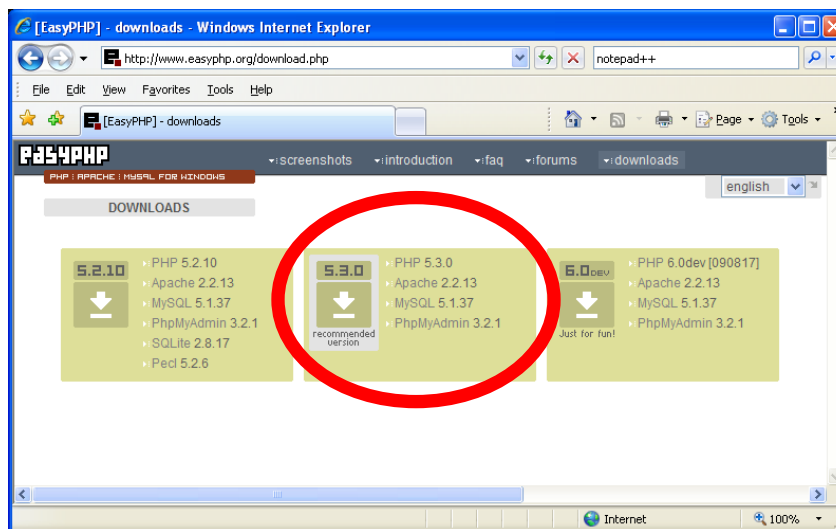
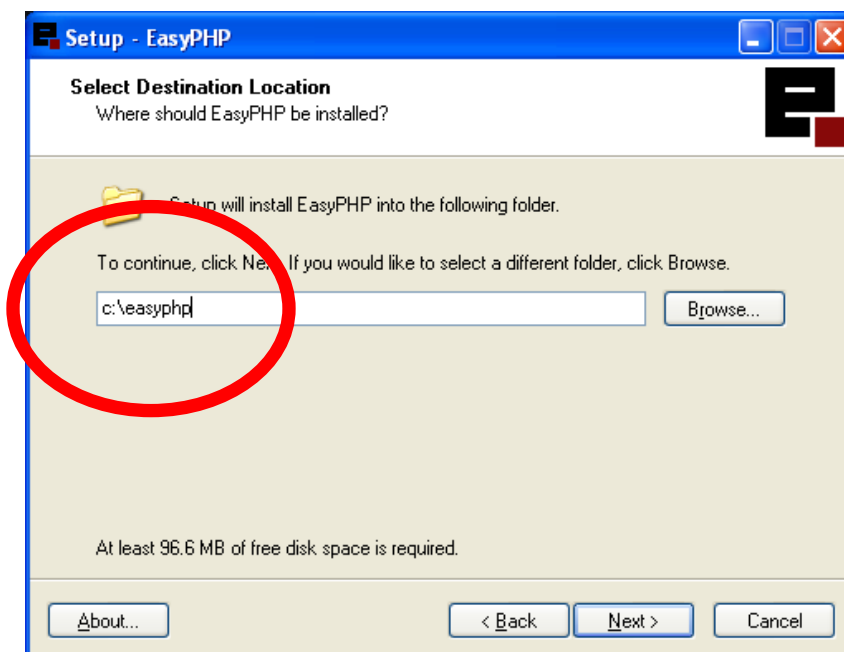


