PHP : XML

L3 Informatique - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2020

Rappels SGML, XML et XSL

Qu'est ce que SGML ?

SGML - Standard Generalized Markup Language

- Langage de description de langages à balises.
- Norme ISO 8879 : 1986.
- Utilisé par les professionnels des domaines de la documentation et de l'édition.
- Assez complexe.
- Séparation du contenu et de la forme.

Qu'est ce que XML ?

XML - eXtensible Markup Language

- Issu de SGML.
- Standard qui permet de représenter l'information de manière structurée de façon et lui donner une sémantique.
- Repose sur un langage de balises prédéfinies.
- Facilite l'échange d'informations entre systèmes hétérogènes.

Avantages et inconvénients de XML

Avantages de XML

- Séparation du contenu et de la forme.
- Extensible.
- Données et méta-données.
- Supporté par les applications Web.
- Fonctions de recherche et transformation.
- Lié à SOAP (Simple Object Access Protocol) pour l'invocation de méthodes d'objets distants (Web services).

Inconvénients de XML

- Verbeux (augmentation de la taille des fichiers).
- Consommateur de ressources (représentation en mémoire, processeur).

Exemple de fichier XML

bibliotheque.xml

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!-- Déclaration externe de DTD -->
<!DOCTYPE bibliotheque SYSTEM "bibliotheque.dtd">
<!-- Référence à un fichier XSLT pour mise en forme -->
<?xml-stylesheet type="text/xsl" href="bibliotheque-liste.xsl" ?>
<bibliotheque>
<livre>
  <titre>Apprendre XML en 10 mois</titre>
  <liste_auteurs>
    <auteur nom="Escargot" prenom="Jean" />
    <auteur nom="Snail" prenom="John" />
  </liste_auteurs>
  <prix>20</prix>
</livre>
<livre>
  <titre>Learn Java in 10 seconds</titre>
  <liste_auteurs>
    <auteur nom="Fast" prenom="Bob" />
  </liste_auteurs>
  <prix monnaie="dollars">25</prix>
</livre>
</bibliotheque>
```

Prologue du document XML

bibliotheque.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
```

Première ligne indiquant

- La version de XML utilisée.
- Le jeu de caractères utilisé dans le document (`utf-8` recommandé).
- La dépendance du document à un document externe (eg. DTD, XSL) ou non.

Description du fichier XML

Une hiérarchie d'éléments

Contenant des sous-éléments, ou ne contenant que du texte, ou sans contenu (balise auto-fermante).

Elements

- `bibliotheque` est l'élément racine du document.
- `bibliotheque` comporte au moins un élément `livre`.
- Chaque `livre` comporte un `titre`, une `liste_auteurs`, et un `prix`.
- Chaque `liste_auteurs` comporte au moins un `auteur`.
- `titre` et `prix` contiennent du texte, `auteur` est vide.

Attributs

- Chaque `auteur` a deux attributs `nom` et `prenom`.
- L'attribut `monnaie` (devise) peut être donné pour un `prix`.

Qu'est ce que XSL ?

XSL - eXtensible Stylesheet Language

Une famille de langages pour la transformation et la présentation de documents XML :

- XSLT (XSL Transformations) : langage de transformation de document XML.
- XPath (XML Path Language) : langage d'expressions utilisé par XSLT pour désigner une/des parties d'un arbre XML.
- XSL-FO (XSL-Formatting Objects) : un vocabulaire XML pour spécifier une sémantique de mise en forme.

XSLT

- Permet de transformer un document XML vers tout autre schéma ou format : XHTML, HTML, texte . . .
- Se compose d'un analyseur XSLT qui sur la base d'un document XML et d'un document XSLT produit un fichier de sortie au format désiré.

