Introduction à

1. Introduction 3
1.1 Qu'est-ce que l'XML ? 3
1.2 Quelle est sa relation avec l'HTML ? Est-il destiné à remplacer l'HTML ? 4
1.3 Conclusion 4
2. Affichage d'un document XML 4
2.1 Mon premier document XML 4
2.2 Affichage d'un document XML sans feuille de style 5
2.3 Exercices 8
2.4 Création d'un document XML bien formé 8
2.4.1 Généralités 8
2.4.2 Les commentaires et la section CDATA 9
2.4.3 Les attributs 10
2.4.4 Exercices 11
3. Création d'un document XML valide 11
3.1 Création de la DTD 12
3.1.1 Déclarations de type d'éléments 12
3.1.2 Déclarations de listes d'attributs 18
3.2 Exercices 19
3.3 XML Schema Definition (XSD) 23
3.3.1 Exemple de base 23
3.3.2 Règles de base 24
3.3.3 Exercices 25
4. Affichage d'un document XML avec une feuille de style 26
4.1 Petit rappel sur les règles des feuilles de style 27
4.2 Exercices 30
5. Les feuilles de styles XSL 30
5.1 Principes de base 31
5.1.1 Principe de fonctionnement 31
5.1.2 Principales instructions XSLT 31
5.1.3 value of select 32
5.1.4 *for each select* 33
5.1.5 *apply-templates select* 34
5.1.6 sort select 35
5.1.7 if test="expression" 36

5.1.8 choose 37
5.1.9 Commentaire 37
5.1.10 Transformation XML → _ HTML 38
5.2 XPATH 38
5.2.1 Les expressions XPATH 38
5.2.2 Les fonctions XPATH 39
5.2.3 Exemples d'expressions XPATH, utilisation des filtres 39
5.2.4 Les axes de recherche 40
5.3 XSL/FO 41
5.4 Exercices 43
6. Affichage d'un document XML avec un langage de script DOM 44
6.1 Liste des principaux objets DOM pour un document XML 44
6.2 Mozilla vs Internet Explorer 47
6.3 Mozilla vs Internet Explorer : la fonction « childNodes » 47
6.4 Exercices 48
7. PHP 5 49
7.1 SimpleXML 49
7.2 Exercices 50
7.3 DOM 51
7.4 Inclure une page XML associée à une page XSLT dans PHP 54
7.5 Exercices 55
8. RSS 55
8.1 Objectifs des flux RSS : 55
8.2 Création d'un fichier RSS 56
8.3 Afficher un flux RSS 57
8.4 Exemple de création d'un fichier statique RSS à partir d'une table mysql 58
9. Pour aller plus loin… 59
9.1 XQuery : XML comme une base de données 59
9.2 XForms 59
9.1 ASP (Active Server Page) 59
9.2 Liens évolués XLL : XPOINTER et XLINK 60
9.3 Images au format SVG 61
9.4 SAX 61
9.5 AJAX 61
10. Liens utiles 62

1. Introduction

1.1 Qu'est-ce que l'XML ?

XML est un acronyme d'**eXtensible Markup Language**. Le langage XML est utilisé pour le **stockage**, la **préparation** et la **livraison** d'informations sur le net. C'est une solution idéale pour le traitement de l'information qui ne cesse d'évoluer en quantité et complexité. Ce langage permet de décrire virtuellement tout type d'information depuis une simple recette de cuisine à un arbre généalogique.

Le principal objectif d'XML est **l'échange de données** ou documents entre sites distants ou applications différentes.

On pourrait dire que c'est un « langage universel permettant d'interfacer des systèmes ne parlant pas la même langue ».

C'est un standard parfaitement adapté par exemple aux contextes suivants :

- eCommerce ;
- Gestion électronique de documents, publication et reporting ;
- Gestion de transactions financières et boursières ;
- Stockage d'informations, dans un contexte de base de données aussi bien « objets » que « relationnelles » ;
- Interopérabilité entre systèmes hétérogènes ;
- Description de structures moléculaires, ADN ;
- Ecriture de partitions musicales ;
- Uniformisation des canaux de publication (PDA, téléphone, TV, PC, ...);
- Agrégation de **RSS** (par exemple, récupérer des dépêches.)
- Elaboration de documents à partir d'une seule source vers différents formats : PDF, HTML, WML, image SVG, …)
- La technologie **Google** utilise XML (SOAP et XML-RPC)
- …

Le XML a été défini par le groupe de travail du **W3C** (World Wide Web Consortium) en 1996 et finalisé deux ans après en février 1998. Il a décidé de définir un sous ensemble de **SGML**[1], comme un sorte de version simplifiée de celui-ci, optimisée pour le web. Le W3C a défini dix objectifs pour l'XML, à titre informatif :

- XML doit être directement utilisable sur Internet.
- XML doit prendre en charge une grande variété d'applications. (XML peut être utilisé pour l'échange d'informations entre programmes différents, bien que son but premier soit de délivrer de l'information.)
- XML doit être compatible avec le SGML (XML, comme l'HTML est un sous-ensemble de SGML).
- Il doit être facile d'écrire des programmes pour le traitement des documents XML. (c'était le gros inconvénient du SGML)
- Le nombre de caractéristiques optionnelles dans XML doit être maintenu à un minimum absolu, l'idéal étant zéro.
- Les documents XML doivent être lisibles et clairs (permettre aux non programmeurs de comprendre le code).
- XML doit faire l'objet d'un développement rapide.
- La conception de l'XML doit être formelle et concise (respect de la norme EBNF (Extended Basckus Naur Form))
- Les documents XML doivent être faciles à créer.
- La concision du marquage XML n'a qu'une importance minime

[1] SGML = Structured Generalized Markup Language. Premier essai de normalisation concernant les documents électroniques, adopté comme standard en 1986. C'est le père de tous les langages de marquages. Le SGML n'est pas ou peu utilisé à cause de sa lourdeur pour délivrer efficacement de l'information sur le web. En effet, les caractéristiques offertes pas le SGML rendaient très difficile l'écriture des logiciels nécessaires au traitement et à l'affichage de l'information SGML dans les navigateurs web.

1.2 Quelle est sa relation avec l'HTML ? Est-il destiné à remplacer l'HTML ?

Plutôt que de remplacer l'HTML, XML s'utilise couramment en symbiose avec lui pour augmenter la faculté des pages web.
XML n'est pas le remplaçant de HTML, tout simplement parce qu'ils n'ont pas le même but : XML sert à structurer l'information, tandis que HTML a pour but de la présenter et de lui donner du sens

1.3 Conclusion

Pour conclure cette introduction nous retiendrons :

L'XML sert à stocker des données structurées (arbres) dans un fichier texte. L'XML est un standard ouvert et accepté L'XML offre une solution complète de stockage, manipulation, transformation, etc. de données structurées. L'XML n'est pas une fin mais un moyen.

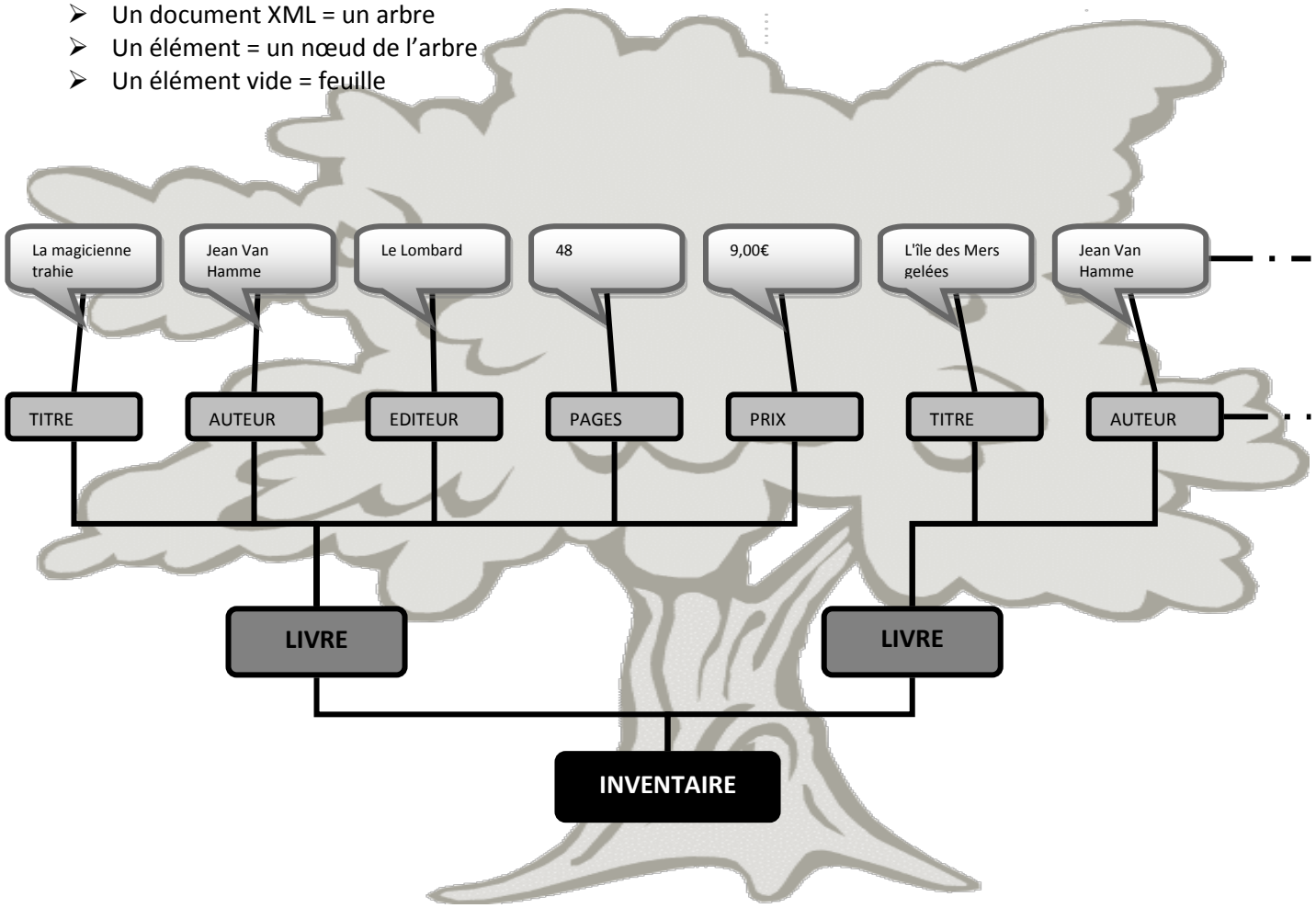
2. Affichage d'un document XML

2.1 Mon premier document XML

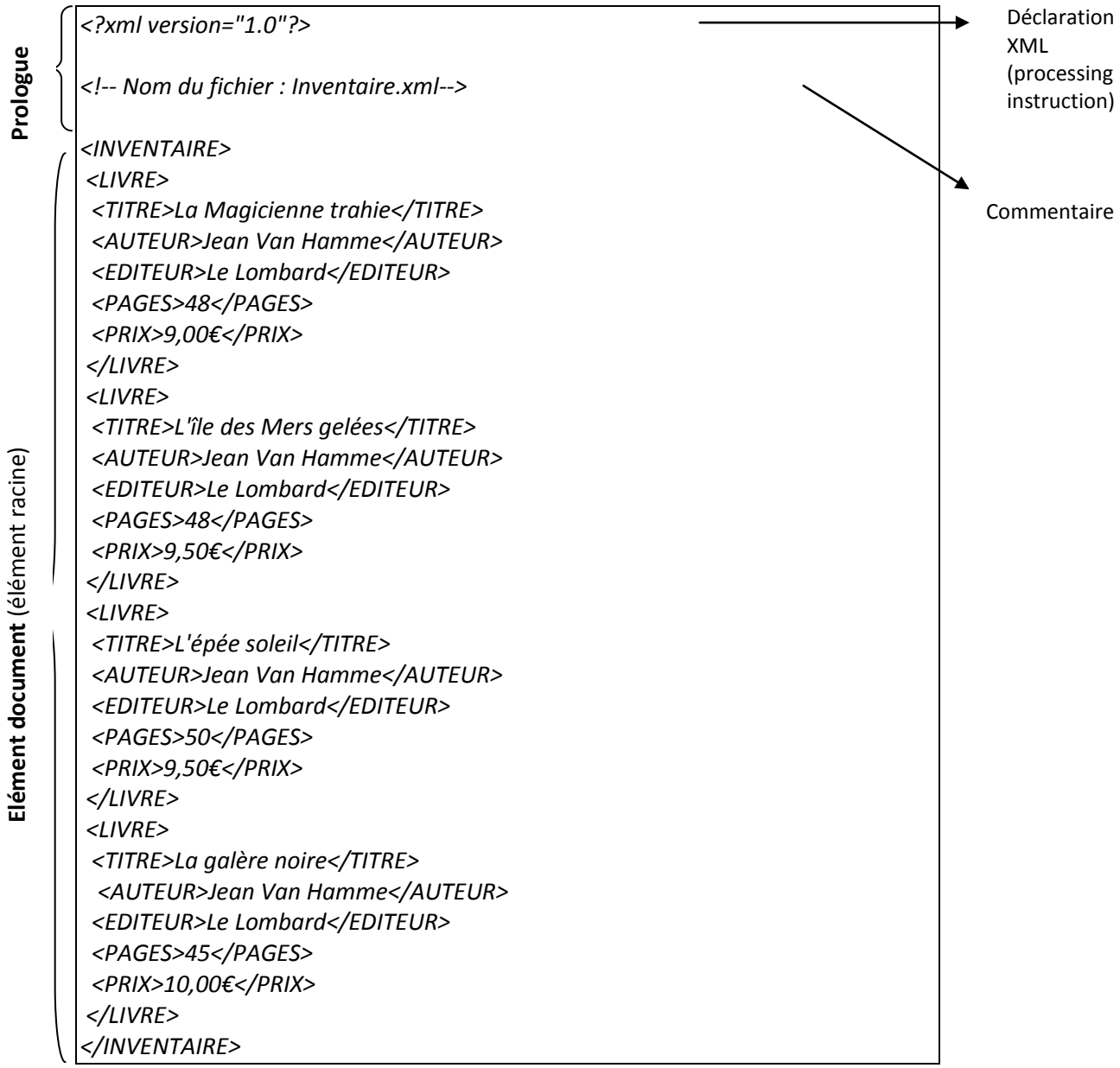
Comme premier exemple de document XML, nous allons recréer[2] le document suivant qui traite par exemple d'un inventaire de bandes dessinées :

La structure d'un document XML est celle d'un arbre :

- Un document XML = un arbre
- Un élément = un nœud de l'arbre
- Un élément vide = feuille



[2] Comme l'HTML, XML est écrit en mode texte. Vous pouvez utiliser le bloc-notes de Windows pour encoder vos documents, ou alors, sur le site http://www.allhtml.com/telechargement/categorie9.php , vous trouverez divers éditeurs XML téléchargeables gratuitement.



Le **prologue** d'un document XML comprend la déclaration XML (optionnelle)

<?xml version ="1.0" ?>

et peut aussi comprendre des commentaires (comme en HTML entre <!-- et -->)

Nous verrons plus tard qu'un prologue peut aussi contenir

- une **DTD** (document type definition) ou un lien vers un schéma XSD
- un ou plusieurs **instructions de traitement** (comme par exemple l'inclusion d'une feuille de style en CSS)

L'**élément document** est la seconde partie d'un document XML, correspond à un élément **unique**, élément racine (comparable au <BODY> en HTML) qui peut contenir d'autres éléments.

Un document XML est caractérisé par :

- le fait qu'il est un fichier XML **bien formé** (voir suivant)
- le fait qu'il est un fichier XML **valide** (voir chapitre 5)

2.2 Affichage d'un document XML sans feuille de style

Si vous voulez afficher un simple document XML, il faudra ouvrir la page dans votre navigateur.

Que constatez-vous avec ce fichier? S'affiche-t-il correctement?

Vous aurez un problème avec les caractères spéciaux…

Les normes ISO servent à codifier les caractères avec accents ou symboles pour qu'ils soient lisibles partout dans le monde. Si nous ne le faisons pas il est probable qu'ils apparaissent avec des symboles illisibles pour quelqu'un qui voudrait les consulter dans un pays n'utilisant pas notre norme.
La norme internationale comprend les caractères suivants, qui sont lus par tous les ordinateurs :

!"#$%&'()*+,-./
0123456789
:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`
abcdefghijklmnopqrstuvwxyz
{|}~

Tous les autres doivent être codifiés.

Effectivement un document XML doit se conformer à la norme ISO définie par le W3C.
Les deux pages suivantes contiennent une liste des principaux caractères accentués et des caractères spéciaux. Il suffit d'insérer le code ISO du caractère pour l'obtenir dans votre page. Avec cette manipulation, vous garantissez que l'accent sera reconnu correctement par tous les navigateurs.
Le jeu de caractère ISO-8859-1 (Latin 1) permet de codifier la plupart des langues de l'Europe occidentale. Vous pouvez dès lors placer dans l'en-tête d'un document XML la ligne suivante :

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

Ceci, afin de ne pas devoir coder chaque caractère spécial.
L'attribut *encoding* permet d'indiquer la représentation physique des caractères du fichier.

