PHP Tutorial -- Adapted from web by J. Scott 2008

Part 1 – Introduction to PHP

1.1 Introduction

Up until recently, scripting on the internet was something which very few people even attempted, let alone mastered. Recently though, more and more people have been building their own websites and scripting languages have become more important. Because of this, scripting languages are becomming easier to learn and PHP is one of the easiest and most powerful yet.

1.2 What Is PHP and Why Do I Want to Use PHP?

PHP stands for Hypertext Preprocessor and is a server-side language. This means that the script is run on your web server, not on the user's browser, so you do not need to worry about compatibility issues. PHP is relatively new (compared to languages such as Perl (CGI) and Java) but is quickly becomming one of the most popular scripting languages on the internet.

You may be wondering why you should choose PHP over other languages such as Perl or even why you should learn a scripting language at all. I will deal with learning scripting languages first. Learning a scripting language, or even understanding one, can open up huge new possibilities for your website. Although you can download pre-made scripts from sites like Hotscripts, these will often contain advertising for the author or will not do exactly what you want. With an understanding of a scripting language you can easily edit these scripts to do what you want, or even create your own scripts.

Using scripts on your website allows you to add many new 'interactive' features like feedback forms, guestbooks, message boards, counters and even more advanced features like portal systems, content management, advertising managers etc. With these sort of things on your website you will find that it gives a more professional image. As well as this, anyone wanting to work in the site development industry will find that it is much easier to get a job if they know a scripting language.

What Do I Need?

As mentioned earlier, PHP is a server-side scripting language. This means that, although your users will not need to install new software, you web host will need to have PHP set up on their server. It should be listed as part of your package but if you don't know if it is installed you can find out using the first script in this tutorial. If you server does not support PHP you can ask your web host to install it for you as it is free to download and install. If you need a low cost web host which supports PHP I would recommmend HostRocket.

1.3 Writing PHP and Declaring PHP

Writing PHP on your computer is actually very simple. You don't need any special software, except for a text editor (like Notepad in Windows). Run this and you are ready to write your first PHP script.

PHP scripts are always enclosed in between two PHP tags. This tells your server to parse the information between them as PHP. The three different forms are as follows, and all accomplish the same thing: let's get PHP started for our use.

Php simple start	More Definitive Start	Start in Script Format
<? PHP Code In Here ?>	<?php PHP Code In Here php?>	<script language="php"> PHP Code In Here </script>

Table 1: 3 Ways to Start PHP Rolling

All of these work in exactly the same way but in this tutorial we will use the first and simplest option (<? and ?>). You can use either of the options. You must remember, though, to start and end your code with the same tag, e.g., if you start with <? and end with </script>, your code will not work properly.

1.4 Writing and Finishing Your First Script

The first PHP script you will be writing is very basic. All it will do is print out all the information about PHP on your server. Type the following code into your text editor:

```
<?
phpinfo();
?>
```

As you can see, your first PHP script is actually just one line of code. It is a standard PHP function called phpinfo which will tell the server to print out a standard table of information giving you information on the setup of the server.

One other thing you should notice in this example is that the line ends in a semicolon. This is very important. As with many other scripting and programming languages nearly all lines are ended with a semicolon and if you miss it out you will get an error.

Now you have written your first PHP script, save it as phpinfo.php and upload it to your server in the normal way. Now, using your browser, go to the URL of the script, which will be "yourwebsite.address/phpinfo.php". Once you have done this, and PHP is installed on your server, you should get a huge page full of the information about PHP on your server.

If your script doesn't work and a blank page displays, you have either mistyped your code or your server does not support PHP. Normally, you can install PHP yourself if it is not already there, but some web hosting company packages do not support PHP. If, instead of a page being displayed, you are prompted to download the file, PHP is not installed on your server and you should either search for a new web host or ask your current web hosting company to install PHP.

It is a good idea to keep this script for future reference.

Part 2 - Displaying Information & Variables

2.1 Introduction

In Part 1, we explained some of the advantages of PHP as a scripting language and showed you how to test your server for PHP. Now, we will learn the basics of showing information in the browser and how you can use variables to hold information.

