XML DOM avec PHP

Code: php-dom

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/php-dom/php-dom.html**

url: **http://tecfa.unige.ch/guides/tie/pdf/files/php-dom.pdf**

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 0.3 (modifié le 6/3/07 par DKS)

Prérequis

Module technique précédent: xml-dom (Concepts XML de base)

Module technique précédent: xml-tech (Introduction technique à XML)

Module technique précédent: php-intro (Introduction à PhP)

Module technique précédent: php-libs (Objets PHP)

Module technique précédent: php-xml (Introduction à XML avec PhP)

Abstract

Petite introduction au DOM du W3C avec PHP

Très peu complet pour le moment !!

Objectifs

1. Introduction générale au DOM
2. Utilisation simple du DOM avec PHP

1. Table des matières détaillée

2. Document Object Model (DOM) définition

Le Modèle Objet de Document (DOM) est une interface de programmation d'application (API) pour des documents HTML valides et XML bien-formés. Il définit la structure logique d'un document (au sens large tu terme!) et la manière d'y accéder et de le manipuler.

Avec le DOM, les programmeurs peuvent construire des documents, naviguer dans leur structure ainsi qu'ajouter, modifier ou effacer des éléments et leur contenu. Tout ce qui se trouve dans un document HTML, ou XML, peut être touché, modifié effacé ou ajouté en utilisant le Modèle Objet de Document, à quelques exceptions près...

l'API DOM de PhP qui ressemble à celui d'autres langages de programmation qui implémentent DOM level 2 défini par le W3C.

Il existe 3 classes principales qu'on introduira dans la suite:

- DOMNode
- DOMElement (étend DOMNode)
- DOMDocument (étend DOMNode)

Il existe 12 autres classes ....

2.1 Le modèle de structure du DOM

Le contenu d'un arbre XML

- ***Document*** --> Element (un, au maximum), ProcessingInstruction, Comment, DocumentType (un, au maximum)
- ***DocumentFragment*** --> Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- ***DocumentType*** --> aucun enfant
- ***EntityReference*** --> Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- ***Element*** --> Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
- ***Attr*** --> Text, EntityReference
- ProcessingInstruction --> aucun enfant
- Comment --> aucun enfant
- Text --> aucun enfant
- CDATASection --> aucun enfant
- Entity --> Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- Notation --> aucun enfant

Eléments supplémentaires:

- une interface ***NodeList*** pour la gestion de listes ordonnées d'objets Node (typiquement on a cela après avoir fait une requête XPath etc.)
- une interface ***NamedNodeMap*** pour la gestion de noeuds référencé par leur nom d'attribut,

2.2 Petite présentation des classes principales

A. La classe DOMDocument

- Sert par exemple à créer une réprésentation DOM d'un fichier xml

  Voici l'essentiel (Il existe pleins d'autres methodes et propriétés !):

Création d'un objet DOM

```
Syntaxe: new DOMDocument ()
ex: $doc = new DOMDocument();
```

Remplir un objet DOM à partir d'une chaîne de caractères

```
Syntaxe: DOMDocument->loadXML()
ex: $dom->loadXML("<?xml version=\"1.0\?> <hello> </hello> ?>");
```

Remplir un objet DOM à partir d'un fichier

```
Syntaxe: DOMDocument->load()
ex: $dom->load("test.xml");
```

DOMDocument étend la classe DOMNode
(autrement hérite de ses attributs et méthodes présentés ci-dessous).

B. La classe "DomNode"

DOMNode représente un noeud (toutes les sortes) dans l'arbre du document.

Utilité de la classe DomNode

- tester et extraire nom, valeur et type d'élément (genre "balise", "commentaire", "CDATA (texte)",
- extraire ou modifier ses composants (noeds et attributs).

Quelques Propriétés:

- nodeName = le nom du noeud
```
$el = .... // $el contien un noeud XML
$el->nodeName; // retourne son nom
```
- nodeValue = contenu (si c'est un text node)
- nodeType = parmi une liste des types de constantes qui définit le type d'objet qui est le noeud
- attributes = donne les attributs
- childNodes = donne une DOMNodeList d'éléments

Quelques Méthodes:

- hasChildNodes() = indique s'il existe des nodes enfants
- hasAttributes() = indique s'il existe des attributs
```
if ($el->hasAttributes()) echo "super, on a des attributs"
```

C. La classe DomElement

- DOMElement représente un élément dans un document HTML ou XML.
- Cette classe hérite toutes les méthodes de DomNode et ajoute les choses applicables aux éléments seulement (éléments = balise ouvrante, fermante et son contenu).

Quelques Propriétés:

- tagName = le nom de la balise (identique à nodeName hérité)

Quelques Méthodes:

- getElementsByTagName() = retourne une liste d'éléments

```
$exercise = ...  // $exercise est un node de type DOMElement
$exercise->getElementsByTagName("staf");
          // retourne la liste d'élément "stafs"
```

- getAttribute() - Returns value d'un attribut

```
$exercice->getAttribute("id"); // retourne la valeur de l'attribut id
```

D. Les NodeList

- la méthode item() retourne le nième élément d'une nodeList

```
$doc = new DOMDocument();
$doc->load("travaux.xml") or die("*** There is no such file ***");
$student_list = $doc->getElementsByTagName('student');
$student      = $student_list->item(0);
```

- on utilise `foreach` pour traverser les éléments d'une liste

```
$exercises = $student->getElementsByTagName('exercise');
foreach ($exercises as $exercise) {
   $staf = $exercise->getElementsByTagName("staf")
   ...
}
```

`Syntaxe: foreach ($NodeList as $Noeud) { ..... }`

C.f. le manuel pour une définition de foreach. C'est la seule façon de faire, oubliez les autres boucles !!

