Last updated: Fri, 26 Feb 2010

The DOMNodeList class

Class synopsis

DOMNodeList

DOMNodeList {

/* Properties */

readonly public int $length ;

/* Methods */

DOMNode DOMNodelist::item (int *$index*)

}

Properties

length

The number of nodes in the list. The range of valid child node indices is 0 to *length - 1* inclusive.

Table of Contents

User Contributed Notes
DOMNodeList

walt at brookhouse dot co dot uk
16-Oct-2009 04:53

```
Below a.lepe says, "Using NodeLists to store results of an XQuery..."

To clarify, this should say, "...the results of an XPath query...".

XQuery contains XPath but has additional functionality such as FLOWR expressions. PHP does not directly support XQuery (yet!).
```

A dot Lepe dev+php at alepe dot com
02-Oct-2009 05:53

```
Using NodeLists to store results of an XQuery is not very functional if you are planning to do some operations with those Nodes. Instead,
you can use an array in this way:

<?php
$nodelist = $xpath->query($xpath, $DOMdoc);
foreach ($nodelist as $node) {
  // you can add here a special conditions, as searching for
  // regular expression matches in your nodes names/values/attributes
  // which can not be achieved with XSLT 1.0
  if( .... )  $nodearr[] = $node;
}
?>

This way, you can access nodes like (for example):

<?php
$nodearr[0]->nodeValue;
?>

solving the problem described by "saad0105050" (06-Jul-2008)

Even more, you can perform other operations in your results
as removing elements, reversing order, randomize, callbacks, etc.
```

nascoedu
18-Sep-2008 10:44

```
Get some nodes from an existing xml file and add them to a new file:

<?php
$dom = new DOMDocument;
$dom->load($xmlsource);

/*create the xPath object _after_  loading the xml source, otherwise the query won't work:*/
$xPath = new DOMXPath($dom);

/*now get the nodes in a DOMNodeList:*/
$nodeList = $xPath->query($anXPathExpr);

/*create a new DOMDocument and add a root element:*/
$newDom = new DOMDocument('1.0','UTF-8');
$root = $newDom->createElement('root');
$root = $newDom->appendChild($root);

/* append all nodes from $nodeList to the new dom, as children of $root:*/
foreach ($nodeList as $domElement){
 $domNode = $newDom->importNode($domElement, true);
 $root->appendChild($domNode);
}
/*please note: importNode does not cast a DOMElement to a DOMNode!*/

/*save the new dom */
echo 'Wrote: ' . $newDom->save('newDOM.xml') . ' bytes';
?>

the following won't work and you'll end up with a DOMException: 'Wrong Document Error' (at least I did):

<?php
foreach ($nodeList as $element){
 $root->appendChild($element);
}
?>

Cheers! ;-)
---
WindowsXP, WAMP5 (appache 2.2.6), PHP 5.2.5, DOM/XML API Version 20031129, libxml 2.6.26
---
```

c dot 1 at smithies dot org
27-Aug-2008 10:26

```
I doubt the accuracy of what saad105050 wrote below. In particular, in his example, he seems to assume that
$element->getElementsByTagName() will return NULL if there are no matching nodes. This is not what happens; as per the documentation, a
DOMNodeList is returned with the length property zero.
```

bobvandell at hotmail dot com
26-Aug-2008 04:12

```
That's actually incorrect. You can use function results as objects. It makes building an API for your database very clean and neat. For
example:

Our code:

$articles = Node::screate('tags', 123456)->assets('like:title:test')->articles;

We use the above code to get articles that are linked to assets that are linked to a specific tag in our database.
```

rohypnol
14-Jul-2008 07:28

```
That's simply because up to and including PHP 5 (latest version at this time) you can't use a function result as an object even if it is
an object.

Work:
1) $obj->sub_obj->method();
2) $sub_obj = $obj->get_sub_obj();
 $sub_obj->method();

Don't work:
1) $obj->get_sub_obj()->method();
2) echo $obj->get_sub_obj()->property;
```

