

Introduction to PHP Training



with examples and
hands-on exercises

WEBUCATOR

Copyright © 2021 by Webucator. All rights reserved.

No part of this manual may be reproduced or used in any manner without written permission of the copyright owner.

Version: 5.2.12

The Author

Nat Dunn

Nat Dunn is the founder of Webucator (www.webucator.com), a company that has provided training for tens of thousands of students from thousands of organizations. Nat started the company in 2003 to combine his passion for technical training with his business expertise, and to help companies benefit from both. His previous experience was in sales, business and technical training, and management. Nat has an MBA from Harvard Business School and a BA in International Relations from Pomona College.

Follow Nat on Twitter at [@natdunn](https://twitter.com/natdunn) and Webucator at [@webucator](https://twitter.com/webucator).

Class Files

Download the class files used in this manual at
<https://static.webucator.com/media/public/materials/classfiles/PHP101-5.2.12.zip>.

Errata

Corrections to errors in the manual can be found at
<https://www.webucator.com/books/errata/>.

Table of Contents

LESSON 1. PHP Basics.....	1
Welcome to the Server-side.....	2
Google Chrome DevTools: Network Tab.....	4
How PHP Works.....	8
Comments.....	11
PHP Statements and Whitespace.....	11
PHP Functions.....	11
php.net.....	13
📄 Exercise 1: Using php.net.....	16
Variables.....	18
📄 Exercise 2: First PHP Script.....	21
Variable Scope.....	23
Single Quotes vs. Double Quotes.....	24
Concatenation.....	26
Passing Variables on the URL.....	26
📄 Exercise 3: Passing Variables via the Query String.....	29
User-defined Functions (UDFs).....	35
Introduction to the Poet Tree Club.....	42
Including Files.....	54
📄 Exercise 4: Using Header and Footer Includes.....	57
Constants.....	65
Error Reporting.....	67
📄 Exercise 5: Displaying Errors.....	70
Including a Secure Configuration File.....	73
📄 Exercise 6: Including a Configuration File.....	74
LESSON 2. PHP Conditionals.....	77
if / if - else / if - elseif - else.....	77
False Equivalents: Falsy Values.....	81
Testing for Variable Existence.....	82
📄 Exercise 7: Checking for Variable Existence.....	85
switch/case.....	87
📄 Exercise 8: Working with Conditions.....	92
Ternary Operator.....	94
📄 Exercise 9: The Ternary Operator.....	96
Null Coalescing Operator.....	98

LESSON 3. Arithmetic Operators and Loops.....	101
Arithmetic Operators.....	101
The Modulus Operator.....	102
Loops.....	103
📄 Exercise 10: Working with Loops.....	106
LESSON 4. Arrays.....	111
Indexed Arrays.....	111
📄 Exercise 11: Working with Indexed Arrays.....	115
Associative Arrays.....	116
📄 Exercise 12: Working with Associative Arrays.....	120
Superglobal Arrays.....	122
Multi-dimensional Arrays.....	125
Array Manipulation Functions.....	132
in_array() Function.....	133
📄 Exercise 13: Array Practice.....	134
LESSON 5. Working with Databases.....	143
Objects.....	143
Attributes / Properties.....	144
Behaviors / Methods.....	144
Classes vs. Objects.....	145
Connecting to a Database with PDO.....	146
Introducing the Poetree Database.....	146
phpMyAdmin.....	148
Querying Records with PHP.....	152
📄 Exercise 14: Creating a Single Poem Page.....	160
Queries Returning Multiple Rows	165
📄 Exercise 15: Creating the Poems Listings.....	168
📄 Exercise 16: Adding Pagination.....	177
📄 Exercise 17: Sorting.....	182
Anticipating Foul Play.....	185
📄 Exercise 18: Filtering.....	187
📄 Exercise 19: Adding Filtering Links to the Single Poem Page.....	195

LESSON 6. Exception Handling.....	199
Uncaught Exceptions.....	199
Throwing Your Own Exceptions.....	201
Catching Exceptions.....	203
📄 Exercise 20: Division Form.....	206
PDOExceptions.....	210
📄 Exercise 21: Logging Errors.....	212
📄 Exercise 22: The dbConnect() Function.....	217
When Queries Fail to Execute.....	219
📄 Exercise 23: Catching Errors in the PHP Poetry Website.....	222
LESSON 7. PHP and HTML Forms.....	229
HTML Forms.....	229
Form Submissions	238
Sanitizing Form Data.....	239
Validating Form Data.....	245
📄 Exercise 24: Processing Form Input.....	250
LESSON 8. Sending Email with PHP	265
mail().....	265
Setting Up PHPMailer.....	267
Mail Server.....	269
📄 Exercise 25: Including a Mail Configuration File.....	271
Sending Email with PHPMailer.....	275
PHPMailer Methods and Properties.....	277
📄 Exercise 26: Creating a Contact Form.....	279
LESSON 9. Authentication with PHP and SQL.....	289
The Registration Process.....	289
Passwords and Pass Phrases.....	290
Registration with Tokens.....	293
📄 Exercise 27: Creating a Registration Form.....	296
Sessions.....	305
Cookies.....	307
📄 Exercise 28: Logging in.....	315
Logging Out.....	328
\$_REQUEST Variables.....	328
📄 Exercise 29: Resetting the Pass Phrase.....	332

LESSON 10. LAB: Inserting, Updating, and Deleting Poems.....	341
📄 Exercise 30: Submitting a New Poem.....	342
📄 Exercise 31: Showing All User’s Poems in on the Poems Page.....	350
📄 Exercise 32: Editing an Existing Poem.....	354
📄 Exercise 33: Deleting a Poem.....	362
LESSON 11. Uploading Files.....	367
Enabling File Info Functions on Windows.....	367
Uploading Images via an HTML Form.....	368
Resizing Images.....	372
📄 Exercise 34: Uploading a Profile Picture.....	376
LESSON 12. Admin Site.....	381
📄 Exercise 35: Adding the Admin Pages.....	382
📄 Exercise 36: Creating the isAdmin() Function.....	386
📄 Exercise 37: Completing the Admin Home Page.....	387
📄 Exercise 38: Completing the Admin Poems Page.....	389
📄 Exercise 39: Approving, Editing, and Deleting Poems.....	391
📄 Exercise 40: Completing the Admin Users Page.....	393
📄 Exercise 41: Make the My Account Page Spoofable.....	395

LESSON 1

PHP Basics

Topics Covered

- Server-side programming.
- How PHP works.
- A simple PHP page.
- Variable scope.
- The difference between single and double quotes.
- Passing values from one page to another via the URL.
- Documentation on PHP functions.
- Simple PHP variables.
- User-defined functions.
- Include files.
- Error messages.

Introduction

In this lesson, you will learn how server-side programming works, how PHP works, to write a simple PHP page, about variable scope, the difference between single and double quotes, to pass values from one page to another via the URL, to look up documentation on PHP functions, to understand and work with simple PHP variables, to write user-defined functions, to include files in other files to avoid rewriting the same content, and to display error messages during development.

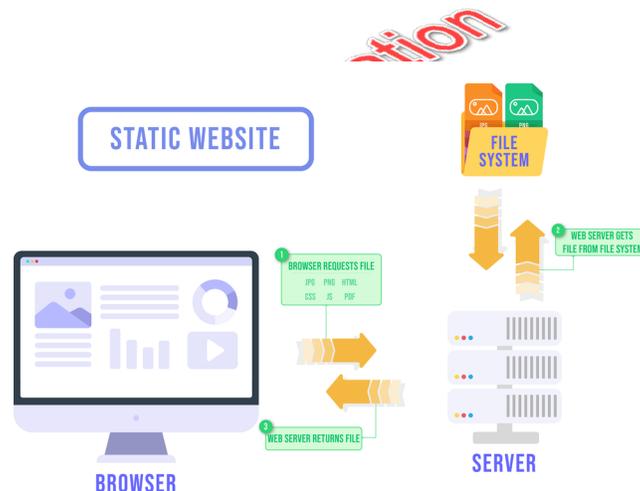


1.1. Welcome to the Server-side

As this may be your first experience programming on the server, we'll start with an introduction to how the server-side works. Client-side pages (HTML, CSS, and JavaScript) are executed by the browser, but the browser needs to get those pages somehow. Generally, they are delivered by a server. For static websites, the server will simply fetch and deliver those pages. For dynamic websites, some magic gets done on the server, which could affect the content of the pages returned.

❖ 1.1.1. What is a web server?

The first step to understanding server-side programming is to understand how a web server works. The diagram below shows how a web server delivers static pages, such as HTML, JavaScript, CSS, image files, audio and video files, PDFs, all of which browsers have a built-in way of handling; and other files that can be downloaded but not handled by the browser, such as Microsoft Word documents, zip files, and executables. All these files, both the ones the browser handles and the ones it just downloads, are *static*, meaning they are simply fetched by the web server and returned to the client without any processing on the server.



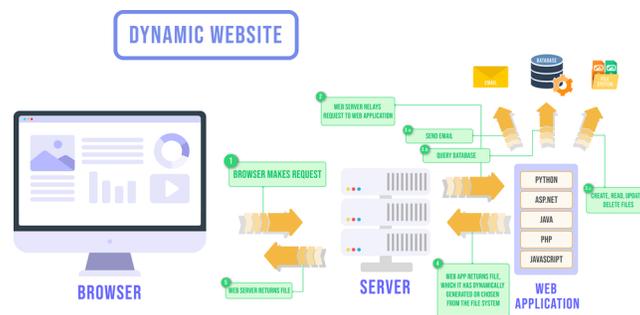
❖ 1.1.2. Dynamic Websites

Dynamic websites are websites that do more than just fetch and return files. They have software on the server that reviews the client request before deciding what to do. Depending on the client request, the server may just return a static file or it may perform any number of processes on the server before returning a dynamically created file to the client. Here are some examples of what a dynamic site might do when it receives a request:

1. Perform a database search and return a list of search results.

2. Log a user into a website by checking the database for the user's credentials.
3. Redirect the user to a login page if the user requests a members-only page.
4. Record a user's support request in a database, email the user a friendly "we-will-be-in-touch-soon" message and the auto-generated support ticket number, email the support team letting them know a new request has come in, and return a dynamically created HTML page with a friendly "we-will-be-in-touch-soon" message and the auto-generated support ticket number.

Web servers can have access to all sorts of software on the computer on which they sit and can even reach across networks to make requests of other servers, so the variety of tasks they can perform is infinite. Follow the numbers in the diagram below to see how a dynamic website works:



At the time of this writing, the most widely used web servers¹ are:

1. Apache
2. nginx
3. Microsoft-IIS

And the most widely used server-side programming languages² are:

1. PHP
2. ASP.NET
3. Java
4. Ruby
5. Python
6. Scala

1. https://w3techs.com/technologies/overview/web_server/all
 2. https://w3techs.com/technologies/overview/programming_language/all

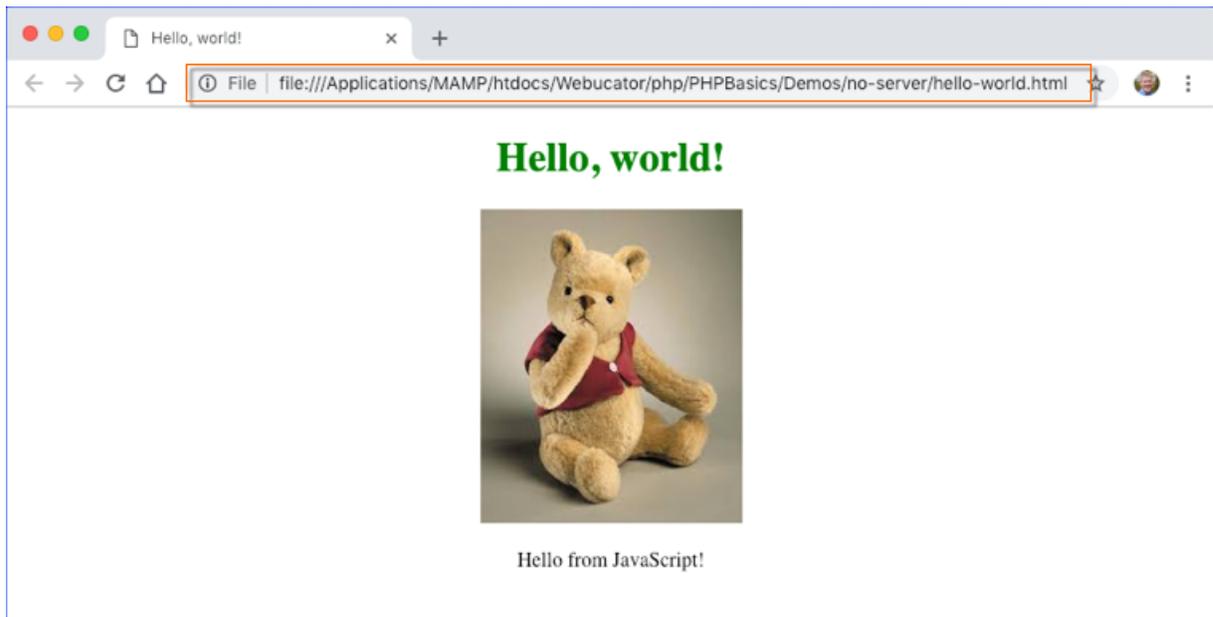
7. JavaScript³
8. ColdFusion
9. Perl



1.2. Google Chrome DevTools: Network Tab

The Google Chrome DevTools' **Network** tab shows which files are delivered when the browser makes a request. Let's first take a look at what it shows when we request a file from our local file system without going through a server:

1. Open `PhpBasics/Demos/no-server/hello-world.html` from your class files in Google Chrome:

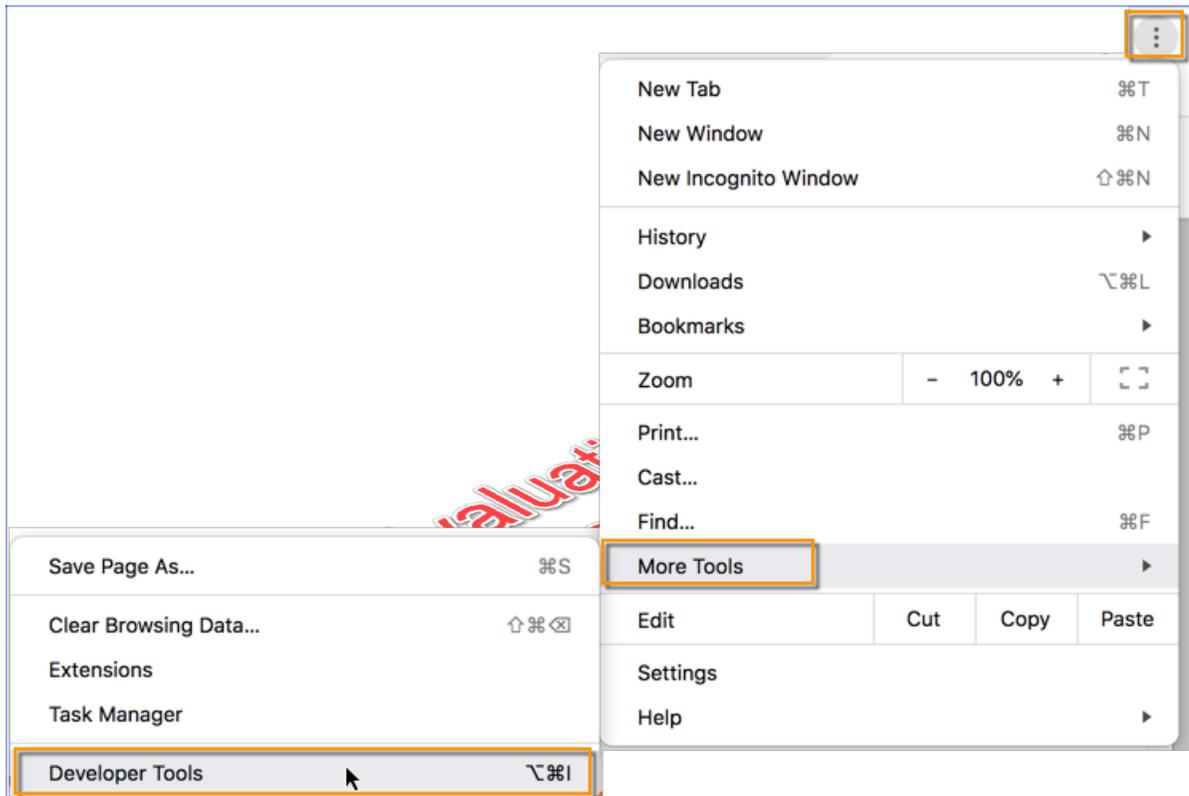


The URL in the browser's location bar should not begin with `http`. If it does, close the file and re-open it by navigating to the folder in your file system (not in Visual Studio Code) and double-clicking it.

