PHP & COM

By Harish Kamath

Table of Contents

Dot Com

You'll remember how, just a few weeks ago, I was playing around with PHP's Java extension, using it to access Java classes and class methods through my PHP scripts. At that time, I thought that PHP's Java extension was a powerful demonstration of the way PHP could be integrated with other technologies. But little did I know that that was just the tip of the iceberg...

You see, in addition to the Java extension, PHP also comes with a very neat little appendage in the form of a COM (that's Component Object Model) extension. A reusable component−based architecture developed by Microsoft, COM makes it possible to do all kinds of cool OO−type things on the Microsoft platform. It's fairly mature, supports many different applications, and is in use on millions of systems worldwide. And − as if that wasn't quite enough − you can now use it with PHP to do all kinds of nifty things through your Web browser.

I'm not going to get into the details of COM here, but will assume you know the basics (in case you don't, there are reams and reams of information on the subject at http://www.microsoft.com/com/). I'll also assume that you have a working PHP−compliant Web server (either Apache or IIS will do), and know the basics of PHP. In case you don't, get yourself organized, and then flip the page to get started.

Striving To Excel

Since COM is a Microsoft invention, it won't surprise you to hear that PHP's COM extension is only available for the Windows version of PHP. If you're using *NIX − well, you're outta luck. Shut down your browser and read a book instead.

If you're still interested, though, pop open your favourite editor and create a PHP script containing the following lines of code (this example assumes that you have a properly−installed copy of Microsoft Excel on your system):

```
<?php

// create an object instance
$excel = new COM("Excel.Application") or die("Excel could not
be
started");

// pop open the Excel application
$excel->Visible = 1;

// turn off alerts
$excel->DisplayAlerts = 0;

// add a workbook
$excel->Workbooks->Add();

// save
$excel->Workbooks[1]->SaveAs("C:\\Inventory.xls");

// close the application
$excel->Quit();

$excel = null;

?>
```

Now, when you run this script through your Web browser, the following things should happen very fast:

1. Microsoft Excel will start up automatically.

2. A new workbook will be added.

3. The workbook will be automatically saved as "C:\Inventory.xls"

4. Microsoft Excel will automatically shut itself down.

Wondering how this happened? Keep reading!

The Number Game

PHP's COM extension makes it easy to access and manipulate COM objects that have been registered on your system. One of these objects is the Excel.Application object, which exposes a number of methods and properties that can be creatively used by an imaginative developer.

The first step to using a COM object in your PHP script involves creating an instance of the COM class; to quote the PHP manual, this class provides you with "...a framework to integrate COM into your PHP scripts".

```
<?php

// create an object instance
$excel = new COM("Excel.Application") or die("Excel could not
be
started");

?>
```

The argument passed to the class constructor is the name of the component to be used. In this case, since I want an instance of Microsoft Excel, I've used the Excel.Application object (more information on this object can be obtained at http://msdn.microsoft.com/library/en-us/modcore/html/deovrworkingwithmicrosoftexcelobjects.asp)

By default, PHP assumes that the component is available on the local server. In the event that you would like the component to be fetched from a remote DCOM server, the server name can be specified as a second, optional argument to the constructor. Note, however, that in order for this to work, the PHP configuration variable

```
com.allow_dcom = true
```

must be set, either in the PHP configuration file or via the ini_set() function call.

Once an object instance has been created, object methods and properties can be accessed using standard OO conventions.

```
<?php

// pop open the Excel application
$excel->Visible = 1;

// turn off alerts
$excel->DisplayAlerts = 0;
```

```
// add a workbook
$excel->Workbooks->Add();

// save
$excel->Workbooks[1]->SaveAs("C:\\Inventory.xls");

// close the application
$excel->Quit();

?>
```

Since the Excel.Application object exposes a number of different methods and properties, you can use it to create workbooks, add sheets to the workbook, add or delete content to the cells of the worksheet, apply formulae to the cells, save the workbook to the disk...in fact, everything that you would be able to do with the application itself.

