PHP

Manipulation de XML avec DOM

DOM

Introduction

- DOM est une API standard :
  - Les méthodes et procédures de manipulation DOM sont exactement les mêmes dans tous les langages
  - Vous pouvez passer d'un langage à l'autre sans avoir à apprendre plusieurs noms de fonctions et plusieurs API

DOM

Structure générale

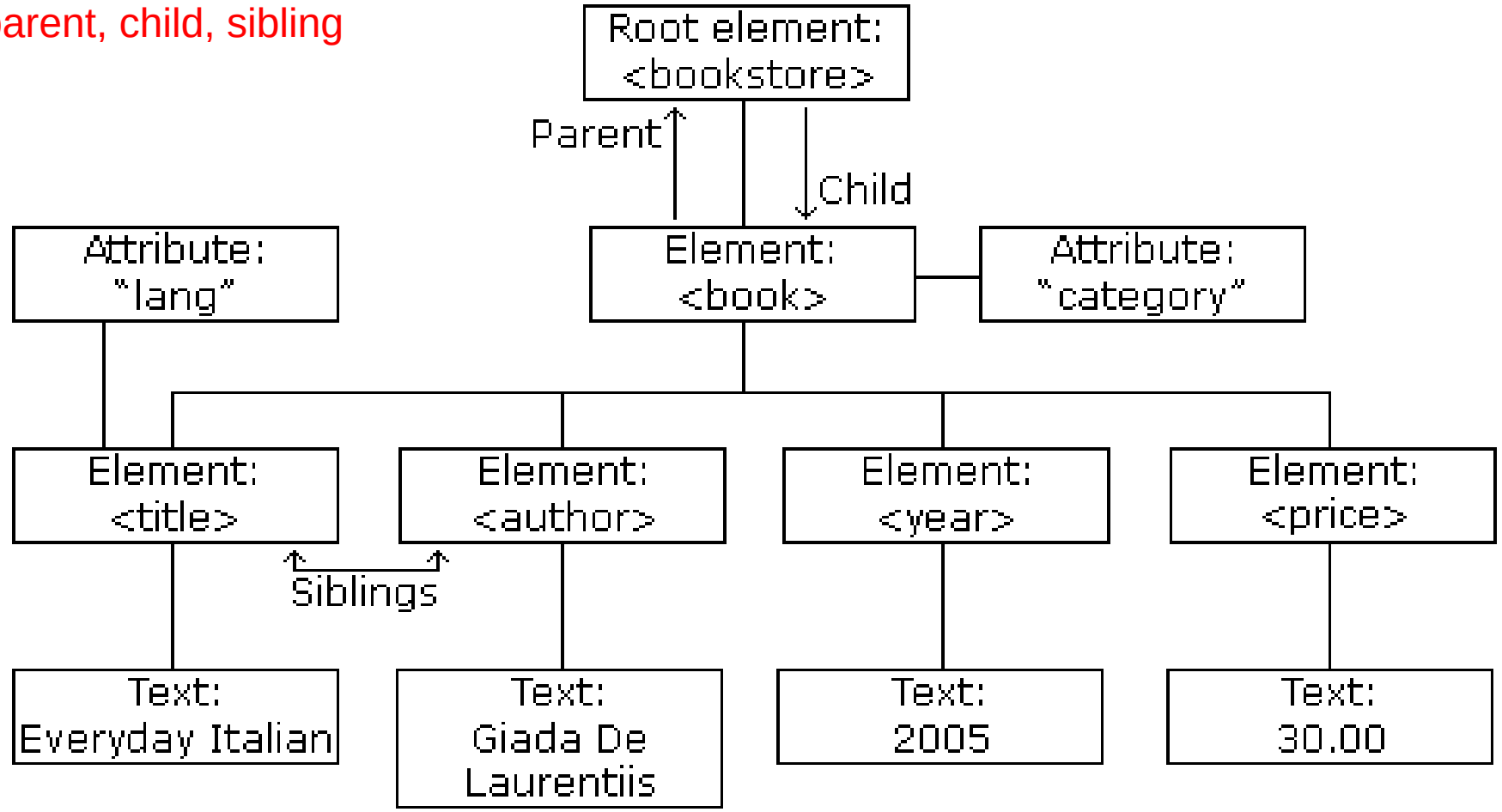
- DOM représente l'arbre XML sous la forme d'objets et relations entre eux
- Aussi bien le document que chaque noeud devient un objet à part entière avec sa définition, ses propriétés et ses méthodes
- Il devient alors facile de :
  - connaître les propriétés d'un objet
  - lire des objets
  - les modifier
  - les enregistrer dans un fichier
  - passer d'un objet à l'autre par une relation privilégiée

