XMLReader

Introduction

The XMLReader extension is an XML Pull parser. The reader acts as a cursor going forward on the document stream and stopping at each node on the way.

Encoding

It is important to note that internally, libxml uses the UTF-8 encoding and as such, the encoding of the retrieved contents will always be in UTF-8 encoding.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

The XMLReader extension is available in PECL as of PHP 5.0.0 and is included and enabled as of PHP 5.1.0 by default. It can be enabled by adding the argument *--enable-xmlreader* (or *--with-xmlreader* before 5.1.0) to your configure line. The libxml extension is required.

Runtime Configuration

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

Resource Types

This extension has no resource types defined.

The XMLReader class

Introduction

The XMLReader extension is an XML Pull parser. The reader acts as a cursor going forward on the document stream and stopping at each node on the way.

Class synopsis

XMLReader

```
XMLReader {

    /*  Constants */

    const int XMLReader::NONE = 0;

    const int XMLReader::ELEMENT = 1;

    const int XMLReader::ATTRIBUTE = 2;

    const int XMLReader::TEXT = 3;

    const int XMLReader::CDATA = 4;

    const int XMLReader::ENTITY_REF = 5;

    const int XMLReader::ENTITY = 6;

    const int XMLReader::PI = 7;

    const int XMLReader::COMMENT = 8;

    const int XMLReader::DOC = 9;

    const int XMLReader::DOC_TYPE = 10;

    const int XMLReader::DOC_FRAGMENT = 11;

    const int XMLReader::NOTATION = 12;

    const int XMLReader::WHITESPACE = 13;
```

const int *XMLReader::SIGNIFICANT_WHITESPACE* = 14;

const int *XMLReader::END_ELEMENT* = 15;

const int *XMLReader::END_ENTITY* = 16;

const int *XMLReader::XML_DECLARATION* = 17;

const int *XMLReader::LOADDTD* = 1;

const int *XMLReader::DEFAULTATTRS* = 2;

const int *XMLReader::VALIDATE* = 3;

const int *XMLReader::SUBST_ENTITIES* = 4;

/* Properties */

public readonly int *attributeCount*;

public readonly string *baseURI*;

public readonly int *depth*;

public readonly bool *hasAttributes*;

public readonly bool *hasValue*;

public readonly bool *isDefault*;

public readonly bool *isEmptyElement*;

public readonly string *localName*;

public readonly string *name*;

public readonly string *namespaceURI*;

public readonly int *nodeType*;

public readonly string *prefix*;

public readonly string *value*;

public readonly string *xmlLang*;

/* Methods */

bool **XMLReader::close** (void)

DOMNode **XMLReader::expand** (void)

string **XMLReader::getAttribute** (string `$name`)

string **XMLReader::getAttributeNo** (int `$index`)

string **XMLReader::getAttributeNs** (string `$localName`, string `$namespaceURI`)

bool **XMLReader::getParserProperty** (int `$property`)

bool **XMLReader::isValid** (void)

bool **XMLReader::lookupNamespace** (string `$prefix`)

bool **XMLReader::moveToAttribute** (string `$name`)

bool **XMLReader::moveToAttributeNo** (int `$index`)

bool **XMLReader::moveToAttributeNs** (string `$localName`, string `$namespaceURI`)

bool **XMLReader::moveToElement** (void)

bool **XMLReader::moveToFirstAttribute** (void)

bool **XMLReader::moveToNextAttribute** (void)

bool **XMLReader::next** ([string `$localname`])

bool **XMLReader::open** (string `$URI` [, string `$encoding` [, int `$options`]])

bool **XMLReader::read** (void)

bool **XMLReader::setParserProperty** (int `$property`, bool `$value`)

bool **XMLReader::setRelaxNGSchema** (string `$filename`)

bool **XMLReader::setRelaxNGSchemaSource** (string `$source`)

bool **XMLReader::xml** (string `$source` [, string `$encoding` [, int `$options`]])
}

