**Développer des applications XML**

Table des matières

[2. Le langage XML 1](#_Toc465948189)

[a. Definition 1](#_Toc465948190)

[b. Concevoir un document XML 1](#_Toc465948191)

[3. DTD Document Type Definition 3](#_Toc465948192)

[c. definition 4](#_Toc465948193)

[d. Structure d’une DTD: 5](#_Toc465948194)

[4. schéma XSD (XML Schema Definition). 8](#_Toc465948195)

[5. Transformation d’un document XML(XSL) 16](#_Toc465948196)

[e. Transformer un document XML en page HTML 17](#_Toc465948197)

# Le langage XML

## Definition

(eXtensible Markup Language) langage de balisage pour la description de documents structurés

Rôle fondamental : l'échange de données

utilisable avec n’importe quelle plateforme libre ou commerciale (Linux, Windows, Mac OS, Symbian, Windows CE…) ;ou bien n importe quelle langage java c# …

## Concevoir un document XML

Un document XML doit commencer par une déclaration, qui indique l’utilisation du langage XML (version n°1.0) et le jeu de caractère latin (ISO-8859-1) que nous utilisons en typographie Française (caractères avec des accents par exemple).

<?xml version="1.0" encoding="ISO-8859-1"? >

La description du contenu se fait grâce à des balises. Les balises encadrent le contenu de ce que l’on veut dire. Elles s’écrivent de cette manière : <balise>contenu</balise>

<?xml version="1.0" encoding="ISO-8859-1"? >

<titre>Technicien spécialisé</titre>

Les balises encadrent ce que l’on appelle des éléments. Dans l’exemple l’élément est un titre.

Un document XML est composé d'un seul élément racine

<?xml version="1.0" encoding="iso-8859-1" ?>

<article> imprimante </article>

<article> PC </article>

Invalide, il faut la corriger par l’ajout de la racine:

<?xml version="1.0" encoding="iso-8859-1" ?>

<stock>

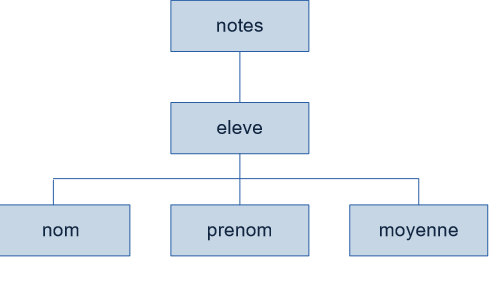
<article> imprimante </article>

<article> PC </article>

</stock>

* **Spécifications** du langage xml:
  + Un document doit commencer par une déclaration XML
  + Toutes les balises avec un contenu doivent être fermées
  + Toutes les balises sans contenu doivent se terminer par les caractères **/>**
  + Le document doit contenir un et un seul élément racine
  + Les balises ne doivent pas se chevaucher
  + Les valeurs d’attributs doivent être entre guillemets

Exemple :



<?xml version="1.0" encoding="ISO-8859-1"?>

<notes>

<eleve>

<nom>ROBIN</nom>

<prenom>Marc</prenom>

<moyenne>12,5</moyenne>

</eleve>

<eleve>

<nom>GRUNG</nom>

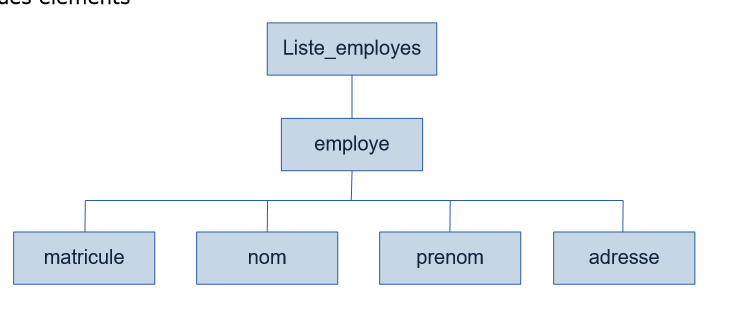
<prenom>Stéphanie</prenom>

<moyenne>11,3</moyenne>

</eleve>

</notes>

Exercices :



# DTD Document Type Definition

## definition

* Pour créer un document xml **valide**, il faut qu’il soit accompagné par son lot de validation : **DTD**
* Les DTD (*Document Type Definition*) sont très simples à manipuler, et on en trouve deux types:
  + ***Internes***: incorporées au document XML
  + ***Externes***: créées dans un fichier à part, et qui comptent encore deux autres types:
    - *Privées:* accessibles uniquement en local (sur la machine de développement)
    - *Publiques:* disponibles pour tout le monde (grâce à un URI)
* Exemple de déclarations pour une DTD ***interne*** :

