



Lesson #3

PHP Basics

By

Lynette Chandler
Tech Based Training

<http://TechBasedTraining.com/>

**NOTICE: You Do NOT Have the Right
to Reprint or Resell this Report!**

You Also MAY NOT Give Away,
Sell or Share the Content Herein

If you obtained this report from anywhere other than **Tech Based Training.com** , you have a pirated copy.

Help stop internet piracy by letting me know. Send email to
customerservice@techbasedsupport.com

© 2010 Copyright Lynette Chandler

Legal Disclaimer: Keeping things simple here, I have to say that I cannot promise you success. I can give you direction and advice based on my experiences and good internet marketing practices. What you do with this information is up to you. As a Tech Based Training member you agree to not hold me responsible for your results.

About The Sandbox Blog...

Since the last lesson, I've received some questions about the purpose of the sandbox blog.

Will I have to build a sandbox for every client?

The answer is no.

The sandbox is strictly for your own use. When you're developing or editing themes, there will be a period especially in the beginning phases where you'll want to work offline. This way, you can work with some peace especially when you get a client who likes to snoop and point out things you're already working on. Snooping clients can often slow down your initial development considerably.

You need to be able to load the themes somewhere while you're working on it, hence the need for a sandbox blog. You also don't have to build a new sandbox blog for every client. All you need to do is simply switch out the themes in WordPress dashboard.

For example, in my sandbox blog, there are always around 5-6 different themes in the themes folder. When I need to work on the theme I just activate it. When I'm ready to reveal a draft to the client, I simply upload it to the client's blog for them to see. Then, the client and I will discuss the specifics, their likes and dislikes and make changes accordingly.

Hope that clears the air. Now, let's get back to this week's lesson...

Beginnings of PHP

This week, we're going to set aside WordPress and the Sandbox. You will need to have your server or web host account in order. So if you have not set up Uniform Server or a web hosting account do so now or you won't be able to move forward. If you have already set it up, start your server.

Recognizing PHP Code

PHP code is enclosed in the following tags

```
<?php ?>
```

Whenever you see these tags, that means the web server will attempt to process anything in between those tags as PHP code. It is also necessary to name your files with the php extension (filename.php). If you name your file with any other extension the PHP code will not be processed.

Hello World!

The first step is to learn how to display text in your browser. For this exercise, we are going to develop a web page using PHP that will display the text "Hello World!" when loaded in your browser.

Open PsPad. On the menu, click File >> New File >> Find PHP >> OK

You should now see a brand new tab with the following <?php ?> code in it. In between the opening (<?php) and closing (?>) tags, type the following.

```
echo "Hello World!";
```

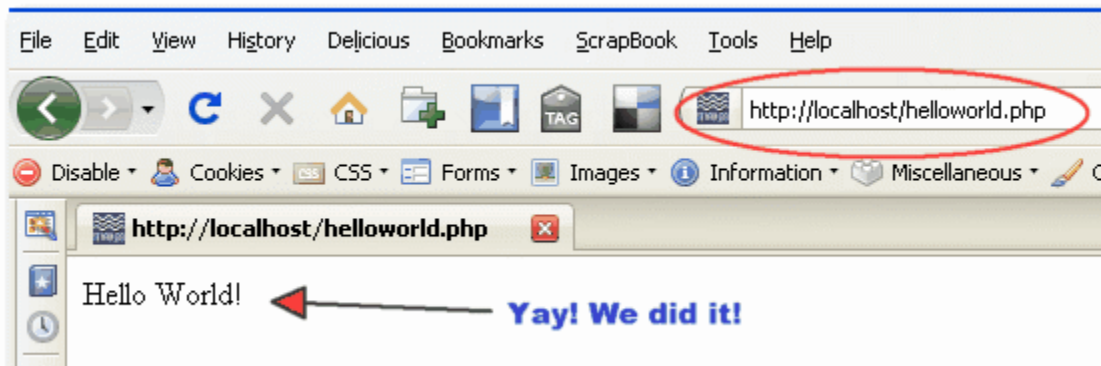
This is called an echo statement. Notice that the text (also referred to in the programming world as strings) are enclosed in quotation marks. Text should always be enclosed in quotation marks. Also note the semi-colon. A semi-colon indicates the end of a line of code, essentially like the period at the end of a sentence.