Caractère	Code Iso	Entité	Caractère	Code Iso	Entité	Caractère	Code Iso	Entité
À	À	À	Œ	Œ		ö	ö	ö
Á	Á	Á	[illegible]			÷	÷	÷
Â	Â	Â	Ž	Ž		ø	ø	ø
Ã	Ã	Ã	[illegible]			ù	ù	ù
Ä	Ä	Ä	[illegible]			ú	ú	ú
Å	Å	Å	‘	‘		û	û	û
Æ	Æ	&Aelig;	’	’		ü	ü	ü
Ç	Ç	Ç	“	“		ý	ý	ý
È	È	È	”	”		þ	þ	þ
É	É	É	•	•		ÿ	ÿ	ÿ
Ê	Ê	Ê	–	–		¡	¡	¡
Ë	Ë	Ë	—	—		¢	¢	¢
Ì	Ì	Ì	˜	˜		£	£	£
Í	Í	Í	™	™		¤	¤	¤
Î	Î	Î	š	š		¥	¥	¥
Ï	Ï	Ï	›	›		¦	¦	¦
Ð	Ð	Ð	œ	œ		§	§	§
Ñ	Ñ	Ñ	[illegible]			¨	¨	¨
Ò	Ò	Ò	ž	ž		©	©	©
Ó	Ó	Ó	Ÿ	Ÿ		ª	ª	ª
Ô	Ô	Ô	Espace			«	«	«
Õ	Õ	Õ	à	à	à	¬	¬	¬
Ö	Ö	Ö	á	á	á			
×	×	×	â	â	â	®	®	®
Ø	Ø	Ø	ã	ã	ã	¯	¯	&masr;
Ù	Ù	Ù	ä	ä	ä	°	°	°
Ú	Ú	Ú	å	å	å	±	±	±
Û	Û	Û	æ	æ	æ	²	²	²
Ü	Ü	Ü	ç	ç	ç	³	³	³
Ý	Ý	Ý	è	è	è	´	´	´
Þ	Þ	Þ	é	é	é	µ	µ	µ

ß	ß	ß	ê	ê	ê	¶	¶	¶
			ë	ë	ë	·	·	·
‚	‚		ì	ì	ì	¸	¸	¸
ƒ	ƒ		í	í	í	¹	¹	&supl;
„	„		î	î	î	º	º	º
…	…		ï	ï	ï	»	»	»
†	†		ð	ð	ð	¼	¼	¼
‡	‡		ñ	ñ	ñ	½	½	½
ˆ	ˆ		ò	ò	ò	¾	¾	¾
‰	‰		ó	ó	ó	¿	¿	¿
Š	Š		ô	ô	ô	€	€ €	€
‹	‹		õ	õ	õ			

Après modification, notre document devient :

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

<!-- Nom du fichier : Inventaire.xml-->

<INVENTAIRE>
 <LIVRE>
  <TITRE>La Magicienne trahie</TITRE>
  <AUTEUR>Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>48</PAGES>
  <PRIX>9,00&#8364;</PRIX>
 </LIVRE>
 <LIVRE>
  <TITRE>L'&#238;le des Mers gel&#233;es</TITRE>
  <AUTEUR>Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>48</PAGES>
  <PRIX>9,50&#8364;</PRIX>
 </LIVRE>
 <LIVRE>
  <TITRE>L'&#233;p&#233;e soleil</TITRE>
  <AUTEUR>Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>50</PAGES>
  <PRIX>9,50&#8364;</PRIX>
 </LIVRE>
 <LIVRE>
  <TITRE>Les bleus de la marine</TITRE>
  <AUTEUR>LAMBIL/CAUVIN </AUTEUR>
  <EDITEUR>Dupuis</EDITEUR>
  <PAGES>45</PAGES>
  <PRIX>10,00&#8364;</PRIX>
 </LIVRE>
</INVENTAIRE>
```

Vous pouvez maintenant ouvrir le document dans votre navigateur : (Internet explorer adaptera automatiquement une feuille de style pour le document XML)

En cliquant sur les « **+** » et « **-** », vous aurez la possibilité d'ouvrir ou de fermer des éléments du fichier XML :

2.3 Exercices

1) **[traducteur.xml]** Écrivez un document XML qui contiendra un élément racine <TRADUCTEUR> et qui proposera une liste mots avec leurs traductions en français, anglais et néerlandais.
   Par exemple, le mot Bonjour :
   Sous le tag français, on aura le contenu « Bonjour » ;
   Sous le tag anglais, on aura le contenu « Hello » ;
   Sous le tag néerlandais, on aura le contenu « Dag ».
   Ajouter au moins 8 mots dans ce document.
2) **[listedvd.xml]** Faites un document XML qui reprend la liste de tous vos DVD, avec, par DVD, le titre du film, la date de sortie du film et une zone « commentaire ».

2.4 Création d'un document XML bien formé

2.4.1 Généralités

Un document XML **bien formé** dépend des conditions suivantes :

- Il y a un élément racine unique qui contient tous les autres éléments (INVENTAIRE dans l'exemple),
- Les éléments doivent être correctement emboîtés. Aucun chevauchement des éléments n'est accepté. L'agencement suivant est incorrect :

  <LIVRE> La Magicienne trahie
  <AUTEUR>Jean Van Hamme</LIVRE></AUTEUR>

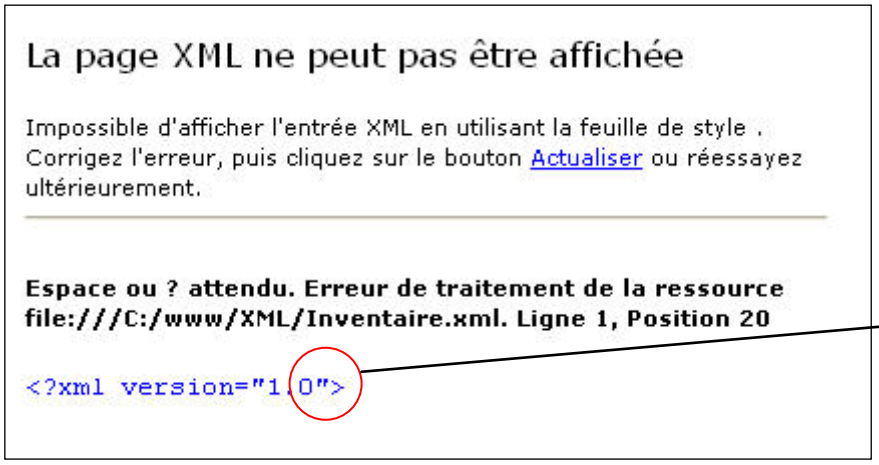
- Chaque élément doit avoir un marqueur d'**ouverture** et un marqueur de **fermeture**. Sauf l'élément sans contenu exemple : <LIVRE/> (Les balises uniques doivent être de la forme <balise/>)

- Les noms d'éléments sont sensibles à la casse
- Les commentaires se marquent comme en HTML entre < !-- et -->
- Dans notre premier document, l'élément <LIVRE> est appelé « élément parent », et l'élément <TITRE> un « élément enfant ».
- Le nom des balises peut contenir des lettres, des chiffres, des tirets (-), des points, des tirets bas (_),et doit contenir au moins une lettre.
- Le nom doit débuter par une lettre ou un souligné (_) suivi éventuellement de plusieurs lettres, chiffres, points, tirets ou soulignés.

Les éléments des balises suivantes sont-ils légaux ?

	OUI	NON
Part	❑	❑
1erPlace	❑	❑
_1erePlace	❑	❑
Section B	❑	❑
Section/B	❑	❑
A	❑	❑
:Leçon	❑	❑
SECTION-B	❑	❑
Rue.Adresse.1	❑	❑
A:Section	❑	❑

Le non respect des conditions précédentes entraînera systématiquement une erreur. De nombreux outils, éditeurs ou « parser » XML examinent le code XML et signalent si le document XML est bien formé.
Si vous commettez une erreur, le navigateur vous affichera un message du style :



Ici, il manque un "?" à la première ligne…

<?xml version="1.0"?>

2.4.2 Les commentaires et la section CDATA

En XML, les commentaires débutent par **< !--** et se terminent par **-->**. Vous pouvez placer des commentaires n'importe où dans un document XML, en dehors du marquage lui-même.
La section CDATA est un bloc de texte dans lequel vous pouvez insérer librement n'importe quel caractère (&, <, …) excepté la chaîne **]]>**. La section CDATA commence par **< ![CDATA[** et se termine par **]]>**.
Exemple :

```
<LIVRE>
< ![CDATA[
<HTML>
Vous pouvez taper ici n'importe quel caractère sauf deux crochets droits suivis par un symbole plus grand que
</HTML>
]]>
</LIVRE>
```

Sans la section CDATA, le processeur supposerait que <HTML>, par exemple, est le début d'un élément HTML à interpréter, plutôt que du texte inséré dans l'élément LIVRE.

2.4.3 Les attributs

```
<?xml version="1.0"?>
<!-- Nom du fichier : Inventaire.xml-->

<?xml-stylesheet type="text/css" href="inventaire.css"?>

<INVENTAIRE>
 <LIVRE>
  <COUVERTURE Source="th1_cov.jpg"/>
  <TITRE>La Magicienne trahie</TITRE>
  <AUTEUR>Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>48</PAGES>
  <PRIX>9,00&#8364;</PRIX>
 </LIVRE>
 <LIVRE>
  <COUVERTURE Source="th2_cov.jpg"/>
  <TITRE>L'&#238;le des Mers gel&#233;es</TITRE>
  <AUTEUR>Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>48</PAGES>
  <PRIX>9,50&#8364;</PRIX>
 </LIVRE>
 <LIVRE>
  <COUVERTURE Source="th3_cov.jpg"/>
  <TITRE>L'&#233;p&#233;e soleil</TITRE>
  <AUTEUR>Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>50</PAGES>
  <PRIX>9,50&#8364;</PRIX>
 </LIVRE>
 <LIVRE>
  <COUVERTURE Source="th4_cov.jpg"/>
  <TITRE>La gal&#232;re noir</TITRE>
  <AUTEUR>Jean Van Hamme </AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>45</PAGES>
  <PRIX>10,00&#8364;</PRIX>
 </LIVRE>
</INVENTAIRE>
```

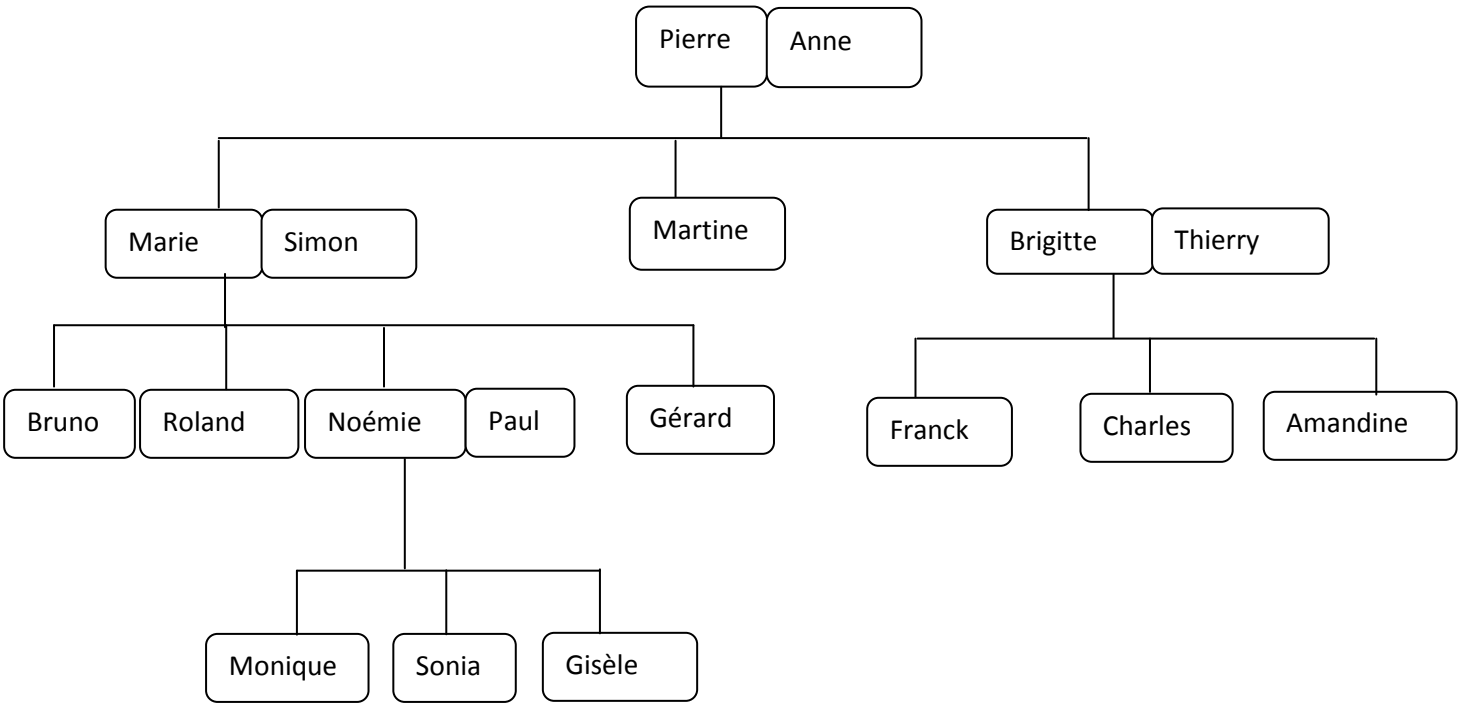
Dans le document XML ci-dessous, un élément vide nommé COUVERTURE au début de chaque élément LIVRE a été ajouté. C'est un attribut. Le but de cet élément, dans ce cas-ci, est d'indiquer à l'application XML d'afficher l'image spécifiée pour la couverture du livre. Cependant, pour utiliser un tel élément, nous aurons besoin des scripts ou des feuilles de style XSL (voir plus loin).

Définissons déjà les différentes propriétés des attributs

- Les valeurs des attributs doivent être entre " " ou entre ' ';
- Par convention, les noms des attributs sont écrits en minuscule ;
- Le nom des attributs peut contenir des lettres, des chiffres, des tirets (-), des tirets bas (_),et doit contenir au moins une lettre.
- Le nom doit débuter par une lettre ou un souligné (_) suivi éventuellement de plusieurs lettres, chiffres, points, tirets ou soulignés.
- Les noms d'attributs commençant par xml sont normalement réservés à des fins de standardisation.
- Un nom d'attribut ne peut apparaître qu'une seule fois dans un même marqueur.

2.4.4 Exercices

1) **[arbre.xml]** Réalisez un fichier XML avec l'arbre généalogique ci-dessus.



2) **[commune.xml]** Créez un fichier XML, pour la commune, qui contiendrait toutes les informations nécessaires de ses habitants.
3) **[html.xml]** Créez un fichier XML qui contient toutes les balises HTML et leurs explications.

3. Création d'un document XML valide

Qu'est-ce qu'un document XML valide ?

C'est un document bien formé, qui répond en outre à deux exigences :

- Le prologue du document contient une DTD (déclaration de type de document) ou un schéma W3C qui définit la structure du document
- Le reste du document doit se conformer à ce(tte) DTD/Schéma WC3

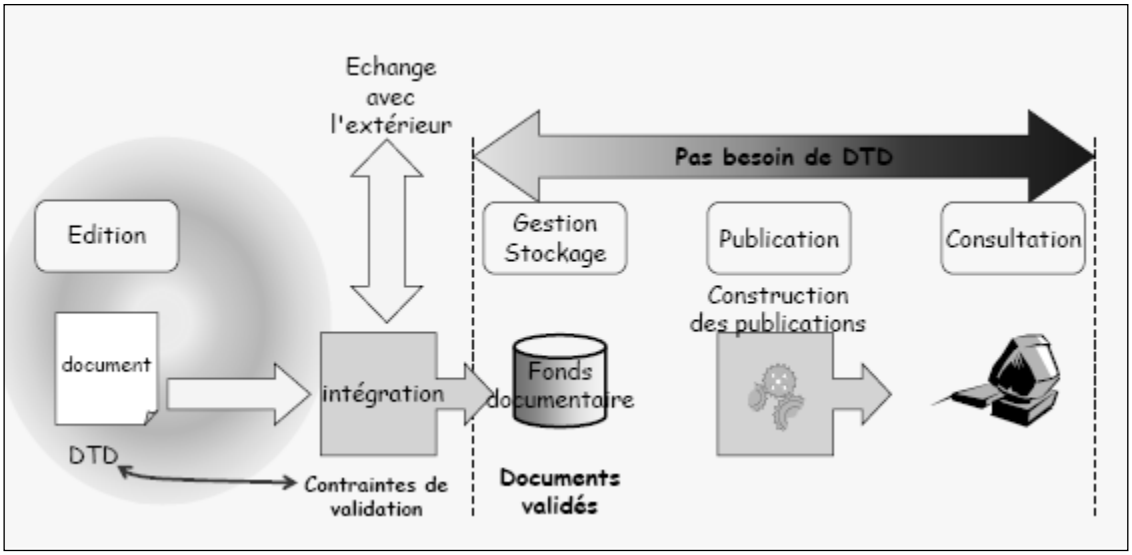
Pourquoi un document XML valide ?

- Uniformité « grammaire pour une classe de documents »
- Si plusieurs personnes travaillent sur un même document XML, la DTD garantit que tout le monde respectera la structure définie.
- Rendre plusieurs documents XML utilisables par un même logiciel de traitement

Cependant, le processeur d'Internet ne vérifie pas la validité d'un document, il vérifie uniquement si le document est bien formé (pas d'erreur fatale).

Le document commence par la déclaration XML : <?xml version="1.0" standalone="yes"?> (le "yes" de l'attribut standalone est une indication facultative qui signifie qu'il n'y a pas de DTD pour le document XML, "no" signifie qu'il y a une DTD)

Votre document DTD (composant optionnel d'un doc XML) peut être, entre autre, validé sur le site **http://validator.w3.org**, ou encore dans certains éditeurs XML (p.e. xmlSPY).



3.1 Création de la DTD

Une DTD se place dans le prologue d'un document XML.
La DTD se compose d'un crochet gauche **[** suivi d'une série de déclarations de marquage, suivi d'un crochet droit **]**

Exemple :

```
<?xml version="1.0" standalone="no" ?>
<!-- Nom du fichier : livresimple.xml-->
<!DOCTYPE INVENTAIRE
[
<!ELEMENT LIVRE ANY>
]
>
<INVENTAIRE>
<LIVRE>
Voici un document XML très simple
</LIVRE>
</INVENTAIRE>
```

Déclaration de type de document

Dans cet exemple, notre document XML ne peut contenir que des éléments de type LIVRE, et un élément TITRE peut avoir tout type de contenu possible (ANY)

Une DTD peut contenir les types de déclarations de marquage suivants :

3.1.1 Déclarations de type d'éléments

EMPTY : signifie que l'élément doit être vide, qu'il ne peut pas y avoir de contenu

```
< !ELEMENT IMAGE EMPTY>
```

Par exemple : <IMAGE></IMAGE> ou encore <IMAGE/>

ANY : Signifie que l'élément peut contenir tout type de contenu légal. Donc zéro, un ou plusieurs éléments enfants, dans n'importe quel ordre ou avec n'importe quel nombre de répétitions avec ou sans données texte disséminées.

Contenu d'élément (contenu de filiation)
L'élément peut contenir une filiation d'élément, mais ne peut pas contenir directement du texte.