<!DOCTYPE biblio[  
<!ELEMENT **biblio** (livre)\*>  
<!ELEMENT **livre** (titre, auteur, nb\_pages)>  
<!ATTLIST **livre** type (roman | nouvelles | poemes |théâtre) #IMPLIED  lang CDATA "fr"  >  
<!ELEMENT **titre** (#PCDATA)>  
<!ELEMENT **auteur** (#PCDATA)>  
<!ELEMENT **nb\_pages** (#PCDATA)>  
]>

* Exemple d’une DTD ***externe*** de type ***privé***:

<!DOCTYPE biblio SYSTEM "bibliographie.dtd">

* Le fichier associé est le suivant :

<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT **biblio** (livre\*)>

<!ELEMENT **livre** (titre, auteur, nb\_pages)>

<!ATTLIST **livre** type (roman|nouvelles|poemes|théâtre) #IMPLIED lang CDATA "fr">

<!ELEMENT **titre** (#PCDATA)>

<!ELEMENT **auteur** (#PCDATA)>

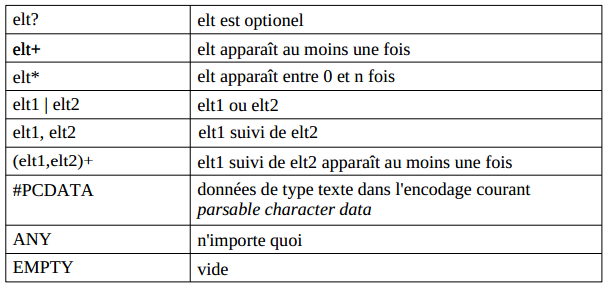
<!ELEMENT **nb\_pages** (#PCDATA)>

## Structure d’une DTD:

* **Définition d'éléments :**

**<!ELEMENT** nom\_elt (contenu\_elt) **>**

* Le contenu d’un élément peut être:



* **Définition d'attributs :**

**<!ATTLIST** elt nom\_attr type\_attr valeur\_défaut**>**

* Le type d’attribut peut être:
  + ***CDATA*** : n’importe quelle chaine de caractères

<!ATTLIST formateur nom **CDATA**>

* + ***ID*** : l’attribut doit être unique dans le document

<!ATTLIST formateur matricule **ID** #REQUIERED>

* + ***Énumération***: (pas de mot clé en xml) le modèle du contenu est plutôt une liste de toutes les valeurs possibles pour l’attribut, séparées par des barres verticales

<!ATTLIST formateur categorie **(A|B|C|D|E)**>

* Un attribut peut être:
  + #IMPLIED : optionnel (*avec ou sans valeur par défaut)*

<!ATTLIST personne att1 **"tri"**>

<!ATTLIST personne att1 CDATA **#IMPLIED**>

* + #REQUIRED : obligatoire

<!ATTLIST personne att1 CDATA **#REQUIRED**>

* + #FIXED : fixe (*il faut mettre la valeur par défaut*)

<!ATTLIST personne att1 CDATA **#FIXED "tri"**>

* La déclaration peut comporter **plusieurs** attributs:

<!ATTLIST personne att1 CDATA # **REQUIRED** att2 CDATA #IMPLIED>

<employes>

<employe id="1" departement="informatique">

<matricule>HUYTT</matricule>

<nom>KOUTBI</nom>

<prenom>Ahmed</prenom>

<telephone>212 06 07 65 34 54</telephone>

<email></email>

<travaux>

<travail></travail>

<travail></travail>

</travaux>

</employe>

<employe id="2" departement="admin">

<matricule>FHGFH</matricule>

<nom>KOUTBI</nom>

<prenom>Ahmed</prenom>

<telephone>212 06 07 65 34 54</telephone>

<email></email>

<travaux>

<travail>creation d’une application XML</travail>

</travaux>

</employe>

</employes>

**Exemple**

**Schema DTD**

<?xml version="1.0" encoding="ISO-8859-1"?>

< !element employes

<!ELEMENT employe (matricule+,nom+,prenom+,telephone+,mail?,travaux+)>

<!ATTLIST employe

id ID #REQUIRED

departement (informatique|admin) "informatique"