Your code should look like this.

```
1 <?php
2     echo "Hello World";
3
4 ?>
5
```

Save the file to your www folder (in Uniform Server remember?). Keep in mind to save the file in .php extension. I've saved mine as helloworld.php inside the www folder (C:\Program Files\Uniform Server\udrive\www)

Now, when I go to my browser, and type in <http://localhost/helloworld.php>, this is what I would see.



That was easy right?

Just like a normal website, pay attention where you save your file. In this example, if you saved your file in the same folder as I did, you will point your browser to <http://localhost/helloworld.php>. If you saved your file into

*C:\Program Files\Uniform
Server\drive\www\someotherfolder then you would point
your browser to
<http://localhost/someotherfolder/helloworld.php>*

Another Way To Echo Text

In PHP, there are two ways to echo (or display) text. One you've already seen above. The other way is to write your code like this.

```
<?php ?>  
Hello World
```

Notice now that the string (text) is completely outside of the PHP tags. When you close a PHP tag, it knows to stop processing and start displaying text or HTML in the browser.

WordPress themes generally use the 2nd method of display HTML. Either way will work but if you plan on distributing your theme and for the simple sake of keeping in line with WordPress standards, we'll be using the 2nd method from now on.

You may be wondering what's the purpose of this? Good question. Separating PHP from HTML is actually a tad more versatile. Also, it allows you to isolate HTML from PHP processing. But let's not jump too far ahead. It'll become clearer to you shortly.

Since I've briefly talked about HTML, you may be wondering, how do you add HTML in this Hello World page? You do it this way:

```
<?php  
echo "<html><body><h1>Hello  
World!</h1></body></html>";  
?>
```

Try it.

As you can see, the code above is pretty long. Imagine when you have a lot of HTML in your document it's going to be a challenge making sure your quotes are closed properly. This is why the 2nd method is cleaner and clearer to the human eye. It doesn't matter that much to the server as line breaks and spaces are ignored.

```
<html>
<body>
<h1>Hello World!</h1>
</body>
</html>
<?php ?>
```

Note that in this example, my php code is below the HTML. It doesn't matter for this page because we aren't processing any PHP commands yet. Only HTML. Once we actually have something for PHP to process, then it becomes important. So let's make it do something.

Variables

Variables are little bits of information you can use to display in a web page. The nice thing about variables are, you can use it across different pages. Let's say we are building two simple pages for Blog Theme Bootcamp. I want the name "Blog Theme Bootcamp" to be used throughout the site. I also want a footer with the copyright year on all pages. Instead of editing each page manually, we'll use variables.

Declaring Variables

Variables can be written on the web page itself. Let's go back to helloworld.php in PsPad to add the following to the top of the page, before any HTML:

```
<?php
```

```
$sitename = "<i>Blog Theme Bootcamp</i> ";  
$copyright = "Copyright &copy; 2008";  
?>
```

A few explanations

1. Always declare or write out the variables before you use it. If you declare the variables after, it won't work. This is why the variable is on the very top of the page in our example.
2. Variables have a name (\$sitename) which is always prefixed with a \$ sign. You can name your variable anything you like as long as it makes sense and is easy for you.
3. Variable names are case sensitive \$sitename is not the same as \$SiteName so be careful. Have a plan when you are naming them and be consistent.
4. Variables always equate to something. (= "Blog Theme Bootcamp"). This is the data. Data is something you want to be displayed or a statement. Data can also be another variable but for now, we will keep things simple with text only.
5. Always end your variables with a semicolon (;) failing to do so will break your code.

Using Our Variables

Now we have declared them we want to use it at the end of the page. To display the variable, use this code.

```
<?php  
echo $copyright;  
echo $sitename;  
?>
```


When echoing a variable, you don't need to put it in quotes.

```
1 <?php
2 $sitename = "<i>Blog Theme Bootcamp</i> ";
3 $copyright = "Copyright &copy; 2008";
4 ?>
5 <html>
6 <body>
7 <h1>Hello World!</h1>
8 <?php
9     echo $copyright;
10    echo $sitename;
11 ?>
12 </body>
13 </html>
```

Our final code put together looks like this

And the result is this



Pretty neat huh? But remember we wanted to use the variables on other pages as well. These variables aren't very useful right now because they reside on the same

file as helloworld.php. I cannot use those variables if I build a new page (contact.php). How do we solve that problem?

