TECHNOLOGIES XML EXTENSIBLE MARKUP LANGUAGE PARTIE 2

SCHEMA XML

M.ABIK
m.abik@um5s.net.ma

2018/2019

SCHEMA XML

XML Schéma

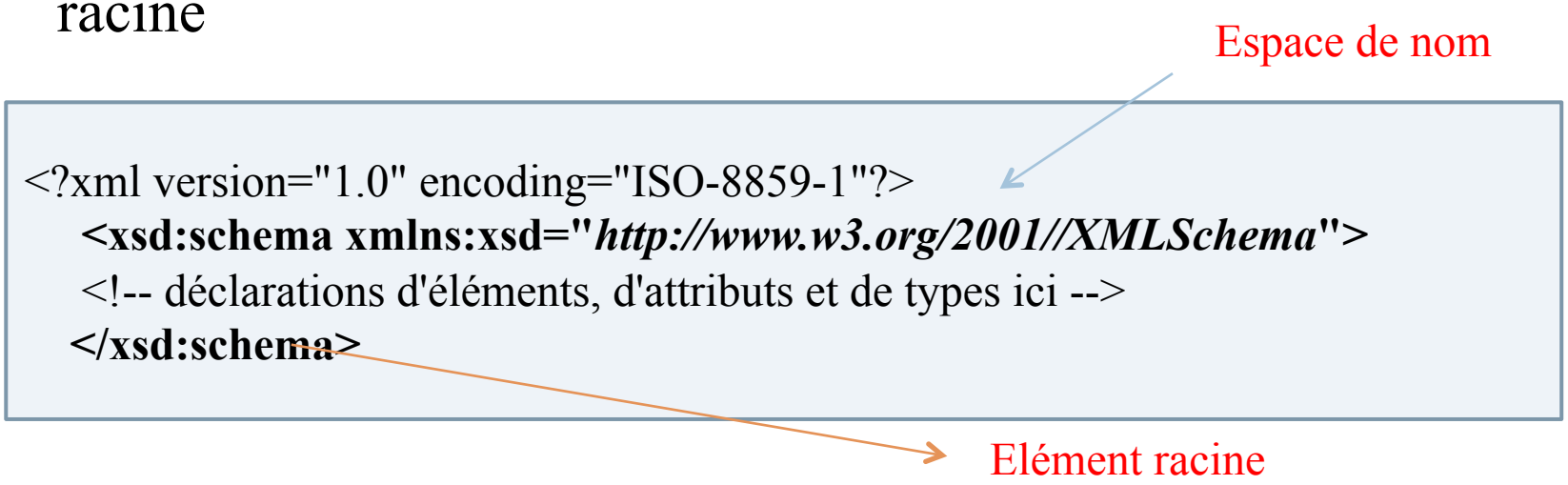
- Les schémas XML sont standardisés depuis 2001 par W3C.
- Un schéma d'un document définit:
  - les éléments possibles dans le document
  - les attributs associés à ces éléments
  - la structure du document
  - les types de données

Schéma XML : Avantages

- Les schémas XML sont décrits en XML.
  - Respecte la syntaxe XML mais ayant l'extension "**.xsd**"
  - Espace de nom spécifique xsd: ou xs:
- Beaucoup de types sont prédéfinis : date, booléen, entier, …
- Possibilité de créer de nouveau type
- Les éléments peuvent hériter du contenu et des attributs d'un autre élément.
- Le support des espaces de nom.
- Possibilité d'indiquer le nombre d'occurrences des éléments

Structure de base

- Un Schema xml commence par un prologue et un élément racine



- Le préfixe **xsd** (XML-Schema Definition) est généralement utilisé pour distinguer les éléments appartenant au langage XSD et ceux définis par un autre schéma. Parfois on utilise **xs** (c'est la même chose).

