ED n° 9 sur XML

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. Pour cela, le plan de cet ED est découpé selon les parties suivantes :

1 1ère PARTIE : GENERALITES

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) versions de la norme :
  - ✓ (octobre 2002) mise à jour pour utiliser Unicode 3 (*candidate recommandation*)
  - ✓ seconde édition du W3C du 6 octobre 2000 - Recommandation W3C
  - ✓ du 10 février 1998
- ✓ 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 ou DTD (elle définit les éléments qui composeront le vocabulaire, les attributs de tous les éléments, ainsi que les entités).

1.2 Pourquoi XML ?

Le Web est confronté à deux problèmes :

- ✓ HTML n'est pas extensible, il ne peut pas répondre aux besoins spécifiques de tous les domaines (mathématiques, chimie, musique, astronomie...) et ne définit plus le contenu du document,
- ✓ SGML qui permettrait de définir de nouveaux langages de balisage spécifiques est complexe.

➔ XML apporte une réponse à ces problèmes

1.2.1 Caractéristiques de HTML

HTML est une application SGML. HTML est restrictif : il définit un ensemble d'Éléments et attributs fixe, qui permet de décrire un document simple (en-têtes, corps, ce dernier contenant des paragraphes, listes, tableaux, illustrations, ...). De manière plus formelle HTML 4.0 est une DTD de SGML (et XHTML1.0 est à présent une DTD XML), c'est-à-dire qu'une grammaire définit tous les éléments et attributs autorisés.

L'ajout de nouveaux éléments est impossible, on ne peut pas créer des balises pour définir pour le document une structure de type résumé, chapitre, index, bibliographie.

HTML qui devait décrire le contenu du document s'est orienté vers la présentation du contenu :

- ✓ à l'origine, l'idée du créateur de HTML était de fournir aux auteurs un outil d'échange de données indépendant des plate-formes et de l'affichage :
    - HTML devait décrire les différentes parties d'un document simple à l'aide d'éléments (Hn, P, STRONG, CITE, ...),
    - les navigateurs devaient se charger de l'affichage final du document.
- ✓ Des éléments d'affichage ont été rajoutés pour personnaliser les pages web (B, I, FONT, CENTER, TABLE dans la version 3.2, FRAME dans la version 4.0) et des éléments ont été détournés (emploi de BLOCKQUOTE pour créer une marge gauche alors que cette balise est censée introduire une citation entre guillemets),
- ✓ Le W3C a supprimé les éléments relatifs à la présentation au profit des feuilles de style CSS, cependant il subsistera des pages HTML non conformes pendant plusieurs années.

Les documents HTML sont rarement conformes aux règles établies par la DTD, les attributs sont rarement entre guillemets, les éléments ne sont pas toujours correctement imbriqués..., ceci est en partie la faute des navigateurs qui essayent d'afficher tous les documents HTML, même s'ils ne sont pas valides.

1.2.2 Caractéristiques de XML

- ✓ XML est un sous-ensemble de SGML, dont les caractéristiques inutiles pour la publication sur le Web ont été supprimées, la création de DTD est plus simple qu'avec SGML.
- ✓ 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 (DTD externe)
- ✓ le document ne sera affiché que s'il est bien formé et valide (s'il suit une DTD).
- ✓ 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).

1.2.3 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.
- ✓ 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.

1.2.4 Applications XML

XML est un métalangage permettant l'élaboration de balisages spécialisés. En fonction du contenu qu'on souhaite publier on définit ses propres balises. Applications existantes :

