Introduction à XML [DVIR3]

Informatique et systèmes finalité Réseaux & télécommunication

2016-2017

XML

1. Introduction
2. Langage XML
3. Document XML
4. DTD : Déclaration de la structure du document
5. XML Schema
6. Adressage de fragments xml : Xpath
7. Contenu et présentation : CSS et XSL
8. Langage de transformation XSLT
9. Langage de formatage XSL-FO
10. PHP - XML

XML

- 1. Introduction
- 2. Langage XML
- 3. Document xml
- 4. DTD : Déclaration de la structure du document
- 5. XML Schema
- 6. Adressage de fragments xml : Xpath
- 7. Contenu et présentation : CSS et XSL
- 8.Langage de transformation XSLT
- 9.Langage de formatage XSL-FO
- **10. PHP - XML**

PHP - XML

- Une application PHP (version>5) peut
  - créer ,modifier, analyser, valider des documents XML,
  - transformer en fichier HTML, PDF ...

utilisation du couple PHP/XML

- Mise en place et exploitation de "Web Services"
- Production, diffusion et récupération de données selon un format d'échange standardisé
- Modularisation et configuration des scripts sur base du format XML
- Production et exploitation de flux RSS
- Mise en forme d'information dans de multiples formats (XML, HTML, WML, PDF, CSV, SVG, etc.)

Api SAX (Simple API for XML)

- - interface de programmation permettant la manipulation de données au format XML.

- - basée sur une approche événementielle, ce qui veut dire que l'analyse XML est régie par l'apparition ou non de certains événements.

Api SAX

- ```
  <?xml version="1.0"?>
  <auteurs>
    <nom>James Ellroy</nom>
    <nom>Jack London</nom>
    <nom>Jules Vernes</nom>
  </auteurs>
  ```
- Analyse SAX :

  ```
  start document
      |
      |-- start element 'auteurs'
      |         |
      |         |-->start element 'nom'
      |         |         |
      |         |         |-->characters 'James Ellroy'
      |         |
      |         |-->end element 'nom'
      |         |
      |         |-->start element 'nom'
      |         |         |
      |         |         |-->characters 'Jack London'
      |         |
      |         |-->end element 'nom'
      |         |
      |         |-->start element 'nom'
      |         |         |
      |         |         |-->characters 'Jules Vernes'
      |         |
      |         |-->end element 'nom'
      |
      |-- end element 'auteurs'
      |
  end document
  ```
- L'analyse SAX repose sur la prise en compte d'événements : début du document, fin du document, début d'un élément, fin d'un élément ... C'est au programmeur à tirer profit de l'apparition ou non de ces événements afin de retirer du document l'information qui l'intéresse.
  ```

Api DOM (Document Object Model)

- - interface spécifiée par le W3C permettant de créer, élaborer, modifier et parcourir une structure de document XML.

- - basée sur une représentation hiérarchique des éléments XML.

Api DOM

- ```
  <?xml version="1.0"?>
  <auteurs>
    <nom>James Ellroy</nom>
    <nom>Jack London</nom>
    <nom>Jules Vernes</nom>
   </auteurs>
  ```

- Représentation DOM :

  ```
  document
      |
      |--élément 'auteurs'
              |
              |-->élément 'nom'
              |          |
              |          |-->élément texte 'James Ellroy'
              |
              |-->élément 'nom'
              |          |
              |          |-->élément texte 'Jack London'
              |
              |-->élément 'nom'
                         |
                         |-->élément texte 'Jules Vernes'
  ```

-

  DOM transpose ici le document dans une structure arborescente composée d'un noeud document qui a pour enfant le noeud élément 'auteurs' qui a lui-même pour enfants les noeuds éléments 'nom'. Ceux-ci, à leur tour, contiennent des noeuds, mais cette fois de type 'texte'.

  Un langage de programmation, pour peu qu'il supporte l'interface DOM, va pouvoir analyser et traiter ces noeuds en les manipulant comme des objets. Il sera alors possible d'interroger leurs propriétés et de leur appliquer des méthodes.
  ```