Exemple : books.xml

```
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday
         Italian</title>
        <author>Giada De Laurentiis
        </author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="WEB">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <author>Per Bothner</author>
        <author>Kurt Cagle</author>
        <author>James Linn</author>
        <author>Vaidyanathan Nagarajan</author>
        <year>2003</year>
        <price>49.99</price>
    </book>
    <book category="WEB">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

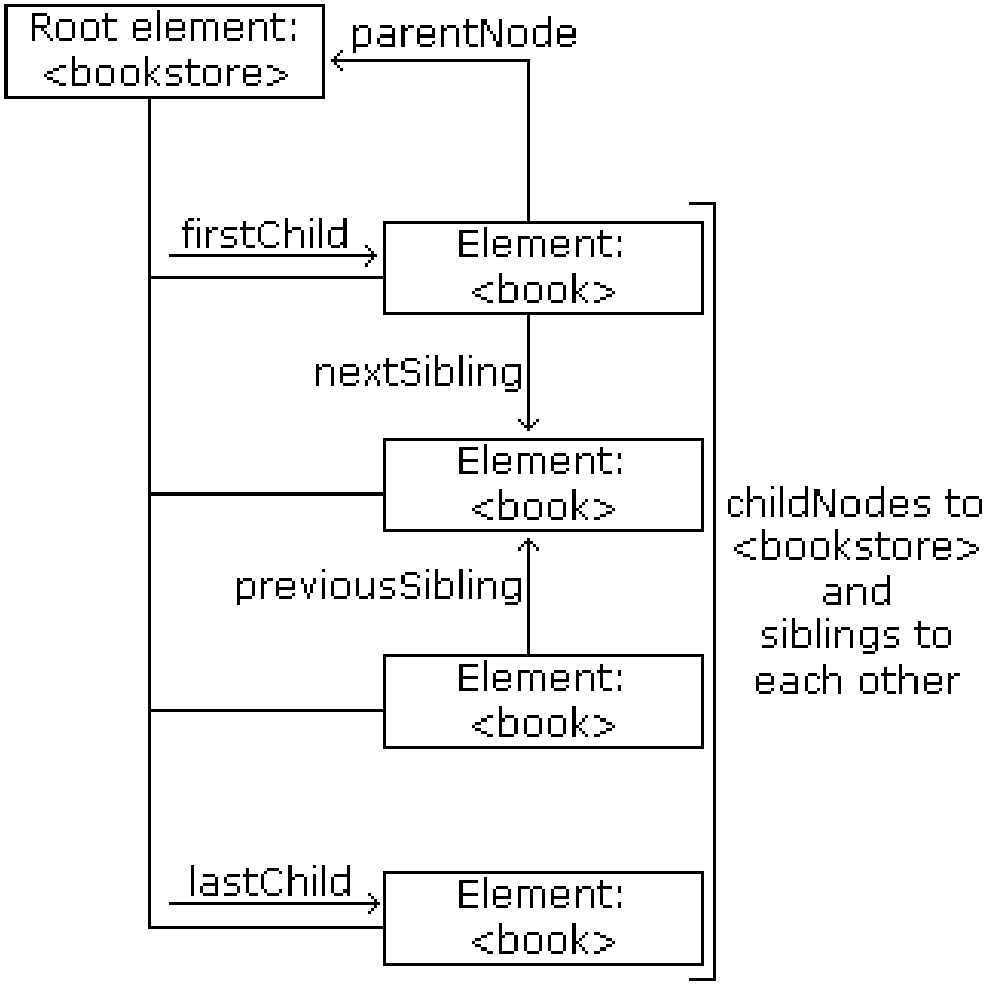
L’arbre DOM correspondant

Les nœuds éléments sont liés par des relations :
parent, child, sibling



L’arbre DOM correspondant

D’autres relations (i.e. méthodes) :



1. Les objets de l’extension DOM

Les objets de DOM

- Les principales classes sont les suivantes :
  - DomNode - objet nœud :
    - documents, éléments, nœuds textuels...
  - DomDocument - objet document
    - (hérite de DomNode)
  - DomElement - objet élément
    - (hérite de DomNode)
  - DomAttr - objet attribut
    - (hérite de DomNode)
  - DomNodeList - objet liste de DomNodes
    - (ce n'est pas un tableau PHP !)

2. L’objet document

- L’objet document représente l’arbre XML
  - Création d’un document
    - Il suffit d’instancier la classe DomDocument