- ✓ AML *Astronomical Markup Language* langage décrivant les différents types de données utilisées en astronomie.
- ✓ MathML *Mathematical Markup Language* notation mathématique sur le web.
- ✓ CML *Chemical Markup Language* pour la publication Internet des formules chimiques, de molécules, des équations, utilise une visionneuse Java nommée Jumbo pour visualiser les molécules.
- ✓ VML *Vector Markup Language* langage de balisage d'information graphique vectorielle.
- ✓ PGML *Precision Graphics Markup Language* décrit les structures de données graphiques complexes avec les primitives du langage Postscript. Il permet la conversion de documents aux formats ps et pdf en XML. Les textes sont des données de caractères XML standard au sein d'un élément qui utilise les attributs x et y pour définir le point de départ du fragment de texte. Il existe des éléments rectangle avec des attributs width et height, fillcolor...
- ✓ SMIL *Synchronized Multimedia Integration Language* pour la création multimédia, il spécifie comment et quand des éléments multimédia peuvent apparaître dans une page web. Par exemple on peut dire que sur la page le texte apparaît suivi d'une série d'images qui sont accompagnées d'une musique. Il est là pour ajouter un aspect temporel aux pages Web. Il permet de contrôler la position dans l'espace des objets et dans le temps.
- ✓ CDF *Channel Definition Format* utilisé par Microsoft pour décrire le contenu Active Channel. Une chaîne délivre des informations directement à l'utilisateur en utilisant la technologie *push* d'un serveur (envoi de contenus web à des utilisateurs sans que ceux-ci aient besoin d'accéder spécifiquement au site). Les chaînes fournissent des informations récentes aux utilisateurs qui peuvent sélectionner le contenu Web qu'ils souhaitent recevoir. CDF propose trois composants de base :
  - du contenu active channel (une page web principale et un jeu de pages qui lui sont reliées pour constituer le contenu de la chaîne),
  - des graphiques pour permettre de mettre un logo,
  - un fichier CDF qui établit la structure et l'agenda de mise à jour du contenu de la chaîne.
- ✓ RDF les applications traitant les données RDF peuvent récupérer les informations (auteur, URL, titre, description) et créer des bases de données permettant la recherche d'information.
- ✓ WML *Wireless Markup Language* le langage de balisage pour l'internet mobile.

2 2ème PARTIE : LE DOCUMENT XML

2.1 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* 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 DTD. Si un document a une DTD associée et qu'il se conforme à cette DTD, il est dit **valide**. S'il n'a pas de DTD 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.

2.2 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.

2.3 Document XML valide

Un document est valide s'il :

- est bien formé,
- fait référence a une DTD,
- se conforme a la DTD.

Les **DTD** *Document Type Definition* définissent les éléments et les règles d'utilisation : noms des éléments, attributs possibles pour un élément, imbrications (HTML 4.0 est une DTD de SGML).

2.4 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.

2.4.1 Le PROLOGUE

Cette partie comprend toutes les déclarations internes et externes.

Déclaration XML :

```
<?xml version="1.0" [encoding = "encodage"] [standalone = "yes|no"] ?>
```

Cette déclaration (qui est en fait une instruction de traitement) contient des informations pour le processeur. Elle indique que ce document est conforme à la version 1.0 de la norme XML. Elle peut préciser le jeu de caractères utilisés dans le document (encoding) et s'il y a des références externes ou non (standalone).

Déclaration de DTD :

La DTD peut être incluse dans le document ou être dans un fichier externe. Elle est placée dans un fichier si elle doit servir dans plusieurs documents XML différents.

Exemple de DTD externe :

```
<!DOCTYPE element_racine SYSTEM|PUBLIC [nom] uri_DTD>
<!DOCTYPE element_racine SYSTEM "uri_DTD">
<!DOCTYPE element_racine PUBLIC "nom" "uri_DTD">
```

`DOCTYPE` permet de déclarer le type du document, le nom de l'élément racine est précisé. `SYSTEM` indique que la DTD est stockée sur l'ordinateur à l'adresse précisée. `PUBLIC` est utilisé quand la DTD est publiée pour beaucoup de personnes, dans ce cas le processeur XML peut utiliser le nom pour retrouver cette DTD (ex une DTD publiée par le W3C), en cas d'échec il utilise l'uri.

Exemple de DTD interne :

La DTD interne se déclare entre crochets dans le DOCTYPE.

```
<!DOCTYPE librairie [
        <!ELEMENT livre (titre, auteur, editeur..., commentaire?)>
        <!ELEMENT titre (#PCDATA)>
        ...
        <!ELEMENT commentaire (#PCDATA)>
]>
```

2.4.2 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>).

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).