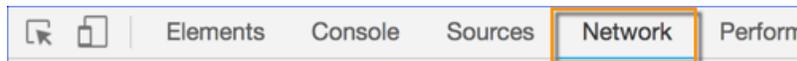
2. Open Chrome DevTools:
 - A. Click the three-vertical-dot icon in the upper right of Google Chrome.

3. While JavaScript is more well known as a client-side technology, it is now often used in server-side programming as well, most often with Node.js.

- B. Select **More Tools**.
- C. Select **Developer Tools**.



- 3. The tools will usually be docked on the right or bottom of your screen. Select the **Network** tab:



- 4. Now reload the page and look at the **Network** tab:

Name	Status	Type	Initiator
hello-world.html	Finished	document	Other
main.css	Finished	stylesheet	hello-world.html
script.js	Finished	script	hello-world.html
pooh.jpg	Finished	jpeg	hello-world.html

This shows what documents were delivered to the browser, their status, what type of documents they were, and what initiated the delivery.

- Here are the same static files delivered from a web server. We will look further at how this was delivered soon:

Name	Status	Type	Initiator
hello-world.html	304	document	Other
main.css	200	stylesheet	hello-world.html
pooh.jpg	200	jpeg	hello-world.html
script.js	200	script	hello-world.html

The only difference is the **Status** column. The web server sends back a status code to let the client know the status of the file. The 200 status code means that everything is fine. The 304 status code means that the file hasn't been modified since the last time it was requested, so the browser can use the version it has in cache if it has one.

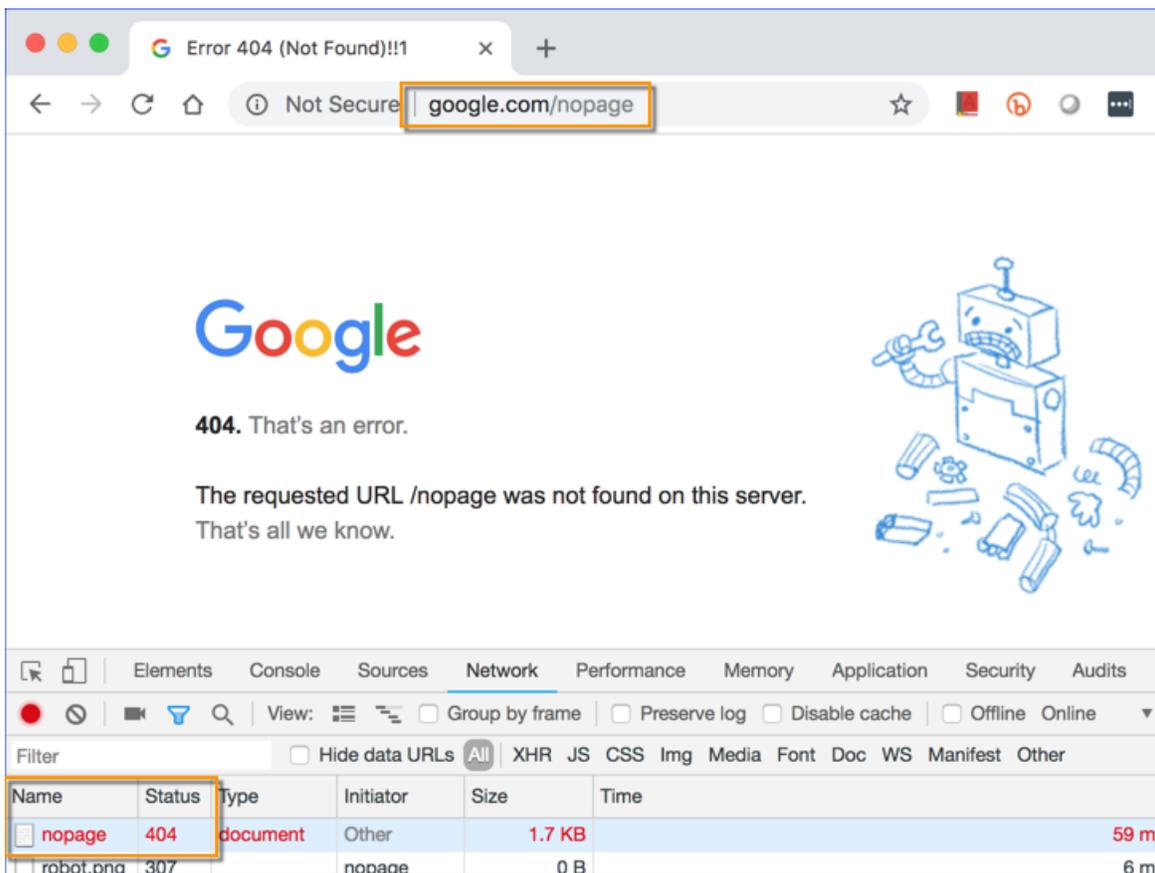
❖ 1.2.1. Status Codes

The most common status codes returned by a web server are listed below along with their meanings:

- 200 **OK**.
- 301 **Moved Permanently**. The file used to be at this URL, but it isn't anymore.
- 304 **Not Modified**. The file hasn't changed from the last time it was sent to the client.
- 400 **Bad Request**. Something about the way the request was made has baffled the web server.

- 401 **Unauthorized**. You have to be logged in to access this file.
- 403 **Forbidden**. You can't have this even if you are logged in.
- 404 **Not Found**. There is no file here.
- 500 **Internal Server Error**. Something went wrong on the server.
- 503 **Service Unavailable**. The web server is down or overloaded.

As the server-side developer, you have the ability to return these status codes and to decide what pages get returned with them. For example, it is common to return a special “404 Page Not Found” page when the user navigates to a URL on your website that doesn't exist. The screenshot below shows how Google handles this:



Notice the 404 status on the **Network** tab.



1.3. How PHP Works

When a user browses to a page that ends with a `.php` extension, the request is sent to a web server, which directs the request to the PHP interpreter. The PHP interpreter processes the page, communicating with file systems, databases, and email servers as necessary, and then delivers a web page to the web server to return to the browser.

❖ 1.3.1. The `php.ini` File

Before we look at PHP syntax, we should briefly mention the `php.ini` file. This is a plain text file that is used to configure PHP. When the PHP interpreter is started, it reads the `php.ini` file to determine what settings to use. We will mention this file from time to time throughout the course, but for now, it is enough that you are aware of its existence.

❖ 1.3.2. PHP Tags

PHP code must be contained in special tags so that the PHP interpreter can identify it. The open tag is:

```
<?php
```

And the close tag is:

```
?>
```

❖ 1.3.3. Hello, World!

It is an unwritten rule that every programming course must contain a “Hello, World!” script. Here it is:

Demo 1.1: PhpBasics/Demos/hello-world.php

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Hello, World!</title>
9. </head>
10. <body>
11. <main>
12. <?php
13.     // Write out Hello World!
14.     echo '<p>Hello, World!</p>';
15. ?>
16. </main>
17. </body>
18. </html>
```

Evaluation
Copy

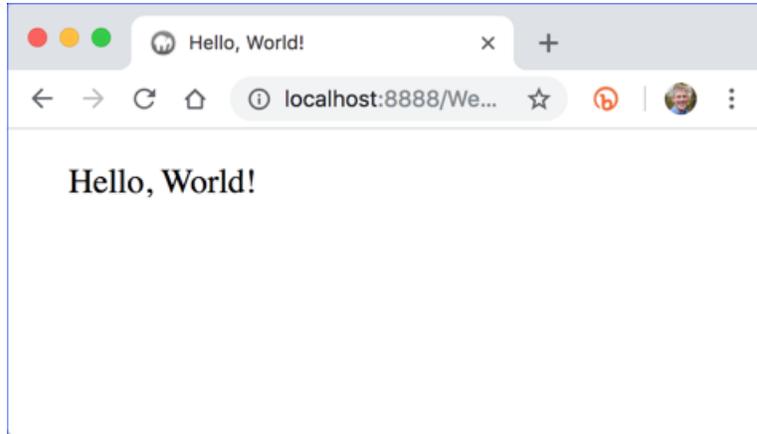
Code Explanation

Notice the following about the above code:

- Code between `<?php` and `?>` is processed by the PHP interpreter.
- The `echo` command is used to print text back to the browser.

This code isn't very exciting. In fact, PHP doesn't buy us anything here as we could have just as easily output the result using straight HTML. There is nothing dynamic about this script. After learning about variables, we'll take a look at some more interesting examples.

Open this page in your browser by navigating to <http://localhost:8888/Webucator/php/PhpBasics/Demos/hello-world.php>. You should see the following page:



Your Local Server's Address

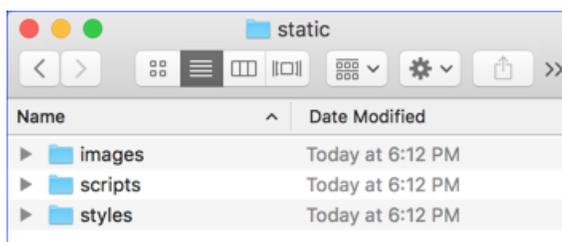
In this course, we assume that you are using port 8888 to serve your PHP pages. We also assume that your class files are located in `htdocs/Webucator/php/`.

If you're using a different port or your file structure is different from this, you will need to modify the `localhost` URLs accordingly.

Evaluation Copy

Static Files

Many of our class files will reference static image, JavaScript, and CSS files that are stored in a folder called `static` in the `php` folder:



These files are used throughout the demo site, which we will introduce soon, and the class files, but don't have any bearing on the PHP you are learning.



1.4. Comments

PHP has two forms of comments:

- Single-line comments begin with a double slash (`//`).
- Multi-line comments begin with `/*` and end with `*/`.

```
// This is a single-line comment
```

```
/*  
  This is  
  a multi-line  
  comment.  
*/
```



1.5. PHP Statements and Whitespace

PHP statements must be inside of PHP tags to be processed by the PHP interpreter. Each PHP statement must end with a semi-colon, which tells the PHP interpreter that the statement is complete. If a semi-colon does not appear at the end of a line, the interpreter will assume that the statement continues onto the next line.

The PHP interpreter condenses all sequential whitespace in PHP scripts to a single whitespace. This convenient feature allows PHP developers to structure their code in a readable format without being concerned about the effects of line breaks and tabs.



1.6. PHP Functions

There are literally hundreds of built-in PHP functions that do everything from returning the current date and time on the server to pulling data out of a database. A function might take zero arguments (e.g, `pi()`, which simply returns the value of π) or it might take multiple arguments (e.g, `trim()`, which takes one required and one optional argument). The syntax for calling a function is straightforward:

```
function_name(arguments);
```

The example below shows how to call a simple function: `phpinfo()`:

Demo 1.2: PhpBasics/Demos/phpinfo.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>PHPINFO</title>
9.  </head>
10. <body>
11. <?php
12.     //Output information on the PHP environment
13.     phpinfo();
14. ?>
15. </body>
16. </html>
```



Code Explanation

Open this page in your browser by navigating to <http://localhost:8888/Webucator/php/PhpBasics/Demos/phpinfo.php>. This will return a page similar to the following that shows how PHP is configured on the server:

PHP Version 7.2.10	
System	Darwin Nats-MBP 18.2.0 Darwin Kernel Version 18.2.0: Mon Nov 12 20:24:46 PST 2018; root:xnu-4903.231.4~2/RELEASE_X86_64 x86_64
Build Date	Oct 9 2018 14:22:23
Configure Command	<pre> ./configure '--with-apxs2=/Applications/MAMP/Library/bin/apxs' '--with-gd' '--with-jpeg-dir=/Applications/MAMP/Library' '--with-png-dir=/Applications/MAMP/Library' '--with-zlib-dir=/Applications/MAMP/Library' '--with-freetype-dir=/Applications/MAMP/Library' '--prefix=/Applications/MAMP/bin/php/php7.2.10' '--exec-prefix=/Applications/MAMP/bin/php/php7.2.10' '--sysconfdir=/Applications/MAMP/bin/php/php7.2.10/conf' '--with-config-file-path=/Applications/MAMP/bin/php/php7.2.10/conf' '--enable-ftp' '--with-bz2=/Applications/MAMP/Library' '--with-ldap' '--with-mysqli=mysqlnd' '--enable-mbstring=all' '--with-curl=/Applications/MAMP/Library' '--enable-sockets' '--enable-bcmath' '--with-imap=shared,/Applications/MAMP/Library/lib/imap-2007f' '--with-imap-ssl=/Applications/MAMP/Library' '--enable-soap' '--with-kerberos' '--enable-calendar' '--with-pgsql=shared,/Applications/MAMP/Library/pg' '--enable-exif' '--with-libxml-dir=/Applications/MAMP/Library' '--with-gettext=shared,/Applications/MAMP/Library' '--with-xsl=/Applications/MAMP/Library' '--with-pdo-mysql=mysqlnd' '--with-pdo-pgsql=shared,/Applications/MAMP/Library/pg' '--with-openssl=/Applications/MAMP/Library' '--enable-zip' '--with-pcre-dir=/Applications/MAMP/Library' '--with-libzip=/Applications/MAMP/Library' '--with-iconv=/Applications/MAMP/Library' '--enable-openssl' '--enable-intl' '--with-tidy=shared' '--with-icu-dir=/Applications/MAMP/Library' '--enable-wddx' '--with-libxpat-dir=/Applications/MAMP/Library' '--with-readline' '--with-mhash' '--with-iconv-dir=/Applications/MAMP/Library' '--with-sodium=/Applications/MAMP/Library' '--with-password-argon2=/Applications/MAMP/Library' '--disable-cgi' '--disable-phpdbg' 'YACC=/Applications/MAMP/Library/bin/bison' </pre>
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/Applications/MAMP/bin/php/php7.2.10/conf



1.7. php.net

PHP functions are well documented at <https://secure.php.net>. You can quickly look up documentation on a function by going to https://secure.php.net/function_name. For example, to see documentation on `phpinfo()`, go to <https://secure.php.net/phpinfo>. You should see something like this:

The screenshot shows the PHP documentation for the `phpinfo()` function. Annotations include:

- Function name:** `phpinfo`
- Short Description:** `phpinfo` — Outputs information about PHP's configuration
- Function Signature:** `bool phpinfo ([int $what = INFO_ALL])`
- Explanation of Parameters:** A callout box pointing to the `what` parameter.

