# Hands on with PHP

*CS106E, Young*

In this handout, we describe PHP and walk you through how to create a PHP file on the Stanford web server. While you won't actually be writing any PHP, you will experiment with PHP files we provide. This will show you how to get things running.

## Running PHP on the Stanford Server

In contrast to HTML and CSS, which you can run right on your personal computer, to run PHP you'll have to use an actual webserver. In order to run PHP at Stanford, you'll first need to request access from Academic Computing. Go to this webpage and request a "CGI" account:

https://tools.stanford.edu/cgi-bin/cgi-request

This will create a new directory on the Leland file servers. This directory acts similar to the `WWW` directory that you can use for regular webpages, except the new directory will be called `cgi-bin`. PHP programs will not work in the regular `WWW` directory, they must be placed in the `cgi-bin` directory. Access a file in your `cgi-bin` directory using cgi.stanford.edu instead of web.stanford.edu. For example if I have a file called `example.php` in my `cgi-bin` directory, I would access it in my web browser using the URL:[1]

http://cgi.stanford.edu/~psyoung/example.php

You can place HTML and CSS files into the `cgi-bin` directory along with PHP files. These files will act normally, except that you'll need to use cgi.stanford.edu instead of web.stanford.edu in their URL.

**Don't forget you'll need to copy any PHP file to your cgi-bin directory or they will not work.** There are a variety of ways to do this. One of the simplest ways to transfer files is to use Stanford's web interface at: https://afs.stanford.edu/ We'll take a look at some alternative methods later in this handout.

## Debugging PHP

Stanford currently has PHP error messages turned off by default. If a PHP file has errors in it, it will fail to work, but no indication will be provided as to what is wrong. Instead, you'll just see a completely blank webpage. While the PHP files we provide shouldn't contain any errors, following this procedure will help us debug things if there are any problems.

I've provided a file named `php.ini` which, if placed in the same directory as your PHP files will turn the error messages back on. **Make sure to copy the `php.ini` file into each directory you are using for PHP on the server.**

[1] You can also access cgi-bin files from web.stanford.edu using a longer URL:
`http://web.stanford.edu/~psyoung/cgi-bin/example.php`

## PHP Overview

PHP is one of the most popular languages used on web servers. Websites built on PHP include Facebook, Tumblr, Wikipedia, and WordPress.

Originally, PHP stood for Personal Home Page, but it now stands for PHP: Hypertext Preprocessor. This new name really does emphasize how PHP is commonly used. Typically we will write an HTML webpage and then in certain places add some PHP code into the HTML. That PHP code we write will be pre-processed before the HTML file is sent to our website visitors.

As of this writing, the latest version of PHP is version 7. However, 85% of PHP servers are running version 5 or earlier, so use newer PHP features with caution.

## Our Simple PHP Webpage

We’ll start with a simple PHP webpage that will illustrate how PHP works. There will be more complex PHP pages on the homework. This file is included in the “h09 PHP Example.zip” file and is called “time.php”. Note that PHP files should end with the extension *.php.

We’ll begin with a pure HTML 5 starter file. We will then give the webpage the title “Time of Day” and then add the following between the <body> and </body> tags:

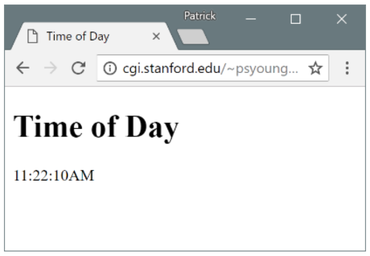
```
<h1>Time of Day</h1>

<?php
$now = new DateTime();

echo $now->format("H:i:sA");
?>
```

Here’s the full webpage:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Time of Day</title>
</head>
<body>
<h1>Time of Day</h1>

