le cnam

Séance d'Exercices Dirigés

XML et XSLT

Au sommaire de cet ED :

L'objectif de cette séance d'exercices dirigés est d'introduire de manière ludique le fonctionnement des processus XML. Le but est de montrer de façon simple, concise et précise aux élèves, les principes de fonctionnement des différents composants XML dans une architecture Web.

1 1ère question

Rappeler les définitions des termes suivants : XML, DTD, XSD, XSL, document XML, éléments, élément vide, attribut d'un élément, document bien formé, document valide. Pourquoi un document XML a une structure d'arbre ?

1.1 Rappel des Généralités

XML *eXtensible Markup Language* (langage extensible de balisage) groupe de travail XML formé par le W3C en 1996 sous l'égide de Jon Bosak de Sun Microsystems (avec des spécialistes du SGML Working Group) :

- ✓ format public,
- ✓ méta-langage = un langage qui permet de définir d'autres langages,
- ✓ sous-ensemble de SGML, but = rendre SGML utilisable sur le Web,
- ✓ permet de concevoir votre langage de balisage personnalisé pour un ensemble de classes de documents (vous pouvez inventer des balises pour répondre à un besoin spécifique : un langage pour votre bibliothèque par exemple),
- ✓ un langage défini par XML est appelé *vocabulaire XML* ou *application XML*,
- ✓ le langage de balisage créé est généralement défini par une définition de type de document (DTD pour XML1) ou un schéma (XSD pour XML2). Ces grammaires définissent les éléments qui composeront le vocabulaire, les attributs de tous les éléments, ainsi que les entités.

1.2 Caractéristiques de XML

XML est un sous-ensemble de SGML, dont les caractéristiques inutiles pour la publication sur le Web ont été supprimées :

- ✓ il est destiné à décrire le contenu du document, pas son affichage (les feuilles de style CSS et XSL gèrent l'affichage),
- ✓ il est flexible, on peut définir ses balises, et les utiliser dans un ou plusieurs documents,
- ✓ le document ne sera affiché que s'il est bien formé et valide (s'il suit une DTD ou un schéma),
- ✓ il est lisible pour l'humain (l'information contenue sera toujours accessible, contrairement aux fichiers de certains logiciels, par exemple, il est impossible de visualiser du RTF sans un logiciel qui connaisse ce format),
- ✓ le document XML est un texte qui n'est pas destiné à être lu par l'humain (mais le fait que ce soit un texte permet aux experts d'utiliser un éditeur de texte pour corriger le fichier).

1.3 Technologies liées à XML

Autour de la spécification XML, il existe une famille de technologies :

- ✓ CSS, permet de définir une feuille de style pour XML.
- ✓ XSL, langage évolué pour la définition de feuilles de style.
- ✓ XSD, grammaire des documents XML, au format XML.
- ✓ Xlink pour ajouter des liens hypertextes à un fichier XML.
- ✓ XPointer pour pointer sur des parties d'un document XML, un XPointer pointe sur des éléments de données au sein d'un fichier XML.
- ✓ DOM *Document Object Model* pour manipuler des fichiers XML (et HTML) à partir d'un langage de programmation.
- ✓ namespaces (domaines de noms) pour distinguer les noms utilisés dans les documents XML.
- ✓ XForm pour les formulaires.

1.4 Contenu d'un document XML

Un document XML est composé d'**éléments**, de blocs qui représentent la structure logique du document. Le document contient à la fois l'information et des méta-informations (information sur l'information). Ces éléments peuvent être :

- ✓ non vides ; ils commencent par une balise ouvrante, peuvent contenir du texte et d'autres éléments et se terminent par une balise fermante.
  <titre>Mort sur le Nil</titre>
- ✓ vides : ils ne contiennent rien, aucun texte, aucun élément. L'élément IMG de HTML est un élément vide. En XML ils s'écrivent avec un / à la fin de la balise ouvrante ou sous la forme d'une paire de balises vide :
  <hr/> ou encore <hr></hr>

Chaque élément présente des caractéristiques appelées **attributs** :
<titre type="policier">Mort sur le Nil</titre>

Ce sont les **DTD** *Document Type Definition* ou les schémas (*XSD*) qui définissent les éléments et les règles d'utilisation (noms des éléments, attributs possibles pour un élément, imbrications). Cependant des documents XML peuvent ne pas avoir de schémas ou de DTD. Si un document a un schéma associé et qu'il se conforme à celui-ci, il est dit **valide**. S'il n'en a pas et qu'il suit les règles définies par XML (par exemple : ses éléments sont correctement imbriqués) il est **bien formé**.

Le document ne contient aucune information concernant l'affichage, c'est sa feuille de style qui définira la présentation sur un média.

1.5 Document XML bien formé

Un document XML est bien formé (l'analyseur XML peut construire son arborescence) si :

- ✓ il contient une déclaration XML ;
- ✓ il contient un ou plusieurs éléments ;
- ✓ il contient un élément racine encapsulant tous les autres éléments et leurs attributs (ex <HTML> ... </HTML>) ;
- ✓ les éléments non vides ont une balise de début et de fin ;
- ✓ les éléments non vides sont correctement imbriqués (<P> <EM> ... </EM> </P>) ;
- ✓ les éléments vides ont un / à la fin de la balise avant le > ;
- ✓ les noms des balises ouvrantes et fermantes correspondent ;
- ✓ un nom d'attribut apparaît uniquement dans la balise ouvrante et une seule fois dans cette balise ;
- ✓ les valeurs des attributs sont entre guillemets ou apostrophes ;
- ✓ la valeur des attributs n'appelle pas d'entités externes directement ou indirectement ;
- ✓ les caractères réservés sont remplacés par des références d'entités (par ex. < pour <) ;

- ✓ toutes les références à des entités non binaires doivent commencer par & et finir par ;
- ✓ s'il n'y a pas de DTD, les seules entités utilisées sont celles réservées de XML & < > ' " ;
- ✓ s'il y a une DTD toutes les entités non réservées utilisées sont déclarées dans la DTD.

