PHP3 Introduction

By W.J. Gilmore

Table of Contents

PHP. What is it?

PHP3.0, one of the hottest scripting languages to be found on the Internet, lends a great many capabilities to the web programmer. Many of these tasks, accomplished with some degree of difficulty in many other languages, can be swiftly executed with but a few lines of PHP3.0 code . The fact that PHP3.0 code can be inserted directly alongside HTML makes the language all the more convenient.

Created by Rasmus Lerdorf in what began as a personal project (titled PHP/FI), the language quickly gained popularity and was almost completely rewritten by a group of six developers, and in turn, reborn as PHP3.0. The language enjoys an extremely active developing environment, due, in large part, to the fact that the language is freely available for download on the web.

Much of PHP3.0 is a combination of Perl, Java, and C concepts. The syntax structure borrows heavily from C, making it an easy language to learn for even the novice programmer. PHP3.0 performs sophisticated mathematical calculations, provides network information, offers mail and regular expression capabilities, and much more. PHP3.0's strongest feature is its database interfacing capability. Connecting a database to the Internet has never been so easy. What's more, it supports many of the most popular database servers on the market, including MySQL, Oracle, Sybase, mSQL, Generic ODBC, and PostgreSQL, to name a few. In fact, the possibilities left to the user as a result of this interface are so great, we will focus upon this subject in the second half of the article. In particular, we will discuss its interfacing capabilities with MySQL, perhaps the most powerful database server found on the market today.

One of the factors that make PHP3.0 so powerful is that is a goal−oriented language. It is written to accomplish things, quickly and cleanly. Before you read on, however, make sure to take note of the following important points:

1. PHP3.0 is what is known as a server−side scripting language. Thus the language interpreter must be installed and configured on the server before one can execute commands. Contact your ISP for more information, or if you are the server administrator, go to http://www.php.net to download the product.
2. As stated earlier, PHP3.0 is a complete rewrite of the previous version, titled PHP/FI or PHP2.0. Be sure to ask your ISP which version is installed; if it happens to be the old version, ask if they could upgrade. In short, PHP3.0 is better. It is much faster and more capable than its predecessor. In addition, all syntax within this article will be written using PHP3.0 syntax.
3. Once installed and configured, it is wise to obtain a copy of the PHP3.0 documentation. This documentation highlights all of the available commands and syntax structures, and will serve as an irreplaceable guide to learning the language.

With that in mind, let's begin..

Getting your feet wet

Again, PHP3.0 commands can be easily inserted alongside HTML. This provides an easy way to incorporate dynamic information within what was previously a static document. For example, if we would like to insert the current date into the HTML document that visitor has called, we might do the following:

```
<HTML> <HEAD> <TITLE>Our first PHP3.0 script</TITLE> </HEAD> <BODY>
<CENTER>Our first PHP3.0 script</CENTER> <? /* the above "<?" signals that the PHP
script has begun */ $today = date("Y−m−d"); PRINT "<CENTER>Today is:
$today.</CENTER>"; # the following "?>" closes the script ?> </BODY> </HTML>
```

Assuming today is August 27, 1998, our output would be:

Our first PHP3.0 script

Today is: 1998−08−27.

A few points to make:

1. All PHP3.0 commands must be enclosed within the "<?" and "?>" brackets. A second method to denote PHP3.0 commands is by enclosing them within "<?php...?>" brackets.
2. Comments can be made within the script, and are enclosed within "/*" and "*/" brackets, or by placing a "#" sign at the beginning of the line.
3. All statements that are to be outputted to the screen must be enclosed in double quotations ("), and led by the PRINT statement.
4. Almost every PHP3.0 command must end in a semi−colon (;).
5. Any HTML commands placed within the PRINT statements will be interpreted by the browser, and perform their usual actions.
6. Documents including PHP3.0 statements must be saved with the extension *.php3 (For example, myphpfile.php3). This tells the PHP3.0 interpreter to execute any commands found within the script.
7. In short, the date command operates as follows:
   Syntax: string date(string format, int timestamp);

   The function can take two variables (timestamp) is optional. If supplied with a timestamp, the function returns a string containing a date formatted according to the parameters within the format string. In the above example, Y−m−d signifies year, month, and day, respectively, all in numerical format. There are many other format characters that can be placed within the format string. For a complete list, click here.