2.2 Printing Text

Having PHP script output text is straightforward, and you can do it in a variety of different ways. We will concentrate on a commonly used way, i.e., the PHP print command. Print will allow you to output text, variables, or a combination of the two so that they display on the screen. The print statement is used in the following way:

```
print("Hello world!");
```

Print is the PHP command. What is in the parentheses is what is PHP will print, in this case, Hello world! Because you are outputting literal text, you enclose it inside quotation marks. Finally, the print command must end in a semicolon. You would, of course, have to enclose this in your standard PHP tags, making the following code:

```
<?
print("Hello world!");
?>
```

Which will display Hello world! on the screen.

2.3 Defining PHP Variables and a Few Rules for PHP Variables

PHP allows you to define many types of variables, but the most common is called a String. It can hold text and numbers. All strings begin with a $ sign. To assign some text to a string you would use the following code:

```
$welcome_text = "Hello and welcome to my website.";
```

Everything inside the quotation marks will be assigned to the string. You must remember a few rules about strings though. First and foremost, following the unix tradition, string variables are case sensitive. Thus "$Welcome_Text" is not the same as "$welcome_text ". String names can contain letters, numbers and underscores, but unlike some programming languages, PHP string variables cannot begin with a number or underscore. Numbers a numbers, so when assigning numbers to strings, you do not need to include the quotes. Thus $user_id = 987 would be allowed.

2.4 Outputting Variables and Formatting Your Text

To display a variable on the screen uses exactly the same code as to display text but in a slightly different form. The following code would display your welcome text:

```
<?
$welcome_text = "Hello and welcome to my website.";
print($welcome_text);
?>
```

As you can see, the only major difference is that you do not need the quotation marks if you are printing a variable.

Unfortunately, accepting the default output from your PHP programs can quite boring, as everything is just output in the browser's default font. To add more zest to your outputs, you can use HTML formatting. Since PHP is a server side language, the code is executed before the page is sent to the browser. This means that only the resulting information from the script is sent, so in the example above the browser would just be sent the text in a plain format.

Hello and welcome to my website.

You can include standard HTML markup, but many HTML tags require the " sign. This will normally clash with the quotation marks used to print your text. This means that you must tell the script which quotes should be used (the ones at the beginning and end of the output) and which ones should be ignored (the ones in the HTML code).

For this example, we will change the text to the Arial font in red. The normal code for this would be:

```
<font face="Arial" color="#FF0000"></font>
```

As you can see this code contains 4 quotation marks so would confuse the script. Because of this you must add a backslash before each quotation mark to make the PHP script ignore it. The code would change to:

```
<font face=\"Arial\" color=\"#FF0000\"></font>
```

You can now include this in your print statement:

```
print("<font face=\"Arial\" color\"#FF0000\">Hello and welcome
to my website.</font>");
```

which will make the browser display: Hello and welcome to my website.

because it has only been sent the code:

```
<font face="Arial" color="#FF0000">Hello and welcome to my website.</font>
```

This seems tedious, but later we will show easier another way to output text without the messy backslashes.

Part 3 – Using IF Statements with PHP

3.1 Introduction

Over the past two parts I have shown you the basics of text in PHP and how to store it as variables. In this part of the tutorial I will show you how to use IF statements to make decisions in your scripts.

3.2 The Basics Of IF

If statements are used to compare two values and carry out different actions based on the results of the test. If statements take the form IF, THEN, ELSE. Basically the IF part checks for a condition. If it is true, the then statement is executed. If not, the else statement is executed. The structure of an IF statement is as follows:

```
IF (something == something else)
{
THEN Statement
} else {
ELSE Statement
}
```

The most common use of an IF statement is to compare a variable to another piece of text, a number, or another variable. For example, the if test,

```
if ($username == "webmaster")
```

which would compare the contents of the variable to the text string. The THEN section of code will only be executed if the variable is exactly the same as the contents of the quotation marks so if the variable contained 'Webmaster' or 'WEBMASTER' it will be false.