Ces deux types d'entités étaient particuliers car prédéfinis, vous pouvez si vous le souhaitez définir vos propres entités par le biais de la DTD. La déclaration suivante :
`<!ENTITY deg "&#176;">`
permettra d'utiliser l'abréviation ° dans le document, par exemple :
`il fait 25&deg;C.`

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). Dans un document XML, l'attribut doit être déclaré de type ENTITY dans la DTD et la déclaration d'entité doit spécifier une NOTATION déclarée. Voici un extrait de DTD qui définit plage comme une notation JPEG et l'utilise comme valeur de l'attribut source de l'élément vide image.

```
&!-- extrait de la DTD -->
<!NOTATION JPEG SYSTEM "Joint Photographic Experts Group">
<!ENTITY plage SYSTEM "plage.jpg" NDATA JPEG>
<!ATTLIST image source ENTITY #REQUIRED>


&!-- appel dans le document -->
<image source="plage"/>
```

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. Ils peuvent aussi être incorporés avec le mécanisme NOTATION et ENTITY.

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 & on peut aussi déclarer une entité pour les représenter :

```
<!ENTITY deg "&176;">
Il fait 30&deg; celsius.
```

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. ;

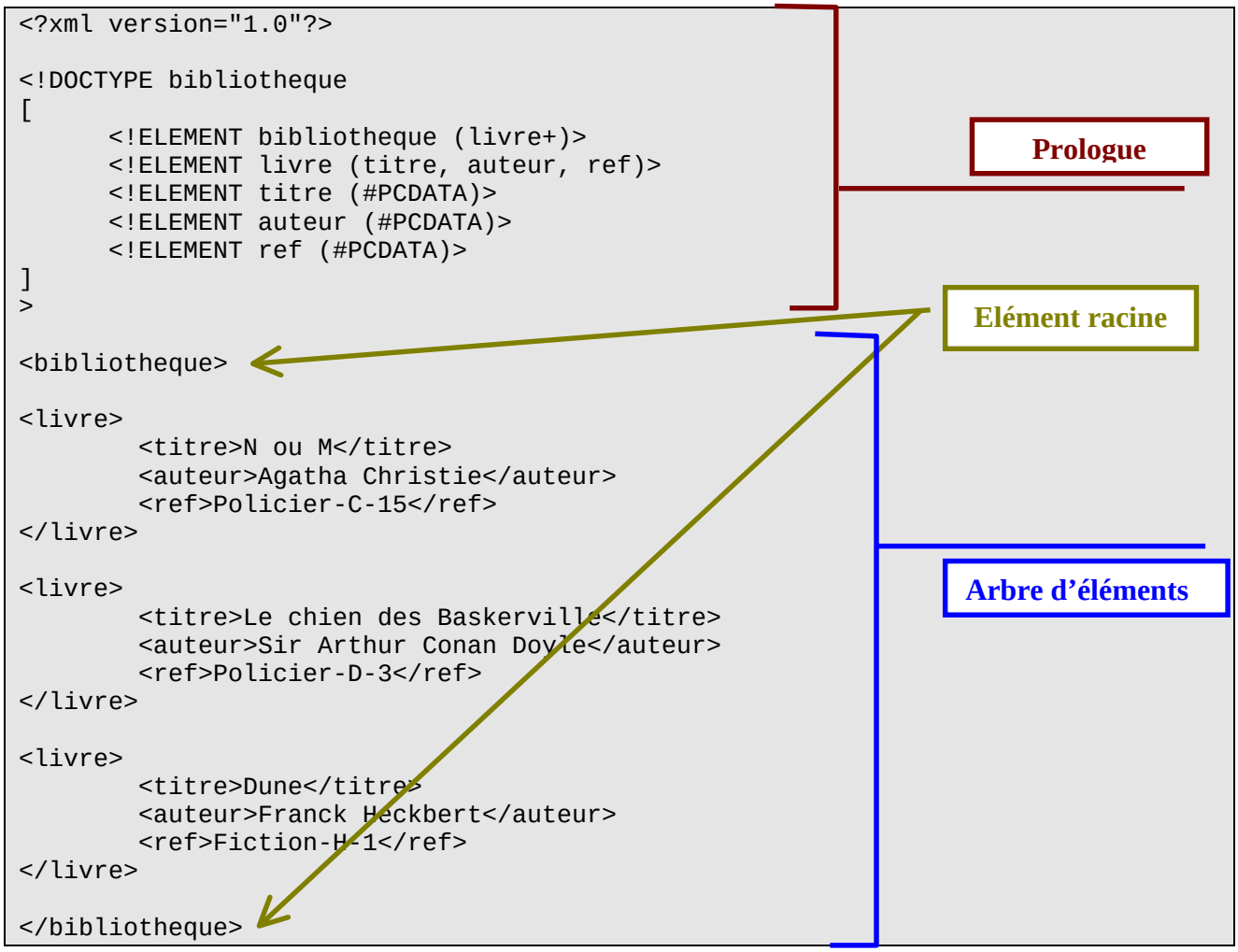
`<p <!-- balise ouvrante de paragraphe --> >` est illéga.

