Cours XML XSL

XML

Utilité du XML

On a une structuration de document au moyen de balises.
Ex.

```
<livre>
    <chapitre type= "1">          <!-- les valeurs sont toujours entre " " -->
        <text>
            Blabla…
        </text>
    </chapitre>
    …
    <chapitre type= "n">
        <text>
            Blabla…
        </text>
    </chapitre>
</livre>
```

Utilisation de XML

Les principaux éditeurs : Word, Finale (éditeur de partition musicale), etc. ont la possibilité d'enregistrer les fichiers sous format XML. Les flux RSS sont aussi en XML.

Détail du document XML

- **Les commentaires**
  Un commentaire est encadré par des balises `<!- -` pour l'ouvrir et `- ->` pour le fermer.

- **Les déclarations**
  `Ex. : <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>`
  - `version : c'est le n° de version d'XML`
  - `encoding : code des caractères. Voir http://fr.selfhtml.org/xml/regles/declarationxml.htm`
  - `Standalone : dépendance par rapport à un style de document`

- **Les traitements**
  `Ex. : <?xml-stylesheet type="text/xsl" href="biblio.xsl"?>`
  - `xml-stylesheet` encoding : style de document XML
  - type : XSL propre au XML ou CSS pour du HTML
  - href : le fichier référence pour la mise en forme du document

- **La racine**
  Elle est unique. Dans notre ex. <livre>.

- **Les éléments**
  - Les balises doivent commencées par une lettre. Ou _ ou : s'il y a 2 lettres. 💣 Case sensitive
  - ils forment la structure du document. : `<titre>Les Misérables</titre>`
  - Ils peuvent contenir des attributs. : `<livre lang="en">`
  - Certains peuvent être vide. : `<couverture couleur="rouge" />` (Ne pas oublier le `/`)
    `Autre ex. <img src="rose.png" alt="rose rouge" width="56" height="100" />`

- **Les sections CDATA**
  `<![CDATA[Une balise commence par un < et se termine par un >`
  `Permet de définir un bloc qui ne doit pas être analysé par le processeur XML. Cela permet de conserver du texte tel quel. Ex. un document qui a une partie code en PHP à afficher.`

Les schémas XML

Le problème d'XML est que n'importe qui peut définir n'importe quelle balise. Il faut, si l'on veut être rigoureux, définir un seul model de fichier XML : le schéma.
C'est un fichier *nom.xsd* contenant une description de ce qu'il doit y avoir dans un fichier XML. Ainsi un groupe de personnes travaillant sur un fichier XML vont avoir une structure unique pour tous.

Ex.

```
1  <xsd:element name="pages" type="xsd:positiveInteger" />
2  <xsd:element name="auteur" type="xsd:string" />
3  <xsd:element name="livre">
4    <xsd:complexType>
5      <xsd:sequence>
6        <xsd:element ref="auteur" />
7        <xsd:element ref="pages" />
8      </xsd:sequence>
9    </xsd:complexType>
10 </xsd:element>
```

En ① & ③ on énonce les éléments "simples" et leurs types, qui seront utilisés.
En ③ on défini qu'il y a un élément livre composé d'un type complexe④ qui sont en séquence⑤ : l'auteur dont le type est défini en ② et le nbr. De pages vu en ①.

Comme tout documents XML, il y a une en-tête

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
    Déclarations d'éléments, d'attributs et de types
  </xsd:schema>
```

- **Les types simples**

Voir le dessin dans http://gilles-chagnon.developpez.com/cours/xml/dtd-et-schemas/?page=schemas
et http://www.w3.org/TR/xmlschema-0/#CreatDt

Nota : On peut indiquer des restrictions fortes sur les types simples. Voir les facette page suivante.

- **Le type liste**

On veut une liste d'éléments ayant le même type.

```
Ex. <xsd:simpleType name="Telephone">
      <xsd:list itemType="xsd:unsignedByte" />
    </xsd:simpleType>
```

Un élément conforme à cette définition serait <tel>01 23 45 67 89 >/tel>
Les types listes de base sont ; NMTOKENS, ENTITIES et IDREFS

- **Le type union**

On veut un ensemble de plusieurs types possible.

```
Ex. <xsd:simpleType name="Telephone">
      <xsd:union memberTypes="xsd:string xsd:unsignedByte" />
    </xsd:simpleType>
```

Dans cet ex. on peut avoir soit le n° soit le nom

```
<telephone>18</telephone>
<telephone>Pompiers</telephone>
```

- **Le type composé**

Le type dit complexe est l'équivalent d'une structure comportant plusieurs champs.

- ❖ On indique l'ordre dans lequel on doit avoir les déclarations. C'est une séquence.