Description: Outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

Because every system is setup differently, `phpinfo()` is commonly used to check [configuration settings](#) and for available [predefined variables](#) on a given system.

`phpinfo()` is also a valuable debugging tool as it contains all EGPCS (Environment, GET, POST, Cookie, Server) data.

Parameters:

what
The output may be customized by passing one or more of the following *constants* bitwise values summed together in the optional **what** parameter. One can also combine the respective constants or bitwise values together with the [bitwise or operator](#).

The **function signature** shows what a function will return and what parameters it must or can take. Here is how it is broken out:

```
returnType function_name (required parameters, optional parameters)
```

1. **returnType** : The type of object returned by the function. Some possible values are:
 - A. int
 - B. bool
 - C. string
 - D. array
 - E. mixed - the function can return different types of data
2. **required parameters:** Any required parameters must be listed first. The `php_info()` function doesn't have any required parameters.

3. **optional parameters:** Any optional parameters must be listed after the required parameters. In the signature, optional parameters appear in square brackets and must have default values. The `$what` parameter of `phpinfo()` is optional and has the default value of `INFO_ALL`, which is a constant (to be covered soon).

Take a look at the function signature for `date` (from <https://secure.php.net/date>):

```
string date (string $format [, int $timestamp = time() ])
```

1. The function returns a string.
2. The `$format` parameter is required and must be a string.
3. The `$timestamp` parameter is optional and defaults to the value returned by the build-in `time()` function, which you will explore in the following exercise.

Exercise 1: Using php.net

 10 to 15 minutes

Find and read through the php.net documentation on the following functions:

1. `date()`
2. `time()`
3. `mktime()`
4. `min()`
5. `max()`

**Evaluation
Copy**

Pay attention to the function signature and the parameter descriptions. You may also find it interesting to read some of the user-contributed comments. Those can sometimes be very useful.

Challenge

Create a new PHP file and experiment with the functions.

Solution

There is no real solution to this exercise. The documentation for the functions is located at:

1. `date()` - <https://secure.php.net/date>
2. `time()` - <https://secure.php.net/time>
3. `mktime()` - <https://secure.php.net/mktime>
4. `min()` - <https://secure.php.net/min>
5. `max()` - <https://secure.php.net/max>



1.8. Variables

Variables are used to hold data in memory.

❖ 1.8.1. Variable Types

The main variable types in PHP are listed in the table below:

Variable Type	Explanation
Integer	whole number
Float	floating-point number
String	string of characters
Boolean	true or false
Array	list of items
Object	instance of a class

❖ 1.8.2. Variable Names (Identifiers)⁴

Variable names and all other identifiers in PHP (e.g., function names) are case sensitive and must begin with a dollar sign followed by letters, digits, and underscores. As a

4. Variable, function and class names are all identifiers and all follow these rules, with the exception that function names are not case sensitive, though the best practice is to treat function names as if they were.

convention, variable names in PHP are written in lowerCamelCase. Here are some example variable assignments:

```
$firstName = 'Thomas';  
$birthWeight = 9.6;  
$yearOfBirth = 1743;  
$favoriteFruit = 'banana';  
$isAdmin = false;
```

❖ 1.8.3. Type Juggling and Casting

PHP variables are not assigned a type (e.g, integer) at the time they are declared. Rather, the type of a PHP variable is determined at runtime by the value the variable holds and the way in which it is used. Most of the time, this is useful, but it can cause problems as we will see later in the course. Because of these potential problems, it can be useful to explicitly change the type of a variable by *casting* it to a different type. This is done by putting the new type in parentheses before the variable to be cast, like this:

```
$a = '1'; // $a is a string  
$a = (int) $a; // $a is now an integer  
$a = (bool) $a; // $a is now a boolean
```

See `PhpBasics/Demos/gettype.php`, which uses the `gettype()` function to show the current type of `$a` after each cast. Then run the file by visiting `http://localhost:8888/Webucator/php/PhpBasics/Demos/gettype.php`.

❖ 1.8.4. Hello Variables!

Here is the “Hello, World!” script again, but this time we use a variable:

Demo 1.3: PhpBasics/Demos/hello-variables.php

```
1.  <?php
2.    $greeting = 'Hello, World!';
3.  ?>
4.  <!DOCTYPE html>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <link rel="stylesheet" href="../../static/styles/normalize.css">
10. <link rel="stylesheet" href="../../static/styles/styles.css">
11. <title><?php echo $greeting; ?></title>
12. </head>
13. <body>
14. <main>
15.   <p>
16.     <?php
17.       echo $greeting;
18.     ?>
19.   </p>
20. </main>
21. </body>
22. </html>
```



Code Explanation

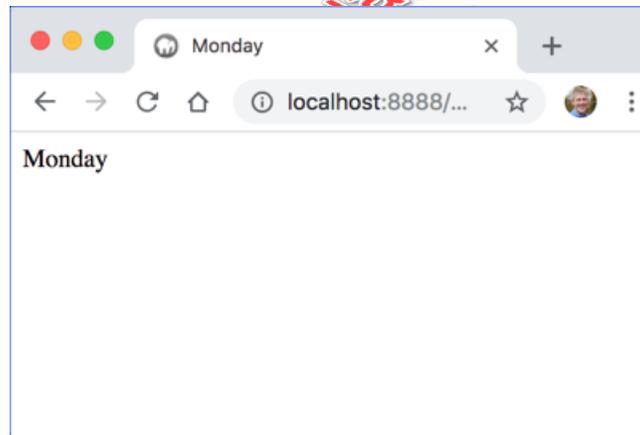
This time the string “Hello, World!” is stored in the `$greeting` variable, which is output in the `title` and `body` of the page with an `echo` command.

Exercise 2: First PHP Script

 5 to 10 minutes

In this exercise, you will write a simple PHP script from scratch. The script will declare a variable called `$today` that stores the day of the week and outputs it to the page in the `title` and the `body`.

1. Open a new document and save it as `today.php` in the `PhpBasics/Exercises` folder.
2. Declare a variable called `$today` that holds the current day of the week as literal text.
3. Output `$today` in the `title` and body of the page.
4. Navigate to `http://localhost:8888/Webucator/php/PhpBasics/Exercises/today.php` in your browser to test your solution. The resulting HTML page should look like this:



Challenge

Instead of assigning a literal string (e.g., “Monday”) to `$today`, use the built-in `date()` function so that the script won't have to be manually updated every day to stay current. For documentation, visit <https://secure.php.net/date>.

Solution: PhpBasics/Solutions/today.php

```
1.  <?php
2.    $today = 'Monday';
3.  ?>
4.  <!DOCTYPE html>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <title><?php echo $today; ?></title>
10. </head>
11. <body>
12. <?php
13.   echo $today;
14. ?>
15. </body>
16. </html>
```

Challenge Solution: PhpBasics/Solutions/today-challenge.php

```
1.  <?php
2.    $today = date('l');
3.  ?>
4.  <!DOCTYPE html>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <title><?php echo $today; ?></title>
10. </head>
11. <body>
12. <?php
13.   echo $today;
14. ?>
15. </body>
16. </html>
```



1.9. Variable Scope

A variable's scope determines the locations from which the variable can be accessed. PHP variables are either superglobal, global, or local.

Variable Scope	Explanation
superglobal	Superglobal variables, also called <i>automatic globals</i> , are predefined arrays, including <code>\$_POST</code> and <code>\$_GET</code> . They are accessible everywhere.
global	Global variables are accessible throughout the script in which they are declared and any files that are included by that script. However, they are not visible within functions in the script unless they are re-declared within the function as global variables.
local	Variables in the function scope are called local variables. Local variables are local to the function in which they are declared.

❖ 1.9.1. Superglobals

Again, superglobal variables are predefined arrays, including `$_POST` and `$_GET`, and are accessible from anywhere on the page. The list of superglobals is shown below:

- `$_GET` - variables passed into a page on the query string.
- `$_POST` - variables passed into a page through a form using the post method.
- `$_SERVER` - server environment variables (e.g, `$_SERVER['HTTP_REFERER']` returns the URL of the referring page).
- `$_COOKIE` - cookie variables.
- `$_FILES` - variables containing information about uploaded files.
- `$_ENV` - PHP environment variables (e.g, `$_ENV['HTTP_HOST']` returns the name of the host server).
- `$_REQUEST` - variables passed into a page through forms, the query string and cookies.
- `$_SESSION` - session variables.

The elements within superglobal variables are accessed using the following syntax:

```
$_GET[ 'var-name' ]
```

We will revisit superglobals throughout the course.



1.10. Single Quotes vs. Double Quotes

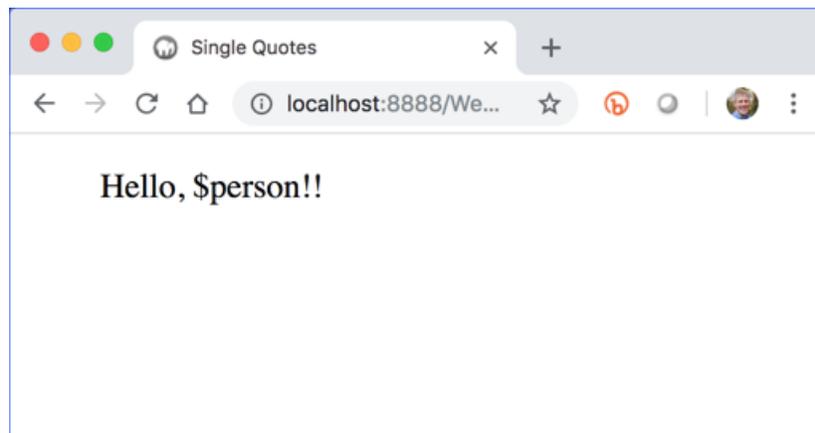
In PHP, for simple strings you can use single quotes and double quotes interchangeably. However, there is one important difference of which you need to be aware. Text within single quotes will not be parsed for variables and *escape sequences* (e.g., `\n` for a newline and `\t` for a tab). Compare the examples below:

Demo 1.4: PhpBasics/Demos/single-quotes.php

```
-----Lines 1 through 10 Omitted-----  
11. <?php  
12.     $person = 'George';  
13.     echo '<p>Hello, $person!!</p>';  
14. ?>  
-----Lines 15 through 16 Omitted-----
```

Evaluation Copy

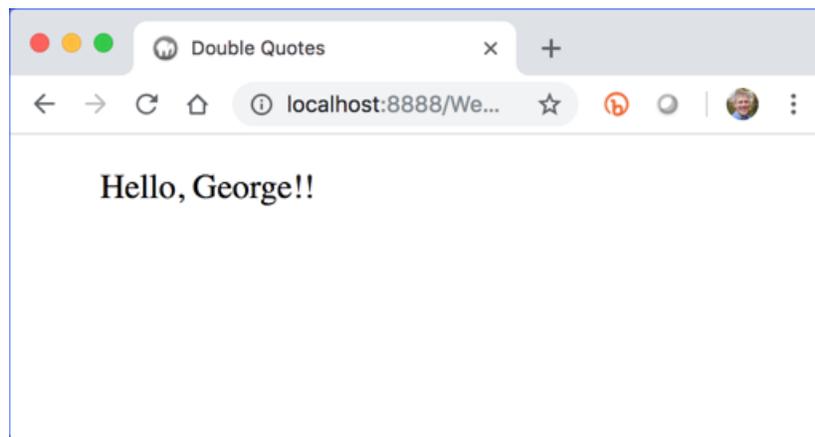
Because of the use of single quotes above, the string `<p>Hello, $person!!</p>` will be output literally, as shown below:



Demo 1.5: PhpBasics/Demos/double-quotes.php

```
-----Lines 1 through 10 Omitted-----  
11. <?php  
12.     $person = 'George';  
13.     echo "<p>Hello, $person!!</p>";  
14. ?>  
-----Lines 15 through 16 Omitted-----
```

This time, because of the double quotes, the string will be parsed for variables and special characters and will be output as shown below:



Escaping Quotation Marks

To include an apostrophe in a string denoted with single quotes, use a backslash to *escape* the apostrophe. For example:

```
$author = 'Madeleine L\'Engle';
```

Do the same thing to include a quotation mark in a string denoted with double quotes:

```
echo "Madeleine L'Engle said \"The great thing about getting older  
is that you don't lose all the other ages you've been.\"";
```



1.11. Concatenation

Concatenation is a fancy word for stringing strings of text together. In PHP, concatenation is done with the `.` operator. It is often used to combine variables with literals as in the following example:

```
$firstName = 'Nat';  
echo 'Hello, ' . $firstName;
```

Using the `.=` operator, it is possible to do one-step concatenation, in which you append a string on to the end of another string as shown below:

```
$name = 'Nat';  
$name .= ' Dunn';  
echo $name; // will echo 'Nat Dunn'
```

*Evaluation
*
Copy*

1.12. Passing Variables on the URL

A common way to pass values from the browser to the server is by appending them to the URL as follows:

```
https://www.example.com/hello.php?greet=Hello&who=World
```

The part of the URL that follows the question mark is called the query string. One or more name-value pairs can be passed to the server in this way. Each name-value pair is separated by an ampersand (&). The processing page can read these name-value pairs and use them to determine its response.

The HTML page below shows an example of how these name-value pairs might be passed.

Demo 1.6: PhpBasics/Demos/hello-hi.html

```
-----Lines 1 through 11 Omitted-----
12. <p>Do you prefer a formal or informal greeting?</p>
13. <ul>
14. <li><a href="hello-hi.php?greeting=Hello">Formal</a></li>
15. <li><a href="hello-hi.php?greeting=Hi">Informal</a></li>
16. <li><a href="hello-hi.php?greeting=Howdy">Friendly</a></li>
17. </ul>
-----Lines 18 through 20 Omitted-----
```

Demo 1.7: PhpBasics/Demos/hello-hi.php

```
1. <?php
2.     // Assign the passed variable to a variable with
3.     // a more convenient name.
4.     $greeting = $_GET['greeting'];
5.     ?>
6. <!DOCTYPE html>
7. <html lang="en">
8. <head>
9. <meta charset="UTF-8">
10. <meta name="viewport" content="width=device-width,initial-scale=1">
11. <link rel="stylesheet" href="../../static/styles/normalize.css">
12. <link rel="stylesheet" href="../../static/styles/styles.css">
13. <title><?= $greeting ?>, World!</title>
14. </head>
15. <body>
16. <main>
17. <?php
18.     echo "<p>$greeting, World!</p>";
19.     ?>
20. </main>
21. </body>
22. </html>
```

Code Explanation

Notice the following about the code above.

1. Variable names begin with a dollar sign (\$).
2. Values passed in the query string are part of the `$_GET` array and can be accessed using the following syntax: `$_GET['var-name']`.