Exemple de fichier XSL - rendu en liste HTML

Exécution via navigateur ou en ligne de commande (xsltproc, xalan)

```
$ xsltproc bibliotheque.xml bibliotheque-liste.xsl
```

bibliotheque-liste.xsl

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
 3 <xsl:output method="html"/>
 4 <xsl:template match="/">
 5   <html>
 6     <body bgcolor="#FFFFFF">
 7       <ul>
 8         <xsl:for-each select="bibliotheque/livre">
 9         <li>
10         <xsl:value-of select="titre" />,
11         <xsl:for-each select="liste-auteurs/auteur">
12           <i>
13           <xsl:value-of select="./@prenom" /><xsl:text> </xsl:text>
14           <xsl:value-of select="./@nom" />
15           </i>,
16         </xsl:for-each>
17         prix <xsl:value-of select="prix" /><xsl:text> </xsl:text>
18         <xsl:value-of select="prix/@monnaie" />,
19         </li>
20         </xsl:for-each>
21       </ul>
22     </body>
23   </html>
24 </xsl:template>
25 </xsl:stylesheet>
```

Rendu sous forme de liste

- Apprendre XML en 10 mois,
  Jean Escargot, *Brad Snail*,
  prix 20
- Learn Java in 10 seconds,
  John Fast,
  prix 25 dollars

Exemple de fichier XSL - rendu en table HTML

Exécution via navigateur ou en ligne de commande (`xsltproc`, `xalan`)

```
$ xsltproc bibliotheque.xml bibliotheque-table.xsl
```

bibliotheque-table.xsl

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
 3 <xsl:output method="html"/>
 4 <xsl:template match="/">
 5   <html>
 6     <body>
 7       <table border="1" cellpadding="4">
 8       <xsl:for-each select="bibliotheque/livre">
 9         <tr valign="top">
10         <td><xsl:value-of select="titre"/></td>
11         <td><xsl:for-each select="liste_auteurs/auteur">
12                <i>
13                <xsl:value-of select="./@prenom"/><xsl:text> </xsl:text>
14                <xsl:value-of select="./@nom"/>
15                </i><br/>
16             </xsl:for-each>
17         </td>
18         <td><xsl:text>prix </xsl:text>
19           <xsl:value-of select="prix"/><xsl:text> </xsl:text>
20           <xsl:value-of select="prix/@monnaie"/>
21         </td>
22         </tr>
23       </xsl:for-each>
24       </table>
25     </body>
26   </html>
27 </xsl:template>
28 </xsl:stylesheet>
```

Rendu sous forme de table

Apprendre XML en 10 mois	*Jean Escargot* *Brad Snail*	prix 20
Learn Java in 10 seconds	*John Fast*	prix 25 dollars

Document Type Definition

Définition du type de document

Document Type Definition (DTD)

Utilisé pour normer des documents XML.

- En décrit la grammaire - la liste des éléments et balises autorisés, leurs attributs, contenu et agencement - ainsi que le vocabulaire supplémentaire sous la forme d'une liste d'entités de caractères.

XML Schema est une alternative plus puissante.

Utilisation d'une DTD (fichier d'extension `.dtd`

- En la définissant directement dans le document XML.
- En la définissant dans un fichier externe déclaré dans le document XML (recommandé).

DTD pour documents (X)HTML

XHTML 1.0 Strict

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

XHTML 1.0 Transitional

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

XHTML 1.0 Frameset

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

HTML 5

```
1 <!DOCTYPE html>
```

Exemple de fichier XML déclarant une DTD externe

bibliotheque.xml

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!-- Déclaration externe de DTD -->
<!DOCTYPE bibliotheque SYSTEM "bibliotheque.dtd">
<!-- Référence à un fichier XSLT pour mise en forme -->
<?xml-stylesheet type="text/xsl" href="bibliotheque-liste.xsl" ?>
<bibliotheque>
<livre>
  <titre>Apprendre XML en 10 mois</titre>
  <liste_auteurs>
    <auteur nom="Escargot" prenom="Jean" />
    <auteur nom="Snail" prenom="John" />
  </liste_auteurs>
  <prix>20</prix>
</livre>
<livre>
  <titre>Learn Java in 10 seconds</titre>
  <liste_auteurs>
    <auteur nom="Fast" prenom="Bob" />
  </liste_auteurs>
  <prix monnaie="dollars">25</prix>
</livre>
</bibliotheque>
```

Déclaration du type du document

bibliotheque.xml

