <poids>97</poids>
   </homme>
   <homme id="3">
      <prenom>Max</prenom>
      <age>25</age>
      <poids>73</poids>
      <pere>2</pere>
      <mere>1</mere>
   </homme>
   <femme id="4">
      <prenom>Marie</prenom>
      <age>18</age>
      <poids>54</poids>
      <pere>2</pere>
      <mere>1</mere>
   </femme>
   <homme id="5">
      <prenom>Paul</prenom>
      <age>5</age>
      <poids>10</poids>
      <pere>3</pere>
   </homme>
</famille>
```

La boucle `for`

```
"for" "$"VarName "in" Expr
      ("," "$"VarName "in" Expr)*
"return" Expr
```

Exemple :

```
for $p in /famille/*,
    $f in /famille/*[$p/@id eq pere]
return ($p/prenom , $f/prenom)
```

```
<prenom>Romeo</prenom>
<prenom>Max</prenom>
<prenom>Romeo</prenom>
<prenom>Marie</prenom>
<prenom>Max</prenom>
<prenom>Paul</prenom>
```

```
<famille id="MARTIN">
   <femme id="1">
      <prenom>Juliette</prenom>
      <age>63</age>
      <poids>58</poids>
   </femme>
   <homme id="2">
      <prenom>Romeo</prenom>
      <age>65</age>
      <poids>97</poids>
   </homme>
   <homme id="3">
      <prenom>Max</prenom>
      <age>25</age>
      <poids>73</poids>
      <pere>2</pere>
      <mere>1</mere>
   </homme>
   <femme id="4">
      <prenom>Marie</prenom>
      <age>18</age>
      <poids>54</poids>
      <pere>2</pere>
      <mere>1</mere>
   </femme>
   <homme id="5">
      <prenom>Paul</prenom>
      <age>5</age>
      <poids>10</poids>
      <pere>3</pere>
   </homme>
</famille>
```

Composabilité / transparence référentielle

Exemple :

```
if (for $p in /famille/*, $f in /famille/*[$p/@id eq pere]
     return ($p/prenom , $f/prenom)) then
  for $p in /famille/*, $f in /famille/*[$p/@id eq pere]
  return ($p/prenom , $f/prenom)
else
   <result>pas de résultat</result>
```

```
<prenom>Romeo</prenom>
<prenom>Max</prenom>
<prenom>Romeo</prenom>
<prenom>Marie</prenom>
<prenom>Max</prenom>
<prenom>Paul</prenom>
```

Composabilité / transparence référentielle

Exemple :

```
if (some $x in //*,  $y in //*
      satisfies $x/@id eq $y/pere) then
  for $p in /famille/*, $f in /famille/*[$p/@id eq pere]
  return ($p/prenom , $f/prenom)
else
   <result>pas de résultat</result>
```

```
<prenom>Romeo</prenom>
<prenom>Max</prenom>
<prenom>Romeo</prenom>
<prenom>Marie</prenom>
<prenom>Max</prenom>
<prenom>Paul</prenom>
```

Composabilité / transparence référentielle

Exemple :

```
for $n in
    for $p in /famille/*, $f in /famille/*[$p/@id eq pere]
    return ($p/prenom , $f/prenom)
return $n/../@id
```

2
3
2
4
3
5

Typage

XPATH 2.0 est typé!

- `xs = http://www.w3.org/2001/XMLSchema`
- `fn = http://www.w3.org/2005/xpath-functions`

Tout item a une valeur typée et une valeur textuelle.

La valeur typée d'un item est une séquence de valeurs atomiques qui peuvent être extraites avec la fonction `fn:data`

La valeur textuelle d'un noeud est une chaîne de caractères qui peut être extraite avec la fonction `fn:string`

Typage des séquences

Les valeurs typées s'appuient sur les types prédéfinis de XML-Schema. S'y ajoutent des informations de types spécifiques aux nœuds ou aux séquences.

```
xs:integer
xs:decimal
xs:boolean
xs:string
xs:date
xs:time
xs:dateTime
...
```

```
empty-sequence()
document-node()
item()
node()
element()
element(name)
attribute()
+ , * , ?
```

Typage des séquences

Type d'une séquence
=
séquence des types des éléments de la séquence

Quelques exemples :

- `xs:date` type d'une séquence contenant un seul item de type `date`
- `attribute()?` type d'une séquence contenant 0 ou 1 noeud attribut
- `element()` type d'une sequence contenant 1 élément quelconque
- `element(prenom)*` type d'une sequence contenant un nombre arbitraire d'éléments de nom `prenom`
- `node()*` type d'une sequence contenant un nombre arbitraire de noeuds quelconques
- `item()+` type d'une sequence contenant un nombre arbitraire non nul de noeuds ou de valeurs atomiques

Typage : opérateurs

```
instance of
    ‘1’ instance of xs:string                  true()
    1.0 instance of xs:decimal                 true()
    true() instance of xs:boolean              true()
    '1' instance of xs:decimal                 false()

cast as
    10.1 cast as xs:integer                      10
    '1.2' cast as xs:decimal                    1.2
    '1' cast as xs:boolean                     true()


castable as
    'x1.2' castable as xs:decimal              false()
    '12' castable as xs:integer                true()
    '12.0' castable as xs:integer              false()
    12.0 castable as xs:integer                true()
```

Typage : opérateurs

```
is
  //*[@id='4'] is //femme[prenom='Marie']
                  true()

//*[@id='4']/pere is //*[@id='3']/pere
                  false()