>

<!ELEMENT matricule (#PCDATA) >

<!ELEMENT nom (#PCDATA) >

<!ELEMENT prenom (#PCDATA) >

<!ELEMENT telephone (#PCDATA) >

<!ELEMENT mail (EMPTY) >

<!ELEMENT travaux (travail\*)>

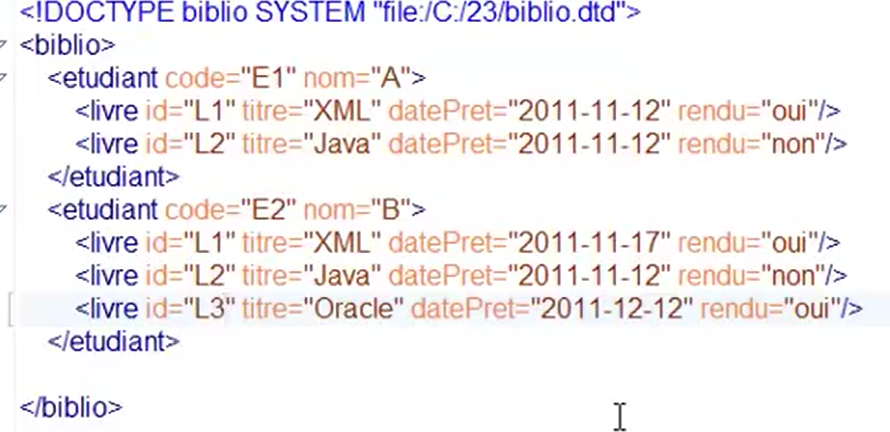
<!ELEMENT travail (#PCDATA)>

]>

<employes>

<employe id="1" departement="informatique">….

**Exe**

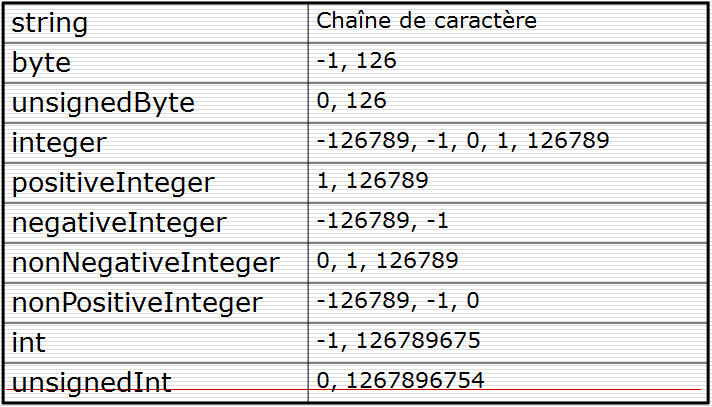
****

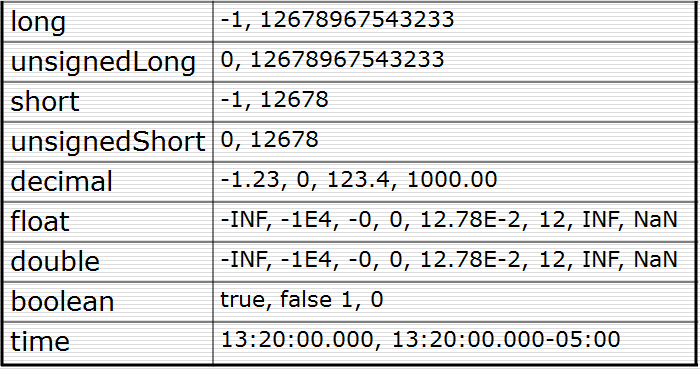
# schéma XSD (XML Schema Definition).

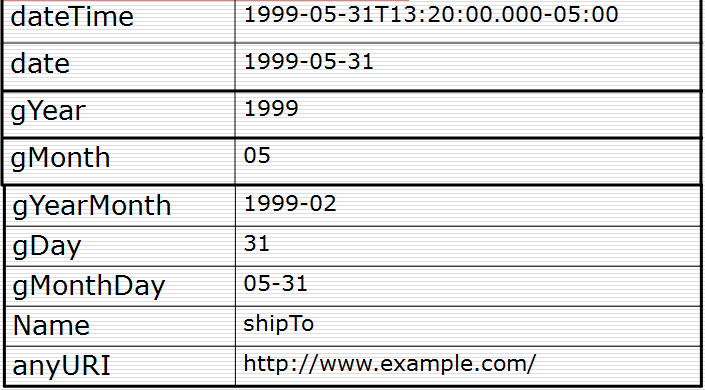
Les documents XML peuvent être normés grâce à un schéma XSD (XML Schema Definition). Cela sert à donner des règles de structure pour un document XML. Lorsqu’un document XML est échangé entre deux partenaires, le fait de joindre un schéma permet à chacun de compléter le document de la même manière et avec la même rigueur. Sans cela l’échange pourrait être compromis ou n’aurait peutêtre pas de sens.