Schéma XML et Espace de nom

- Deux espace de nom sont dédiés aux schémas (obligatoire):
  - Espace de nom utilisé pour les éléments XML Schema
    - `http://www.w3.org/2001/XMLSchema`
    - Préfixe recommandé `xsd:` (XML-Schema Definition *pour distinguer les éléments appartenant au langage XSD et ceux définis par un autre schéma. Parfois on utilise xs).*
  - Espace de nom utilisé dans les documents d'instance
    - `http://www.w3.org/2001/XMLSchema-instance`
    - Préfixe recommandé `xsi:` (XML-Schema Instance)
- Espace de nom cible
  - Possibilité de déclarer des espaces de nom auquel le document XML doit correspondre

Assignation d'un schéma à un document XML

- L’assignation se fait au niveau de l'élément racine du fichier XML grâce à l'utilisation de deux attributs.
  - **L'espace de noms** utilisé pour les extensions XML schema utilisé dans les documents instance:
    - **xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance**
  - **La location (**on utilise l’un de ces attributs):
    - xsi:schemaLocation= "URI_espace_de_nom votre_fichier.xsd" : Prendre en compte un espace de nom cible du document
      - xsi:SchemaLocation=" http://ensias.ma xxx.xsd"
    - xsi:noNamespaceSchemaLocation="votre_fichier.xsd" : réservé aux références à des schémas décrivant des vocabulaires sans espaces de noms
      - xsi:noNamespaceSchemaLocation= "xxx.xsd"

Exemple : Ignorer les espaces de noms du document


```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xs="http://www.w3.org/2001/
XMLSchema"
<xs:element name="bibliographie">
                .....
</xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<bibliographie xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xsi:noNamespaceSchemaLocation ="Monschema1.xsd">
<livre>
</livre>
</bibliographie>
```

Exemple : Espace de nom cible


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns="http://ensias.ma/"
targetNamespace="http://ensias.ma/"
elementFormDefault="qualified">

<xs:element name="bibliographie">
                …
</xs:element>
</xs:schema>
```

Renseigne que le schéma décrit un espace de noms

les balises créées appartiennent d'office à l'espace de noms lié au schéma (défini par tagetNamespace »

```
<?xml version="1.0" encoding="UTF-8"?>
<bibliographie  xmlns="http://ensias.ma"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:SchemaLocation=" http://ensias.ma Monschema2.xsd">
….
</bibliographie>
```

Cas de plusieurs espace de nom

- Et si on veut définir plusieurs espace de nom ???
  - Càd on a plusieurs vocabulaires (éléments qualifiés avec des espaces de nom différents);
- On doit définir plusieurs XSD ;
  - Chaque XSD valide dans un espace de noms bien défini en définissant le targetNamespace de cet espace.
- On importe dans le XSD principal les autres XSD
- On **référence** dans le schéma principal les éléments qui sont définis dans les autres XSD que l'on importe.

Exemple


```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:th="http://vocab1"
 xmlns:pr="http://vocab2"
 targetNamespace="http://ensias.ma"
>
 <xs:import namespace="http://vocab1" schemaLocation="vocab1.xsd"/>
 <xs:import namespace="http://vocab2" schemaLocation="vocab2.xsd"/>
 <xs:element name="cours">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="th:theorique"/>
    <xs:element ref="pr:pratique" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://vocab1">
 <xs:element name="theorique" />
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://vocab2">
 <xs:element name="pratique" />
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<cours xmlns="http://ensias.ma" xmlns:th="http://vocab1" xmlns:pr="http://vocab2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ensias.ma C:\Users\m.abik\Desktop\web_semantique\Master_BigData\XML\TPS\exemples_schema_ns\cours.xsd">
  <th:theorique/>
  <pr:pratique/>
</cours>
```

Déclaration d'élément

- Chaque élément du document XML est décrit par la balise <**xsd:element** >

```
<xsd:element name="livre" maxOccurs="unbounded">
 <xs:complexType>
   <xs:sequence>
     <xs:element name="titre" type="xs:string"/>
     <xs:element name="auteur" type="xs:string"/>
   </xs:sequence>
<xs:attribute name="lang" type="xs:NMTOKEN" use="required"/>
</xs:complexType>
</xsd:element>
```

Déclaration d'élément

- les attributs d'un élément sont:
  - name: nom de l'élément
  - type: précise le type de l'élément (type simple ou type complexe).
    - Type complexe : quand l'élément a des enfants et/ou des attributs
    - Type simple: l'élément contient que du texte et sans d'attribut
  - fixed: valeur de l'élément fixée ou à préciser
  - Default : valeur par défaut lorsque l'élément est présent mais avec un contenu vide
  - minOccurs, maxOccurs : spécifier les cardinalités min et max
    - minOccurs=0 -> optionel && maxOccurs=unbounded -> l'infini
  - nillable : true ou false sert à combler le manque de valeurs
    - Par défaut c'est false

Déclaration d'élément(suite)

- Lorsqu'on n'attribue pas de type à un élément, il est considéré comme de type xs:anyType et peut donc contenir n'importe quoi
- L'élément peut être défini de deux manières :
  - *Localement* : dans la définition d'un type complexe à contenu complexe
  - *Globalement*: Directement lié à la racine du schéma, il sera alors référencé dans la définition d'un type complexe à contenu complexe.