Properties

attributeCount
: The number of attributes on the node

baseURI
: The base URI of the node

depth
: Depth of the node in the tree, starting at 0

hasAttributes
Indicates if node has attributes

hasValue
Indicates if node has a text value

isDefault
Indicates if attribute is defaulted from DTD

isEmptyElement
Indicates if node is an empty element tag

localName
The local name of the node

name
The qualified name of the node

namespaceURI
The URI of the namespace associated with the node

nodeType
The node type for the node

prefix
The prefix of the namespace associated with thenode

value
The text value of the node

xmlLang
The xml:lang scope which the node resides

Predefined Constants

XMLReader Node Types

XMLReader::NONE
No node type

XMLReader::ELEMENT
Start element

XMLReader::ATTRIBUTE
Attribute node

XMLReader::TEXT
Text node

XMLReader::CDATA
CDATA node

XMLReader::ENTITY_REF
Entity Reference node

XMLReader::ENTITY
Entity Declaration node

XMLReader::PI
Processing Instruction node

XMLReader::COMMENT
Comment node

XMLReader::DOC
Document node

XMLReader::DOC_TYPE
Document Type node

XMLReader::DOC_FRAGMENT
Document Fragment node

XMLReader::NOTATION
Notation node

XMLReader::WHITESPACE
Whitespace node

XMLReader::SIGNIFICANT_WHITESPACE
Significant Whitespace node

XMLReader::END_ELEMENT
End Element

XMLReader::END_ENTITY
End Entity

XMLReader::XML_DECLARATION
XML Declaration node

XMLReader Parser Options

XMLReader::LOADDTD
Load DTD but do not validate

XMLReader::DEFAULTATTRS
Load DTD and default attributes but do not validate

XMLReader::VALIDATE
Load DTD and validate while parsing

XMLReader::SUBST_ENTITIES
Substitute entities and expand references

XMLReader::close

XMLReader::close -- Close the XMLReader input

Description

bool **XMLReader::close** (void)

Closes the input the XMLReader object is currently parsing.

Return Values

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

See Also

- XMLReader::open
- XMLReader::XML

XMLReader::expand

XMLReader::expand -- Returns a copy of the current node as a DOM object

Description

DOMNode **XMLReader::expand** (void)

This method copies the current node and returns the appropriate DOM object.

Return Values

The resulting DOMNode or **FALSE** on error.

XMLReader::getAttribute

XMLReader::getAttribute -- Get the value of a named attribute

Description

string **XMLReader::getAttribute** (string `$name`)

Returns the value of a named attribute or an empty string if attribute does not exist or not positioned on an element node.

Parameters

`name`
: The name of the attribute.

Return Values

The value of the attribute, or an empty string if no attribute with the given *`name`* is found or not positioned of element.

See Also

- XMLReader::getAttributeNo
- XMLReader::getAttributeNs

XMLReader::getAttributeNo

XMLReader::getAttributeNo -- Get the value of an attribute by index

Description

string **XMLReader::getAttributeNo** (int `$index`)

Returns the value of an attribute based on its position or an empty string if attribute does not exist or not positioned on an element node.

Parameters

`index`
: The position of the attribute.

Return Values

The value of the attribute, or an empty string if no attribute exists at *`index`* or not positioned of element.

See Also

- XMLReader::getAttribute
- XMLReader::getAttributeNs

XMLReader::getAttributeNs

XMLReader::getAttributeNs -- Get the value of an attribute by localname and URI

Description

string **XMLReader::getAttributeNs** (string `$localName`, string `$namespaceURI`)

Returns the value of an attribute by name and namespace URI or an empty string if attribute does not exist or not positioned on an element node.

Parameters

`localName`
: The local name.

`namespaceURI`
: The namespace URI.

Return Values

The value of the attribute, or an empty string if no attribute with the given *`localName`* and *`namespaceURI`* is found or not positioned of element.

See Also

- XMLReader::getAttribute
- XMLReader::getAttributeNo

XMLReader::getParserProperty

XMLReader::getParserProperty -- Indicates if specified property has been set

Description

bool **XMLReader::getParserProperty** (int `$property`)

Indicates if specified property has been set.

Parameters

`property`
: One of the parser option constants.

Return Values

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

See Also

- XMLReader::setParserProperty

XMLReader::isValid

XMLReader::isValid -- Indicates if the parsed document is valid

Description

bool **XMLReader::isValid** (void)

Returns a boolean indicating if the document being parsed is currently valid.

Return Values

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

Examples

Example #1 - Validating XML

```
<?php
$xml = XMLReader::open('test.xml');

// You must to use it
$xml->setParserProperty(XMLReader::VALIDATE, true);

var_dump($xml->isValid());
?>
```

See Also

- XMLReader::setParserProperty
- XMLReader::setRelaxNGSchema
- XMLReader::setRelaxNGSchemaSource

XMLReader::lookupNamespace

XMLReader::lookupNamespace -- Lookup namespace for a prefix

Description

bool **XMLReader::lookupNamespace** (string `$prefix`)

Lookup in scope namespace for a given prefix.

Parameters

`prefix`
: String containing the prefix.

Return Values

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

XMLReader::moveToAttribute

XMLReader::moveToAttribute -- Move cursor to a named attribute

Description

bool **XMLReader::moveToAttribute** (string `$name`)

Positions cursor on the named attribute.

Parameters

name
: The name of the attribute.

Return Values

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

See Also

- XMLReader::moveToElement
- XMLReader::moveToAttributeNo
- XMLReader::moveToAttributeNs
- XMLReader::moveToFirstAttribute

XMLReader::moveToAttributeNo

XMLReader::moveToAttributeNo -- Move cursor to an attribute by index

Description

bool **XMLReader::moveToAttributeNo** (int `$index`)

Positions cursor on attribute based on its position.