Le schéma va permettre, dans un premier temps, d’indiquer l’arborescence du document

Dans un second temps, le schéma permet d’indiquer le type des éléments du document :







Dans un troisième temps, le schéma permet de préciser le nombre d’occurrences (nombre de fois ou sont répétés les éléments).

minOccurs= "0" ou "1"

maxOccurs= "1" ou "unbounded" (illimité)

**XSD : type simple**

* Un élément simple peut avoir une valeur par défaut ou valeur fixée:

<xsd:element name="color" type ="xsd:string" **default ="red"**> </xsd:element>

<xsd:element name="color" type ="xsd:string" **fixed="red"**> </xsd:element>

* Les types **listes** sont des suites de types simples:

<xs:element name="intvalues" type="valuelist"/>

<**xs:simpleType** name="valuelist">  
  <**xs:list** itemType="xs:integer"/>  
</**xs:simpleType**>

* Un élément conforme à ce schéma serai de la forme:

<intvalues>100 34 56 -23 1567</intvalues>

* + Valeurs comprises entre 0 et 99
* <xsd:simpleType name="monEntier">  
    <**xsd:restriction** base="nonNegativeInteger">  
      <**xsd:maxExclusive** value="100" />  
    </**xsd:restriction**>  
  </xsd:simpleType>
  + Limiter les valeurs à un certain nombre d’éléments
* <xsd:simpleType name="jourSemaine">  
    <**xsd:restriction** base="xsd:string">  
      <**xsd:enumeration** value="lundi" />  
      <**xsd:enumeration** value="mardi" />  
      <**xsd:enumeration** value="mercredi" />  
      <**xsd:enumeration** value="jeudi" />  
      <**xsd:enumeration** value="vendredi" />  
      <**xsd:enumeration** value="samedi" />  
      <**xsd:enumeration** value="dimanche" />  
    </**xsd:restriction**>  
  </xsd:simpleType>
  + Limiter la longueur d'une chaîne
* <xsd:simpleType name="typeMotLangueFrancaise">  
    <**xsd:restriction** base="xsd:string">  
   <**xsd:length** value="21" />  
    </**xsd:restriction**>  
  </xsd:simpleType>
  + Utiliser des expressions régulières
* <xsd:simpleType name="typeISBN">  
    <**xsd:restriction** base="xsd:string">  
      <**xsd:pattern** value="[0-9]{10}" />  
    </**xsd:restriction**>  
  </xsd:simpleType>
* **XSD : type complexe**
  + **Les séquences d’éléments** : déclarer un élément comme pouvant contenir une suite de sous-éléments
* **<xsd:complexType name="personne">  
    <xsd:sequence>**  
      <xsd:element name="nom" type="xsd:string" />  
      <xsd:element name="prénom" type="xsd:string" />  
      <xsd:element name="dateDeNaissance" type="xsd:date" />  
      <xsd:element name="adr" type="xsd:string" />  
      <xsd:element name="adrElectronique" type="xsd:string" />  
      <xsd:element name="téléph" type="numéroDeTél" />  
    **</xsd:sequence>  
  </xsd:complexType>**
* **XSD : attribut**

Un **attribut**, dans un schéma, se déclare avec la balise **<xsd:attribute>** et ne peut être que de **type simple**

**<xsd:attribute** name="nom" type="xsd:string" /**>**

* Tous les attributs sont optionnels par défaut, et pour le faire explicitement:

**<**xsd:attribute name="nom" type="xsd:string" **use="optional"**/**>**

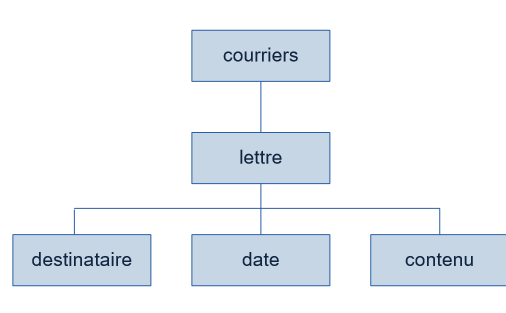
* Pour déclarer un attribut obligatoire:

**<**xsd:attribute name="nom" type="xsd:string" **use="required"**/**>**

**<poids unite=”kg”>56</poids>**

 <xsd:complexType name="typePoids">  
 **<xsd:simpleContent>**  
   **<xsd:extension** base="xsd:integer">  
      <xsd:attribute name="unite" type="xsd:string" />  
   **</xsd:extension>**  
  **</xsd:simpleContent>**  
</xsd:complexType>