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 <).

2.5 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>
```

3 3ème PARTIE : LA DTD

3.1 Description générale

La définition de type de document donne la grammaire de l'application XML. Elle décrit :

- ✓ les éléments types : elle donne les noms de tous les éléments et leur modèle de contenu,
- ✓ les attributs pour chaque élément (nom, type et valeur par défaut),
- ✓ les entités binaires ou textuelles pouvant être incluses dans un document (les caractères non ASCII par exemple),
- ✓ les notations qui servent à identifier les types spécifiques de données externes binaires.

Rappelons qu'il n'est pas nécessaire de créer une DTD pour afficher un document XML.

3.2 Déclarations de bases

Un document XML peut ne pas avoir de DTD, un tel document définit son propre balisage de manière informelle. Il doit cependant être bien formé, sinon il ne sera pas affiché par un navigateur.

Lorsque aucune DTD n'est utilisée le document doit préciser dans la déclaration XML qu'il est autonome (Standalone Document Declaration ou SDD).

D'autre part, un document peut avoir une DTD externe ou une DTD interne ou encore une DTD interne et une DTD externe. Pour un document la DTD totale est la réunion du sous-ensemble externe et du sous-ensemble interne.

Une DTD externe s'applique à plusieurs documents, quand on modifie un élément la modification s'applique à tous les documents (pour une DTD interne il faudrait faire la modification dans toutes les DTD internes).

Le sous-ensemble interne n'aura aucun effet sur les autres documents faisant référence au sous-ensemble externe de la DTD. Le sous-ensemble interne a priorité sur le sous-ensemble externe ce qui permet de modifier localement un document.

La déclaration de type de document s'effectue dans le prologue avec la déclaration `DOCTYPE` qui permet de donner un sous-ensemble interne et externe.

```
<!DOCTYPE racine SYSTEM|PUBLIC "url"
   [ sous_ensemble_interne ] >
```

3.2.1 Déclarations internes

Le sous-ensemble interne de la DTD est placé dans le `DOCTYPE` après la déclaration XML. L'attribut `standalone` prendra dans la déclaration XML la valeur `"yes"` s'il n'y a pas à rechercher de DTD externe, `"no"` sinon.

```
<?xml version="1.0" standalone="yes">
<!DOCTYPE racine [ <!-- sous-ensemble de DTD interne --> ]>
<racine>
  ...
</racine>
```

3.2.2 Déclarations externes

Le sous-ensemble externe est placé dans un fichier séparé. L'appel à ce fichier qui doit avoir le même nom que la racine du document (élément qui englobe tous les autres) est effectué dans le `DOCTYPE` après la déclaration XML. L'attribut `standalone` prendra dans la déclaration XML la valeur `"no"`.