Utilisation de SAX

- le document xml à traiter est fort volumineux
- la rapidité du traitement est prioritaire
- l'objectif du parsing XML est la sélection d'informations bien ciblées
- le traitement peut être réalisé en une seule passe

Utilisation de DOM

- des modifications doivent être effectuées dans le document XML
- la structure XML doit être exploitée finement
- l'objectif est de créer un document XML
- impose de construire un arbre en mémoire contenant l'intégralité des éléments du document.

Installation des outils XML/XSLT sous PHP5

Extension XSL

- implémente le standard XSL par défaut en PHP5
- fait des transformations XSLT à l'aide de la bibliothèque libxslt (http://xmlsoft.org/XSLT/).
- peut être activée très simplement -sous Windows- en enlevant le signe de commentaire ';' devant la directive '**extension=php_xsl.dll**' dans le fichier 'php.ini'.
- php_xsl.dll, doit être installé dans le répertoire extension précisée dans la directive *extension_dir* directive de ‘php.ini’

Sources : extension XSL

- http://be.php.net/manual/fr/ref.xslt.php
- http://www.zlatkovic.com/libxml.en.html
- http://xmlsoft.org/XSLT/

Extension DOM

- permet de manipuler des documents XML avec l'API DOM.

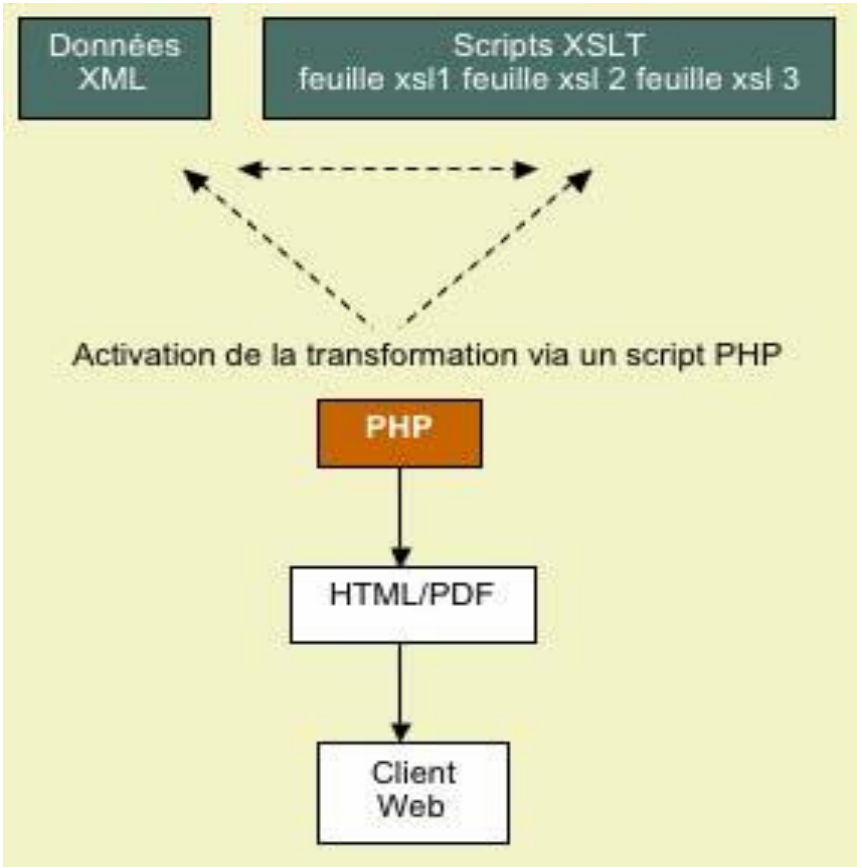
- PHP 5 inclut et active l'extension DOM par défaut.