//*[@id='4']/pere = //*[@id='3']/pere
                  true()
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<famille id="MARTIN">
   <femme id="1">
      <prenom>Juliette</prenom>
      <age>63</age>
      <poids>58</poids>
   </femme>
   <homme id="2">
      <prenom>Romeo</prenom>
      <age>65</age>
      <poids>97</poids>
   </homme>
   <homme id="3">
      <prenom>Max</prenom>
      <age>25</age>
      <poids>73</poids>
      <pere>2</pere>
      <mere>1</mere>
   </homme>
   <femme id="4">
      <prenom>Marie</prenom>
      <age>18</age>
      <poids>54</poids>
      <pere>2</pere>
      <mere>1</mere>
   </femme>
   <homme id="5">
      <prenom>Paul</prenom>
      <age>5</age>
      <poids>10</poids>
      <pere>3</pere>
   </homme>
</famille>
```

XPath et Namespaces - @

- `//@*`

  Martin, 1, 2, 3, 4, 5

- `//@id`

  Martin

- `declare namespace x="http:/humain.fr/identite";`
  `//@x:id`

  1, 2, 3, 4, 5

- `//@*/name()`

  id, id:id, id:id, id:id, id:id, id:id

- `//@*/local-name()`

  id, id, id, id, id, id

```
<famille id="MARTIN"
   xmlns="http://humain.fr/famille"
   xmlns:id="http:/humain.fr/identite"
   xmlns:car=
     "http:/humain.fr/caracteristique"
>
   <id:femme id:id="1">
      <id:prenom>Juliette</id:prenom>
      <car:age>63</car:age>
      <car:poids>58</car:poids>
   </id:femme>
   <id:homme id:id="2">
      <id:prenom>Romeo</id:prenom>
      <car:age>65</car:age>
      <car:poids>97</car:poids>
   </id:homme>
   <id:homme id:id="3">
      <id:prenom>Max</id:prenom>
      <car:age>25</car:age>
      <car:poids>73</car:poids>
      <id:pere>2</id:pere>
      <id:mere>1</id:mere>
   </id:homme>
   <id:femme id:id="4">
      <id:prenom>Marie</id:prenom>
      <car:age>18</car:age>
      <car:poids>54</car:poids>
      <id:pere>2</id:pere>
      <id:mere>1</id:mere>
   </id:femme>
   <id:homme id:id="5">
      <id:prenom>Paul</id:prenom>
      <car:age>5</car:age>
      <car:poids>10</car:poids>
      <id:pere>3</id:pere>
   </id:homme>
</famille>
```

XPath et Namespaces - node()

- `//femme`

  `()`

- `//*[local-name()="femme"]`

  `id:femme, id:femme`

- `//*[local-name()="femme"]/@id`

  `()`

- `//*[local-name()=« femme »]/@*[local-name()="id"]`

  `1, 4`

- `declare namespace x="http:/humain.fr/identite";`
  `//x:femme/@x:id`

  `1, 4`

```
<famille id="MARTIN"
    xmlns="http://humain.fr/famille"
    xmlns:id="http:/humain.fr/identite"
    xmlns:car=
      "http:/humain.fr/caracteristique"
>
    <id:femme id:id="1">
      <id:prenom>Juliette</id:prenom>
      <car:age>63</car:age>
      <car:poids>58</car:poids>
    </id:femme>
    <id:homme id:id="2">
      <id:prenom>Romeo</id:prenom>
      <car:age>65</car:age>
      <car:poids>97</car:poids>
    </id:homme>
    <id:homme id:id="3">
      <id:prenom>Max</id:prenom>
      <car:age>25</car:age>
      <car:poids>73</car:poids>
      <id:pere>2</id:pere>
      <id:mere>1</id:mere>
    </id:homme>
    <id:femme id:id="4">
      <id:prenom>Marie</id:prenom>
      <car:age>18</car:age>
      <car:poids>54</car:poids>
      <id:pere>2</id:pere>
      <id:mere>1</id:mere>
    </id:femme>
    <id:homme id:id="5">
      <id:prenom>Paul</id:prenom>
      <car:age>5</car:age>
      <car:poids>10</car:poids>
      <id:pere>3</id:pere>
    </id:homme>
</famille>
```

XPath et default Namespaces

- `//famille`

  `()`

- `declare namespace f="http:/humain.fr/famille"; //f:famille`

  `famille`

- `//@id`

  `Martin`

```
<famille id="MARTIN"
   xmlns="http://humain.fr/famille"
   xmlns:id="http:/humain.fr/identite"
   xmlns:car=
     "http:/humain.fr/caracteristique"
>
   <id:femme id:id="1">
      <id:prenom>Juliette</id:prenom>
      <car:age>63</car:age>
      <car:poids>58</car:poids>
   </id:femme>
   <id:homme id:id="2">
      <id:prenom>Romeo</id:prenom>
      <car:age>65</car:age>
      <car:poids>97</car:poids>
   </id:homme>
   <id:homme id:id="3">
      <id:prenom>Max</id:prenom>
      <car:age>25</car:age>
      <car:poids>73</car:poids>
      <id:pere>2</id:pere>
      <id:mere>1</id:mere>
   </id:homme>
   <id:femme id:id="4">
      <id:prenom>Marie</id:prenom>
      <car:age>18</car:age>
      <car:poids>54</car:poids>
      <id:pere>2</id:pere>
      <id:mere>1</id:mere>
   </id:femme>
   <id:homme id:id="5">
      <id:prenom>Paul</id:prenom>
      <car:age>5</car:age>
      <car:poids>10</car:poids>
      <id:pere>3</id:pere>
   </id:homme>
</famille>
```

Plein de fonctions!

---

http://www.mulberrytech.com/quickref/functions.pdf

Exemples:

```
current-date() as xs:date
current-time() as xs:time
avg(xs:anyAtomicType*) as xs:anyAtomicType
max(xs:anyAtomicType*) as xs:anyAtomicType?
sum(xs:anyAtomicType*) as xs:anyAtomicType
node-name(node()?) as xs:QName?
distinct-values(xs:anyAtomicType*) as xs:anyAtomicType*
doc(xs:string?) as document-node?
(node()) << (node()) as xs:boolean
(node()) >> (node()) as xs:boolean
```

Retour sur les espaces de noms

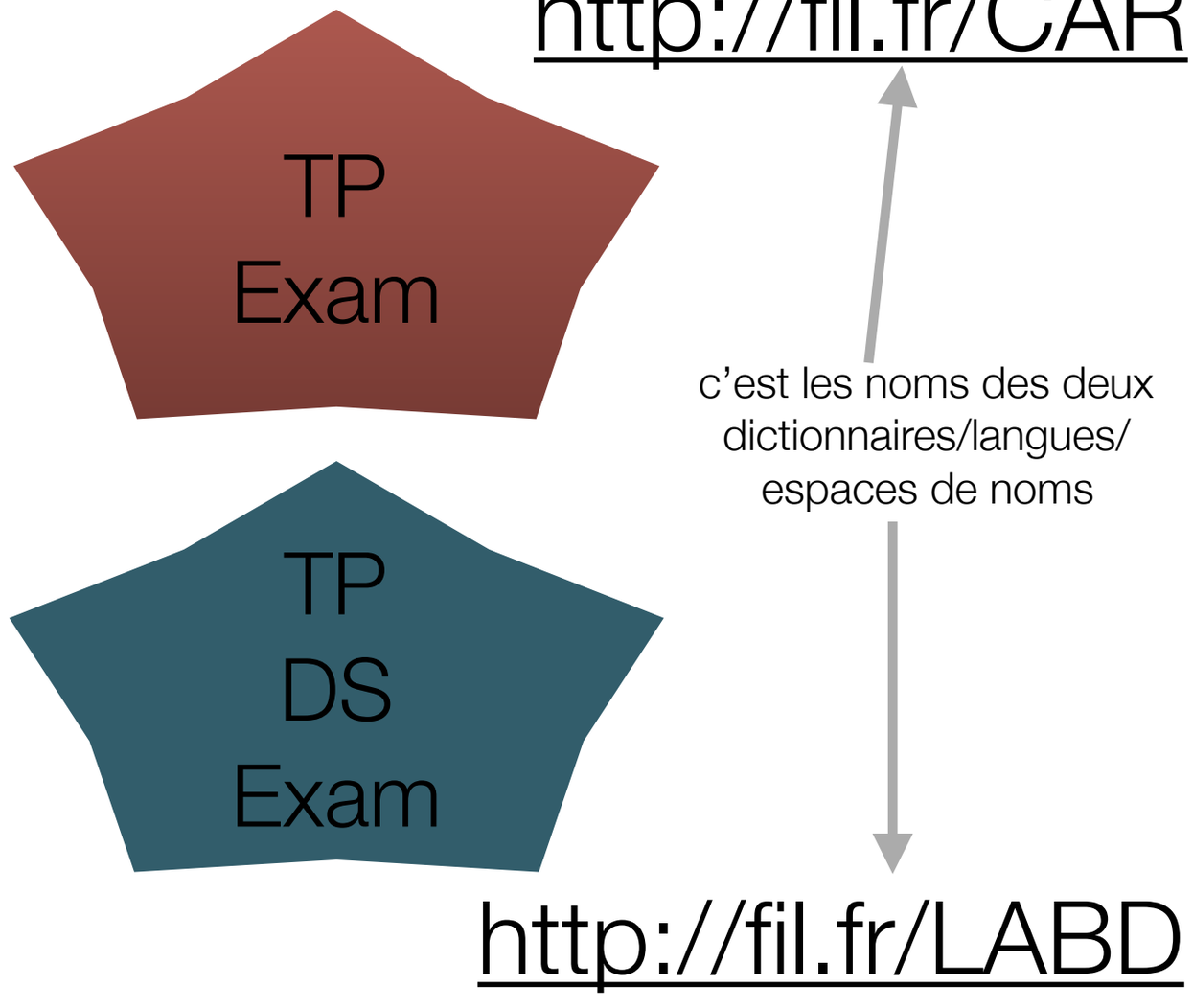
Prenons l'exemple des notes d'un étudiant de M1 au 2e semestre :

- selon l'option, l'évaluation est différente : TP et/ou EXAM et/ou DS
- selon l'option, l'importance des notes TP, EXAM, DS varie
- définir les éléments TP, EXAM, DS dans des espaces de noms différents, un par option.
    - ex : TP, EXAM spécifiques pour CAR
    - ex : TP, EXAM, DS spécifiques pour LABD

Les notes d'un étudiant de M1 au 2e semestre

Les éléments `exam` et `TP` de LABD et CAR sont écrits de la même façon, mais se calculent et ont une importance différente selon que l'on parle de LABD ou CAR. Donc, il vaut mieux les définir séparément, dans des « espaces »/« vocabulaires »/« langues » différents.

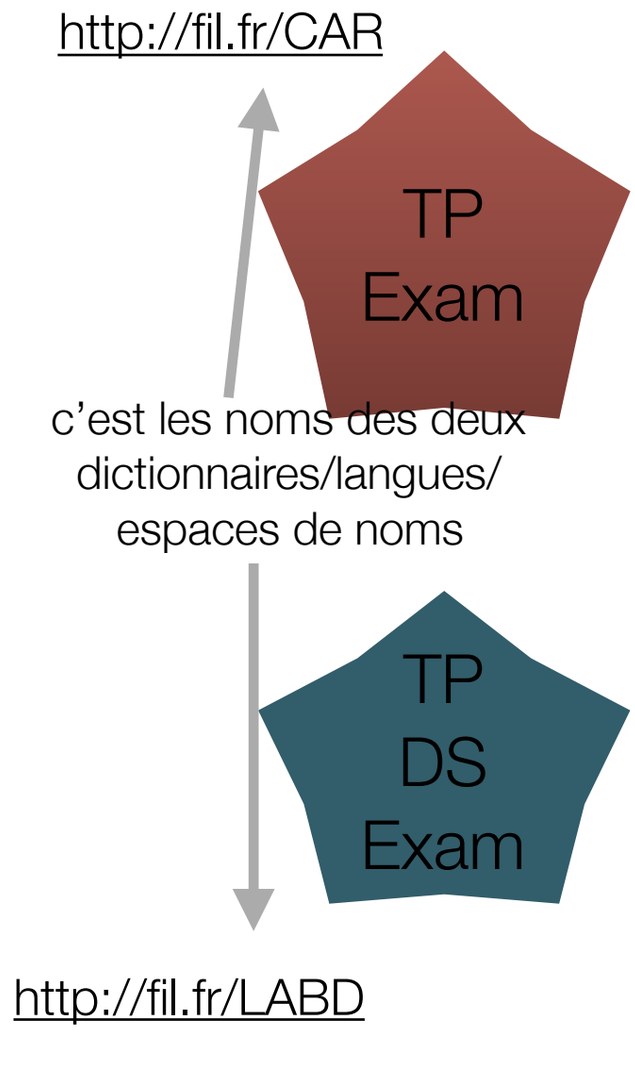
```
<etudiant eid="123">
  <semestre sid="m1s2">
    <car>
      <exam>12</exam>
      <tp>10</tp>
    </car>
    <labd>
       <exam>10</exam>
       <ds>12</ds>
       <tp>13</tp>
    </labd>
    ..
  </semestre>
</etudiant>
```



Les notes d'un étudiant de M1 au 2e semestre

Pour rendre l'attachement d' `exam, ds, TP` à leur espaces de noms, nous introduisons à l'aide de **`xmlns:`** explicitement les espaces de noms à travers des préfixes dans le document

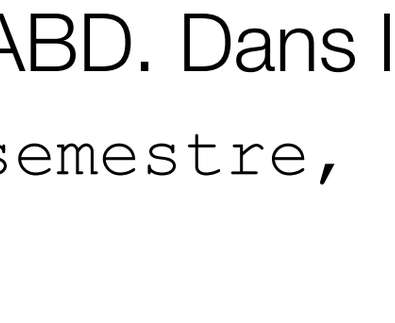
```
<etudiant eid="123"
          xmlns:car="http://fil.fr/car"
          xmlns:labd="http://fil.fr/labd">
  <semestre sid="m1s2">
    <car>
      <car:exam>12</car:exam>
      <car:tp>10</car:tp>
    </car>
    <labd>
       <labd:exam>10</labd:exam>
       <labd:ds>12</labd:ds>
       <labd:tp>13</labd:tp>
    </labd>
    …
  </semestre>
</etudiant>
```



Les notes d'un étudiant de M1 au 2e semestre

Maintenant, il est clair qu'il faut interpréter spécifiquement `exam` et `TP` de CAR, par rapport à `exam` et `TP` de LABD. Dans l'écriture actuelle, il ne reste que les éléments `étudiant, semestre, car, labd` sans espace de nom.