The include statement

Another very powerful use of PHP3.0 is the capability of building HTML templates, which are very useful when one is developing a cohesive site with many pages. Let's assume one would like to place a footer at the bottom of each page, for example:

But what happens when the site increases in size to say 50 pages, and suddenly the footer must include another statement? This would involve the modification of each page, one by one, taking up a lot of time and effort.

However, PHP3.0 offers the include statement, which allows one to include a separate file within an HTML document. Let's insert the above copyright statement into a separate text file, and save it as "footer.txt". Then, one only has to insert the following command at the bottom of the HTML file:

```
<? include("footer.txt")?>
```

and the footer will appear when loaded into the browser! Supposing the site grows to 50 pages, and that copyright needs to be modified. The only step the developer has to take is to open the footer file, make the necessary modification, and Voila!. 50 pages are immediately updated.

Interesting, huh? Now, let's look at some of the more dynamic aspects of the language, incorporating HTML forms and variables passing/modification.

HTML forms and variables

Another powerful feature of PHP3.0 is it's capability of modifying variables passed from HTML forms. With these variables, one can accomplish many a feat, including such tasks as: sending web−based email, outputting information to the screen, and passing data to and from a database. Let's construct a small automated email program, demonstrating many of these capabilities:

Let's assume we have the following HTML form:

```
<HTML> <HEAD> <TITLE>Request for more information</TITLE> <BODY>
<CENTER>Would you like more information about our company? <P> <TABLE WIDTH =
400><TR><TD align = right> <FORM ACTION="email.php3" METHOD="POST"> Your
name:<BR> <INPUT TYPE="text" NAME="name" SIZE="20" MAXLENGTH="30"> <P>
Your email address:<BR> <INPUT TYPE="text" NAME="email" SIZE="20"
MAXLENGTH="30"> <P> I prefer: <SELECT NAME="preference"> <OPTION value =
Apples>Apples <OPTION value = Oranges>Oranges </SELECT> <P> <INPUT
TYPE="submit" VALUE="Send it!"> </FORM> </TD></TR></TABLE></CENTER>
</BODY> </HTML>
```

Save the above HTML as `moreinfo.html`.

Notice that the ACTION points to the file *email.php3*. This file will contain the PHP3.0 script that will carry out several commands.

The `email.php3` file:

```
<? /* this script will handle the variables passed from the moreinfo.html file */ PRINT
"<CENTER>"; PRINT "Hello, $name."; PRINT "<BR><BR>"; PRINT "Thank you for your
interest.<BR><BR>"; PRINT "We will send information to $email, and have noted that you
like $preference."; PRINT "</CENTER>"; ?>
```

Don't forget to save the above file as email.php3.

When the user types in their name and email within the HTML form, and presses the "Send it!" button, the form will call the email.php3 file, in turn returning the following output (assuming the person's name is Bill, has the email address bgates@devshed.com, and likes Apples):

Hello, Bill.

Thank you for your interest.

We will send information to bgates@devshed.com, and have noted that you like Apples.

However, our project is not yet complete, as we would not know who had inserted any information, since we are not keeping any records of what happened. Thus we would not be able to send Bill an email.

One way to do so, in turn lessening the work load of sending standard email messages manually, is by implementing PHP3.0's MAIL() command.
Syntax: void mail(string to, string subject, string message, string add_headers);

- to − the whom the email is directed.
- subject − phrase to be inserted in subject line of email message.
- message − the actual message.
- add_headers − use this to insert a string at the end of the header. (optional)

Thus, if we inserted the following commands after the last PRINT statement in the preceding script, we could automatically send email both to the person requesting information, and to the site administrator, letting us know who requested the information:

```
<? mail("$email", "Your request for information", "$namen Thank you for your interest!n We sell fresh corn daily over the Internet! Place your order at http://www.buycorn.com, and receive a free package of $preference!"); mail("administration@buycorn.com", "Visitor request for info.", "$name requested for information.n The email address is $email. n The visitor prefers $preference."); ?>
```

Important Note: The MAIL() function only works when SENDMAIL is installed on the server. In most cases, it is. However, be sure to check with your ISP before attempting to use this function.