Keeping this in mind, the example above is fairly simplistic − all it does is create an Excel workbook and save it as a file − but it nevertheless demonstrates how easy it is to do something that, in theory at least, sounds very difficult ("hey Matt, any idea how to create an Excel spreadsheet in Windows using just PHP?").

Export Potential

Let's look at something a little more interesting. I've always considered Microsoft Word to be one of the best editing tools to ever come out of Redmond − and one of its nicest features, especially for Web newbies, is the ability to quickly convert a formatted Word document into an HTML file suitable for use on a Web site.

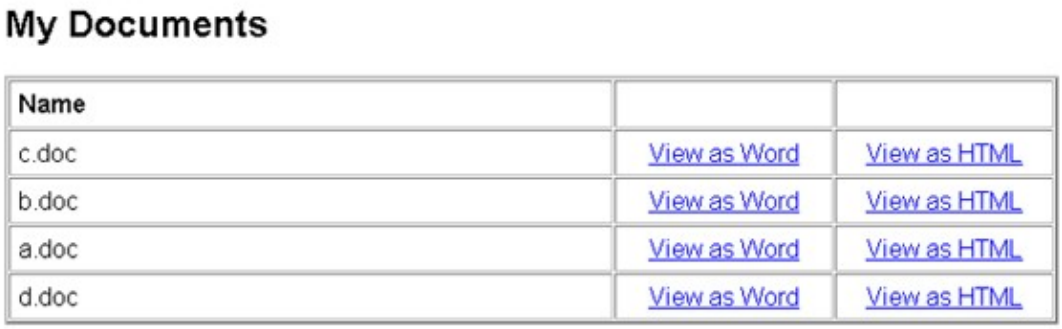
That's where my next example comes in. It uses Word's built−in export−to−HTML feature in combination with a PHP script to give new meaning to the term "tighter Web integration". Watch, and be awed!

The first step here is to write a script that displays a list of all the Word documents available on the system. There's nothing fancy here − this is just very basic usage of PHP's file and directory functions.

```
<?php

// where are the files stored?
$PathToDocumentFolder = "C:\\My Stuff\\Documents";

?>
<html>
<head>
<basefont face="Arial">
<title>My Documents</title>
</head>
<body>
<h2>My Documents</h2>
<table border="2" cellpadding="5" cellspacing="1" width="90%">
<tr>
<th width="55%" align="left">Name</th>
<th width="20%" align="center"> </th>
<th width="20%" align="center"> </th>
</tr>
<?php

// get a handle to the document folder
$dHandle = dir($PathToDocumentFolder);

// iterate through the file list
while($dContent=$dHandle−>read())
{

// only list the .doc files
if(substr($dContent,−4)== ".doc")
{
?>

<tr>
<td width="55%" align="left"><?php echo $dContent;
```

```
?></td>
<td width="20%" align="center"><a href="<?php echo
$PathToDocumentFolder."\\".$dContent; ?>">View as
Word</a></td>
<td width="20%" align="center"><a href="<?php echo
"htmlviewer.php?DocumentPath=".$PathToDocumentFolder."\\".$dContent;
?>">View as HTML</a></td>
</tr>

<?php
}
}

// close the handle to the directory
$dHandle->close();

?>

</table>
</body>
</html>
```

Here's what it looks like:



My Documents

Name		
c.doc	View as Word	View as HTML
b.doc	View as Word	View as HTML
a.doc	View as Word	View as HTML
d.doc	View as Word	View as HTML

The first step here is to initialize a variable containing the location of the directory containing the Word documents.

```
<?php

// where are the files stored?
$PathToDocumentFolder = "C:\\My Stuff\\Documents\\";

?>
```