3.3 Constructing The THEN and ELSE Statements

To add to your script, you can now add a THEN statement:

```
if ($username == "webmaster") {
echo "Please enter your password below";
}
```

This will only display this text if the username is webmaster. If not, nothing will be displayed. You can actually leave an IF statement like this, as there is no actual

requirement to have an ELSE part. This is especially useful if you are using multiple IF statements. To add an ELSE statement, just add some extra code:

```
if ($username == "webmaster") {
echo "Please enter your password below";
} else {
echo "We are sorry but you are not a recognised user";
}
```

Of course, you are not limited to just one line of code. You can add any PHP commands in between the curly brackets. You can even include other IF statements, which are called nested statements.

3.4 Using Conditional Operators in a PHP IF Statement

There are other ways you can use your IF statement to compare values. You could use the standard equal operator (==) to compare two different variables to see if their values match, as in,

```
if ($enteredpass == $password)
```

You can also use the standard comparison symbols (<, <=, >, >=) to check to see if one variable is greater than or less than another:

```
if ($age < "13")  or  if ($date > $finished)
```

You can also check for multiple tests in one IF statement. In this case, the two vertical bars ||, mean OR. For instance, if you have a form and you want to check if any of the fields were left blank you could use:

```
if ($name == "" || $email == "" || $password == "") {
echo "Please fill in all the fields";
}
```

Part 4 – Using Loops and Arrays in PHP

4.1 Introduction

Previously, this tutorial has showed you how to deal with text and variables in PHP and how to use IF statements to compare them and to make decisions. In Part 4, we are going to show you how to use an indispensable part of PHP, loops.

4.2 The WHILE Loop

The WHILE loop is one of the most useful commands in PHP. A WHILE loop will, as the name suggests, execute a piece of code until a certain condition is met. If you want to repeat a piece of code several times without retyping it, you can use a while loop. Say you simply wanted to print out the words "Hello World" 5 times. Here is the PHP code to achieve this command:

Code Line Number	Actual PGP Code	Explanation of the PGP Code
1	$times = 5;	Sets up Variable $times to equal 5
2	$x = 0;	Sets $x variable to equal 0
3	while ($x < $times) {	Begins While Loop, will keep doing it until $x is equal to 5
4	echo "Hello World";	Each time through, hello world comes out
5	++$x;	Increment $x by one, same as $x = $x + 1;
6	}	End of WHILE loop

Table 2: PGP While Loop Code Explained

The following PGP code types out every single one or use the following code:

Line	PGP Code	Explanation of the PGP Code
1	$times = 5;	Sets up Variable $times to 1000
2	$current = 0;	Sets $current variable to equal 1000
3	while ($current < $number) {	Begins While Loop,
4	++$current;	Increment $current by 1
5	Echo $current ;	Print value of $current and new line
6	}	End of WHILE loop

Table 3: Second Code Snippet - Print out 1000 lines

4.3 Setting Up and Using Arrays with PGP

Arrays are common to many programming languages. They are special variables which can hold more than one value, each stored in its own numbered 'space' in the array. Arrays are extremely useful, especially when using WHILE loops.

Setting up an array is slightly different to setting up a normal variable, because in addition to defining the contents, we must also define the placeholder value in the square brackets []. Below, we setup an array called $names, with five names in it. As you can see, the parts of an array are all numbered, starting from 0.

```
$names[0] = 'John';  $names[1] = 'Paul';  $names[2] = 'Steven';
$names[3] = 'George'; $names[4] = 'David';
```

4.4 Reading From An Array using PHP

Reading from an array is just the same as putting information in. All you have to do is to refer to the array and the number of the piece of data in the array. So if I wanted to print out the third name I could use the code:

```
echo "The third name is $names[2]";
```

Which would output: The third name is Steven

4.5 Using Arrays And Loops in PHP

One of the best uses of a loop is to output the information in an array. For instance if I wanted to print out the following list of names in our array, the code is:

Line	PGP Code	Explanation of the PGP Code
1	$number = 5;	Sets up Variable $number to be 5
2	$x = 0;	Sets $x to be 0
3	while ($x < $number) {	Begins While Loop,
4	++$current;	Increment $current by 1
5	echo "Name $namenumber is $names[$x] ";	Print value of $current and new line
6	}	End of WHILE loop

Table 4: Printing Out Array Elements

Part 5 - E-mail With PHP

5.1 Introduction

One of the major uses of a server side scripting language is to provide a way of sending e-mail from the server and, in particular, to take form input and output it to an e-mail address. In this part, you will learn how to send e-mail messages using PHP.