```
<!DOCTYPE bibliotheque SYSTEM "bibliotheque.dtd">
```

Document Type Declaration (DOCTYPE)

La déclaration du type du document indique

- La DTD à laquelle doit se conformer le document XML.
- La balise de l'élément racine : `bibliothèque`.

Ne pas confondre *Document Type Definition* (DTD) et *Document Type Declaration* (DOCTYPE).

Exemple de DTD

bibliotheque.dtd

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <!ELEMENT bibliotheque (livre+) >
3  <!ELEMENT livre (titre, liste_auteurs, prix?) >
4  <!ELEMENT titre (#PCDATA) >
5  <!ELEMENT liste_auteurs (auteur+) >
6  <!ELEMENT auteur EMPTY>
7  <!ATTLIST auteur nom CDATA #REQUIRED >
8  <!ATTLIST auteur prenom CDATA #IMPLIED >
9  <!ELEMENT prix (#PCDATA) >
10 <!ATTLIST prix monnaie CDATA #IMPLIED>
```

Structure de la DTD

bibliotheque.dtd

```
<!ELEMENT bibliotheque (livre+)>
```

Règle imposée

L'élément (de balise) `bibliotheque` contiendra un ensemble d'éléments `livre`.
On peut quantifier le nombre d'éléments `livre` :

- `+` : au moins 1.
- `*` : 0 ou plus.
- `?` : 0 ou 1.
- Sans indication, on attend une seule occurrence.

Structure de la DTD

bibliotheque.dtd

```
<!ELEMENT livre (titre, liste_auteurs, prix?)>
```

Règle imposée

Un élément `livre` contiendra :

- Un élément `titre`.
- Suivie d'un élément `liste_auteurs`.
- Suivie éventuellement d'un élément `prix`.

`a | b` signifie *soit* `a`, *soit* `b`.

Structure de la DTD

bibliotheque.dtd

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

Règle imposée

Un `titre` contiendra une chaîne de caractères.

- `PCDATA` : la chaîne sera analysée comme du XML (balisage reconnu comme tel et entités XML automatiquement développées). Utiliser les entités XML correspondant aux symboles $<$, $>$ et &. Réservé pour le contenu d'élément.
- `CDATA` : la chaîne sera donnée entre guillemets mais non analysée. Réservé pour les attributs d'éléments.
- `ANY` : toute combinaison de données analysables.

Structure de la DTD

bibliotheque.dtd

```
<!ELEMENT auteur EMPTY>
<!ATTLIST auteur nom CDATA #REQUIRED>
<!ATTLIST auteur prenom CDATA #REQUIRED>
```

Règle imposée

Un `auteur` est un élément sans contenu (`EMPTY`) et qui possède deux attributs `nom` et `prenom`.
On peut contraindre chaque attribut :

- `#REQUIRED` : l'attribut doit être défini.
- `#IMPLIED` : l'attribut est optionnel.
- `#DEFAULT` : l'attribut possède une valeur par défaut définie par une valeur entre guillemets.

Entités

Alias pour du texte

Equivalent du `#define` en C.

Définition dans la DTD

```
<!ENTITY progc "programmation en langage C">
```

Usage dans le XML

```
<keyword> &proc </keyword>
```

Manipuler des documents XML

Manipuler XML : DOM et SAX

DOM et SAX

Il existe deux interfaces principales de manipulation pour la lecture des documents XML :

- DOM (Document Object Model) basée sur une représentation hiérarchique.
- SAX (Simple API for XML) basée sur des déclencheurs d'actions.

Le DOM

Interface de programmation (API)

Standardisée pour manipuler par scripts une page HTML chargée dans un navigateur et plus généralement tout document XML.