```
<xs:element ref="nom_elt" >
```

EX: Déclaration d'élément(suite)


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="titre" type="xs:string"/>

<xs:element name="livre">
        <xs:complexType>
                <xs:sequence>
                  <xs:element ref="titre"/>
                    <xs:element name="auteur" type="xs:string"/>
                </xs:sequence>
                <xs:attribute name="key" type="xs:NMTOKEN" use="required"/>
                <xs:attribute name="lang" type="xs:NMTOKEN" use="required"/>
            </xs:complexType>
    </xs:element>
</xs:schema>
```

Définition globale d'élément
- peut être réutilisée partout

Déclaration d'attribut

- Les attributs des éléments sont déclarés à l'intérieur de la balise xsd:element après les sous-éléments.
- La définition d'attributs se fait via la balise <xsd:attribute>

<xs:attribute name="lang" type="xs:NMTOKEN" use="required"/>

- possède les attributs suivants:
  - name : nom de l'attribut.
  - type : type de l'attribut qui est forcément un type simple.
  - fixed : valeur de l'attribut fixée;
  - default : valeur par défaut ;
  - use : peut prendre une des valeurs:
    - required : obligatoire
    - optional : facultatif
    - prohibited : l'attribut ne doit pas apparaître

Déclaration d'attribut(suite)

- Un attribut peut être défini de deux manières :
  - *localement*, dans la définition d'un type complexe (à contenu simple ou complexe)

    <xs:attribute name="nom_att" >
  - *globalement*, c'est à dire directement sous la racine du schéma, il sera alors référencé dans la définition d'un type complexe (à contenu simple ou complexe).

Déclaration d'attribut (suite)