5.2 The PGP Mail Command for Sending E-Mails

Sending e-mail using PHP is straightforward. There is actually just one command, mail() for sending mail. The syntax for the PHP mail() command is as follows:

```
mail($to,$subject,$body,$headers);
```

This command uses descriptive variable names, but you could also just place text in the mail command. The following table explains the four parameters:

Parameter	Explanation of this parameter
$to	Contains the e-mail address of the intended recipient
$subject	The normal subject for the e-mail (such as “Scheduled Meeting”)
$body	The actual text of the e-mail
$headers	Used for any additional information, such as the From field, or other headers, such as cc or bcc.

Table 5: PGP Mail command Parameters Explained

5.3 Sending An E-mail with PGP

Before sending your mail, if you are using variables, you must, of course, set up the variable content beforehand. Here is some simple code for sending a message:

```
$to = "php@gmail.com";
$subject = "PHP Is Great";
$body = "PHP is one of the best scripting languages around";
$headers = "From: webmaster@gmail.com\n";
mail($to,$subject,$body,$headers);
echo "Mail sent to $to";
```

Table 6: PGP Mail Command Example

This code will acutally do two things. Firstly it will send a message to php@gmail.com with the subject 'PHP Is Great' and the text:

PHP is one of the best scripting languages around

and the e-mail will be from webmaster@gmail.com. Secondly, it outputs the text:
Mail sent to php@gowansnet.com to the browser.

5.4 Formatting E-mail and Controlling E-mail errors

You may have noticed that the From line ended with \n in our e-mail example. This character tells PHP to take a new line in an e-mail. It is very important that this is put in after each header you add so that your e-mail will follow the international standards and will be delivered. The \n code can also be used in the body section of the e-mail to put line breaks in but should not be used in the subject or the To field.

The e-mail above could have been sent using different variable names (it is the position of the variables in relation to the commas, not the name of them which decides on their use). It could also have been done on one line using text like this:

```
mail("php@gmail.com","PHP Is Great","PHP is one of the best scripting
languages around","From: webmaster@gmail.com\n");
```

This type PGP code works, and could be used for quick-and-dirty items and testing, but is not recommended. If you have lots of emails in a list retrieved from a database, you need to use the first form of our email.

Because scripts “run right now”, it is easy to make coding mistakes, such as inputting an invalid e-mail address. Good e-mail programming practice is to add in a small piece of code which will check if the e-mail is sent. The snippet below echoes a correct message or a problem back to the sender.

```
if(mail($to,$subject,$body,$headers)) {
echo "An e-mail was sent to $to with the subject: $subject";
} else {
echo "There was a problem sending the mail. Check your code
and make sure that the e-mail address $to is valid";
}
```

Table 7: E-mail Error Checking with PGP

Part 6 - PHP With Forms

6.1 Introduction

In Part 6, we will show you how to use PHP and forms together to make your PHP scripts useful.

6.2 Setting Up Your Web Form

Setting up a form for use with a PHP script is exactly the same as normal in HTML. We will discuss the three main pieces of code you must know: First, we show a simple text box

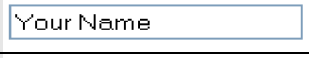
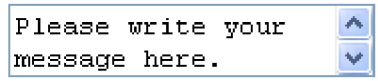
PGP Code	Resulting Display
<input type="text" name="thebox" value="Your Name">	
<textarea name="message"> Please write your message here.</textarea>	
<input type="submit" value="Submit">	

Table 8: HTML Code and a What It Renders to Your Browser

The value section in the first code is optional. The information defined by name will be the name of this text box, relative to your PHP code, and should be unique. The second snippet adds a text box that you can add information to, which is quite useful and you have seen this in many web forms. The third code snippet adds a submit button to your form. Forms for database use are discussed in the PGP/MySQL tutorial.