Next, a table is generated and a PHP "while" loop is used to populate it with a file list of Word documents in that directory.

```
<?php

// get a handler to the document folder
$dHandle = dir($PathToDocumentFolder);

// iterate through the file list
while($dContent=$dHandle->read())
{

// only list the .doc files
if(substr($dContent,-4)== ".doc")
{
?>

<tr>
<td width="55%" align="left"><?php echo $dContent;
?></td>
<td width="20%" align="center"><a href="<?php echo
$PathToDocumentFolder."\\".$dContent; ?>">View as
Word</a></td>
<td width="20%" align="center"><a href="<?php echo
"htmlviewer.php?DocumentPath=".$PathToDocumentFolder."\\".$dContent;
?>">View as HTML</a></td>
</tr>

<?php
}
}

?>
```

Within this code block, the first step is to create a directory handle for the named location. This "pseudo-object" gives me access to several methods that will allows us to read the contents of the directory, move back to the root folder, close this handle and so on.

Once I have the object created, I've used a "while" loop to read the contents of the directory, and an "if" test to filter out all the non-Word files, on the basis of the ".doc" file extension. Each of the files in the final list is then displayed as a separate row in the table, complete with links to view it, either as a Word document within the browser itself, or as an HTML document via the "htmlviewer.php" script.

This "htmlviewer.php" script is where all the meat really is. Let's take a look at that next.

Keeping It Simple

As I'm sure you've figured out by now, converting a Word document to HTML with PHP isn't really as difficult as it sounds. I have no intention of wasting my weekend writing complex search−replace algorithms to perform this task. Instead, I'm going to make my life (and yours) a whole lot simpler by having a Microsoft Word COM object (and its built−in methods) take care of it for me.

```
<?php

// htmlviewer.php
// convert a Word doc to an HTML file

$DocumentPath = str_replace("\\\\", "\\", $DocumentPath);

// create an instance of the Word application
$word = new COM("word.application") or die("Unable to
instantiate
application object");

// creating an instance of the Word Document object
$wordDocument = new
COM("word.document") or die("Unable to instantiate document
object");

// open up an empty document
$wordDocument = $word−>Documents−>Open($DocumentPath);

// create the filename for the HTML version
$HTMLPath = substr_replace($DocumentPath, 'html', −3, 3);

// save the document as HTML
$wordDocument−>SaveAs($HTMLPath, 8);

// clean up
$wordDocument = null;
$word−>Quit();
$word = null;

// redirect the browser to the newly−created document
header("Location:"
. $HTMLPath);

?>
```

As you can see, this is fairly simple, and quite similar to the script I wrote a few pages back for Microsoft Excel. Again, the first step is to use the COM extension to create an instance of the Microsoft Word

application object, followed by an instance of the Word document object

```
<?php

// create an instance of the Word application
$word = new COM("word.application") or die("Unable to
instantiate
application object");

?>
```

Once that's done, the next step is to open up the specified document in Word and use the object's SaveAs() method to save it as HTML.

```
<?php

// open up an empty document
$wordDocument = $word->Documents->Open($DocumentPath);

// create the filename for the HTML version
$HTMLPath = substr_replace($DocumentPath, 'html', -3, 3);

// save the document as HTML
$wordDocument->SaveAs($HTMLPath, 8);

?>
```

Note the second argument passed to the SaveAs() method, the integer 8 - this is a numeric code which tells Word to save the document as HTML. Feel free to experiment with this number and create different file formats - the Web page at http://msdn.microsoft.com/library/en-us/modcore/html/deovrWorkingWithMicrosoftWordObjects.asp has more information on the API for this object.

Once that's done, all that's left is to clean up and redirect the Web browser to the specified HTML file via a call to header().

```
<?php

// clean up
$wordDocument = null;
$word->Quit();
$word = null;

// redirect the browser to the newly-created document
```

```
header("Location:"
. $HTMLPath);

?>
```

Note also the call to str_replace() at the top of the script; this is needed in order to create a valid Windows file path, and remove the extraneous escape characters (slashes) that PHP adds to the GET URL string.

```
<?php

$DocumentPath = str_replace("\\\\", "\\", $DocumentPath);

?>
```