```
<?php
    $document = new DomDocument();
?>
```

  - Chargement des données XML
    - Il suffit d’utiliser la méthode load()

```
<?php
    $document = new DomDocument();
    $document->load(‘monfichier.xml’);
?>
```

2. L’objet document

- Chargement depuis une chaîne XML

```
<?php
        $xml = "<livre><titre>PHP 5 avancé </titre></livre>";
        $document = new DomDocument();
        $document->loadXML($xml);
?>
```

- Chargement d’un fichier HTML
  - Il est possible de charger un fichier HTML en DOM et de le manipuler comme si c’était du XML

```
<?php
   $doc = DOMDocument::loadHTMLFile("filename.html");
   print $doc->saveHTML();

   $doc = new DOMDocument();
   $doc->loadHTMLFile("filename.html");
   print $doc->saveHTML();
?>
```

2. Le document DomDocument

- Enregistrement d'un document XML

```
<?php
    $dom->save('nouveauFichier.xml');
?>
```

- Enregistrement dans une variable

```
<?php
    $chaineXML = $dom->saveXML(); //saveXML() renvoie le contenu sous
    //la forme d'une chaîne de caractères
?>
```

- Exemple : load-save-xml.php

```
<?php
  $dom = new DomDocument();
  $dom->load('test.xml');
  $dom->save('nouveauFichier.xml');
?>
```

3. Lire un document

Rechercher et récupérer un élément

- Il y a plusieurs moyens pour trouver des éléments
- Accéder à l'élément racine

```
<?php
   $document = new DomDocument;
   $document->load('fichier.xml');
   $racine = $document->documentElement;
?>
```

3. Lire un document

- Accéder au document depuis un nœud
  - La fonction inverse de l'attribut documentElement s'appelle ownerDocument
  - Il permet de récupérer l'objet document à partir d'un nœud quelconque

```
<?php
    $document = new DomDocument();
    $document->load('monfichier.xml');
    $racine = $document->documentElement;
    $document = $racine->ownerDocument;
?>
```

3. Lire un document

Description d'un nœud

- Les nœuds sont des objets de la classe DomNode
- On peut connaître le type d'un nœud à partir de son attribut propriété : nodeType
- Il retourne un entier

Liste des types de nœuds

Valeur	Signification	Constante
1	élément	XML_ELEMENT_NODE
2	attribut	XML_ATTRIBUTE_NODE
3	nœud 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

3. Lire un document

Description d'un nœud

- Exemple

```
<?php
   $xml = "<livre>Alice</livre>";
   $document = new DomDocument();
   $document->loadXML($xml);
   //On se place au niveau du premier nœud
   $livre = $document->documentElement;
   echo $livre->nodeType; //Affiche 1
   $texte = $livre->firstChild;
   echo $texte->nodeType; //Affiche 3
?>
```

3. Lire un document

Description d'un nœud

- Nom d'un nœud :
  - deux attributs de DomNode : nodeName et tagName
- Exemple

