DOM XML

Introduction

The DOM XML extension has been overhauled in PHP 4.3.0 to better comply with the DOM standard. The extension still contains many old functions, but they should no longer be used. In particular, functions that are not object-oriented should be avoided.

The extension allows you to operate on an XML document with the DOM API. It also provides a function domxml_xmltree() to turn the complete XML document into a tree of PHP objects. Currently, this tree should be considered read-only - you can modify it, but this would not make any sense since DomDocument_dump_mem() cannot be applied to it. Therefore, if you want to read an XML file and write a modified version, use DomDocument_create_element(), DomDocument_create_text_node(), **set_attribute()**, etc. and finally the DomDocument_dump_mem() function.

Note
This extension has been moved to the » PECL repository and is no longer bundled with PHP as of PHP 5.0.0.

Note
This extension is no longer marked experimental. It will, however, never be released with PHP 5, and will only be distributed with PHP 4. If you need DOM XML support with PHP 5 you can use the DOM extension. This *domxml* extension is not compatible with the DOM extension.

Installing/Configuring

Requirements

This extension makes use of the » GNOME XML library. Download and install this library. You will need at least libxml-2.4.14. To use DOM XSLT features you can use the » libxslt library and EXSLT enhancements from » http://www.exslt.org/. Download and install these libraries if you plan to use (enhanced) XSLT features. You will need at least libxslt-1.0.18.

Installation

This » PECL extension is not bundled with PHP. Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here: » http://pecl.php.net/package/domxml.

In PHP 4 this PECL extensions source can be found in the *ext/* directory within the PHP source or at the PECL link above. This extension is only available if PHP was configured with *--with-dom[=DIR]*. Add *--with-dom-xslt[=DIR]* to include DOM XSLT support. DIR is the libxslt install directory. Add *--with-dom-exslt[=DIR]* to include DOM EXSLT support, where DIR is the libexslt install directory.

Windows users will enable *php_domxml.dll* inside of *php.ini* in order to use these functions. In PHP 4 this DLL resides in the *extensions/* directory within the PHP Windows binaries download. The DLL for this PECL extension may be downloaded from either the » PHP Downloads page or from » http://pecl4win.php.net/ Also, there is one additional DLL that must be made available to your system's PATH in order for this extension to work. In PHP 4 this is in the *dlls/* directory. It's name: For PHP <= 4.2.0, it's *libxml2.dll*. For PHP >= 4.3.0, it's *iconv.dll*. And as of PHP 5.0.0, iconv is compiled into your Windows PHP binaries by default so no extra DLL is needed.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

XML constants

Constant	Value	Description
XML_ELEMENT_NODE (integer)	1	Node is an element
XML_ATTRIBUTE_NODE (integer)	2	Node is an attribute
XML_TEXT_NODE (integer)	3	Node is a piece of text
XML_CDATA_SECTION_NODE (integer)	4	
XML_ENTITY_REF_NODE (integer)	5	
XML_ENTITY_NODE (integer)	6	Node is an entity like
XML_PI_NODE (integer)	7	Node is a processing instruction
XML_COMMENT_NODE (integer)	8	Node is a comment
XML_DOCUMENT_NODE (integer)	9	Node is a document
XML_DOCUMENT_TYPE_NODE (integer)	10	
XML_DOCUMENT_FRAG_NODE (integer)	11	
XML_NOTATION_NODE (integer)	12	
XML_GLOBAL_NAMESPACE (integer)	1	
XML_LOCAL_NAMESPAC	**2**	

E (integer)		
XML_HTML_DOCUMENT_NODE (integer)		
XML_DTD_NODE (integer)		
XML_ELEMENT_DECL_NODE (integer)		
XML_ATTRIBUTE_DECL_NODE (integer)		
XML_ENTITY_DECL_NODE (integer)		
XML_NAMESPACE_DECL_NODE (integer)		
XML_ATTRIBUTE_CDATA (integer)		
XML_ATTRIBUTE_ID (integer)		
XML_ATTRIBUTE_IDREF (integer)		
XML_ATTRIBUTE_IDREFS (integer)		
XML_ATTRIBUTE_ENTITY (integer)		
XML_ATTRIBUTE_NMTOKEN (integer)		
XML_ATTRIBUTE_NMTOKENS (integer)		
XML_ATTRIBUTE_ENUMERATION (integer)		
XML_ATTRIBUTE_NOTATION (integer)		
XPATH_UNDEFINED (integer)		
XPATH_NODESET (integer)		

XPATH_BOOLEAN (integer)		
XPATH_NUMBER (integer)		
XPATH_STRING (integer)		
XPATH_POINT (integer)		
XPATH_RANGE (integer)		
XPATH_LOCATIONSET (integer)		
XPATH_USERS (integer)		
XPATH_NUMBER (integer)		

DOM XML Functions

Deprecated functions

There are quite a few functions that do not fit into the DOM standard and should no longer be used. These functions are listed in the following table. The function DomNode_append_child() has changed its behaviour. It now adds a child and not a sibling. If this breaks your application, use the non-DOM function DomNode_append_sibling().

Deprecated functions and their replacements

Old function	New function
xmldoc	domxml_open_mem()
xmldocfile	domxml_open_file()
domxml_new_xmldoc	domxml_new_doc()
domxml_dump_mem	DomDocument_dump_mem()
domxml_dump_mem_file	DomDocument_dump_file()
DomDocument_dump_mem_file	DomDocument_dump_file()
DomDocument_add_root	DomDocument_create_element() followed by DomNode_append_child()
DomDocument_dtd	DomDocument_doctype()
DomDocument_root	DomDocument_document_element()
DomDocument_children	DomNode_child_nodes()
DomDocument_imported_node	No replacement.
DomNode_add_child	Create a new node with e.g. DomDocument_create_element() and add it with DomNode_append_child().
DomNode_children	DomNode_child_nodes()
DomNode_parent	DomNode_parent_node()

DomNode_new_child	Create a new node with e.g. DomDocument_create_element() and add it with DomNode_append_child().
DomNode_set_content	Create a new node with e.g. DomDocument_create_text_node() and add it with DomNode_append_child().
DomNode_get_content	Content is just a text node and can be accessed with DomNode_child_nodes().
DomNode_set_content	Content is just a text node and can be added with DomNode_append_child().

Classes

The API of the module follows the DOM Level 2 standard as closely as possible. Consequently, the API is fully object-oriented. It is a good idea to have the DOM standard available when using this module. Though the API is object-oriented, there are many functions which can be called in a non-object-oriented way by passing the object to operate on as the first argument. These functions are mainly to retain compatibility to older versions of the extension, and should not be used when creating new scripts.

This API differs from the official DOM API in two ways. First, all class attributes are implemented as functions with the same name. Secondly, the function names follow the PHP naming convention. This means that a DOM function lastChild() will be written as last_child().