```
<etudiant eid="123"
          xmlns:car="http://fil.fr/car"
          xmlns:labd="http://fil.fr/labd">
  <semestre sid="m1s2">
    <car>
      <car:exam>12</car:exam>
      <car:tp>10</car:tp>
    </car>
    <labd>
       <labd:exam>10</labd:exam>
       <labd:ds>12</labd:ds>
       <labd:tp>13</labd:tp>
    </labd>…
  </semestre>
</etudiant>
```



http://fil.fr/CAR

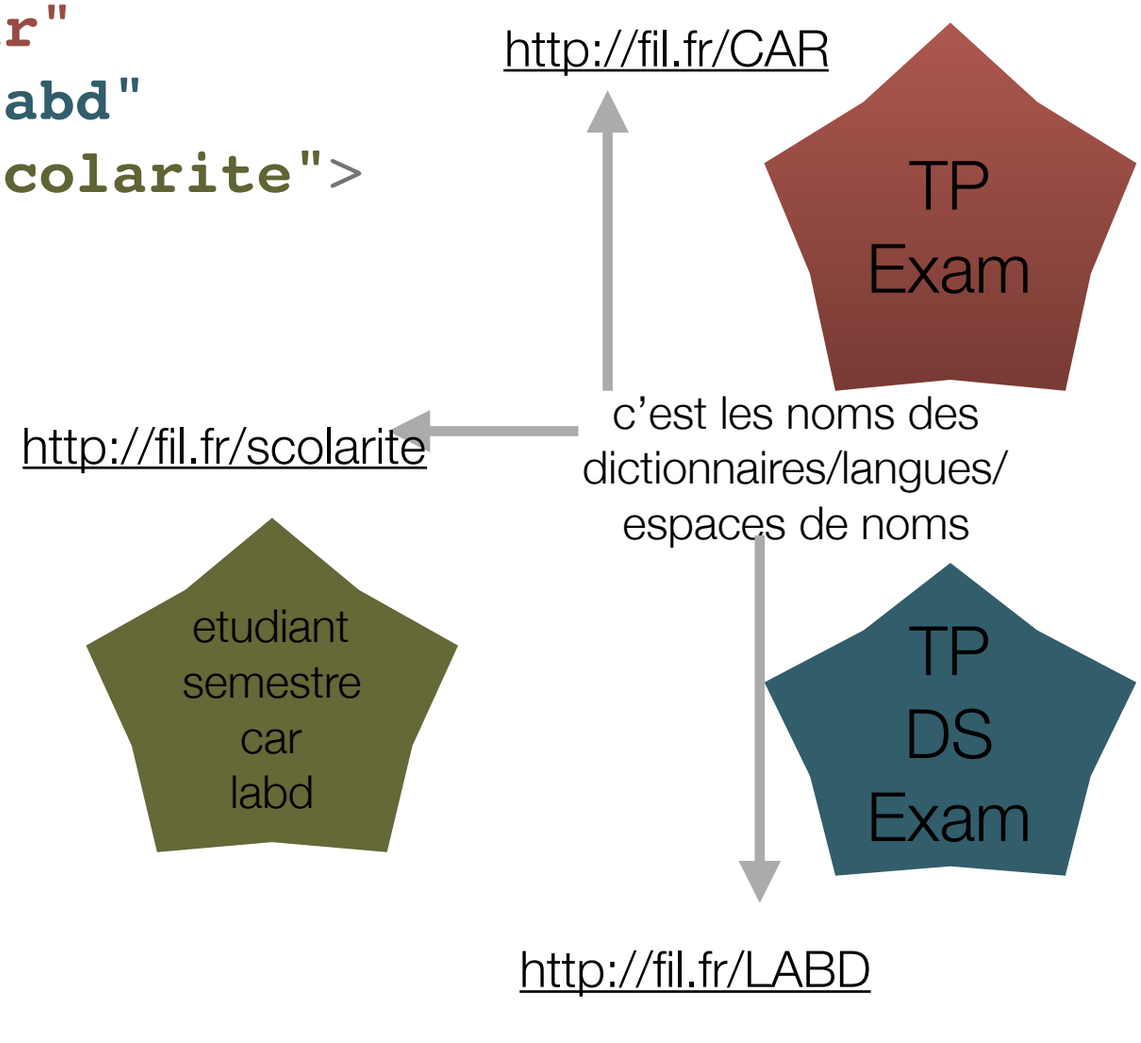
c'est les noms des deux dictionnaires/langues/ espaces de noms

http://fil.fr/LABD

Les notes d'un étudiant de M1 au 2e semestre

Supposons qu'il existe un espace de noms `http://fil.fr/scolarite` qui définit `examen, semestre, car et labd.`
Nous pouvons les lier à cet espace en utilisant le préfixe `scol`. Les attributs `eid, sid` ne sont pas ici liés à `scol`.