3. We use double quotes so that the variable is evaluated and not output literally.
4. A shortcut for

```
<?php echo 'text to print'; ?>
```

is:

```
<?= 'text to print' ?>
```

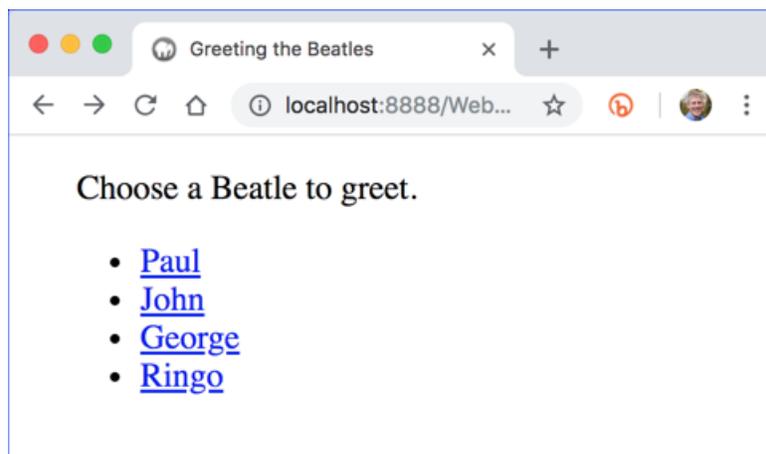
**Evaluation
Copy**

Exercise 3: Passing Variables via the Query String

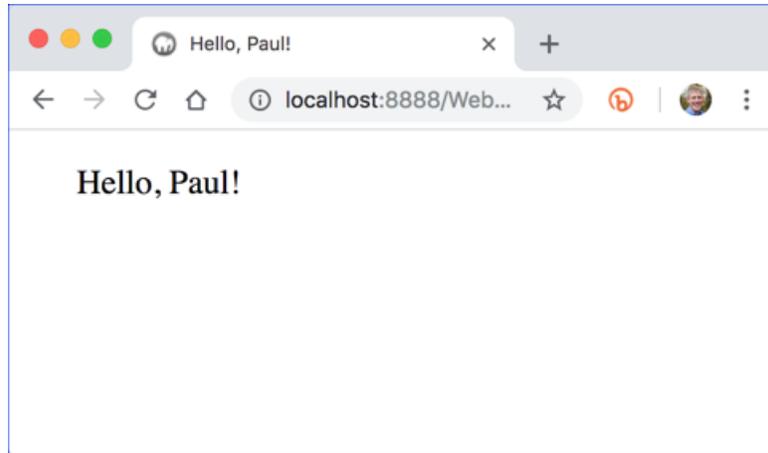
 10 to 15 minutes

In this exercise, you will write a script that says “hello” to different people based on what is passed through the query string.

1. Open `PhpBasics/Exercises/hello-who.html` in your editor. You will see that it is the same as the demo we looked at earlier.
2. Modify `hello-who.html` so that it has four links, each of which passes the name of one of the Beatles (Paul, John, George, and Ringo) to `hello-who.php`, which is in the same directory.
3. Open `PhpBasics/Exercises/hello-who.php` in your editor. Modify the code so that it outputs a greeting based on the link clicked `hello-who.html`.
4. Navigate to `http://localhost:8888/Webucator/php/PhpBasics/Exercises/hello-who.html` in your browser to test your solution. The page should look like this:



5. Clicking a link should take you to a page that looks like this:



Challenge

Change the links so that each Beatle gets a custom greeting (e.g, 'Howdy, Paul!', 'Hi, John!', 'Bye, George!', 'Hey, Ringo!').

Solution: PhpBasics/Solutions/hello-who.html

```
1. <!DOCTYPE HTML>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Greeting the Beatles</title>
9. </head>
10. <body>
11. <main>
12. <p>Choose a Beatle to greet.</p>
13. <ul>
14. <li><a href="hello-who.php?beatle=Paul">Paul</a></li>
15. <li><a href="hello-who.php?beatle=John">John</a></li>
16. <li><a href="hello-who.php?beatle=George">George</a></li>
17. <li><a href="hello-who.php?beatle=Ringo">Ringo</a></li>
18. </ul>
19. </main>
20. </body>
21. </html>
```

Solution: PhpBasics/Solutions/hello-who.php

```
1.  <?php
2.    $beatle = $_GET['beatle'];
3.  ?>
4.  <!DOCTYPE HTML>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <link rel="stylesheet" href="../../static/styles/normalize.css">
10. <link rel="stylesheet" href="../../static/styles/styles.css">
11. <title>Hello, <?= $beatle ?>!</title>
12. </head>
13. <body>
14. <main>
15. <?php
16.   echo "<p>Hello, $beatle!</p>";
17. ?>
18. </main>
19. </body>
20. </html>
```

Challenge Solution: PhpBasics/Solutions/hello-who-challenge.html

```
1.  <!DOCTYPE HTML>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>Greeting the Beatles</title>
9.  </head>
10. <body>
11. <main>
12.   <p>Choose a Beatle to greet.</p>
13.   <ul>
14.     <li>
15.       <a href="hello-who-challenge.php?beatle=Paul&greeting=Hi">
16.         Paul</a>
17.     </li>
18.     <li>
19.       <a href="hello-who-challenge.php?beatle=John&greeting=Hello">
20.         John</a>
21.     </li>
22.     <li>
23.       <a href="hello-who-challenge.php?beatle=George&greeting=Bye">
24.         George</a>
25.     </li>
26.     <li>
27.       <a href="hello-who-challenge.php?beatle=Ringo&greeting=Hey">
28.         Ringo</a>
29.     </li>
30.   </ul>
31. </main>
32. </body>
33. </html>
```

Challenge Solution: PhpBasics/Solutions/hello-who-challenge.php

```
1.  <?php
2.    $beatle = $_GET['beatle'];
3.    $greeting = $_GET['greeting'];
4.  ?>
5.  <!DOCTYPE HTML>
6.  <html lang="en">
7.  <head>
8.    <meta charset="UTF-8">
9.    <meta name="viewport" content="width=device-width,initial-scale=1">
10.  <link rel="stylesheet" href="../../static/styles/normalize.css">
11.  <link rel="stylesheet" href="../../static/styles/styles.css">
12.  <title><?= $greeting . ', ' . $beatle ?>!</title>
13. </head>
14. <body>
15. <main>
16. <?php
17.   echo "<p>$greeting, $beatle!</p>";
18. ?>
19. </main>
20. </body>
21. </html>
```

Evaluation
Copy

1.13. User-defined Functions (UDFs)

User-defined functions (UDFs) are used to make common tasks easier and to make code more modular and easier to read.

❖ 1.13.1. Defining and Calling Functions

A simple function is defined as follows:

```
function myFunction() {
    doThis();
    doThat();
    doThisOtherThing();
}
```

Like built-in functions, UDFs can receive parameters. To define a function with parameters, place receiving variables in the parentheses:

```
function addNums($param1, $param2, $param3) {  
    $sum = $param1 + $param2 + $param3;  
    echo 'The sum is ' . $sum;  
}
```

UDFs can also return values:

```
function addNums($param1, $param2, $param3) {  
    $sum = $param1 + $param2 + $param3;  
    return $sum;  
}
```

**Evaluation
Copy**

UDFs are called in the same way that built-in functions are. For example, the following code calls the `addNums()` function to get the sum of three numbers:

```
$total = addNums(1,3,5);
```

The following file shows the above code in action:

Demo 1.8: PhpBasics/Demos/simple-udf.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>Simple User-defined Function</title>
9.  </head>
10. <body>
11. <main>
12. <?php
13. function addNums($param1, $param2, $param3) {
14.     $sum = $param1 + $param2 + $param3;
15.     return $sum;
16. }
17.
18. $total = addNums(1,3,5);
19.
20. echo $total;
21. ?>
22. </main>
23. </body>
24. </html>
```



Code Explanation

Note that calling `addNums()` without passing in the correct number of values will result in a fatal error. You can fix this by providing default values.

❖ 1.13.2. Default Values

You can make function parameters optional by assigning default values to them as shown in the example below:

Demo 1.9: PhpBasics/Demos/default-values.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>User-defined Function with Default Values</title>
9.  </head>
10. <body>
11. <main>
12. <?php
13.     function addNums($param1=0, $param2=0, $param3=0) {
14.         $sum = $param1 + $param2 + $param3;
15.         return $sum;
16.     }
17.
18.     $total = addNums(1,3);
19.
20.     echo $total;
21. ?>
22. </main>
23. </body>
24. </html>
```



Code Explanation

In this case, if you don't pass a value into the function for one or more of the parameters, the default value of 0 will be used. When defining a function, all required parameters must precede optional parameters.

❖ 1.13.3. Variable Scope

In PHP, variables declared outside of functions are not available by default inside of functions. The following code illustrates this:

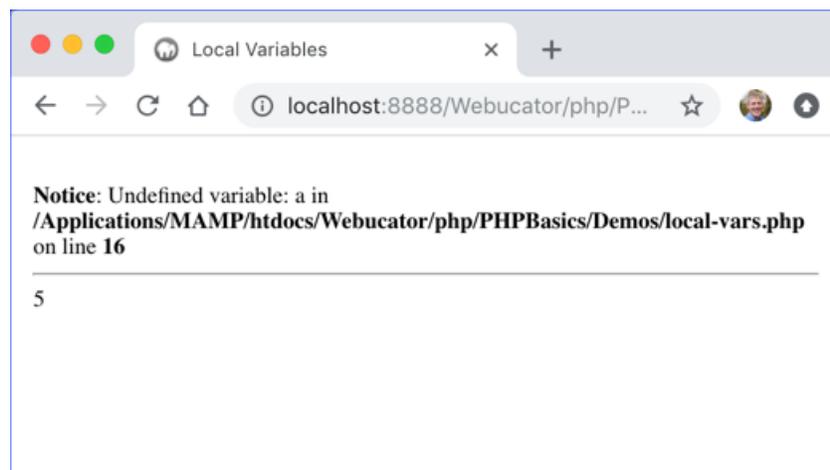
Demo 1.10: PhpBasics/Demos/local-vars.php

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Local Variables</title>
9. </head>
10. <body>
11. <main>
12. <?php
13.     $a = 10;
14.
15.     function incrNumBy($incr) {
16.         return $a + $incr;
17.     }
18.     $c = incrNumBy(5);
19.     echo '<hr>';
20.     echo $c; // outputs 5 to the browser
21. ?>
22. </main>
23. </body>
24. </html>
```



Code Explanation

As `$a` is not available within `incrNumBy()`, a warning is generated, which depending on your settings (more on this soon) may or may not be output to the browser:



Notice that the function doesn't take into account the value of `$a` and returns only the value passed in as `$incr`: 5.

The global Keyword

To make global variables available within a function, they must be declared within the function as global variables using the `global` keyword as shown below:

Demo 1.11: PhpBasics/Demos/global-vars.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>Global Variables</title>
9.  </head>
10. <body>
11. <main>
12. <?php
13.     $a = 10;
14.
15.     function incrNumBy($incr) {
16.         global $a;
17.         return $a + $incr;
18.     }
19.     $c = incrNumBy(5);
20.     echo $c; // outputs 15 to the browser
21. ?>
22. </main>
23. </body>
24. </html>
```



Code Explanation

This time the call to `incrNumBy()` will output 15 as expected.

❖ 1.13.4. By Reference vs. By Value

By default, variables are passed to functions by value, meaning that the function's receiving variables get copies of the values received rather than pointers to them. If the receiving variables are modified, the passed variables remain unaffected. The following code illustrates this:

Demo 1.12: PhpBasics/Demos/by-value.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>By Value</title>
9.  </head>
10. <body>
11. <main>
12. <?php
13.     $a = 10;
14.     $b = 5;
15.     function incrNumBy($num,$incr)  {
16.         $num += $incr;
17.     }
18.
19.     incrNumBy($a,$b);
20.     echo $a; //outputs 10 to the browser
21. ?>
22. </main>
23. </body>
24. </html>
```



Code Explanation

The above code outputs “10” to the browser. Although \$num was incremented by 5, \$a was unaffected by the function call. To pass a variable by reference, put an ampersand (&) before the parameter in the function definition as shown below:

Demo 1.13: PhpBasics/Demos/by-reference.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>By Reference</title>
9.  </head>
10. <body>
11. <main>
12. <?php
13.     $a = 10;
14.     $b = 5;
15.     function incrNumBy(&$num,$incr)  {
16.         $num += $incr;
17.     }
18.
19.     incrNumBy($a,$b);
20.     echo $a; //outputs 15 to the browser
21. ?>
22. </main>
23. </body>
24. </html>
```

Evaluation
Copy

Code Explanation

This time the function outputs “15” because `$num` references the variable `$a` itself. So, any change in `$num` will affect `$a`.



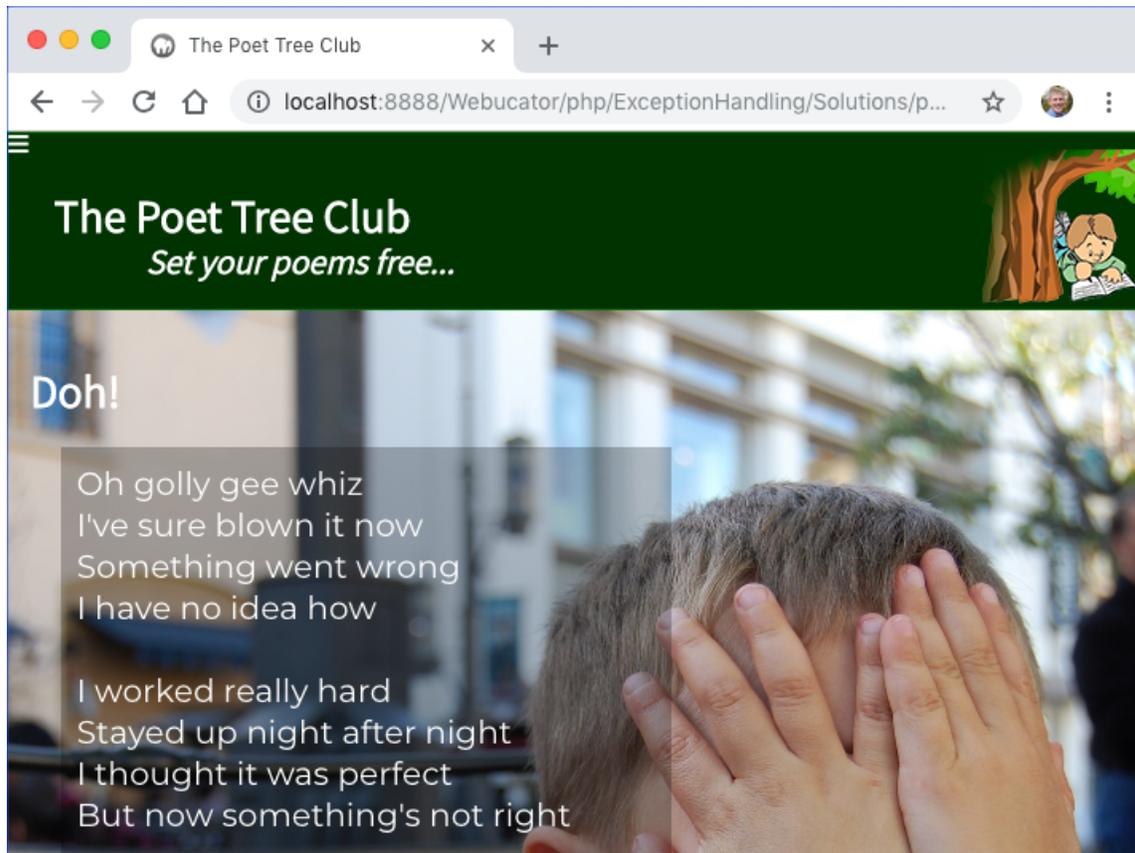
1.14. Introduction to the Poet Tree Club

Throughout this course, we will be building a complete website for reading and writing poems. Some of the features the website includes are:

1. User authentication - anyone can read poems, but you must be logged in to submit your own.
2. Poem browsing, including filtering and pagination.

