An Introduction to PHP

Webworks – A Workshop Series in Web Design (Session 5)

Table of Contents:

1. Introduction

Since the Internet is packed with millions of acronyms, PHP is another one! PHP stands for PHP: *Hypertext PreProcessor.* Yes, the acronym is recursive and for the purposes of this class, we will just call it "PHP."

PHP is a scripting language, and is very similar to C, Java, and Perl. PHP allows the programmer to dynamically generate content, instead of statically like regular 'ol HTML. This tutorial will cover uses of PHP from simple data processing of forms to parsing a page for relevant information.

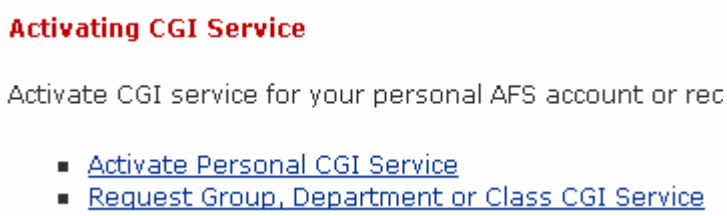
2. PHP hosts?

Free webhosting services with PHP exist, but a function like sending e-mail is probably disabled. Stanford (fortunately) provides limited PHP support sufficient enough for the scope of this course.

Stanford

Point your browser to http://cgi.stanford.edu

There's some useful documentation here, but we'll ignore them for now... scroll down and click on "Activate Personal CGI Service."



Click on "Request CGI Form." You will be prompted to type in your Stanford ID/password and keeping following the links until you see an "Activate" button.

And now, just wait for about 24 hours or so...

Other PHP hosting services?
These sites offer free PHP hosting, in exchange for massive amounts of banners, pops, and spam email. For learning purposes, these hosts are fine to test scripts with.

- Tripod - http://www.tripod.lycos.com/
- Spaceports - http://www.spaceports.com/
- T35 Hosting - http://www.t35.com/

Alternatives?
If you still want to learn PHP and cannot find a good hosting site, you can run scripts off your own computer. This is beyond the scope of this tutorial, but if you are fairly comfortable with messing with computer settings, try looking at this link on setting up PHP.

http://webmonkey.wired.com/webmonkey/00/44/index4a.html

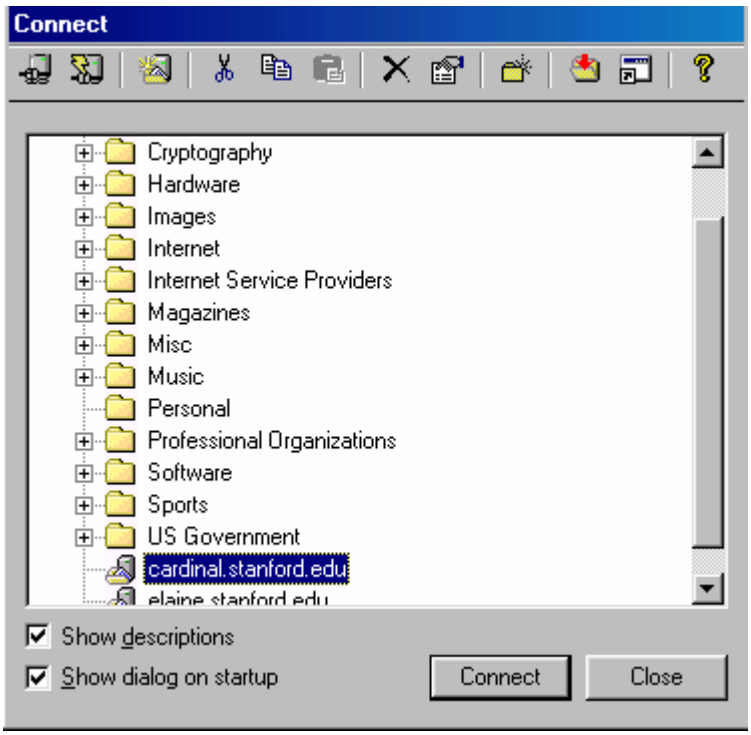
*Side note: Webmonkey is a great resources for web-related programming stuff. Take a look at it!

Uploading
This is for Stanford's webspace only!