This module defines a number of classes, which are listed - including their method - in the following tables. Classes with an equivalent in the DOM standard are named DOMxxx.

List of classes

Class name	**Parent classes**
DomAttribute	DomNode
DomCData	DomNode
DomComment	DomCData : DomNode
DomDocument	DomNode
DomDocumentType	DomNode
DomElement	DomNode

DomEntity	DomNode
DomEntityReference	DomNode
DomProcessingInstruction	DomNode
DomText	DomCData : DomNode
Parser	Currently still called DomParser
XPathContext	

DomDocument class (DomDocument : DomNode)

Method name	Function name	Remark
doctype	DomDocument_doctype()	
document_element	DomDocument_document_element()	
create_element	DomDocument_create_element()	
create_text_node	DomDocument_create_text_node()	
create_comment	DomDocument_create_comment()	
create_cdata_section	DomDocument_create_cdata_section()	
create_processing_instruction	DomDocument_create_processing_instruction()	
create_attribute	DomDocument_create_attribute()	
create_entity_reference	DomDocument_create_entity_reference()	
get_elements_by_tagname	DomDocument_get_elements_by_tagname()	
get_element_by_id	DomDocument_get_element_by_id()	

dump_mem	DomDocument_dump_mem()	not DOM standard
dump_file	DomDocument_dump_file()	not DOM standard
html_dump_mem	DomDocument_html_dump_mem()	not DOM standard
xpath_init	xpath_init	not DOM standard
xpath_new_context	xpath_new_context	not DOM standard
xptr_new_context	xptr_new_context	not DOM standard

DomElement class (DomElement : DomNode)

Method name	**Function name**	**Remark**
tagname	**DomElement_tagname()**	
get_attribute	**DomElement_get_attribute()**	
set_attribute	**DomElement_set_attribute()**	
remove_attribute	**DomElement_remove_attribute()**	
get_attribute_node	**DomElement_get_attribute_node()**	
set_attribute_node	**DomElement_set_attribute_node()**	
get_elements_by_tagname	**DomElement_get_elements_by_tagname()**	
has_attribute	**DomElement_has_attribute()**	

DomNode class

Method name	**Remark**

DomNode_node_name()	
DomNode_node_value()	
DomNode_node_type()	
DomNode_last_child()	
DomNode_first_child()	
DomNode_child_nodes()	
DomNode_previous_sibling()	
DomNode_next_sibling()	
DomNode_parent_node()	
DomNode_owner_document()	
DomNode_insert_before()	
DomNode_append_child()	
DomNode_append_sibling()	Not in DOM standard. This function emulates the former behaviour of DomNode_append_child().
DomNode_remove_child()	
DomNode_has_child_nodes()	
DomNode_has_attributes()	
DomNode_clone_node()	
DomNode_attributes()	
DomNode_unlink_node()	Not in DOM standard
DomNode_replace_node()	Not in DOM standard
DomNode_set_content()	Not in DOM standard, deprecated
DomNode_get_content()	Not in DOM standard, deprecated
DomNode_dump_node()	Not in DOM standard
DomNode_is_blank_node()	Not in DOM standard

DomAttribute class (DomAttribute : DomNode)

Method name			**Remark**
name	DomAttribute_name()		
value	DomAttribute_value()		
specified	DomAttribute_specified()		

DomProcessingInstruction class (DomProcessingInstruction : DomNode)

Method name	**Function name**	**Remark**
target	DomProcessingInstruction_target()	
data	DomProcessingInstruction_data()	

Parser class

Method name	**Function name**	**Remark**
add_chunk	**Parser_add_chunk()**	
end	**Parser_end()**	

XPathContext class

Method name	**Function name**	**Remark**
eval	**XPathContext_eval()**	
eval_expression	**XPathContext_eval_expression()**	
register_ns	**XPathContext_register_ns()**	

DomDocumentType class (DomDocumentType : DomNode)

Method name	Function name	Remark
name	**DomDocumentType_name()**	
entities	**DomDocumentType_entities()**	
notations	**DomDocumentType_notations()**	
public_id	**DomDocumentType_public_id()**	
system_id	**DomDocumentType_system_id()**	
internal_subset	**DomDocumentType_internal_subset()**	

The classes DomDtd is derived from DomNode. DomComment is derived from DomCData.

Examples

Many examples in this reference require an XML string. Instead of repeating this string in every example, it will be put into a file which will be included by each example. This include file is shown in the following example section. Alternatively, you could create an XML document and read it with **DomDocument_open_file()**.

Example #1 - Include file example.inc with XML string

```
<?php
$xmlstr = "<?xml version='1.0' standalone='yes'?>
<!DOCTYPE chapter SYSTEM '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'
[ <!ENTITY sp \"spanish\">
]>
<!-- lsfj  -->
<chapter language='en'><title language='en'>Title</title>
<para language='ge'>
 &sp;
 <!-- comment -->
 <informaltable ID='findme' language='&sp;'>
  <tgroup cols='3'>
   <tbody>
    <row><entry>a1</entry><entry
morerows='1'>b1</entry><entry>c1</entry></row>
```

```
<row><entry>a2</entry><entry>c2</entry></row>
    <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
   </tbody>
  </tgroup>
 </informaltable>
</para>
</chapter>";
?>
```

DomAttribute->name

DomAttribute->name -- Returns the name of attribute

Description

DomAttribute

string **name** (void)

Gets the name of the attribute.

Return Values

Returns the name of the attribute.

Migrating to PHP 5

Use the name property of DOMAttr.

See Also

- DomAttribute->value for an example

DomAttribute->set_value

DomAttribute->set_value -- Sets the value of an attribute

Description

DomAttribute

bool **set_value** (string `$content`)

This function sets the value of an attribute.

Parameters

`content`
The new value.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Migrating to PHP 5

Set the value property of DOMAttr.

See Also

- DomAttribute->value

DomAttribute->specified

DomAttribute->specified -- Checks if attribute is specified

Description

DomAttribute

bool **specified** (void)

Checks if the attribute was explicitly given a value in the original document.

Note
This method is not implemented yet.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- The definition of » specified in the DOM Recommendations

DomAttribute->value

DomAttribute->value -- Returns value of attribute

Description

DomAttribute

string **value** (void)

This function returns the value of the attribute.