All the elements for your form must be enclosed in the <form> tags. They are used as follows:

```
<form action="process.php" method="post">
Form elements and formatting etc.
</form>
```

The form's action tells PGP what script to send its data to for processing. In our example, it is process.php. This can also be a full URL, such as

http://www.mysite.com/scripts/private/processors/process.php.

The method normally tells the form how to submit its data. The POST method shown in our form snippet will send the data in a data stream to the script when it

is requested. GET is the other option. GET will send the form data in the form of the url so it would appear after a question mark e.g.

http://www.mysite.com/process.php?name=david

It is normally better to use POST if you are using passwords or sensitive information as they should not be shown in the browser's address bar.

6.3 Getting The Form Information

The next step is understanding how to get the data your form has submitted into your script so that you can do something with it. There are basically two different methods of getting the data into PHP, which depend on how they were submitted. The two submission methods, GET and POST, can both be used by forms. The difference between the two is that using GET, the variables and data will be shown in the page address, but using POST it is invisible. The benefit of GET, though is that you can submit information to the script without a form, by simply editing the URL.

This works the same as submitting a form using GET. The advantage of this is that you can create links to your scripts which do different things depending on the link clicked. For example you could create a script which will show different pages depending on the link clicked:

yourpage.php?user=david could show David's page and:
yourpage.php?user=tom could show Tom's page, using the same script.

It is also possible to pass more than one piece of information to the script using this system by separating them with the & symbol:

yourpage.php?user=david&referrer=gmail&area=6

These could be accessed separately using the GET variables user, referrer and area.

To get a variable which has been sent to a script using the POST method you use the following code: `$variablename=$_POST['variable'];` which basically takes the variable from the POST (the name of a form field) and assigns it to the variable $variablename.
Similarly, if you are using the GET method you should use the form:
`$variablename=$_GET['variable'];` This should be done for each variable you wish to use from your form (or URL).

6.4 Creating The Form To Mail Script

Finally, we will not put several items together to show you how to create a system which will e-mail a user's comments to you. First off, create the form shown below for your HTML page. Note the extra COLS=40 ROWS=6 parameter in the text box. There are many optional HTML command parameters, such as the one shown here.

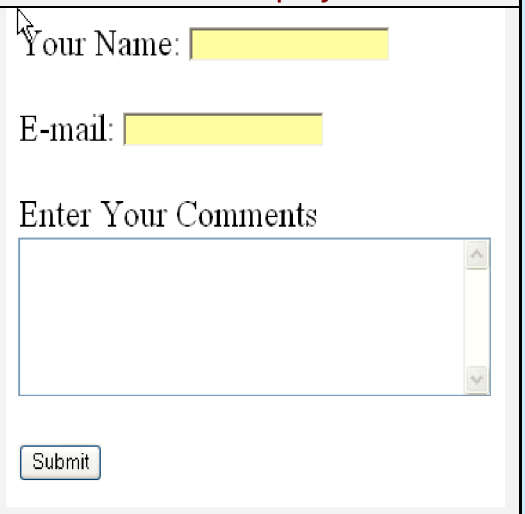
PHP Code for the E-mail form	How this Code Displays on Screen
<form action="mail.php" method="post"> Your Name: <input type="text" name="name"> E-mail: <input type="text" name = "email"> Enter Your Comments <textarea name="comments" COLS=40 ROWS=6></textarea> </textarea> <input type="submit" value="Submit"> </form>	

Table 9: An E-mail form and its Display for Use with PHP

This will make a simple form where the user can enter their e-mail address, their name and their comments. You can, of course, add extra parts to this form but remember to update the script too.

6.5 Creating the Script to Run the E-Mail Form

We are almost finished putting the two items together. Next, we have to create a script to run the form. Since this is for E-mail, it also contains some items to be more secure and avoiding spam.