Download Secure FX from http://www.stanford.edu/dept/itss/ess/pc/index.html

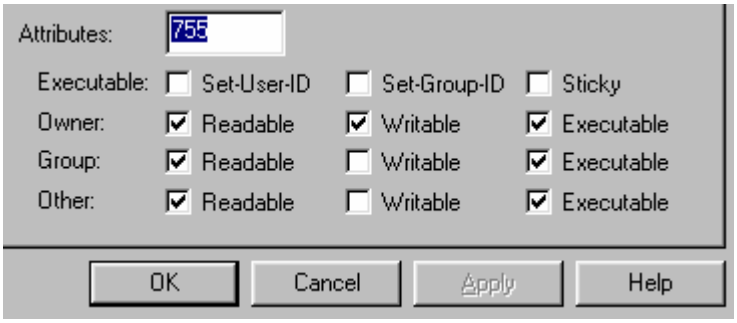
Open it up and connect to cardinal.stanford.edu.

Type in your SUNet ID, password etc.



Upload all your .php files (ASCII, not binary) in the cgi-bin by dragging and dropping the right files from your computer:

You might have to set permissions (755) on your php files before they will execute on the webspace. Right-click on the file and click on properties.



Click "ok." You should be all set!

3. Hello.php

Like in *all* programming courses, we must announce to the world our presence. Load up Notepad.exe (the best editor besides VI) and here we go:

```
#!/usr/pubsw/bin/php
```

Location to the php processor

All php scripts hosted on the Stanford server require this. Other servers may or may not require this line – always check with the server first or you might spend a long time trying to debug code that's actually working.

```
<html>
<head>
<title> Uber Hello World </title>
</head>

<body>
```

Basic HTML Setup

We're just setting up the page, in case we're going to add colors and other cool stuff.

```
<?php
print ("Hello World! HI HI HI!");
?>
```

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

First Block of PHP
Notice the "<?php" tag. This tells the PHP parser that the next block of code will contain PHP code, and the server will execute it. The end tag is "?>"

We print text with the print( ); function – Notice the double-quotes around the block of text inside. The PHP parser will print anything inside those quotes.

What if you want to print quotes? Use the escape character "\" so PHP will not use the quotes as code.

print (" \"hello! \" "); will show up as: "Hello!"

That's it for the first script!

Print Vs. Echo
We can also use "echo();" instead of "print();" What is the difference? Very fast explanation - Echo tends to be faster, but you can use Print for outputs. For the rest of this tutorial, Echo will be used instead.

4. Syntax

Just keep these in mind while coding, it might save some trouble in the long run while debugging.

File Saving
Save your text files as *.php

Comments
People tend to overlook this, but commenting is key! This will aid in debugging or understanding code after several months of not looking at it. You can use "//" to comment a line, or "/*" and "*/" to comment out several lines.

Here's an example:

```
/* Team Potato's Awesome Script
 */

#!/usr/pubsw/bin/php  // location of php parser (IMPORTANT FOR STANFORD USERS!)

<html>
<body>

// la la la this won't be displayed!!!

<?php

echo ("Hello World! HI HI HI!");
```

```
?>

</body>
</html>
```

5. If-else

If is a very important construct in many programming languages (including PHP.) *If* allows conditional fragments of code to execute.

The general structure is:

```
<?php
if(expression)
  statement
?>
```

Else is used to execute another statement when another statement is not executed because a condition is not met. *Elseif* allows another *if* statement instead an *if* block of code. *Elseif* is NOT the same as *else*, because *elseif* will take in another expression.

```
<?php
if(expression)
  statement
else
  statement
elseif(expression2)
  statement
?>
```

6. Variables

Remember variables in algebra? They're back! Since PHP is a real scripting language, you have access to variables, unlike HTML. Variables hold a value and you can do operations on them like add, subtract, concatenate…

Variables in PHP start off with a dollar sign in front, followed by the name of the variable.

Debugging tip: REMEMBER that the variable name is case-sensitive.

Assigning Values

The "=' is used to assign a value.

```
$num = 10000;
$asdf = "Hegabadado";
$hi_10 = "-10.3";
```

Notice you can assign numbers and strings in a variable. Just in case, a string "strings" characters together.

Playing with Variables
PHP supports basic mathematics!