Examples

Example #2 - Getting all the attributes of a node

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
    echo "Error while parsing the document\n";
    exit;
}

$root = $dom->document_element();
$attrs = $root->attributes();

echo 'Attributes of ' . $root->node_name() . "\n";
foreach ($attrs as $attribute) {
    echo ' - ' . $attribute->name . ' : ' . $attribute->value . "\n";
}

?>
```

The above example will output:

```
Attributes of chapter
- language : en
```

Return Values

Returns the value of the attribute.

Migrating to PHP 5

Use the value property of DOMAttr.

See Also

- DomAttribute->set_value
- DomAttribute->name

DomDocument->add_root

DomDocument->add_root -- Adds a root node [deprecated]

Description

domelement **DomDocument->add_root** (string `$name`)

Adds a root element node to a dom document and returns the new node. The element name is given in the passed parameter.

Example #3 - Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->add_root("html");
$head = $root->new_child("head", "");
$head->new_child("title", "Hier der Titel");
echo htmlentities($doc->dump_mem());
?>
```

DomDocument->create_attribute

DomDocument->create_attribute -- Create new attribute

Description

domattribute **DomDocument->create_attribute** (string `$name`, string `$value`)

This function returns a new instance of class DomAttribute. The name of the attribute is the value of the first parameter. The value of the attribute is the value of the second parameter. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_cdata_section(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), and domnode_insert_before().

DomDocument->create_cdata_section

DomDocument->create_cdata_section -- Create new cdata node

Description

domcdata **DomDocument->create_cdata_section** (string `$content`)

This function returns a new instance of class DomCData. The content of the cdata is the value of the passed parameter. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), and domnode_insert_before().

DomDocument->create_comment

DomDocument->create_comment -- Create new comment node

Description

domcomment **DomDocument->create_comment** (string `$content`)

This function returns a new instance of class DomComment. The content of the comment is the value of the passed parameter. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), and domnode_insert_before().

DomDocument->create_element_ns

DomDocument->create_element_ns -- Create new element node with an associated namespace

Description

domelement **DomDocument->create_element_ns** (string `$uri`, string `$name` [, string `$prefix`])

This function returns a new instance of class DomElement. The tag name of the element is the value of the passed parameter *`name`*. The URI of the namespace is the value of the passed parameter *`uri`*. If there is already a namespace declaration with the same uri in the root-node of the document, the prefix of this is taken, otherwise it will take the one provided in the optional parameter *`prefix`* or generate a random one. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domdocument_create_element_ns(), domnode_add_namespace(), domnode_set_namespace(), domnode_append_child(), **domdocument_create_text()**, domdocument_create_comment(), domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), and domnode_insert_before().

DomDocument->create_element

DomDocument->create_element -- Create new element node

Description

domelement **DomDocument->create_element** (string `$name`)

This function returns a new instance of class DomElement. The tag name of the element is the value of the passed parameter. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domdocument_create_element_ns(), domnode_append_child(), **domdocument_create_text()**, domdocument_create_comment(), domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), and domnode_insert_before().

DomDocument->create_entity_reference

DomDocument->create_entity_reference -- Create an entity reference

Description

domentityreference **DomDocument->create_entity_reference** (string `$content`)

This function returns a new instance of class DomEntityReference. The content of the entity reference is the value of the passed parameter. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_cdata_section(), domdocument_create_processing_instruction(), domdocument_create_attribute(), and domnode_insert_before().

DomDocument->create_processing_instruction

DomDocument->create_processing_instruction -- Creates new PI node

Description

domprocessinginstruction **DomDocument->create_processing_instruction** (string `$content`)

This function returns a new instance of class DomCData. The content of the pi is the value of the passed parameter. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_cdata_section(), domdocument_create_attribute(), domdocument_create_entity_reference(), and domnode_insert_before().

DomDocument->create_text_node

DomDocument->create_text_node -- Create new text node

Description

domtext **DomDocument->create_text_node** (string `$content`)

This function returns a new instance of class DomText. The content of the text is the value of the passed parameter. This node will not show up in the document unless it is inserted with (e.g.) domnode_append_child().

The return value is **FALSE** if an error occurred.

See also domnode_append_child(), domdocument_create_element(), domdocument_create_comment(), **domdocument_create_text()**, domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), and domnode_insert_before().

DomDocument->doctype

DomDocument->doctype -- Returns the document type

Description

domdocumenttype **DomDocument->doctype** (void)

This function returns an object of class DomDocumentType. In versions of PHP before 4.3 this has been the class Dtd, but the DOM Standard does not know such a class.

See also the methods of class DomDocumentType.

DomDocument->document_element

DomDocument->document_element -- Returns root element node

Description

domelement **DomDocument->document_element** (void)

This function returns the root element node of a document.

The following example returns just the element with name CHAPTER and prints it. The other node -- the comment -- is not returned.

Example #4 - Retrieving root element

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$root = $dom->document_element();
print_r($root);
?>
```

The above example will output:

```
domelement Object
(
   [type] => 1
   [tagname] => chapter
   [0] => 6
   [1] => 137960648
)
```

DomDocument->dump_file

DomDocument->dump_file -- Dumps the internal XML tree back into a file

Description

string **DomDocument->dump_file** (string `$filename` [, bool `$compressionmode` [, bool `$format`]])

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *`format`* specifies whether the output should be neatly formatted, or not. The first parameter specifies the name of the filename and the second parameter, whether it should be compressed or not.

Example #5 - Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
$doc->dump_file("/tmp/test.xml", false, true);
?>
```

See also domdocument_dump_mem(), and domdocument_html_dump_mem().

DomDocument->dump_mem

DomDocument->dump_mem -- Dumps the internal XML tree back into a string

Description

string **DomDocument->dump_mem** ([bool `$format` [, string `$encoding`]])

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *`format`* specifies whether the output should be neatly formatted, or not.

Example #6 - Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->dump_mem(true));
echo "</PRE>";
?>
```

Note

The first parameter was added in PHP 4.3.0.

See also domdocument_dump_file(), and domdocument_html_dump_mem().

DomDocument->get_element_by_id

DomDocument->get_element_by_id -- Searches for an element with a certain id

Description

domelement **DomDocument->get_element_by_id** (string `$id`)

This function is similar to domdocument_get_elements_by_tagname() but searches for an element with a given id. According to the DOM standard this requires a DTD which defines the attribute ID to be of type ID, though the current implementation simply does an xpath search for "//*[@ID = '%s']". This does not comply to the DOM standard which requires to return null if it is not known which attribute is of type id. This behaviour is likely to be fixed, so do not rely on the current behaviour.

See also domdocument_get_elements_by_tagname()

DomDocument->get_elements_by_tagname

DomDocument->get_elements_by_tagname -- Returns array with nodes with given tagname in document or empty array, if not found

Description

array **DomDocument->get_elements_by_tagname** (string `$name`)

See also domdocument_add_root()

DomDocument->html_dump_mem

DomDocument->html_dump_mem -- Dumps the internal XML tree back into a string as HTML

Description

string **DomDocument->html_dump_mem** (void)

Creates an HTML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below.

Example #7 - Creating a simple HTML document header

```
<?php

// Creates the document
$doc = domxml_new_doc("1.0");

