Using Variable Variables in PHP

By Eric Seufert

Table of Contents

Introduction

Quite possibly one of PHP's least utilized yet most valuable features is its support for Variable Variables. I stumbled across this concept just recently while working on a "Fiction Portal" project for a gaming website. Users of the service could submit their own fictional stories to a program which would then dump them into a MySQL table and populate that table's columns with the characteristics of the story that the author provided. The way the table had been setup was such that each possible characteristic of a story was a separate column, and each story that held that characteristic would hold a value of 1 in the column whereas each story that didn't would hold a value of 0 in it. Therefore, a sample story might look like this in the table:

```
Author Title Date Horror Comedy Adventure
Test "Test Story" 1/1/01 0 0 1
```

Everything went smoothly until I discovered that this method of sorting made searching for stories with only certain criteria nearly impossible. Cue Variable Variables.

Variable Variables?

A variable variable is, in short, a method of using the value of one variable to call the name of another. For instance, if I had a variable $x with the value of "Eric", I could set a variable $Eric to "Seufert" by simply writing $$x = "Seufert". This process calls the value of $x and then creates a variable out of it. Therefore,

```
echo "$x ${$x}";
```

would produce the same output as,

```
echo "$x $Eric";
```

which is:

```
Eric Seufert
```

Notice the curly braces around the variable variable in the first echo statement. This is the standard syntax for outputting variable variables. Omitting the curly braces would have resulted with the following output:

```
Eric $Eric
```

Using Variable Variables with Arrays

Variable Variables make working with forms and other such user−input easier because they allow you to centralize all the available options. Getting back to my aforementioned problem, I had to figure out a way to select only the stories with the criteria that the person selected. The easiest way to go about this was to create an array to hold only the stories that fit the criteria the user specified and then create a MySQL query string with the values. Here's the code I used:

```
$atts = array('comedy', 'tragedy', 'horror', 'adventure',
'poetry', 'screenplay', 'satire');

$vals = array();
for ($i = 0; $i < sizeof($atts); $i++) {
if (${$atts[$i]} == 1) {
$last = sizeof($vals);
$vals[$last] = "$atts[$i]";
}
}
$query_build = "";
for ($i = 0; $i < sizeof($vals); $i++) {
$curval = "$vals[$i] = '1'";
if ($i != (sizeof($vals) − 1)) {
$query_build .= $curval." OR ";
} else {
$query_build .= $curval;
}
}
```

(*note* The system running the program did not have PHP 4.0 installed at the time, which is why I did not just use array_push to add values to the array in the first loop)

The actual values of the variable variables created from the $atts array having been already passed to the program, everything was accomplished in a few lines of code rather than replicating the same process for each different characteristic, showing the efficiency of variable variables when used with forms.

Building arrays from variable variables is equally simple. If, perhaps, I wanted to include my middle and last name within the $$x variable, I could initialize it as an array with the following code:

```
$$x = array("Benjamin", "Seufert");
```

However, array elements cannot be directly accessed in variable variable form, so outputting my entire name could be handled in one of two ways:

```
$temp1 = ${$x}[0];
$temp2 = ${$x}[1];
echo "$x $temp1 $temp2";
```

or,

```
echo "$x $Eric[0] $Eric[1]";
```

both of which would print,

```
Eric Benjamin Seufert
```

Had I tried to directly print the variable variable array elements, I would have received the following output:

```
Eric Array[0] Array[1]
```

Variable Variables with Functions

A great yet widely unused element of variable variables is their ability to be employed with functions. The function name can be set within a variable and then run through calling the variable variable of that variable. Here's an example:

```
$function1 = "sort_stuff";
${$function1()}; #this sets the result of the sort_stuff()
function into a variable
${$function1($var1, $var2, $var3)}; #this does the same thing
but sends the
function parameters
```

Here's a quick example of this in action, building on the same program segment I had introduced variable variables with:

```
$x = "Eric";
$$x = "Seufert";

$function = "value";

echo "${$function()}";

function value() {
return "Eric";
}
```

This code would output:

```
Seufert
```

as the echo statement is printing the value of the variable that gets created from the resultant of the value() function, "Eric". As we have already set $$x (which, following the variable variable, is the same as $Eric) to

"Seufert", the same value is outputted, only this time resulting from a function call.

Variable Variables with Classes

Variable Variables work equally well with class objects, the basic syntax of which is almost the same as the use of variable variables with functions. Set the class name into a variable, call the variable variable of that variable and you then have access to variables within that class. Here's an example:

```
class Eric {
var $foo = "bar";
}

$Eric = new Eric;
$v = "Eric";
echo $$v->foo;
```

the output of which would be:

```
bar
```

The Never−ending Variable

Another interesting quirk about variable variables is that they are seemingly infinite. The following code works just fine to produce "The Great Thing About Variable Variables Is They Never End!":

```
$x = "The";
$$x = "Great";
$$$x = "Thing";
$$$$x = "About";
$$$$$x = "Variable";
$$$$$$x = "Variables";
$$$$$$$x = "Is";
$$$$$$$$x = "They";
$$$$$$$$$x = "Never";
$$$$$$$$$$x = "End!";

echo "$x ${$x} ${${$x}} ${${${$x}}} ${${${${$x}}}}
${${${${${$x}}}}} ${${${${${${$x}}}}}} ${${${${${${${$x}}}}}}}
${${${${${${${${$x}}}}}}}} ${${${${${${${${${$x}}}}}}}}}";
```

Conclusion

Variable Variables are one of those little workarounds that make writing code that much easier. Hopefully you have picked up some helpful new techniques from this article to carry with you into practical, everyday use.