Examples:
```
$num = $num + 1;
```

This might seem very confusing, but here is the breakdown:
1) PHP first adds $num and 1 together
2) PHP will always evaluate statements before assignments
3) PHP will use the old value of $num first

Try this code:

```
<?php

echo ("Ok I have \$num set to 5\n");
echo "<br>";

$num = 5;

echo ("Now I add 1 to $num");
echo "<br>";

$num = $num + 1;

echo "This should equal $num";

?>
```

Syntax Simplification
If you're lazy, or just want to save your wrists...

Adding 1 to a variable:
```
$answer++;
```

Subtracting 1 from a variable:
```
$answer--;
```

Adding a value (5) to a variable:
```
$answer+=5;
```

You can also subtract a value (10):
```
$answer -= 10;
```

String Manipulation
For now, the only string manipulation necessary is the "." operator to combine strings. This is also known as "concatenation."

Example:

```
<?php
```

```
$doop = "Hello ";
$blah = "Mister ";
$jump = "Tree. ";

$message = $doop.$blah.$jump; // I'm adding all of the strings from above!

echo $message;

?>
```

7. Forms Processing

Time to combine everything learned so far! One of the useful things you can do with PHP is process data from forums. For example, a simple poll, a little feedback column can all be done with PHP.

POST Vs. GET

POST and GET are not functions, they are actually variables. The Get variable will attach to the URL as: http://www.site.com/script.php?post=**input_text**
This is not as secure as Post, which will hide information being sent. Get will allow the user to modify **input_text** to change parameters of the script.

These variables are used to pass in data from a form to variables in the script.

To make things easier for forms processing, they are split up into two files. One is the html file that is supposed to take input from the user. The other file is the php file and it does the "dirty work" of processing.

For the html side, the form code has to look something like this:

```
<form action = "get.php" method="get">

....
Insert fancy drop down boxes, radio buttons here!
...

</form>
```

Action takes in the name of the php script, and **Method** takes in either "get" or "post."

Inside the script itself, use the following syntax to retrieve variables sent from the html file:

GET: $variable = $_GET["name_of_variable"];
POST: $variable = $_POST["name_of_variable"];

GET Example

```
<!-- Save this as get.html -->
#!/usr/pubsw/bin/php

<html>
```

```
<head>
<title> Get Form </title>
<body>

<form action = "get.php" method="get">
Name:<br>
<input type = "text" name = "name">

<input type="submit" value="Click!">
</form>
</body>
</html>


<!-- Save this as get.php -->
#!/usr/pubsw/bin/php

<html>
<head>
<title> Get Script </title>
<body>

<?php

$name = $_GET["name"];
echo "Look above at the address bar in your browser! <br>";
echo "Name: <b>$name</b>";

?>

</body>
</html>
```

Ok, the url in your browser should read: http://cgi.stanford.edu/~user/get.php?name=**input**
Change “input” to whatever you want, and hit enter… you’ll notice the name below changing.

There are other ways to use (or abuse) this feature, but this is just a brief look at Get.

POST EXAMPLE

```
#!/usr/pubsw/bin/php

<!-- Save this as get.html -->
<html>
<head>
<title> POST Form </title>
<body>

<form action = "post.php" method="post">
Name:<br>
<input type = "text" name = "name">
<input type="submit" value="Click!">

</form>
</body>
```

```
</html>

#!/usr/pubsw/bin/php

<!- Save this as post.php -->
<html>
<head>
<title> Post Script </title>
<body>

<?php

$name = $_POST["name"];
echo "Look above at the address bar in your browser! It's not there!<br>";
echo "Name: <b>$name</b>";

?>

</body>
</html>
```

More Form processing?
What about radio buttons, combo boxes, text fields, and the submit button? PHP can handle those as well! Remember that all of the form inputs contain a "name" field. The name field actually determines the variable name for the script.

In the examples above, the variable name was also "name." To make this less confusing, I could set:

```
<input type = "text" name = "new_variable_name!">
```

List of inputs and html code (not a comprehensive list):

Radio Box: `<input type = "radio" name = "radio1">`
Text Field `<input type = "text" name = "comment_field">`
Text Area `<textarea name="stuff" rows="10" cols="10"></textarea>`

8. Moving On

There's only so much that can be covered in one hour, but here are possible areas of PHP to look at: Advanced forms manipulation (form validation, email checking), cookies, games, data parsers, and anything the programmer (you!) wants to come up with.

For future PHP references, your best friends are:

http://php.net
http://google.com

Pretty much anything about this subject can be found on the Internet. Good luck!