XSL

Introduction

The XSL extension implements the XSL standard, performing » XSLT transformations using the » libxslt library

Installing/Configuring

Requirements

This extension uses libxslt which can be found at » http://xmlsoft.org/XSLT/. libxslt version 1.1.0 or greater is required.

Installation

PHP 5 includes the XSL extension by default and can be enabled by adding the argument *--with-xsl[=DIR]* to your configure line. *DIR* is the libxslt installation directory.

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.

XSL_CLONE_AUTO (integer)

XSL_CLONE_NEVER (integer)

XSL_CLONE_ALWAYS (integer)

LIBXSLT_VERSION (integer)
libxslt version like 10117. Available as of PHP 5.1.2.

LIBXSLT_DOTTED_VERSION (string)
libxslt version like 1.1.17. Available as of PHP 5.1.2.

LIBEXSLT_VERSION (integer)
libexslt version like 813. Available as of PHP 5.1.2.

LIBEXSLT_DOTTED_VERSION (string)
libexslt version like 1.1.17. Available as of PHP 5.1.2.

Examples

Many examples in this reference require both an XML and an XSL file. We will use *collection.xml* and *collection.xsl* that contains the following:

Example #1 - collection.xml

```
<collection>
<cd>
 <title>Fight for your mind</title>
 <artist>Ben Harper</artist>
 <year>1995</year>
</cd>
<cd>
 <title>Electric Ladyland</title>
 <artist>Jimi Hendrix</artist>
 <year>1997</year>
</cd>
</collection>
```

Example #2 - collection.xsl

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="owner" select="'Nicolas Eliaszewicz'"/>
<xsl:output method="html" encoding="iso-8859-1" indent="no"/>
<xsl:template match="collection">
 Hey! Welcome to <xsl:value-of select="$owner"/>'s sweet CD collection!
 <xsl:apply-templates/>
</xsl:template>
<xsl:template match="cd">
 <h1><xsl:value-of select="title"/></h1>
 <h2>by <xsl:value-of select="artist"/> - <xsl:value-of select="year"/></h2>
 <hr />
</xsl:template>
</xsl:stylesheet>
```

The XSLTProcessor class

Introduction

Description of the class.

Class synopsis

XSLTProcessor

XSLTProcessor {

/* Methods */

string **getParameter** (string `$namespaceURI`, string `$localName`)

bool **hasExsltSupport** (void)

void **importStylesheet** (DOMDocument `$stylesheet`)

void **registerPHPFunctions** ([mixed `$restrict`])

bool **removeParameter** (string `$namespaceURI`, string `$localName`)

bool **setParameter** (string `$namespace`, string `$name`, string `$value`)

DOMDocument **transformToDoc** (DOMNode `$doc`)

int **transformToURI** (DOMDocument `$doc`, string `$uri`)

string **transformToXML** (DOMDocument `$doc`)

}

XSLTProcessor::__construct

XSLTProcessor::__construct -- Creates a new XSLTProcessor object

Description

XSLTProcessor

__**construct** (void)

Creates a new XSLTProcessor object.

Parameters

This function has no parameters.

Return Values

No value is returned.

Examples

Example #3 - Creating an XSLTProcessor

```
<?php

$doc = new DOMDocument();
$xsl = new XSLTProcessor();

$doc->load($xsl_filename);
$xsl->importStyleSheet($doc);

$doc->load($xml_filename);
echo $xsl->transformToXML($doc);

?>
```

XSLTProcessor::getParameter

XSLTProcessor::getParameter -- Get value of a parameter

Description

XSLTProcessor

string **getParameter** (string `$namespaceURI`, string `$localName`)

Gets a parameter if previously set by XSLTProcessor::setParameter().

Parameters

`namespaceURI`
: The namespace URI of the XSLT parameter.

`localName`
: The local name of the XSLT parameter.

Return Values

The value of the parameter or **NULL** if it's not set.

See Also

- XSLTProcessor::setParameter()
- XSLTProcessor::removeParameter()

XSLTProcessor::hasExsltSupport

XSLTProcessor::hasExsltSupport -- Determine if PHP has EXSLT support

Description

XSLTProcessor

bool **hasExsltSupport** (void)

This method determine if PHP was built with the » EXSLT library.

Return Values

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

Examples

Example #4 - Testing EXSLT support

```
<?php

$proc = new XSLTProcessor;
if (!$proc->hasExsltSupport()) {
   die('EXSLT support not available');
}

// do EXSLT stuff here ..

?>
```

XSLTProcessor::importStylesheet

XSLTProcessor::importStylesheet -- Import stylesheet

Description

XSLTProcessor

void **importStylesheet** (DOMDocument `$stylesheet`)

This method import the stylesheet into the XSLTProcessor for transformations.

Parameters

`stylesheet`
The imported style sheet as a DOMDocument object.

Return Values

No value is returned.

XSLTProcessor::registerPHPFunctions

XSLTProcessor::registerPHPFunctions -- Enables the ability to use PHP functions as XSLT functions

Description

XSLTProcessor

void **registerPHPFunctions** ([mixed `$restrict`])

This method enables the ability to use PHP functions as XSLT functions within XSL stylesheets.

Parameters

`restrict`
Use this parameter to only allow certain functions to be called from XSLT. This parameter can be either a string (a function name) or an array of functions.

Return Values

No value is returned.

Examples

Example #5 - Simple PHP Function call from a stylesheet