```
<scol:etudiant eid="123"
          xmlns:car="http://fil.fr/car"
          xmlns:labd="http://fil.fr/labd"
          xmlns:scol="http://fil.fr/scolarite">
  <scol:semestre sid="m1s2">
    <scol:car>
      <car:exam>12</car:exam>
      <car:tp>10</car:tp>
    </scol:car>
    <scol:labd>
       <labd:exam>10</labd:exam>
       <labd:ds>12</labd:ds>
       <labd:tp>13</labd:tp>
    </scol:labd>…
  </semestre>
</etudiant>
```

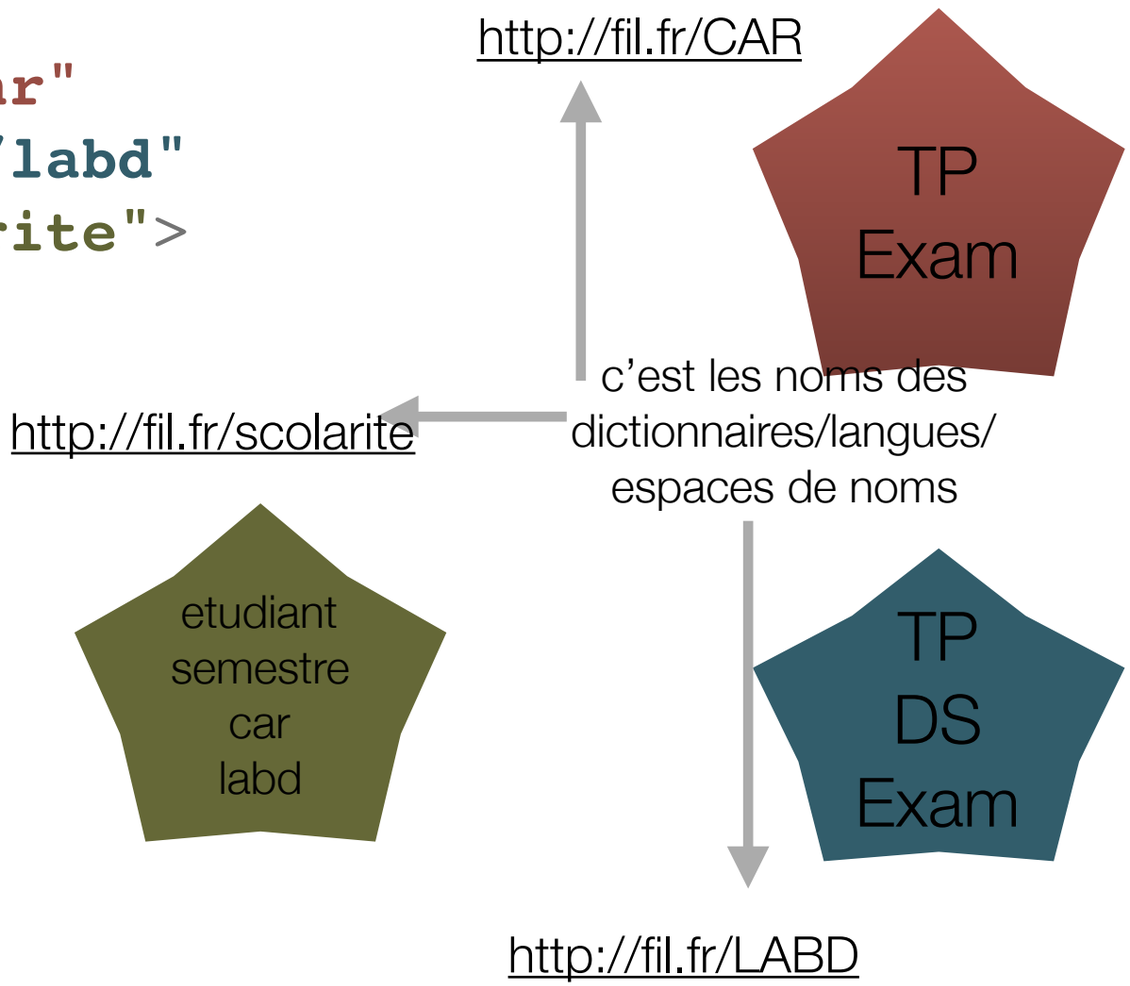


Les notes d'un étudiant de M1 au 2e semestre

L'association à l'espace de noms `http://fil.fr/scolarite` d'`examen, semestre, car, labd` aurait pu également se faire en définissant un espace de noms par défaut.

Les attributs `eid, sid` ne sont pas ici liés à `http://fil.fr/scolarite`.

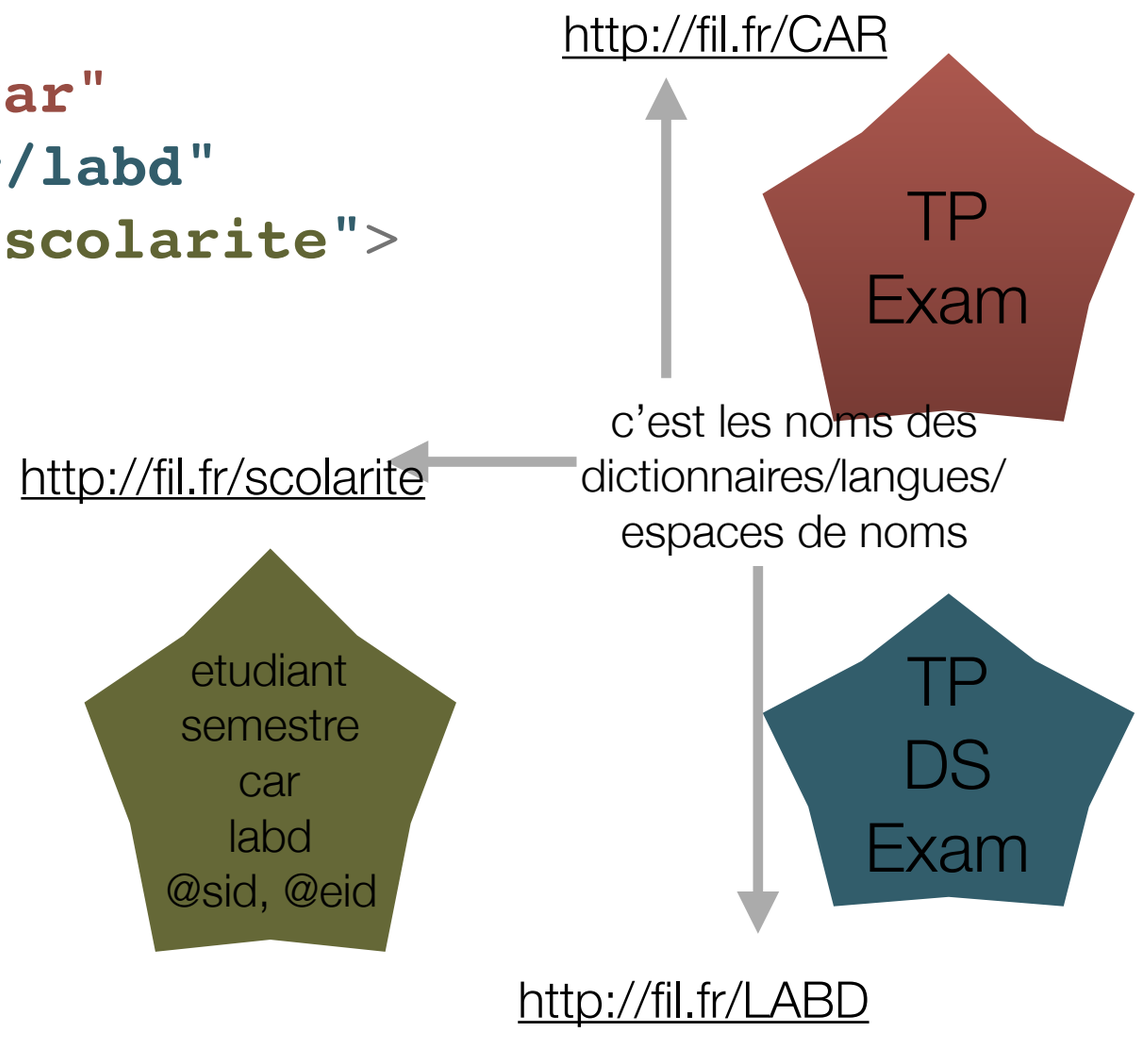
```
<etudiant eid="123"
          xmlns:car="http://fil.fr/car"
          xmlns:labd=« http://fil.fr/labd"
          xmlns="http://fil.fr/scolarite">
  <semestre sid="m1s2">
    <car>
      <car:exam>12</car:exam>
      <car:tp>10</car:tp>
    </car>
    <labd>
       <labd:exam>10</labd:exam>
       <labd:ds>12</labd:ds>
       <labd:tp>13</labd:tp>