<?php
$now = new DateTime();

echo $now->format("h:i:sA");
?>

</body>
</html>
```

This webpage will display the current time of day. Note that we can’t write a static webpage to display the time of day, because the time of day changes – this is an example where an HTML webpage won’t work, we need to include programming on our webpage to determine the current time of day and insert it into our webpage.

Here's a sample of what our webpage output looks like:



Time of Day

11:22:10AM

In our PHP file the "<?php" and "?>" indicate that this particular section of our webpage is actually PHP code. Everything else in the *.php file is just standard HTML code and will be sent directly to our webpage visitors. The <?php … ?> section will be preprocessed by running the PHP code it contains on our webserver.

Let's take a closer look at our two lines of PHP:

```
$now = new DateTime();
```

This line creates a variable called $now – in PHP all variable names must begin with a dollar sign '$'. We then create a new object of the DateTime class. As with many (but not all) object-oriented languages we must explicitly use the keyword "`new`" to indicate that we are creating a new object. Because we are passing in no parameters to our DateTime constructor, the new DateTime created corresponds to the time right now.

```
echo $now->format("h:i:sA");
```

In PHP echo is used in place of print.[2] Any string listed after echo will be output to our HTML file and will appear where our original "<?php … ?>" PHP code was located in the file.

We call the `format` method on the DateTime object we just created and stored in the $now variable. This creates a new String which we will output onto our webpage. In PHP we use the arrow (formed from a dash '-' followed by a greater than '>') to access methods and variables on our objects.

The String passed into the format method indicates the exact format of the string that will be generated. The "h" indicates hours, "i" indicates minutes, "s" indicates seconds, and "A" indicates either a capitalized "AM" or "PM". Additional options include the ability to display the date, month, year, and day-of-week — see details here.

You may find it instructive to look at the page source when viewing the webpage in the web browser – while this varies from one web browser to another, typically you can get there by right-mouse clicking on a webpage in the web browser and selecting "View Source". Notice that the source looks like regular HTML with no sign of your PHP code. The PHP code never gets sent to the web browser, it only resides on the webserver itself. All PHP code in the original is replaced by whatever text is generated by the "echo" statements in the PHP code.

[2] PHP actually does have a print function that works very similarly to echo. The main difference being that echo is a "language construct" and the item output is simply listed after the keyword "echo" whereas for print the output would be enclosed in parentheses. However, as echo is much, much more widely used, I will use "echo" in our class so that the PHP code examples you see online and tutorials you read will look familiar.

## Getting PHP File on the Web

You'll need to transfer the PHP file to your "cgi-bin" directory on the Stanford file system to test it. There are a number of methods available.

1) Use the website: https://afs.stanford.edu/. This website will list all your files on the Stanford server. It will allow you to create new directories and to upload and download files from the Stanford server.

2) Use a file transfer program. You can get either Fetch for the Macintosh or SecureFX for the PC from the Essential Stanford Software website:

   http://ess.stanford.edu/

3) Use OpenAFS (also available at ess.stanford.edu) to mount your Stanford webspace as a drive on your personal computer. Using this option, you'll get an extra folder on your personal computer's desktop that is actually accessing your Stanford storage space.

Remember if you place the file "time.php" directly in your "cgi-bin" directory[3] you can either access it using:

http://cgi.stanford.edu/~*yourSUNetID*/time.php

(where you replace *yourSUNetID* with your 3-8 character long Stanford University Network ID – do make sure to include the tilde character '~' though) or using:

http://web.stanford.edu/~*yourSUNetID*/cgi-bin/time.php

If you get "Object Not Found" when you try to load your webpage that means you've either mistyped the URL or you placed the *.php file in the wrong directory.

If you get a completely blank webpage, this probably means that your PHP file has an error in it and that you did not turn error reporting on. As I mentioned on the first page of this handout, you will need to copy the "php.ini" file we provided to you into every directory on the Stanford servers that you use PHP files on, otherwise, no error messages will be displayed, instead your PHP programs will simply silently fail to run.

[3] You can create sub-directories in the "cgi-bin" directory just as you can with regular webpages in the "WWW" directory.