Extension SimpleXML

- permet de manipuler sans prérequis particulier des documents xml simples.

- PHP 5 inclut et active l'extension SimpleXML par défaut.

PHP et XSLT


Données XML
Scripts XSLT
feuille xsl1 feuille xsl 2 feuille xsl 3
Activation de la transformation via un script PHP
PHP
HTML/PDF
Client Web

Transformation à partir d'un parseur externe

- Ex : msxsl.exe

[http://msdn2.microsoft.com/en-us/library/aa468552.aspx
http://msdn.microsoft.com/xml/
]

```
<?php
 passthru("c:/msxml4/msxsl.exe collection.xml      collection.xsl");
 ?>
```

[msxsl.php]

Transformation de base avec l'extension XSL de php5

```php
<?php
 $xml = new DomDocument;
 $xml->load('data.xml');
 $xsl = new DomDocument;
 $xsl->load('style.xsl');
 $proc = new XSLTProcessor();
 $proc->importStyleSheet($xsl);
 echo $proc->transformToXML($xml);
?>
```

[xslt.php]

Transformation en html

```php
<?php

$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configuration du transformateur
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attachement des règles xsl

$proc->transformToURI($xml, '/tmp/out.html');
?>
```

[xsl-xsltprocessor-transform-to-uri.php]

Exercice : transformation de base

Transformer en php

le document 'book.xml' en fonction de

la feuille de style 'book.xsl'pour obtenir

le document 'book.htm'

[book.php]

Exercice : transformation de base

Transformer en php

le document 'book.xml' en fonction de

la feuille de style 'book.xsl'pour obtenir

le document 'book.htm'

en appelant le parseur tiers ‘msxml.exe’