    </labd>…
  </semestre>
</etudiant>
```



Les notes d'un étudiant de M1 au 2e semestre

Si les attributs `eid, sid` étaient également associés à l'espace `scol`, il faut utiliser les préfixes devant les attributs également.

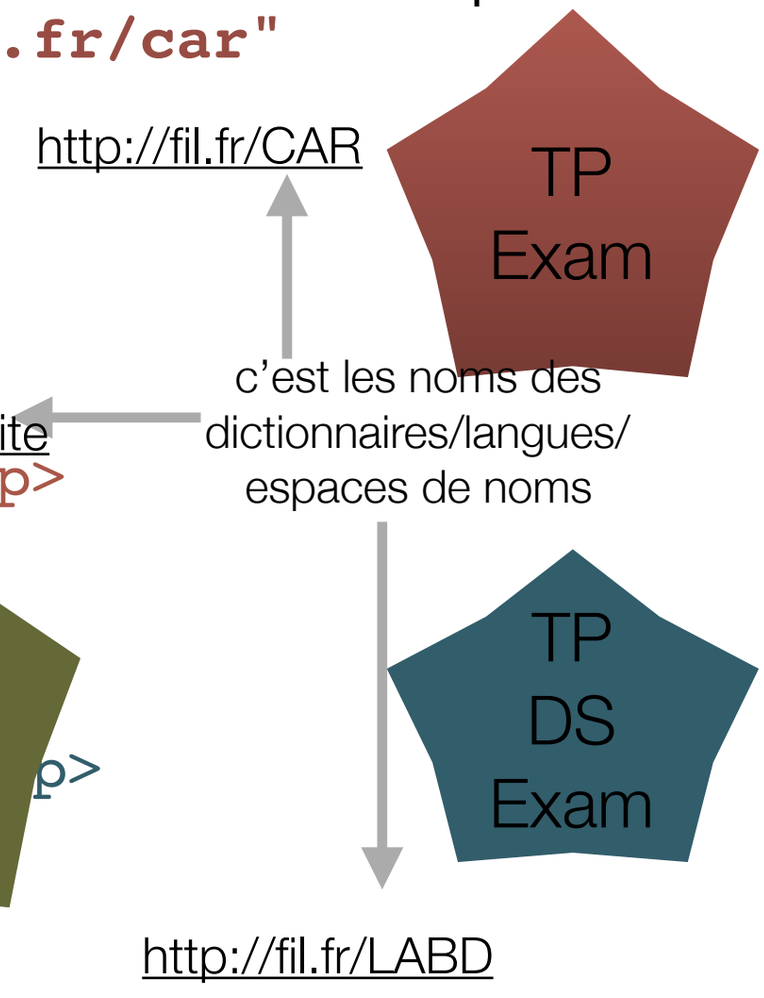
```
<scol:etudiant scol:eid=« 123"
          xmlns:car="http://fil.fr/car"
          xmlns:labd=« http://fil.fr/labd"
          xmlns:scol="http://fil.fr/scolarite">
  <scol:semestre scol:sid="m1s2">
    <scol:car>
      <car:exam>12</car:exam>
      <car:tp>10</car:tp>
    </scol:car>
    <scol:labd>
       <labd:exam>10</labd:exam>
       <labd:ds>12</labd:ds>
       <labd:tp>13</labd:tp>
    </scol:labd>
    ..
  </semestre>
</etudiant>
```



Les notes d'un étudiant de M1 au 2e semestre

Si les attributs `eid, sid` étaient également associés à l'espace scol, il faut utiliser les préfixes devant les attributs également. Et cela malgré l'usage d'un espace de noms par défaut. Les attributs ne sont pas affectés par l'espace de noms par défaut (**xmlns="http://fil.fr/scolarite"**), ce qui oblige d'inclure explicitement un préfixe associé à ce même espace.

