


<pre> </xsd:sequence> <xsd:extension> </xsd:complexContent> </xsd:complexType> </pre>	<pre> <xsd:element name="name" type="nomDuNewType" /> </pre>	<h1>#Jaizappé le XML et XSchema</h1>
Faire un choix :	<pre> <xsd:simpleType name="valAutorise"> <xsd:restriction base="xsd:integer"> <xsd:minInclusive value="-5" fixed="true"/> <xsd:maxInclusive value="5"/> </xsd:restriction> </xsd:simpleType> </pre>	<h2>Histoire</h2> <p>La première version de XML (1.0) est apparu en février 1998. Le XML est inspiré du SGML et HTML. XML signifie eXtensible Markup Language.</p>
<pre> <xsd:complexType name="pere"> <xsd:choice minOccurs="0" maxOccurs="unbounded"> <xsd:element name="n1" type="xsd:string"/> <xsd:element name="n2" type="xsd:string"/> </xsd:choice> </xsd:complexType> </pre>	<h2>Commentaires</h2> <p>Un commentaire est un message caché à l'exécution.</p> <pre> <!-- Un com. de 1 ou + ligne(s) --> </pre>	
<pre> <!ELEMENT element(n1 n2)*> </pre>	<h2>Élément vide</h2> <pre> <xsd:element name="rienAvec1Attri"> <xsd:complexType> <xsd:attribute name="attribut1" type="xsd:string"/> </xsd:complexType> </xsd:element> </pre>	<h2>Base</h2> <p>Un fichier XML a pour extension <i>.xml</i> et commence par :</p> <pre> <?xml version="1.0"> </pre>
<p>On peut aussi restreindre certaines valeurs, ici on redéfinit le maximum d'éléments :</p> <pre> <xsd:ComplexContent> <xsd:restriction base="base"> <xsd:sequence> <xsd:element name="elemDeBase" type="xsd:integer" maxOccurs="100"/> </xsd:sequence> </xsd:restriction> </xsd:ComplexContent> </pre>	<h2>Formes normales</h2> <ul style="list-style-type: none"> — 1NF : Tous les attributs sont atomiques. — 2NF : Un attribut non clé ne dépend pas d'une partie de la clé. — 3NF : Un attribut non clé ne dépend pas d'un ou plusieurs attributs ne participant pas à la clé. — BCNF : Tous les attributs non-clé ne sont pas source de dépendance fonctionnelle vers une partie de la clé. 	<p>Ensuite on ajoute des éléments :</p> <pre> <?xml version="1.0"> <!DOCTYPE livreDef> <livre> <pages> <page numero="1"> Première page </page> <page numero="2"> Seconde page </page> </pages> <auteur>Michel</auteur> <isbn value="XXXXXXXXx" /> </livre> </pre>
<p>Pour un xsd:string :</p> <pre> <xsd:restriction base="xsd:string"> <xsd:enumeration value="value1"/> <xsd:enumeration value="value2"/> </xsd:restriction> </pre>		<p>Ici, on a décrit un livre. <i>numero="X"</i> est un attribut de <i>page</i>. <i><livre></i> est une balise ouvrante, <i></livre></i> en est une fermante. <i>Michel</i> est le contenu de <i>auteur</i>.</p>
<p>Enfin le xsd:complexType peut recevoir l'argument :</p> <ul style="list-style-type: none"> — <i>final="restriction"</i> : Restriction impossible ; — <i>final="derivation"</i> : Dérivation impossible ; — <i>final="#All"</i> : Les deux. 		<h2>Appel externe</h2> <p>On peut appeler une application externe lors de l'affichage</p> <pre> <?application code ?> <!-- Exemple --> <?php echo "Hello World"; ?> </pre>
<h2>Créer un nouveau type</h2> <p>On le structure ainsi :</p> <pre> <xsd:simpleType name="nomDuNewType"> <xsd:restriction base="xsd:typeDeBase"> <xsd:pattern value="regex"/> </xsd:restriction> </xsd:simpleType> </pre>		<h2>DTD</h2> <p>DTD permet de formaliser un fichier <i>XML</i>, elle se place juste après la ligne : <i>< ?xml ...></i>.</p>
<p>On doit lui donner un nom, et un type de base ; le plus simple est un String.</p> <p>Ensuite, on met un regex pour formaliser. On l'utilise :</p>	http://github.com/Servuc/jaizappe	<p>  Licence GPLv3 </p>

Élément simple

C'est un élément sans balise dans son contenu :

```
<!ELEMENT page(#PCDATA)>
```

Élément vide

C'est un élément sans contenu ni balise :

```
<!ELEMENT isbn EMPTY>
<!ATTLIST isbn value(CDATA)>
```

Élément d'élément(s)

C'est un élément avec QUE des balises :

```
<!ELEMENT livre(pages, auteur+)>
<!ELEMENT pages(page+)>
```

Le + après *page* signifie que l'on a au moins un élément *page* : [1,+∞[. * = [0,+∞[, ? = [0,1]. Si on ne met rien, cela veut dire 1.