1.6 Document XML valide

Un document est valide s'il :

- ✓ est bien formé,
- ✓ fait référence a une grammaire (schéma XSD ou DTD),
- ✓ se conforme à cette grammaire (schéma XSD ou DTD).

1.7 Structure d'un document XML

Un document XML comporte des éléments avec ou sans attributs qui fournissent des méta-informations sur l'information ou sur le contenu du document. Un document XML comporte :

- ✓ un prologue qui contient toutes les informations autres que les données ou les éléments,
- ✓ l'arbre des éléments avec un élément racine,
- ✓ éventuellement des commentaires.

1.8 Représentation du document sous forme d'arbre des éléments

Tout document a une structure sous forme d'arbre, prenons comme exemple ce fichier HTML :

```
<HTML>
    <HEAD>
        <TITLE>essai</TITLE>
    </HEAD>
   <BODY>
        <P>paragraphe <EM>important</EM> du document essai</P>
        <P>paragraphe normal du document essai</P>
    </BODY>
</HTML>
Balise racine HTML
|
|_________ HEAD
|           |
|           |____ TITLE
|
|
|________ BODY
            |
            |____ P
            |     |
            |     |____ EM
            |
            |____ P
```

Il y a des parents, des enfants, des frères. HTML est le parent des éléments HEAD et BODY qui sont des frères. EM est un enfant de P qui est un enfant de BODY. Le document a donc une structure logique. L'élément document, est l'élément racine qui contient tous les autres éléments et données du document (<HTML>...</HTML>).

2 2ème question

Commenter le document XML `biblio.xml` suivant :

```
<?xml version="1.0"?>

<!DOCTYPE bibliotheque
[
      <!ELEMENT bibliotheque (livre+)>
      <!ELEMENT livre (titre, auteur, ref)>
      <!ELEMENT titre (#PCDATA)>
      <!ELEMENT auteur (#PCDATA)>
      <!ELEMENT ref (#PCDATA)>
]
>

<bibliotheque>

<livre>
        <titre>N ou M</titre>
        <auteur>Agatha Christie</auteur>
        <ref>Policier-C-15</ref>
</livre>

<livre>
        <titre>Le chien des Baskerville</titre>
        <auteur>Sir Arthur Conan Doyle</auteur>
        <ref>Policier-D-3</ref>
</livre>

<livre>
        <titre>Dune</titre>
        <auteur>Franck Heckbert</auteur>
        <ref>Fiction-H-1</ref>
</livre>

</bibliotheque>
```

Indiquer comment écrire ce document en définissant sa grammaire associée de type schéma (xsd) dans un fichier externe `biblio.xsd` ?

2.1 Etudes des composants

Éléments :

Un élément non vide est constitué de trois parties, une balise ouvrante qui peut contenir des attributs, un contenu (des donnés et/ou d'autres éléments) et une balise fermante.

Les éléments vides ne contiennent ni texte, ni autres éléments, ils peuvent avoir des attributs.

Un nom d'élément doit commencer par une lettre ou un souligné, il peut comporter des chiffres, des lettres, des traits d'union, des points, double-points ou soulignés.

Il faut noter que les éléments sont sensibles à la casse, l'exemple ci-dessous est illégal :

`<titre>...</TITRE>`

Domaines de noms :

Une nom d'élément peut être divisé en deux parties : `domaine_de_nom:nom_element`. Par exemple, `xsl:template` indique que l'élément template fait partie de xsl.

L'utilisation des domaines de noms n'est pas obligatoire, mais celà permet d'éviter les collisions lorsqu'on fusionne des éléments de mêmes domaines d'activité provenant de différentes sources.

les espaces de noms sont déclarés dans le document avec l'attribut xmlns:id, l'url permet de donner un domaine de nom par défaut (il peut y avoir plusieurs attributs de domaines de noms dans l'élément)

`<element xmlns:prefixe=uri>`

La portée est limitée à l'élément (si on le place dans la racine = tout le document). Mais on peut rencontrer dans les éléments inclus une autre déclaration de domaine de noms avec un préfixe identique, il remplace alors le précédent.

le préfixe permet d'associer un nom à un domaine de noms (utilisé quand il y a plusieurs domaines de noms dans l'élément parent). `<pref:element>...</pref:element>`

Attributs :

`propriete = "valeur"` ou `propriete = 'valeur'` Les attributs peuvent être facultatifs ou obligatoires, ils donnent des informations supplémentaires sur les éléments. Ils apparaissent uniquement dans la balise ouvrante d'un élément.

Attributs réservés :

xml:lang : Sa valeur indique le langage de l'élément. Cette valeur est un code de langue ISO 639 (en minuscules) : `fr, en, it,...` suivi s'il y a des variantes pour la langue d'un tiret et d'un code de pays ISO 3166 (en majuscules). `<livre xml:lang="en-US">`

xml:space = "default | preserve" : Sa valeur indique si un espace blanc à l'intérieur d'un élément est significatif et ne doit pas être altéré par le processeur XML. Avec `default` le processeur XML est libre de faire ce qu'il veut avec les espaces. Si un élément doit se comporter comme le <pre> de HTML il faut utiliser `preserve`.