```
<!DOCTYPE LIVRE
[
        <!ELEMENT LIVRE (TITRE,AUTEUR)>
```

```
        <!ELEMENT TITRE (#PCDATA)>
        <!ELEMENT AUTEUR (#PCDATA)>
]
>
```

L'élément LIVRE est déclaré comme ayant un contenu d'élément. L'expression (TITRE, AUTEUR) est appelé *modèle de contenu.* Celui-ci indique les types d'éléments enfants et leur ordre.
Le modèle de contenu peut avoir deux formes :

- (TITRE, AUTEUR)

→ L'élément LIVRE doit avoir un élément enfant TITRE suivi d'un élément enfant AUTEUR

- (TITRE | AUTEUR)

→ L'élément LIVRE peut contenir un élément TITRE ou un élément AUTEUR (attention il s'agit d'un ou exclusif, c'est soit l'un soit l'autre mais pas les deux !)
Vous pouvez modifier ces modèles de contenu en utilisant le « ? », « + » ou « * »

?	0 ou 1 occurrence de l'élément précédent
+	1 ou plusieurs occurrences de l'élément précédent
*	0, 1 ou plusieurs occurrences de l'élément précédent

Par exemple :

```
<!ELEMENT LIVRE(NOM+, AUTEUR ?, PRIX)>
<!ELEMENT NOM (#PCDATA)>
<!ELEMENT AUTEUR  (#PCDATA)>
<!ELEMENT PRIX (#PCDATA)>
```

L'élément suivant serait légal :

```
<LIVRE>
        <NOM>Roméo & Juliette</NOM>
        <NOM>Roméo et Juliette</NOM>
        <PRIX>10,00€</PRIX>
</LIVRE>
```

Vous pouvez aussi emboîter des modèles de contenu dans des modèles de séquence (et vice-versa)
Exemple :

```
<!ELEMENT LIVRE (TITRE, PRIX,(AUTEUR | DESSINATEUR | SCENARISTE))>
<!ELEMENT TITRE (#PCDATA)>
<!ELEMENT PRIX (#PCDATA)>
<!ELEMENT AUTEUR (#PCDATA)>
<!ELEMENT DESSINATEUR (#PCDATA)>
<!ELEMENT SCENARISTE (#PCDATA)>
```

Contenu mixte

Un élément peut avoir un contenu mixte soit

- Des données de caractères uniquement (#PCDATA)

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

(nb : l'élément SOUSTITRE peut rester vide ou il peut contenir des données de caractères)

- Des données de caractères plus une filiation optionnelle d'éléments

```
<!ELEMENT TITRE (#PCDATA | SOUSTITRE)*>
```

Exemple :

1 Soit la DTD suivante :

```
<!DOCTYPE INVENTAIRE
[
        <!ELEMENT INVENTAIRE (TITRE)>
        <!ELEMENT TITRE (#PCDATA | SOUSTITRE)*>
        <!ELEMENT SOUSTITRE (#PCDATA)>
]>
```

Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci zéro, une ou plusieurs fois (il s'agit donc d'un **OU inclusif**)
Mais notre document XML ne peut contenir qu'un seul « TITRE ».

Pour les éléments XML suivants :

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        <SOUSTITRE>Les archers</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
        Thorgal
        <SOUSTITRE> Le viking</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

2 Soit la DTD suivante :

```
<!DOCTYPE INVENTAIRE
[
        <!ELEMENT INVENTAIRE (TITRE)*>
        <!ELEMENT TITRE (#PCDATA | SOUSTITRE)>
        <!ELEMENT SOUSTITRE (#PCDATA)>
]
>
```

Elle signifie que pour un élément TITRE, on peut avoir :

- soit des données caractères
- soit un élément SOUSTITRE

Et ceci une et une seule fois (il s'agit donc d'un **OU exclusif**).
Mais le documents XML peut contenir zéro, un ou plusieurs « TITRE ».

Pour les éléments suivants :

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

✓

```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        <SOUSTITRE>Les archers</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
        Thorgal
        <SOUSTITRE> Le viking</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

3 Soit la DTD suivante :

```
<!DOCTYPE INVENTAIRE
[
        <!ELEMENT INVENTAIRE (TITRE)>
        <!ELEMENT TITRE (#PCDATA , SOUSTITRE)*>
        <!ELEMENT SOUSTITRE (#PCDATA)>
]
>
```

Elle signifie que pour un élément TITRE, on doit avoir :

- des données caractères

ET

- un élément SOUSTITRE

Et ceci <u>zéro, une ou plusieurs fois</u> (il s'agit donc d'un **ET inclusif**)
Mais notre document XML ne peut contenir qu'un seul TITRE.

Pour les éléments XML suivants :

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

✓

```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        <SOUSTITRE>Les archers</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
</TITRE>
</INVENTAIRE>
```

✓

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
        Thorgal
        <SOUSTITRE> Le viking</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

✓

4 Soit la DTD suivante :

```
<!DOCTYPE INVENTAIRE
[
        <!ELEMENT INVENTAIRE (TITRE)*>
        <!ELEMENT TITRE (#PCDATA , SOUSTITRE)>
        <!ELEMENT SOUSTITRE (#PCDATA)>
]
>
```

Elle signifie que pour un élément TITRE, on doit avoir :

- des données caractères

ET

- un élément SOUSTITRE

Et ceci un et une seule fois (il s'agit donc d'un **ET exclusif**)
Mais le document XML peut contenir zéro, un ou plusieurs « TITRE ».

Pour les éléments XML suivants :

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

✓

```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        Thorgal
</TITRE>
</INVENTAIRE>
```

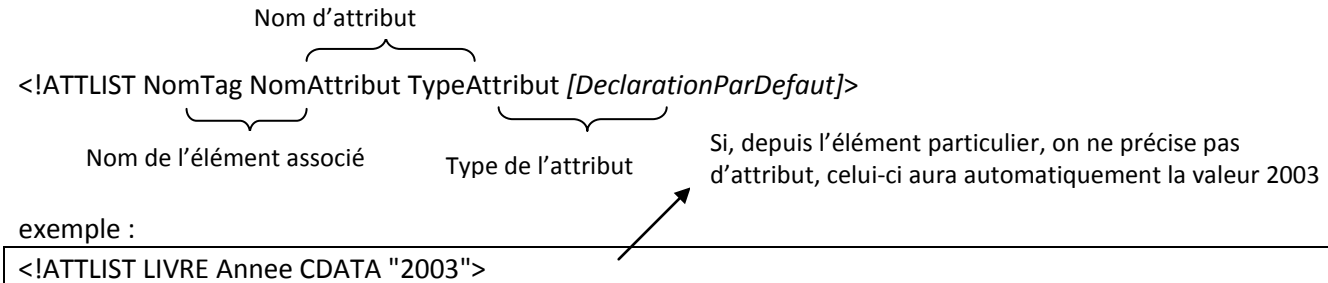
```
<INVENTAIRE>
<TITRE>
        <SOUSTITRE>La magicienne trahie</SOUSTITRE>
        <SOUSTITRE>Les archers</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
</TITRE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<TITRE>
        Thorgal
        <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
        Thorgal
        <SOUSTITRE> Le viking</SOUSTITRE>
</TITRE>
</INVENTAIRE>
```

3.1.2 Déclarations de listes d'attributs

Une déclaration de liste d'attributs a la forme générale suivante



exemple :

```
<!ATTLIST LIVRE Annee CDATA "2003">
```

ou encore

```
<!ATTLIST LIVRE Annee CDATA #REQUIRED>
```

- #REQUIRED : Une valeur **doit** être affectée à l'attribut Année.
- #IMPLIED : l'attribut est facultatif (on peut inclure ou non l'attribut dans un élément de type associé)
- #FIXED valeur par défaut (et uniquement celle-là)
- Ou encore directement une valeur par défaut entre guillemets

Remarques :

- vous pouvez déclarer plusieurs attributs à la fois
- Si vous déclarez plusieurs fois le même attribut, seule la première déclaration sera prise en compte, les suivantes seront ignorées.

Type d'attribut

- **type de chaîne**
  - **CDATA**: tout type de chaîne conforme aux règles d'attribut (voir point 4.3)
- **type tokenisé**

Même principe qu'un type de chaîne, mais on ajoute en plus une contrainte particulière ;
Exemple

```
<!ELEMENT INVENTAIRE (LIVRE*)>
<!ELEMENT LIVRE (#PCDATA)>
<!ATTLIST LIVRE ISBN ID #REQUIRED>
```

Le mot-clé **ID** précise que l'attribut ISBN doit avoir une valeur unique (un peu comme une clé primaire dans une table)

IDREF : La valeur de l'attribut doit correspondre à la valeur d'un attribut de type ID dans un élément à l'intérieur du document (un peu comme une clé étrangère dans une table)
Exemple

```
<!ELEMENT ITEM (#PCDATA)>
<!ATTLIST ITEM CodeStock ID #REQUIRED VaAvec IDREF #IMPLIED>
…
<ITEM CodeStock="S034">Moulin  à café électrique</ITEM>
<ITEM CodeStock="S047" VaAvec="S034"> Brosse pour moulin à café</ITEM>
```

IDREFS : Comme IDREF, sauf que celui-ci peut inclure une référence à plusieurs identificateurs (séparés par des espaces)

ENTITY : valeur d'attribut correspondant au nom d'une entité non analysée déclarée dans la DTD (un fichier externe)
Exemple

```
<!ELEMENT IMAGE EMPTY>
<IMAGE Source ENTITY #REQUIRED>
…
<IMAGE Source="logo">
```

ENTITIES : Comme ENTITY sauf que celui-ci peut inclure une référence à plusieurs entités (séparées par des espaces)

NMTOKEN : la valeur est un token de noms, c'est-à-dire que vous pouvez utiliser 1 ou plusieurs lettres, chiffres, points, tirets et soulignés (également les double-points sauf à la première position). Le NMTOKEN peut commencer par un chiffre (pas les autres types !)

```
< !ELEMENT livre (#PCDATA)>
< ATTLIST livre ISBN NMTOKEN #REQUIRED>
```

NMTOKENS : Comme NMTOKEN sauf que celui-ci peut inclure une référence à plusieurs tokens de noms (séparés par des espaces)

- **type énuméré**
  - Les déclarations d'attributs peuvent aussi prendre la forme suivante :
    une parenthèse ouverte suivie par une liste de tokens de noms séparés par des « | » suivis d'une parenthèse.

Exemple :

```
<!ELEMENT LIVRE (TITRE, AUTEUR)>
<!ATTLIST LIVRE Genre (Fantastique | Policier | Humour) "Fantastique" >
<!ELEMENT TITRE (#PCADATA)>
<!ELEMENT AUTEUR (#PCADATA)>
```

Si on omet l'attribut genre, la valeur Fantastique est mise par défaut.

- Le mot-clé NOTATION[3] suivi d'un espace, puis d'une parenthèse puis d'une liste de noms de notations séparés par des « | » suivis d'une parenthèse.

Exemple :

```
<!ELEMENT EXEMPLE_DOCUMENT (#PCDATA)>
<!ATTLIST EXEMPLE_DOCUMENT Format NOTATION (HTML|SGML|RTF) #REQUIRED>
```

HTML, SGML et RTF doivent être déclarés dans votre DTD.

Remarque : si vous utilisez directement un style CSS dans un document XML,

```
<TITRE STYLE='font-style :normal ;font-size :14pt'>
        Tintin au congo
</TITRE>
```

Il faut déclarer l'attribut style dans la DTD :

```
<!ATTLIST TITRE STYLE CDATA #IMPLIED>
```

3.2 Exercices

1) Soit la DTD suivante :

```
<!DOCTYPE INVENTAIRE
[
<!ELEMENT INVENTAIRE(LIVRE)+>
<!ELEMENT LIVRE (TITRE, PRIX,(AUTEUR | DESSINATEUR | SCENARISTE))>
<!ELEMENT TITRE (#PCDATA | SOUSTITRE)*>
<!ELEMENT PRIX (#PCDATA)>
<!ELEMENT AUTEUR (#PCDATA)>
<!ELEMENT DESSINATEUR (#PCDATA)>
<!ELEMENT SCENARISTE (#PCDATA)>
<!ELEMENT SOUSTITRE (#PCDATA)>
]>
```

[3] Une notation décris le format de données ou identifie le programme utilisée pour traiter un format particulier.

Les éléments suivants sont-ils conformes à cette DTD ? Si non pourquoi ?

1

```
<INVENTAIRE>
<LIVRE>
        <TITRE>
                <SOUSTITRE>Et la magicienne trahie</SOUSTITRE>
        </TITRE>
        <PRIX>6,00€</PRIX>
        <AUTEUR>Jean Van Hamme</AUTEUR>
</LIVRE>
</INVENTAIRE>
```

2

```
<INVENTAIRE>
<LIVRE>
        <TITRE>
                Thorgal à la plage
        </TITRE>
        <PRIX>66,00€</PRIX>
        < SCENARISTE >Jean Van Hamme</ SCENARISTE >
</LIVRE>
<LIVRE>
        <TITRE>
                Thorgal à la mer
        </TITRE>
        <PRIX>65,00€</PRIX>
        < SCENARISTE >Jean Van Hamme</ SCENARISTE >
</LIVRE>
</INVENTAIRE>
```

3

```
<INVENTAIRE>
<LIVRE>
        <TITRE>
                <SOUSTITRE>Schmaracolpote</SOUSTITRE>
                <SOUSTITRE>#é&@ç)àç#{i#@{i{} »'(</SOUSTITRE>
        </TITRE>
        <PRIX>10,00€</PRIX>
        <AUTEUR>César Jules</AUTEUR>
</LIVRE>
</INVENTAIRE>
```

4

```
<INVENTAIRE>
<LIVRE>
        <TITRE>
                EICW – Brochure
        </LIVRE>
        <PRIX>Gratuit</PRIX>
        <DESSINATEUR>Coluche</DESSINATEUR>
</TITRE>
</INVENTAIRE>
```

2) Soit une série de DTD : les exemples qui suivent sont-ils conformes à la DTD ? Si non, que manque-t-il dans la DTD ?

A)

```
< !DOCTYPE INVENTAIRE (PERSONNE)*
[
<!ELEMENT INVENTAIRE (PERSONNE)*
<!ELEMENT PERSONNE (NOM+,AGE,TAILLE?,POIDS)*>
<!ELEMENT NOM (#PCDATA)>
<!ELEMENT AGE (#PCDATA)>
<!ELEMENT TAILLE (#PCDATA)>
<!ELEMENT POIDS (#PCDATA)>
]
>
```

```
<INVENTAIRE>
<PERSONNE>
        <NOM>Charles</NOM>
        <NOM>Martin</NOM>
        <AGE>42</AGE>
        <POIDS>50</POIDS>
</PERSONNE>
</INVENTAIRE>
```

```
<INVENTAIRE>
<PERSONNE>
        <NOM>Charles</NOM>
<AGE>22</AGE>
<TAILLE>1m70<TAILLE>
<POIDS>65</POIDS>
</PERSONNE>
<PERSONNE>
<AGE>22</AGE>
<TAILLE>1m70<TAILLE>
<POIDS>65</POIDS>
</PERSONNE>
</INVENTAIRE>
```

B)

```
< !DOCTYPE FILM
[
< !ELEMENT FILM (ACTEUR* | REALISATEUR | ASSISTANT)>
<!ELEMENT ACTEUR (#PCDATA)>
<!ELEMENT REALISATEUR (#PCDATA)>
<!ELEMENT ASSISTANT (#PCDATA)>
]>
```

```
<FILM>
        <ACTEUR>Tom Hanks</ACTEUR>
        <ACTEUR>Mel Gibson</ACTEUR>
        <REALISATEUR>Dany Boyle<REALISATEUR>
</FILM>
```

```
<FILM>
        <ACTEUR>Tom Hanks</ACTEUR>
        <ACTEUR>Mel Gibson</ACTEUR>
</FILM>
```

```
<FILM>
        <REALISATEUR>Dany Boyle<REALISATEUR>
        <ASSISTANT>Dany Boyle<ASSISTANT>
</FILM>
```

```
<FILM>
        <ACTEUR>Tom Hanks</ACTEUR>
        <ACTEUR>Mel Gibson</ACTEUR>
</FILM>
<FILM>
        <ACTEUR>Brad Pitt</ACTEUR>
</FILM>
```

3) **[inventaire_dtd.xml]** Ajoutez la DTD au document suivant pour le rendre valide :

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

<!-- Nom du fichier : Inventaire.xml-->

<INVENTAIRE>
 <LIVRE EnStock="oui">
  <TITRE>La Magicienne trahie</TITRE>
  <AUTEUR Naissance="1967">Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>48</PAGES>
  <PRIX>9,00&#8364;</PRIX>
 </LIVRE>
 <LIVRE EnStock="oui">
  <TITRE>L'&#238;le des Mers gel&#233;es</TITRE>
  <AUTEUR Naissance="1967">Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>48</PAGES>
  <PRIX>9,50&#8364;</PRIX>
 </LIVRE>
 <LIVRE EnStock="non">
  <TITRE>L'&#233;p&#233;e soleil</TITRE>
  <AUTEUR Naissance="1967">Jean Van Hamme</AUTEUR>
  <EDITEUR>Le Lombard</EDITEUR>
  <PAGES>50</PAGES>
  <PRIX>9,50&#8364;</PRIX>
 </LIVRE>
 <LIVRE EnStock="oui">
  <TITRE>Les bleus de la marine</TITRE>
  <AUTEUR>LAMBIL/CAUVIN </AUTEUR>
  <EDITEUR>Dupuis</EDITEUR>
  <PAGES>45</PAGES>
  <PRIX>10,00&#8364;</PRIX>
 </LIVRE>
</INVENTAIRE>
```

3) **[traducteur_dtd.xml]** Créez une DTD pour votre document « traducteur.xml »
4) **[listedvd_dtd.xml]** Créez une DTD pour votre document « listedvd.xml »
5) Ajoutez l'attribut "acteur" aux champs "titre", et modifiez la dtd de l'exercice précédent.

Le DTD externe suivra la syntaxe suivante :