We could copy the variables into the contact.php file. That would solve the problem but it would be a site management nightmare when we start adding more pages. We certainly don't want to edit every single file when we change our copyright notice.

A better way is to create yet another file and name it variables.php, move the variables into that file and include it in each new page we create. Sound complicated? Don't worry. Let's do the next exercise and you'll see how this all makes sense.

Need Some One-on-One Help?

Every Blog Theme Bootcamp lesson is written with simplicity in mind. I try not to cram too many concepts and jargon into one lesson and strive to write in simple everyday English.

But sometimes, you just want someone to talk you through these things. A live person to clarify and ask a question, get immediate answers and feedback.

Now You Can!

I'm offering personal one-on-one help by telephone. Get all the details from

<http://blogthemebootcamp.com/oneonone.php>

Require & Include

Require and Include is a feature of PHP that allows you to well... include files. If you are familiar with web page building, they work like Frontpage's included content or ServerSideIncludes (SSI).

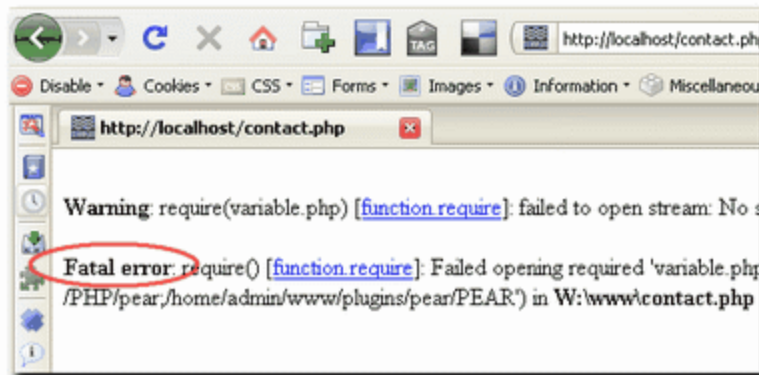
When you require or include a file, you're telling PHP to bring the data from the file you specify into the file it's displaying. This is a great way to share data

between pages – like headers, footers, menus. It also makes life for us web developers a whole lot easier.

Require and include commands look like this.

```
<?php include 'variables.php'; ?>  
<?php require 'variables.php' ?>
```

What's the difference between them? Actually, very little. The only difference is, if PHP cannot file the file, require will give you a *Fatal Error* and stop processing the rest of the code.



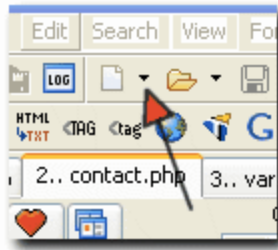
While require will still produce a *Warning* but continues processing the code.



Which you use will depend on your intent or whatever makes the most sense. If you absolutely need the data in order to display a page properly, use require. If the data is not critical to the page then use include. This is not a strict rule. Just something that's at your discretion as a developer. The standard for WordPress theme development is to use include so that's what we'll focus on.

Back to our pages. First thing we do is create a new PHP file then move the variables into that file.

1. In PsPad, click File >> New File >> PHP >> OK. You can also click the drop down next to this icon to open a new php file.

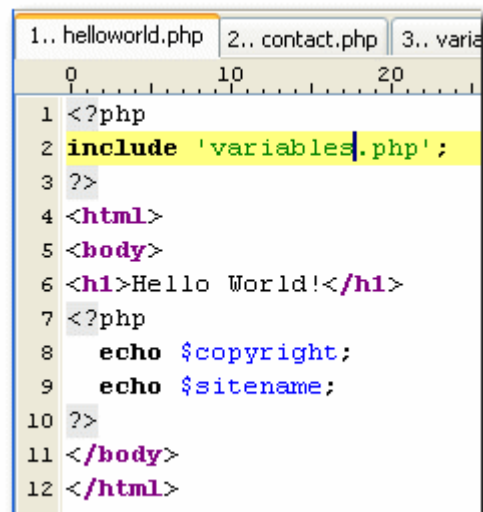


2. Highlight the variables from helloworld.php, cut and paste them into this



new document.