3. New poem submission.
4. Contact form.
5. Image uploading.
6. Administration area for approving poems and viewing user data.

Let's take a look at the complete site, a live version of which is available at <https://www.phppoetry.com>:



1. Navigate to <https://www.phppoetry.com> in your browser:

The Poet Tree Club
Set your poems free...

Home Poems Submit Poem Log in / Register Contact us

Latest Poems

Poem	Category	Author	Published
Carrots and Camels	Funny	LimerickMan	01/11/2019
Dancing Dogs in Dungarees	Funny	LimerickMan	01/11/2019
The Geriatric General	Funny	LimerickMan	01/11/2019
More Poems			

Copyright © 2019 The Poet Tree Club. | [About us](#)

2. Click the **More Poems** link in the bottom right of the table:

Poems

Total Poems: 8

Poem	Category	Author	Published
Carrots and Camels	Funny	LimerickMan	01/11/2019
Dancing Dogs in Dungarees	Funny	LimerickMan	01/11/2019
The Geriatric General	Funny	LimerickMan	01/11/2019

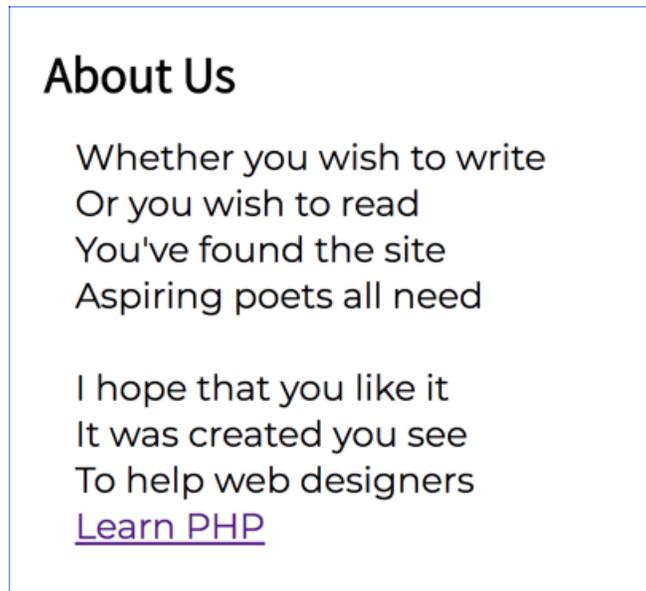
Previous [Next](#)

Filtering

Category:

Author:

- A. Note that we are currently only showing three poems per page, but you will learn how to configure this to show any number.
 - B. Click the **Next** and **Previous** links to page through the poems.
 - C. Click the table headers to sort the poems.
 - D. Filter the poems by category and author.
3. Click the **About us** link in the footer to see the **About us** page:



4. Click the **Contact us** link in the top navigation. You will see a contact form:

Contact Us

First Name*:

Last Name*:

Email*:

Message*:

Be it poetry. Be it prose.
This is where your message goes.

Send

- A. Press **Send** without filling out the form. You should see the errors shown below followed by the same form:

Contact Us

Please correct the following errors:

1. First name is required.
2. Last name is required.
3. Tomatoes and cucumbers go well in salad
But the email you entered doesn't seem valid
4. Your message must be at least 10 characters long.

- B. Fill out the form with valid data, including a valid email address that you can check. Submit the form. You should receive an email letting you know that your message has been received:

 **Webucator Classes**
to Nat ▾

Oh, happy day, we are touched
You took the time to contact us
We *will* take note
Of what you wrote
Thank you *very, very* much!

Name: Nat Dunn

Email: [nat@](#)

Your Message

I once grew a poet tree
With branches that all rhymed
And they would always know 'twas me
When up them I would climb

5. Click the **Submit Poem** link in the top navigation. You'll see that you must be logged in to submit a poem:

Login

We're happy you're here
You're welcome, my friend
Log in or register
And then try it again

Username:

Pass Phrase:

Remember me

[Forgot your pass phrase?](#)

6. Click the **Login / Register** link in the top navigation bar and then the **Register** link to register:

The screenshot shows a registration form with the following fields and content:

- Register** (Section Header)
- Already have an account? [Login](#)
- First Name:**
- Last Name:**
- Email:**
- Username:**
- Pass Phrase:**
-

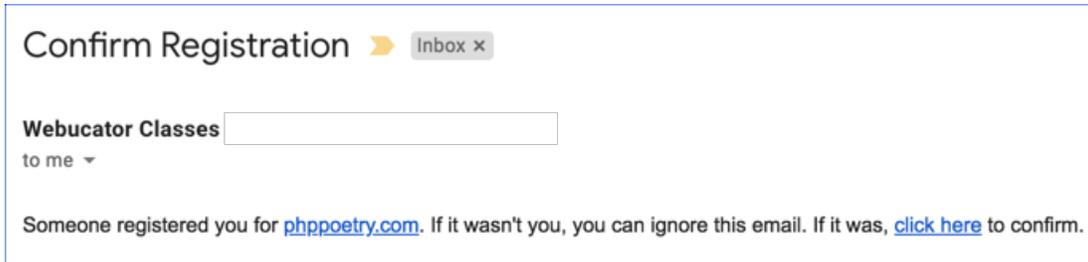
7. If you submit an invalid form, you will get errors. If you submit a valid form, you will get a success message telling you to check your email:

The screenshot shows a registration form with the following content:

- Register** (Section Header)
- We have sent you an email with instructions. Check your email.** (Success message)

Note that you will have to confirm your registration before you can log in. This is to prevent other people from registering you for the site.

8. You should receive an email similar to this one:



9. Click the [click here](#) link. That will take you to a login page:

A screenshot of a login page titled "Login". It features a green message: "You are now registered After logging in to site In addition to reading, You will be able to write". Below this are two input fields: "Username:" and "Pass Phrase:". There is a "Remember me" checkbox and a "Login" button. A link "[Forgot your pass phrase?](#)" is located at the bottom left.

Note that when you go to this same page in the future, it won't show the "You are now registered" poem. The `login.php` page knows that you have arrived by clicking the link in the email.

10. Log in with your username and password. If you check the **Remember Me** checkbox, you will not have to log in the next time you visit.
11. On a successful login, you will see the home page again:

The Poet Tree Club
Set your poems free...

Home Poems Submit Poem **My Account** Contact us

Latest Poems

Poem	Category	Author	Published
Carrots and Camels	Funny	LimerickMan	01/11/2019
Dancing Dogs in Dungarees	Funny	LimerickMan	01/11/2019
The Geriatric General	Funny	LimerickMan	01/11/2019
More Poems			

Copyright © 2019 The Poet Tree Club. | **Log out** | About us

The site knows you are logged in. Notice that the menus have changed. There is now a **My Account** link in place of the **Login / Register** link in the top navigation and there is a new **Log out** link in the footer.

- Click the **Submit Poem** link in the top navigation. Now that you are logged in, you can access this page:

Submit Poem

Title*:

Category:

- ✓ --Choose a Category
- Celebratory
- Funny
- Nature
- Other
- Religious
- Romantic
- Scary
- Serious

13. Go ahead and try your hand at some poetry and submit your poem. If there are no errors, it should take you to a page showing your new poem:

The Poet Tree

Submitted on 01/11/2019 at 12:15PM by CoolUserName [Edit](#) [Delete](#)

Pending Approval

I once grew a poet tree
With branches that all rhymed
And they would all know 'twas me
When up them I would climb

- [More Nature Poems](#)
- [More Poems by CoolUserName](#)
- [All Poems](#)

14. Because you wrote the poem, you get **Edit** and **Delete** buttons. If you click the **Delete** button, you will get a confirmation page:

Delete Poem

Are you sure you want to delete *The Poet Tree*?

Confirm Delete

Delete the poem if you like. You will get a page letting you know it was deleted.

15. Now click the **My Account** link in the top navigation:

My Account

First Name*:

Last Name*:

Email*:

Username*:

Profile Picture:

For best results, choose a 200px by 200px image.

Choose File No file chosen

Pass Phrase:

Leave blank to keep same pass phrase.

Update

16. Press the **Choose File** button below the **Profile Picture** heading and select a profile picture (a png or jpg) from your computer to upload. Then press the **Update** button.

My Account

Your account has been updated

First Name*:

Last Name*:

Email*:

Username*:

Profile Picture:


17. Log out.

Throughout this course, you will learn to build this “PHP Poetry” site. The site you build will also have an “admin” section, accessible via an **Admin** link in the footer:



Admins can edit, delete, and approve and unapprove poems. They can also update users and make other users administrators.



1.15. Including Files

Most websites, including PHP Poetry, have a header and footer that are common to many pages. Rather than rewrite the header and footer with every new page, wouldn't it be nice if you could just write them once and include them on multiple pages on the site? Enter PHP.

PHP provides two common constructs for including files in web pages: `require` and `include`. They are basically the same with one minor difference: `require` throws a fatal error when it fails; whereas, `include` only gives a warning. If the included file is necessary (i.e., *required*) to continue to process the page, you should use `require`.

It is important to keep in mind that a PHP tag cannot start in a calling file and continue in an included file. All PHP code in the included file must be nested in PHP tags.

❖ 1.15.1. `require`

`require` is not actually a function, but a language construct, so `require` statements do not need parentheses:

```
require path_to_file;
```

`path_to_file` can be an absolute or a relative path.

❖ 1.15.2. `require_once`

`require_once` can be used just like `require`. The difference is that if the included file has already been included by earlier code, it will not be re-included.

The following code samples demonstrate how to include files using `require`:

Demo 1.14: PhpBasics/Demos/includes/header.php

```
1. <?php
2.     $headerText = 'Hello!';
3.     $footerText = 'Goodbye!';
4.     echo $headerText;
5. ?>
6. <hr>
```

Demo 1.15: PhpBasics/Demos/includes/footer.php

```
1. <hr>
2. <?php
3.     echo $footerText;
4. ?>
```

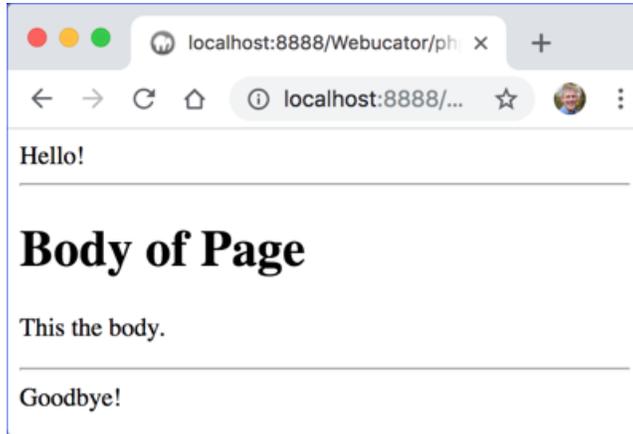
Demo 1.16: PhpBasics/Demos/include-demo.php

```
1. <?php
2.     require 'includes/header.php';
3. ?>
4. <h1>Body of Page</h1>
5. <p>This the body.</p>
6. <?php
7.     require 'includes/footer.php';
8. ?>
```

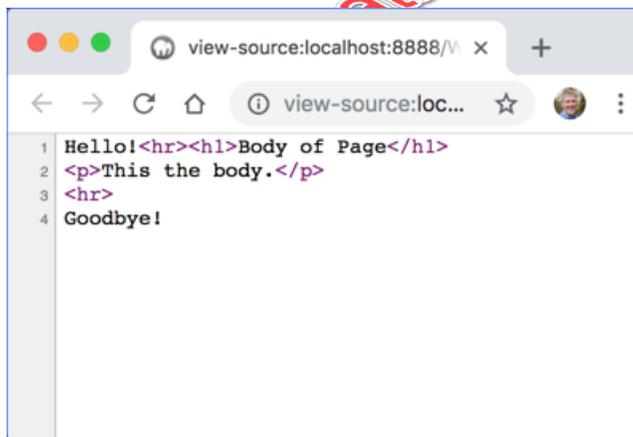
Code Explanation

The above code is relatively straightforward. `include-demo.php` contains two included (required) files: `header.php` and `footer.php`, both of which are in the `includes` folder.

Visit <http://localhost:8888/Webucator/php/PhpBasics/Demos/include-demo.php> to see the page in the browser. When the browser requests `include-demo.php`, PHP combines the two included files into the main file and the web server returns the result to the browser:



View the page source to see the resulting markup of the page, showing you exactly what PHP created:



Note that this demo is intentionally kept very simple. Normally, you would create a complete HTML page starting with the open `<html>` tag and ending with the close `</html>` tag.

Exercise 4: Using Header and Footer Includes

🕒 15 to 20 minutes

In this exercise, you will start with a full HTML page and break out the reusable parts into include files.

Navigate to <http://localhost:8888/Webucator/php/PhpBasics/Exercises/phppoetry.com> in your browser:

The Poet Tree Club
Set your poems free...

Home Poems Submit Poem My Account Contact us

Latest Poems

Poem	Category	Author	Published
Carrots and Camels	Funny	LimerickMan	01/11/2019
Dancing Dogs in Dungarees	Funny	LimerickMan	01/11/2019
The Geriatric General	Funny	LimerickMan	01/11/2019
More Poems			

Copyright © 2019 The Poet Tree Club. | Log out | Admin | About us

Clicking the **More Poems** link will bring you to the poem listing:

Poems | The Poet Tree Club

localhost:8888/Webucator/php/PHPBasics/Exercis...

Home Poems Submit Poem My Account Contact us

The Poet Tree Club

Set your poems free...

