DOM and PHP

another article on DOM and PHP

Loading XML via DOM

```php
.sourceforge = 'sitemap.xml';

$dom = new DomDocument();
$dom->load($source);

// load as string
$dom2 = new DomDocument();
$dom2->loadXML(file_get_contents($source));
```

DOM and XPATH

```php
.sourceforge = 'sitemap.xml';
$dom = new DomDocument();
$dom->load($source);

$xpath = new DomXPath($dom);

$xpath->registerNamespace('c',
'http://www.google.com/schemas/sitemap/0.84');

$result = $xpath->query("//c:loc/text()");
echo $result->length.'
';
//echo $result->item(3)->data;
foreach($result as $b) {
    echo $b->data.'
';
}
```
# Adding Data via DOM

To add new data to a loaded DOM document, we need to create new `DomElement` objects by using the [DomDocument::createElement()](https://www.php.net/manual/en/class.domdocument.php), [DomDocument::createElementNS()](https://www.php.net/manual/en/class.domdocument.php) and [DomDocument::createTextNode()](https://www.php.net/manual/en/class.domdocument.php) methods. In the following we will add a new URL to our URL set:

```php
$sitemap = 'sitemap.xml';
$dom = new DomDocument();
$dom->load($source);

// url element
$url = $dom->createElement('url');

// location
$loc = $dom->createElement('loc');
$text = $dom->createTextNode('http://php-coding-practices.com/article/');
$loc->appendChild($text);

// last modification
$lastmod = $dom->createElement('lastmod');
$text = $dom->createTextNode('2007-04-20T10:24:32+00:00');
$lastmod->appendChild($text);

// change frequency
$changefreq = $dom->createElement('changefreq');
$text = $dom->createTextNode('weekly');
$changefreq->appendChild($text);

// priority
$priority = $dom->createElement('priority');
$text = $dom->createTextNode('0.3');
$priority->appendChild($text);

// add the elements to the URL
$url->appendChild($loc);
$url->appendChild($lastmod);
$url->appendChild($changefreq);
$url->appendChild($priority);

// add the new URL to the root element (urlset)
$dom->documentElement->appendChild($url);

echo $dom->saveHtml();
```

**output:**

```
....
```
Adding Attributes To Nodes

Via `DomDocument::setAttribute()` we can easily add an attribute to a node object. Example:

```php
1. $url = $dom->createElement('url');
2. ...
3. $url->setAttribute('meta:level','3');
```

Here we set a fictive meta:level attribute with the value 3 to our url NodeElement from above.

Moving Data

Moving data is not as obvious as you might expect, as the DOM extension does not provide a real method that takes care of that, explicitly. Instead we will have to use a combination of `DomDocument::insertBefore()`. As an example, suppose we want to move our new url from above just before the very first url:

```php
1. $xpath = new DomXPath($dom);
2. $xpath->registerNamespace("c","http://www.google.com/schemas/sitemap/0.84");
3. $result = $xpath->query("//c:url");
4. $result->item(1)->parentNode->insertBefore($result->item(1),$result->item(0));
```
DomDocument::insertBefore() takes two parameters, the new node and the reference node. It inserts the new node before the reference node. In our example, we insert the second url ($result->item(1)) before the first one ($result->item(0)).

I hear you asking why we use DomDocument::insertBefore() on the $result->item(1)->parentNode node. Couldn't we just as easily use simply $result->item(0)? No of course not, as we need to execute DomDocument::insertBefore() on the root element, urlset, and not a specific url (look at our xpath query).

We could use the following code which is perfectly valid and gets us the same results, though:

We could use the following code which is perfectly valid and gets us the same results, though:

```php
1. $result->item(0)->parentNode->insertBefore($result->item(1),$result->item(0));
```

If we wanted to append the first url at the bottom of the sitemap, the following code is the way to go:

```php
1. $result->item(0)->parentNode->appendChild($result->item(0));
2. // or $dom->documentElement->appendChild($result->item(0)); respectively
```

Easy is it not? :) DomDocument::insertBefore() and DomNode::appendChild() automatically move (and not copy and then move) the corresponding nodes. If you wish to clone a node first before moving it, use DomNode::cloneNode() first:

```php
1. $source = 'sitemap.xml';
2. $dom = new DomDocument();
3. $dom->load($source);
4. $xpath = new DomXPath($dom);
5. $xpath->registerNamespace("c","http://www.google.com/schemas/sitemap/0.84");
6. $result = $xpath->query("/c:url");
```
8. $clone = $result->item(0)->cloneNode(true);
9. $result->item(4)->parentNode->appendChild($clone);
10. echo $dom->saveXML();

The important thing here is that you have to supply omNode::cloneNode() with a true parameter (default is false), so that it copies all descendant nodes as well. If we had left that to false, we would have gotten an empty <url></url> node, which is not desirable. ;)

Modifying Node Data

When modifying node data, you want to modify the CDATA within a node. You can use xpath again to find the node you want to edit and then simply supply a new value to its data property:

php

1. $source = 'sitemap.xml';
2. $dom = new DomDocument();
3. $dom->load($source);
4. $xpath = new DomXPath($dom);
5. $xpath->registerNamespace("c","http://www.google.com/schemas/sitemap/0.84");
6. $result = $xpath->query("//c:loc/text()"),
7. $node = $result->item(1);
8. $node->data = strtoupper($node->data);
9. echo $dom->saveXML();

This code transforms the location data of the second url to uppercase letters.