```
<!DOCTYPE élément-racine SYSTEM "nom_du_fichier.dtd">
```

Vous pouvez faire en sorte d'ignorer certaines parties de votre DTD en utilisant une section IGNORE

```
<![IGNORE[
<!--bloc d'option désactivé temporairement-->
<!ATTLIST LIVRE Enstock (oui|non) #REQUIRED >
<!ELEMENT TITRE (#PCDATA)>
]]>
```

Remarque :

- Il existe beaucoup de logiciels et de sites Internet qui créeront pour vous la DTD à partir de votre fichier XML. (XMLSPY)

3.3 XML Schema Definition (XSD)

Pour rendre un document valide, nous avons vu les DTD, malheureusement celles-ci sont limitées :

- pas de support des namespaces
- syntaxe non XML
- modularité assez limitée
- pas de contrainte pour les données textes et attributs.
- Actuellement il existe deux grandes technologies (d'autres, moins connues, existent également) :

- **Les schémas du W3C** (XSD) (recommandation du 2 mai 2001)
  + norme officielle
  + modèle de contenu très évolué et très riche
  - verbeux
  - très complexe

Remarque : Les schémas doivent obligatoirement être définis dans un document externe.

3.3.1 Exemple de base

Prenons un exemple d'un fichier test.xml qui contient un élément <a> et un nombre arbitraire de <b> vides : (« personne.xml »)

```
<?xml version="1.0" encoding="UTF-8"?>
  <personne xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personne.xsd">
    <nom>MBODJ</nom>
    <prenom>Babacar</prenom>
    <date_naissance>1996-10-06</date_naissance>
    <etablissement>NIIT</etablissement>
    <num_tel>764704140</num_tel>
  </personne>
```

Le schéma W3C correspondant est : (« personne.xsd »)

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="personne">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="nom" type="xs:string" />
          <xs:element name="prenom" type="xs:string" />
          <xs:element name="date_naissance" type="xs:date" />
          <xs:element name="etablissement" type="xs:string" />
          <xs:element name="num_tel" type="xs:string" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Remarque :
Pour utiliser notre fichier schéma W3C au lieu de notre DTD, il faudra préciser l'URI du schéma utilisé :

```
<?xml version="1.0"?>
<a xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personne.xsd">
<a><b/><b/></a>
```

3.3.2 Règles de base

Les schémas W3C sont basés sur la notion de type :

- **types simples (simpleType) :**

Chaînes de caractères, valeurs numériques, etc…(uniquement élément feuille et sans attribut)

- **types complexes (complexType) :**

Tout le reste

Pour définir un type **élément,** il faut le faire soit directement au niveau global, soit dans la définition d'un type complexe, soit avec un élément *element* possédant les attributs suivants :

- *name* (nom de l'élément)
- *type* (type de l'élément)
- *fixed* (valeur de l'élément fixée et précisée)
- *default* (valeur par défaut quand l'élément est présent mais avec un contenu vide)
- Pour définir un **attribut,** il faut le faire soit directement au niveau global, soit dans la définition d'un type complexe, soit avec un élément *attribut* possédant les attributs suivants :
- *name* (nom de l'attribut)
- *type* (type de l'attribut, doit être un type simple)
- *fixed* (valeur de l'attribut fixée et précisée)
- *default* (valeur par défaut)

1. Types simples :
De nombreux types simples sont disponibles :

- string (remplace #PCDATA)
- byte, int
- float, double
- dates et heures
- URI
- Etc.

2. Types dérivés :
On peut restreindre chaque type grâce à des *facets* :

- Valeurs minimales, maximales

```
<xsd:simpleType name="petitEntier">
  <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="100"/>
  </xsd:restriction>
</xsd:simpleType>
```

- Nombre de chiffres
- Expressions régulières

```
<xsd:simpleType name="telephone10">
  <xsd:restriction base="xsd:string">
        <xsd:pattern value="(\d{2} ){4}\d{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

- Enumération des valeurs possibles

```
<xsd:restriction base="xsd:string">
        <xsd:enumeration value="portable"/>
        <xsd:enumeration value="fixe"/>
        <xsd:enumeration value="fax"/>
</xsd:restriction>
```

- Etc.

3. Types complexes

Un type complexe se définit comme suit :

- Elément sequence
- Elément *element* pour les fils exigés
  - Attribut *ref* (permet d'utiliser un élément déjà défini)
  - Attributs *minOccurs* et *maxOccurs* (nombre d'occurrence)
  - [Valeur par défaut]
- contenu mixte (mixed ="true")
- attribut
  - Attribut ref (permet d'utiliser un élément déjà défini)
  - Attribut use (précise si l'attribut est optional, prohibited, required)
  - [Valeur par défaut]

Exemple pour notre fichier « personne.xml » (personne.xsd)

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="liste">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="personne" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        </xs:element>
        <xs:element name="personne">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="prénom"/>
                        <xs:element ref="nom"/>
                </xs:sequence>
        </xs:complexType>
        </xs:element>
</xs:schema>
```

Les schémas W3C permettent d'introduire des commentaires (annotations), de créer des groupes nommés d'éléments (macro, modèles, etc…)
Les schémas W3C permettent aussi la définition des *namespaces* , de construire un schéma à partir d'autres schémas, de faire des références croisées, etc…

Pour plus d'informations concernant les schémas W3C :
http://www.w3.org/XML/Schema

3.3.3 Exercices

1) **[traducteur.xsd]** Créez le schéma XSD pour votre document « traducteur.xml »
2) **[listedvd.xsd]** Créez le schéma XSD pour votre document « listedvd.xml »

4. Affichage d'un document XML avec une feuille de style

Comme pour le langage HTML, pour lier une feuille de style à un document XML, il faut inclure celui-ci comme suit :

```
<?xml-stylesheet type="text/css" href="inventaire.css"?>
```

inventaire.css :

```
LIVRE
        {display:block;
        margin-top:12pt;
        font-size:10pt;}
TITRE
        {font-style:italic;}
AUTEUR
        {font-weight:bold;}
EDITEUR
        {display:block;
        font-size:12pt;
        font-weight:bold;}
PAGES
        {display:none;}
PRIX
        {}
```

Dans le fichier “inventaire.css”, nous avons défini les éléments comme ci-dessous :

LIVRE
- 12 points d’espace libre au-dessus de lui (margin-top :12pt)
- Césure de ligne au-dessus et en-dessous (display : block)
- Taille de police de 10 points (font-size : 10pt)

TITRE
- Elément en italique (font-style : italic)

AUTEUR
- Elément en gras (font-weight : bold)

EDITEUR
- Césure de ligne au-dessus et en-dessous (display : block)
- Taille de police de 12 points (font-size : 12pt)
- Elément en gras (font-weight : bold)

PAGE
- Ne pas afficher cet élément (display :none)

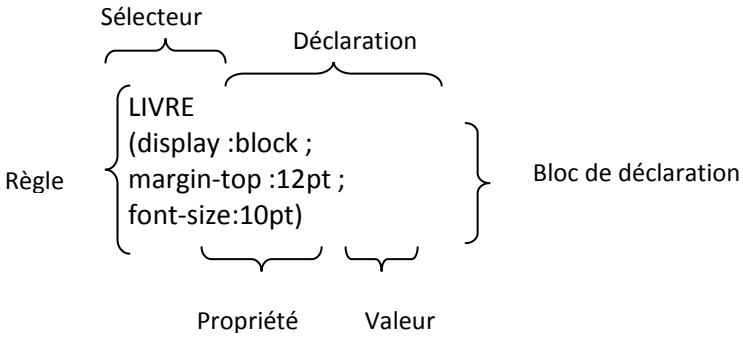
PRIX
- Normal

Son affichage dans le navigateur donnera :

Il faut savoir que les CSS ne vous autorise pas à accéder aux attributs. Pour accéder à ceux-ci, vous devrez utiliser soit les liaisons avec un document HTML (voir point 6) soit avec un langage de script (voir point 7).

Les feuilles de styles CSS ne permettent pas de **manipuler les éléments**, de transformer un document XML source en un autre ainsi, il est déconseillé pour le formatage des documents XML, nous verrons dans le chapitre suivant qu'il est moins puissant que **XSL**, néanmoins, le XSL aura besoin du CSS pour la mise en forme.

4.1 Petit rappel sur les règles des feuilles de style



Les commentaires se font entre **/*** et ***/**
Le CSS est insensible à la casse.

Exemple : création d'une marge dans un document : **MARGE.CSS**

```
ARTICLE
        {font-size:12pt;}
```

```
TITRE
        {font-size:36pt;}
ARTICLE,LIGNE,TITRE,TEXTE,MARGE,ADRESSE_INTERNET
        {
        display:block;
        }
MARGE
        {
        color:red;
        border-style:solid;
        border-width:2px;
        text-align:center;
        width:2in;
        height:1in;
        float:left;
        }
ADRESSE_INTERNET
        {
        color:blue;
        font-style:italic;
        }
```

MARGE.XML

```
<?xml version="1.0"?>
<!-- Nom du fichier : marge.xml-->
<?xml-stylesheet type="text/css" href="marge.css"?>
<ARTICLE>
<TITRE>Thorgal</TITRE>
<MARGE>
Le th&#232;me des univers parall&#232;les rapport&#233; aux viking.
Thorgal, seul rescap&#233; d'un vaisseau spatial echoue sur Terre,
et est recueilli par les vikings.
</MARGE>
<TEXTE>
<LIGNE>1-2 : Face &#224; la magicienne</LIGNE>
<LIGNE>3 : vivre eternellement</LIGNE>
<LIGNE>4-5-6 : Contre Shardar le Mal&#233;fique</LIGNE>
<LIGNE>7 et 14 : albums cl&#233;s, tout sur Thorgal et Aaricia</LIGNE>
<LIGNE>8 : Jolan et l'enfant cauchemar</LIGNE>
<LIGNE>9-10-11-12-13 : Entre les faux Dieux</LIGNE>
<LIGNE>15 : quand le temps devient fou !</LIGNE>
<LIGNE>16 : petite fille n&#233;e avec les loups</LIGNE>
<LIGNE>17 : Thorgal face &#224; Thorgal</LIGNE>
<LIGNE>18-19-20-21-22-23 : Dans les griffes de Kriss</LIGNE>
</TEXTE>
<ADRESSE_INTERNET>
http://cyberpingui.free.fr/bd/_detail_albums.php?idserie=142
</ADRESSE_INTERNET>
</ARTICLE>
```

Dans le navigateur vous obtiendrez :

Thorgal

Le thème des univers parallèles rapporté aux Vikings. Thorgal, seul rescapé d'un vaisseau spatial échoue sur Terre, et est recueilli par les Vikings.

1-2 : Face à la magicienne
3 : Vivre éternellement
4-5-6 : Contre Shardar le Maléfique
7 et 14 : Albums clés, tout sur Thorgal et Aaricia
8 : Jolan et l'enfant cauchemar
9-10-11-12-13 : Entre les faux Dieux
15 : Quand le temps devient fou !
16 : Petite fille née avec les loups
17 : Thorgal face à Thorgal
18-19-20-21-22-23 : Dans les griffes de Kriss
24 : Quand sa vie ne tient qu'à un fil
25 : Sauvera-t-il sa famille du terrible mal ?
26 : Les origines de Thorgal
http://cyberpingui.free.fr/bd/_detail_albums.php?idserie=142

Vous pouvez aussi ajouter une image dans une zone flottante :
Ajoutez dans le fichier marge.css:

```
IMAGE
        {
        background-image:url(thorgal.jpg);
        background-repeat:no-repeat;
        background-position:right;
        width:200px;
        height:52px;
        float:right;
        }
```

Et ajoutez l'élément <IMAGE/> devant l'élément <TITRE> dans le fichier marge.xml
Vous obtenez bien l'image en haut à droite comme précisée dans le CSS :

Thorgal

Le thème des univers parallèles rapporté aux Vikings. Thorgal, seul rescapé d'un vaisseau spatial échoue sur Terre, et est recueilli par les Vikings.

1-2 : Face à la magicienne
3 : Vivre éternellement
4-5-6 : Contre Shardar le Maléfique
7 et 14 : Albums clés, tout sur Thorgal et Aaricia
8 : Jolan et l'enfant cauchemar
9-10-11-12-13 : Entre les faux Dieux
15 : Quand le temps devient fou !
16 : Petite fille née avec les loups
17 : Thorgal face à Thorgal
18-19-20-21-22-23 : Dans les griffes de Kriss
24 : Quand sa vie ne tient qu'à un fil
25 : Sauvera-t-il sa famille du terrible mal ?
26 : Les origines de Thorgal
http://cyberpingui.free.fr/bd/_detail_albums.php?idserie=142

Insertion d'éléments HTML dans les documents XML

Si vous voulez utiliser un tag HTML, il vous faudra passer par une déclaration d'espace de nom[4] réservé qu'on déclare comme suit :

```
xmlns:html='http://www.w3c.org/TR/REC-html40/'
```

Par exemple pour insérer directement l'image dans le fichier XML

```
<html:IMG xmlns:html='http://www.w3c.org/TR/REC-html40/' SRC='thorgal.jpg'/>
```

Ou encore pour que le lien soit cliquable:

```
<html:A xmlns:html='http://www.w3c.org/TR/REC-html40/'
HREF='http://cyberpingui.free.fr/bd/_detail_albums.php?idserie=142'>
http://cyberpingui.free.fr/bd/_detail_albums.php?idserie=142
</html:A>
```

Héritage des paramètres

Une définition de propriété attribuée à un élément particulier affecte tous les éléments enfants insérés directement ou indirectement sauf si elle est écrasée par une propriété définie à l'élément enfants.

Mais les propriétés suivantes **ne** sont **pas** héritées :

- display (sauf pour l'élément none)
- les propriétés du fond (*background-color, background-image,backgroud-repeat et background-position*)
- *vertical-align*
- les propriétés de cadres

4.2 Exercices

1) **[inventaire2.xml] - [inventaire2.css]** A partir de votre document « inventaire.xml », créez un document « inventaire2.xml », affichez l'élément LIVRE avec 8 points d'espace libre au-dessus de lui, dans une police « Arial », césure de ligne au-dessus et une taille de police de 10 points. Affichez les éléments TITRE, AUTEUR, EDITEUR,PAGE chacun sur une ligne séparée en « Verdana ». Indenter les éléments AUTEUR et EDITEUR de 10 points. Affichez l'élément TITRE en gras et affriolez-le d'une couleur bleu. Mettez l'élément EDITEUR en italique et soulignez-le. Affichez le nombre de page en vert. Ne pas afficher le prix.
2) **[traducteur.xml] - [traducteur.css]** Créez un fichier CSS pour votre document « traducteur.xml »
3) **[listedvd2.xml] - [listedvd.css]** Créez un fichier CSS pour votre document « listedvd.xml »

5. Les feuilles de styles XSL

Le XSL (Extensible Style Language) est le langage utilisé pour définir les feuilles de style qui seront associées aux documents XML.
Cette abréviation recouvre en fait trois langages :

- **XPath** désigne un moyen d'accéder à un nœud quelconque de l'arborescence d'un document XML à partir d'un autre nœud quelconque → le langage de navigation dans un document XML.
- **XSLT** signifie eXtensible Stylesheet Langage Transformation → il permet de manipuler les données et de les réorganiser

[4] Imaginons que nous voulons générer du SVG (graphisme vectoriel) avec Xslt. Les deux langages ont une balise <text/>. Or, comment le programme qui traitera notre document pourra savoir que ce <text/>-ci est un élément SVG et que celui-là appartient au langage Xslt ? C'est pour éviter ce genre de problème qu'on appelle collision de nom que les namespaces ont été inventés. Et donc, Le mécanisme de namespace, qu'on peut traduire par "espace de nom", ou "domaine nominal", est une notion assez abstraite : C'est une chaîne de caractères (une adresse internet, mais elle n'est pas lue) qui sert à identifier le langage auquel appartient la balise, ou l'attribut qui fait partie de ce namespace. En conclusion : les espaces de noms (***namespaces***) permettent d'introduire des collections de noms utilisables pour les éléments et attributs d'un document XML Chaque collection est identifiée par un URI (Uniform Resource Identifier) dans l'exemple :
html='http://www.w3c.org/TR/REC-html40/'

XSL-FO signifie eXtensible Stylesheet Langage - Formating Objects, et désigne un langage permettant le contrôle de la mise en page finale de la transformation. Ce langage est particulièrement destiné à la production de contenus au format PDF.

Les feuilles de styles XSL permettent un contrôle plus complet de l'affichage d'un document XML. (Modèle de transformation). Une feuille de style XSL est un fichier qui décrit comment doivent être présentés (c'est-à-dire affichés, imprimés, épelés) les documents XML basés sur une même DTD ou un même schéma.

5.1 Principes de base

5.1.1 Principe de fonctionnement

Le fonctionnement du XSL est fondé sur les manipulations de modèles (templates). Les éléments du document XML d'origine sont remplacés (ou plus ou moins légèrement modifiés) par ces modèles. Un modèle contient ainsi le texte (éléments, attributs, texte...) de remplacement d'un élément donné.

Tout élément pouvant être remplacé dans le fichier de sortie par tout type de contenu texte, XSL est un outil privilégié de production de fichiers HTML à partir de sources XML. PHP fait ainsi appel à des bibliothèques de procédures de type XSL comme libxslt quand il doit gérer l'interfaçage avec des bases de données XML.

Un fichier XSL étant un fichier XML, il doit respecter les normes de syntaxe de ce format.

Une feuille de styles XSLT est composée d'une suite de règles appelées "template rules" (ou règles de gabarit en français).Le processeur XSLT (composant logiciel chargé de la transformation) crée une structure logique arborescente (on parle d'arbre source) à partir du document XML. Il lui applique ensuite des transformations selon les template rules contenues dans la feuille XSLT pour produire un arbre résultat représentant, par exemple, la structure d'un document HTML. Chaque "template rule" définit des traitements à effectuer sur un élément (noeud ou feuille) de l'arbre source. On appelle "patterns" (en français motifs, parfois "éléments cibles") les éléments de l'arbre source. L'arbre source peut être entièrement remodelé et filtré, si bien que l'arbre résultat peut être radicalement différent de l'arbre source.

5.1.2 Principales instructions XSLT

<xsl:stylesheet>

Déclaration de la feuille de style. Tout ce qui se situe à l'intérieur de cette balise relève le la logique de la feuille de style.

<xsl:output method='html'/>

L'élément <xsl:output> indique le format de sortie du document résultant. Il est à placer immédiatement après l'élément <xsl:stylesheet>

<xsl:template match="/">

L'élément <xsl:template> permet de définir une règle de modèle. Ici, par l'intermédiaire de l'attribut 'match' c'est à la racine ('/') du document que l'on applique des règles de style.

<xsl:apply-templates/>

L'élément <apply-templates> permet d'appliquer les modèles (templates) d'une feuille de style sur les fils et les noeuds de type texte du noeud courant.

<xsl:call-template name="...">

L'élément <xsl:call-template> permet d'appeler un modèle (template) directement par son nom.

<xsl:value-of select="..."/>

L'élément <xsl:value-of> permet d'extraire la valeur d'un noeud sélectionné via l'attribut 'select'.

Dans notre fichier « thorgal.xml », nous devons lier celui-ci avec un fichier xsl (comme pour les fichiers css).