```
<!DOCTYPE racine (SYSTEM "url") | (PUBLIC "identifiant_public"
"url"?)>
```

3.2.3 Déclaration PUBLIC

`PUBLIC` est utilisé lorsque la DTD est une norme ou qu'elle est enregistrée sous forme de norme ISO par l'auteur.

`identifiant_public` contient les caractéristiques : type_enregistrement // proprietaire // DTD description // langue, avec :

- ✓ type_enregistrement : un signe + si c'est selon la norme ISO 9070, un signe - sinon ;
- ✓ proprietaire : nom du propriétaire (entreprise ou personne) ;
- ✓ DTD description : une description textuelle pour laquelle les espaces sont autorisés ;
- ✓ langue : un code de langue ISO 639.

L'adresse du fichier décrivant la DTD n'est pas obligatoire, le processeur XML peut utiliser les informations de l'identifiant public pour essayer de générer une adresse. Il faut noter cependant qu'il n'est pas toujours possible de trouver l'adresse à partir de l'identifiant, il est donc conseillé de faire suivre l'identifiant par l'adresse du fichier.

Voici la déclaration de type de document pour un document html qui utilise la DTD XHTML1.0 :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

3.2.4 Déclaration SYSTEM

`SYSTEM` est utilisé pour donner l'adresse du fichier qui contient la DTD dans le cas où la DTD n'est pas publique.

```
<!DOCTYPE racine SYSTEM "http://www.serveur.fr/racine.dtd">
```

3.3 Déclarations d'éléments

Rappelons qu'un élément permet de structurer le document (par exemple livre est un élément d'un document bibliothèque, en HTML p, h1, form... sont des éléments). La déclaration d'éléments est de la forme :

```
<!ELEMENT nom_element modele_de_contenu>
```

3.3.1 Élément vide et non vide

Un élément vide ne contient aucun texte, aucun autre élément, comme les éléments IMG, HR, BR de HTML.

```
<!ELEMENT nom_element EMPTY>
```

Par exemple, dans la DTD de XHTML 1.0 l'élément img est déclaré : `<!ELEMENT img EMPTY>`

Un élément non vide est formé d'une balise ouvrante, d'un contenu et d'une balise fermante. Pour décrire ce contenu dans la DTD on utilise un modèle de contenu dans la déclaration d'éléments.

3.3.2 Élément de données

Si l'élément contient uniquement des données (autrement dit, s'il n'y a aucun autre élément inclus dans cet élément) on utilise le type #PCDATA qui signifie *Parsed Character DATA* ou les données caractères analysées :

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

Par exemple, dans la DTD de XHTML 1.0 l'élément title qui permet de donner le titre de la fenêtre est déclaré : `<!ELEMENT title (#PCDATA)>`

3.3.3 Élément conteneurs d'éléments

Si l'élément contient uniquement d'autres éléments, le modèle de contenu définit les éléments pouvant être inclus dans un élément donné et spécifie également les éléments devant apparaître obligatoirement, leur ordre et leur fréquence.

Les parenthèses dans la déclaration d'élément introduisent les éléments enfants de l'élément défini. L'ordre dans lequel les éléments sont précisés est l'ordre dans lequel ils doivent apparaître dans l'élément en cours de définition. Par exemple, dans la DTD de XHTML 1.0 l'élément racine html qui contient les éléments head et body est déclaré `<!ELEMENT html (head, body)>`.

Les indicateurs d'occurrences sont les suivants :

- ✓ ? = 0 ou 1 fois
- ✓ * = 0 ou n fois
- ✓ + = au moins une fois

Par exemple, dans la DTD de XHTML 1.0 l'élément ul est déclaré : `<!ELEMENT ul li+>`

Pour indiquer des choix, il est possible d'utiliser le |. Dans la DTD XHTML1.0 l'élément dl est déclaré : `<!ELEMENT dl (dt|dd)+>`

3.3.4 Élément mixte

Si l'élément a un contenu mixte (données + éléments) il est possible d'utiliser PCDATA et les éléments imbriqués. Par exemple, dans la DTD de XHTML 1.0 l'élément object est déclaré :

```
<!ELEMENT object (#PCDATA | param | %block; | form | %inline; |
%misc;)*>
```

3.3.5 Élément libre

Élément qui peut contenir tout élément déclaré dans la DTD et du texte.

```
<!ELEMENT nom_element ANY>
```

3.4 Exemples d'utilisation de la DTD

Voici la DTD qui peut être utilisée pour le document bibliothèque ; ***biblio.xml***.