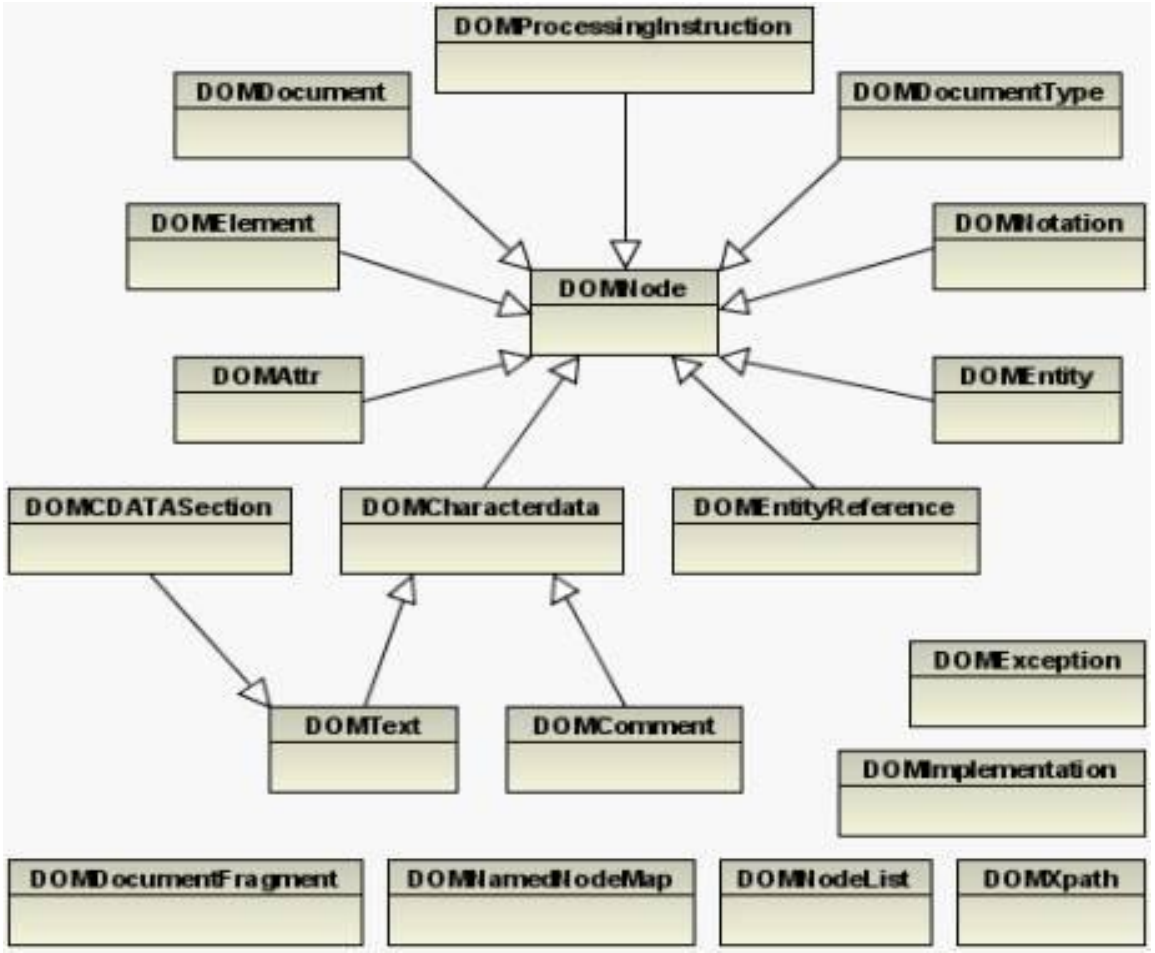
Modélisation DOM : les objets

- **DomNode**
  Représente un noeud. C'est le type d'objet principal (les autres en sont dérivés).
- **DomDocument**
  Représente le noeud document ('/').
- **DomElement**
  Représente un noeud 'élément'.
- **DomAttribute**
  Représente un noeud 'attribut'.

- DOMCharacterData

[http://be2.php.net/manual/fr/ref.dom.php/]

Modélisation DOM : les objets



http://www.laltruiste.com/coursphp/images/domxml.jpg

Classe : DOMDocument

- Propriétés :
  - version
    version xml du document
  - encoding
    type d'encodage pour le texte (jeu de caractères)
  - documentElement
    accès direct à l'élément racine d'un document

Classe : DOMDocument

- Méthodes :
  - load()
    charge un contenu XML à partir d'un fichier
    exemple :
    $doc = new DOMDocument('1.0', 'UTF-8');
    $doc->load('livres.xml');
  - loadXML(source)
    charge un contenu XML à partir d'une chaîne de caractères
    exemple :
    $doc = new DOMDocument();
    $doc->loadXML('<biblio><livre>...</livre></biblio>');
  - save(filename)
    sauvegarde une structure XML dans un fichier.
    exemple :
    $doc->save('data/biblio.xml');

Classe : DOMDocument

- Méthodes :
  - saveXML() : sauvegarde une structure XML dans un string
    exemple :
    echo $doc->saveXML() . "\n";
    # dump du document entier
    echo $doc->saveXML($author) . "\n";
    # dump d'un noeud particulier
  - createElement(name [, value]) : crée un nouvel élément
    exemple :
    $element = $doc->createElement('livre', 'test');
  - createTextNode(text) :crée un nouveau noeud texte (retourne un objet DOMText)
    exemple :
    $dom->createTextNode("Le petit prince")
  - getElementsByTagName(tag_name)
  - : récupération de noeuds sur base de leur nom (retourne un objet DOMNodeList)
    exemple :
    $livres = $dom->getElementsByTagName('livre');
  - validate() : valide un document en se basant sur sa DTD.