Appliqué aux pages Web (XHTML*, HTML5), le DOM

- Offre une représentation structurée et orientée objet des éléments et du contenu.
- Permet la modification des propriétés de ces objets par l'intermédiaire de méthodes.
- Permet l'ajout et la suppression d'objets.
- Permet la gestion des évènements du navigateur.

Caractéristiques des API DOM et SAX

DOM

- Pas vraiment orientée objet.
- Construit une représentation en mémoire du document sous forme d'arbre.
- Adaptée aux petits fichiers et traitement du document entier.

SAX

- Basée sur des déclencheurs (triggers) qui se déclenchent lors de la lecture de balises.
- Adaptée au traitement local de fichiers volumineux.
- Adaptée à des langages compilés comme C++ ou Java.

Les normes du DOM

Recommandations du W3C

- DOM Level 1 (Core + HTML) depuis 1998.
- DOM Level 2 (Core + Event + Style + Views + Traversal + Range) depuis 2000.
- DOM Level 3 (Validation) depuis 2004.

Mise en oeuvre

- Standardisée autour de JavaScript dans tous les navigateurs.
- API dépendante du langage de scripts côté serveur pour la manipulation de documents XML (eg. les classes `DOMDocument`, `DOMElement` ... en PHP).

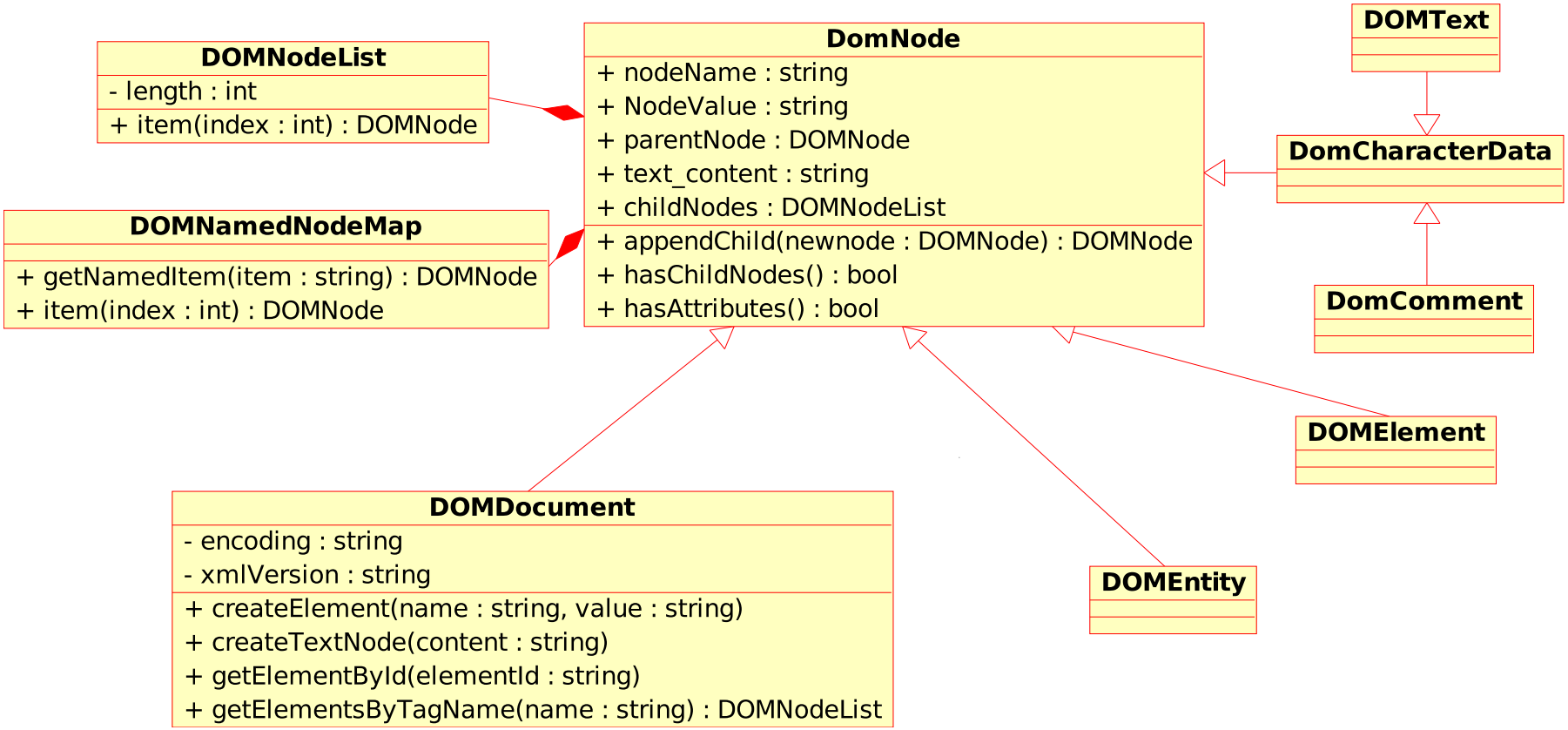
L'arbre d'éléments DOM

Arbre des éléments

- De part sa structure, un document XML est représenté sous forme arborescente.
- Cet arbre possède une racine, des noeuds et des feuilles de différents types.

Visualisation de la structure d'un document HTML sous Firefox avec DOM Inspecteur.

Diagramme UML de l'API DOM



Noeud DOM

DOMNode

Les attributs sont :