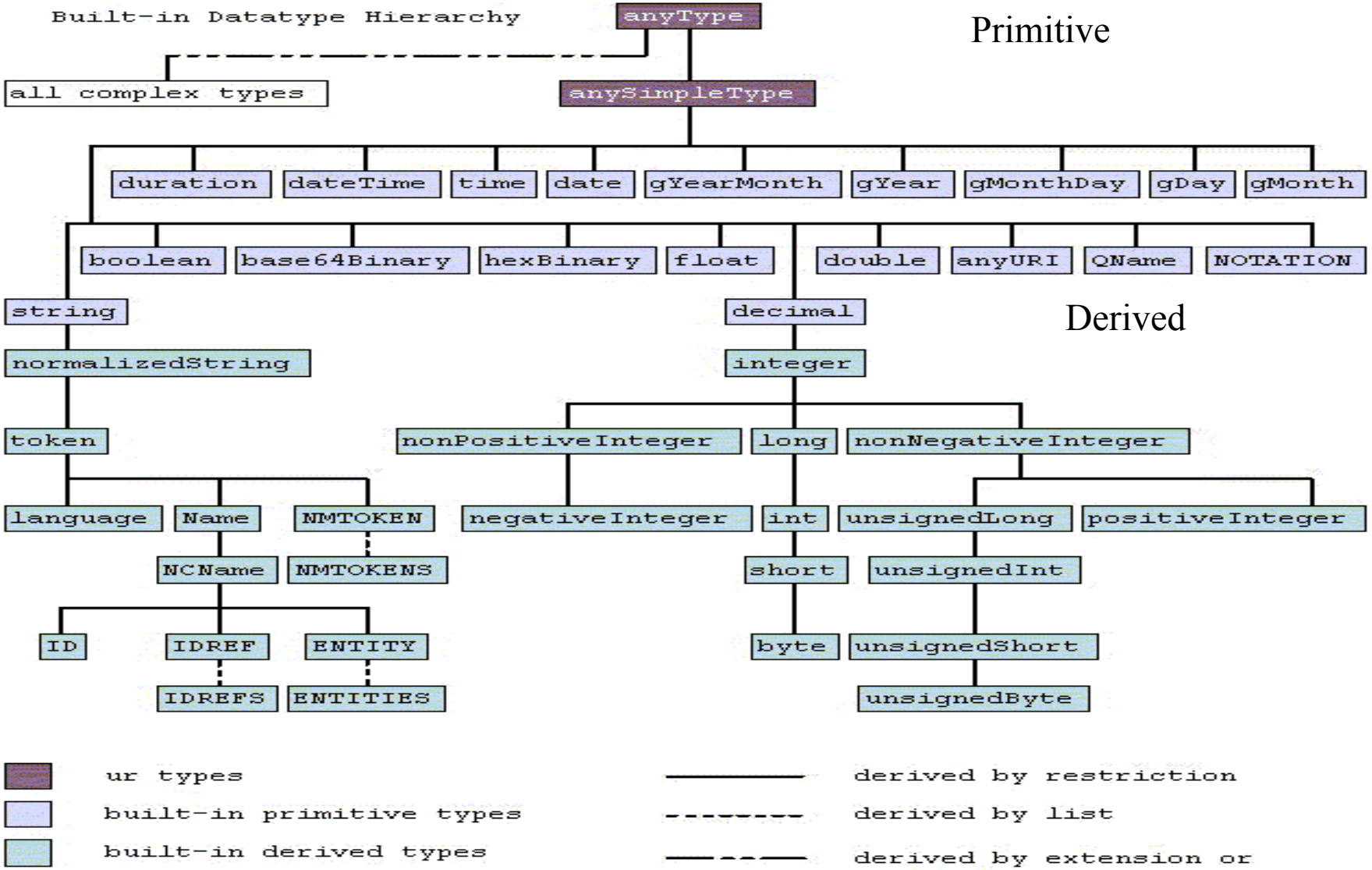
- Il est possible de regrouper des attributs permettant ainsi de faire des appels d'un groupe d'attributs lors des déclarations d'éléments.

```
<xsd:element name= "Test" type="type_Test"/>

<xsd:complexType name="type_Test">
    <!--modèle de contenu -->
<xsd:attributeGroup ref="xxx"/>
</xsd:complexType>

<xsd:attributeGroup name="xxx">
<xsd:attribute name="d" type="xsd:date"/>
<xsd:attribute name="a" type="xsd:string"/>
</xsd:attributeGroup>
```