Access−ing The Web

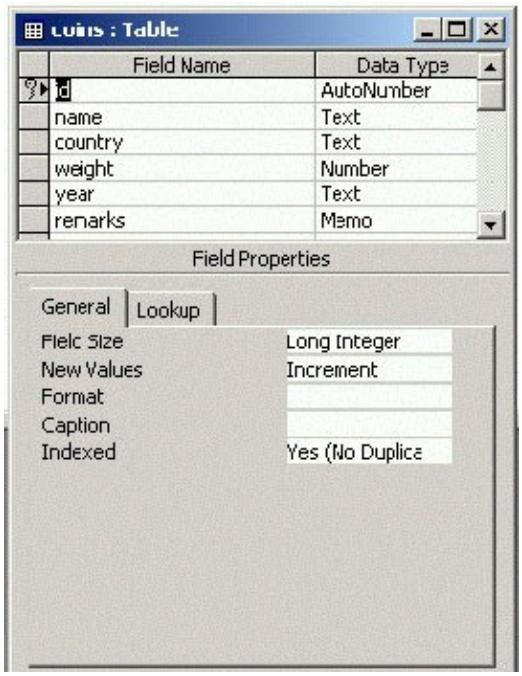
How about one more, this time using the third member of the Microsoft Office family − Microsoft Access?

What do I have in mind for this example? Well, I plan to manipulate (add/edit/delete) records in a Microsoft Access database using a Web browser and a bunch of PHP scripts.

At first glance, this might not seem like a big deal. You're probably already familiar with manipulating a MySQL database with PHP − PHP comes with built−in functions to do this, and the MySQL connectivity in PHP is easily one of the more important reasons for the language's popularity. And you might also be familiar with manipulating Microsoft Access tables via ODBC. But what I have in mind is a little different.

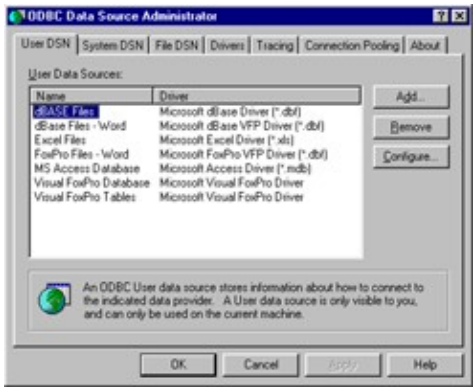
Since this is an article about COM, I don't intend to use either ODBC or PHP's MySQL functions to modify the Microsoft Access database. Instead, I'll be using the methods and properties of the ADODB COM object, in order to demonstrate an alternative technique.

First, here's the Microsoft Access database, holding information on my coin collection in a single table. You may assume that the database is named "phpcom" and the table containing the records is called "coins".

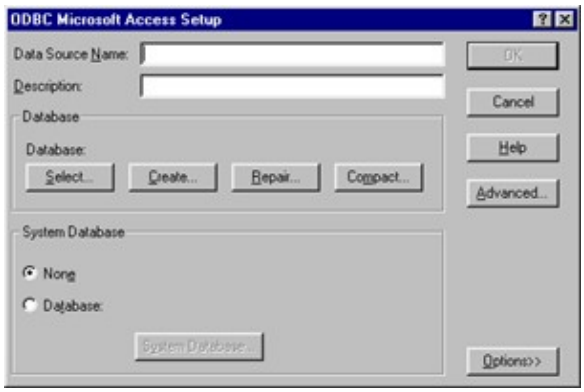


Now, I need to create a Web based interface to modify this database with PHP.

The first step is to create a Data Source Name (DSN) in order to access the database through ODBC. Pop open your Control Panel and use the ODBC module to create this DSN. Tab to the "System DSN" section, and add a new DSN.



Select "Microsoft Access Driver" from the list of available drivers. Set a data source name ("phpcom"), a description ("DSN for PHP/Access connection") and the name of the database to be accessed ("phpcom").



Save your changes, and go back to your PHP editor. It's time to start writing some code.