```
<?xml-stylesheet type="text/xsl" href="thorgal.xsl"?>
```

Un fichier xsl, est un fichier xml, c'est-à-dire qu'il doit se conformer aux règles pour qu'un fichier xml soit bien formé (par exemple; si on veut utiliser un tag html
 il faudra ajouter </BR> ou uniquement
).

5.1.3 value of select

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- nom de fichier bd2.xsl -->

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
	<html>
	<head>
	<title>Exemple value-of select</title>
	<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
	</head>
	<body>

	<h2>Mes Livres </h2>
	<SPAN style="font-style:italic">Auteur : </SPAN>
	<xsl:value-of select="INVENTAIRE/LIVRE/AUTEUR"/><BR/>
	<SPAN style="font-style:italic">Titre : </SPAN>
	<xsl:value-of select="INVENTAIRE/LIVRE/TITRE"/><BR/>
	<SPAN style="font-style:italic">Editeur : </SPAN>
	<xsl:value-of select="INVENTAIRE/LIVRE/EDITEUR"/><BR/>
	<SPAN style="font-style:italic">Page : </SPAN>
	<xsl:value-of select="INVENTAIRE/LIVRE/PAGES"/><BR/>
	</body>
	</html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
→ C'est l'espace de noms qui contient les normes XSL

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
→ Permet d'utiliser/d'interpréter les tags HTML & caractère correspondant à la norme ISO-8859-1

<xsl:template match="/">
→ « / » représente la racine du document XML (c'est comme un adressage relatif, on doit faire la référence à un élément en lui donnant son chemin complet : INVENTAIRE/LIVRE/AUTEUR)
→ Le premier avantage de l'xsl est qu'on peut afficher les éléments dans l'ordre de son choix.

Notre fichier xml avec sa liaison xsl tel qu'il est actuellement n'affichera que le premier élément.

5.1.4 *for each select*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- nom de fichier bd2.xsl -->

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
        <html>
        <head>
        <title>Exemple value-of select /for-each</title>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
        </head>
        <body>

        <h2>Mes Livres </h2>
        <xsl:for-each select="INVENTAIRE/LIVRE">
        <SPAN style="font-style:italic">Auteur : </SPAN>
        <xsl:value-of select="AUTEUR"/><BR/>
        <SPAN style="font-style:italic">Titre : </SPAN>
        <xsl:value-of select="TITRE"/><BR/>
        <SPAN style="font-style:italic">Editeur : </SPAN>
        <xsl:value-of select="EDITEUR"/><BR/>
        <SPAN style="font-style:italic">Page : </SPAN>
        <xsl:value-of select="PAGES"/><BR/>
        <hr/>
        </xsl:for-each>
        </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

→ L'élément courant dans la « boucle » for-each n'est plus l'élément racine « / », mais bien INVENTAIRE/LIVRE.

5.1.5 *apply-templates select*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- nom de fichier bd3.xsl -->

<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>

        <h2>Mes Livres </h2>
        <xsl:apply-templates select="INVENTAIRE/LIVRE"/>
        </body>
        </html>
</xsl:template>

   <xsl:template match="LIVRE">
        <SPAN style="font-style:italic">Auteur : </SPAN>
        <xsl:value-of select="AUTEUR"/><BR/>
        <SPAN style="font-style:italic">Titre : </SPAN>
        <xsl:value-of select="TITRE"/><BR/>
        <SPAN style="font-style:italic">Editeur : </SPAN>
        <xsl:value-of select="EDITEUR"/><BR/>
        <SPAN style="font-style:italic">Page : </SPAN>
        <xsl:value-of select="PAGES"/><BR/>
        <hr/>
   </xsl:template>

</xsl:stylesheet>
```

Le « apply-templates » permet d'être récursif, si dans l'élément sous menu il y a un sous menu, on peut réutiliser le template créé pour le premier sous-menu.

menu.xml

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href="menu.xsl"?>
<menu>
        <sousmenu>
                <commande>abricos</commande>
                <sousmenu>
                        <commande>pomme</commande>
                </sousmenu>
        </sousmenu>
        <sousmenu>
                <commande>orange</commande>
        </sousmenu>
        <sousmenu>
                <commande>banane</commande>
        </sousmenu>
</menu>
```

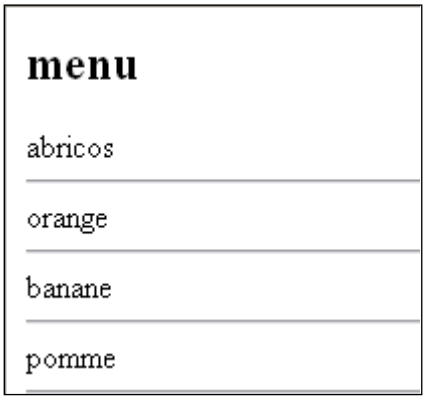
menu.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- nom de fichier bd3.xsl -->

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
        </head>
        <body>
        <h2>menu </h2>
        <xsl:apply-templates select="menu/sousmenu"/>
        <xsl:apply-templates select="menu/sousmenu/sousmenu"/>
        </body>
</html>

</xsl:template>
   <xsl:template match="sousmenu">
                <xsl:value-of select="commande"/><BR/>
                <hr/>
   </xsl:template>
</xsl:stylesheet>
```

Résultat :



5.1.6 sort select

```
<xsl:sort
        select="pattern"
        lang="langue"
        data-type="text|number|nom"
        order="ascending|descending"
        case-order="upper-first|lower-first"/>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- nom de fichier bd2.xsl -->
<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>
        <h2>Mes Livres </h2>
        <xsl:for-each select="INVENTAIRE/LIVRE">
        <xsl:sort select="TITRE" order="ascending"/>
        <SPAN style="font-style:italic">Auteur : </SPAN>
        <xsl:value-of select="AUTEUR"/><BR/>
        <SPAN style="font-style:italic">Titre : </SPAN>
        <xsl:value-of select="TITRE"/><BR/>
        <SPAN style="font-style:italic">Editeur : </SPAN>
        <xsl:value-of select="EDITEUR"/><BR/>
        <SPAN style="font-style:italic">Page : </SPAN>
        <xsl:value-of select="PAGES"/><BR/>
        <hr/>
        </xsl:for-each>
        </body>
        </html>
 </xsl:template>
</xsl:stylesheet>
```

```
<xsl:sort select="TITRE" order="ascending"/>
<xsl:sort select="EDITEUR" order="ascending"/>
```

Affichera les éléments classés par ordre alphabétique (croissant) de titres puis d'éditeurs

```
<xsl:sort select="TITRE" order="descending"/>
<xsl:sort select="EDITEUR" order="ascending"/>
```

Affichera les éléments classés par ordre alphabétique (décroissant) de titres puis d'éditeurs (croissant).

5.1.7 if test="expression"

```
<xsl:if test="expression">
  ...
</xsl:if>
```

```
<?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>
 <body>
  <h2>Mes BD's</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th>Titre</th>
    <th>Auteur</th>
   </tr>
   <xsl:for-each select="INVENTAIRE/LIVRE">
   <xsl:if test="PAGES > 45">
    <tr>
     <td><xsl:value-of select="TITRE"/></td>
     <td><xsl:value-of select="AUTEUR"/></td>
    </tr>
   </xsl:if>
   </xsl:for-each>
  </table>
```

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

5.1.8 choose

```
<xsl:choose>
 <xsl:when test="expression">
  ...
 </xsl:when>
 <xsl:otherwise>
  ...
 </xsl:otherwise>
</xsl:choose>
```

```
<?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>
 <body>
  <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
   </tr>
   <xsl:for-each select="INVENTAIRE/LIVRE">
   <tr>
    <td><xsl:value-of select="TITRE"/></td>
        <xsl:choose>
      <xsl:when test="AUTEUR = 'Y. Sente'">
       <td bgcolor="#ff00ff">
       <xsl:value-of select="AUTEUR"/></td>
      </xsl:when>
      <xsl:otherwise>
       <td><xsl:value-of select="AUTEUR"/></td>
      </xsl:otherwise>
     </xsl:choose>
   </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

5.1.9 Commentaires

Les commentaires existent aussi en XSL

```
<xsl:comment>
Commentaires
</xsl:comment>
```

5.1.10 Transformation XML → HTML

```
<!-- Insérer une image GIF avec le titre comme nom -->
<img SRC="{TITRE}.gif" align="left" height="220"/>
<h1>
<i><xsl:value-of select="TITRE"/></i>
</h1>
```

```
<!-- Insérer une adresse Email dans un lien mailto -->
<A HREF="mailto:{EMAIL}" ><xsl:value-of select="EMAIL"/></A>
```

5.2 XPATH

XPath est un langage d'expressions qui permet de sélectionner des éléments ou des groupes d'éléments au sein d'un document XML. Il est notamment exploité par XSLT lors de la procédure de transformation d'un document XML.

Le point de départ d'Xpath est la modélisation d'un document XML sous la forme d'un arbre composé d'un ensemble de noeuds de plusieurs types :

- un noeud racine (root node), noté " / ", qui représente l'entièreté du document;
- des noeuds éléments (pour chaque élément du document);
- des noeuds attributs (pour chaque attribut au sein d'un élément);
- des noeuds textes (pour chaque données textuelles ou la valeur d'un attribut);
- des noeuds commentaires (pour chaque commentaire dans le document);
- des noeuds instructions de traitement (pour chaque instruction de traitement dans le document);
- des noeuds namespaces (pour chaque namespace utilisé dans le document).

Cette modélisation va être exploitée par XPATH afin de rendre possible la sélection de sous-ensembles de noeuds. Sans cette possibilité de " ciblage " aucun traitement conséquent de l'information XML n'est possible.

5.2.1 Les expressions XPATH

XPATH est formulé et utilisé sous forme d'expressions. Ces expressions, après avoir été évaluées, peuvent fournir en réponse les éléments suivants :

- un ensemble de noeuds;
- un booléen;
- un nombre;
- une chaîne de caractères.

Les expressions Xpath servent en fait à définir ce que l'on appelle couramment des "patterns", c'est-à-dire des chaînes de caractères permettant de repérer un noeud dans un document XML (on peut faire le parallèle avec la technique des expressions régulières). Voici quelques exemples de patterns.

nom

Exemple : livre description : sélectionne des éléments en fonction de leur nom (ici 'livre')

nom[ ]

Exemple : livre[1] description : sélectionne un élément ayant un nom donné en fonction de son ordre d'apparition (ici le premier élément 'livre')

nom[end]

Exemple : livre[end] description : sélectionne le dernier élément ayant un nom donné (ici le dernier élément 'livre')

|

Exemple : livre|revue description : Indique une alternative (l'élément 'livre' ou l'élément 'revue')

/

Exemple : / description : utilisé seul, ce pattern définit l'élément de plus haut niveau de l'arborescence ainsi que (de façon implicite) tous ses éléments fils. exemple : bibliographie/livre description : utilisé entre deux éléments, il décrit la localisation d'un élément dans son arborescence

*****

Exemple : * description : désigne n'importe quel élément

//
Exemple : //livre description : désigne tous les descendants d'un noeud
.
Exemple : . description : désigne le noeud courant
..
Exemple : .. description : désigne le noeud parent
@
Exemple : @titre description : désigne un attribut (la notation @* désigne tous les attributs d'un élément)
text()
Exemple : text() description : désigne le contenu d'un élément (le texte contenu entre ses balises)

5.2.2 Les fonctions XPATH

- **boolean**() : évaluer l'argument comme vrai ou faux
- **ceiling**() : arrondir un nombre
- **concat**() : concaténer des chaînes de caractères
- **contains**() : vérifier certaines sous-chaînes de caractères
- **count**() : rechercher le nombre de nœuds dans un jeu de nœuds
- **current**() : rechercher le nœud actuel
- **document**() : appeler un autre document XML
- **element-available**() : vérifier si un élément XSLT est disponible
- **false**() : créer une valeur booléenne pour "faux":
- **floor**() : arrondir un nombre par défaut
- **format-number**() : transformer un nombre en chaîne de caractères
- **function-available**() : vérifier si une fonction XPath est disponible
- **generate-id**() : générer un identificateur clair pour un élément
- **id**() : choisir un élément avec un identificateur déterminé
- **key**() : choisir un élément grâce à une valeur-clé
- **lang**() : vérifier un code de langue précis pour un élément
- **last**() : rechercher le numéro de position du dernier nœud d'une série
- **local-name**() : rechercher le nom local d'un jeu de nœuds
- **name**() : rechercher le nom d'un jeu de nœuds
- **namespace-uri**() : rechercher l'espace de nommage d'un élément
- **normalize-space**() : retirer les espaces au début et à la fin d'une chaîne de caractères
- **not**() : nier un argument
- **number**() : transformer un argument en nombre
- **position**() : rechercher le numéro de position du jeu de nœuds actuel
- **round**() : arrondi par excès ou par défaut commercial
- **starts-with**() : vérifier si une chaîne de caractères commence par une certaine sous-chaîne
- **string**() : transformer un argument en chaîne de caractères
- **string-length**() : retourne le nombre de caractère de la chaine de caractères
- **substring**() : extraire une sous-chaîne d'une chaîne de caractères
- **substring-after**() : rechercher une sous-chaîne à partir d'une certaine position dans une chaîne de caractères
- **substring-before**() : rechercher une sous-chaîne avant une certaine position dans une chaîne de caractères
- **sum**() : rechercher la somme de valeurs numériques
- **system-property**() : rechercher des propriétés système
- **translate**() : rechercher et remplacer des signes dans une chaîne de caractères
- **true**() : créer une valeur booléenne pour "vrai"
- **unparsed-entity-uri**() : rechercher les éléments de la DTD non vérifiés par l'analyseur syntaxique

5.2.3 Exemples d'expressions XPATH, utilisation des filtres

Si on veut sélectionner l'ensemble des noeuds <TITRE> qui sont enfants du noeud <LIVRE> qui sont eux même enfant de l'élément <INVENTAIRE> (l'élément racine du document), on écrira :

```
INVENTAIRE/LIVRE/TITRE
```

Si on veut sélectionner tous les noeuds <LIVRE> dont le titre est « La Magicienne trahie », on écrira :

```
LIVRE[TITRE='La Magicienne trahie']
```

Si on veut sélectionner tous les noeuds <LIVRE> dont l'auteur est « Jean Van Hamme », on écrira :

```
INVENTAIRE/LIVRE [AUTEUR='Jean Van Hamme']
```

Si on veut sélectionner tous les noeuds <LIVRE> dont l'auteur **n'**est **pas** « Jean Van Hamme », on écrira :

```
INVENTAIRE/LIVRE [AUTEUR!='Jean Van Hamme']
```

Si on veut sélectionner tous les noeuds <TITRE> dont la publication est de 1983, on écrira :

```
INVENTAIRE/LIVRE/TITRE[@publication='1983']"/>
```

Exemple d'utilisation de fonction :

```
position()<10
```

```
<xsl:if test="starts-with(.,'A')">
  <xsl:value-of select="value" />
</xsl:if>
```

```
<xsl:choose>
  <xsl:when test="string-length() < 20">
    trop court...
  </xsl:when>
  <xsl:when test="string-length() > 200">
    trop long...
  </xsl:when>
  <xsl:otherwise>
    <xsl:value-of select="." /></p>
  </xsl:otherwise>
</xsl:choose>
```

```
<xsl:value-of select="substring-before(current(),'=')"/>
<xsl:value-of select="substring-after(current(),'=')"/>
```