Entités Internes :

Appel d'une entité dans un document : `&nom_entite;`

Les caractères réservés de XML sont remplacés par des entités internes. Ces caractères sont les mêmes qu'en HTML : `& < > " '`. Les entités qui permettent de les représenter sont respectivement `& < > &quot; &apos;`

Tous les caractères peuvent être remplacés par une entité qui donne leur code &#code_car; (par ex. A pour le A).

Les entités peuvent appeler d'autres entités et peuvent provoquer leur inclusion dans le document XML.

Entités Externes :

Les entités externes ne sont pas contenues dans le document courant, le processeur XML ignore le contenu de l'entité et le transmet à l'application. Les entités non parsées peuvent être utilisées pour les fichiers images, les fichiers sons, les fichiers vidéo... Elles sont appelées comme valeur d'un attribut (comme en HTML on avait le chemin de l'image comme valeur de l'attribut src de l'élément img).

NB : les graphiques sont simplement des liens vers une image plutôt que vers un texte, ils peuvent donc être créés de n'importe quelle manière supportée par les spécifications XLink et XPointer.

Encodage des caractères :

Le xml prend mieux en charge les caractères accentués que le HTML où on utilise des appels d'entités (par exemple : é pour é). Les spécifications XML indiquent que tous les processeurs XML devront accepter les codages UTF-8 et UTF-16 de la norme ISO 10646 qui couvre la plupart des langages humains :

- ✓ ISO 646 : code ASCII, codage sur 7 bits (128 caractères) utilisé pour les langues non accentuées comme l'anglais (le 8eme bit de l'octet est un bit de contrôle).

- ✓ ISO 8859/1 : codage sur 8 bits (256 caractères, le 8eme bit code les accents) pour les langues européennes accentuées (les 128 premiers caractères sont les mêmes que ceux de l'ISO 646).
- ✓ ISO 10646 : permet de coder toutes les langues européennes et asiatiques. Grâce à des combinaisons d'adressage ou UTF (UCS Transformation Format) il permet d'utiliser un nombre variable de bits, selon les besoins d'une langue, UTF-8 (codage de 8 bits à 48 bits les 128 premiers car sont identiques au code ASCII, par contre il est incompatible avec ISO 8859-1 au delà du code 126), UTF-16 (jusqu'à 32 bits par combinaison).
- ✓ Unicode : 16 bits permet de coder l'arabe, le chinois, ... en fonction du degré de codage, on a des appelations différentes : UCS-4 code sur 4 octets (32 bits), et UCS-2 code sur 16 bits (c'est le standard unicode).

L'encodage se précise dans la déclaration xml :

```
<?xml version="1.0" encoding="UTF-16"?>
```

Tous les ordinateurs ne peuvent pas prendre en charge le jeu de caractères avancé, le dénominateur commun reste l'ASCII. Les autres caractères peuvent être appelés par un codage (numéro ISO 10646) par exemple &#38.

Commentaires :

```
<!-- commentaire -->
```

Les commentaires sont autorisés dans le document (après le prologue). Ils peuvent inclure n'importe quel type de données sauf le `--`, ils ne peuvent pas apparaître à l'intérieur des balises. Le processeur ne les transmet pas forcément à une application.

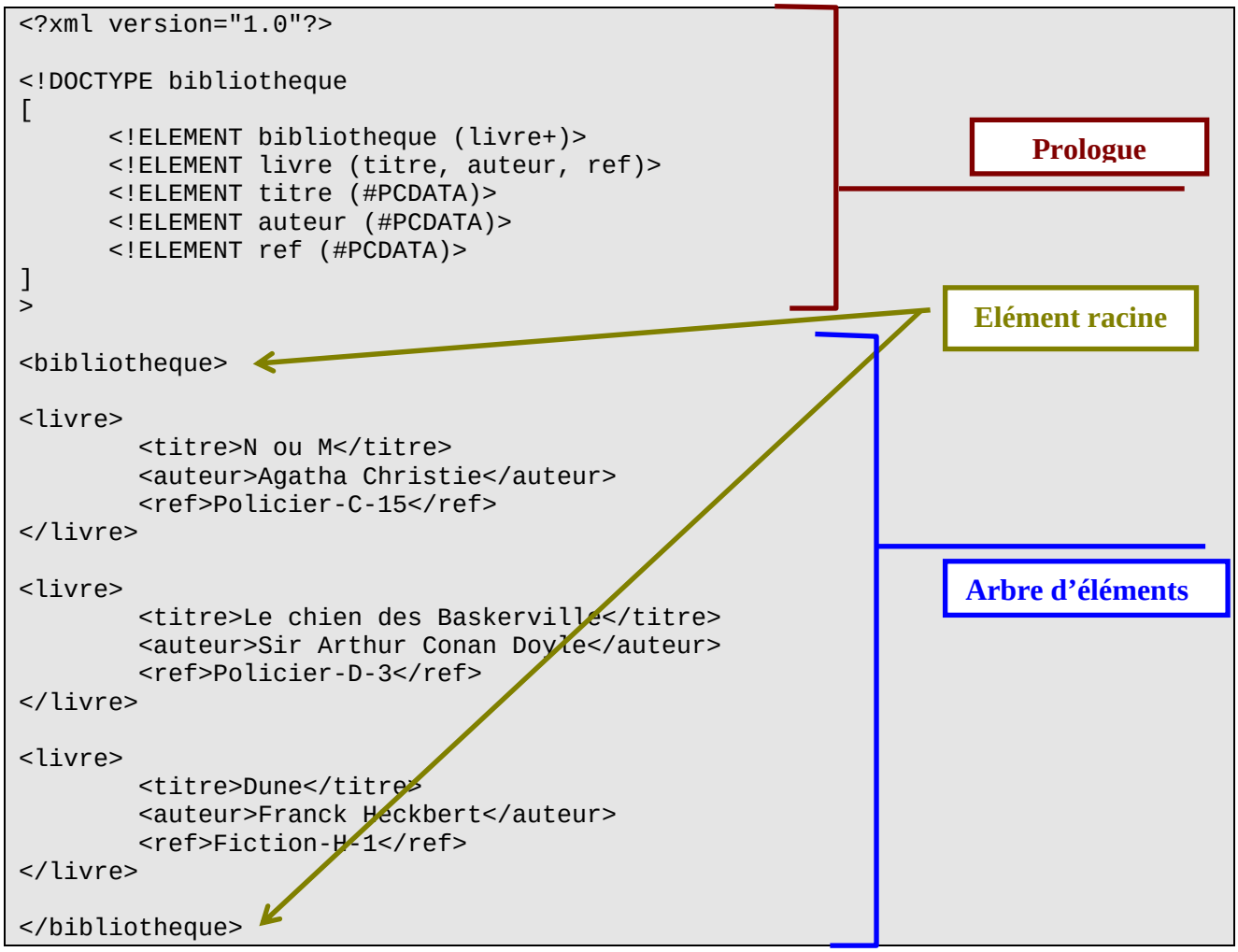
Sections CDATA :