But what happens when so many users are entering information, that you can't keep track of all of the emails? Or, you require a system to track how many prefer Apples to Oranges? The implementation of a database aids greatly in the administration of these types of data. One of the fastest database server on the market, MySQL, is a great choice for reasons of speed and ease of use, as well as it's flexibility and compatibility with PHP3.0.

The next section will deal with the incorporation of the MySQL database with PHP3.0.

MySQL database interfacing

In order to perform these functions, MySQL must be installed and configured on the server. If you are not familiar with MySQL, it is strongly recommended that you read Devshed's MySQL tutorials before continuing on with this article.

Basically, PHP acts as an intermediate between the database server and the web browser, communicating commands from one to another. This communication greatly increases the possibility of interactivity, be it in the form of polls, user−personalized sites, online submission systems, or any other activity requiring user input.

This interaction starts by making a connection to the MySQL database. This connection is accomplished using the following command:
Syntax: int mysql_connect(string hostname, string username, string password);

- hostname − the hostname of the database. (i.e. Devshed's would most likely be: www.devshed.com
- username − the username set to connect to the MySQL database.
- Password − the password set to connect to the MySQL database.
- The int returned is positive if the connection is successful, negative if unsuccessful.

All commands must, as always, be placed within the "<?" and "?>" brackets.

Returning to the user information project, let's suppose the following table has been created within the MySQL database:

```
mysql> CREATE TABLE information (     −> name VARCHAR (25),     −> email
VARCHAR (25),     −> choice VARCHAR (8) );
```

Now, assume we want to insert the user's information into the MySQL database. This can be accomplished by modifying the *email.php3* script to be the following:

```
<? /* this script will handle the variables passed from the moreinfo.html file */ /* declare
some relevant variables */ $hostname = "devshed"; $username = "myusername"; $password =
"mypassword"; $dbName = "mydbname"; /* MySQL table created to store the data */
$userstable = "information"; /* the site administrator's email address */ $adminaddress =
"administration@buycorn.com"; /* make connection to database */
MYSQL_CONNECT($hostname,$username,$password) OR DIE("Unable to connect to
database"); @mysql_select_db("$dbName") or die("Unable to select database"); PRINT
"<CENTER>"; PRINT "Hello, $name."; PRINT "<BR><BR>"; PRINT "Thank you for your
interest.<BR><BR>"; PRINT "We will send information to $email, and have noted that you
like $preference"; PRINT "</CENTER><BR><BR>"; /* Send relevant emails */
mail("$email", "Your request for information", "$name\nThank you for your interest!\n We
```

```
sell fresh corn daily over the Internet! Place your order at http://www.buycorn.com, and
receive a free package of $preference!"); mail("$adminaddress", "Visitor request for info.",
"$name requested for information.\n
The email address is $email. \n The visitor prefers $preference."); /* Insert information into
table */ $query = "INSERT INTO $userstable VALUES('$name','$email', '$preference')";
$result = MYSQL_QUERY($query); PRINT "Your information has also been inserted into
our database, for future reference."; /* Close the database connection */ MYSQL_CLOSE();
?>
```

Some notes:

1. The variables declared at the beginning of the script are for the MYSQL_CONNECT() function. We also could have plugged the values directly into the function. However, as projects begin to grow in size, these values will probably be placed in a separate file and included (using the #include statement) within the file. Thus making for easy modification should any information need to be edited.
2. The @mysql_select_db() function selects a database. Doing this saves some time later in the script, allowing one to execute query statements without having to nominate a database name.
   Syntax: int mysql_select_db(string database_name, int link_identifier);
   * database_name must be one of the database names appearing on the server.
   * link_identifier (optional) refers to the link to the database, for reason of making a connection. If one is not designated, the last opened link is used.
   * returns a true/false value, based on success.
3. The MYSQL_QUERY() function is then used to send queries to the MySQL database.
   Syntax: int mysql_query(string query, int link_identifier);
   * query − the query that you would like to send to the database.
   * link_identifier − the database name. (optional. If not included, the query will use the most recently opened database) Or, if one chooses no to use the @mysql_select_db() function, one must nominate a database name.
   * Returns an integer. Positive if successful, negative if unsuccessful.
4. The MYSQL_CLOSE statement closes the connection to the MySQL database. In the following syntax, the link_identifier refers to a database name. If it is not specified, it will close the last opened database.
   Syntax: int mysql_close(int link_identifier);
   * link_identifier − same as above.
   * Returns an integer. Positive if successful, negative if unsuccessful.