```
<xsl:value-of select="translate(.,
   'abcdefghijklmnopqrstuvwxyz','ABCDEFGHIJKLMNOPQRSTUVWXYZ')" />
```

5.2.4 Les axes de recherche

Les expressions XPATH peuvent faire référence à des axes de recherche. Ces axes de recherche précisent dans quel sens il faut parcourir l'arbre afin de sélectionner des noeuds.

child::

Clause sollicitant les enfants du noeud contexte - c'est l'axe par défaut.

descendant::

Clause sollicitant les descendants du noeud contexte (les enfants, les enfants des enfants, etc.).

cdescendant-or-self::

Clause sollicitant les descendants du noeud contexte et le noeud contexte lui-même.

parent::

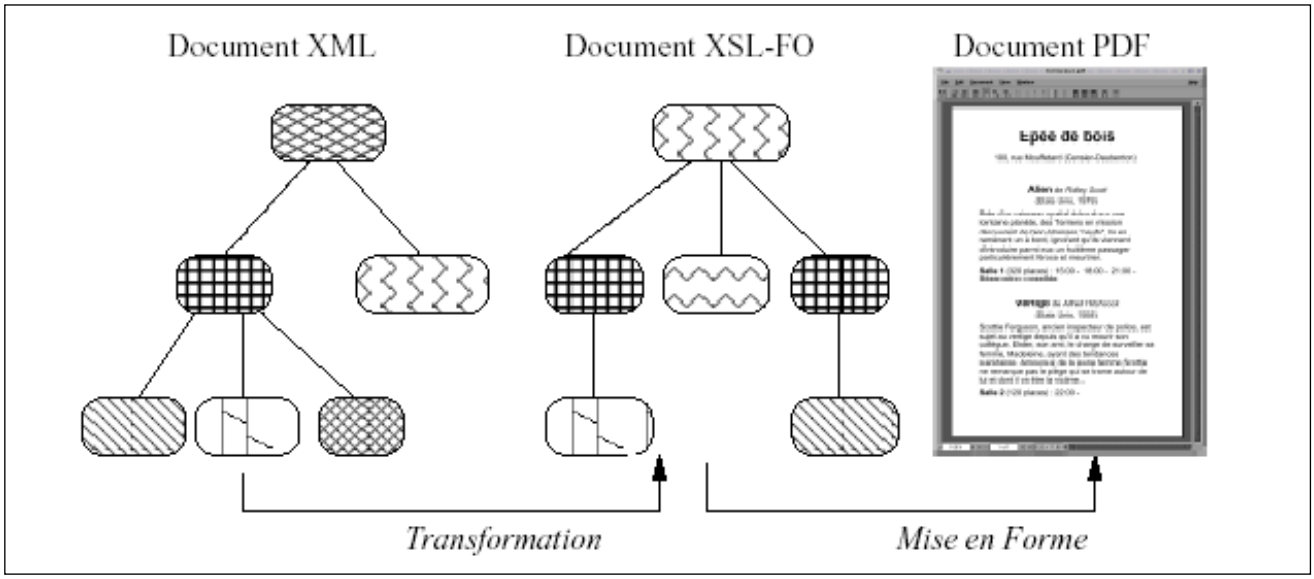
Clause sollicitant le parent du noeud contexte.

ancestor::
Clause sollicitant les ancêtres du noeud contexte (le parent, le parent du parent, etc.).
ancestor-or-self::
Clause sollicitant les ancêtres du noeud contexte et le noeud contexte lui-même.
following-sibling::
Clause sollicitant les frères du noeud contexte (qui ont le même parent) et qui suivent celui-ci dans le contexte.
preceding-sibling::
Clause sollicitant les frères du noeud contexte (qui ont le même parent) et qui précèdent celui-ci dans le contexte.
following::
Clause sollicitant tous les noeuds qui suivent le noeud contexte dans l'ordre du document, à l'exclusion des descendants du noeud contexte (et des noeuds attributs et namespaces).
preceding::
Clause sollicitant tous les noeuds qui précèdent le noeud contexte dans l'ordre du document, à l'exclusion des ancêtres du noeud contexte (et des noeuds attributs et namespaces).
attribute::
Clause sollicitant tous les noeuds attributs enfants du noeud contexte.
namespace::
Clause sollicitant tous les noeuds namespaces enfants du noeud contexte.
self::
Clause sollicitant le noeud contexte lui-même.

5.3 XSL/FO

Le langage XSL-FO « XSL formatting objects » est un « dialecte » XML destiné à décrire la mise en forme d'un document résultat.

Le couple XSLT/XSL-FO permet d'appliquer un processus complet de publication en deux étapes :



1. l'étape de transformation avec XSLT permet d'extraire des informations du document source et de les « marquer » avec des balises XSL-FO pour indiquer la présentation souhaitée : dans le cas d'un document papier, la présentation prend en compte la taille des pages, les marges à respecter, la taille et le poids des caractères, l'alignement du texte, l'espace entre les paragraphes, l'en-tête et le pied de page, etc ;
2. l'étape de mise en forme (formatting en anglais) s'applique au document XSL-FO obtenu précédemment, et consiste à créer un document final adapté au support de publication, dans un format standard comme PDF ou Postscript.

Exemple : « test.xml »

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="hello-page-xslfo.xsl" type="text/xsl"?>
```

```
<page>
 <title> Test pdf</title>
 <content>
Voici le contenu de mon fichier XML
 </content>

 <content>
Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla
Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla
Blabla Blabla Blabla Blabla Blabla Blabla Blabla Blabla
</content>

 <comment>Le 31 octobre 2007</comment>
</page>
```

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

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  version="1.0" >

<!-- rule for the whole document: root element is page -->
<xsl:template match="page">
 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>

  <!-- Definition of a single master page. It is simple (no headers etc.) -->
  <fo:simple-page-master
                                master-name="first"
                                margin-left="2cm"  margin-right="2cm"
                                margin-bottom="0.5cm" margin-top="0.75cm"
                                page-width="21cm" page-height="29.7cm"
                                >
   <!-- required element body -->
   <fo:region-body />
  </fo:simple-page-master>
 </fo:layout-master-set>

 <!-- Definition of a page sequence -->
 <fo:page-sequence master-reference="first">
  <fo:flow flow-name="xsl-region-body" font-size="14pt" line-height="14pt">
    <xsl:apply-templates/>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
</xsl:template>

<!-- A series of XSLT rules that produce fo:blocks to be inserted above -->
 <xsl:template match="page/title">
    <fo:block font-size="36pt" text-align="center" line-height="40pt" space-before="0.5cm" space-after="1.0cm">
    <xsl:apply-templates/></fo:block>
 </xsl:template>

 <xsl:template match="content">
    <fo:block text-align="justify"  space-before="0.5cm">
    <xsl:apply-templates/></fo:block>
 </xsl:template>
```

```
 <xsl:template match="comment">
   <fo:block font-size="12pt" text-align="start" space-before="0.7cm" font-style="italic">
   <xsl:apply-templates/></fo:block>
  </xsl:template>

</xsl:stylesheet>
```

Il faut ensuite un processeur « FOP » qui fait xml+xsl/fo → PDF

Pour installer FOP :
Soit

- Sur le serveur Apache (installer FOP pour apache + JAVA jdk 1.5 + créer un script pour lancer l'application FOP) (-relativement compliqué !)
  http://wiki.apache.org/xmlgraphics-fop/HowTo/PHPJavaBridge
  http://xmlgraphics.apache.org/fop/download.html
  http://tecfa.unige.ch/guides/tie/pdf/files/xml-xslfo.pdf

- Utiliser un éditeur XML comme ***oxygen*** , ***xmlspy*** (payant !) où on effectue cette opération avec un menu.

5.4 Exercices

1. **[bd1.xml] – [bd1.xsl]** Afficher tous les éléments du fichier bd.xml en utilisant une feuille de styles xsl.
2. **[bd2.xml] – [bd2.xsl]** Afficher uniquement les livres dont l'auteur est «Leo» dans le fichier bd.xml
3. **[thorgal2.xml] – [thorgal.xsl]** Afficher le document thorgal_attribut.xml avec les attributs par ordre croissant de titres.
4. **[bd3.xml] – [bd3.xsl]** Afficher uniquement les livres, dans le fichier « bd.xml », dont les titres sont soit « Les archers » soit « O.P.A. ».
5. **[bd4.xml] – [bd4.xsl]** Afficher le fichier « bd.xml » à l'aide d'une feuille de style xsl, en affichant aussi les images.
6. À l'aide du fichier « film.xml » :
   a. **[film1.xml] – [film1.xsl]** Afficher uniquement les films qui ont une cotation supérieure à 7/10.
   b. **[film2.xml] – [film2.xsl]** Afficher uniquement les remarques des films
   c. **[film3.xml] – [film3.xsl]** Afficher tout le document selon le formatage suivant :
   (les liens doivent être cliquables et la liste doit être affichée par ordre de la meilleure cote à la moins bonne.)

Films

Titre	Auteur	Remarque	Site WEB	Cote /10
Titanic	(Cameron)	Avec Leonardo DiCaprio	http://www.cinemovies.fr/fiche_film.php?IDfilm=536	9
Psychose	(Hitchcock)		http://hitchcock.alienor.fr/psychose.html	8
Sacrifice	(Tarkovski)		http://www.wmaker.net/sombreval/Le-Sacrifice-d-Andrei-Tarkovski_a207.html	7
Vertigo	(Hitchcock)		http://hitchcock.alienor.fr/vertigo.html	7
Alien	(Scott)		http://sfstory.free.fr/films/alien.html	7
Volte-Face	(Woo)	fin de stock	http://www.cinemovies.fr/fiche_film.php?IDfilm=3812	6
Kagemusha	(Kurosawa)		http://mathieu.perrin.free.fr/kagemusha.html	6

   d. **[film4.xml] – [film4.xsl]** Affichez uniquement les films dont le titre commence par la lettre "V"

Pour en savoir plus sur le XSLT…
http://www.laltruiste.com/document.php?url=http://www.laltruiste.com/coursxsl/elements.html
http://xmlfr.org/w3c/TR/xslt/
http://www.w3schools.com/xsl/

Pour en savoir plus sur XPATH…
http://www.w3.org/TR/xpath
http://www.w3schools.com/xpath/default.asp

http://www.loribel.com/info/memento/xpath.html

6. Affichage d'un document XML avec un langage de script DOM

Le modèle de programmation DOM (Document Object Model) permet, avec l'utilisation de scripts, de gérer la mise en forme d'un document XML à partir d'une page HTML. Il permet l'affichage de tous types de documents XML, même ceux qui ne sont pas structurés de façon symétrique. Il est composé d'objets, de propriétés et de méthodes permettant la gestion des différents composants d'un document XML.

6.1 Liste des principaux objets DOM pour un document XML

Objet (type de noeud)	Description
Document	Représente le document XML dans sa totalité.
Element	Représente un élément de l'arborescence.
Attribute	Représente un attribut.
Text	Représente le contenu texte d'un élément ou d'un attribut.

Méthode	Objet	Description
documentElement	Document	Contient le nœud racine du document XML.
url	Document	Renvoie l'URL du document XML.
readyState	Document	Renvoie l'état du document XML lors du chargement ou du traitement. Etats possibles : 0 => UNITIALIZED , 1 => LOADING , 2 => LOADED , 3 => INTERACTIVE , 4 => COMPLETED
length	NodeList - Attribute - Text	Renvoie le nombre de nœuds d'une collection ou le nombre de caractères dans le texte d'un nœud.
attributes	Propriété commune	Contient la collection de tous les nœuds attributs d'un nœud parent. Le nom de cette collection est : NamedNodeMap
attributes(numIndice) ou	Propriété commune	Contient le nœud attribut dont l'indice a été spécifié en

attributes.item(numIndice)		argument (à partir de 0).
childNodes(numIndice) ou childNodes.item(numIndice)	Propriété commune	Contient la collection de tous les nœuds enfants (hors attributs) d'un nœud parent dont l'indice a été spécifié en argument (à partir de 0). Le nom de cette collection est : NodeList
firstChild	Propriété commune	Contient le premier nœud enfant (hors attributs) d'un nœud parent.
lastChild	Propriété commune	Contient le dernier nœud enfant (hors attributs) d'un nœud parent.
nextSibling	Propriété commune	Contient le nœud suivant d'un nœud de même niveau.
nodeName	Propriété commune	Renvoie le nom du nœud courant, c'est à dire le nom de l'élément XML.
nodeTypeString	Propriété commune	Renvoie le type du nœud courant. (document, element, attribute,…)
nodeValue	Propriété commune	Retourne la valeur du nœud. Essentiellement pour des nœuds de type Attribute ou Text.
parentNode	Propriété commune	Contient le nœud parent du nœud courant (hors attributs).
previousSibling	Propriété commune	Contient le nœud précédent d'un nœud de même niveau.
text	Propriété commune	Renvoie le contenu texte d'un nœud et de tous ses descendants.
xml	Propriété commune	Renvoie le contenu XML d'un nœud et de tous ses descendants.
nextNode()	NodeList - NamedNodeMap	Contient le nœud suivant de la collection.

nodeFromID("id")	Document	Contient le nœud dont l'attribut de type ID possède la valeur spécifiée en argument.
getAttribute("attr")	Element	Renvoie la valeur de l'attribut spécifié en argument de l'élément courant.
getAttributeNode("attr")	Element	Contient le nœud attribut dont le nom est spécifié en argument.
getElementsByTagName("elmt")	Document - Element	Contient le nœud attribut dont le nom est spécifié en argument.
getNamedItem("attr")	NamedNodeMap	Contient le nœud attribut dont le nom a été spécifié en argument pour la collection NamedNodeMap courante.

Remarque : Vous trouverez des méthodes (DHTML) utiles aux adresses suivantes :
http://www.laltruiste.com/document.php?url=http://www.laltruiste.com/coursdomxml/sommaire.html
http://www.w3schools.com/dom/

Exemple :

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<XML id="dsoThorgal" src="thorgal.xml"></XML>
<script language="javascript">

ThorgalDocument=dsoThorgal.XMLDocument;
alert(ThorgalDocument.documentElement.childNodes(0).text);


</script>
</BODY>
</HTML>
```

Élément racine | 1er fils | Contenu de l'élément

Affichera à l'écran :

Nous sommes donc positionnés sur le 1er fils (*childnodes(0)*) de l'élément racine (*ThorgalDocument.documentElement*). Et nous avons choisi d'afficher le « text », donc le contenu du premier élément.

Si nous avions utilisé :
ThorgalDocument.documentElement.childNodes(0).childNodes(0).text
nous serions positionnés sur l'élément « text » du 1er fils de <LIVRE>, soit le contenu de <COUVERTURE> :

6.2 Mozilla vs Internet Explorer

Pour instancier un objet XML, nous utiliserons la fonction suivante pour détecter le navigateur et utiliser la bonne syntaxe en fonction de celui-ci : (attention sous IE, cette fonction ne fonctionnera que site votre page est consulté sur un serveur).

```
<script type="text/javascript">
function loadXMLDoc(dname)
{
if (window.XMLHttpRequest)
  {
  xhttp=new XMLHttpRequest();
  }
else
  {
  xhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xhttp.open("GET",dname,false);
xhttp.send();
return xhttp.responseXML;
}
</script>
```

6.3 Mozilla vs Internet Explorer : la fonction « childNodes »

Il y a une grande différence entre le DOM selon qu'on utilise Internet Explorer ou Mozilla. Il s'agit de la façon dont ils manipulent les espaces. Lors de l'ouverture d'un fichier XML, il y a des espaces entre les nœuds (par exemple un caractère comme le passage à la ligne, etc...) Quand on utilise la méthode « node.childNodes », Internet Explorer ne tient pas compte de ces espaces tandis que Mozilla en tient compte!
Ainsi, dans notre fichier « thorgal », les navigateurs de Mozilla diront que le 1er livre a 11 noeuds d'enfant, alors que l'Internet Explorer n'en verra que 5. Il faudra donc utilise une référence différente à ces « fils » :

```
<script type="text/javascript">

ThorgalDocument=loadXMLDoc("thorgal.xml");
var livre=ThorgalDocument.documentElement.childNodes;

if (window.ActiveXObject)
{
document.write(livre[0].childNodes[1].childNodes[0].nodeValue + "<br/>")
document.write(livre[0].childNodes[2].childNodes[0].nodeValue + "<br/>")
document.write(livre[0].childNodes[3].childNodes[0].nodeValue + "<br/>")
document.write(livre[0].childNodes[4].childNodes[0].nodeValue + "<br/>")
document.write(livre[0].childNodes[5].childNodes[0].nodeValue + "<br/>")
}
else if (document.implementation && document.implementation.createDocument)
{
document.write(livre[1].childNodes[3].childNodes[0].nodeValue + "<br/>")
document.write(livre[1].childNodes[5].childNodes[0].nodeValue + "<br/>")
```

```
document.write(livre[1].childNodes[7].childNodes[0].nodeValue + "<br/>")
document.write(livre[1].childNodes[9].childNodes[0].nodeValue + "<br/>")
document.write(livre[1].childNodes[11].childNodes[0].nodeValue + "<br/>")
}
</script>
```

En fonction du navigateur nous aurons à l'écran l'affichage du 1er livre :

La Magicienne trahie
Jean Van Hamme
Le Lombard
48
9,00€

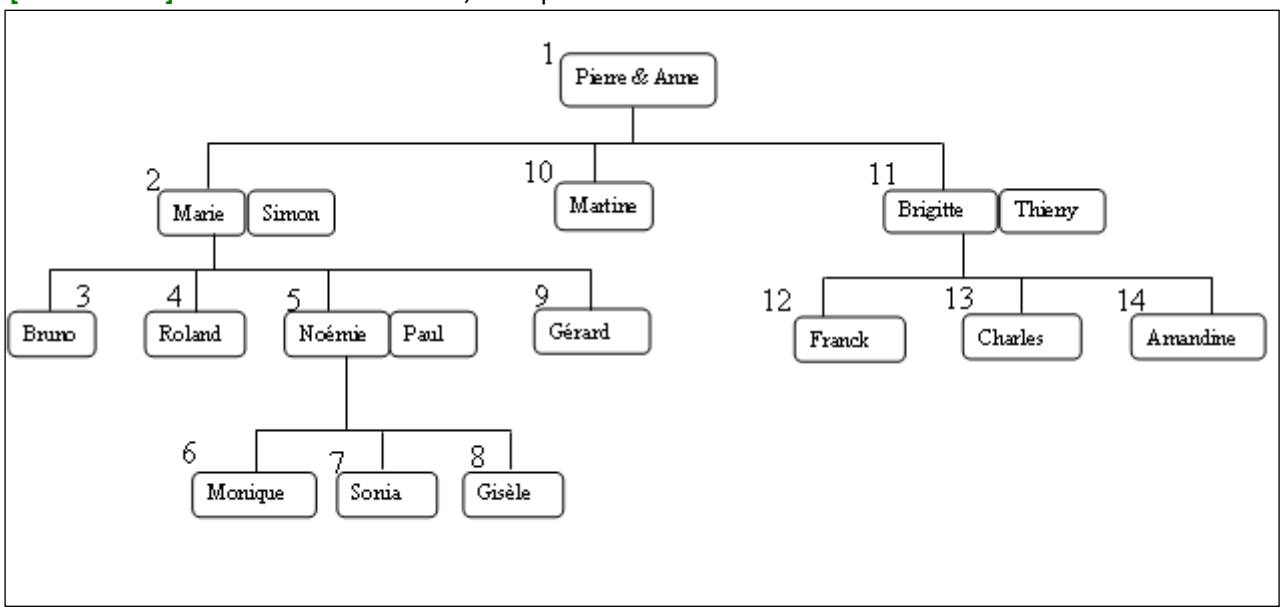
Pour éviter ce problème, il suffit de vérifier le « type de nœud ». Un noeud d'élément a le type 1, un noeud de texte (espaces, passage à la ligne, etc.) a le type 3, etc. La meilleure manière de traiter seulement des noeuds d'élément est de traiter uniquement ceux dont le type de noeud vaut 1 :

```
if (livre[i].nodeType==1)
{
…
}
```