<![CDATA[...]]>

Section de données que le processeur XML n'interprétera pas Ces sections permettent de passer des caractères réservés à une application (par exemple un signe mathématique <).

2.2 Exemple d'un document XML

Voici un document XML (nommé ***biblio.xml***) qui donne les titres de livres d'une bibliothèque, le nom de l'auteur et la référence du livre dans la biblio :




```
<?xml version="1.0"?>

<!DOCTYPE bibliotheque
[
      <!ELEMENT bibliotheque (livre+)>
      <!ELEMENT livre (titre, auteur, ref)>
      <!ELEMENT titre (#PCDATA)>
      <!ELEMENT auteur (#PCDATA)>
      <!ELEMENT ref (#PCDATA)>
]
>

<bibliotheque>

<livre>
        <titre>N ou M</titre>
        <auteur>Agatha Christie</auteur>
        <ref>Policier-C-15</ref>
</livre>

<livre>
        <titre>Le chien des Baskerville</titre>
        <auteur>Sir Arthur Conan Doyle</auteur>
        <ref>Policier-D-3</ref>
</livre>

<livre>
        <titre>Dune</titre>
        <auteur>Franck Heckbert</auteur>
        <ref>Fiction-H-1</ref>
</livre>

</bibliotheque>
```

2.3 Description du schéma *XSD*

XML - Schéma est une alternative XML aux **DTD** (les DTD sont des documents permettant de valider la conformité d'un document XML par rapport à sa définition). En réalité XML Schéma apparaît comme le successeur des DTD car il est par nature extensible et s'appuie sur XML.

Ainsi XML Schéma décrit (en XML) la structure d'un document XML c'est-à-dire :

- ✓ les éléments qui composent un document,
- ✓ les attributs,
- ✓ la hiérarchie entre les éléments,
- ✓ l'ordre des sous éléments,
- ✓ le nombre de sous éléments,
- ✓ les types des éléments et attributs,
- ✓ les valeurs par défaut, le format ou la restriction des valeurs d'un élément ou d'un attribut.

➔ On parle ainsi de XML Schéma Définition (**XSD**).

Eléments de base :

Les types simples les plus courants sont de types :

- ✓ xs:string
- ✓ xs:decimal
- ✓ xs:integer
- ✓ xs:boolean
- ✓ xs:date
- ✓ xs:time

➔ exemple : <xs:element name="Cours XML" type="xs:string"/>

➔ déclaration d'une **valeur par défaut** : <xs:element name="code_postal" type="xs:string" **default**="75000"/>

➔ déclaration d'une **valeur figée** : <xs:element name="universite" type="xs:string" **fixed**="cnam"/>

Eléments complexes :

la définition d'un élément complexe peut se faire directement au niveau de l'élément lui-même ou par référence au nom du type complexe (ce qui permet à plusieurs éléments de partager le même type complexe). La définition se fait alors par l'utilisation du tag **xs:complexType**.

Un type complexe peut enrichir un autre type complexe on non(tag <xs:**extension** base="type_de_base">).

Un type complexe peut aussi en restreindre un autre (exemple <xs:**restriction** base="xs:integer">).

Il est possible de mélanger du texte libre avec des tags (exemple : bonjour <prenom>olivier</prenom>) : <xs:complexType **mixed**="true">.