```
<!ELEMENT bibliotheque (livre+)>
<!ELEMENT livre (titre, auteur, ref)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT ref (#PCDATA)>
```

3.4.1 DTD interne

Avec une DTD interne le document XML serait le 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>
```

3.4.2 DTD externe

Avec la même DTD mais externe on aurait alors dans le fichier externe ***biblio.dtd*** :

```
<!ELEMENT bibliotheque (livre+)>
<!ELEMENT livre (titre, auteur, ref)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT ref (#PCDATA)>
```

et dans le document XML :

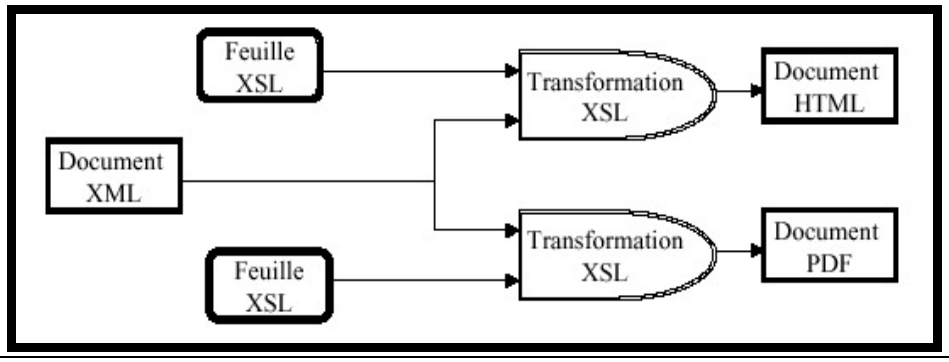
```
<?xml version="1.0"?>

<!DOCTYPE bibliotheque SYSTEM "biblio.dtd">

<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 4ème PARTIE : LES FEUILLES DE STYLES

4.1 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.

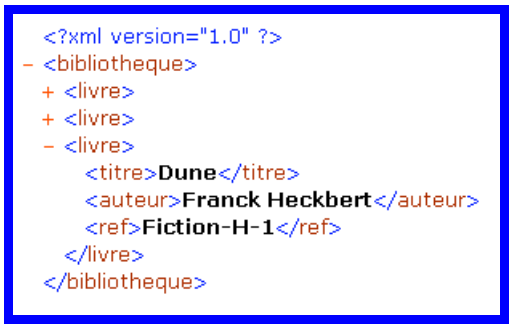
XSL est beaucoup plus qu'un simple langage pour le formatage de documents XML (ce que pourrait très bien faire CSS - Cascading Style Sheets). Il permet de retraiter un document XML et ainsi de réarranger sa structure. En fait XSL permet de transformer un document XML en un autre document, souvent XML,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



4.2 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 :

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



4.3 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***.

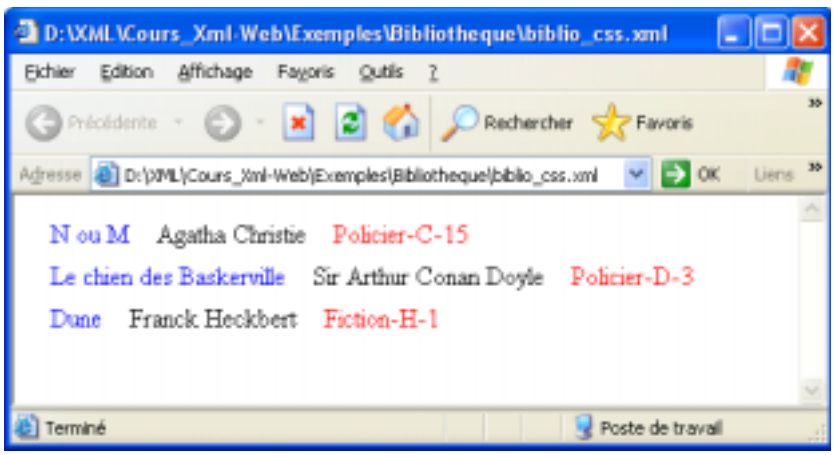
Appel de la feuille de style dans le document :