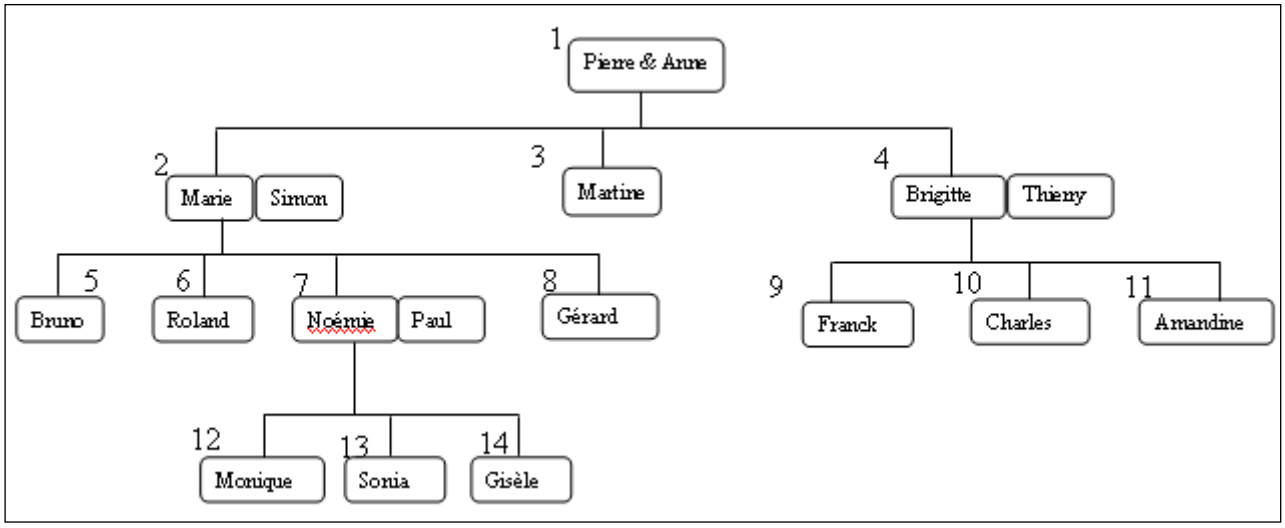
6.4 Exercices

Dans le fichier « thorgal.xml » :

1) **[dom1.html]** Affichez le contenu des 9 <LIVRE> (à l'aide de la propriété « length ») en affichant chaque fois le nom du noeud (*nodeName*) et son contenu.
2) **[dom2.html]** Refaites le même exercice en utilisant une boucle while() (pour passer au nœud suivant → *nextSibling)*
3) **[dom3.html]** Affichez uniquement les contenus de "titre" et ses attributs pour le fichier thorgalattribut.xml
4) A l'aide de la technologie DOM, créez un fichier HTML qui, avec l'arbre généalogique créé à l'exercice 4.4 (1) :
   a) **[arbre.html]** N'affiche que les personnes d'une même génération (exemple Bruno, Roland, Noémie, Gérard, Franck, Charles, Amandine)
   b) **[arbre2.html]** Affiche l'arbre en entier, mais par branche :



   c) **[arbre3.html]** Affiche l'arbre en entier, mais par génération



7. PHP 5

L'intégration de la technologie XML (et outils associés) dans le langage PHP augmente de façon significative les capacités de ce dernier à dialoguer avec des systèmes hétérogènes.

7.1 SimpleXML

Avec l'arrivée de PHP 5, le module SimpleXML permet de manipuler simplement des fichiers XML. Son utilisation est adaptée pour relire ou modifier facilement des fichiers XML simples. (Pour une utilisation de fichiers plus complexes, nous utiliserons la technologie DOM voir point 7.3).

Remarque :

Pour pouvoir utiliser les caractères spéciaux ou accentués, vous devez utiliser la fonction ***utf8_encode()*** et ***utf8_decode()***

Ouverture d'un fichier XML

```
$racine=simplexml_load_file('monfichier.xml') ;
```

Afficher ou exporter une partie du document XML

```
$racine=simplexml_load_file('monfichier.xml') ;
$racine->asXML('copie.xml')
```

Ou

```
echo $racine->asXML()
```

Ici, tout le document sera renvoyé, si vous l'employez sur un nœud du document, seul le sous-arbre concerné sera renvoyé.

Accéder à un nœud par son nom

```
$racine=simplexml_load_file('monfichier.xml') ;
$livre = $racine->livre;
$auteur=$livre->auteur;
echo $auteur ;
```

foreach()

```
$racine=simplexml_load_file('monfichier.xml') ;
$livre=$racine->LIVRE;
$premier_livre=$livre->TITRE[0] ;
foreach($livre->TITRE as $titre) {
        echo utf8_decode($titre), '<br/>' ;
        }
```

Lister les nœuds fils

```
$racine=simplexml_load_file('monfichier.xml') ;
$livre=$racine->LIVRE;
$liste=$livre->children();
echo $liste[0] ;
```

Afficher le contenu textuel d'un nœud

```
$racine=simplexml_load_file('monfichier.xml') ;
$livre=$racine->LIVRE;
echo $livre->TITRE;
```

Remarque :
PHP doit décider s'il traite la variable comme un objet (nœud XML) ou comme une chaîne de caractères. Celui-ci ne saura pas toujours faire la distinction (sur d'autres fonctions par exemple), vous pouvez alors forcer l'utilisation du contenu textuel :

```
$racine=simplexml_load_file('monfichier.xml') ;
$livre=$racine->LIVRE;
echo htmlentities((string) $livre->TITRE);
```

Accéder à un attribut

```
$racine=simplexml_load_file('monfichier.xml') ;
$livre=$racine->LIVRE;
echo $livre->AUTEUR['Naissance'];
```

Remarque : Vous trouverez des méthodes (simpleXML) utiles aux adresses suivantes :
http://developpeur.journaldunet.com/tutoriel/php/040921-php-seguy-simplexml-1a.shtml
http://be.php.net/simplexml
http://www.laltruiste.com/document.php?url=http://www.laltruiste.com/coursphp/fonction_simplexml.html

7.2 Exercices

En utilisant le module simpleXML dans le fichier « thorgal.xml » :

1) Affichez le contenu des 9 <LIVRE> (à l'aide de la l'instruction foreach) en affichant chaque fois le nom du noeud[5] et son contenu.
2) Affichez uniquement les contenus de "titre" et ses attributs pour le fichier thorgalattribut.xml
3) créez un fichier PHP qui, avec l'arbre généalogique créé à l'exercice 4.4 (1) :
    a. N'affiche que les personnes d'une même génération (exemple Bruno, Roland, Noémie, Gérard, Franck, Charles, Amandine)

---

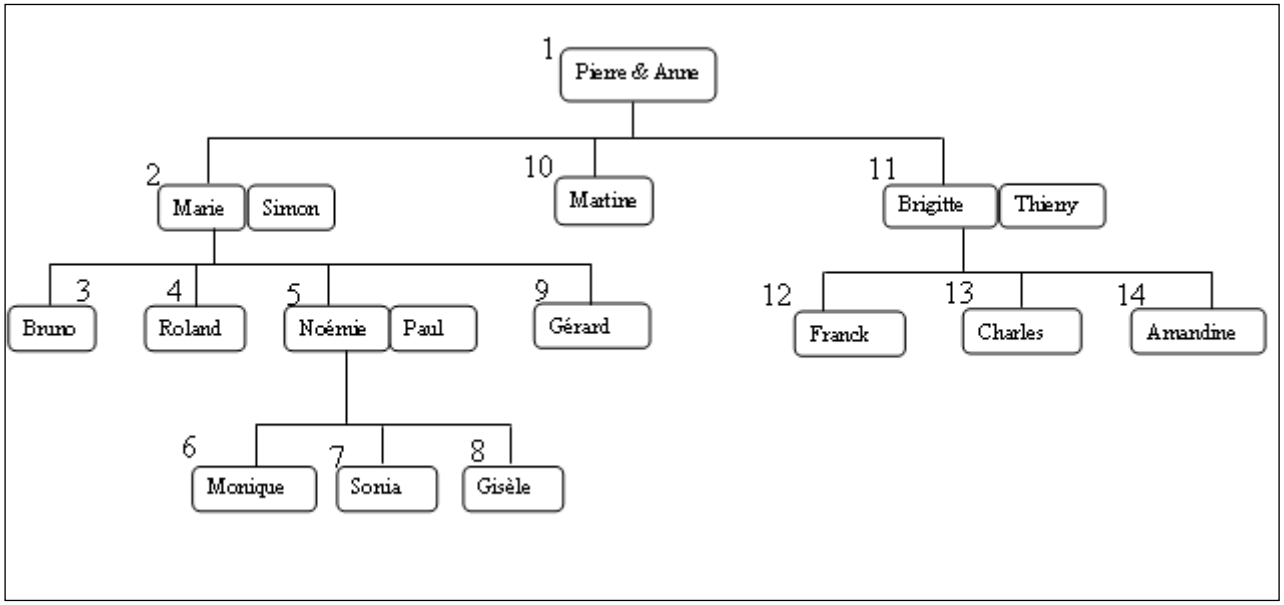
[5] foreach (http://php.net/manual/fr/control-structures.foreach.php)

La commande foreach, comme en Perl ou dans d'autres langages, est un moyen simple de passer en revue un tableau. foreach fonctionne uniquement sur les tableaux et les objets, elle retournera une erreur si vous tentez de l'utiliser sur une variable d'un autre type ou non initialisée.
Il y a deux syntaxes possibles : la seconde est une extension mineure mais pratique de la première.

```
foreach (array_expression as $value)
  commandes
foreach (array_expression as $key => $value)
  commandes
```

La première forme passe en revue le tableau array_expression. À chaque itération, la valeur de l'élément courant est assignée
à $value et le pointeur interne de tableau est avancé d'un élément (ce qui fait qu'à la prochaine itération, on accédera à l'élément suivant). La deuxième forme fait exactement la même chose, mais c'est la clé de l'élément courant qui est assigné à la variable $key.

b. Affiche l'arbre en entier, mais par branche :



[simplexml.zip]→ *dom1s.php, dom3s.php, arbres.php, arbre2s.php*

7.3 DOM

« L'extension DOM de PHP 5 permet de manipuler des documents XML avec une collection d'objets et leurs méthodes et propriétés associées. L'extension DOM de la version 5 de PHP respecte assez fidèlement les spécifications XML (Document Object Model Level 2) du W3C. Les interfaces fondamentales et étendues proposées par le W3C sont toutes présentes dans le modèle objet de l'extension DOM.[6] »

Gestion des erreurs :

```
try  {
        // code DOM avec erreurs potentiells
} catch(DomException $e){
        //traitement de l'erreur
}
```

Codages caractères :

```
utf8_encode()
utf8_decode()
```

Création (instanciation) d'un objet DOM :

```
$document = new DomDocument;
```

Chargement d'un fichier XML :

```
$document->load('monfichier.xml');
```

Import depuis SimpleXML :

```
$monfichier = simplexml_load_file('monfichier.xml);
$monfichierdom = dom_import_simplexml($monfichier);
```

Accéder à l'élément racine :

```
$racine = $document->documentElement;
```

[6] http://www.laltruiste.com/document.php?url=http://www.laltruiste.com/coursphp/domxml.html

Type de noeuds :

```
$document = new DomDocument;
$document->load('monfichier.xml');
$element = $document->documentElement;
echo $element->nodeType;
```

nodeType renvoie un entier :

Valeur	Signification	Constante
1	élément	XML_ELEMENT_NODE
2	attribut	XML_ATTRIBUT_NODE
3	noeud de texte	XML_TEXT_NODE
4	section CDATA	XML_CDATA_SECTION_NODE
5	référence d'entité externe	XML_ENTITY_REF_NODE
6	entité	XML_ENTITY_NODE
7	instruction de traitement	XML_PI_NODE
8	commentaire	XML_COMMENT_NODE
9	document	XML_DOCUMENT_NODE

Nom d'un noeud :

```
$document = new DomDocument;
$document->load('monfichier.xml');
$element = $document->documentElement;
echo $element-> nodeName;
```

Contenu d'un noeud :

```
$document = new DomDocument;
$document->load('monfichier.xml');
$element = $document->documentElement;
echo $element-> nodeValue;
```

Navigation dans l'arbre :

```
$nodelist; // liste de noeud
foreach( $nodelist as $node)
print_r($node)
```

Premier noeud de la liste :

```
$nodelist->item(0)
```

Nombre de noeuds :

```
$node->childNodes->length
```

Liste de noeuds fils :

```
$document = new DomDocument;
$document->load('monfichier.xml');
$element = $document->documentElement;
foreach($element->childNodes as $node)
{
        if ($node->nodeType == XML_ELEMENT_NODE)
        {
        echo $node->tagName;
        echo utf8_decode($node->firstChild->nodeValue);
```

```
        }
}
```

Premier fils

```
$element = $document->documentElement;
$premier = $element->firstChild;
```

Dernier fils

```
$element = $document->documentElement;
$premier = $element->lastChild;
```

Noeud parents :

```
$parent;
$fils = $parent->firstChild;
$parent = $fils->parentNode;
```

Noeud frère :

```
$node->nextSibling;
$node->previousSibling;
```

getElementByTagName:

```
$document = new DomDocument();
$document->load('monfichier.xml');
$auteur = $document->getElementsByTagName('AUTEUR');
```

Lecture d'un attribut :

```
$node->getAttribute('naissance');
```

Modification d'un attribut

```
$node->setAttribute('naissance',’14/12/1970’);
```

Effacer un attribut

```
$node->removeAttribute('naissance');
```

Le moteur DOM, permet de modifier en profondeur le document et de créer de nouveaux nœuds. (SimpleXML permet uniquement des modifications de textes et attributs).

Création d’un élément

```
$document = new DomDocument;
$node = $document -> createElement(‘LIVRE’) ;
```

Création de nœuds de texte

```
$document = new DomDocument;
$livre = $document -> createTextNode(‘Thorgal et Aaricia’) ;
```

Insertion d'un noeud fils (le fils est ajouté en dernier de la liste si d’autre fils existent déjà)

```
$node->appendChild('$texte');
```

Effacer un noeud

```
$livre->removeChild($livre->lastChild);
```

Création d'un document XML complet :

Pour créer le document suivant :

```
<?xml version="1.0"?>
<LIVRE>
 <TITRE>La Magicienne trahie</TITRE>
 <AUTEUR>Jean Van Hamme</AUTEUR>
 <EDITEUR>Le Lombard</EDITEUR>
</LIVRE>
```

```
< ?php
//création du document :
$document = new DomDocument() ;

//on crée l'élément principal <LIVRE>
$livre = $document ->createElement('LIVRE') ;
$document->appendChild($livre) ;

//on ajoute un <TITRE>
$titre = $document ->createElement('TITRE') ;
$livre->appendChild($titre) ;
//et son contenu :
$txt=utf8_encode(''La Magicienne trahie'') ;
$txt =$document->createTextNode($txt) ;
$titre->appendChild($txt) ;

//on ajoute un <AUTEUR>
$auteur = $document ->createElement('AUTEUR') ;
$livre->appendChild($auteur) ;
//et son contenu :
$txt=utf8_encode('Jean Van Hamme'') ;
$txt =$document->createTextNode($txt) ;
$auteur->appendChild($txt) ;

//on ajoute un <EDITEUR>
$editeur = $document ->createElement('EDITEUR') ;
$livre->appendChild($editeur) ;
//et son contenu :
$txt=utf8_encode('Le Lombard'') ;
$txt =$document->createTextNode($txt) ;
$editeur ->appendChild($txt) ;

// affichage du résultat
echo $document->save('fichier.xml');
```

7.4 Inclure une page XML associée à une page XSLT dans PHP

Il est tout à fait possible d'afficher une page XML mise en forme avec XSLT dans une page PHP.
Pour cela, vous devez d'abord vérifier si l'extension XSL de PHP est activée.
Dans WAMP, cliquez sur l'icône de WAMP, choisissez « PHP » puis « Extensions PHP », et cochez l'option « php_xsl ».

```
<?php
  $xslDoc = new DOMDocument();
  $xslDoc->load("thorgal.xsl");
// chargement de la feuille de style XSLT (dans $xslDoc).
```

```
  $xmlDoc = new DOMDocument();
  $xmlDoc->load("thorgal.xml");
// chargement de la page XML (dans $xmlDoc).

  $proc = new XSLTProcessor();
// nouveau container ($proc) qui doit contenir le résultat final
  $proc->importStylesheet($xslDoc);
// application de la feuille de style
  echo $proc->transformToXML($xmlDoc);
// transformation de la page XML à l'aide de la feuille de style et affichage du résultat
?>
```

7.5 Exercices

1,2,3,4) Refaites les exercices du point 6.4 en PHP 5, en utilisant l'API DOM.
[dom.zip]→ *dom1.php, dom2.php, dom3.php, arbre.php, arbre2.php, arbre3.php*
2) Créez le document « traducteur.xml » (exercice 3.2(1)) à l'aide des objets DOM

8. RSS

Le **RSS** «*Really Simple Syndication*» est un format initié par Netscape afin de présenter les titres des articles en publication de manière standardisée. Chacun peut relire ce fichier XML pour connaître les dernières nouvelles et les intégrer à une interface personnelle dans son propre site. Avec une mise à jour dynamique du contenu sans avoir à visiter manuellement le site distant. On parle de **« syndication de données ».**

8.1 Objectifs des flux RSS :

Proposer à vos lecteurs de recevoir automatiquement dans leur logiciel de lecteur de mail, ou dans leur navigateur un résumé de dernières infos ajoutées. Un flux RSS, c'est comme une page web qui se mettrait à jour toute seule en arrière-plan et vous signalerait lorsqu'il y a eu des changements. Le deuxième intérêt d'un flux RSS, c'est que d'autres sites peuvent le récupérer et afficher sur leur site les dernières nouvelles du vôtre. Ils ne récupèrent pas le contenu de votre site, ils affichent seulement votre flux. En cliquant sur les nouvelles, leurs lecteurs arriveront sur votre site. L'intérêt, c'est que des sites web traitant du même sujet peuvent ainsi mettre en commun des nouvelles fraîches, on dit qu'ils se syndiquent. Les webmasters des autres sites ne viendront pas eux-mêmes chercher les fichiers nécessaires chaque fois qu'ils seront mis à jour. Ce processus est géré de façon autonome.