Attributs :

Seuls les éléments complexes peuvent avoir des attributs. La déclaration des attributs par défaut ou fixe est identique aux éléments :

➔ rendre un **attribut obligatoire** : <xs:attribute name="email" type="xs:string" use="**required**"/>

➔ rendre un **attribut facultatif** : <xs:attribute name="url" type="xs:string" use="**optional**"/>

Restrictions :

Il est possible d'apporter des restrictions sur les attributs ou éléments :

- ✓ **Plage de valeur** : <xs:**minInclusive** value="minimum"/> et <xs:**maxInclusive** value="maximum"/>,
- ✓ **Liste de valeur** : <xs:**enumeration** value="une_valeur"/>,
- ✓ **Conformité à un motif** : <xs:**pattern** value="[A-Z][A-Z][A-Z]"/> ou <xs:pattern value="([a-z])*"/>,
- ✓ **Traitement des espaces** : <xs:whiteSpace value="**preserve**"/> (les espaces sont laissés tels-que). Autres valeurs : **replace** (remplacer les LF,CR, TAB... par des espaces) ou **collapse** (remplacer les CR,LF... mais aussi supprimer les espaces avant/après et concaténer les successions d'espace en un seul),
- ✓ sur la **longueur** : <xs:**length** value="8"/> ou <xs:**minLength** value="5"/> et <xs:**maxLength** value="8"/>,
- ✓ il existe aussi des restrictions sur les **décimales** (**fractionDigits** et **totalDigits**).

Indicateurs :

Ils permettent de contrôler comment les éléments vont être utilisés suivants les types ci-dessous.

Indicateurs d'ordre :

- ✓ xs:**all** : les sous-éléments apparaissent dans n'importe quel ordre,
- ✓ xs:**choice** : indique qu'un seul des sous-éléments peut apparaître,
- ✓ xs:**sequence** : ordonne les sous-éléments : ils doivent apparaître dans un ordre précis.

Indicateurs d'occurrence (combien de fois un élément peut apparaître) :

- ✓ **maxOccurs** : nombre maximum (par défaut 1). Pour un nombre illimité, utiliser : maxOccurs="**unbounded**",
- ✓ **minOccurs** : nombre minimum.

Indicateurs de groupe :

- ✓ **group** : permet de regrouper logiquement des éléments,
- ✓ **attributeGroup** : permet de regrouper logiquement des attributs.

Extensions :

Quelques tags permettent de définir des extensions telles que :

- ✓ Le tag **any** permet de rajouter n'importe que élément à la suite de ceux qui sont précisément définis (exemple : <xs:any minOccurs="0"/>)
- ✓ Le tag **anyAttribute** permet d'ajouter des attributs non spécifiés dans le schéma.
- ✓ Le tag **substitutionGroup** permet de définir un schéma s'appliquant à un document XML dont les balises ne porteraient pas toujours le même nom (par exemple <nom /> et <name />). Il est aussi possible de bloquer la substitution.

Voici le fichier ***biblio.xsd*** correspondant à la DTD du document bibliothèque ; ***biblio.xml***.

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

<xsd:schema>
<xsd:element name="bibliotheque">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="livre" type="xsd:string" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="titre" type="xsd:string"/>
         <xsd:element name="auteur" type="xsd:string"/>
         <xsd:element name="ref" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="type" type="xsd:string" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
           <xsd:enumeration value="policier"/>
           <xsd:enumeration value="science-fiction"/>
           <xsd:enumeration value="aventure"/>
           <xsd:enumeration value="historique"/>
           <xsd:enumeration value="fantastique"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Formule d'appel du schéma XSD dans le document original XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--*************************************************************************-->
<?xml-stylesheet type="text/xml" href="biblio.xsl"?>
<!--<?xml-stylesheet type="text/css" href="biblio.css"?>-->
<!--*************************************************************************-->
```

3 3ème question

Rappeler ce qu'est une feuille de style et ce à quoi elle peut servir.
On considère la feuille de style `biblio.css` suivante :

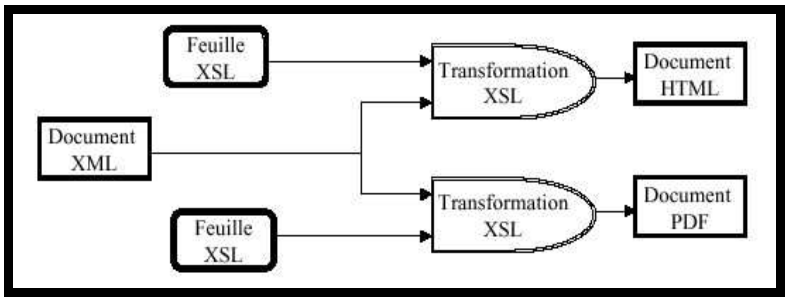
```
livre{
        display:block;
        margin-left:10pt;
        margin-bottom:5pt;
        font-size:12pt
}
titre{
        margin-right:10pt;
        color:blue;
}
auteur{
        margin-right:10pt;
}
ref{
        color:red;
}
```

Indiquer comment associer cette feuille de style au document XML `biblio.xml` ?
Que voit un utilisateur lorsqu'il charge dans un navigateur récent (NN 7.0 ou IE 6 par exemple), ce document XML associé à cette feuille de style ?

3.1 Description générale

L'utilisation d'une feuille de style est obligatoire en XML pour contrôler la mise en page du document, en effet un document XML ne contient que des informations sur la structure, aucune information relative à la mise en page n'apparaît dans le document. Il est possible d'utiliser pour présenter un document XML les feuilles de style CSS ou les feuilles de style XSL (*eXtensible StylesheetLanguage)*. Ces dernières sont issues de DSSSL (*Document Style Semantics and Specification Language*) la norme internationale ISO 10179 de feuilles de style pour les documents SGML.

XSL est beaucoup plus qu'un simple langage pour le formatage de documents XML (ce que pourrait très bien faire CSS - Cascading Style Sheets). Il permet de retraiter un document XML et ainsi de réarranger sa structure. En fait XSL permet de transformer un document XML en un autre document, souvent XML, mais pas forcément, par exemple en HTML, TeX, RTF, PostScript, etc. Le schéma ci-dessous représente le principe de fonctionnement.



3.2 Visualisation d'un document sans feuille de style

Lorsque aucune précision n'est donnée quant à l'affichage (pas de feuille de style), le navigateur affichera le contenu du document XML. Prenons le document ***biblio.xml*** qui décrit une bibliothèque, nous aurons l'affichage suivant :