Removing Data From XML Documents

There are three types of data that you would possibly want to remove from xml documents: elements, attributes and CDATA. The DOM extension provides a method for each of them:
DomElement::removeAttribute(), DomNode::removeChild() and DomCharacterData::deleteData(). We will use a custom xml document and not our
sitemap to demonstrate their behavior. This makes it easier for you to come back to this article and see at first glance how these methods work. Thank Nikos if you want to. ;)

php

```php
1. $xml = <<<XML
2. <xml>
3. <text type="input">This is some really cool text!</text>
4. <text type="input">This is some other really cool text!</text>
5. <text type="misc">This is some cool text!</text>
6. <text type="output">This is text!</text>

7. XML;
8. $dom = new DomDocument();
9. $dom->loadXML($xml);
10. $xpath = new DomXPath($dom);
11. $result = $xpath->query("//text");
12. // remove first node
13. $result->item(0)->parentNode->removeChild($result->item(0));
14. // remove attribute from second node
15. $result->item(1)->removeAttribute('type');
16. //delete data from third element
17. $result = $xpath->query("text()",$result->item(2));
18. $result->item(0)->deleteData(0, $result->item(0)->length);
19. echo $dom->saveXML();
```

The output of this is:

```xml
< ?xml version="1.0"?>
<xml>

<text>This is some other really cool text!</text>
<text type="misc"></text>
<text type="output">This is text!</text>
```

In this example we start by retrieving all text nodes from a document. Then we remove some data from that document. Simple. In fact we remove the first node all together as well as the attribute of the second node.
Finally we truncate the character data of the third node, using xpath to query the corresponding text() node. 
Note that DomCharacterData::deleteData() requires a starting offset and a length parameter. Since we want to truncate the data in our example we supply 0 and the length of the CDATA node.

**DOM And Working With Namespaces**

DOM is very capable of handling namespaces on its own. Most of the time you can ignore them and pass attribute and element names with the appropriate prefix directly to most DOM functions.

```php
$dom = new DomDocument();

$node = $dom->createElement('ns1:somenode');
$node->setAttribute('ns2:someattribute','somevalue');

$node2 = $dom->createElement('ns3:anothernode');
$node->appendChild($node2);

// Set xmlns attributes
$node->setAttribute('xmlns:ns1', 'http://php-coding-practices.com/');
$node->setAttribute('xmlns:ns3', 'http://php-coding-practices.com/sitemap/');
$dom->appendChild($node);

echo $dom->saveXML();
```

The output of this script is:

```xml
<?xml version="1.0"?><ns1:somenode
   ns2:someattribute="somevalue"
xmlns:ns1="http://php-coding-practices.com/"
xmlns:ns2="http://php-coding-practices.com/articles/"
xmlns:ns3="http://php-coding-practices.com/sitemap/"
```
We can simplify the use of namespaces somewhat by using
\texttt{DomDocument::createElementNS()} and \texttt{DomDocument::setAttributeNS()}, which were specifically designed for this purpose:

\begin{verbatim}
php
1. $dom = new DomDocument();
2. $node = $dom->createElementNS('http://php-coding-practices.com/', 'ns1:somenode');
3. $node->setAttributeNS('http://somewebsite.com/ns2', 'ns2:someattribute', 'somevalue');
5. $node3 = $dom->createElementNS('http://php-coding-practices.com/sitemap/', 'ns1:someothernode');
6. $node->appendChild($node2);
7. $node->appendChild($node3);
8. $dom->appendChild($node);
9. echo $dom->saveXML();
\end{verbatim}

This results in the following output:

\begin{verbatim}
xml
<?xml version="1.0"?>
<ns1 :somenode
 xmlns:ns1="http://php-coding-practices.com/"
 xmlns:ns2="http://somewebsite.com/ns2"
 xmlns:ns3="http://php-coding-practices.com/articles/"
    ns2:someattribute="somevalue">
  </ns3 :anothernode>
</ns1 :somenode>
\end{verbatim}
Interfacing With SimpleXML

As I have mentioned at the start of our little DOM journey it is very easy to exchange loaded documents between SimpleXML and DOM. Therefore, you can take advantage of both systems' strengths - SimpleXML's simplicity and DOM's power.

You can import SimpleXML object into DOM by using PHP's dom_import_simplexml() function:

```php
1. $sxml = simplexml_load_file('sitemap.xml');
2. $node = dom_import_simplexml($sxml);
3. $dom = new DomDocument();
4. $dom->importNode($node, true);
5. $dom->appendChild($node);
```

`DomDocument::importNode()` creates a copy of the node and associates it with the current document. Its second parameter - a boolean value - determines if the method will recursively import the subtree or not.

You can also import a dom object into SimpleXML using simple_xml_import_dom():

```php
$dom = new DomDocument();
$dom->load('sitemap.xml');
$sxe = simplexml_import_dom($dom);
echo $sxe->url[0]->loc;
```