- `nodeName` (`string`) : nom du noeud.
- `nodeType` (`int`) : type du noeud.
- `nodeValue` (`string`) : valeur.
- `childNodes` (`DOMNodeList`) : liste des enfants.
- `parentNode` (`DOMNode`) : noeud parent.
- `firstChild`, `lastChild` (`DOMNode`).
- `previousSibling`, `nextSibling` (`DOMNode`).

Manipuler des documents XML sous PHP

On peut utiliser

- L'API `DOMDocument`.
- L'extension `SimpleXML`.

L'API DOMDocument

Créer un document XML avec `DOMDocument`

On précise :

- La version.
- L'encodage des caractères.

dom-create.php

```php
$dom = new DOMDocument('1.0', 'utf-8');
echo $dom->saveXML();
```

Charger un document XML/HTML

Méthodes booléennes

- `load(string)` pour un fichier XML.
- `loadHTMLFile(string)` pour un fichier HTML.
- `loadXML(string)` à partir d'une chaîne.
- `loadHTML(string)` à partir d'une chaîne.

dom-load.php

```php
error_reporting(E_ALL);
$dom=new DOMDocument();
if ($dom->load(__DIR__.'/bibliotheque.xml')===false) {
    die("error load");
} else {
    echo $dom->saveXML();
}
```

Valider un document XML avec une DTD

dom-validate.php

```php
error_reporting(E_ALL);
$dom = new DOMDocument();
$docs = array();
$docs[] = 'bibliotheque.xml';
$docs[] = 'bibliotheque-invalide.xml';
foreach ($docs as $doc) {
   $dom->load($doc);
   echo "Document ".$doc;
   if ($dom->validate()===true)
      echo " valide\n";
   else
      echo " invalide\n";
}
```

Sauvergarder un document XML/HTML

Retournent le nombre d'octets écrits en cas de succès et `FALSE` en cas d'erreur

- `save(string)` pour un fichier XML.
- `saveHTMLFile(string)` pour un fichier HTML.