3. Go back to helloworld.php to include the variables.php file



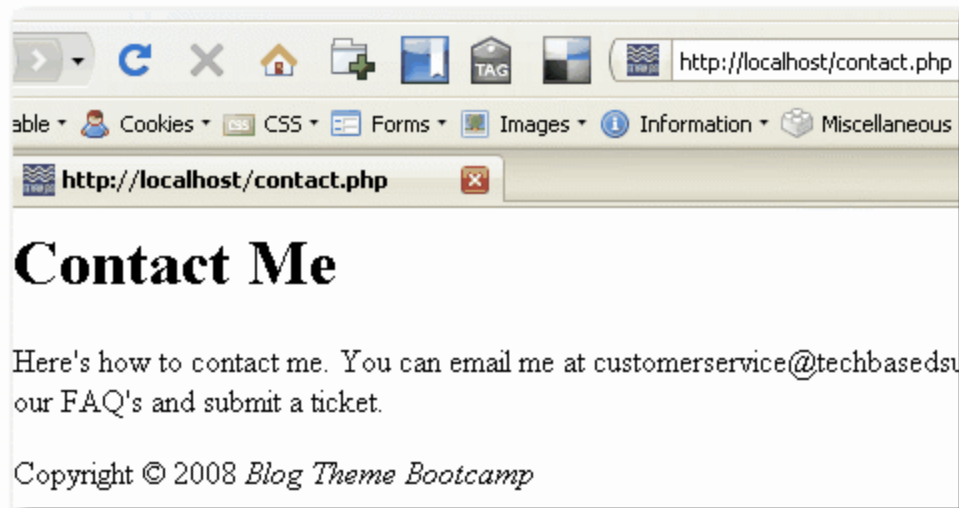
4. Check your work by refreshing helloworld.php in your browser. You should see the copyright and site name at the bottom. Nothing should have changed.

5. Next, create your contact.php page. Be sure to include the variables.php file too. Here's what I've done on mine. Note how I've also echoed the variables out at the bottom of the page.



```
1.. helloworld.php 2.. contact.php 3.. variables.php
0 10 20 30
1 <?php
2 include 'variables.php';
3 >?
4 <html>
5 <body>
6 <h1>Contact Me</h1>
7 <p>Here's how to contact me. You
8 can email me at customerservice@
9 techbasedsupport.com or visit
10 http://techbasedsupport.com to
11 review our FAQ's and submit a
12 ticket.</p>
13 <?php
14 echo $copyright;
15 echo $sitename;
16 >?
17 </body>
18 </html>
```

6. Save the file into your WWW folder then check your work in the browser. The copyright and site name should appear at the bottom of the page. This is how my contact.php page looks like.



7. From now on, whenever you want to change the copyright year or even the site name, you only need to edit in one place (variables.php) and the changes take effect on all pages.

For those of you familiar with HTML web development, you probably realize by now that Include can be used for other things like creating a header.php, sidebar.php (menu links) and footer.php. You're right on the money.

We've introduced several technical jargon and concepts in this lesson so let's keep the lesson light so there's less overwhelm. This should also give you some time to digest and experiment.

Homework!

1. Do all the exercises in this lesson.
 - (a) Familiarize yourself writing and recognizing PHP code. Experiment by echoing text to the browser.
 - (b) Write some variables then echo them out into the browser. You can copy the examples from this lesson.
 - (c) Experiment with includes.
2. You can copy and paste the code from the lesson, but I encourage you to learn to write them independently by typing them out. Doing so will familiarize yourself with writing proper code. It also helps you put on your thinking cap to develop code independently and that's important as you will need to solve problems later on working on WordPress themes. This will require you to literally develop your own code.

Coming Up Next ...

In our next lesson "Converting A Template Into PHP", we'll transform a simple HTML template into PHP using the skills we learned today. This is a crucial step in WordPress theme development as all themes start with a simple HTML template. Stay tuned.

Happy experimenting!



Lynette Chandler

Tech Based Training

P/S: Don't forget, if you need someone to walk you through this, [one-on-one sessions](#) are available.