Poems

Total Poems: 8

Poem	Category	Author	Published
Carrots and Camels	Funny	LimerickMan	01/11/2019
Dancing Dogs in Dungerees	Funny	LimerickMan	01/11/2019

Previous Next

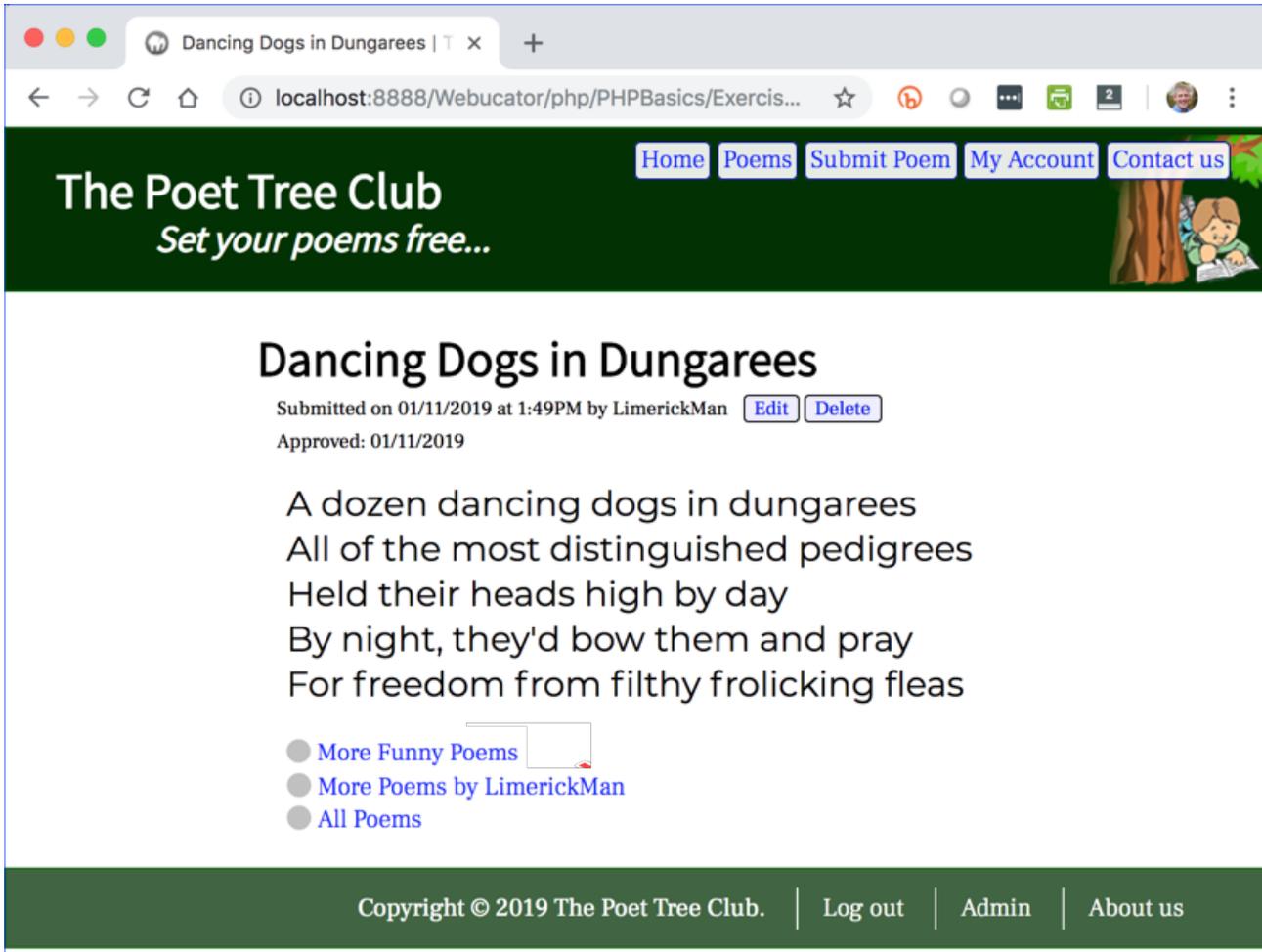
Filtering

Category:

Author:

Copyright © 2019 The Poet Tree Club. | [Log out](#) | [Admin](#) | [About us](#)

Then clicking the [Dancing Dogs in Dungerees](#) link will take you to that poem:



Notice that the top and bottom of these three pages are identical.

1. In your editor, in `PhpBasics/Exercises/phppoetry.com`, review the following three files:
 - A. `index.php`
 - B. `poems.php`
 - C. `poem.php`

Notice that, except for the titles, they begin and end with identical code.

2. Create a new folder called `includes` in the `Exercises/phppoetry.com` folder.
3. Add two files to the new `includes` folder:
 - A. `header.php`

B. footer.php

4. Using copy and paste, take code from index.php to make the header.php and footer.php includes:
 - A. For header.php, include all content from the beginning of the file to the close </header> tag.
 - B. For footer.php, include all content from the open <footer> tag to the end of the file.
5. Replace the common beginning and ending code in index.php, poems.php, and poem.php with PHP code to include the new header.php and footer.php files.
6. Navigate to <http://localhost:8888/Webucator/php/PhpBasics/Exercises/phppoetry.com> and then visit the poems and poem pages to test your solution. They should look exactly as they did before with one exception: they will all now share the same title.
7. Add PHP code to the beginning of poems.php that sets a variable named \$pageTitle to 'Poems'.
8. Add PHP code to the beginning of poem.php that sets a variable named \$pageTitle to 'Dancing Dogs in Dungarees'.
9. Modify header.php so that the text in the <title> tag starts with the \$pageTitle value followed by "| The Poet Tree Club". For example, the title for poem.php, once interpreted by PHP, should end up being:

```
<title>Dancing Dogs in Dungarees | The Poet Tree Club</title>
```


Solution: PhpBasics/Solutions/phppoetry.com/includes/header.php

```
1. <!DOCTYPE HTML>
2. <html lang="en">
3. <head>
   -----Lines 4 through 14 Omitted-----
15. <title><?= $pageTitle ?> | The Poet Tree Club</title>
16. </head>
17. <body>
18. <header>
19.     <nav id="main-nav">
20.         <!-- Bar icon for mobile menu -->
21.         <div id="mobile-menu-icon">
22.             <i class="fa fa-bars"></i>
23.         </div>
24.         <ul>
25.             <li><a href="index.php">Home</a></li>
26.             <li><a href="poems.php">Poems</a></li>
27.             <li><a href="poem-submit.php">Submit Poem</a></li>
28.             <li><a href="my-account.php">My Account</a></li>
29.             <li><a href="contact.php">Contact us</a></li>
30.         </ul>
31.     </nav>
32.     <h1>
33.         <a href="index.php">The Poet Tree Club</a>
34.     </h1>
35.     <h2>Set your poems free...</h2>
36. </header>
```

Solution: PhpBasics/Solutions/phppoetry.com/includes/footer.php

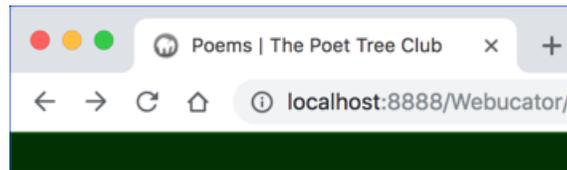
```
1. <footer>
2.     <span>Copyright &copy; 2019 The Poet Tree Club.</span>
3.     <nav>
4.         <a href="logout.php">Log out</a>
5.         <a href="admin/index.php">Admin</a>
6.         <a href="about-us.php">About us</a>
7.     </nav>
8. </footer>
9. </body>
10. </html>
```

Solution: PhpBasics/Solutions/phppoetry.com/poems.php

```
1.  <?php
2.    $pageTitle = 'Poems';
3.    require 'includes/header.php';
4.  ?>
5.  <main id="poems">
    -----Lines 6 through 61 Omitted-----
62. </main>
63. <?php
64.   require 'includes/footer.php';
65. ?>
```

Code Explanation

This page should look just as it did before. Notice the title:



Solution: PhpBasics/Solutions/phppoetry.com/poem.php

```
1.  <?php
2.    $pageTitle = 'Dancing Dogs in Dungarees';
3.    require 'includes/header.php';
4.  ?>
5.  <main id="poem">
    -----Lines 6 through 35 Omitted-----
36. </main>
37. <?php
38.   require 'includes/footer.php';
39. ?>
```

Code Explanation

This page should look just as it did before. Notice the title (you may need to hover over it to see the whole thing):

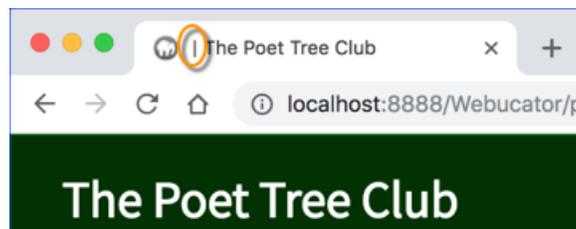


Solution: PhpBasics/Solutions/phppoetry.com/index.php

```
1.  <?php
2.      require 'includes/header.php' ;
3.  ?>
4.  <main>
    -----Lines 5 through 40 Omitted-----
41. </main>
42. <?php
43.     require 'includes/footer.php' ;
44. ?>
```

Code Explanation

This page should look just as it did before with one tiny exception. Notice the pipe at the beginning of the title:



The code in header.php that outputs the title looks like this:

```
<title><?= $pageTitle ?> | The Poet Tree Club</title>
```

But we don't declare a \$pageTitle variable in index.php, so nothing gets output. In fact, this generates a low-level error in PHP called a warning. Normally, warnings are simply ignored, but you can change this as we will soon see.



1.16. Constants

Constants are like variables except that, once assigned a value, they cannot be changed. Constants are created using the `define()` function and by convention (but not by rule) are in all uppercase letters. Constants can be accessed from anywhere on the page.

```
define('CONST_NAME', VALUE);
```

If you try to reassign a value to a constant, it will fail and trigger an `E_NOTICE`, which is a low-level error that will likely go unnoticed. For example:

```
<?php
define('BASE_YEAR', 1970);
define('BASE_YEAR', 1971); // Fails
echo BASE_YEAR; // 1970
?>
```

Evaluation
Copy

PHP includes a large number of predefined constants⁵. You won't have to worry about most of these, but there are a couple of interesting ones:

1. `PHP_VERSION` - The version of PHP that you are running.
2. `DEFAULT_INCLUDE_PATH` - The path to the directory in which PHP will look for included files after it looks in the current directory. More on this soon.

Visit <http://localhost:8888/Webucator/php/PhpBasics/Demos/predefined-constants.php> to see the value of both of these constants. The code for this page is shown below:

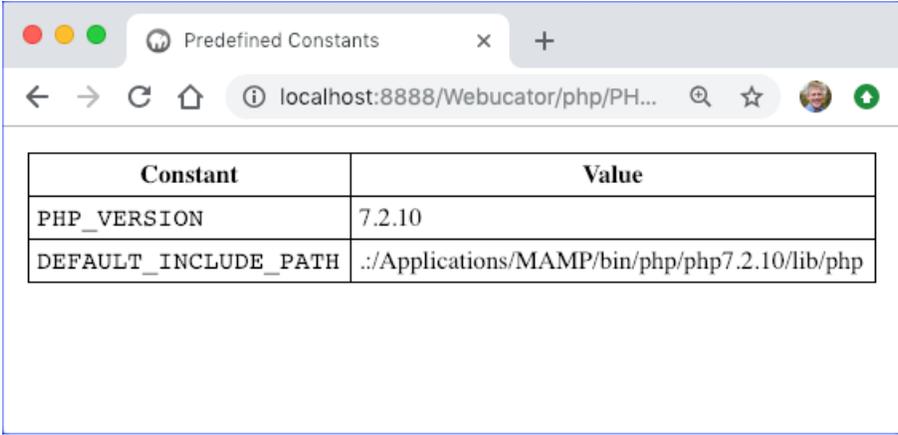
5. <https://secure.php.net/manual/en/reserved.constants.php>

Demo 1.17: PhpBasics/Demos/predefined-constants.php

```
-----Lines 1 through 10 Omitted-----
11. <main>
12.   <table>
13.     <thead>
14.       <tr>
15.         <th>Constant</th>
16.         <th>Value</th>
17.       </tr>
18.     </thead>
19.     <tbody>
20.       <tr>
21.         <td><code>PHP_VERSION</code></td>
22.         <td><?= PHP_VERSION ?></td>
23.       </tr>
24.       <tr>
25.         <td><code>DEFAULT_INCLUDE_PATH</code></td>
26.         <td><?= DEFAULT_INCLUDE_PATH ?></td>
27.       </tr>
28.     </tbody>
29.   </table>
30. </main>
-----Lines 31 through 32 Omitted-----
```

Code Explanation

When you run this file in your browser, you should see something like this:



The screenshot shows a web browser window with the title "Predefined Constants". The address bar shows the URL "localhost:8888/Webucator/php/PH...". The main content of the browser is a table with two columns: "Constant" and "Value". The table contains two rows of data.

Constant	Value
PHP_VERSION	7.2.10
DEFAULT_INCLUDE_PATH	./Applications/MAMP/bin/php/php7.2.10/lib/php



1.17. Error Reporting

Many of the predefined constants in PHP are *error reporting constants*⁶. They are used to set the types of errors that you would like to have reported. All of the error reporting constants are prefixed with `E_`. For example:

1. `E_ERROR`: Report fatal run-time errors - errors that will stop execution of the script.
2. `E_WARNING`: Report run-time warnings - errors that will not stop the execution of the script.
3. `E_DEPRECATED`: Report notices about deprecated code- code that may not work in future versions of PHP.
4. `E_ALL`: Report all errors.

The level of error reporting is set with the `error_reporting` directive in the `php.ini` file and defaults to `E_ALL`. You should most likely leave that default in place.

Two other error-reporting directives in the `php.ini` file are `display_errors` and `log_errors`. On a production server, you should set `display_errors` to `Off` and `log_errors` to `On`. They are almost certainly already set that way, but you can confirm this in the following two ways:

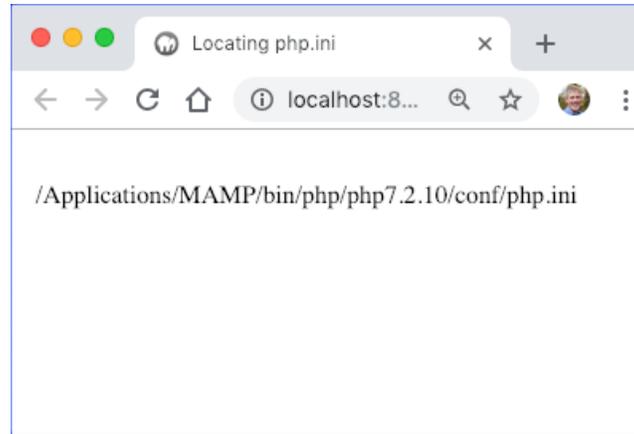
1. **Use `phpinfo()`**. Navigate to `http://localhost:8888/Webucator/php/PhpBasics/Demos/phpinfo.php` and search for the names of the directives:

<code>display_errors</code>	Off
<code>error_reporting</code>	32767
<code>log_errors</code>	On

Remember that this file simply calls the `phpinfo()` function.

2. **Look in the `php.ini` file:**
 - A. Navigate to `http://localhost:8888/Webucator/php/PhpBasics/Demos/find-php.ini.php`. The script uses the `php_ini_loaded_file()` function to let you know where the `php.ini` file is located. You should see something like this:

6. <https://secure.php.net/manual/en/errorfunc.constants.php>



B. Open that file in a text editor and search for the names of the directives:

```
454 : http://php.net/error-reporting
455 error_reporting = E_ALL
456
457 ; This directive controls whether or not and where PHP will output errors,
458 ; notices and warnings too. Error output is very useful during development, but
459 ; it could be very dangerous in production environments. Depending on the code
460 ; which is triggering the error, sensitive information could potentially leak
461 ; out of your application such as database usernames and passwords or worse.
462 ; For production environments, we recommend logging errors rather than
463 ; sending them to STDOUT.
464 ; Possible Values:
465 ; - Off = Do not display any errors
466 ; - stderr = Display errors to STDERR (affects only CGI/CLI binaries!)
467 ; - On or stdout = Display errors to STDOUT
468 ; Default Value: On
469 ; Development Value: On
470 ; Production Value: Off
471 : http://php.net/display-errors
472 display_errors = Off
473
-----
485 ; Besides displaying errors, PHP can also log errors to locations such as a
486 ; server-specific log, STDERR, or a location specified by the error_log
487 ; directive found below. While errors should not be displayed on productions
488 ; servers they should still be monitored and logging is a great way to do that.
489 ; Default Value: Off
490 ; Development Value: On
491 ; Production Value: On
492 : http://php.net/log-errors
493 log_errors = On
494
```

Boolean Values in php.ini

Within the `php.ini` file, you can turn boolean directives on using `1`, `On`, `True` or `Yes`.