```
<etudiant scol:eid="123" xmlns:car="http://fil.fr/car"
          xmlns:labd=« http://fil.fr/labd"
          xmlns:scol="http://fil.fr/scolarite"
          xmlns="http://fil.fr/scolarite">
  <semestre scol:sid="m1s2">
    <car>
      <car:exam>12</car:exam><car:tp>10</car:tp>
    </car>
    <labd>
       <labd:exam>10</labd:exam>
       <labd:ds>12</labd:ds><labd:tp>13<    p>
    </scol:labd>..
  </semestre>
</etudiant>
```

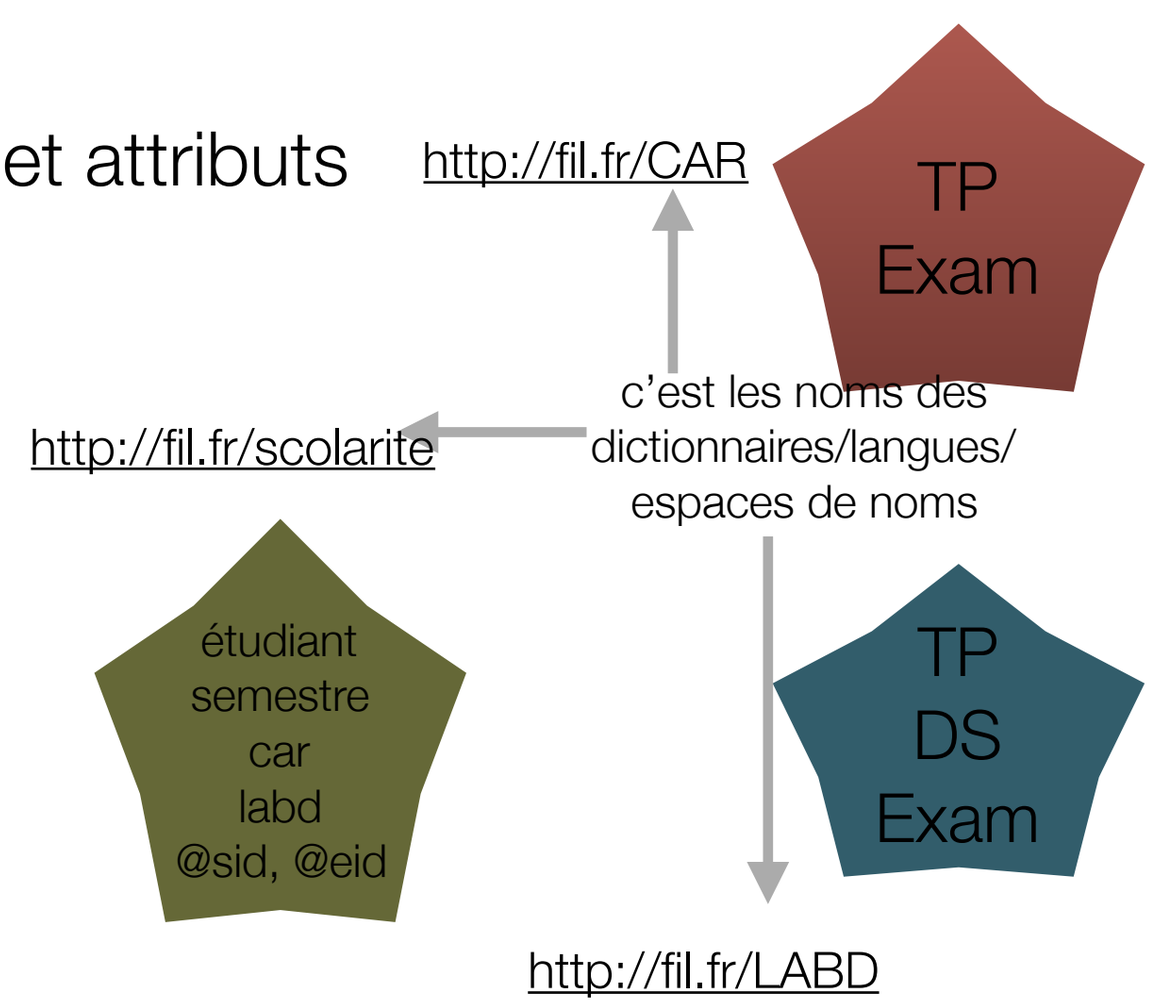


Les notes d'un étudiant de M1 au 2e semestre

Mais comment les noms suivants ont été définis dans leur espace de noms respectifs ?

- `étudiant, semestre, car, labd, @sid, @eid` à **http://fil.fr/scolarite**
- `tp, ds, exam` à **`http://fil.fr/LABD`**
- `tp, exam` à **`http://fil.fr/CAR`**

Pour rappel, la définition des éléments et attributs se fait en utilisant XMLSchema.



Les notes d'un étudiant de M1 au 2e semestre

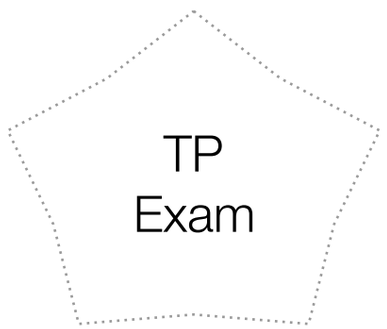
Ci-dessous un exemple de définition des éléments avec XMLSchema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="tp" type="xsd:decimal">
  <xsd:element name="exam" type="xsd:decimal">
</xsd:schema>
```