WEEK 11, LAB: Basic PHP Continued

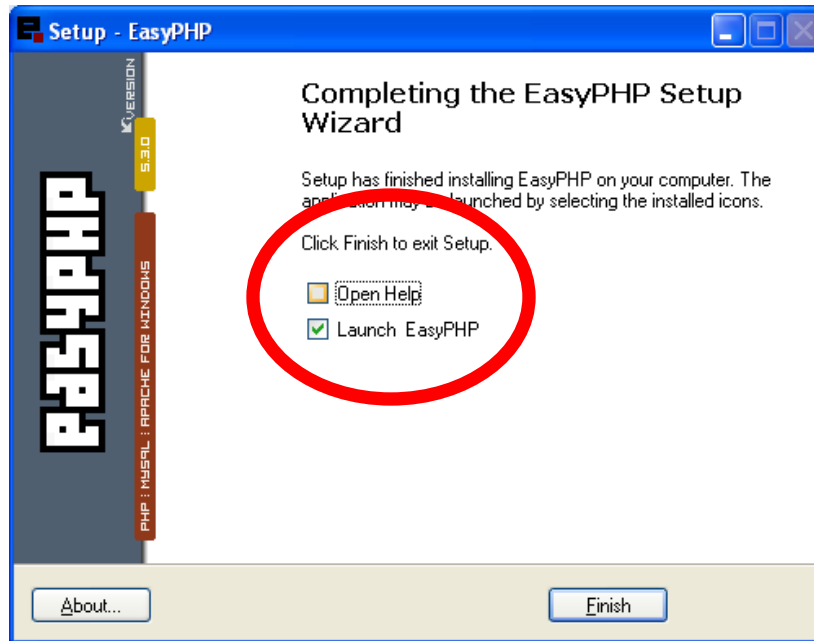
This lab is designed to expand on your familiarization with basic PHP scripting. Before starting these exercises, you will again need to set up your lab computer to emulate the PHP server environment. Your computer may already have the HTTP port in use. If this is the case, please refer to the appendix at the end of this lab. Point a browser to <http://www.easyphp.org/download.php> and download version 5.3.0. EasyPHP is an emulator – it creates a complete web server environment known as WAMP (Windows, Apache, MySQL, PHP). If you want to do this on your Mac, you can download and install MAMP or on Linux, it is likely already there.



Once you've downloaded the file, double click on it and install it. When you reach the Select Destination Location, change the folder to c:\easyphp – this will make it easier to find later.



When you reach the Completed dialog, uncheck Open Help and click Finish.

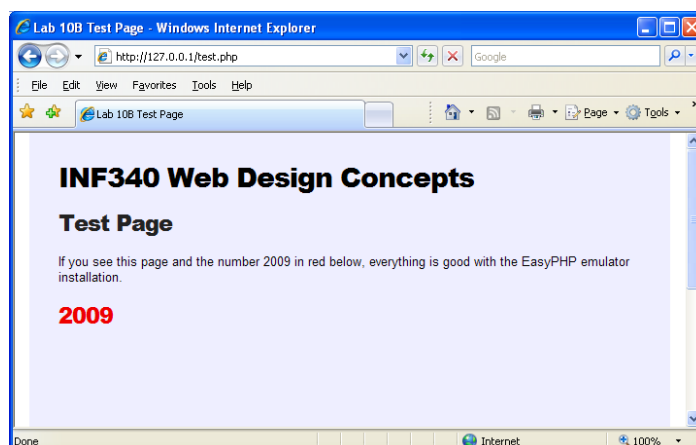


A small icon will appear in the system tray (near the clock). This is the indicator that EasyPHP is running. Do not close it. In addition, you will need an editor such as Notepad++ [[link here](#)].



You can use either MSIE or Firefox for viewing the output from the files. Since all the processing is done on the server side, there is little reason to use Firebug for this lab.

Download the lab files ZIP from matsays.com/inf340. Unzip the files and move them to `c:\easyphp\www` – this is now your web root. The web root is where all of the web files are served from. If this were the web root for my site, it would resolve in the URL as <http://www.matsays.com>. Since the computer itself does not have a DNS CNAME entry, we instead have to refer to it by IP address. Open your browser and type in <http://127.0.0.1/test.html> and if you get the following, you are on the right path.



127.0.0.1 is called the local loopback – it is an IP address that is used specifically for referring to the machine itself within the HTTP protocol.

Lab 1: Modularity

Like most programming languages, the key to writing efficient code is to create modules that work together. These modules are usually called functions. Writing compact functions for various processing removes the need to write arduous tasks over and over and make the code much easier to follow (or so we hope).