saad0105050 at gmail dot com
06-Jul-2008 05:10

```
Problem with accessing return value of DOMNodeList::item() method.

When using the following code segment:
---------------------

$messageNodes = $doc->getElementsByTagName( "message" );
if( $messageNodes != NULL )
{
  $messageStr = $messageNodes->item( 0 )->firstChild->nodeValue;
}
---------------------
PHP gives an error: "syntax error, unexpected T_OBJECT_OPERATOR"

But if I modify it like the following:
---------------------
$messageNodes = $doc->getElementsByTagName( "message" );
if( $messageNodes != NULL )
{
  $messageNode = $messageNodes->item( 0 );
  $messageStr =  messageNode->firstChild->nodeValue;
}
---------------------
There is no error.

I guess, the return element of DOMNodeList::item() function can not be readily accessed via "->" operator.
```

mark at codedesigner dot nl
05-Jun-2008 10:14

```
$newNode = $dom->createElement('newNode') ;
foreach ($nodeList as $node) {
echo $node->nodeValue ;
$newNode->appendChild($node) ;
}

the problem lies with the fact that foreach works on a copy of your object. The solution is simple, add & to $node

$newNode = $dom->createElement('newNode') ;
foreach ($nodeList as &$node) {
echo $node->nodeValue ;
$newNode->appendChild($node) ;
}
```

a dot buffa at sns dot it
29-May-2008 11:28

```
I agree with drichter at muvicom dot de.

For istance, in order to delete each child node of a particular parent node,
<?php

while ($parentNode->hasChildNodes()){
$domNodeList = $parentNode->childNodes;
$parentNode->removeChild($domNodeList->item(0));
}

?>

In other word you have to uptade the DomNodeList on every iteration.

In my opinion, the DomNodeList class is useless.
```

c dot 1 at smithies dot org
23-May-2008 12:43

```
You can modify, and even delete, nodes from a DOMNodeList if you iterate backwards:

$els = $document->getElementsByTagName('input');
for ($i = $els->length; --$i >= 0; ) {
$el = $els->item($i);
switch ($el->getAttribute('name')) {
  case 'MAX_FILE_SIZE' :
   $el->parentNode->removeChild($el);
  break;
  case 'inputfile' :
   $el->setAttribute('type', 'text');
  //break;
}
}
```

drichter at muvicom dot de
14-May-2008 01:11

```
Addition to my first note:

An traditional for-loop does not allow you to change the DOM-tree while looping - the effects are the nearly the same as with foreach. So
you have to collect the nodes in an array and do the tree-altering stuff within a second loop (looping the array this time ...)
```

drichter at muvicom dot de
14-May-2008 11:56

```
I have done some testing and have found 2 results:
(My System: Win XP with PHP 5.2.1)

1) Iteration with foreach does function correctly as "james dot j dot hackett at gmail dot com" writes, _if_ you only do readonly stuff
with foreach or minor writings of some attributes.

2) foreach does not function, if you are doing some DOM-Operations while iterating. In my situation it was adding the iterated $node as an
child to an new node:

$newNode = $dom->createElement('newNode') ;
foreach ($nodeList as $node) {
echo $node->nodeValue ;
$newNode->appendChild($node) ;
}

This only gives you the first element ...

I'm interpreting it as an confusing but correct behavior because of the changes within the $dom-object while appending the node at an
additional place ...

So, if you want to do something like 2) use for, length and item() :)
```

james dot j dot hackett at gmail dot com
08-May-2008 04:47

```
In Response to 'kassah at gmail'

You don't need to convert a DOMNodeList to an array in order iterate through it using 'foreach'.  You can use foreach directly with the
DOMNodeList.

$nodeList = $someDomDocument->getElementsbytagname('user');

foreach ($nodeList as $node) {
  echo $node->nodeValue;
}
```

kassah at gmail dot com
05-May-2008 12:06

```
// Converts a DOMNodeList to an Array that can be easily foreached
function dnl2array($domnodelist) {
  $return = array();
  for ($i = 0; $i < $domnodelist->length; ++$i) {
    $return[] = $domnodelist->item($i);
  }
  return $return;
}
```

brack at wjp dot de
21-Apr-2008 09:35

```
In PHP 5.2.5 (Windows) it is not possible to iterate correctly over the DOMNodeList object returned by DOMNode->childNodes using foreach.
Instead I had to use the for loop in conjunction with the item() method of DOMNodeList for iterating over all child nodes correctly.

I don't know whether this is really a bug, but apparently it is.
```