Now create the PHP script:

PGP Script Code	Explanation
<? function checkOK($field) { if (eregi("\r",$field) || eregi("\n",$field)){ die("Invalid Input!"); } } $name=$_POST['name']; checkOK($name); $email=$_POST['email']; checkOK($email); $comments=$_POST['comments']; checkOK($comments); $to="php@gmail.com"; $message="$name just filled in your comments form. They said:\n$comments\n\nTheir e-mail address was: $email"; if(mail($to,"Comments From Your Site",$message,"From: $email\n")) { echo "Thanks for your comments."; } else { echo "There was a problem sending the mail. Please check that you filled in the form correctly."; } ?>	This eregi code piece stops spammers from using your form to send their spam messages by checking special characters are not present in the input. These could be used to trick the computer into sending messages to other addresses. It is a function which checks for these characters, and if they are found, stops running the script. After this, each variable is checked for validity.

Table 10: PGP Code Working with Form for E-Mail

Remember to replace php@gmailt.com with your own e-mail address. This script should be saved as mail.php and both should be uploaded. Now, all you need to do is to fill in your comments form.

Part 7 - Final Notes

7.1 Introduction

This tutorial was designed to lead you through the basics of writing PHP. In this final section, we will discuss a few items that don't really warrant a section of their own.

7.2 Using Comments with PHP

PHP is no different from any other programming language: good programmers must comment their work and their scripts. If you are working on a script with someone else you must let them know what you code does. If you plan to distribute your script, others will often need to edit your work. This means that documentation is called for. Even if you are the only one who will use your script it is useful to comment so that you can edit it at a later date. Programmers have a very short memory for intricate pieces of code and good commenting is the only way to quickly go back to change a snippet here and there.

PHP supports both single and multiple line comments. A single line comment starts with a pair of forward slashes, and is the simplest to explain. It looks like this: `// Your comment can go in here` Everything after the // will be ignored when the script is executed. You can even place these on the end of another line e.g. `print "Hello $name"; // Welcome to the user .`

Multiline comments follow the old “C style: and look like the following:

```
/* The following piece of code will take the input
the user gave and will check that it is valid before
adding it to the database */
```

Anything between the /* and the */ will be ignored. You must close this type of comment, as not doing so could make your script not work properly.

7.3 Print, Echo and HTML

During this tutorial, we have used 4 different ways of outputting information to the browser:

```
echo("Text here");
echo "Text here";
print("Text here";
print "Text here";
```

Table 11: Four Forms of Outputs to Browser

To clarify, four commands do the same thing. You can use any or all of them in a script. There is no reason to even use the same type all through a script. The only problem you may find is that all the " in the HTML code must be replaced with \" which, if you have a lot of code, could take a very long time.

Suppose the code you need has three parts as shown in the table below.

PHP Code	Explanation of Why You Flipped PHP
<? Top PHP code in here ?> HTML Code <? Bottom PHP code in here ?>	First PHP is on. Remember the " must be replaced with \". But with no real HTML here, you are okay. Then turn PHP off, so that the HTML is okay. Then turn PHP back on, again with little HTML.

Table 12: Flipping PHP On and Off

Your top part is dynamic PHP, then you have STATIC HTML, then you finish with dynamic PHP. This gets even better as the PHP code will just continue when you flip it. In a complex PHP statement, like a big IF – ELSE, flipping PHP on and off can be quite helpful.

```
<?
IF Statement {
?>
HTML For IF Being Correct
<?
} else {
?>
HTML For IF Being Wrong
<?
}
?>
```

Always remember to close IF statements and loops, though, as it is very easy to forget.

7.4 One Line Prints

Being able to place HTML code into your PHP is very useful, but what happens if you want to put the value of a variable into the code. Unlike when using an echo or print statement, you can't just put in the variable name as this section is not actually part of the PHP code. Instead you must just put in a little PHP.

For example if you wanted to print someone's name from a script with HTML formatting you would do the following:

```
<font face="Arial" size="7" color="red"><b><? echo($variablename); ?></b></font>
```

In the above code you have just added in the following PHP:

`<? echo($variablename); ?>` which is exactly the same as the following PHP code:

```
<?
echo($variablename);
?>
```

but all put onto one line.

7.5 Conclusion

This tutorial has given you some of the basics of PHP and should allow you to do many useful things with PHP. For more depth, visit PHP.net, the official homepage of PHP. Our companion tutorial, PhPMySQL, shows how to allow these two tools to interoperate. These capabilities are part of many LAMP solutions: Linux, Apache, MySQL, and PHP. Good Luck with PHP!