```
<?xml version="1.0" ?>
- <bibliotheque>
  - <livre>
      <titre>N ou M</titre>
      <auteur>Agatha Christie</auteur>
      <ref>Policier-C-15</ref>
    </livre>
  - <livre>
      <titre>Le chien des Baskerville</titre>
      <auteur>Sir Arthur Conan Doyle</auteur>
      <ref>Policier-D-3</ref>
    </livre>
  - <livre>
      <titre>Dune</titre>
      <auteur>Franck Heckbert</auteur>
      <ref>Fiction-H-1</ref>
    </livre>
  </bibliotheque>
```

Il est possible de modifier l'affichage en appuyant sur les signes -

```
<?xml version="1.0" ?>
- <bibliotheque>
  + <livre>
  + <livre>
  - <livre>
      <titre>Dune</titre>
      <auteur>Franck Heckbert</auteur>
      <ref>Fiction-H-1</ref>
    </livre>
  </bibliotheque>
```

3.3 Utilisation de feuille de style CSS

Nous voulons à présent afficher ce document XML avec le titre en bleu et la référence en rouge, nous allons pour cela définir une feuille de style CSS dans le fichier ***biblio.css*** et nous allons l'appeler dans le document XML ***biblio.xml***.

Appel de la feuille de style dans le document :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="biblio.css"?>

<bibliotheque>

<livre>
        <titre>N ou M</titre>
        <auteur>Agatha Christie</auteur>
        <ref>Policier-C-15</ref>
```

```
</livre>
<livre>
        <titre>Le chien des Baskerville</titre>
        <auteur>Sir Arthur Conan Doyle</auteur>
        <ref>Policier-D-3</ref>
</livre>
<livre>
        <titre>Dune</titre>
        <auteur>Franck Heckbert</auteur>
        <ref>Fiction-H-1</ref>
</livre>

</bibliotheque>
```

Contenu de la feuille de style ***biblio.css*** :

```
livre{
        display:block;
        margin-left:10pt;
        margin-bottom:5pt;
        font-size:12pt
}
titre{
        margin-right:10pt;
        color:blue;
}
auteur{
        margin-right:10pt;
}
ref{
        color:red;
}
```

Affichage dans un navigateur Web :

4 4ème question

On veut construire une feuille de style avec la technique XSL. Qu'apporte XSL en plus de CSS ? Avec quelle syntaxe est écrite une feuille de style XSL ? Comment associer cette feuille de style XSL au document XML `biblio.xml` ?

4.1 Utilisation de feuille de style XSL

La feuille de style XSL est enregistrée dans un fichier externe et son nom comporte l'extension ".xsl". Dans le document XML, l'instruction de traitement suivante indique le type de la feuille de style et son emplacement :

```
<?xml-stylesheet type="text/xml" href="URL"?>
```

Prenons l'exemple de la bibliothèque, chaque livre a un titre, un auteur et une référence, le document XML ci-dessous appelle une feuille de style XSL qui est dans le fichier ***biblio.xsl*** dans le répertoire courant. Appel de la feuille de style dans le document XML :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="biblio.xsl"?>

<bibliotheque>

<livre>
        <titre>N ou M</titre>
        <auteur>Agatha Christie</auteur>
        <ref>Policier-C-15</ref>
</livre>
<livre>
        <titre>Le chien des Baskerville</titre>
        <auteur>Sir Arthur Conan Doyle</auteur>
        <ref>Policier-D-3</ref>
</livre>
<livre>
        <titre>Dune</titre>
        <auteur>Franck Heckbert</auteur>
        <ref>Fiction-H-1</ref>
</livre>