$root = $doc->create_element("html");
$root = $doc->append_child($root);

$head = $doc->create_element("head");
$head = $root->append_child($head);

$title = $doc->create_element("title");
$title = $head->append_child($title);

$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);

echo $doc->html_dump_mem();
?>
```

The above example will output:

```
<html><head><title>This is the title</title></head></html>
```

See also domdocument_dump_file(), and domdocument_html_dump_mem().

DomDocument->xinclude

DomDocument->xinclude -- Substitutes XIncludes in a DomDocument Object

Description

int **DomDocument->xinclude** (void)

This function substitutes » XIncludes in a DomDocument object.

Example #8 - Substituting Xincludes

```
<?php

// include.xml contains :
// <child>test</child>

$xml = '<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
 <xi:include href="include.xml">
   <xi:fallback>
     <error>xinclude: include.xml not found</error>
   </xi:fallback>
 </xi:include>
</root>';

$domxml = domxml_open_mem($xml);
$domxml->xinclude();

echo $domxml->dump_mem();

?>
```

The above example will output:

```
<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
 <child>test</child>
</root>
```

If *include.xml* doesn't exist, you'll see:

```
<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
 <error>xinclude:dom.xml not found</error>
</root>
```

DomDocumentType->entities()

DomDocumentType->entities() -- Returns list of entities

Description

DomDocumentType

array **entities** (void)

Warning
This function is currently not documented; only its argument list is available.

Migrating to PHP 5

Use the entities property of the DOMDocumentType object.

DomDocumentType->internal_subset()

DomDocumentType->internal_subset() -- Returns internal subset

Description

DomDocumentType

bool **internal_subset** (void)

Warning
This function is currently not documented; only its argument list is available.

Migrating to PHP 5

Use the internalSubset property of the DOMDocumentType object.

DomDocumentType->name()

DomDocumentType->name() -- Returns name of document type

Description

DomDocumentType

string **name** (void)

This function returns the name of the document type.

Return Values

Returns the name of the DomDocumentType, as a string.

Examples

Example #9 - Getting the document type's name

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document\n";
   exit;
}

$doctype = $dom->doctype();
echo $doctype->name(); // chapter

?>
```

Migrating to PHP 5

Use the name property of the DOMDocumentType object.

DomDocumentType->notations()

DomDocumentType->notations() -- Returns list of notations

Description

DomDocumentType

array **notations** (void)

Warning

This function is currently not documented; only its argument list is available.

Migrating to PHP 5

Use the notations property of the DOMDocumentType object.

DomDocumentType->public_id()

DomDocumentType->public_id() -- Returns public id of document type

Description

DomDocumentType

string **public_id** (void)

This function returns the public id of the document type.

Return Values

Returns the public id of the DomDocumentType, as a string.

Examples

The following example echos nothing.

Example #10 - Retrieving the public id

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$doctype = $dom->doctype();
echo $doctype->public_id();
?>
```

Migrating to PHP 5

Use the publicId property of the DOMDocumentType object.

DomDocumentType->system_id()

DomDocumentType->system_id() -- Returns the system id of document type

Description

DomDocumentType

string **system_id** (void)

Returns the system id of the document type.

Return Values

Returns the system id of the DomDocumentType, as a string.

Examples

Example #11 - Retrieving the system id

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$doctype = $dom->doctype();
echo $doctype->system_id();
?>
```

The above example will output:

```
/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd
```

Migrating to PHP 5

Use the systemId property of the DOMDocumentType object.

DomElement->get_attribute_node()

DomElement->get_attribute_node() -- Returns the node of the given attribute

Description

DomElement

DomAttribute **get_attribute_node** (string `$name`)

Returns the node of the given attribute in the current element.

Parameters

name
The name of the seeked attribute. This parameter is case sensitive.

Return Values

Returns the node of the attribute as a DomAttribute or **FALSE** if no attribute with the given *name* is found.

Examples

Example #12 - Getting an attribute node

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document\n";
   exit;
}

$root = $dom->document_element();
if ($attribute = $root->get_attribute_node('language')) {
   echo 'Language is: ' . $attribute->value() . "\n";
}

?>
```

Migrating to PHP 5

Use DOMElement::getAttributeNode.

See Also

- DomElement->get_attribute()
- DomElement->set_attribute()

DomElement->get_attribute()

DomElement->get_attribute() -- Returns the value of the given attribute

Description

DomElement

string **get_attribute** (string `$name`)

Returns the value of the given attribute in the current element.

Since PHP 4.3, if no attribute with given *`name`* is found, an empty string is returned.

Parameters

`name`
The name of the seeked attribute. This parameter is case sensitive.

Return Values

Returns the name of the attribute as a string or an empty string if no attribute with the given *`name`* is found.

Examples

Example #13 - Getting the value of an attribute

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document\n";
   exit;
}

// get chapter
$root = $dom->document_element();
echo $root->get_attribute('language'); // en

?>
```

Migrating to PHP 5

Use DOMElement::getAttribute.

See Also

- DomElement->get_attribute_node()
- DomElement->set_attribute()

DomElement->get_elements_by_tagname()

DomElement->get_elements_by_tagname() -- Gets elements by tagname

Description

DomElement

array **get_elements_by_tagname** (string `$name`)

Gets all the sub elements with the specific *`name`* within the current element.

Parameters

`name`
The name of the seeked element.

Return Values

Returns an array of DomElement objects.

Examples

Example #14 - Getting a content

```
<?php
if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$root = $dom->document_element();

$node_array = $root->get_elements_by_tagname('element');

foreach ($node_array as $node) {
   echo ' - ' . $node->get_content() . "\n";
}

?>
```

Migrating to PHP 5

Use DOMElement::getElementsByTagName.

DomElement->has_attribute()

DomElement->has_attribute() -- Checks to see if an attribute exists in the current node

Description

DomElement

bool **has_attribute** (string `$name`)

This functions checks to see if an attribute named *`name`* exists in the current node.

Parameters

`name`
The name of the tested attribute.

Return Values

Returns **TRUE** if the asked attribute exists, **FALSE** otherwise.