View the script.

If you have actually set this up on your server and executed it, you will now see that the information has in fact been added to the information table. In the next section, we will learn how to extract this data from the MySQL database, and display it onto the screen.

Outputting data from MySQL

Ok, we have had great success with our request for user information, and a considerable amount of information is now stored within the database. Yet, how does one go about searching through this data, and returning useful results from it? This is accomplished through the execution of three steps:

1. Request for data fitting the required restraints.
2. Tally of all results fitting restraints.
3. Output of data fitting restraints.

We are interested in outputting to the browser all users that prefer Apples to Oranges.

```
<? /* script to output to screen all users preferring Apples to Oranges */ /* declare some relevant variables */ $hostname = "devshed"; $username = "myusername"; $password = "mypassword"; $userstable = "information"; $dbName = "mydbname"; /* make connection to database */ MYSQL_CONNECT($hostname, $username, $password) OR DIE("Unable to connect to database"); @mysql_select_db( "$dbName") or die( "Unable to select database"); /* Select all users with the preference Apples */ $query = "SELECT * FROM $userstable WHERE choice = 'Apples'"; $result = MYSQL_QUERY($query); /* How many of these users are there? */ $number = MYSQL_NUMROWS($result); /* Print these results to the screen in a nice format */ $i = 0; IF ($number == 0) : PRINT "<CENTER><P>Nobody in the database prefers Apples!</CENTER>"; ELSEIF ($number > 0) : PRINT "<CENTER><P>Users preferring Apples: $number<BR><BR>"; WHILE ($i < $number): $name = mysql_result($result,$i,"name"); $email = mysql_result($result,$i,"email"); PRINT "Visitor $name likes Apples.<BR>"; PRINT "Email address: $email."; PRINT "<BR><BR>"; $i++; ENDWHILE; PRINT "</CENTER>"; ENDIF; ?>
```

Save this as `apples.php3`.

Explanation of new functions:

```
$number = MYSQL_NUMROWS($result);
```

Syntax: int mysql_num_rows(string result);
* result − the array of records returned from the MYSQL_QUERY function.
* returns the numerical value of the number of rows that have been stored within $result.

Finally, we output the results of this search. If there is nobody within the database that likes Apples, output will ensue stating so; If there are users within the database preferring Apples, each user name and email will be duly outputted. This output takes place through a WHILE statement.

```
$name = MYSQL_RESULT($result,$i,"name");
```

Syntax: int mysql_result(int result, int i, column);
The mysql_result() function breaks apart each record, placing each piece into a variable. In short,
* $result refers to the array in question
* $i refers to the particular count that we have arrived.
* column is the actual name of the column within the MySQL table. Can also be the offset value, or the field table dot field name (i.e. fieldname.tablename).

Thus, using a simple WHILE loop, we are capable to easily output data to the browser.

View the script.

Other functions:

Using the MYSQL_QUERY() function, one is able to execute a number of SQL functions. Among these include the DELETE and UPDATE functions.

Delete
Let's assume we want to delete all records containing the name "Bunny":

```
$query = "DELETE FROM $userstable WHERE name = "Bunny";
MYSQL_QUERY($query);
```

Update
Or perhaps we want to modify all records containing the name "Bunny":

```
$query = "UPDATE $userstable SET name = "Bugs Bunny" WHERE name = "Bunny";
MYSQL_QUERY($query);
```

In this article, we have gotten quite a start learning how to incorporate PHP3.0 into web site development. We have learned how to create dynamic pages, as well as how to use PHP3.0 to tie a MySQL database to the web. However, this is but a small piece of what PHP3.0 is capable of.

Perhaps the most wise advice I can give in reference to PHP3.0 is to read; Rather memorize, the documentation. Written by PHP's authors, there is no other resource available that can sufficiently demonstrate the capabilities of PHP as well.

The documentation, as well as the latest news and other resources about PHP3.0 is readily available at

http://www.php.net

MySQL documentation, resources, and news are available at http://www.mysql.com

In addition, become an active member of the PHP mailing list. Simply by reading some of the questions/answers posted each day, you will quickly become very familiar with the intricacies of the language.