1. Open page01.php in your editor and look for the HTML comment region (as shown below in green) and enter the code below. Be sure to use the correct single or double quotes.

```
<!-- BEGIN CODING HERE -->
<?php
$mixer = 3;
for($a=0;$a<10;$a++){
    echo myCalc($a, $mixer).'\n';
}
function myCalc($value, $m){
    return (((($value+2)*$m)+9));
}
?>
<!-- END CODING HERE -->
```

This code should look familiar – it's the same one we ended with last week. If we were to track the progress of the code during processing, it would look like:

```
PSEUDO OUTPUT TRACE
-> enter for loop with a=0, mixer=3
-> -> jump to myCalc(0,3)
-> -> -> return (((0+2)*3)+9) => 15
-> -> output 15 plus an HTML line break
-> next iteration of loop with a=1, mixer=3
-> -> jump to myCalc(1,3)
-> -> -> return (((1+2)*3)+9) => 18
-> -> output 18 plus an HTML line break
...and so on
```

The net effect of the function, therefore, is to output multiples of 3 beginning with 15 (unless we start with a negative number).

2. Change the line in blue above to:

```
$t = ($value--) * $m;
return (($t+1)*($t/2)+$m);
```

What does your result set look like now? Why? Can you trace the output?

Lab 2: Manipulating Strings

At the root of most languages is the ability to manipulate strings. String manipulation simply means using methods to alter the output of a string.

1. Open page02.php in your editor, look for the HTML comment, then enter the code below. Be sure to use the correct single or double quotes.

```
<!-- BEGIN CODING HERE -->
<?php
$txt = '99 bottles of beer on the wall, take one down, pass it around...';

echo strtoupper($txt). '<br/>';
echo ucwords($txt). '<br/>';
echo strtolower($txt). '<br/>';
echo str_replace('wall', 'shelf', $txt). '<br/>';
$txt2 = str_replace('99', 'ninety-nine', $txt). '<br/>';
echo ucfirst($txt2);
echo strlen($txt). '<br/>';
echo strpos($txt, 'beer');
echo strpos($txt, 'web');
?>
<!-- END CODING HERE -->
```

There are over 50 basic string manipulation function alone in PHP...this is just a small sampling.

Lab 3: File Access

Because PHP has its origins in Unix (and more specifically in Perl), it has very powerful file access capabilities. We separate these into ASCII and binary access, since the latter is invariably more difficult.

1. Open page03.php in your editor, look for the HTML comment, then enter the code below

```
<!-- BEGIN CODING HERE -->
<?php
$file = 'file1.txt';
//FILE WRITE
//FILE READ
$handle = fopen($file, "r");
$content = fread($handle, filesize($file));
fclose($handle);
echo strtoupper($content);
?>
<!-- END CODING HERE -->
```

What did you get as a result? If you see the content of file1.txt in all upper case, you're on the right track.

2. Now obviously just READING the file contents is a bit boring. That was kind of the basis of Perl (searching and outputting). What makes the PHP system very powerful is the ease at which we are able to add on to the file.

BELOW THE **//FILE WRITE** and ABOVE the **//FILE READ** comment, add the following

```
$newcontent = "This is some text that we will add to the file.";
if (is_writable($file)) {
    if (!$handle = fopen($file, 'a')) {
        echo 'Cannot open file '.$file;
        exit;
    }
    if (fwrite($handle, $newcontent) === FALSE) {
        echo 'Cannot write to file '.$file;
        exit;
    }
    fclose($handle);
} else {
    echo 'The file '.$file.' is not writable';
}
echo '<hr/>';
```

This part looks complex but really isn't. What's been added here that you haven't seen is that we're using conditionals to test whether or not we can perform an action. If the conditional is true, it means we cannot perform the action and thus exit the program (which if the conditional is false, simply means the program continues on).

Now reload the page, re-open file.txt and see what happened to it.

3. Change the lines of code in red as shown (comment out some and add others in their place).

```
//$newcontent = "This is some text that we will add to the file.";
$newcontent = "The current date and time is ".date("F j, Y g:i a");
if (is_writable($file)) {
    //if (!$handle = fopen($file, 'a')) {
    if (!$handle = fopen($file, 'w')) {
        echo 'Cannot open file '.$file;
        exit;
    }
    if (fwrite($handle, $newcontent) === FALSE) {
        echo 'Cannot write to file '.$file;
        exit;
    }
    fclose($handle);
} else {
    echo 'The file '.$file.' is not writable';
}
echo '<hr/>';
```

Changing the "a" to a "w" changes the purpose of the opening from "append" to "write". Appending is the act of concatenating to the end of a file, while writing means to place the pointer at the beginning and then write. These two small switches change the entire focus of the page!