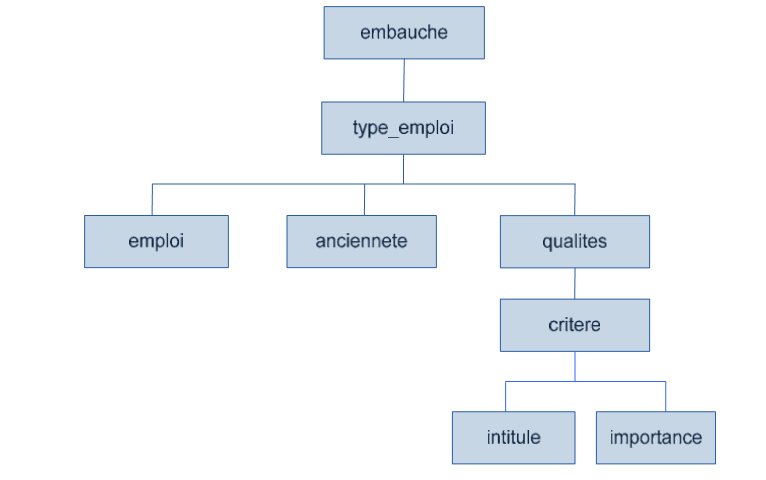
exemple



Solution



Exemple 2



**Exercice**

<employes>

<employe id="1" departement="informatique">

<matricule>HUYTT</matricule>

<nom>KOUTBI</nom>

<prenom>Ahmed</prenom>

<telephone>212 06 07 65 34 54</telephone>

<email></email>

<travaux>

<travail></travail>

<travail></travail>

</travaux>

</employe>

<employe id="2" departement="admin">

<matricule>FHGFH</matricule>

<nom>KOUTBI</nom>

<prenom>Ahmed</prenom>

<telephone>212 06 07 65 34 54</telephone>

<email></email>

<travaux>

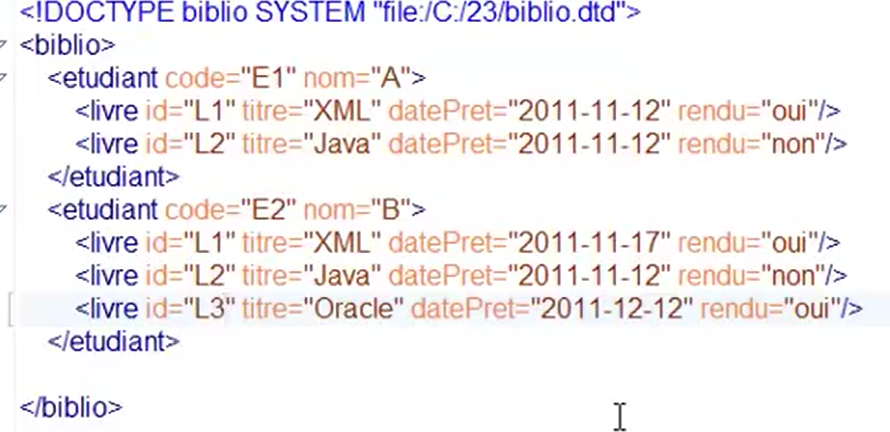
<travail>creation d’une application XML</travail>

</travaux>

</employe>

</employes>

**Exercice:**

****

# Transformation d’un document XML(XSL)

Vous aurez pu vous en rendre compte, un document XML, même accompagné d’un schéma n’est pas satisfaisant au niveau de la présentation. Il y a deux raisons à cela :

- c’est un format d’échange donc on ne peut pas mettre en forme le document sachant que l’on ne sait pas ce qu’il deviendra (sera t-il lu sur un ordinateur ou un PDA ? Avec quel logiciel va-ton l’utiliser ?…) ;