All For One, And One For All

What I need now is a script that lets me view records, add records, edit records and delete records. For convenience, I'm going to put all these functions into a single script, with a series of "if" tests to determine which function to activate every time the script is executed.

Here's the broad outline of what the script will look like:

```
<?php

// start storing the content in a buffer
ob_start();

// page header

// set some defaults
if(!isset($nextopid) && !isset($previousopid))
{
$nextopid = "list";
$previousopid = "";
}

// make a choice as to what function is needed
if($nextopid == "list")
{
// code to list all records
}
else if($nextopid == "add" || ($nextopid == "edit" &&
isset($id)))
{
// code to display HTML form to add or modify records
}
else if($nextopid == "add_process")
{
// code to add new record
}
else if($nextopid == "edit_process")
{
// code to update selected record
}
else if($nextopid == "delete")
{
// code to delete selected record
}

// page footer
```

```
// display buffer contents
ob_end_flush();

?>
```

As you can see from the code above, there are just two variables used to control the entire script (they're initialized with default values at the top of the script). The more important of the two is the $nextopid variable, which tells the script which operation to perform; it can contain any one of the following values:

"list" − list all records in the table;

"add" − display a form for adding new records;

"edit" − display a form for editing an existing record (must be passed the ID of the record as well);

"add_process" − process the form data and INSERT the record into the table;

"edit_process" − process the form data and UPDATE the record in the table;

"delete" − DELETE the selected record from the table.

Let's look at the code for each of these in detail.

```
<?php

// code to list all records
if($nextopid == "list")
{

// check to see what the previous operation was
// if null, do nothing
// if value exists, display an appropriate message
// to indicate the results of that operation
if(isset($previousopid) && $previousopid != "")
{
switch ($previousopid)
{
case "add_process":
echo "<h3>Record successfully
added</h3>";
break;

case "edit_process":
echo "<h3>Record successfully
updated</h3>";
break;
```

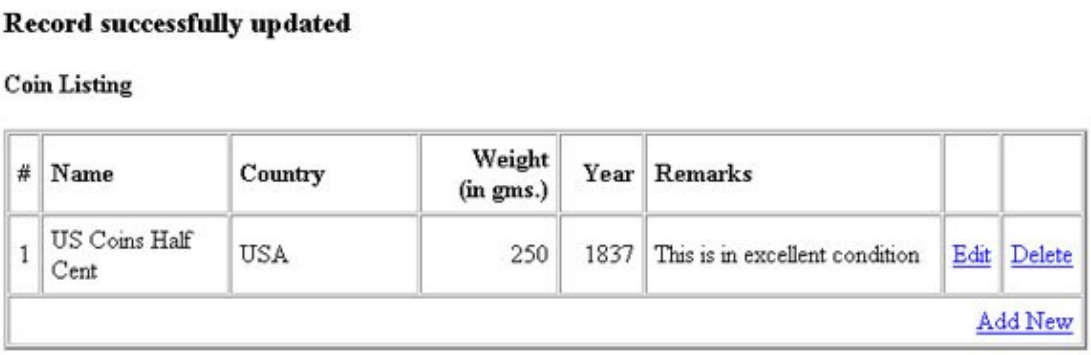
```
case "delete":
echo "<h3>Record successfully
deleted</h3>";
break;

default:
echo " ";
}
}
?>

<h4 align="left">Coin Listing</h4>
<table border="2" cellpadding="5" cellspacing="1" width="90%">
<tr>
<th width="5%" align="right">#</th>
<th width="20%" align="left">Name</th>
<th width="20%" align="left">Country</th>
<th width="15%" align="right">Weight <br>(in gms.)</th>
<th width="5%" align="right" >Year</th>
<th width="20%" align="left">Remarks</th>
<th width="5%" align="right" > </th>
<th width="5%" align="right" > </th>
</tr>

<?php

// open up a connection to the database
$DB_Conn = new COM("ADODB.Connection") or die("Cannot start
ADO");
$DB_Conn->Open("phpcom");

// execute a query
$RS_Record = $DB_Conn->Execute("SELECT * FROM coins");

// iterate through the recordset
while (!$RS_Record->EOF)
{
// get the field data into variables
$id = $RS_Record->Fields('id');
$name = $RS_Record->Fields('name');
$country = $RS_Record->Fields('country');
$weight = $RS_Record->Fields('weight');
$year = $RS_Record->Fields('year');
$remarks = $RS_Record->Fields('remarks');
?>

<tr>
<td width="5%" align="right"><?php echo $id->value;
?></td>
```

```
<td width="20%" align="left"><?php echo $name->value;
?></td>
<td width="20%" align="left"><?php echo $country->value;
?></td>
<td width="5%" align="right"><?php echo $weight->value;
?></td>
<td width="10%" align="right"><?php echo $year->value;
?></td>
<td width="35%" align="left"><?php echo $remarks->value;
?></td>
<td width="5%" align="right" ><a href="<?php echo
$PHP_SELF."?nextopid=edit&previousopid=list&id=".$id->value;?>">Edit</a>
</td>
<td width="5%" align="right" ><a href="<?php echo
$PHP_SELF."?nextopid=delete&previousopid=list&id=".$id->value;?>">Delete
</a>
</td>
</tr>

<?php
// go to the next record
$RS_Record->MoveNext();
}
?>

<tr>
<td colspan="8" align="right"><a href="<?php echo
$PHP_SELF."?nextopid=add&previousopid=list";?>">Add
New</a></td>
</tr>
</table>

<?php

// clean up
$RS_Record->Close();
$DB_Conn->Close();

$RS_Record = null;
$DB_Conn = null;

// end of "list" block
}

?>
```