Classe : DomNode

- Propriétés :
  - nodeName
    retourne le nom d'un noeud
    exemple :
    if ($curNode->nodeName == "titre")
  - nodeType
    retourne le type d'un noeud
    exemple :
    if ($curNode->nodeType == XML_ELEMENT_NODE)
  - nodeValue
    retourne la valeur d'un noeud (en fct de son type)
    exemple :
    $content = trim($element->nodeValue);

Classe : DomNode

- Propriétés :
  - childNodes
    retourne les noeuds enfants (retourne un objet DOMNodeList)
    exemple :
    $root = $doc->documentElement;
    $children = $root->childNodes;
  - firstChild
    retourne le premier noeud enfant (retourne un objet DOMNode)
  - lastChild
    retourne le dernier noeud (retourne un objet DOMNode)
  - parentNode
    retourne le noeud parent (retourne un objet DOMNode)

Classe : DomNode

- Méthodes :
  - hasAttributes()
    vérifie si un noeud possède des attributs
    exemple :
    if (!$livre->hasAttributes())
  - hasChildNodes()
    vérifie si un noeud possède des enfants
    exemple :
    if (!$livre->hasChildNodes())
  - appendChild(object newnode)
    ajoute un noeud enfant
    exemple :
    $livres->appendChild($livre);
  - removeChild(object child_node)
    supprime un noeud d'une liste de noeuds enfants
    exemple :
    $rnode = $livre->removeChild($prix);

Classe : DomElement (étend DOMNode)

- Propriétés :
  - tagName
    le nom de l'élément
    exemple :
    if ($childNode->tagName == 'livre')

Classe : DomElement

- Méthodes :
  - getElementsByTagName(tag_name)
    récupération de noeuds sur base de leur nom (retourne un objet DOMNodeList)
    exemple :
    $auteurs = $livre->getElementsByTagName('auteur');
  - setAttribute(name, value)
    ajoute un nouvel attribut à un noeud
    exemple :
    $bibliographie->setAttribute("auteur", "Circum Net");
  - hasAttribute(attribute_name)
    teste si un attribut existe
    exemple :
    if ($livre->hasAttribute("id"))
  - getAttribute(attribute_name)
    récupère la valeur d'un attribut
    exemple :
    $id = $livre->getAttribute("id");

Opérations DOM :

- 1. Construction d’un arbre
- 2. Ajout d’un nœud
- 3. Validation d’un arbre
- 4. Parcours d’un arbre DOM
- 5. Exploitation d’XPATH

Construction d’un arbre

- **createComment(value)**
  création d'un commentaire
- createElement(node_name [,value])
  création d'un noeud élément / obtention d'un objet DomElement
- createTextNode(text)
  création d'un noeud texte
- **setAttribute(name, value)**
  ajout d'un attribut
- **appendChild(new_node)**
  ajout d'un enfant à un noeud
- saveXML([node])
  récupère l'arbre DOM sous forme d'une chaîne

Construction d’un arbre

- structure simple

```
<?php
# Création d'une nouvelle structure DOM
 $doc = new DOMDocument('1.0', 'UTF-8');

# Création et ajout d'un élément racine
$element = $doc->createElement('livre', 'contenu de test');
$doc->appendChild($element);

# Conversion de la structure DOM en une chaîne de caractères
# Affichage de la chaîne obtenue
echo $doc->saveXML();

# Résultat obtenu et affiché :
# <?xml version="1.0" encoding="UTF-8"?> <livre>contenu de test</livre>
?>
```

[arbre_constr1.php]

Construction d'un arbre

- structure élaborée (1/3)

```
<?php
# Création d'une nouvelle structure DOM
$dom = new DOMDocument('1.0', 'UTF-8');
# Création et ajout d'un commentaire
$comment = $dom->createComment("Test DOM/PHP");
$dom->appendChild($comment);
# Création et ajout d'un élément racine
$bibliographie = $dom->createElement("bibliographie");
$bibliographie->setAttribute("auteur", "Circum Net");
$dom->appendChild($bibliographie);
$livres = $dom->createElement("livres");
$bibliographie->appendChild($livres);
$livre = $dom->createElement("livre");
$titre = $dom->createElement("titre");
$titre->appendChild($dom->createTextNode("Clandestin"));
$livre->appendChild($titre);
$auteur = $dom->createElement("auteur");
$auteur->appendChild($dom->createTextNode("James Ellroy"));
```