</bibliotheque>
```

4.2 Structure d'une feuille de style XSL

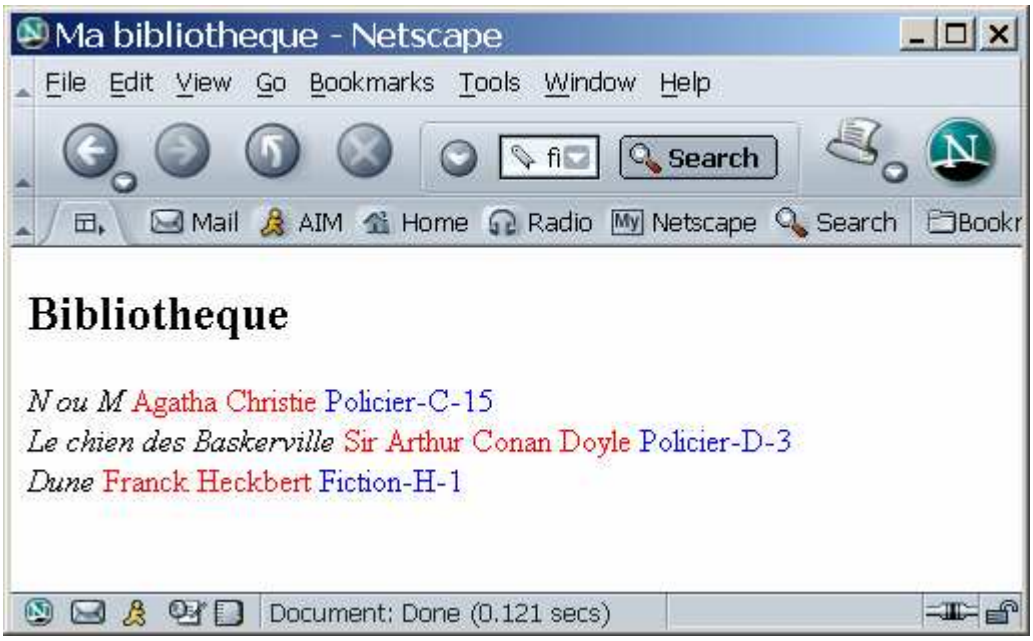
XSL est une application XML, une feuille de style XSL est donc un document XML. La feuille de style contient donc une déclaration XML et tous ses éléments sont placés dans l'élément racine. D'autre part, les éléments XSL sont préfixés par xsl: (XSL utilise les domaines de noms).

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    modèles
</xsl:stylesheet>
```

L'élément racine contient principalement des modèles (templates) pour l'affichage du document XML.

5 5ème question

Donnez la feuille XSL qui transforme le document `biblio.xml` en la page HTML suivante affichée ainsi dans un navigateur :



Une feuille de styles XSL contient un ou plusieurs modèles (templates), chaque modèle contient des informations sur l'affichage d'une branche des éléments du document. S'il n'y a qu'un seul modèle alors il s'applique sur la racine du document XML.

Prenons la feuille de style ***biblio.xsl*** qui applique un modèle unique au document XML bibliothèque ***biblio.xml*** :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>

<xsl:template match="/">
 <html>
   <head>
     <title>Ma bibliotheque</title>
   </head>
   <body>
         <H2>Bibliotheque</H2>

         <xsl:for-each select="bibliotheque/livre">
         <SPAN style="font-style:italic; padding-right:3pt">
             <xsl:value-of select="titre"/>
         </SPAN>
         <SPAN style="color:red; padding-right:3pt">
             <xsl:value-of select="auteur"/>
         </SPAN>
         <SPAN style="color:blue">
             <xsl:value-of select="ref"/>
         </SPAN>
         <br />
         </xsl:for-each>
    </body>
    </html>

</xsl:template>
</xsl:stylesheet>
```

Cette feuille de style comporte une déclaration XML, un élément racine `xsl:stylesheet` qui englobe tous les autres éléments et précise que les éléments préfixés par `xsl:` appartiennent au domaine de nom xsl.

Le modèle est appliqué à la branche spécifiée par l'attribut `match` de l'élément `template`. En CSS la valeur de l'attribut match correspondrait au sélecteur de la règle. Dans notre exemple c'est à la racine du document XML (ne pas confondre avec l'élément racine `bibliotheque` qui est un enfant de la racine du document) que le modèle s'applique. La transformation d'un document XML par une feuille de style XSL s'effectue donc par un modèle traitant un noeud donné. Chaque modèle est divisé en deux parties : un noeud cible indiqué par l'attribut `match` et une action sur le noeud :

```
<xsl:template match="noeud_cible">
    action
</xsl:template>
```

Un modèle contient deux types d'éléments :

- ✓ des éléments XML bien formés pour représenter les éléments html, c'est le cas de H2, SPAN et BR dans notre exemple.
- ✓ des éléments XSL, dans notre exemple le `xsl:value-of` qui permettent d'accéder au contenu des éléments du document XML, l'attribut `select` indique le nom de l'élément XML auquel on veut accéder. Ce nom est donné à partir de l'élément courant (ici c'est `bibliotheque/livre` donc on accède au titre du livre en donnant la valeur `titre` à l'attribut `select`)

L'élément `xsl:for-each select="bibliotheque/livre"` permet de parcourir tous les éléments qui portent le nom `livre` et qui sont des enfants de `bibliotheque` et donc d'afficher tous les éléments `livre` du document XML. Ce qui permet d'obtenir l'affichage :

5.1 Classement

Si nous souhaitons classer la bibliothèque par titre en respectant l'ordre alphabétique nous utiliserons : `<xsl:sort select="." order="ascending" />`, la feuille de style devient alors :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
 <html>
   <head>
     <title>Ma bibliotheque</title>
   </head>
   <body>
         <H2>Bibliotheque</H2>

         <xsl:for-each select="bibliotheque/livre">
         <xsl:sort select="." order="ascending" />
         <SPAN style="font-style:italic; padding-right:3pt">
```

```
            <xsl:value-of select="titre"/>
        </SPAN>
        <SPAN style="color:red; padding-right:3pt">
            <xsl:value-of select="auteur"/>
        </SPAN>
        <SPAN style="color:blue">
            <xsl:value-of select="ref"/>
        </SPAN>
        <br />
        </xsl:for-each>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

L'élément <xsl:sort select="."/> permet de définir l'ordre dans lequel les éléments seront affichés. C'est un sous-élément de xsl:apply-templates ou de xsl:for-each. L'attribut order = "ascending" | "descending" permet d'obtenir un ordre croissant ou décroissant.

Ce qui fournit l'affichage ci-dessous dans le navigateur :

Bibliotheque

Dune Franck Heckbert Fiction-H-1
Le chien des Baskerville Sir Arthur Conan Doyle Policier-D-3
N ou M Agatha Christie Policier-C-15

5.1.1 Filtrage

Ajoutons dans le document XML un attribut `type` qui correspond au type du livre :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="biblio.xsl"?>

<bibliotheque>
<livre type="policier">
        <titre>N ou M</titre>
        <auteur>Agatha Christie</auteur>
        <ref>Policier-C-15</ref>
</livre>
<livre type="policier">
        <titre>Le chien des Baskerville</titre>
        <auteur>Sir Arthur Conan Doyle</auteur>
        <ref>Policier-D-3</ref>
</livre>
<livre type="science-fiction">
        <titre>Dune</titre>
        <auteur>Franck Heckbert</auteur>
        <ref>Fiction-H-1</ref>
</livre>
</bibliotheque>
```