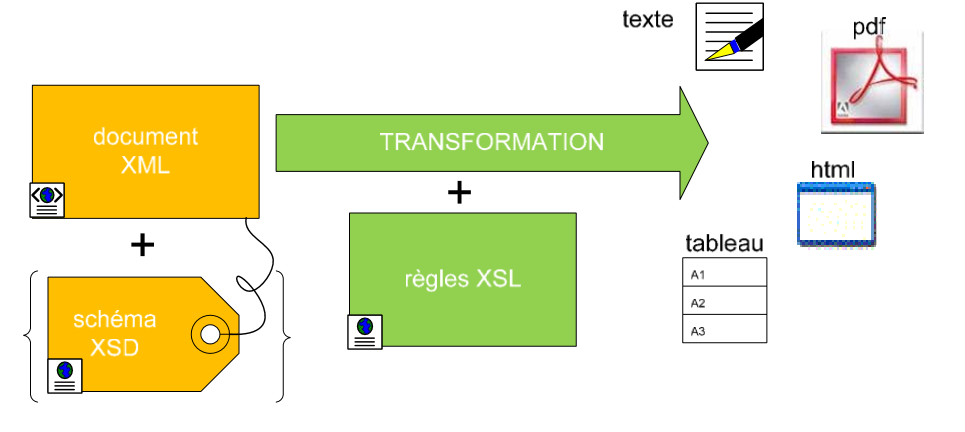
- seules les informations contenues comptent, la présentation n’est pas une information mais juste un moyen de représenter ces informations. On appelle cela séparation du contenu et de la mise en forme.

XML a la particularité d’être transformable. Transformable suivant la destination du document :

- Si on veut le lire avec un tableur, Open Office ou Excel à partir de la version 2003 Professionnel peuvent transformer les documents XML en feuille de tableur (avec mise ne forme des cellules, etc.) ;

- Si on veut le lire avec un traitement de texte, Open Office ou Word à partir de la version 2003 Professionnel peuvent transformer les documents XML (avec mise en forme du texte, titrage, etc.) ;

- Si on veut le transformer en page Web, en document PDF, l’’incorporer à une base de données, c’est possible également



## Transformer un document XML en page HTML

Un document XML peut subir une transformation grâce à XSLT (eXtended Stylesheet Language Transformations). Un fichier XSL étant un fichier XML, il doit respecter les normes de syntaxe de ce format

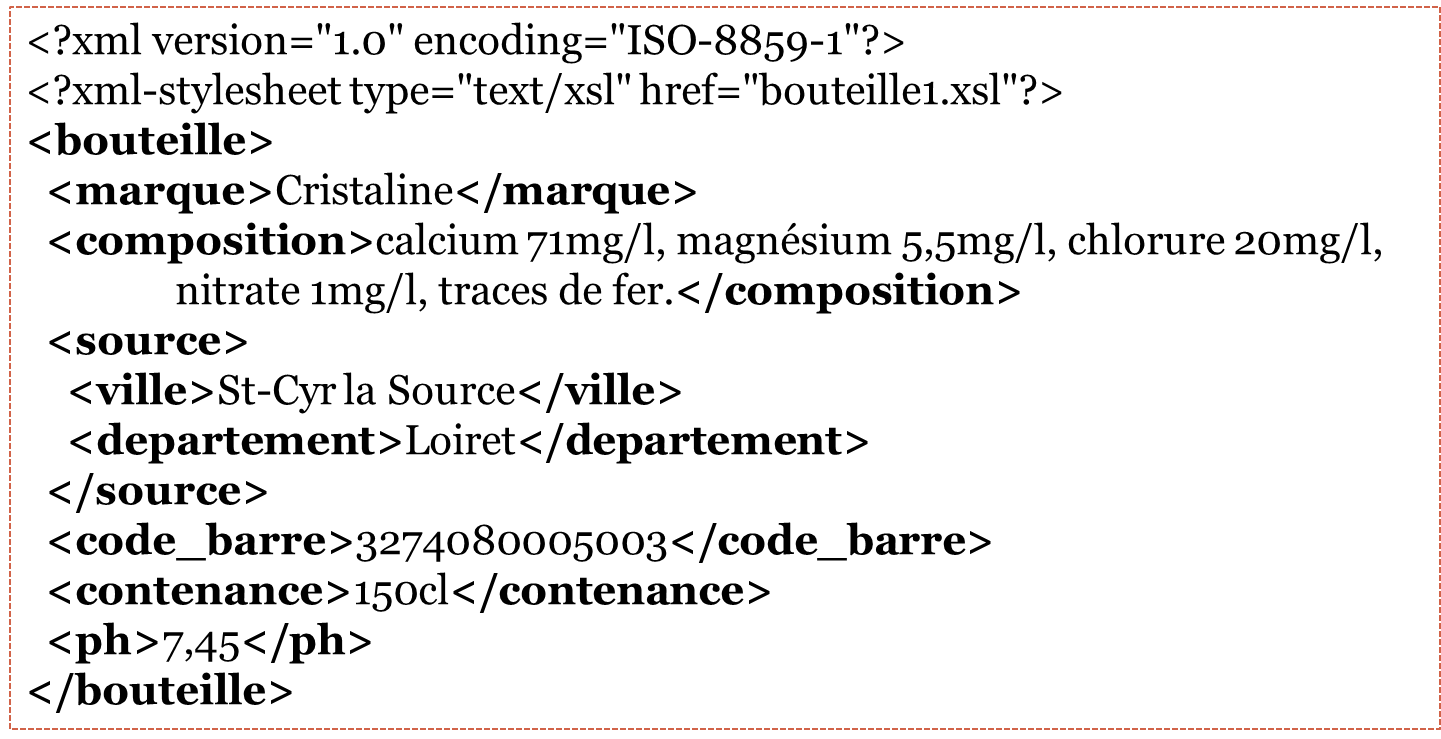
<?xml version="1.0" encoding="ISO-8859-1"?>  
**<xsl:stylesheet** version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