Construction d’un arbre

- structure élaborée (2/3)

```
$livre->appendChild($auteur);
$editeur = $dom->createElement("editeur");
$editeur->appendChild($dom->createTextNode("Rivages"));
$livre->appendChild($editeur);
$livres->appendChild($livre);
$livre = $dom->createElement("livre");
$titre = $dom->createElement("titre");
$titre->appendChild($dom->createTextNode("Le Petit Prince"));
$livre->appendChild($titre);
$auteur = $dom->createElement("auteur");
$auteur->appendChild($dom->createTextNode(""));
$livre->appendChild($auteur);
$editeur = $dom->createElement("editeur");
$editeur->appendChild($dom->createTextNode("Gallimard"));
$livre->appendChild($editeur);
$livres->appendChild($livre);
```

Construction d’un arbre

- structure élaborée (3/3)

```
$livre = $dom->createElement("livre");
$titre = $dom->createElement("titre");
$titre->appendChild($dom->createTextNode("Barberousse"));
$livre->appendChild($titre);
$auteur = $dom->createElement("auteur");
$auteur->appendChild($dom->createTextNode("Michel Tournier"));
$livre->appendChild($auteur);
$editeur = $dom->createElement("editeur");
$editeur->appendChild($dom->createTextNode("Gallimard"));
$livre->appendChild($editeur);
$livres->appendChild($livre);
# Conversion de la structure DOM en une chaîne de caractères
# Affichage de la chaîne obtenue
echo $dom->saveXML();
# Sauvegarde l'arbre XML dans un fichier
$dom->save("xml/biblio.xml");
?>
```

Construction d’un arbre

- à partir d’une requête mySql (1/3)

```
<?php
# Connexion au serveur et sélection de la base
 $db = new PDO('mysql:host=localhost;dbname=biblio', 'root', '');
# Si tout va bien, on continue
$query = "SELECT * FROM livres";
$stm = $db->prepare($query);
$stm->execute();
$result = $stm->fetchAll(PDO::FETCH_ASSOC);

# Création d'une nouvelle structure DOM
$dom = new DOMDocument('1.0', 'UTF-8');
```

[Mysql.php]

Construction d'un arbre

- à partir d'une requête mySql (2/3)

```
# Création et ajout d'un élément racine
$biblio = $dom->createElement("bibliographie");
$biblio->setAttribute("auteur", "Circum Net");
$dom->appendChild($biblio);

# Traitement -une à une- des lignes du résultat
foreach ($result as $myrow) {
    # Ajout d'un noeud pour chaque ligne
    $book = $dom->createElement("livre");
    $biblio->appendChild($book);
     # Ajout d'un noeud enfant pour chaque champs
    foreach ($row as $fieldname => $fieldvalue) {
            # Création et ajout du noeud enfant
           $child = $dom->createElement($fieldname);
```

Construction d’un arbre

- à partir d’une requête mySql (3/3)

```
            $book->appendChild($child);
            # Ajout de la valeur du champs   en tant que noeud texte
            $value = $dom->createTextNode($fieldvalue);
            $child->appendChild($value);
        }
}
# Sauvegarde l'arbre XML dans un fichier
$dom->save("biblio.xml");
# Fermeture de la connexion
$db=NULL;
?>
```

Ajout d'un nœud (1/2)

```
<?php
Class books extends domDocument
{
function __construct()
{
parent::__construct();
}
function addBook($title, $author, $editor)
{
$eBook = $this->createElement("livre");
$eTitle = $this->createElement("titre");
$eTitle->appendChild($this->createTextNode($title));
$eBook->appendChild($eTitle);
$eAuthor = $this->createElement("auteur");
$eAuthor->appendChild($this->createTextNode($author));
$eBook->appendChild($eAuthor);
$eEditor = $this->createElement("editeur");
$eEditor->appendChild($this->createTextNode($editor));
```