You can turn them off using `0`, `Off`, `False` or `No`.

As already mentioned, on a production server, you will likely leave these settings unchanged, but you can change them on your development server if you like. Alternatively, you can change them using the `ini_set()` function. The most likely setting you will want to change during development is `display_errors`, which controls whether errors are output to the browser. During production, you definitely don't want that to happen, but during development, you probably do. To do so, at the beginning of every script or in the included `header.php` file, add the following code:

```
ini_set('display_errors', '1');
```

Exercise 5: Displaying Errors

 5 to 10 minutes

In this exercise, you will turn on the `display_errors` setting whenever the site is not running on the production server.

1. In `PhpBasics/Exercises/phppoetry.com/includes`, create a new file called `utilities.php`. You will keep useful user-defined functions in this file.
2. Add the following code to `utilities.php`:

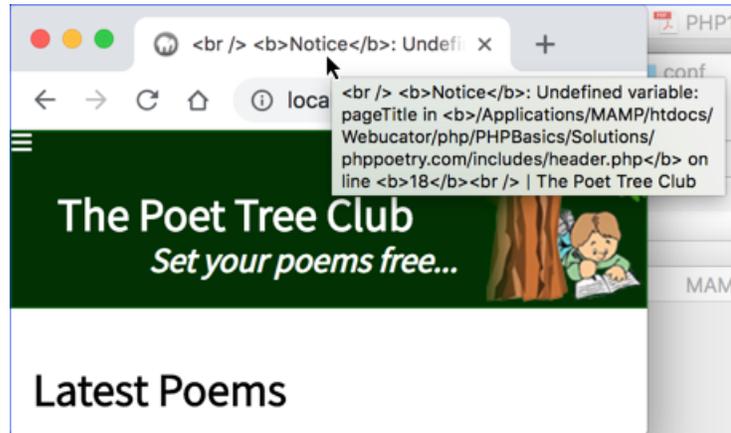
```
function isProduction() {  
    // Provide way of knowing if the code is on production server  
    return false;  
}  
  
function isDebugMode() {  
    // Set to true if not on production  
    // You may want to provide other ways for setting debug mode  
    return !isProduction();  
}
```

- The `isProduction()` function will provide a mechanism for the code to know if it is running on a development or production server.
 - You will use the `isDebugMode()` function now. Note that eventually you might have a more nuanced way of setting debug mode. For example, you might give superadmin users a way of turning on debug mode in their settings in an admin section of the website.
3. Open `PhpBasics/Exercises/phppoetry.com/includes/header.php` in your editor.
 4. Add code to include `utilities.php` **once**. Note that it's possible we will want to include the `utilities.php` file on pages that may or may not include the `header.php`. On such pages, we want to make sure that we don't include the file twice.
 5. Below the code that includes `utilities.php`, write the following:

```
if (isDebugMode()) {  
    ini_set('display_errors', '1');  
}
```

This code uses `ini_set()` to turn on the `display_errors` setting, but only when in debug mode.

6. Navigate to `http://localhost:8888/Webucator/php/PhpBasics/Exercises/phppoetry.com/index.php` in your browser.
7. As the error takes place in the `<title>` tag, the error message shows up on the browser tab. Hover over it to see the full error message:



The beginning of the error reads:

```
Notice: Undefined variable: pageTitle
```

Solution: PhpBasics/Solutions/phppoetry.com/includes/utilities.php

```
1.  <?php
2.      function isProduction() {
3.          // Provide way of knowing if the code is on production server
4.          return false;
5.      }
6.
7.      function isDebugMode() {
8.          // You may want to provide other ways for setting debug mode
9.          return !isProduction();
10.     }
11.  ?>
```

Solution: PhpBasics/Solutions/phppoetry.com/includes/header-2.php

```
1.  <?php
2.      require_once 'utilities.php';
3.      if (isDebugMode()) {
4.          ini_set('display_errors', '1');
5.      }
6.  ?>
7.  <!DOCTYPE HTML>
8.  <html lang="en">
9.  <head>
10.     -----Lines 10 through 20 Omitted-----
21.  <title><?= $pageTitle ?> | The Poet Tree Club</title>
22.  </head>
    -----Lines 23 through 42 Omitted-----
```



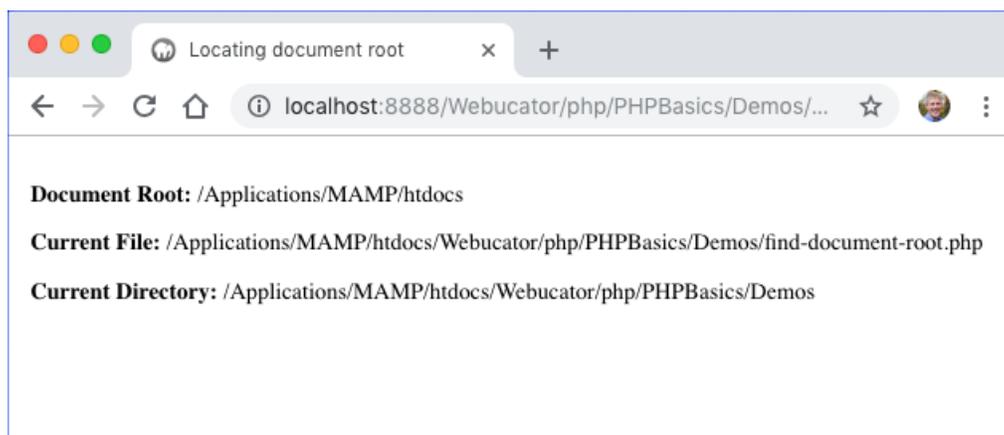
Code Explanation

Setting `display_errors` to 1 during development allows us to catch these errors, so that we can improve our code. In the next lesson, we will learn how to fix this error, so that on our home page, we do not try to prepend anything to the site name (i.e., “The Poet Tree Club”).



1.18. Including a Secure Configuration File

Any file under the web root can potentially be accessed over the web. Navigate to `http://localhost:8888/Webucator/php/PhpBasics/Demos/find-document-root.php` to find the path to your web root. That will output the path to the root and the paths to the current files and directory:



If included files are under the web root, they can be accessed just as any other file can. If they have an extension such as `inc` then the browser may display them as plain text. With other extensions, the browser may attempt to download the file. If the included file is a PHP file and a user navigates to it, the server will try to process the file and may return errors. It is also possible that your web server becomes misconfigured and returns a PHP file as plain text showing your sensitive data in the browser. As a precaution, you should place any included files with sensitive data in a directory outside of the web root. This will prevent users from accessing the files directly.

In order to be able to include files that are outside of the web root, you will need to modify the `include_path` directive in the `php.ini` file. The `include_path` directive takes a list of paths to directories in which PHP should look for included files. In the following exercise, you will change your setup to be able to include a configuration file outside of your web root.

Exercise 6: Including a Configuration File

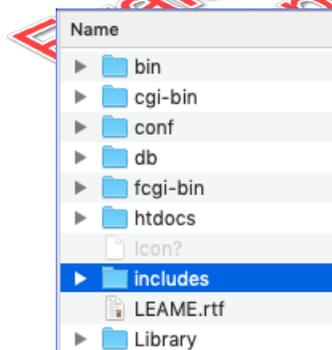
 10 to 15 minutes

In this exercise, you will:

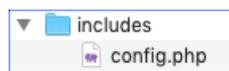
1. Create an `includes` directory outside of the web root and place a configuration file in that directory.
2. Modify the `include_path` directive in the `php.ini` file.
3. Run a test to make sure you can include the configuration file in the new `includes` directory.

Follow these instructions:

1. In Windows Explorer or Finder, navigate to the folder that contains your document root (as shown in the last demo). For example, if your document root is `/Applications/MAMP/htdocs`, navigate to `/Applications/MAMP`. Add a new folder called `includes` next to your root folder (e.g., `htdocs`):



2. In `PhpBasics/Exercises`, you will find a file called `config-sample.php`. Copy and paste that file into your new `includes` folder and then rename it `config.php`:



3. Open the `php.ini` file in your editor. Remember that you can find out where it is by navigating to `http://localhost:8888/Webucator/php/PhpBasics/Demos/find-php.ini.php`.

- In the `php.ini` file, find the `include_path` directive and append a colon (Mac) or semi-colon (Windows) followed by the path to the new `includes` folder to the end.
 - On a Mac, the path will look something like this:

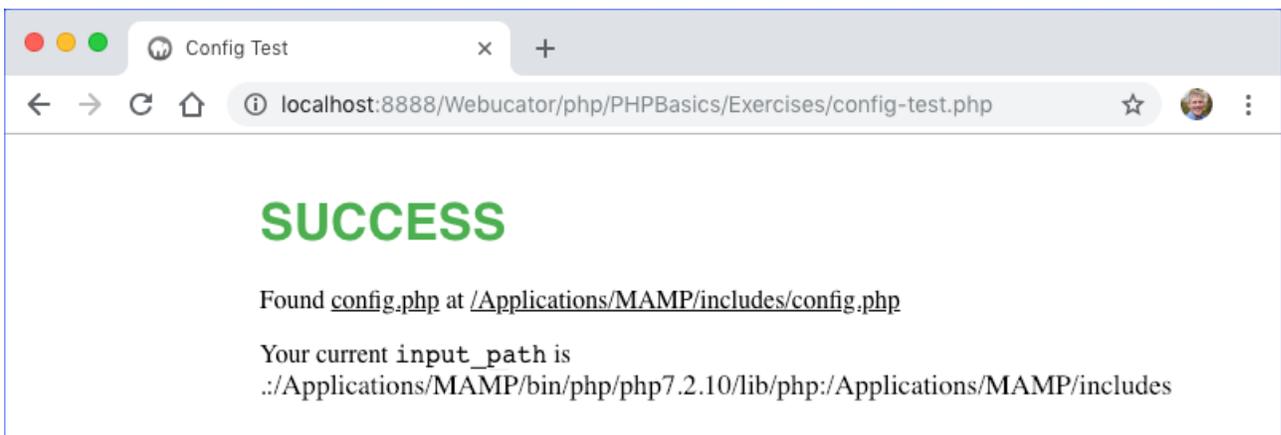
```
; UNIX: "/path1:/path2"  
include_path = "./Applications/MAMP/bin/php/php7.2.10/lib/php:/Applica  
tions/MAMP/includes"
```

- On Windows, the path will look something like this:

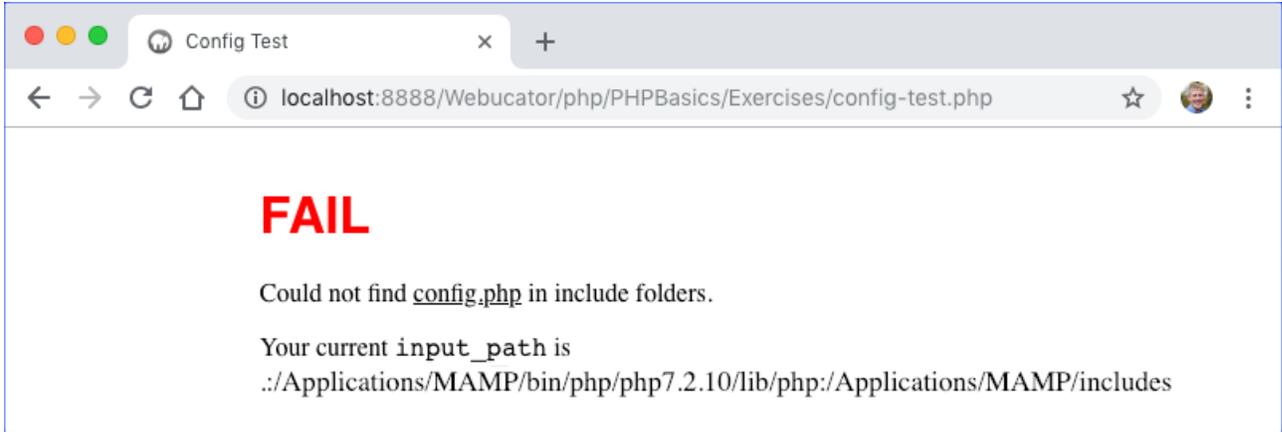
```
; Windows: "\path1;\path2"  
include_path = ".;c:\php\includes;c:\MAMP\includes"
```

Note that semi-colons at the beginning of lines in the `php.ini` file denote comments. If there is a semi-colon before the `include_path` line, you should remove it.

- Run `http://localhost:8888/Webucator/php/PhpBasics/Exercises/config-test.php` in the browser to test your setup. If successful, it will output:



If it fails, it will output:



6. There is no solution file for this exercise as it relates to your computer setup. It's important for future lessons that you have this working correctly, so make sure it does before moving on.

Conclusion

In this lesson, you have learned:

1. How server-side programming generally, and PHP specifically, work.
2. To write a simple PHP page.
3. About variable scope.
4. The difference between single and double quotes.
5. To pass values from one page to another via the URL.
6. To recognize and look up documentation on PHP functions.
7. To write your own user-defined functions.
8. To understand and work with simple PHP variables.
9. To include files in other files to avoid rewriting the same content.
10. To display error messages during development.
11. To include files outside of the web root.

LESSON 2

PHP Conditionals

Topics Covered

- ☑ if - elseif - else conditions.
- ☑ switch / case statements.
- ☑ The ternary operator.
- ☑ The null coalescing operator.

Introduction

Conditional processing allows programmers to execute different code based on specific conditions. There are two conditional structures in PHP - if - elseif - else and switch / case. There is also a special operator called the ternary operator for short conditional expressions.

Evaluating Copy

*

2.1. if / if - else / if - elseif - else

Conditional statements are used to determine whether or not to run a block of code or to decide which block of code to execute.

❖ 2.1.1. Simple if Condition

The syntax for a simple if condition is as follows:

```
if (expression) {  
    doThis();  
    doThat();  
    doThisOtherThing();  
}
```

The lines of code affected by the `if` condition are put in a code block, which is surrounded by curly brackets to indicate that all of the code either should or should not be executed depending on the true-false value of the expression.

❖ 2.1.2. if-else Condition

The syntax for an `if-else` condition is as follows:

```
if (expression) {  
    doThis();  
} else {  
    doThat();  
}
```

In the code above, either `doThis();` or `doThat();` will run depending on the true-false value of the expression.

❖ 2.1.3. if-elseif-else statement

The syntax for an `if - elseif - else` condition is as follows:

```
if (expression) {  
    doThis();  
} elseif (otherExpression) {  
    doThat();  
} else {  
    doThisOtherThing();  
}
```

An `if - elseif - else` condition can have any number of `elseif` blocks, but only zero or one `else` blocks.

The following table shows PHP's comparison operators:

Comparison Operators

Operator	Description
==	Equals
===	Identical (same value and same type)
!=	Doesn't equal
!==	Not identical
>	Is greater than
<	Is less than
>=	Is greater than or equal to
<=	Is less than or equal to

Note the difference between == (equals) and === (identical). For two objects to be **identical**, they must be of the same value **and** the same type, whereas to be **equal**, they must only have the same value. For example, 1 is *equal* to, but not *identical* to, "1".

It is almost always better to use the **identical** operator (===) and the corresponding **not identical** operator (!==) as these help avoid unanticipated errors.