Nous pouvons par exemple décider de n'afficher que les livres dont le type est "policier", il suffit d'ajouter dans la feuille de style dans le tri le nom de l'attribut. Ce nom est précédé de @ pour indique que `type` n'est pas un élément mais un attribut (de la même manière il est possible d'accéder à la valeur d'un attribut dans un value-of en précédant l'attribut de @).

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
 <html>
   <head>
     <title>Ma bibliotheque</title>
   </head>
   <body>
         <H2>Bibliotheque</H2>

         <xsl:for-each select="bibliotheque/livre[@type='policier']">
         <SPAN style="font-style:italic; padding-right:3pt">
             <xsl:value-of select="titre"/>
         </SPAN>
         <SPAN style="color:red; padding-right:3pt">
             <xsl:value-of select="auteur"/>
         </SPAN>
         <SPAN style="color:blue">
             <xsl:value-of select="ref"/>
         </SPAN>
         <br />
         </xsl:for-each>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Ce qui fournit dans le navigateur l'affichage ci-dessous :

Bibliotheque

N ou M Agatha Christie Policier-C-15
Le chien des Baskerville Sir Arthur Conan Doyle Policier-D-3

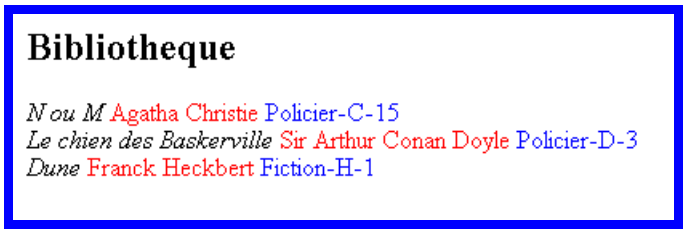
5.1.2 Modèle multiples

Pour afficher un élément XML comme `livre` dans notre exemple, il est possible d'utiliser `xsl:apply-templates` dans un modèle multiple (à la place du `xsl:for-each`).

Lorsqu'il y a plusieurs modèles il faut toujours qu'il y en ait un pour l'affichage de la racine du document (le /). Dans le premier modèle le `xsl:apply-template` indique que pour chaque élément livre enfant de bibliothèque il faut appliquer le deuxième modèle (celui pour lequel l'attribut `match` a pour valeur `livre`).

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>

<xsl:template match="/">

 <html>
   <head>
     <title>Ma bibliotheque</title>
   </head>
   <body>
         <H2>Bibliotheque</H2>
         <xsl:apply-templates select="bibliotheque/livre" />
    </body>
    </html>
</xsl:template>

<xsl:template match="livre">
         <SPAN style="font-style:italic; padding-right:3pt">
             <xsl:value-of select="titre"/>
         </SPAN>
         <SPAN style="color:red; padding-right:3pt">
             <xsl:value-of select="auteur"/>
         </SPAN>
         <SPAN style="color:blue">
             <xsl:value-of select="ref"/>
         </SPAN>
         <br />
</xsl:template>

</xsl:stylesheet>
```

Le navigateur affichera comme avec le `for-each` la page suivante :



6 En conclusion

Comment visualiser un document XML à l'aide d'une feuille de style XSL :

1- Appel de la feuille de style dans le document XML :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="biblio.xsl"?>

<bibliotheque>

<livre>
        <titre>N ou M</titre>
        <auteur>Agatha Christie</auteur>
        <ref>Policier-C-15</ref>
</livre>

<livre>
        <titre>Le chien des Baskerville</titre>
        <auteur>Sir Arthur Conan Doyle</auteur>
        <ref>Policier-D-3</ref>
</livre>

<livre>
        <titre>Dune</titre>
        <auteur>Franck Heckbert</auteur>
        <ref>Fiction-H-1</ref>
</livre>

</bibliotheque>
```

2- Contenu de la feuille de style ***biblio.xsl*** :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>

<xsl:template match="/">

 <html>
   <head>
     <title>Ma bibliotheque</title>
   </head>
   <body>
         <H2>Bibliotheque</H2>

         <xsl:for-each select="bibliotheque/livre">
         <SPAN style="font-style:italic; padding-right:3pt">
             <xsl:value-of select="titre"/>
         </SPAN>
         <SPAN style="color:red; padding-right:3pt">
             <xsl:value-of select="auteur"/>
         </SPAN>
         <SPAN style="color:blue">
             <xsl:value-of select="ref"/>
         </SPAN>
         <br />
         </xsl:for-each>

    </body>
    </html>

</xsl:template>
</xsl:stylesheet>
```

3- Affichage dans un navigateur Web :