**<xsl:template match="/">**

...

**</xsl:stylesheet>**

**Exemple1:**



Solution

<?xml version="1.0" encoding="ISO-8859-1"?>  
**<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">**  
**<xsl:template match="/">**  
  **<html>**  
    **<head>**  
      **<title>**Exemple de sortie HTML**</title>**  
      **<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />**  
    **</head>**  
    **<body>**  
      **<h1>**Bouteille de marque **<xsl:value-of select="bouteille/marque" /></h1>**  
      **<h2>**Composition:**</h2>**  
      **<p><xsl:value-of select="bouteille/composition" /></p>**

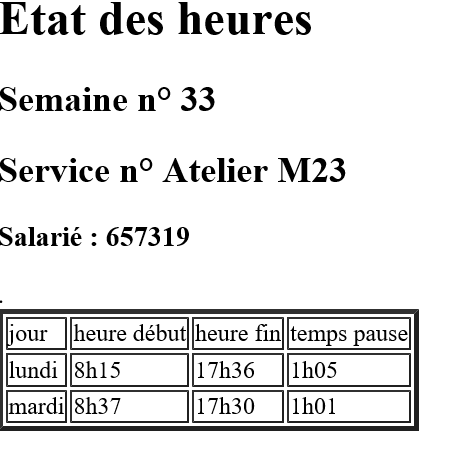
**<h2>**Lieu d'origine:**</h2>**  
      **<p>**Ville de **<b><xsl:value-of select="bouteille/source/ville" /></b>**, dans le département **<b><xsl:value-of select="bouteille/source/departement" /></b></p>**

**<h2>**Autres informations**</h2>**  
      **<ul>**  
        **<li>**Contenance: **<xsl:value-of select="bouteille/contenance" /></li>**  
        **<li>**pH: **<xsl:value-of select="bouteille/ph" /></li>**  
      **</ul>**  
    **</body>**  
  **</html>**  
**</xsl:template>**  
**</xsl:stylesheet>**

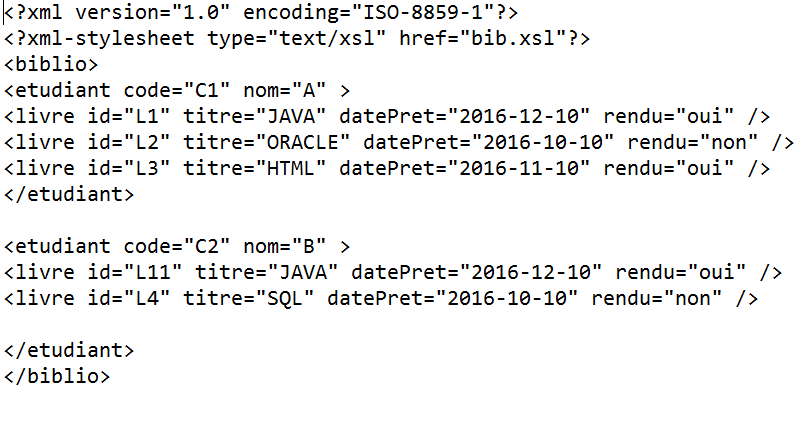
* Il suffit pour cela d'introduire dans la feuille de style l’élément **<xsl:for-each>**:

**Exemple2:**

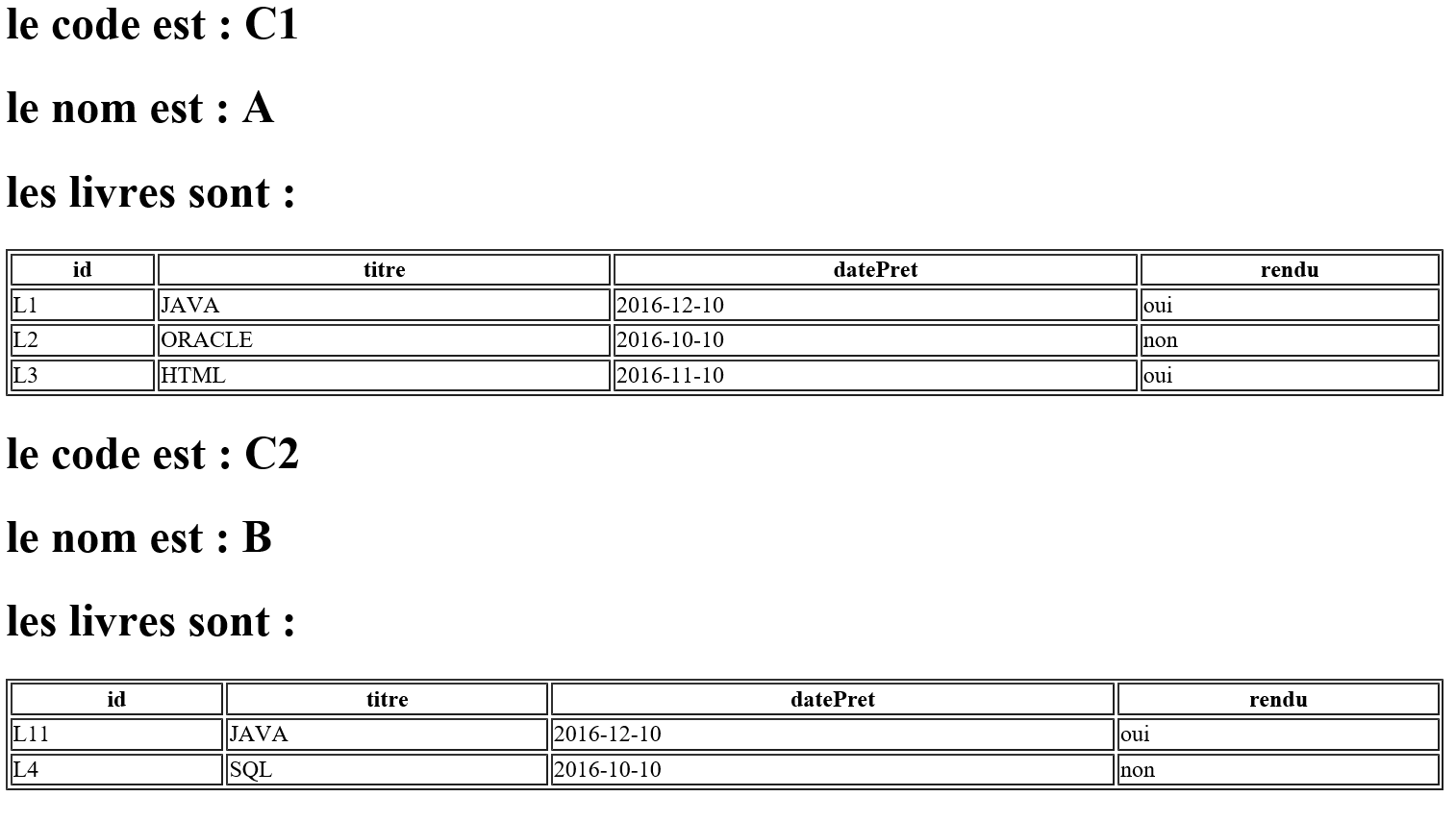




**Exercice:**

****

Solution



Le premier livre

<xsl:for-each select="livre[1]" >

Les livres rendus

<xsl:for-each select="livre[@rendu='non']" >

Le nombre de livres :

<h1>le nombres des livres est :

<xsl:value-of select="count(livre)" /></h1>

le nombres des livres non rendus est

<xsl:value-of select="count(livre[@rendu='non'])" />

Soit la structure arborescente d’un livre :

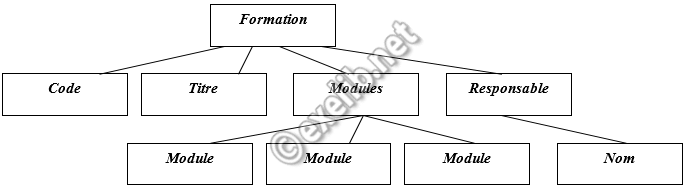


Schéma XML

1. Structurer les informations schématisées dans la figure ci-dessus sous forme d’un document XML **(1,5 pt)**
2. Etablir la DTD du document XML, sachant que l’élément **Responsable** est optionnel et qu’il contient nécessairement un attribut **Date**.**(3 pts)**
3. Ecrire les instructions XSL permettant d’afficher la liste des modules dans des listes numérotées. **(1,5 pt)**