Retournent une chaîne XML/HTML en cas de succès et `FALSE` en cas d'erreur

- `string saveXML(void).`
- `string saveHTML(void).`

Exemple de sauvegarde

dom-save.php

```php
$dom=new DOMDocument();
$root=$dom->createElement('bibliotheque','some text');
$dom->appendChild($root);
if ($dom->save(__DIR__.'/dummy.xml')==false)
   die("error save");
echo $dom->saveXML();
```

dummy.xml

```xml
<?xml version="1.0"?>
<bibliotheque>some text</bibliotheque>
```

Créer un document XML/HTML

Méthodes principales

- `DOMElement createElement(string name, string value)`
- `DOMText createTextNode(string name)`
- `DOMAttr createAttribute(string name)`
- `appendChild(DOMNode node)`

Exemple de création

dom-create-element.php

```php
$dom=new DOMDocument('1.0', 'iso-8859-1');
$root_element=$dom->createElement('bibliotheque',");
$dom->appendChild($root_element);
$livre=$dom->createElement('livre',");
$root_element->appendChild($livre);
$titre=$dom->createElement('titre','Apprendre XML');
$livre->appendChild($titre);
$prix=$dom->createElement('prix','20');
$livre->appendChild($prix);
$monnaie=$dom->createAttribute('monnaie');
$montant=$dom->createTextNode('euro');
$monnaie->appendChild($montant);
$prix->appendChild($monnaie);
echo $dom->saveXML();
```

Parcourir un document XML/HTML

Méthodes principales

- `DOMElement getElementById(string elementId)`
- `DOMNodeList getElementsByTagName(string name)`

Exemple de parcours

dom-traverse.php

```php
function parcours_attributes($node) {
   if (!$node->hasAttributes()) return ;
   foreach ($node->attributes as $key => $subnode)
      echo " $key=".$subnode->nodeValue;
}
function parcours($node,$nbr) {
   if ($node->hasChildNodes())
      foreach (range(0,$node->childNodes->length-1) as $index) {
         $subnode=$node->childNodes->item($index);
         if ($subnode->nodeType==XML_ELEMENT_NODE) {
            foreach (range(0,4*$nbr) as $i) echo " ";
            echo "+".$subnode->nodeName." = ".trim($subnode->nodeValue);
            parcours_attributes($subnode); echo "\n";
            parcours($subnode,$nbr+1);
         }
      }
}
$dom=new DOMDocument();
$dom->load(__DIR__.'/bibliotheque.xml');
$liste=$dom->getElementsByTagName('livre');
foreach (range(0,$liste->length-1) as $ndx_livre) {
   $livre=$liste->item($ndx_livre);
   echo "livre $ndx_livre\n";
   parcours($livre,1);
}
```

Résultat du parcours

dom-traverse-out.txt

```
livre 0
     +titre = Apprendre XML en 10 mois
     +liste_auteurs =
          +auteur =  nom=Escargot prenom=Jean
          +auteur =  nom=Snail prenom=John
     +prix = 20
livre 1
     +titre = Learn Java in 10 seconds
     +liste_auteurs =
          +auteur =  nom=Fast prenom=Bob
     +prix = 25 monnaie=dollars
```

L'extension SimpleXML

L'extension `SimpleXML`

Fournit

- Des fonctions simples de chargement de fichiers ou chaînes XML.
- Une classe `SimpleXMLElement` dont les instances permettent d'accéder via leurs propriétés et méthodes aux contenu des éléments, de le modifier ou d'effectuer des recherches.
- Une classe `SimpleXMLIterator` pour itérer récursivement sur les éléments d'instances `SimpleXMLElement`.

Exemple de parcours avec SimpleXML

simplexml.php

```php
$xml=simplexml_load_file(__DIR__."/bibliotheque.xml");
print_r($xml);
foreach ($xml->livre as $livre) {
   echo "------------\n";
   echo "titre : ".$livre->titre."\n";
   $liste_auteurs=$livre->liste_auteurs->auteur;
   foreach ($liste_auteurs as $auteur)
      echo "auteur : ".$auteur["prenom"]
            ." ".$auteur["nom"]."\n";
   echo "prix : ".$livre->prix." "
            .$livre->prix["monnaie"]."\n";
}
```