```
<?php
   $xml = "<livre>Alice</livre>";
   $document = new DomDocument();
   $document->loadXML($xml);
   $livre = $document->documentElement;
   echo $livre->nodeName; //Affiche livre
   echo $livre->firstChild->nodeName; //Affiche #text
?>
```

3. Lire un document

- Contenu d'un nœud :
  - S'obtient avec l'attribut : nodeValue
  - Attention, un élément n'a pas de valeur, ce sont éventuellement ses fils, de type nœuds de texte, qui en ont
- Exemple : dom1.php

```
<?php
    $xml = "<livre type='conte'>Alice</livre>";
    $document = new DomDocument();
    $document->loadXML($xml);
    $livre = $document->documentElement;
    echo $livre->nodeName; //Affiche livre
    echo $livre->firstChild->nodeValue; //Affiche Alice
    $type = $livre->getAttributeNode('type');
    echo $type->nodeValue; //Affiche conte
?>
```

3. Lire un document

Navigation dans l'arbre

- Liste des nœuds
  - Dans les recherches futures, on va récupérer les nœuds de l'arbre XML dans une liste : objet de type : DomNodeList
  - On peut parcourir cette liste avec foreach()
  - Exemple

```
$nodeList; //objet de type DomNodeList;
Foreach($nodeList as $node){
    Print_r($node);
}
```

  - On peut également accéder à un item particulier à l'aide de la méthode : item() et d'un index numérique

```
$nodeList->item(0); //Premier nœud de la liste
```

  - La quantité de nœuds dans une liste peut être récupérée avec l'attribut length de l'objet DomNodeList

```
echo "Il y a ", $node->childNodes->length, " nœuds fils " ;
```

3. Lire un document

Navigation dans l'arbre

- Nœuds fils
  - La liste des nœuds fils d'un nœud peut être connue via l'attribut : childNodes du nœud père. L'objet renvoyé est un objet de type DomNodeList
  - Exemple : dom-fils.php

```
<?php
    $xml = "<livre><titre>PHP 5</titre><auteur>E. D</auteur><auteur>C.
     PdG</auteur></livre>";
    $document = new DomDocument();
    $document->loadXML($xml);
    $livre = $document->documentElement;
    //Affichage des fils de $livre
    foreach($livre->childNodes as $node){
        if($node->nodeType ==XML_ELEMENT_NODE){
            echo 'Balise <b>', $node->tagName, '</b><br>';
            echo 'Contenu : <b>';
            echo utf8_decode($node->firstChild->nodeValue),'</b><br>';
            }
    }
?>
```

3. Lire un document

Navigation dans l'arbre

- **Nœuds fils**
  - Il est possible d'accéder directement au premier ou au dernier nœud fils à l'aide des attributs : firstChild et lastChild
  - Exemple : dom-fils2.php

```
<?php
    $xml = "<versions>
        <version>3</version>
        <version>4</version>
        <version>5</version>
        </versions>";
    $document = new DomDocument();
    $document->loadXML($xml);
    $versions = $document->documentElement;
    //On récupère le premier fils
    $trois = $versions->firsChild;
    //On récupère le dernier fils
    $cinq = $versions->lastChild;
?>
```

- **Nœuds fils**
  - On peut tester la présence de nœuds fils à l'aide de la méthode : hasChildNodes()
  - Exemple : dom-fils3.php