Examples

Example #15 - Testing the existence of an attribute

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document\n";
   exit;
}

$root = $dom->document_element();

$buffer = '<html';
if ($root->has_attribute('language')) {
   $buffer .= 'lang="' . $root->get_attribute('language') . '"';
}
$buffer .= '>';

?>
```

Migrating to PHP 5

Use DOMElement::hasAttribute.

DomElement->remove_attribute()

DomElement->remove_attribute() -- Removes attribute

Description

DomElement

bool **remove_attribute** (string `$name`)

Removes an attribute from the current DomElement node.

Parameters

`name`
The name of the attribute to remove.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Migrating to PHP 5

Use DOMElement::removeAttribute.

DomElement->set_attribute_node()

DomElement->set_attribute_node() -- Adds new attribute

Description

DomElement

DomNode **set_attribute_node** (DomNode `$attr`)

Warning

This function is currently not documented; only its argument list is available.

DomElement->set_attribute()

DomElement->set_attribute() -- Sets the value of an attribute

Description

DomElement

DomAttribute **set_attribute** (string `$name`, string `$value`)

Sets an attribute with name *`name`* to the given *`value`*.

Parameters

`name`
The name of the attribute. If this attribute doesn't exist, it will be created.

`value`
The value of the attribute.

Return Values

Returns the old DomAttribute node, or the new one if you are creating the attribute for the first time.

Examples

Example #16 - Setting an attribute

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

Migrating to PHP 5

Use DOMElement::setAttribute.

See Also

- DomElement->get_attribute_node()
- DomElement->get_attribute()

DomElement->tagname()

DomElement->tagname() -- Returns the name of the current element

Description

DomElement

string **tagname** (void)

Returns the name of the current node. Calling this function is the same as accessing the tagname property, or calling DomNode->node_name on the current node.

Return Values

Returns the name of the current DomElement node.

Examples

Example #17 - Getting the node name

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document\n";
   exit;
}

$root = $dom->document_element();
echo $root->tagname();   // chapter
echo $root->tagname;     // chapter
echo $root->node_name(); // chapter


?>
```

Migrating to PHP 5

Use the tagName property of the DOMElement object.

DomNode->add_namespace

DomNode->add_namespace -- Adds a namespace declaration to a node

Description

DOMNode

bool **add_namespace** (string `$uri`, string `$prefix`)

This method adds a namespace declaration to a node.

Note
This method is not part of the DOM specification.

Parameters

`uri`
The namespace URI of the node.

`prefix`
The namespace prefix of the node.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Migrating to PHP 5

You can set the namespace URI and prefix of a DOMElement or a DOMAttr at creation time by using DOMDocument::createElementNS or DOMDocument::createAttributeNS.

Note
Remember the an attribute does not inherit its namespace from the element it is attached to.

See Also

- DomDocument->create_element_ns
- DomNode->set_namespace

DomNode->append_child

DomNode->append_child -- Adds a new child at the end of the children

Description

DOMNode

DOMNode **append_child** (DOMNode `$newnode`)

This functions appends a child to an existing list of children or creates a new list of children.

Parameters

newnode

The node being appended. It can be created with e.g. DomDocument->create_element , DomDocument->create_text_node etc. or simply by using any other node.

Note
You can not append a DOMAttribute using this method. Use DomElement->set_attribute() instead.

Return Values

Returns the appended node on success or **FALSE** on failure.

ChangeLog

Version	Description
4.3.0	You are not allowed anymore to insert a node from another document.
4.3.0	Prior to PHP 4.3.0, the new child is duplicated before being appended. Therefore the new child is a completely new

	copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of an XML document. The return value is the appended child. If you plan to do further modifications on the appended child you must use the returned node.
4.3.0 and 4.3.1	The new child *newnode* is first unlinked from its existing context, if it's already a child of DomNode. Therefore the *newnode* is moved and not copies anymore. This is the behaviour according to the W3C specifications. If you need the old behaviour, use DomNode->clone_node before appending.
4.3.2	The new child *newnode* is first unlinked from its existing context, if it's already in the tree. Same rules apply.

Examples

The following example adds a new element node to a fresh document and sets the attribute *align* to *left*.

Example #18 - Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

The above example could also be written as the following:

Example #19 - Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node->set_attribute("align", "left");
```

```
$newnode = $doc->append_child($node);
?>
```

A more complex example is the one below. It first searches for a certain element, duplicates it including its children and adds it as a sibling. Finally a new attribute is added to one of the children of the new sibling and the whole document is dumped.

Example #20 - Adding a child

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$parent = $element->parent_node();
$newnode = $parent->append_child($element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
?>
```

The above example could also be done with DomNode->insert_before instead of DomNode->append_child.

Migrating to PHP 5

You should use DOMNode::appendChild.

See Also

- DomNode->insert_before
- DomNode->clone_node

DomNode->append_sibling

DomNode->append_sibling -- Adds new sibling to a node

Description

domelement **DomNode->append_sibling** (domelement `$newnode`)

This functions appends a sibling to an existing node. The child can be created with e.g. domdocument_create_element(), **domdocument_create_text()** etc. or simply by using any other node.

Before a new sibling is added it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of an XML document. The return value is the added sibling. If you plan to do further modifications on the added sibling you must use the returned node.

This function has been added to provide the behaviour of domnode_append_child() as it works till PHP 4.2.

See also **domnode_append_before()**.

DomNode->attributes

DomNode->attributes -- Returns list of attributes

Description

array **DomNode->attributes** (void)

This function only returns an array of attributes if the node is of type XML_ELEMENT_NODE.

(PHP >= 4.3 only) If no attributes are found, NULL is returned.

DomNode->child_nodes

DomNode->child_nodes -- Returns children of node

Description

array **DomNode->child_nodes** (void)

Returns all children of the node.

See also domnode_next_sibling(), and domnode_previous_sibling().

DomNode->clone_node

DomNode->clone_node -- Clones a node

Description

domelement **DomNode->clone_node** (void)

Warning
This function is currently not documented; only its argument list is available.

DomNode->dump_node

DomNode->dump_node -- Dumps a single node

Description

string **DomNode->dump_node** (void)

Warning
This function is currently not documented; only its argument list is available.

See also domdocument_dump_mem().

DomNode->first_child

DomNode->first_child -- Returns first child of node

Description

domelement **DomNode->first_child** (void)

Returns the first child of the node.

(PHP >= 4.3 only) If no first child is found, NULL is returned.

See also domnode_last_child(), and domnode_next_sibling(), domnode_previous_sibling()
.

DomNode->get_content

DomNode->get_content -- Gets content of node

Description

string **DomNode->get_content** (void)

This function returns the content of the actual node.

Example #21 - Getting a content

```
<?php
if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$root = $dom->document_element();

$node_array = $root->get_elements_by_tagname("element");

for ($i = 0; $i<count($node_array); $i++) {
   $node = $node_array[$i];
   echo "The element[$i] is: " . $node->get_content();
}

?>
```

DomNode->has_attributes

DomNode->has_attributes -- Checks if node has attributes

Description

bool **DomNode->has_attributes** (void)

This function checks if the node has attributes.

See also domnode_has_child_nodes().

DomNode->has_child_nodes

DomNode->has_child_nodes -- Checks if node has children

Description

bool **DomNode->has_child_nodes** (void)

This function checks if the node has children.

See also domnode_child_nodes().