Lab 4: Using Files for Data

1. Open page04.php in your editor, look for the HTML comment, then enter the code below:

```
<!-- BEGIN CODING HERE -->
```

```

<?php
$file = 'file2.txt';
//FILE READ TO ARRAY
$data = file($file);
echo '<table border="1" cellpadding="2">';
foreach($data as $row){
    $rowdata = explode(',',$row);
    echo '<tr>';
    echo ' <td>'.trim($rowdata[0]).'</td>';
    echo ' <td>'.trim($rowdata[1]).'</td>';
    echo ' <td>'.trim($rowdata[2]).'</td>';
    echo ' <td>'.trim($rowdata[3]).'</td>';
    echo ' <td>'.trim($rowdata[4]).'</td>';
    echo '</tr>';
}
echo '</table>';
?>
<!-- END CODING HERE -->

```

The file() construct also reads files just like fopen()+fread() but instead of reading the contents as a string, it sees every new line (existence of a carriage return) as a set of data that gets added to an array. This array can then be output as vectored information.

Before the advent of RDBMS, it was more common to store data in flat text files. Many legacy systems, in fact, still do. Even dictionary-based security systems such as HTACCESS still use flat files. In this case, we've added a delimiter to the data in order to allow that row itself to generate into an array.

Lab 5: Mix It Up

1. Open page05.php in your editor, look for the HTML comment, then enter the code below:

```

<!-- BEGIN CODING HERE -->
<form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
<table border="1">
<tr>
    <td><strong>#</strong></td>
    <td><strong>Name</strong></td>
    <td><strong>Email</strong></td>
    <td><strong>Added</strong></td>
</tr>
<?php
$file = 'file3.txt';
if(isset($_POST['name']) && isset($_POST['email'])){
    appendFile($file);
}
readMyFile($file);
function appendFile($file){
    if(isset($_POST['name']) && isset($_POST['email'])){
        $newdata = time().'|||.mRand().'|||';
        $newdata .= trim($_POST['name']).'|||.trim($_POST['email'])."\n";
        if (is_writable($file)) {
            if (!$handle = fopen($file, 'a')) {
                echo 'Cannot open file '.$file;
            }
        }
    }
}

```

```

        exit;
    }
    if (fwrite($handle, $newdata) === FALSE) {
        echo 'Cannot write to file '.$file;
        exit;
    }
    fclose($handle);
} else {
    echo 'The file '.$file.' is not writable';
}
}
}
function mRand(){
    return rand(1,10000);
}
function readMyFile($file){
    $data = file($file);
    for($i=0;$i<sizeof($data);$i++){
        echoData($data,$i,'|||');
    }
    //return $data;
}
function echoData($array,$vector,$delim){
    $thisd = explode($delim,$array[$vector]);
    echo '<tr>';
    echo ' <td>'.$thisd[1].</td>';
    echo ' <td>'.$thisd[2].</td>';
    echo ' <td>'.$thisd[3].</td>';
    echo ' <td>'.date("m/d/y g:ia",$thisd[0]).</td>';
    echo ' <td><a href="'.$_SERVER['PHP_SELF'].'?row='.$thisd[1].'">';
    echo 'remove</a></td>'; //this part doesn't do anything - see CHALLENGE
    echo '</tr>';
}
?>
<tr>
    <td></td>
    <td><input type="text" name="name" value=""/></td>
    <td><input type="text" name="email" value=""/></td>
    <td><input type="submit" value="Add"/></td>
</tr>
</table>
</form>
<!-- END CODING HERE -->

```

On the first go around, you should see my name in the table with a second row with entry boxes and a button. Put your name into the first box and email in the second. Does it add it? Can you trace the flow of the program?

CHALLENGE

For 25 extra credit points ... I've added a link (in purple above) that initiates a page turn that should delete a row. Can you create a function that will remove a specified row from the file?

Due Tuesday, November 9 at 6:55pm if you want to try it!