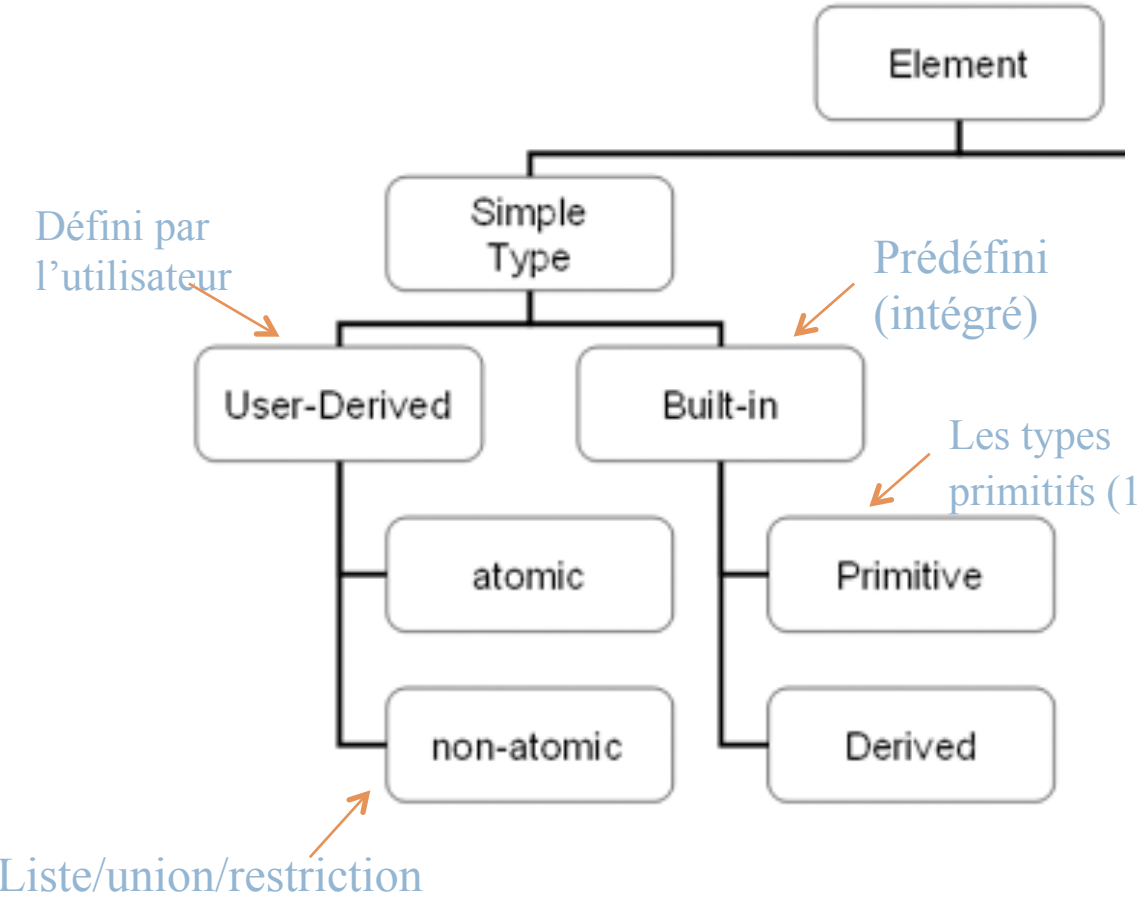
Typage

- Existence de types prédéfinis : hiérarchie de types dont la racine est le type anyType(integer, string...)
- Possibilité de définir de nouveaux types par restriction ou extension (ex: définition d'un mot de passe avec 8 caractères)
- Il y a deux types: types simples et types complexes
  - Les types simples sont utilisés pour les déclarations d'attributs, les déclarations d'éléments dont le contenu se limite à des données atomiques(intégrées), et qui n'ont pas d'attributs.
  - Les types complexes s'utilisent dans tous les autres cas

Types de base intégrés



Type simple



string
boolean
decimal
float
double
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
hexBinary
base64Binary
anyURI
QName
NOTATION

XML schéma offre plus de 44 types prédéfinis

XML Schéma: Types simples

- Un type simple peut être
  - Primitif (ne dérivant pas d'un autre)
  - Dérivés d'un autre prédéfini
  - Défini par l'utilisateur
    - Atomique : string, entiers…
    - (non-atomique)
      - Par une ***restriction*** *(usage des facettes appelé aussi contraintes)*
      - Par une ***liste*** : séquence de types atomique séparés par des blancs
      - Par une ***union*** : union d'autres types simples

Type défini par l'utilisateur
1cas: Dérivation du type simple par restriction

- La dérivation par restriction permet de créer de nouveaux types simples en imposant des contraintes à des types simples prédéfinis.
  - restreint l'ensemble des valeurs d'un type en limitant certaines caractéristiques (ex: longueur d'une chaine)
- La restriction est définie par des contraintes de facettes du type de base:
  - **length, minLength, maxLength** (s'applique au string ou list)
  - **enumeration** (un ensemble discret de valeurs s'applique à tous les types simple + union et list)
  - **pattern** (défini par une expression régulière hérité du langage Perl s'applique à tous les types simple + union et list)
  - **whitespace** (préserver, remplacer, réduire les espaces. s'applique aux types dérivés du type string)
  - **minInclusive,maxInclusive** (intervalles bornés de valeurs s'appliquent à tous les types numériques ainsi qu'à tous les types de date et d'heure)
  - **minExlusive,maxExclusive**
  - **totalDigits, fractionDigits** (s'applique aux types numériques dérivés de xsd:decimal)

Type simple: Facette (contrainte)

- Application d'une facette au type prédéfini:

```
<xsd:simpleType name=" nom">
  <xsd:restriction base=" type-predefini">
     Définition de la facette
     <xsd:restriction>
     </xsd: simpleType>
```

Exemple Restriction


```
<xsd:simpleType name= "Jours">
        <xsd:restriction base="xsd:string">
              <xsd:enumeration value= "Lundi"/>
              <xsd:enumeration value="Mardi"/>
             ...
         </xsd:restriction>
</xsd:simpleType>
```