DomNode->insert_before

DomNode->insert_before -- Inserts new node as child

Description

domelement **DomNode->insert_before** (domelement `$newnode`, domelement `$refnode`)

This function inserts the new node *`newnode`* right before the node *`refnode`*. The return value is the inserted node. If you plan to do further modifications on the appended child you must use the returned node.

(PHP >= 4.3 only) If *`newnode`* already is part of a document, it will be first unlinked from its existing context. If *`refnode`* is NULL, then *`newnode`* will be inserted at the end of the list of children.

domnode_insert_before() is very similar to domnode_append_child() as the following example shows which does the same as the example at domnode_append_child().

Example #22 - Adding a child

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$newnode = $element->insert_before($element, $element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<pre>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</pre>";
?>
```

See also domnode_append_child().

DomNode->is_blank_node

DomNode->is_blank_node -- Checks if node is blank

Description

bool **DomNode->is_blank_node** (void)

Warning
This function is currently not documented; only its argument list is available.

DomNode->last_child

DomNode->last_child -- Returns last child of node

Description

domelement **DomNode->last_child** (void)

Returns the last child of the node.

(PHP >= 4.3 only) If no last child is found, NULL is returned.

See also domnode_first_child(), and domnode_next_sibling(), domnode_previous_sibling()
.

DomNode->next_sibling

DomNode->next_sibling -- Returns the next sibling of node

Description

domelement **DomNode->next_sibling** (void)

This function returns the next sibling of the current node. If there is no next sibling it returns **FALSE** (< 4.3) or null (>= 4.3). You can use this function to iterate over all children of a node as shown in the example.

Example #23 - Iterate over children

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$child = $element->first_child();

while ($child) {
  print_r($child);
  $child = $child->next_sibling();
}
?>
```

See also domnode_previous_sibling().

DomNode->node_name

DomNode->node_name -- Returns name of node

Description

string **DomNode->node_name** (void)

Returns name of the node. The name has different meanings for the different types of nodes as illustrated in the following table.

Meaning of value

Type	**Meaning**
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	#cdata-section
DomComment	#comment
DomDocument	#document
DomDocumentType	document type name
DomElement	tag name
DomEntity	name of entity
DomEntityReference	name of entity reference
DomNotation	notation name
DomProcessingInstruction	target
DomText	#text

DomNode->node_type

DomNode->node_type -- Returns type of node

Description

int **DomNode->node_type** (void)

Returns the type of the node. All possible types are listed in the table in the introduction.

Example #24

```
<?php

include 'example.inc';

$dom = domxml_open_mem($xmlstr);

$chapter = $dom->document_element();

// Let's see the elements contained in chapter
foreach($chapter->child_nodes() as $node) {
 if ($node->node_type() == XML_ELEMENT_NODE) {
   echo $node->node_name() . "\n";
 }
}

?>
```

The above example will output:

```
title
para
```

DomNode->node_value

DomNode->node_value -- Returns value of a node

Description

string **DomNode->node_value** (void)

Returns value of the node. The value has different meanings for the different types of nodes as illustrated in the following table.

Meaning of value

Type	**Meaning**
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	content
DomComment	content of comment
DomDocument	null
DomDocumentType	null
DomElement	null
DomEntity	null
DomEntityReference	null
DomNotation	null
DomProcessingInstruction	entire content without target
DomText	content of text

DomNode->owner_document

DomNode->owner_document -- Returns the document this node belongs to

Description

domdocument **DomNode->owner_document** (void)

This function returns the document the current node belongs to.

The following example will create two identical lists of children.

Example #25 - Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->owner_document();
$children = $doc2->children();
print_r($children);
?>
```

See also domnode_insert_before().

DomNode->parent_node

DomNode->parent_node -- Returns the parent of the node

Description

domnode **DomNode->parent_node** (void)

This function returns the parent node.

(PHP >= 4.3 only) If no parent is found, NULL is returned.

The following example will show two identical lists of children.

Example #26 - Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->parent_node();
$children = $doc2->children();
print_r($children);
?>
```

DomNode->prefix

DomNode->prefix -- Returns name space prefix of node

Description

string **DomNode->prefix** (void)

Returns the name space prefix of the node.

DomNode->previous_sibling

DomNode->previous_sibling -- Returns the previous sibling of node

Description

domelement **DomNode->previous_sibling** (void)

This function returns the previous sibling of the current node. If there is no previous sibling it returns **FALSE** (< 4.3) or **NULL** (>= 4.3). You can use this function to iterate over all children of a node as shown in the example.

See also domnode_next_sibling().

DomNode->remove_child

DomNode->remove_child -- Removes child from list of children

Description

domtext **DomNode->remove_child** (domtext `$oldchild`)

This functions removes a child from a list of children. If child cannot be removed or is not a child the function will return **FALSE**. If the child could be removed the functions returns the old child.

Example #27 - Removing a child

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$children = $element->child_nodes();
$child = $element->remove_child($children[0]);

echo "<PRE>";
$xmlfile = $dom->dump_mem(true);
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

See also domnode_append_child().

DomNode->replace_child

DomNode->replace_child -- Replaces a child

Description

domelement **DomNode->replace_child** (domelement `$newnode`, domelement `$oldnode`)

(PHP 4.2) This function replaces the child *`oldnode`* with the passed new node. If the new node is already a child it will not be added a second time. If the old node cannot be found the function returns **FALSE**. If the replacement succeeds the old node is returned.

(PHP 4.3) This function replaces the child *`oldnode`* with the passed *`newnode`*, even if the new node already is a child of the DomNode. If *`newnode`* was already inserted in the document it is first unlinked from its existing context. If the old node cannot be found the function returns **FALSE**. If the replacement succeeds the old node is returned. (This behaviour is according to the W3C specs).

See also domnode_append_child()

DomNode->replace_node

DomNode->replace_node -- Replaces node

Description

domelement **DomNode->replace_node** (domelement `$newnode`)

(PHP 4.2) This function replaces an existing node with the passed new node. Before the replacement *newnode* is copied if it has a parent to make sure a node which is already in the document will not be inserted a second time. This behaviour enforces doing all modifications on the node before the replacement or to refetch the inserted node afterwards with functions like domnode_first_child(), domnode_child_nodes() etc..

(PHP 4.3) This function replaces an existing node with the passed new node. It is not copied anymore. If *newnode* was already inserted in the document it is first unlinked from its existing context. If the replacement succeeds the old node is returned.

See also domnode_append_child()

DomNode->set_content

DomNode->set_content -- Sets content of node

Description

bool **DomNode->set_content** (string `$content`)

Warning
This function is currently not documented; only its argument list is available.

DomNode->set_name

DomNode->set_name -- Sets name of node

Description

bool **DomNode->set_name** (void)

Sets name of node.

See also domnode_node_name().