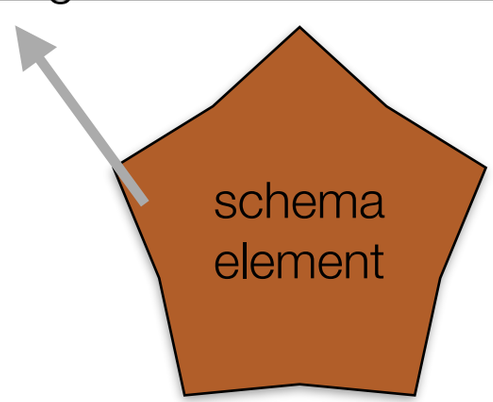
C'est bien une définition XMLSchema, car les éléments `schema, element` sont bien associé à l'espace de nom « http://www.w3.org/2001/XMLSchema » à travers le préfixe `xsd:`.

Une définition équivalente en liant l'espace de nom par défaut à « `http://www.w3.org/2001/XMLSchema` » est donnée ci-dessous.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="tp" type="decimal">
  <element name="exam" type="decimal">
</schema>
```



aucun espace de nom



Dans les deux cas, les éléments `tp et exam` définis ci-dessous ne sont pas liés à l'espace de nom souhaité (http://fil.fr/LABD). L'espace de nom par défaut, n'influe pas sur les attributs `name`. En revanche, les attributs `type` et `ref` sont affectés par le `xmlns` : d'où l'absence de préfix `xsd:` dans la définition des types dans le 2e exemple, et l'obligation d'utiliser le préfixe `xsd:` dans le 1er.

Les notes d'un étudiant de M1 au 2e semestre

Pour associer les noms définis dans les XML Schemas à des espaces de noms cibles il faut utiliser l'attribut **targetNamespace** dans la racine du XML Schema.

Il ne peut y avoir qu'un espace de nom cible par document. Donc, pour l'exemple discutée ici, il faudra 3 schemas, chacune ciblant un des espaces de noms.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://fil.fr/CAR">
  <xsd:element name="tp" type="xsd:decimal">
  <xsd:element name="exam" type="xsd:decimal">
</xsd:schema>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http//fil.fr/LABD">
  <xsd:element name="tp" type="xsd:decimal">
  <xsd:element name="exam" type="xsd:decimal">
  <xsd:element name="ds" type="xsd:decimal">
</xsd:schema>
```

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://fil.fr/scolarite">
  <element name="etudiant" ...></element>
  <element name="car" ...></element>
  <attribute name="eid" ...></attribute>
  ...
```