```
<?php
    $xml = "<versions>
        <version>3</version>
        <version>4</version>
        <version>5</version>
        </versions>";
    $document = new DomDocument();
    $document->loadXML($xml);
    $versions = $document->documentElement;
    //Affichage des fils de $versions
    if($versions->hasChildNodes()){
        foreach($versions->childNodes as $node){
            if($node->nodeType == XML_ELEMENT_NODE){
                echo $node->tagName, ' : ';
                echo utf8_decode($node->firstChild->nodeValue), '<br>';
            }}}
?>
```

3. Lire un document

Navigation dans l'arbre

- Nœud parent
  - Le nœud parent d'un nœud peut être connu via l'attribut parentNode du nœuds fils. Il ,renvoie un objet de type DomNode

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

3. Lire un document

Navigation dans l'arbre

- Nœuds frères
  - Deux possibilités :
    - Soit remonter au père, puis lister les fils
    - Soit plus simplement : via les attributs previousSibling et nextSibling

```
Print_r($node->childNode->item(0));
Print_r($node->childNode->item(1));
//est équivalent à :
$node = $node->childNodes->item(0);
Print_r($node);
Print_r($node->nextSibling);
```

3. Lire un document

- Recherche d'élément par le nom
  - DomDocument::getElementsByTagName()
    - Recherche dans tout le document
  - DomElement::getElementsByTagName()
    - Recherche dans les descendants de l'élément considéré
  - Ces fonctions retournent un objet DomNodeList

3. Lire un document

- Exemple de getElementsByTagName()

```
<?php
    $dom = new DomDocument();
    $dom->load("test.xml");
    $listePays = $dom->getElementsByTagName('pays');
    foreach($listePays as $pays)
      echo $pays->firstChild->nodeValue . "<br />";
      echo "---<br />";
      $europe = $dom->getElementsByTagName('europe')->item(0);
      $listePaysEurope = $europe->getElementsByTagName('pays');
     foreach($listePaysEurope as $pays)
       echo $pays->firstChild->nodeValue . "<br />";
?>
```

3. Lire un document

- getElementsByTagName.php
  - Résultat :

France
Belgique
Espagne
Japon
Inde
Etats-Unis
Canada
Tunisie
Cameroun

France
Belgique
Espagne

3. Lire un document

Lire les attributs

- Modifier le fichier XML (à la main) pour ajouter des attributs donnant le régime politique des pays cités (on suppose que la DTD aura également été modifiée en conséquence, si l'on veut profiter de la validation) :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE continents SYSTEM "test.dtd">
<continents>
    <europe>
        <pays regime="republique">France</pays>
        <pays regime="monarchie constitutionnelle">Belgique</pays>
        <pays regime="monarchie constitutionnelle">Espagne</pays>
    </europe>
    <asie>
        <pays regime="empire">Japon</pays>
        <pays>Inde</pays>
    </asie>
</continents>
```

3. Lire un document

- Lire les attributs par getAttribute : getAttribute.php

```
<?php
    $listePays = $dom->getElementsByTagName("pays");
    foreach($listePays as $pays) {
        echo $pays->nodeValue;
        if ($pays->hasAttribute("regime")) {
          echo " - " . $pays->getAttribute("regime");
        }
       echo "<br />";
    }
>
```

- Pour éviter les erreurs, vérifier l'existence de l'attribut avec la fonction hasAttribute() qui prend aussi le nom de l'attribut en paramètre, et qui renvoie un booléen qui dit si l'attribut est présent ou pas

3. Lire un document

- Lire par getAttribute
  - Résultat

    France - republique

    Belgique - monarchie constitutionnelle

    Espagne - monarchie constitutionnelle

    Japon - empire

    Inde

3. Lire un document

Lire les nœuds textuels

- On l'a déjà vu, on peut récupérer la valeur d'un nœud textuel avec l'attribut nodeValue

```
<?php
    $pays = $dom->getElementsByTagName("pays");
    foreach($pays as $c) {
        echo $c->nodeValue . " " . $c->firstChild->nodeValue;
        echo "<br />";
    }
?>
```

3. Lire un document

- Lire les nœuds textuels
  - Résultat :

    France France

    Belgique Belgique

    Espagne Espagne

    Japon Japon

    Inde Inde

4. Modifier un document

Modification

- Voyons maintenant comment modifier les différents éléments d'un document XML déjà existant