Over here, I've used COM to create an instance of the ADODB.Connection class − this allows me to open a connection to the DSN created on the previous page. Next, I've executed a query to SELECT all the records

present in the table and store them in a record set. Then I've iterated through the record set and displayed all the fields on a Web page. Note my usage of the Fields() method to do this.

Here's what the output looks like:



Record successfully updated

Coin Listing

#	Name	Country	Weight (in gms.)	Year	Remarks		
1	US Coins Half Cent	USA	250	1837	This is in excellent condition	Edit	Delete
							Add New

Next up, adding and editing coins. Since the form displayed for both these operations is similar, I can combine the two operations into a single block. If the operation is an "add" operation, the form will be displayed with empty fields; if it is an "edit" operation, the record ID passed to the script will be used to query the table for the record and pre−fill the form fields with data.

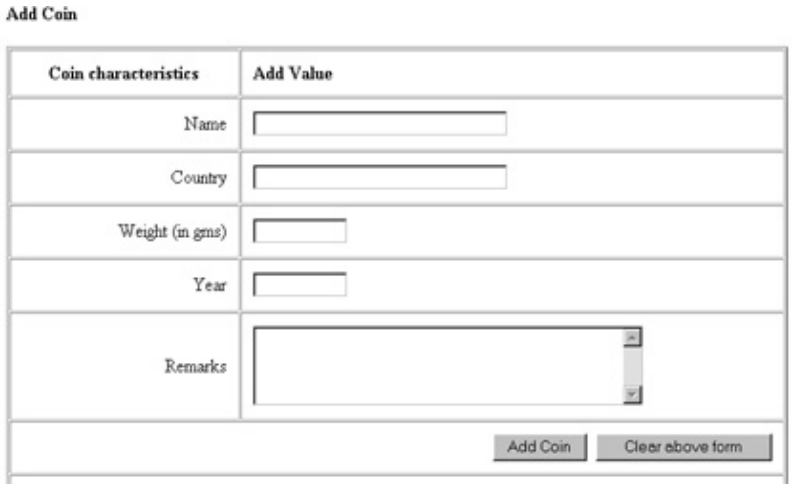
```
<?php

if($nextopid == "add" || ($nextopid == "edit" && isset($id)))
{
// code to display HTML form to add or modify records

?>
<h4 align="left"><?php echo ucfirst($nextopid)." Coin";
?></h4> <?php
if ($nextopid == "edit" && isset($id)){

// if this is an edit operation
// use the ID to retrieve the record from the table
$DB_Conn = new COM("ADODB.Connection") or die("Cannot
start ADO");
$DB_Conn−>Open("phpcom");

// run query
$query = "SELECT name, country, weight, year, remarks
FROM coins where id = $id";
$RS_Record = $DB_Conn−>Execute($query);

// iterate through recordset
while (!$RS_Record−>EOF)
{
$name = $RS_Record−>Fields('name');
```