```
<?php
$xml = <<<EOB
<allusers>
<user>
 <uid>bob</uid>
</user>
<user>
 <uid>joe</uid>
</user>
</allusers>
EOB;
$xsl = <<<EOB
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
    xmlns:php="http://php.net/xsl">
<xsl:output method="html" encoding="utf-8" indent="yes"/>
<xsl:template match="allusers">
 <html><body>
   <h2>Users</h2>
   <table>
   <xsl:for-each select="user">
     <tr><td>
       <xsl:value-of
            select="php:function('ucfirst',string(uid))"/>
     </td></tr>
   </xsl:for-each>
   </table>
 </body></html>
</xsl:template>
</xsl:stylesheet>
EOB;
$xmldoc = DOMDocument::loadXML($xml);
$xsldoc = DOMDocument::loadXML($xsl);

$proc = new XSLTProcessor();
$proc->registerPHPFunctions();
$proc->importStyleSheet($xsldoc);
echo $proc->transformToXML($xmldoc);
?>
```

ChangeLog

Version	Description
5.1.0	The *restrict* parameter was added.

XSLTProcessor::removeParameter

XSLTProcessor::removeParameter -- Remove parameter

Description

XSLTProcessor

bool **removeParameter** (string `$namespaceURI`, string `$localName`)

Removes a parameter, if set. This will make the processor use the default value for the parameter as specified in the stylesheet.

Parameters

`namespaceURI`
: The namespace URI of the XSLT parameter.

`localName`
: The local name of the XSLT parameter.

Return Values

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

See Also

- XSLTProcessor::setParameter()
- XSLTProcessor::getParameter()

XSLTProcessor::setParameter

XSLTProcessor::setParameter -- Set value for a parameter

Description

XSLTProcessor

bool **setParameter** (string `$namespace`, string `$name`, string `$value`)

XSLTProcessor

bool **setParameter** (string `$namespace`, array `$options`)

Sets the value of one or more parameters to be used in subsequent transformations with XSLTProcessor. If the parameter doesn't exist in the stylesheet it will be ignored.

Parameters

`namespace`
The namespace URI of the XSLT parameter.

`name`
The local name of the XSLT parameter.

`value`
The new value of the XSLT parameter.

`options`
An array of *name => value* pairs. This syntax is available since PHP 5.1.0.

Return Values

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

Examples

Example #6 - Changing the owner before the transformation

```
<?php

$collections = array(
   'Marc Rutkowski' => 'marc',
   'Olivier Parmentier' => 'olivier'
);

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

foreach ($collections as $name => $file) {
   // Load the XML source
   $xml = new DOMDocument;
   $xml->load('collection_' . $file . '.xml');

   $proc->setParameter('', 'owner', $name);
   $proc->transformToURI($xml, 'file:///tmp/' . $file . '.html');
}

?>
```

See Also

- XSLTProcessor::getParameter()
- XSLTProcessor::removeParameter()

XSLTProcessor::transformToDoc

XSLTProcessor::transformToDoc -- Transform to a DOMDocument

Description

XSLTProcessor

DOMDocument **transformToDoc** (DOMNode `$doc`)

Transforms the source node to a DOMDocument applying the stylesheet given by the XSLTProcessor::importStylesheet() method.

Parameters

doc
The node to be transformed.

Return Values

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

Examples

Example #7 - Transforming to a DOMDocument

```
<?php

// Load the XML source
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

echo trim($proc->transformToDoc($xml)->firstChild->wholeText);

?>
```

The above example will output:

```
Hey! Welcome to Nicolas Eliaszewicz's sweet CD collection!
```

See Also

- XSLTProcessor::transformToUri()
- XSLTProcessor::transformToXml()

XSLTProcessor::transformToURI

XSLTProcessor::transformToURI -- Transform to URI

Description

XSLTProcessor

int **transformToURI** (DOMDocument `$doc`, string `$uri`)

Transforms the source node to an URI applying the stylesheet given by the XSLTProcessor::importStylesheet() method.

Parameters

doc
The transformed document.

uri

Return Values

Returns the number of bytes written or **FALSE** if an error occurred.

Examples

Example #8 - Transforming to a HTML file

```
<?php

// Load the XML source
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules
```

```
$proc->transformToURI($xml, 'file:///tmp/out.html');

?>
```

See Also

- XSLTProcessor::transformToDoc()
- XSLTProcessor::transformToXml()

XSLTProcessor::transformToXML

XSLTProcessor::transformToXML -- Transform to XML

Description

XSLTProcessor

string **transformToXML** (DOMDocument `$doc`)

Transforms the source node to a string applying the stylesheet given by the xsltprocessor::importStylesheet() method.

Parameters

doc
The transformed document.

Return Values

The result of the transformation as a string or **FALSE** on error.

Examples

Example #9 - Transforming to a string

```
<?php

// Load the XML source
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

echo $proc->transformToXML($xml);

?>
```

The above example will output:

```
Hey! Welcome to Nicolas Eliaszewicz's sweet CD collection!

<h1>Fight for your mind</h1><h2>by Ben Harper - 1995</h2><hr>
<h1>Electric Ladyland</h1><h2>by Jimi Hendrix - 1997</h2><hr>
```

See Also

- XSLTProcessor::transformToDoc()
- XSLTProcessor::transformToUri()