Créer un noeud

- La méthode DomDocument::createElement permet très simplement de créer des éléments XML, en passant en paramètre le nom du nœud
  - Créer un élément

```
<?php
    $nouveauPays = $dom->createElement("pays");
?>
```

  - Créer un nœud textuel : si on veut ajouter un nœud textuel à cet élément (pour donner un nom de pays par ex.) : createTextNode

```
<php
    $nomPays = $dom->createTextNode("Royaume-Uni");
?>
```

4. Modifier un document

- Créer un nœud par copie d’un nœud existant

```
<?php
    $paysIdentique = $pays->cloneNode();
?
```

- Cette méthode accepte un argument facultatif, un booléen (FALSE par défaut). S'il est à TRUE, tout les nœuds fils seront copiés également, et donc toute une partie de l'arborescence peut être dupliquée par ce biais.

4. Modifier un document

- Modifier un attribut
  - Il nous faut maintenant ajouter un attribut à notre nouveau nœud, afin de préciser le régime politique : utiliser setAttribute
  - Création ou modification d'attribut

```
<?php
    $nouveauPays->setAttribute("regime", "monarchie
    constitutionnelle");
?>
```

  - On peut supprimer un attribut avec DomElement::removeAttribute (avec le nom de l'attribut en paramètre)

4. Modifier un document

Insérer un nœud dans le document

- Nous avons vu comment créer les éléments et les nœuds textuels, mais encore faut-il les placer dans le document XML, et au bon endroit
- L'insertion se fait par la méthode DomNode::appendChild qui ajoute le nœud passé en paramètre à la liste des enfants du nœud sur lequel il est appelé
- Insertion de nouveaux éléments

```
<?php
    $nouveauPays->appendChild($nomPays);
    $europe = $dom->getElementsByTagName("europe")->item(0);
   $europe->appendChild($nouveauPays);
?>
```

- ajoute le nœud textuel $nomPays au nœud $nouveauPays, et ajoute ensuite celui-ci au nœud "europe"

4. Modifier un document

Supprimer un nœud

- Utiliser DomNode::removeChild en l'appelant sur le parent du nœud à supprimer et en passant en paramètre une référence sur le nœud à supprimer
- Tous les descendants du nœud supprimé seront également exterminés

```
<?php
    $europe->removeChild($nouveauPays);
?>
```

6. Recherche Xpath

Initialisation

- Le moteur DOM permet de gérer des requêtes Xpath
- Il utilise un objet pour gérer la requête
- Cet objet est de la classe DomXpath qui attend un document DOM en argument

```
$document = new DomDocument();
$document->loadXml($xml)
$xpath = new DomXpath($document);
```

6. Recherche Xpath

Lancer une requête Xpath

- Pour faire une recherche, il faut faire appel à la méthode query()

```
<?php
   $xml = file_get_contents('fichier.xhtml');
   $xml = utf8_encode($xml);
   $document = new DomDocument();
   $document->loadXml($xml);
   $xpath = new DomXpath($document);
   //rechercher tous les formulaires à envoyer
   $result = $xpath->query("/html/body//form[@action='post']");
?>
```

6. Recherche Xpath

Lancer une requête Xpath

- Par défaut, la recherche est faite à partir de l'élément racine
- On peut toutefois préciser le nœud pour la recherche

```
$requete ="form[action='post']";
$reference = $document->documentElement->lastChild;
$result = $xpath->query($requete, $reference);
```

- Le résultat renvoyé est un objet liste de nœuds DOM classiques

```
$document = new DomDocument();
$document->loadXml($xml);
$xpath = new DomXpath($document);
//recherche tous les formulaires à envoyer avec la méthode POST
$result = $xpath->query("/html/body//form[@action='post']");
echo "Il y a ", $result->length, " formulaire(s) en POST";
```