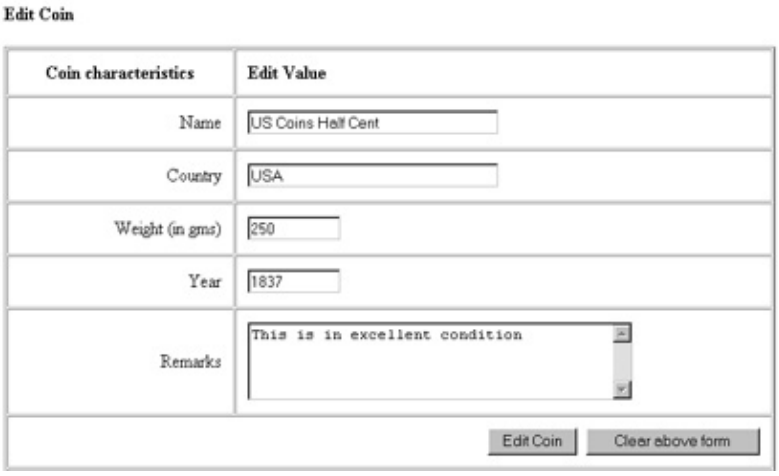
```
$name = $name->value;
$country = $RS_Record->Fields('country');
$country = $country->value;
$weight = $RS_Record->Fields('weight');
$weight = $weight->value;
$year = $RS_Record->Fields('year');
$year = $year->value;
$remarks = $RS_Record->Fields('remarks');
$remarks = $remarks->value;

$RS_Record->MoveNext();
}

// clean up
$RS_Record->Close();
$DB_Conn->Close();

$RS_Record = null;
$DB_Conn = null;
}
?>

<form action="<?php echo $PHP_SELF;?>" method="post" >

<?php if(isset($id)) {?><input type="hidden" name="id"
value="<?php echo
$id;?>"><?php } ?>

<input type="hidden" name="previousopid" value="<?php echo
$nextopid;?>">

<input type="hidden" name="nextopid" value="<?php echo
$nextopid;?>_process">

<table border="2" cellpadding="10" cellspacing="1"
width="90%">
<tr>
<th align="left">Coin characteristics</th>
<th align="left"><?php echo ucfirst($nextopid); ?>
Value</th>
</tr>

<tr>
<td align="right">Name</td>
<td align="left"><input type="text" name="name"
size="30" maxlength="30" value="<?php if(isset($name)) echo
trim($name);?>"></td>
</tr>
```

```
<tr>
<td align="right">Country</td>
<td align="left"><input type="text" name="country"
size="30" maxlength="30" value="<?php if(isset($country)) echo
trim($country);?>"></td>
</tr>

<tr>
<td align="right">Weight (in gms)</td>
<td align="left"><input type="text" name="weight"
size="10" maxlength="10" value="<?php if(isset($weight)) echo
trim($weight);?>"></td>
</tr>

<tr>
<td align="right">Year</td>
<td align="left"><input type="text" name="year"
size="10" maxlength="10" value="<?php if(isset($year)) echo
trim($year);?>"></td>
</tr>

<tr>
<td align="right">Remarks</td>
<td align="left"><textarea name="remarks" cols="40"
rows="4"><?php
if(isset($remarks)) echo trim($remarks);?></textarea></td>
</tr>

<tr>
<td colspan="2" align="right"><input type="submit"
value="<?php echo ucfirst($nextopid);?> Coin"
>  <input
type="reset" value="Clear above form"></td>
</tr>

<tr>
<td colspan="2" align="right"><a href="<?php echo
$PHP_SELF."?nextopid=list"; ?>">Back to listing</a></td>
</tr>
</table>
</form>
<?

// end of "add" or "edit" condition
}
?>
```

