NAME

COM – COM class

SYNOPSIS

$obj = new COM ("Application.ID")

DESCRIPTION

The COM class allows you to instantiate an OLE compatible COM object and call its methods and access its properties.

METHODS

SYNOPSIS

com **COM::COM** (string *$module_name*, [mixed *$server_name*], [int *$codepage*], [string *$typelib*])

COM class constructor. The parameters have the following meanings:

- module_name – Can be a ProgID, Class ID or Moniker that names the component to load. A ProgID is typically the application or DLL name, followed by a period, followed by the object name. e.g: *Word.Application*. A Class ID is the UUID that uniquely identifies a given class. A Moniker is a special form of naming, similar in concept to a URL scheme, that identifies a resource and specifies how it should be loaded. As an example, you could load up Word and get an object representing a word document by specifying the full path to the word document as the module name, or you can use *LDAP:* as a moniker to use the ADSI interface to LDAP.
- server_name – The name of the DCOM server on which the component should be loaded and run. If **NULL**, the object is run using the default for the application. The default is typically to run it on the local machine, although the administrator might have configured the application to launch on a different machine. If you specify a non- **NULL** value for server, PHP will refuse to load the object unless the "" configuration option is set to **TRUE**. If *$server_name* is an array, it should contain the following elements (case sensitive!). Note that they are all optional (although you need to specify both Username and Password together); if you omit the Server setting, the default server will be used (as mentioned above), and the instantiation of the object will not be affected by the "" directive.

DCOM server name

$server_name key	type	description
Server	string	The name of the server.
Username	string	The username to connect as.
Password	string	The password for *$Username*.
Flags	integer	One or more of the following constants, logically OR'd together: **CLSCTX_INPROC_SERVER**, **CLSCTX_INPROC_HANDLER**, **CLSCTX_LOCAL_SERVER**, **CLSCTX_REMOTE_SERVER**, **CLSCTX_SERVER** and **CLSCTX_ALL**. The default value if not specified here is **CLSCTX_SERVER** if you also omit *$Server*, or **CLSCTX_REMOTE_SERVER** if you do specify a server. You should consult the Microsoft documentation for CoCreateInstance for more information on the meaning of these constants; you will typically never have to use them.

- codepage – Specifies the codepage that is used to convert strings to unicode-strings and vice versa. The conversion is applied whenever a PHP string is passed as a parameter or returned from a method of this COM object. The code page is sticky in PHP 5, which means that it will propagate to objects and variants returned from the object. Possible values are **CP_ACP** (use system default ANSI code page - the default if this parameter is omitted), **CP_MACCP**, **CP_OEMCP**, **CP_SYMBOL**, **CP_THREAD_ACP** (use codepage/locale set for the current executing thread), **CP_UTF7** and **CP_UTF8**. You may also use the number for a given codepage; consult the Microsoft documentation for more details on codepages and their numeric values.

OVERLOADED METHODS

The returned object is an overloaded object, which means that PHP does not see any fixed methods as it does with regular classes; instead, any property or method accesses are passed through to COM.

Starting with PHP 5, PHP will automatically detect methods that accept parameters by reference, and will automatically convert regular PHP variables to a form that can be passed by reference. This means that you can call the method very naturally; you needn't go to any extra effort in your code.

In PHP 4, to pass parameters by reference you need to create an instance of the "VARIANT" class to wrap the byref parameters.

PSEUDO METHODS

In PHP versions prior to 5, a number of not very pleasant hacks meant that the following method names were not passed through to COM and were handled directly by PHP. PHP 5 eliminates these things; read the details below to determine how to fix your scripts. These magic method names are case insensitive.

SYNOPSIS

void **COM::AddRef** (void)

Artificially adds a reference count to the COM object.

Warning

You should never need to use this method. It exists as a logical complement to the Release() method below.

SYNOPSIS

void **COM::Release** (void)

Artificially removes a reference count from the COM object.

Warning

You should never need to use this method. Its existence in PHP is a bug designed to work around a bug that keeps COM objects running longer than they should.

PSEUDO METHODS FOR ITERATING

These pseudo methods are only available if **com_isenum**(3) returns **TRUE**, in which case, they hide any methods with the same names that might otherwise be provided by the COM object. These methods have all been eliminated in PHP 5, and you should use "For Each" instead.

SYNOPSIS

variant **COM::All** (void)

Returns a variant representing a SafeArray that has 10 elements; each element will be an empty/null variant. This function was supposed to return an array containing all the elements from the iterator, but was never completed. Do not use.

SYNOPSIS

variant **COM::Next** (void)

Returns a variant representing the next element available from the iterator, or **FALSE** when there are no more elements.

SYNOPSIS

variant **COM::Prev** (void)

Returns a variant representing the previous element available from the iterator, or **FALSE** when there are no more elements.

SYNOPSIS

void **COM::Reset** (void)

Rewinds the iterator back to the start.

COM EXAMPLES

Example #1

COM example (1)

```
<?php
// starting word
$word = new COM("word.application") or die("Unable to instantiate Word");
echo "Loaded Word, version {$word->Version}0;

//bring it to front
$word->Visible = 1;

//open an empty document
$word->Documents->Add();

//do some weird stuff
$word->Selection->TypeText("This is a test...");
$word->Documents[1]->SaveAs("Useless test.doc");

//closing word
$word->Quit();

//free the object
$word = null;
?>
```

Example #2

COM example (2)

```
<?php

$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "0;

for ($i=0; $i < $num_columns; $i++) {
   $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF) {
   for ($i=0; $i < $num_columns; $i++) {
      echo $fld[$i]->value . "";
   }
   echo "0;
   $rowcount++;            // increments rowcount
   $rs->MoveNext();
}

$rs->Close();
$conn->Close();
```

```
$rs = null;
$conn = null;

?>
```