6. Recherche Xpath

Lancer une requête Xpath

- Soit le fichier XML suivant : php5avance.xml

```
<?xml version="1.0" encoding="utf-8"?>
<livre>
<titre>PHP 5 avancé 5ème édition</titre>
<auteur>Eric Daspet</auteur>
<auteur>Cyril Pierre de Geyer</auteur>
<preface>Damien Séguy</preface>
<relecteur>Hugo Hamon</relecteur>
</livre>
```

- On cherche à récupérer tous les nœuds auteur et à afficher leur contenu texte dans une liste non ordonnée HTML : dom-xpath1.php

```php
<?php
    $xml = file_get_contents('php5avance.xml');
    $xml = utf8_encode($xml);
    $document = new DomDocument();
    $document->preserveWhiteSpace = false;
    $document->loadXML($xml);
    $xpath = new DomXpath($document);

    //recherche tous les noeuds <auteur>
    $result = $xpath->query("/livre/auteur");
    echo "Il y a ", $result->length, "auteur(s) pour ce livre :";
    echo "<ul>\n";
    foreach($result as $auteur) {
        echo "<li>$auteur->nodeValue</li>\n";
    }
    echo "</ul>\n;"
?>
```

7. Validation

- Validation par rapport à une DTD
  - Il est possible de vérifier la conformité d'un document avec un fichier grâce à la méthode booléenne validate()
  - Exemple :

```
<?php
    $dom = new DOMDocument;
    $dom->Load('book.xml');
    if ($dom->validate()) {
        echo "Ce document est valide !\n";
    }
?>
```

Validation

Exemple : test2.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE continents SYSTEM "test.dtd">
<continents>
  <europe>
    <pays>France</pays>
    <pays>Belgique</pays>
    <pays>Espagne</pays>
  </europe>
  <asie>
    <pays>Japon</pays>
    <pays>Inde</pays>
  </asie>
  <amerique>
    <pays>Etats-Unis</pays>
    <pays>Canada</pays>
  </amerique>
  <afrique>
    <pays>Tunisie</pays>
    <pays>Cameroun</pays>
  </afrique>
</continents>
```

Validation : Exemple

- test2.dtd

```
<!ELEMENT continents (europe?, asie?, amerique?, afrique?)>
<!ELEMENT europe (pays*)>
<!ELEMENT pays (#PCDATA)>
<!ELEMENT asie (pays*)>
<!ELEMENT amerique (pays*)>
<!ELEMENT afrique (pays*)>
```

- Exemple : dom-validate2.php

8. Transformation XML par XSLT

Initialisation

- Le moteur XSLT s'utilise via un objet propre, un peu comme Xpath
- Il faut donc commencer par instancier un objet de la classe XSLTProcessor
  - $moteurXslt = new xsltProcessor();

Chargement de la feuille de style

- Ceci se fait via la méthode importStylesheet() qui accepte un document DOM en argument
  - $moteurXslt = new xsltProcessor();
  - $style = new domDocument();
  - $style->load('style.xsl');
  - $moteurXslt->importStylesheet($style);

8. Transformation XML par XSLT

Transformation

- Elle se fait via la méthode transformToXml()
- Elle accepte un document DOM en argument et renvoie le XML produit

```
$moteurXslt = new XSLTProcessor();
$style = new domDocument();
$style->load('style.xsl');
$moteurXslt->importStylesheet($style);
$source = new DomDocument();
$source->load('source.xml');
echo $moteurXslt->transformToXML($source);
```

- Attention : aller dans php.ini et enlever le ; devant

```
extension=php_xsl.dll
```

2. Le document DomDocument

Import depuis SimpleXML

- Si vous avez utilisé SimpleXML pour lire rapidement un fichier et que vous souhaitez faire quelques manipulations DOM, il est possible d'importer l'objet SimpleXML pour construire un objet DOM de manière transparente

```
<?php
   $s = SimpleXML_load_file('fichier.xml');
   $dom = dom_import_simplexml($s);
   print $dom->ownerDocument->saveXML();
?>
```

- ownerDocument : retourne la racine du document
- saveXML() : crée une représentation XML depuis la représentation DOM