DomNode->set_namespace

DomNode->set_namespace -- Sets namespace of a node

Description

void **DomNode->set_namespace** (string `$uri` [, string `$prefix`])

Sets the namespace of a node to *`uri`*. If there is already a namespace declaration with the same uri in one of the parent nodes of the node, the prefix of this is taken, otherwise it will take the one provided in the optional parameter *`prefix`* or generate a random one.

See also domdocument_create_element_ns(), and domnode_add_namespace()

DomNode->unlink_node

DomNode->unlink_node -- Deletes node

Description

void **DomNode->unlink_node** (void)

Warning
This function is currently not documented; only its argument list is available.

DomProcessingInstruction->data

DomProcessingInstruction->data -- Returns the data of ProcessingInstruction node

Description

DomProcessingInstruction

string **data** (void)

This method gets the data of the ProcessingInstruction node.

Return Values

Returns the data of the Processing Instruction.

Migrating to PHP 5

Use the data property of DOMProcessingInstruction.

DomProcessingInstruction->target

DomProcessingInstruction->target -- Returns the target of a ProcessingInstruction node

Description

DomProcessingInstruction

string **target** (void)

This method gets the target of the ProcessingInstruction node.

Return Values

Returns the target of the Processing Instruction.

Migrating to PHP 5

Use the target property of DOMProcessingInstruction.

DomXsltStylesheet->process()

DomXsltStylesheet->process() -- Applies the XSLT-Transformation on a DomDocument Object

Description

DomXsltStylesheet

DomDocument **process** (DomDocument `$xml_doc` [, array `$xslt_params` [, bool `$is_xpath_param` [, string `$profile_filename`]]])

Applies an XSLT Transformation on the given DomDocument object.

Parameters

`xml_doc`
The XML document being transformed, as a DomDocument object.

`xslt_params`
An associative array that takes pairs of parameter names and values.

`is_xpath_param`
If set to **FALSE** the values of the *`xslt_params`* will be quoted. This is the default behavior. It allows you to pass the values as PHP strings.

Note

If your strings contains both single and double quotes, you must take care of quoting all the values by yourself and set this parameter to **TRUE**.

`profile_filename`
Set this to the path of a filename, if you want profiling information.

Return Values

Returns the result of the processing, as a DomDocument object.

Migrating to PHP 5

Use XSLTProcessor::setParameter() and **XSLTProcessor::transform-to-doc()**.

ChangeLog

Version	Description
4.3.0	The *profile_filename* parameter was added.

See Also

- domxml_xslt_stylesheet()
- domxml_xslt_stylesheet_file()
- domxml_xslt_stylesheet_doc()

DomXsltStylesheet->result_dump_file()

DomXsltStylesheet->result_dump_file() -- Dumps the result from a XSLT-Transformation into a file

Description

DomXsltStylesheet

string **result_dump_file** (DomDocument `$xmldoc`, string `$filename`)

Since DomXsltStylesheet->process() always returns a well-formed XML DomDocument, no matter what output method was declared in<xsl:output> and similar attributes/elements, it's of not much use, if you want to output HTML 4 or text data.

This function on the contrary honors<xsl:output method="html|text"> and other output control directives. See the example for instruction on how to use it.

Examples

Example #28 - Saving the result of a XSLT transformation in a file

```
<?php
$filename = "stylesheet.xsl";
$xmldoc = domxml_open_file("data.xml");
$xsldoc = domxml_xslt_stylesheet_file($filename);
$result =  $xsldoc->process($xmldoc);
echo $xsldoc->result_dump_file($result, "filename");
?>
```

See Also

- DomXsltStylesheet->result_dump_mem()
- DomXsltStylesheet->process()

DomXsltStylesheet->result_dump_mem()

DomXsltStylesheet->result_dump_mem() -- Dumps the result from a XSLT-Transformation back into a string

Description

DomXsltStylesheet

string **result_dump_mem** (DomDocument `$xmldoc`)

Since DomXsltStylesheet->process() always returns a well-formed XML DomDocument, no matter what output method was declared in<xsl:output> and similar attributes/elements, it's of not much use, if you want to output HTML 4 or text data.

This function on the contrary honors<xsl:output method="html|text"> and other output control directives. See the example for instruction on how to use it.

Examples

Example #29 - Outputting the result of a XSLT transformation

```
<?php
$filename = "stylesheet.xsl";
$xmldoc = domxml_open_file("data.xml");
$xsldoc = domxml_xslt_stylesheet_file($filename);
$result =  $xsldoc->process($xmldoc);
echo $xsldoc->result_dump_mem($result);
?>
```

See Also

- DomXsltStylesheet->result_dump_file()
- DomXsltStylesheet->process()

domxml_new_doc

domxml_new_doc -- Creates new empty XML document

Description

DomDocument **domxml_new_doc** (string `$version`)

Creates a new Dom document from scratch and returns it.

Parameters

`version`
The XML version number of the document.

Return Values

Returns a new DomDocument instance.

domxml_open_file

domxml_open_file -- Creates a DOM object from an XML file

Description

DomDocument **domxml_open_file** (string `$filename` [, int `$mode` [, array `&$error`]])

The function parses the XML document in the given file.

Parameters

`filename`
The path to the XML file. The file is accessed in read-only mode.

`mode`
This optional parameter can be used to change the behavior of this function. You can use one of the following constants for it: **DOMXML_LOAD_PARSING** (default), **DOMXML_LOAD_VALIDATING** or **DOMXML_LOAD_RECOVERING**. You can add to it also **DOMXML_LOAD_DONT_KEEP_BLANKS**, **DOMXML_LOAD_SUBSTITUTE_ENTITIES** and **DOMXML_LOAD_COMPLETE_ATTRS** by bitwise or.

`error`
If used, it will contain the error messages. *`error`* must be passed in by reference.

Return Values

Returns a DomDocument instance of the given file.

Examples

Example #30 - Opening an XML document from a file

```
<?php

if (!$dom = domxml_open_file("example.xml")) {
 echo "Error while parsing the document\n";
 exit;
}

$root = $dom->document_element();
?>
```

ChangeLog

Version	Description
4.3.0	The parameters *`mode`* and *`error`* were added.

See Also

- domxml_open_mem()
- domxml_new_doc()

domxml_open_mem

domxml_open_mem -- Creates a DOM object of an XML document

Description

DomDocument **domxml_open_mem** (string `$str` [, int `$mode` [, array `&$error`]])

The function parses the XML document in the given string.

Parameters

`str`
The contents of the XML file.

`mode`
This optional parameter can be used to change the behavior of this function. You can use one of the following constants for it: **DOMXML_LOAD_PARSING** (default), **DOMXML_LOAD_VALIDATING** or **DOMXML_LOAD_RECOVERING**. You can add to it also **DOMXML_LOAD_DONT_KEEP_BLANKS**, **DOMXML_LOAD_SUBSTITUTE_ENTITIES** and **DOMXML_LOAD_COMPLETE_ATTRS** by bitwise or.