```
<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="nom"                  type="xsd:string" />
    <xsd:element name="prénom"               type="xsd:string" />
    <xsd:element name="dateDeNaissance"      type="xsd:date" />
      <xsd:choice>
     <xsd:element name="adresse"             type="xsd:string" />
     <xsd:element name="adresseElectronique" type="xsd:string" />
   </xsd:choice>
    <xsd:element name="téléphone"            type="Telephone" />   <!—déjà vu -->
```

```
  </xsd:sequence>
</xsd:complexType>
```

On peut aussi vouloir indiquer soit l'adresse réelle soit l'adresse mail. Cela est fait avec la balise *choise*.

- On peut vouloir avoir des éléments **dans n'importe quel ordre**. Cela se fait avec la balise **all**.

```
<xsd:complexType>
  <xsd:all>
    <xsd:element name="nom" type="xsd:string" />
    <xsd:element name="prénom" type="xsd:string" />
    <xsd:element name="dateDeNaissance" type="xsd:date" />
  </xsd:all>
</xsd:complexType>
```

- Les occurrences

caractère	min	max
*	0	unbounded
+	1	unbounded
?	0	1
impossible	n	m < n

- **Les facettes**

On indique une limite au champ

Ex1.
```
<xsd:simpleType name="monEntier">
  <xsd:restriction base="nonNegativeInteger">
    <xsd:maxExclusive value="100" />
  </xsd:restriction>
</xsd:simpleType>
```

Ex2.
```
<xsd:attribute name="jour" type="typeJourSemaine" use="required" />
<xsd:simpleType name="typeJourSemaine">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="lundi" />
      Tous les autres jours de la semaine
  </xsd:restriction>
</xsd:simpleType>
```

Ex3.
```
<xsd:simpleType name="typeMotLangueFrancaise">
  <xsd:restriction base="xsd:string">
    <xsd:length value="21" />
  </xsd:restriction>
</xsd:simpleType>
```

Ex4.
```
Un ISBN est un numéro international pour une publication il a 10 ou
13 chiffres
   <xsd:simpleType name="typeISBN">
     <xsd:restriction base="xsd:string">
       <xsd:pattern value="[0-9]{10 | 13}" />
     </xsd:restriction>
   </xsd:simpleType>
```

- **Les inclusions**

Le fichier de schéma deviens très vite énorme et peux lisible. On utilise les inclusions pour définir un schéma général à partir de morceaux de schéma.

```
<xsd:include schemaLocation="http://www.monsite.org/schemas/biblio.xsd" />
```

Le XML et CSS

Ex. de fichier XML	<?xml version="1.0" encoding="ISO-8859-1"?> <?xml-stylesheet href="**test.css**" type="text/css"?> <racine> <enfant> <nom>Loïc</nom> <lien>garçon</lien> <date>07/11/83</date> <data>Le petit qui me dépasse d'une tête.</data> </enfant> <enfant> <nom>Marine</nom> <lien>fille</lien> <date>20/12/85</date> <data>La petite fille chérie à son papa.</data> </enfant> </racine>
Fichier CSS associé	<style type="text/css"> racine , enfant {} nom { display: block; width: 250px; font-size: 16pt ; font-family: arial ; font-weight: bold; background-color: teal; color: white; padding-left: 10px; } lien { display: block; font-size: 12pt; padding-left: 10px; } date { display: block; font-size: 12pt; color: red ; font-weight: bold; padding-left: 10px; } data { display: block; 12 font-size: 11pt ; font-style: italic; font-family: arial ; padding-left: 10px; } </style>

On voit très bien le lien sur le fichier CSS qui est effectué dans l'en-tête XML <?xml-stylesheet href="**test.css**" type="text/css"?>

Le XML et XSL

XSL sert à mettre en page le XML. C'est aussi un puissant manipulateur d'éléments de XML.

Ex. de fichier XML	<?xml version="1.0" encoding="ISO-8859-1" ?> <?xml-stylesheet type="text/xsl" href="test7.xsl"?> <personnes> <personne nom="Dupond" service="Achats" /> <personne service="Achats" nom="Durand" /> <personne nom="Dupuis" service="Courrier" /> </personnes>
Fichier CSS associé	<?xml version="1.0" ?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:output method="html" encoding="ISO-8859-1" /> <xsl:template match="/"> <html> <head> <title> Liste des personnes </title> </head> <body> Liste des personnes <blockquote> <xsl:apply-templates select="//personne" /> </blockquote> </body> </html> </xsl:template> <xsl:template match="personne"> <xsl:value-of select="@nom" /> </xsl:template>

Cet ex. est assez complet car il prend des champs avec des attributs c'est pourquoi on a des syntaxes comme "//personne" et "@nom".

Voir tous les ex. qui sont dans les fichiers test/xml+xsl

Voir : http://tecfa.unige.ch/guides/tie/html/xml-xslt/xml-xslt-4.html Pour le tableau des valeurs.

Les éléments de xsl sont listés dans : http://www.w3schools.com/XSL/xsl_w3celementref.asp

Vous avez aussi un semblant de langage avec xsl.

Pour cela voir : http://www.haypocalc.com/wiki/Programmer_avec_XSLT