Ex1: définition d'un nouveau type par restriction

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="Password">
        <xs:restriction base="xs:string">
                <xs:length value="8"/>
        </xs:restriction>
</xs:simpleType>
<xs:element name="user">
        <xs:complexType>
            <xs:sequence>
        <xs:element name="PW" type="Password"/>
            ….
          </xs:sequence>
        </xs:complexType>
 </xs:element>
</xs:schema>
```

Facette

Ex2: définition d'un nouveau type par restriction


```
<xs:element name="user">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="PW">
                    <xs:simpleType >
                        <xs:restriction base="xs:string">
                          <xs:minLength value="6"/>
                            <xs:maxLength value="12"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element >

            </xs:sequence>
        </xs:complexType>
 </xs:element>
</xs:schema>
```

Ex3: définition d'un nouveau type par restriction

~~<isbn>1422814094</isbn>~~

```
<xs:simpleType name="isbn">
<xs:restriction
base="xs:unsignedLong">
~~<xs:totalDigits value="10"/>~~
<xs:pattern value="\d{10}"/>
<xs:pattern value="\d{2}\s+
\d{2}\s+\d{2}\s+\d{2}\s+\d{2}"/>


</xs:restriction> </xs:simpleType>
```

<isbn>**14 22 81 40 94**</isbn>

<email>**xxxx@gmail.com** </email>

```
<xs:simpleType name="emailType">
<xs:restriction base="xs:string">
<xs:pattern value="[a-z]+@[a-z]+\.
[a-z]{2,3}"/>
</xs:restriction>
</xs:simpleType>
```

De 2 à 3

\s : blanc ;

\S : tout sauf un blanc ;

\d : un chiffre ;

\D : tout sauf un chiffre ;

\w : caractère alphanumérique plus "_" ;

\W : tout sauf un caractère alphanumérique plus "_".

Ex: définition d'un nouveau type par List


```
<xsd:simpleType name="numéros" >
<xsd:list itemType="xsd:unsignedByte" /> </
xsd:simpleType>
```

Entier non signé sur 8 bits

<telephone> 05 37 55 00 01</telephone>

```
<xs:simpleType name="Phone_number">
        <xs:restriction base="numéros">
            <xs:length value="5"/>
        </xs:restriction>
</xs:simpleType>
```

Exemple défini par l'utilisateur (Union)

```
<xs:simpleType name="taille_num">
    <xs:restriction base="xs:positiveInteger">
        <xs:maxInclusive value="44" />
</xs:restriction>
</xs:simpleType>
```

```
<xs:element name="taille"
type="ref_taille"/>
```

```
<xs:simpleType name="taille_str">
  <xs:restriction base="xs:string">
     <xs:enumeration value="small"/>
     <xs:enumeration value="medium"/>
     <xs:enumeration value="large"/>
   </xs:restriction>
</xs:simpleType>
```

```
<taille>      44    <taille/>
<taille>      large    <taille/>
```

```
<xs:simpleType name="ref_taille">
            <xs:union memberTypes="taille_num taille_str" />
</xs:simpleType>
```

Type complexe

- types des éléments qui contiennent d'autres éléments ou des attributs (au moins un attribut ou un fils)

```
<xs:complexeType name="…" mixed="…">
  <! Modèle de contenu-->
</xs:complexeType>
```

Modèle de contenu

- Les contenus de type complexe sont de quatre types:
  - Contenu vide avec attribut
  - Contenu simple (composé que d'attributs et d'un texte de type simple)
  - Contenu composé d'éléments
  - Contenu mixte (Attribut mixed= "true | false" contenu mixte ou non)

Type complexe: Contenu vide avec attribut

****

```
<xs:element name="img">
<xs:complexType>
  <xs:attribute name="src" type="xs:string"/>
 </xs:complexType>
</xs:element>
```

Type complexe: Contenu simple

<auteur nom="xxx" > chaine de caractère </auteur>

```
<xs:element name="auteur">
<xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
       <xs:attribute name="nom" type="xs:string"/>
       </xs:extension>
     </xs:simpleContent>
</xs:complexType>
</xs:element>
```

L'élément auteur est un type complexe dont le contenu simple (typé xs:string) a été étendu pour lui ajouter un attribut nom.

Type complexe: Contenu simple

<poids unite= "kg"> 70 </poids>

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

Type complexe: Contenu composé d'éléments