Résutat du parcours avec SimpleXML (1/2)

simplexml-out1.txt

```
SimpleXMLElement Object
(
    [livre] => Array
        (
            [0] => SimpleXMLElement Object
                (
                    [titre] => Apprendre XML en 10 mois
                    [liste_auteurs] => SimpleXMLElement Object
                        (
                            [auteur] => Array
                                (
                                    [0] => SimpleXMLElement Object
                                        (
                                            [@attributes] => Array
                                                (
                                                    [nom] => Escargot
                                                    [prenom] => Jean
                                                )

                                        )

                                    [1] => SimpleXMLElement Object
                                        (
                                            [@attributes] => Array
                                                (
                                                    [nom] => Snail
                                                    [prenom] => John
                                                )

                                        )

                                )

                        )
```

Résutat du parcours avec SimpleXML (2/2)

simplexml-out2.txt

```
------------
titre : Apprendre XML en 10 mois
auteur : Jean Escargot
auteur : Brad Snail
prix : 20
------------
titre : Learn Java in 10 seconds
auteur : John Fast
prix : 25 dollars
```

XPath

Interrogation d'un document XML avec XPath

XPath - XML Path Language

- XPath est un élément du standard XSLT.
- XPath 3.0 est une recommendation du W3C (2014).

Syntaxe

- De type “chemin" pour identifier et parcourir des noeuds de document XML.
- Nombreuses autres fonctions pour différents types de données.

Les expressions XPath sont directement utilisables en

- JavaScript.
- PHP avec la classe `DOMXPath` ou la méthode `xpath` de `SimpleXMLElement`.
- Java ...

Exemples de requêtes XPath

`titre`	Tous les éléments `titre`.
`/`	L'élément racine.
`//titre`	Tous les noeuds `titre` sous le noeud courant.
`.`	Le noeud courant.
`..`	Le noeud parent.
`@`	Des attributs.
`//liste_auteurs/auteur`	Tous les éléments `auteur` à l'intérieur de `liste_auteurs`.
`//prix[@monnaie]`	Tous les éléments `prix` qui ont un attribut `monnaie`.
`//prix[@monnaie='dollars']`	Tous les éléments `prix` qui ont un attribut `monnaie` égal à `dollars`.

Exemple de traitement avec DOMXPath

dom-xpath.php

```php
$doc = new DOMDocument();
$doc->load("bibliotheque.xml");
$xpath = new DOMXpath($doc);
echo "traiter les titres\n";
$elements=$xpath->query("//titre");
if (!is_null($elements))
   foreach ($elements as $element) {
      echo "=> ". $element->nodeName. " : ";
      $nodes = $element->childNodes;
      foreach ($nodes as $node) echo $node->nodeValue. "\n";
   }
echo "\ntraiter les noms des auteurs\n";
$elements = $xpath->query("//auteur[@nom]");
if (!is_null($elements))
   foreach ($elements as $element)
      echo "=>". $element->nodeName. " " . $element->getAttribute("nom")."\n";
echo "\ntraiter les auteurs et prénoms des auteurs\n";
$elements = $xpath->query("//auteur[@prenom]");
if (!is_null($elements))
   foreach ($elements as $element)
      echo "=>". $element->nodeName. " " . $element->getAttribute("nom")
         ." ".$element->getAttribute("prenom")."\n";
```

Résultat du traitement avec DOMXPath

dom-xpath-out.txt

```
traiter les titres
=> titre : Apprendre XML en 10 mois
=> titre : Learn Java in 10 seconds

traiter les noms des auteurs
=>auteur Escargot
=>auteur Snail
=>auteur Fast

traiter les noms et prenoms des auteurs
=>auteur Escargot Jean
=>auteur Snail Brad
=>auteur Fast John
```

Utilitaire `xpath`

`xpath -e <query> <file>`

Utilitaire PERL pour interroger un fichier XML par requête XPath.

Options :

- `-e <query>` : Requête à exécuter.
- `-q` : Mode silencieux.
- `-s <suffix>` : Placer le suffixe (par défaut retour chariot) à la fin de chaque ligne.
- `-p <préfix>` : Placer le préfixe (par défaut rien) au début de chaque ligne.