`error`
If used, it will contain the error messages. *`error`* must be passed in by reference.

Return Values

Returns a DomDocument instance of the given XML contents.

ChangeLog

Version	Description
4.3.0	The *`mode`* and *`error`* parameters were added.

Examples

Example #31 - Opening an XML document in a string

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
 echo "Error while parsing the document\n";
 exit;
}

$root = $dom->document_element();
?>
```

See Also

- domxml_open_file()
- domxml_new_doc()

domxml_version

domxml_version -- Gets the XML library version

Description

string **domxml_version** (void)

Gets the version of the XML library currently used.

Return Values

The version of the XML library, as a string.

Examples

Example #32 - domxml_version() **Example**

```
<?php

echo domxml_version();

?>
```

The above example will output something similar to:

```
20607
```

domxml_xmltree

domxml_xmltree -- Creates a tree of PHP objects from an XML document

Description

DomDocument **domxml_xmltree** (string `$str`)

The function parses the XML document in *`str`* and returns a tree PHP objects as the parsed document.

This function is isolated from the other functions, which means you cannot access the tree with any of the other functions. Modifying it, for example by adding nodes, makes no sense since there is currently no way to dump it as an XML file.

However this function may be valuable if you want to read a file and investigate the content.

Parameters

`str`
The contents of the XML file.

Return Values

Returns a tree of Dom objects starting by a DomDocument.

domxml_xslt_stylesheet_doc

domxml_xslt_stylesheet_doc -- Creates a DomXsltStylesheet Object from a DomDocument Object

Description

DomXsltStylesheet **domxml_xslt_stylesheet_doc** (DomDocument `$xsl_doc`)

Creates a DomXsltStylesheet object from the given XSL document.

Parameters

`xsl_doc`
The XSL document, as a DomDocument object.

Return Values

Returns a new instance of DomXsltStylesheet.

Migrating to PHP 5

Call XSLTProcessor::importStylesheet() with the *`xsl_doc`* parameter.

See Also

- DomXsltStylesheet->process()
- domxml_xslt_stylesheet()
- domxml_xslt_stylesheet_file()

domxml_xslt_stylesheet_file

domxml_xslt_stylesheet_file -- Creates a DomXsltStylesheet Object from an XSL document in a file

Description

DomXsltStylesheet **domxml_xslt_stylesheet_file** (string `$xsl_file`)

Creates a DomXsltStylesheet object from the given XSL file.

Parameters

`xsl_file`
: The path to the XSL document, as a string.

Return Values

Returns a new instance of DomXsltStylesheet.

Migrating to PHP 5

Call XSLTProcessor::importStylesheet() with *DOMDocument::load($xsl_file)* as parameter.

See Also

- DomXsltStylesheet->process()
- domxml_xslt_stylesheet()
- domxml_xslt_stylesheet_doc()

domxml_xslt_stylesheet

domxml_xslt_stylesheet -- Creates a DomXsltStylesheet object from an XSL document in a string

Description

DomXsltStylesheet **domxml_xslt_stylesheet** (string `$xsl_buf`)

Creates a DomXsltStylesheet object from the given XSL buffer.

Parameters

`xsl_buf`
The XSL document, as a string.

Return Values

Returns a new instance of DomXsltStylesheet.

Migrating to PHP 5

Call XSLTProcessor::importStylesheet() with *DOMDocument::loadXML($xsl_buf)* as parameter.

See Also

- DomXsltStylesheet->process()
- domxml_xslt_stylesheet_file()
- domxml_xslt_stylesheet_doc()

domxml_xslt_version

domxml_xslt_version -- Gets the XSLT library version

Description

int **domxml_xslt_version** (void)

Gets the XSLT library version.

Return Values

Returns the version number of the XSLT library, as an integer.

Examples

Example #33 - domxml_xslt_version() **Example**

```
<?php

echo domxml_xslt_version();

?>
```

The above example will output something similar to:

```
10112
```

See Also

- domxml_version()

xpath_eval_expression

xpath_eval_expression -- Evaluates the XPath Location Path in the given string

Description

XPathContext

XPathObject **xpath_eval_expression** (string `$expression` [, domnode `$contextnode`])

XPathObject **xpath_eval_expression** (XPathContext `$xpath_context`, string `$expression` [, domnode `$contextnode`])

Example #34 - xpath_eval_expression() Example

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document\n";
   exit;
}

$xpath = xpath_new_context($dom);
var_dump(xpath_eval_expression($xpath, '/chapter/@language'));

?>
```

The above example will output:

```
object(XPathObject)(2) {
  ["type"]=>
  int(1)
  ["nodeset"]=>
  array(1) {
    [0]=>
    object(domattribute)(5) {
      ["type"]=>
      int(2)
      ["name"]=>
      string(8) "language"
      ["value"]=>
      string(2) "en"
      [0]=>
      int(7)
      [1]=>
      int(138004256)
```

```
    }
  }
}
```

See also xpath_eval().

xpath_eval

xpath_eval -- Evaluates the XPath Location Path in the given string

Description

XPathContext

XPathObject **xpath_eval** (string `$xpath_expression` [, domnode `$contextnode`])

XPathObject **xpath_eval** (XPathContext `$xpath_context`, string `$xpath_expression` [, domnode `$contextnode`])

The optional *`contextnode`* can be specified for doing relative XPath queries.

See also xpath_new_context().

xpath_new_context

xpath_new_context -- Creates new xpath context

Description

XPathContext **xpath_new_context** (domdocument `$dom_document`)

Creates a new xpath context.

See Also

- xpath_eval()

xpath_register_ns_auto

xpath_register_ns_auto -- Register the given namespace in the passed XPath context

Description

bool **xpath_register_ns_auto** (XPathContext `$xpath_context` [, object `$context_node`])

Warning
This function is currently not documented; only its argument list is available.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- xpath_register_ns()

xpath_register_ns

xpath_register_ns -- Register the given namespace in the passed XPath context

Description

bool **xpath_register_ns** (XPathContext `$xpath_context`, string `$prefix`, string `$uri`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- xpath_register_ns_auto()

xptr_eval

xptr_eval -- Evaluate the XPtr Location Path in the given string

Description

XPathContext

int **xptr_eval** (string `$eval_str` [, domnode `$contextnode`])

int **xptr_eval** (XPathContext `$xpath_context`, string `$eval_str` [, domnode `$contextnode`])

Warning

This function is currently not documented; only its argument list is available.

xptr_new_context

xptr_new_context -- Create new XPath Context

Description

XPathContext **xptr_new_context** (void)

Warning
This function is currently not documented; only its argument list is available.