```
<xs:element name="livre" maxOccurs="unbounded">
<xs:complexType>
        <xs:sequence>
    <xs:element name="titre" type="xs:string"/>
    <xs:element name="auteur" type="xs:string"/>
            …
     </xs:sequence>
 <xs:attribute name="key" type="xs:NMTOKEN" use="required"/>
</xs:complexType>
</xs:element>
```

Connecteur

Trois types de connecteurs:

- **<xsd:sequence>:** définit une liste ordonnée de sous éléments
- **<xsd:choice>:** définit un groupe d'éléments dont un seul devra être présent
- **<xsd:all>:** définit un ensemble non ordonnés d'éléments

Type complexe: Contenu mixte


```
<xsd:element name="auteur">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name=" nom" type="xsd:string"/>
      <xsd:element name="prenom"  type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<auteur>
  Nom : <nom>xxxx</nom>,
 Prénom: <prenom>yyyy</prenom>.
</auteur>
```

```
xsd:complexType name="Bibliography">
 <xsd:sequence>
 <xsd:element name="livre" minOccurs="1"
          maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
  <xsd:element name="titre"     type="xsd:string"/>
  <xsd:element name="auteur"    type="xsd:string"/>
  <xsd:element name="url"  type="xsd:string" minOccurs="0"/>
 </xsd:sequence>
    <xsd:attribute name="key"  type="xsd:NMTOKEN" use="required"/>
     <xsd:attribute name="lang" type="xsd:NMTOKEN" use="required"/>
 </xsd:complexType>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>
```

```
<xsd:element name="bibliographie"  type="Bibliography"/>
</xsd:schema>
```

Webographie

- Livres

  Livre XML cours et Exercice :
  https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf
- Espace de nom :
  http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf
- DTD :
  http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf
- XPATH:

  http://www.w3.org/TR/xpath/

  http://www.loria.fr/~abelaid/Enseignement/miage-m1/XSL-Xpath.pdf
- XSLT:

  http://miage.univ-nantes.fr/miage/D2X1/chapitre_xslt/section_regles.htm#22
- DOM et SAX

  http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/06-xml-dom-sax.pdf

  https://www.lri.fr/~benzaken/documents/sldomsax.pdf

Webographie

- Livre XML cours et Exercice : https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf
- http://www.teluq.ca/inf6450/mod2/chapitre6.xml
- Espace de nom : http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf
- Dtd : http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf
- DOM et SAX http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/06-xml-dom-sax.pdf
  - https://www.lri.fr/~benzaken/documents/sldomsax.pdf
  - http://www-lium.univ-lemans.fr/~lehuen/master1/xml/cours/xmldiapo3.pdf
- XQUERY
  - Chapitre 6 de base de donnés avancées XML-Requête Xquery, Yannik Benezeth http://ilt.u-bourgogne.fr/benezeth/teaching/DUTIQ/BDDS3/CM6.pdf
  - Xquery : Une extension de Xpath à la mode SQL Un concurrent d'XSLT ?, Yves Bekkers https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=18&cad=rja&uact=8&ved=0ahUKEwjFw83x6NfQAhXGbRQKHVG2DQ04ChAWCE0wBw&url=http%3A%2F%2Fwww.irisa.fr%2FLIS%2FMembers%2Fbekkers%2Fxmldossier%2Fxquery%2Fattachment_download%2Ffile&usg=AFQjCNE11H2QwPJ21fNx68jzdzItpuOdQw&sig2=63NlKIMEdlwb4dPfFFxe5w
  - https://www.ibisc.univ-evry.fr/~serena/coursBDA0910_4.pdf
- XML et SGBD
  - http://miage.univ-nantes.fr/miage/D2X1/chapitre_bdxml/section_sgbd_xml.htm
  - http://peccatte.karefil.com/software/rbourret/xmlbd.htm
  - http://code.ulb.ac.be/dbfiles/Ver2006amastersthesis.pdf

Limites XQUERY

- ne respecte pas la syntaxe XML.
  - XQueryX a donc été proposé qui est une syntaxe alternative pour Xquery où une requête est représentée comme un document XML bien formé.
  - Mais !!!!