Ajout d'un nœud (2/2)

```
$eBook->appendChild($eEditor);
$books=$this->documentElement->getElementsByTagName("livres");
$books->item(0)->appendChild($eBook);
}
}
 header("Content-type: text/xml");
$dom = new books();
$dom->formatOutput = true;
$dom->load("biblio.xml");
$dom->addBook("Lune sanglante", "James Ellroy", "Rivages");
print $dom->saveXML();
$dom->save("biblio.xml");
?>
```

Validation d'un document XML (1/4)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bibliographie SYSTEM "biblio2.dtd">
<bibliographie>
<livre>
<titre>Clandestin</titre>
<auteur>James Ellroy</auteur>
<editeur>Rivages</editeur>
<lieu>Paris</lieu>
<date>1990</date>
<pages>445</pages>
</livre>
<livre> <titre>Le Dahlia noir</titre> <auteur>James Ellroy</auteur> <editeur>Rivages</editeur> <lieu>Paris</lieu>
      <date>1990</date> <pages>472</pages>
</livre>
</bibliographie>
```

[biblio2.xml]

Validation d'un document XML (2/4)

```
<!ELEMENT bibliographie (livre*)>
<!ELEMENT livre (titre, auteur, editeur, lieu, date, pages)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT editeur (#PCDATA)>
<!ELEMENT lieu (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT pages (#PCDATA)>
```

[biblio2.dtd]

Validation d'un document XML (3/4)

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

[bibliographie.php]

Validation d'un document XML (4/4)

```
<?xml version="1.0" encoding="UTF-8"?>
xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="bibliographie">
<xs:complexType>
<xs:sequence>
<xs:element name="livre" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="titre" type="xs:string"/>
<xs:element name="auteur" type="xs:string"/>
<xs:element name="editeur" type="xs:string"/>
<xs:element name="lieu" type="xs:string"/>
<xs:element name="date" type="xs:gYear"/>
<xs:element name="pages" type="xs:short"/>
</xs:sequence></xs:complexType></xs:element>
</xs:sequence></xs:complexType></xs:element>
</xs:schema>
```

[biblio3.xsd]

Exercice :
validation par rapport au schéma

```
<?php
$dom = new DOMDocument;
$dom->load(‘xml/biblio3.xml');
if ($dom->schemaValidate('xsd/biblio3.xsd'))
{
 echo "Ce document est valide !\n";
 }
?>
```

[biblio3.php]

Parcours d'un arbre DOM

```
<?php
#Récupération et affichage des noeuds 'texte'  des  titres
$biblio = new DOMDocument();
$biblio->load('xml/biblio.xml');
$titres = $biblio->getElementsByTagName('titre');
foreach ($titres as $titre)
{
$val = $titre->nodeValue;
print utf8_decode($val) . "<br/>";
}
?>
```

[parcours.php]

Utilisation d’XPATH

Récupération du titre d'entête et du premier titre de niveau 2 d'une page en HTML

```
<?php
$dom = new DOMDocument();
$dom->loadHTMLFile("http://www.w3.org/");
$xp = new DOMXpath($dom);
$result = $xp->query("/html/head/title");
print $result->item(0)->firstChild->data;
$xp = new DOMXpath($dom);
$result = $xp->query("/html/body/h2[1]");
print $result->item(0)->firstChild->data;
?>
```

[xpath_use.php]

Webographie

http://www.w3schools.com/xml/default.asp
http://www.w3.org/TR/REC-xml
http://xmlfr.org
http://tecfa.unige.ch/guides/xml/
http://francexml.free.fr
http://www.xml.com
http://www.xml.org