Exemple 2-1: Extraction des attributs

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/attribweb.php**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/attribweb.phps**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/attribweb.text (source)**

- Ce bout de code montre comment accéder à des éléments par leur nom et ensuite comment extraire la valeur d'un attribut selon son nom.

```
<?php

$doc = new DOMDocument();
$doc->load("test.xml") or die("*** There is no such file ***");

$nodelist = $doc->getElementsByTagName("para");

foreach ($nodelist as $node) {
  $content = $node->nodeValue;
  $id = $node->getAttribute('id');
  printf("ID = %s : %s <p>", $id, $content);
}
?>
```

Exemple 2-2: Extraction des éléments et attributs d'un arbre

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/show_xml.php**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/show_xml.phps**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/show_xml.text**

Copiez ces fonctions pour debugger votre code (ou faites mieux !)

```
function debug_node_list ($list, $infos) {
  echo "<br>This list has " . $list->length . " elements\n";
  if (count($list) == 0) echo "<br>This list seems to be empty\n";
  else {
    echo "<ol>\n";
    foreach ($list as $el) {
      echo "<li>\n"; debug_element_rec ($el,""); echo "</li>\n";
    }
    echo "</ol>\n"; }
 }

function debug_element_rec ($el,$info) {
  $nodeType = $el->nodeType;
  echo "<br> Node Type = " . $nodeType . "\n";
 // ici on va faire des choses en fonction du type de noeud
 // (il en manquent certains ... comme les PI ou les entités)
  switch ($nodeType) {
  case XML_TEXT_NODE: {
    echo "<br>Node Value =" . $el->nodeValue;
    break;
  }
```

```
  case XML_ELEMENT_NODE: {
    echo "<br>Element Name =" . $el->nodeName;
    break;
  }
  case XML_ATTRIBUTE_NODE: {
    echo "<br>Attribute Name =" . $el->nodeName;
    break;
  }
  default:
    echo "<br>Strange Node !!! TYPE of element = " . gettype($el) . "\n";
  }
  if ($el->hasAttributes()) {
   echo "<br>Attributes: ";
   foreach ($el->attributes as $attr) {
     echo "[ " . $attr->name . " = " . $attr->value . " ]\n";
   }
  }
  if ($el->hasChildNodes()) {
    debug_node_list ($el->childNodes, "Children of " . $el->nodeName);
  } }

// ici on charge une fichier XML et on lance la fonction ci-dessus

$doc = new DOMDocument();
$doc->load("travaux.xml") or die("*** There is no such file ***");
# show a few things
debug_node_list ($student_list,"The tree");
```

Exemple 2-3: (complet) Extraction des travaux staf14 d'un étudiant staf-g

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/extract_travaux.php**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/extract_travaux.phps**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/extract_travaux.text (source)**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/travaux.xml**

url: **http://tecfa.unige.ch/guides/php/examples/xml_with_dom/travaux.dtd**

```
// ********************************** Aux functions

function get_value_of_first_named_child ($el, $child_name) {
  // debug_element ($el, "get_first_element for " . $child_name);
  $liste = $el->getElementsByTagName($child_name);
  return $liste->item(0)->nodeValue;
}

// ********************************** Main program
# create DOM object (DOMDocument class) and load xml
$doc = new DOMDocument();
$doc->load("travaux.xml") or die("*** There is no such file ***");

# get student nodes
$student_list = $doc->getElementsByTagName('student');
$student      = $student_list->item(0);
```

```
if ($student->nodeType == XML_ELEMENT_NODE) {

# get some data from personal-data
  $first_name    = get_value_of_first_named_child ($student, 'first-name');
  $family_name   = get_value_of_first_named_child ($student, 'family-name');
  // a less elegant way to do it :)
  $home_url_L    = $student->getElementsByTagName('home-url');
  $home_url      = $home_url_L->item(0)->nodeValue;

    // echo this information
    echo "<h2><a href='$home_url'>". $first_name . " " . $family_name . "</a></h2>\n";

    // get the student's list of exercices

    $exercises = $student->getElementsByTagName('exercise');

    // extract staf-14 exercises, should rather be done with xpath
    foreach ($exercises as $exercise) {
      $staf      = $exercise->getElementsByTagName("staf")->item(0);
      $attr_staf = $staf->getAttribute("no");
      $attr_no   = $staf->getAttribute("ex-number");
      $title     = get_value_of_first_named_child($exercise,"title");
      $url       = get_value_of_first_named_child($exercise,"url");
      if ($attr_staf == "14") {
 echo "<b>Staf14-ex".$attr_no.": </b> <a href='".$url."'>".$title."</a><br>\n";}
    }
}
```

3. XPath avec le DOM

Au lieu de programmer des boucles/conditions compliquées on peut aussi extraire des éléments avec une expression XPath

Exemple 3-1: DOM et XPath

url: **http://tecfa.unige.ch/guides/php/examples/xpath/simple-xpath2.php**

url: **http://tecfa.unige.ch/guides/php/examples/xpath/simple-xpath2.phps**

url: **http://tecfa.unige.ch/guides/php/examples/xpath/simple-xpath2.text**

url: **http://tecfa.unige.ch/guides/php/examples/xpath/**

```
$dom_object = new DomDocument();
$dom_object->load("student.xml");
// create DOMXPath object with our DOMObject
$xpath = new Domxpath($dom_object);

// Get all exercise nodes that are <staf no="14">
// The $result is a DOMNodeList and not just a a simple array.

$result = $xpath->query("//student/exercises/exercise/staf[@no='14']/..");

print ("<ul>");
foreach ($result as $exercise) {
  print ("<li>");
  $title = $xpath->query ("title", $exercise);
  print ($title->item(0)->nodeValue . "</li>\n");
}
print ("</ul>");
```