Once I know that the user is editing a record, it's trivial to create another ADODB connection to the database to fetch the details of the entry and display a pre−filled form. If, on the other hand, the user is adding a record, I can skip the entire process of fetching the record from the database and just display a blank form.

Here's what the two forms look like:





Note the values of $nextopid in both cases above − for an "add" operation, $nextopid is set to "add_process", and for an "edit" operation, it's set to "edit_process". Let's take a look at those next.

New Coins For Old

If the user chooses to add a new record, the value of $nextopid is set to "add_process". And when the PHP script is called to process the form data, this is the code block that gets executed.

```
<?php

if($nextopid == "add_process")
{

// open connection
$DB_Conn = new COM("ADODB.Connection") or die("Cannot start
ADO");
$DB_Conn->Open("phpcom");

// run query
$Query = "INSERT INTO coins(name, country, weight, year,
remarks)
VALUES('".trim($name)."','".trim($country)."',".trim($weight).",".trim($
year
).",'".trim($remarks)."')";
$RS_Record = $DB_Conn->Execute($Query);

// redirect browser to list
header("Location:".$PHP_SELF."?nextopid=list&previousopid=".$nextopid);

// end of "add" processor
}

?>
```

Similarly, when the script is called to edit a record, this code block will get executed.

```
<?php

if($nextopid == "edit_process")
{

// open connection
$DB_Conn = new COM("ADODB.Connection") or die("Cannot start
ADO");
$DB_Conn->Open("phpcom");

// execute query
$Query = "UPDATE coins SET name = '".trim($name)."',country =
```

```
'".trim($country)."',weight = ".trim($weight).",year =
".trim($year).",remarks = '".trim($remarks)."' WHERE id =
$id";
$RS_Record = $DB_Conn->Execute($Query);

// redirect browser back to list
header("Location:".$PHP_SELF."?nextopid=list&previousopid=".$nextopid);

// end of "edit" processor
}

?>
```

Fairly simple and self−explanatory, this − depending on the type of operation, an appropriate SQL query is generated and executed.

Finally, record deletion. This is activated by calling the script with the value of $nextopid set to "delete", and an additional record ID to identify which record is to be deleted. Take a look:

```
<?php

if($nextopid == "delete")
{

// delete selected records
$DB_Conn = new COM("ADODB.Connection") or die("Cannot start
ADO");
$DB_Conn->Open("phpcom"); $Query = "DELETE FROM coins WHERE id
= $id";
$RS_Record = $DB_Conn->Execute($Query);
header("Location:".$PHP_SELF."?nextopid=list&previousopid=".$nextopid);

// end of "delete" condition
}

?>
```

Short and sweet. And when you try the entire thing out, you'll see that you can add, edit and delete records to the Access database, via the COM object. Ain't it cool?

Link Zone

And that's about it for this week. As you can see, PHP doesn't include hooks for just Java; it also allows you to access Microsoft's COM objects (albeit only on Windows) and use these objects to do all kinds of nifty things. This article examined some of the COM components that are typically available on most systems; however, there's nothing to stop you from buying off−the−shelf components (or rolling your own) and using those components in a PHP script.

In case you're interested in learning more about PHP and COM, consider visiting the documentation on the COM support functions for Windows in the PHP manual, at http://www.php.net/manual/en/ref.com.php... and till next time, stay healthy!

Note: All examples in this article have been tested on Windows 95 with Apache 1.3.20 and PHP 4.1.1. Examples are illustrative only, and are not meant for a production environment. Melonfire provides no warranties or support for the source code described in this article. YMMV!