Élément d'élément(s) ou simple

Admettons que l'on est une *page* seulement, on peut directement mettre le contenu

```
<!ELEMENT pages(#PCDATA|page+)>
```

```
<?xml version="1.0">
<!DOCTYPE livreDef>
<livre>
    <pages>Page 1</pages>
    <auteur>Michel</auteur>
</livre>
```

Entités

Elles servent de raccourci :

```
<!ENTITY cle="valeur">
<!-- Et dans le XML -->
<noeud attr="&autreCle;">&cle;</noeud>
```

Attribut d'élément

Équivalent à des paramètres :

```
<!ATTLIST balise nomAttri (type) option>
<!-- exemple -->
<!ATTLIST page numero (CDATA) #REQUIRED>
```

Les types sont :

- **CDATA** : Une chaîne de caractères;
- **ID** : Attribut unique;
- **IDREF** : Fait référence à un **ID**;
- **valeur1|valeur2|...** : On met les possibilités;

- **ENTITY** : Fait référence à une entité.

Les options sont :

- **#REQUIRED** : Attribut obligatoire;
- **#IMPLIED** : (Par défaut), attribut non obligatoire;
- **#FIXED valeur** : Indique une valeur par défaut.

Doctype

```
<!DOCTYPE livreDef [ <!-- DTD ici --> ]>
```

Ensuite, on met le **XML**. On peut l'inclure par un lien système (Ensuite on met le **XML**) :

```
<!DOCTYPE livreDef SYSTEM "livreDef.dtd">
```

Fichiers externes

Il s'agit d'images, sons, documents, ... :

```
<!DOCTYPE vrml [
  <!ELEMENT vrml (#PCDATA)>
  <!NOTATION vrml PUBLIC "VRML 1.0">
]>
<!DOCTYPE img [
  <!ELEMENT img EMPTY>
  <!ATTLIST img src ENTITY #REQUIRED>
  <!ENTITY logo SYSTEM "logo.gif" NDATA gif>
  <!NOTATION gif PUBLIC "/usr/bin/gimp">
]>
<!-- XML -->

<vrml>Instructions VRML</vrml>
```

XML Schema

On peut voir le **XML Schema** ou **XSD** comme une amélioration de la **DTD**.

Commentaires

```
<xsd:annotation>Com.</xsd:annotation>
```

Base

```
<?xml version="1.0">
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.librairie.org"
  xmlns="http://www.librairie.org">
  <!-- code XSD -->
</xsd:schema>
```

Du fait de sa taille, je ne l'écrirais plus ensuite.

Élément de base

xsd:string	Chaîne normale
xsd:normalizedString	Idem que prec. sans \r \n \t
xsd:token	Idem que prec. sans espace.
xsd:time	hh:mm:ss
xsd:date	AAAA-MM-JJ
xsd:dateTime	AAAA-MM-JJThh:mm:ss
xsd:duration	PnAnMnJnTnHnMnS
xsd:int xsd:long xsd:byte xsd:short xsd:decimal	
xsd:unsignedXXXXXX	XXXXXX : Int Short ...
xsd:boolean	1 : true, 0 : false
xsd:hexBinary	Données Hexa
xsd:anyURI	URL

Le **T** dans *dateTime* fait la séparation.
Pour *duration*, *nX* : nombre de XXXXX.
On l'utilise ainsi :

```
<xsd:element name="nom" type="xsd:xxxxxxx"/>
```

On peut ajouter les attributs suivants aux **xsd:element** :

- **maxInclusive**, **minInclusive**, **maxExclusive**, **minExclusive** : pour encadrer les nombres;
- **length**, **maxLength**, **minLength** : Longueur du texte;
- **pattern** : Une regex.

Élément complexe

Il faut voir ça comme les éléments souvent basés sur des ensembles :

```
<xsd:element name="nom">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="autreNom"/>
      <xsd:element name="autreNom2"
        type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Sur les *xsd:element* avec l'attribut *ref* peuvent avoir ces attributs :

- **minOccurs** : unbounded ou un nombre : Nombre minimum d'occurrence;
- **maxOccurs** : unbounded ou un nombre : Nombre maximum d'occurrence;

On peut même faire de l'héritage :

```
<xsd:complexType name="nomFils">
  <xsd:ComplexContent>
    <xsd:extension base="nom">
      <xsd:sequence>
        <xsd:element name="autreNom3"
          type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:ComplexContent>
</xsd:complexType>
```