Parameters

`index`
: The position of the attribute.

Return Values

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

See Also

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNs
- XMLReader::moveToFirstAttribute

XMLReader::moveToAttributeNs

XMLReader::moveToAttributeNs -- Move cursor to a named attribute

Description

bool **XMLReader::moveToAttributeNs** (string `$localName`, string `$namespaceURI`)

Positions cursor on the named attribute in specified namespace.

Parameters

`localName`
The local name.

`namespaceURI`
The namespace URI.

Return Values

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

See Also

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNo
- XMLReader::moveToFirstAttribute

XMLReader::moveToElement

XMLReader::moveToElement -- Position cursor on the parent Element of current Attribute

Description

bool **XMLReader::moveToElement** (void)

Moves cursor to the parent Element of current Attribute.

Return Values

Returns **TRUE** if successful and **FALSE** if it fails or not positioned on Attribute when this method is called.

See Also

- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNo
- XMLReader::moveToAttributeNs
- XMLReader::moveToFirstAttribute

XMLReader::moveToFirstAttribute

XMLReader::moveToFirstAttribute -- Position cursor on the first Attribute

Description

bool **XMLReader::moveToFirstAttribute** (void)

Moves cursor to the first Attribute.

Return Values

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

See Also

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNo
- XMLReader::moveToAttributeNs
- XMLReader::moveToNextAttribute

XMLReader::moveToNextAttribute

XMLReader::moveToNextAttribute -- Position cursor on the next Attribute

Description

bool **XMLReader::moveToNextAttribute** (void)

Moves cursor to the next Attribute if positioned on an Attribute or moves to first attribute if positioned on an Element.

Return Values

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

See Also

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNo
- XMLReader::moveToAttributeNs
- XMLReader::moveToFirstAttribute

XMLReader::next

XMLReader::next -- Move cursor to next node skipping all subtrees

Description

bool **XMLReader::next** ([string `$localname`])

Positions cursor on the next node skipping all subtrees.

Parameters

`localname`
: The name of the next node to move to.

Return Values

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

See Also

- XMLReader::next
- XMLReader::moveToElement
- XMLReader::moveToAttribute

XMLReader::open

XMLReader::open -- Set the URI containing the XML to parse

Description

bool **XMLReader::open** (string `$URI` [, string `$encoding` [, int `$options`]])

Set the URI containing the XML document to be parsed.

Parameters

`URI`
: URI pointing to the document.

`encoding`
: The document encoding or **NULL**.

`options`
: A bitmask of the XMLReader Options constants.

Return Values

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

ChangeLog

Version	Description
5.2.0	*`encoding`* and *`options`* were added.

See Also

- XMLReader::XML
- XMLReader::close

XMLReader::read

XMLReader::read -- Move to next node in document

Description

bool **XMLReader::read** (void)

Moves cursor to the next node in the document.

Return Values

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

See Also

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::next

XMLReader::setParserProperty

XMLReader::setParserProperty -- Set or Unset parser options

Description

bool **XMLReader::setParserProperty** (int `$property`, bool `$value`)

Set or Unset parser option for the parser. The options must be set after xmlreader-open() or xmlreader-xml() are called and before the first xmlreader-read() call.

Parameters

`property`
One of the parser option constants.

`value`
If set to **TRUE** the option will be enabled otherwise will be disabled.

Return Values

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

XMLReader::setRelaxNGSchema

XMLReader::setRelaxNGSchema -- Set the filename or URI for a RelaxNG Schema

Description

bool **XMLReader::setRelaxNGSchema** (string `$filename`)

Set the filename or URI for the RelaxNG Schema to use for validation.

Parameters

`filename`
: filename or URI pointing to a RelaxNG Schema.

Return Values

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

See Also

- XMLReader::setRelaxNGSchemaSource

XMLReader::setRelaxNGSchemaSource

XMLReader::setRelaxNGSchemaSource -- Set the data containing a RelaxNG Schema

Description

bool **XMLReader::setRelaxNGSchemaSource** (string `$source`)

Set the data containing a RelaxNG Schema to use for validation.

Parameters

`source`
: String containing the RelaxNG Schema.

Return Values

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

See Also

- XMLReader::setRelaxNGSchema

XMLReader::XML

XMLReader::XML -- Set the data containing the XML to parse

Description

bool **XMLReader::xml** (string `$source` [, string `$encoding` [, int `$options`]])

Set the data containing the XML to parse.

Parameters

`source`
: String containing the XML to be parsed.

`encoding`
: The document encoding or **NULL**.

`options`
: A bitmask of the XMLReader Options constants.

Return Values

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

ChangeLog

Version	Description
5.2.0	*`encoding`* and *`options`* were added.

See Also

- XMLReader::open
- XMLReader::close