8.2 Création d'un fichier RSS

Un flux rss est écrit avec le langage XML, les balises en gras sont **obligatoires**.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
<channel>
 <title>Le titre du site</title>
 <link>http://www....  lien vers le site</link>
 <description>brève description du contenu général du site</description>
 <language>fr</language>
 <copyright>apparaît tout en bas à droite</copyright>
 <image>
        <title>titre de l'image (équivalent du alt)</title>
        <url>chemin absolu de l'image http://.... (pas plus de 88px de haut)</url>
        <link>http://  lien appelé quand on clique sur l'image</link>
 </image>
 <pubdate>Sat, 30 Apr 2005 21:19:19 +0100</pubdate>

 <item>
 <title>Thorgal : BDVD </title>
 <link>http://www.bdvd.net/</link>
 <pubDate>Sat, 23 Spet 2005 00:00:00 +0100</pubDate>
 <description>La fusion du cinéma et de la bande dessinée </description>
 </item>

 <item>
 <title>...</title>
 <link>...</link>
 <pubDate>...</pubDate>
 <description>... </description>
 </item>

  </channel>
 </rss>
```

Vous pouvez remplir à la main ce genre de fichier. Il vous suffit ensuite de l'enregistrer sous par exemple « fluxrss.xml » et de le transférer sur votre site.

Donc, pour résumer, le bloc **channel** doit obligatoirement contenir

- une balise **title**, contenant le titre du flux RSS
- une balise **link**, contenant le lien (URL) principal du flux RSS
- une balise **description**, contenant la description du flux RSS
- au moins un bloc **item**

Le bloc **item** doit obligatoirement contenir au moins une balise **title** (titre de l'article du flux) ou **description** (description de l'article du flux).

→Le bloc **channel** peut être enrichi avec les informations suivantes:

- une balise **language** précisant la langue du flux RSS
- une balise **copyright** précisant une information de copyright
- une balise **managingEditor** précisant l'adresse email du responsable éditorial du flux RSS
- une balise **webMaster** précisant l'adresse email du responsable technique du flux RSS
- une balise **pubDate** précisant la date de publication, au format RFC 822. Autrement dit, au format: [Nom du jour en anglais sur 3 lettres ("Mon", "Tue", ...) suivi d'une virgule,] jour sur 2 chiffres, espace, mois en anglais sur 3 lettres ("Jan", "Feb", etc.), espace, année sur 2 chiffres, espace, heures sur 2 chiffres, deux-points, minutes sur 2 chiffres[, deux-points, secondes sur 2 chiffres], espace, fuseau horaire ("UT", "GMT", "EST", etc.). Les données entre crochets sont optionnels. L'année peut également être précisée avec 4 chiffres.
- une balise **lastBuildDate** précisant la date (au format RFC 822) de dernière modification du contenu.
- une ou plusieurs balise(s) **category** précisant à quelle catégorie appartient (i.e. a quel sujet se rapporte) le

flux RSS.

- une balise **generator** précisant le nom du programme ayant généré le flux RSS.
- une balise **docs** précisant l'adresse (URL) du document décrivant le format RSS utilisé.
- une balise **ttl** précisant la durée (en minutes) pendant laquelle le contenu du flux RSS peut être conservé en cache avant qu'une mise à jour ne soit nécessaire.
- un bloc **image** contenant
  - Une balise **url** précisant l'adresse (URL) du fichier image
  - Une balise **title** précisant le titre de l'image
  - Une balise **link** précisant une URL associé à l'image (à priori l'URL du flux RSS)
  - Une balise **width** précisant la largeur (en pixels) de l'image
  - Une balise **height** précisant la hauteur (en pixels) de l'image
  - Une balise **description** ajoutant une description à l'image

→Les blocs **item** peuvent être enrichis avec les informations suivantes:

- une balise **author** précisant l'adresse email de l'auteur de l'article
- une ou plusieurs balises category précisant le sujet auquel se rapporte l'article
- une balise comments précisant l'adresse (URL) d'un document contenant les commentaires associés à l'article
- une balise guid définissant un identifiant unique pour l'article
- une balise pubDate précisant la date (au format RFC 822) de publication de l'article
- une balise source précisant l'adresse (URL) du flux RSS d'où vient l'article

8.3 Afficher un flux RSS

Une fois que vous connaissez une adresse RSS ou XML, que ce soit la vôtre ou celle d'un autre site, vous pouvez :

- soit en faire afficher le contenu sur un lecteur RSS, auquel cas vous téléchargez un de ces lecteurs, le plus souvent gratuits
  Par exemple :
  - Newsgator
  - google reader
  - Pluck
  - Rss Explorer
  - Lektora…

  Et ensuite vous ajouterez l'URL de ce fil RSS dans ceux que ce lecteur surveillera pour vous.
- Soit en faire afficher le contenu sur votre propre site, auquel cas vous devrez ajouter du code dans votre page selon la mise en page désirée.
  - Le site http://jade.mcli.dist.maricopa.edu/feed/ génère pour vous en javascript l'affichage de fil RSS (mais la sources en caché, donc pas de modification possible)
  - MagpieRSS : RSS for PHP, http://magpierss.sourceforge.net/ est un outil en PHP disponible sous licence GPL (basé sur la librairie ***Expat*** de J. Clark)

Après avoir téléchargé la classe *magpierss,* le code suivant lie un fichier rss :

```
<?php
// inclusion de la classe magpierss
require_once("rss_fetch.inc");

$rss = fetch_rss("http://bono/fluxrss.xml");
print_r ($rss->items[0]["title"]);
?>
```

Nous sommes en présence de 3 tableaux, dont un à 2 dimensions.

Premier tableau, les informations sur le flux ($rss->channel) :

- $rss->channel["title"] : le titre du flux
- $rss->channel["link"] : le lien vers le site générateur du flux

- $rss->channel["description"] : la description du flux

etc..

Deuxième tableau, les informations sur la vignette du flux ($rss->image) :

- $rss->image["title"] : le titre de l'image
- $rss->image["url"] : l'adresse de l'image
- $rss->image["link"] : le lien associé à l'image
- $rss->image["description"] : la description de l'image

Troisième tableau, à 2 dimensions, les informations ($rss->items) :

- $rss->items[0]["title"] : le titre de l'info
- $rss->items[0]["link"] : le lien associé à l'info
- $rss->items[0]["description"] : la description de l'info
- $rss->items[0]["pubdate"] : la date de publication de l'info
- ...

Pour plus d'informations concernant l'utilisation de cette classe :
http://www.atome77.com/forum/index.php?showtopic=286
http://www.biologeek.com/journal/index.php/2005/02/09/35-magpie-rss-installation-et-utilisations-en-tout-genre-du-parser-php

8.4 Exemple de création d'un fichier statique RSS à partir d'une table mysql

Si vous gérez votre site, ou tout au moins les nouvelles pour lesquelles vous voulez générer le flux, à l'aide d'une table mysql sur votre serveur, vous pouvez l'utiliser pour générer les données du fichier xml. Vous allez pour cela créer un fichier php qui interroge la table mysql, sort les données récentes (votre table doit obligatoirement comporter un champ date ou time stamp) et les met en forme. Ce fichier que l'on appellera « maj-rss.php » va être lui-même créer le fichier « fluxrss.xml » contenant le flux rss. Chaque fois que vous modifierez votre site, il faudra ouvrir via votre navigateur le fichier « majrss.php » (sur le serveur) pour que celui-ci recrée un fichier xml tenant compte des dernières modifications. Pour l'exemple, considérons une table mysql toute simple, nommée « nouvelles » avec 4 champs : un titre, un lien (adresse où l'on peut trouver la nouvelle complète), une date, et le contenu de la nouvelle. Les 3 premiers sont obligatoires. Il peut y en avoir d'autres.

Voici ce que devra contenir votre page « maj-rss.php » :

```
<?php
$xml = '<?xml version="1.0" encoding="iso-8859-1"?><rss version="2.0">';
$xml .= '<channel>';
$xml .= '<title>MonTitre</title>';
$xml .= '<link>http://www.monsite.net</link>';
$xml .= '<description>Mon Site est le meilleur</description>';
$xml .= '<copyright>© MonSite 2005</copyright>';
$xml .= '<language>fr</language>';
$xml .= '<image>';
$xml .= '    <title>MonSite</title>';
$xml .= '    <url>http://www.MonSite.net/images/MonImage.gif</url>';
$xml .= '    <link>http://www.MonSite.net</link>';
$xml .= '</image>';
$jourdui= date("D, d M Y H:i:s +0100");
//Sat, 23 Apr 2005 00:01:00 +0100
$xml .= '<pubdate>'.$jourdui.'</pubdate>';

require ('../include/connect.php');
// Vous pouvez soit faire appel à un fichier externe contenant
//toutes les infos de connexion à votre base et table
// ou vous pouvez écrire ici ces infos directement

$res=mysql_query("select * from nouvelles order by date desc limit 0, 10");
```

```
// extraction des 10 dernières nouvelles
while($lig=mysql_fetch_array($res)){
                        $titre=$lig[tag];
                        $adresse=$lig[adresse];
                        $contenu=$lig[contenu];
                        $madate=$lig[date];
                        $datephp=date("D, d M Y H:i:s +0100", strtotime($madate));

        $xml .= '<item>';
        $xml .= '<title>'.$titre.'</title>';
        $xml .= '<link>'.$adresse.'</link>';
                                $xml .= '<pubDate>'.$datephp.'</pubDate>';
        $xml .= '<description>'.$contenu.'</description>';
        $xml .= '</item>';
            }//fin du while

$xml .= '</channel>';
$xml .= '</rss>';

$fp = fopen("fluxrss.xml", 'w+');
fputs($fp, $xml);
fclose($fp);

echo 'Export XML effectue !<br><a href="fluxrss.xml">Voir le fichier</a>';
?>
```

Nous voyons au passage que la date est écrite dans les fichiers rss d'une façon un peu particulière pour nous autres francophones. Mais les fichiers rss sont au départ destinés à être lus par des machines. Suivant la façon dont votre date est entrée dans votre table mysql, vous serez amené à modifier le formatage dans ce fichier.
Sources :
http://www.commentcamarche.net/
http://thot.cursus.edu/rubrique.asp?no=21277

9. Pour aller plus loin…

9.1 XQuery : XML comme une base de données

Il existe un langage pour extraire des informations d'un document XML : XQuery.
Il adopte une syntaxe très proche de SQL, et utilise XPath pour retirer n'importe quelle donnée d'un fichier XML.
Il utilise aussi Xpath pour se repérer dans le document.

Ce langage permet en plus toutes les opérations que l'on peut vouloir faire avec SQL : insertion de données, mise à jour, suppression, etc.

9.2 XForms

XForms est la prochaine génération de formulaires web. C'est un langage vraiment très puissant. Il permet entre autres la soumission des données en XML. La vérification des données en direct, grâce à CSS et XML Schéma, tout ce que permet XPath pour le calcul des données soumises, de s'affranchir de scripts pour la répétition d'éléments, etc.

9.1 ASP (Active Server Page)

« Les langages de programmation tels que Java, C++ ou ASP (Active Server Pages) s'appuient sur le DOM afin de traiter des documents XML. Développé par le W3C (World Wide Web Consortium), le modèle d'objet est universel,

indépendant d'un quelconque langage ou plateforme. Seule l'implémentation, l'application capable d'exécuter des opérations à partir du DOM, est à la charge d'un fournisseur tel que Sun Microsystems, Microsoft, ou autre. »[7]

L'ASP utilise les mêmes fonctions & méthodes que le XMLDOM. La connexion à l'objet XMLDOM se fait comme suit :

```
<%
Set objDOM = Server.CreateObject("Microsoft.XMLDOM")
objDOM.async = false
objDOM.Load Server.MapPath("thorgal.xml")

Set NodeList = objDOM.getElementsByTagName("LIVRE")
For each elem in Nodelist
        response.write "LIVRE : " & elem.firstChild.nodevalue & "<BR>"
next
%>
```

9.2 Liens évolués XLL : XPOINTER et XLINK

Le **XLL** (eXtended Linking Language) est un dérivé du langage XML.
Il permet de gérer les liens dans un document XML.
Le langage étendu des liens est composé de deux parties :

- Les XLinks (Liens XML)
- Les XPointers (Pointeurs XML)

Les XLinks permettent d'accéder à un document par l'intermédiaire d'un URI (Uniform Resource Identifier) à l'image de la balise <a href="URL"> du langage HTML.

Les **Xpointers** ajoutent des fonctionnalités aux liens étendus par l'intermédiaire d'une expression qui apporte plus de précisions sur la cible à atteindre.

Ainsi, le XLL avec ses deux composantes, offre une gestion plus efficace des liens vers des ressources externes.

Dans HTML, la gestion des liens est limitée :

- il n'est possible d'accéder qu'à un seul document à la fois,
- l'accès à une section particulière d'un document n'est opérable que si, auparavant, l'ancrage ... a été soigneusement opéré lors de la conception de ce document,
- aucune prise en charge de l'historique des liens n'est prise en compte par le langage HTML, pour cela il est nécessaire d'utiliser les langages de script,
- les liens et ancres sont déterminés par avance et ne laissent donc plus aucune possibilité d'extension dynamique.

Ne se contentant pas de reprendre en charge les fonctions des liens HTML, le langage XLL propose de les améliorer et de les étendre à un niveau d'efficacité remarquable :

- N'importe quel élément XML peut devenir un lien.
- Tous les types de ressources peuvent être non seulement accédés, mais également pointés en des points précis.
- Les liens multidirectionnels sont réalisables.
- Les liens peuvent être organisés en groupe de connexions.
- La gestion des arcs de liens devient effective.

Le langage XLL faisant encore aujourd'hui l'objet d'étude par le W3C, les spécifications évoluent et ne sont pas encore parfaitement opérationnelles au sein de toutes les applications XML. Cependant, cet outil prometteur devrait à terme constituer l'ossature de la navigation sur le Web.
Néanmoins si vous voulez en savoir plus :
http://www.laltruiste.com/document.php?compteur=3&page=1&rep=6

[7] L'altruiste (http://www.laltruiste.com/document.php?compteur=4&rep=8&evolution=2)

9.3 Images au format SVG

Le SVG (Scalable Vector Graphics) est un format vectoriel d'image basé lui aussi sur XML. (Basée sur des formes simples qui se redimensionnent sans dégradation)
Le principe est d'intégrer une image SVG à une page Web sans se préoccuper de la taille d'affichage chez le visiteur, afin d'obtenir une image d'une qualité parfaite. (Et qui prend très peu de temps à être créée/manipulée, car le XML n'est que du texte.)

Pour en savoir plus…
http://svground.free.fr/
http://ptaff.ca/svg/
http://fr.wikipedia.org/wiki/SVG
http://www.w3.org/Graphics/SVG/
http://gilles.chagnon.free.fr/cours/xml/svg.html

9.4 SAX

SAX ("Simple API for XML" ou en français "API simple pour XML") est une interface de programmation qui permet l'analyse du format XML. Elle a été mise au point par David Megginson et est progressivement devenue un standard de fait.

La particularité de SAX, c'est qu'elle traite un document XML comme un flux. Ainsi, cette API fournit à l'application qui l'utilise les éléments constitutifs du document au fur et à mesure de la lecture. On dit qu'il s'agit d'une approche de type événementiel.

Une application qui exploite l'API SAX va, en conséquence, mettre en oeuvre des gestionnaires d'événements. A mesure que l'analyse du document XML se poursuit, des événements sont relevés et signifiés au programme qui peut ensuite les traiter. Par exemple, un événement peut être généré pour chaque nouvelle ouverture de balise rencontrée, ou pour chaque fermeture de balise.

L'API SAX est une alternative à la technologie DOM, elle permet donc aussi d'analyser un flux de données XML. Le gros avantage de son fonctionnement est sa faible consommation de ressources car SAX traite les documents élément par élément au fur et à mesure qu'ils sont rencontrés. Cependant le programmeur peut très bien recueillir les données qui l'intéressent dans les structures de son choix.
Pour en savoir plus …
http://www.laltruiste.com/document.php?url=http://www.laltruiste.com/coursjava/apixml_sax_parser.html
http://www.dcs.bbk.ac.uk/~mick/academic/xml/dip/sax-dom.shtml
http://www.laltruiste.com/document.php?url=http://www.laltruiste.com/coursxsl/xpath.html

9.5 AJAX

AJAX[8], ou « Asynchronous JavaScript And XML » (« XML et Javascript asynchrones »), est un acronyme désignant une méthode informatique de développement d'applications Web.
À l'image de DHTML ou de LAMP, AJAX n'est pas une technologie en elle-même, mais un terme qui évoque l'utilisation conjointe d'un ensemble de technologies couramment utilisées sur le Web :

- HTML (ou XHTML) pour la structure sémantique des informations ;
- CSS pour la présentation des informations ;
- DOM et JavaScript pour afficher et interagir dynamiquement avec l'information présentée ;
- l'objet XMLHttpRequest pour échanger et manipuler les données de manière asynchrone avec le serveur web.
- XML et XSLT

[8] Sources : http://fr.wikipedia.org/

En alternative au couple XML/XSLT, les applications AJAX peuvent utiliser d'autres technologies: le HTML préformaté, les fichiers texte plats, JSON et JSON-RPC.

10. Liens utiles

- Guide des langages web
  http://www.laltruiste.com/
- Manipulation de l'XML avec ActiveX Data Object
  http://www.laltruiste.com/document.php?compteur=4&rep=8&evolution=61
- XML parsing avec PHP
  http://tecfa.unige.ch/guides/tie/html/php-xml/php-xml.html
- Javascript DOM
  http://selfhtml.selfhtml.com.fr/javascript/objets/elementshtml.htm
- initiation à XML
  http://www.asp-php.net/tutorial/xml/index.php
- XML + XSL
  http://www.ccim.be/ccim328/xml/
- W3C Markup Validation Service
  http://validator.w3.org/
- Création d'une DTD
  http://grds.ebsi.umontreal.ca/remillc/inu1011/2004/lecon-03.htm
- THE LARGEST WEB DEVELOPER SITE ON THE NET - Full Web Building Tutorials
  http://www.w3schools.com/