```
<?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 :



4.4 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>
```

4.4.1 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.

4.4.2 Modèle unique pour un document XML

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.

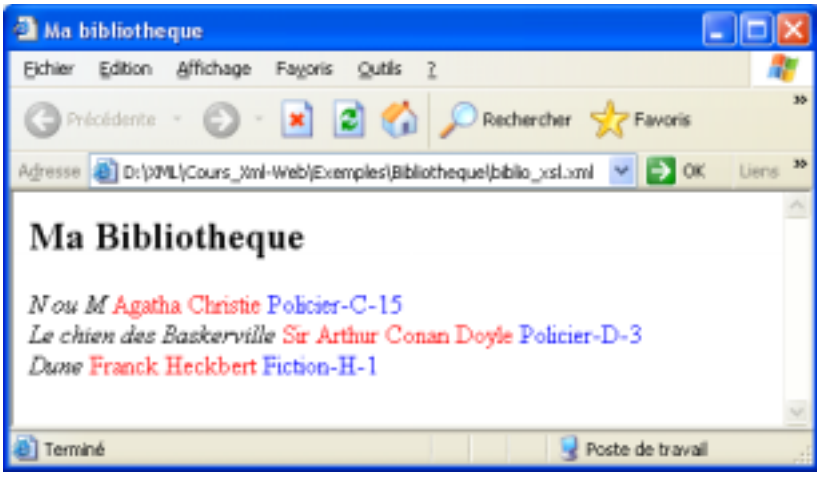
Le modèle est appliqué à la branche spécifiée par l'attribut `match` de l'élément `template`. En CSS la valeur de l'attribut match correspondrait au sélecteur de la règle. Dans notre exemple c'est à la racine du document XML (ne pas confondre avec l'élément racine `bibliotheque` qui est un enfant de 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 :



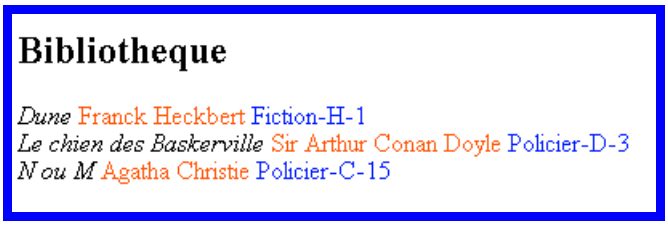
4.4.3 Classement

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 :



4.4.4 Filtrage

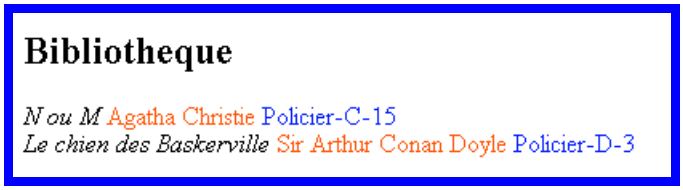
Ajoutons dans le document XML un attribut `type` qui correspond au type du livre :

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

<!DOCTYPE bibliotheque
[
      <!ELEMENT bibliotheque (livre+)>
      <!ELEMENT livre (titre, auteur, ref)>
      <!ATTLIST livre type (policier | science-fiction | aventure) #REQUIRED>
      <!ELEMENT titre (#PCDATA)>
      <!ELEMENT auteur (#PCDATA)>
      <!ELEMENT ref (#PCDATA)>
  ]
>

<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 :



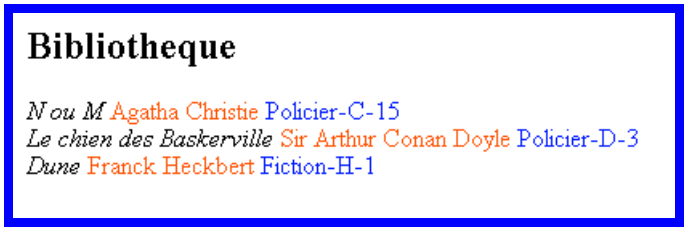
4.4.5 Modèle multiples

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 :



5 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 :