The following example demonstrates an if - elseif - else statement:

Demo 2.1: Conditionals/Demos/if.php

```
-----Lines 1 through 11 Omitted-----
12. <?php
13. $age = 21;
14. if ($age >= 21) {
15.     echo 'You can vote and drink!';
16. } elseif ($age >= 18) {
17.     echo 'You can vote, but can\'t drink.';
18. } else {
19.     echo 'You cannot vote or drink.';
20. }
21. ?>
-----Lines 22 through 24 Omitted-----
```

Code Explanation

The file is relatively simple. You can see the different results by changing the value of \$age and visiting <http://localhost:8888/Webucator/php/Conditionals/Demos/if.php>.

Compound If Statements

More complex if statements often require that several conditions be checked. The table below shows *and* and *or* operators for checking multiple conditions and the *not* operator for negating a boolean value (i.e, turning true to false or vice versa).

Logical Operators

Operator	Name	Example
&&	AND	\$a && \$b
	OR	\$a \$b
!	NOT	!\$b

The following example shows these logical operators in practice:

Demo 2.2: Conditionals/Demos/if-2.php

```
-----Lines 1 through 11 Omitted-----
12. <?php
13. $age = 21;
14. $citizen = false;
15. if ($age >= 21 && !$citizen) {
16.     echo 'You can drink, but can\'t vote.';
17. } elseif ($age >= 21) {
18.     echo 'You can vote and drink!';
19. } elseif ($age >= 18 && $citizen) {
20.     echo 'You can vote, but can\'t drink.';
21. } else {
22.     echo 'You cannot vote or drink.';
23. }
24. ?>
-----Lines 25 through 27 Omitted-----
```



2.2. False Equivalents: Falsy Values

PHP can change the type of a value on the fly. For example, the following expression will evaluate to 10:

```
5 + '5'
```

Every expression in PHP can be converted to a boolean value (`true` or `false`). For example, the value `0`, when treated like a boolean, is evaluated as `false`. All other numbers are evaluated as `true`. Examine the following code:

```
if (1) {  
    // This code will run.  
}  
  
if (0) {  
    // This code will NOT run.  
}
```

Non-boolean values that are evaluated as `false` when treated as a boolean are called *falsy*. The list below shows the most common *falsy* values:

1. The integer `0`
2. The float `0.0`
3. The string `"0"`
4. An empty string: `""`
5. An empty array.
6. `NULL` (a variable with no value).

To explicitly convert a value to a boolean, pass it to the built-in `boolval()` function:

```
boolval('0');
```

Note that when echo-ing the boolean `false`, it gets converted to an empty string, so nothing will show up in the browser. To force a value to show up, pass it to the `var_dump()` function:

```
echo var_dump(boolval('0'));
```

You will likely only need this function for testing purposes. To see this in action, review `Conditionals/Demos/testing-bools.php` in your editor and run the page in your browser.



2.3. Testing for Variable Existence⁷

In some cases, you won't be sure whether a variable has been declared and/or whether it has a non-null value. PHP provides built-in functions for checking if a variable exists, checking if a variable holds a value, and removing a variable.⁸

Variable Existence

Function	Explanation	Example
<code>isset()</code>	Returns <code>true</code> if a variable exists and has been set. Otherwise, returns <code>false</code> .	<code>isset(\$a)</code>
<code>empty()</code>	Returns <code>true</code> if the variable doesn't exist or contains a falsy value. Otherwise, returns <code>false</code> .	<code>empty(\$a)</code>
<code>unset()</code>	Removes a variable from memory.	<code>unset(\$a)</code>

The `isset()` and `empty()` functions are close to opposites, which would make `!empty()` the same as `isset()`, but that's not quite true. `!empty()` returns `true` if the variable isn't set, but it also returns `true` if it is set to a *falsy* value (e.g., an empty string). The example below illustrates this:

7. For a complete list of variable functions see <https://www.php.net/manual/en/ref.var.php>.

8. To output the results of these functions to a browser, use the `var_dump()` function (e.g. `var_dump(isset($a));`).

Demo 2.3: Conditionals/Demos/hello-hi.php

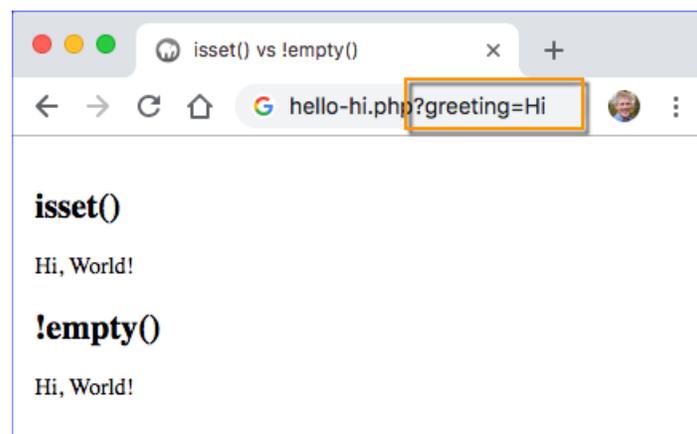
```
1.  <?php
2.    if (isset($_GET['greeting'])) {
3.        $greeting1 = $_GET['greeting'];
4.    } else {
5.        $greeting1 = 'Hello';
6.    }
7.
8.    if (!empty($_GET['greeting'])) {
9.        $greeting2 = $_GET['greeting'];
10.   } else {
11.       $greeting2 = 'Hello';
12.   }
13.   ?>
-----Lines 14 through 23 Omitted-----
24. <main>
25.   <h2>isset()</h2>
26.   <p><?= $greeting1 ?>, World!</p>
27.   <h2>!empty()</h2>
28.   <p><?= $greeting2 ?>, World!</p>
29. </main>
-----Lines 30 through 31 Omitted-----
```

Evaluation
Copy

Code Explanation

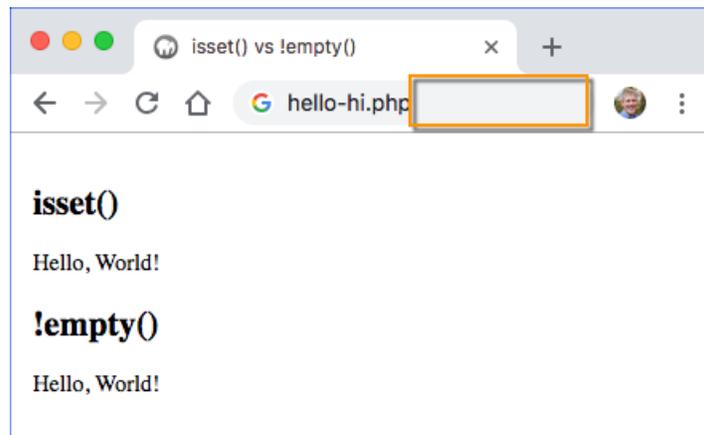
The screenshots below show different outcomes depending on whether and to what value `$_GET['greeting']` is set:

1. `hello-hi.php?greeting=Hi`:



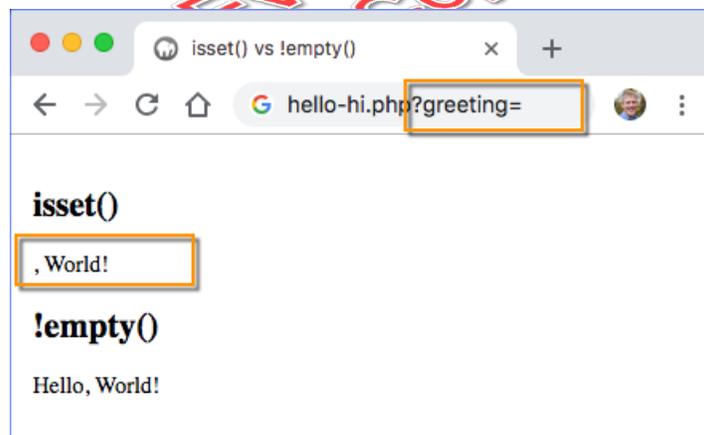
In this case, there is no difference.

2. `hello-hi.php`:



In this case, there is also no difference.

3. `hello-hi.php?greeting=:`

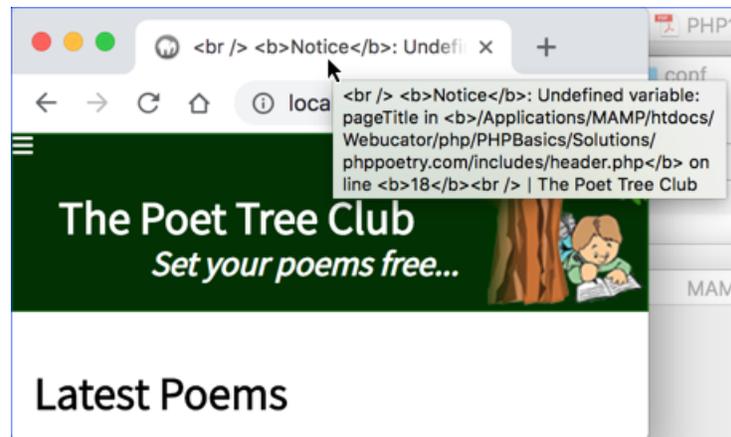


In this case, there is a difference: `$_GET['greeting']` is set, but it is not *not empty* (i.e., it *is* empty).

📄 Exercise 7: Checking for Variable Existence

🕒 15 to 20 minutes

You will remember that we had an issue with the page title of our PHP Poetry site's home page:



In this exercise, we will fix that issue.

1. Open `Conditionals/Exercises/phppoetry.com/includes/header.php` in your editor.
2. In the PHP block at the top, create a new variable called `$pageTitleTag` that will be used as the value of the `<title>` tag. Add code that checks for the existence of `$pageTitle`.
 - A. If the `$pageTitle` variable exists, set `$pageTitleTag` to `$pageTitle` followed by `' | The Poet Tree Club'`.
 - B. If the `$pageTitle` variable does not exist, set `$pageTitleTag` to `'The Poet Tree Club'`.
3. Change the `<title>` tag, so that it just contains the value of `$pageTitleTag`.

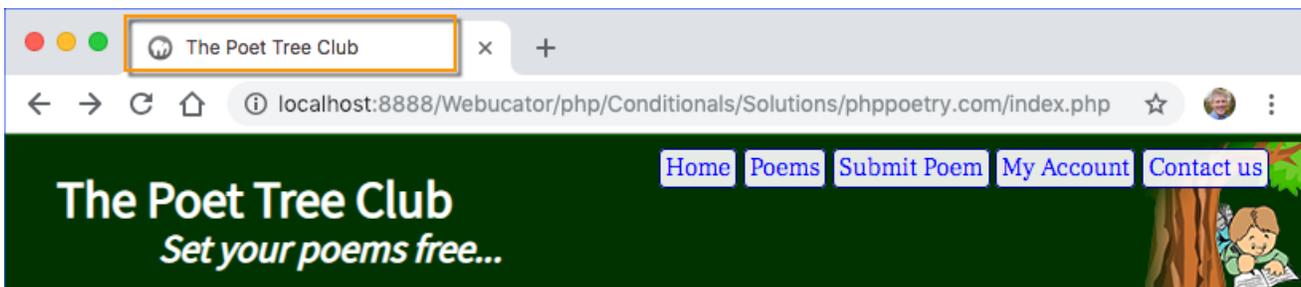
Solution: Conditionals/Solutions/phppoetry.com/includes/header.php

```
1.  <?php
2.      require_once 'utilities.php';
3.      if (isDebugMode()) {
4.          ini_set('display_errors', '1');
5.      }
6.
7.      if (empty($pageTitle)) {
8.          $pageTitleTag = 'The Poet Tree Club';
9.      } else {
10.         $pageTitleTag = $pageTitle . ' | The Poet Tree Club';
11.     }
12.  ?>
-----Lines 13 through 26 Omitted-----
27.  <title><?= $pageTitleTag ?></title>
-----Lines 28 through 48 Omitted-----
```

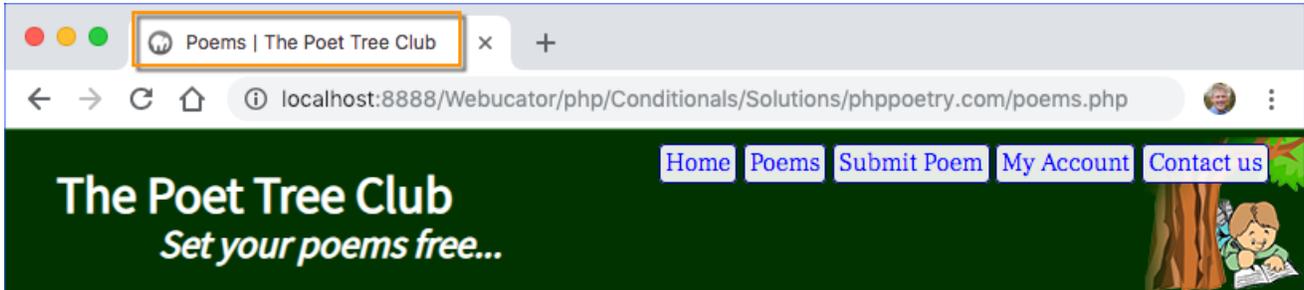
Evaluation Copy

Code Explanation

Now when you visit <http://localhost:8888/Webucator/php/Conditionals/Exercises/phppoetry.com/index.php>, the title should show up as 'The Poet Tree Club':



Other pages should still have the prefixed title:



2.4. switch/case

A switch/case statement is similar to an if statement, except that it can only check for an equality comparison of a single expression. It cannot, for example, be used to check if one value is higher than another.

```
switch (expression)
{
    case 'a' :
        echo 'expression is a';
        break;
    case 'b' :
        echo 'expression is b';
        break;
    case 'c' :
        echo 'expression is c';
        break;
    default :
        echo 'expression is unknown';
}
```

Evaluation
Copy

The break statement is important. Without it, after a single match is found, all following statements will execute.

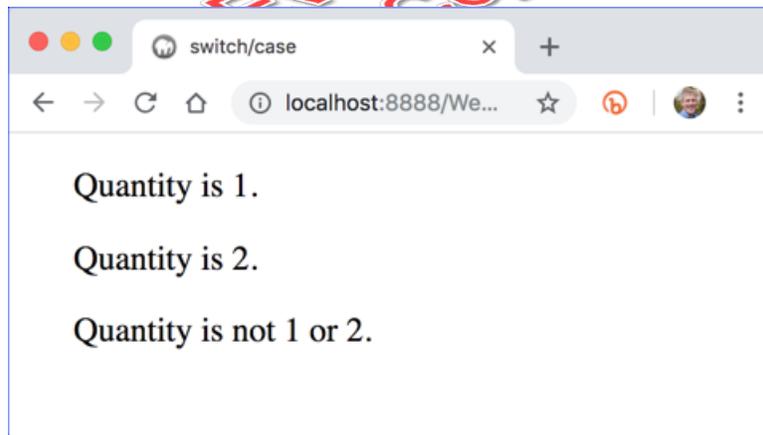
The following example demonstrates a switch/case statement without break statements:

Demo 2.4: Conditionals/Demos/switch-no-break.php

```
-----Lines 1 through 11 Omitted-----  
12. <?php  
13. $quantity = 1;  
14. switch ($quantity) {  
15.     case 1 :  
16.         echo '<p>Quantity is 1.</p>';  
17.     case 2 :  
18.         echo '<p>Quantity is 2.</p>';  
19.     default :  
20.         echo '<p>Quantity is not 1 or 2.</p>';  
21. }  
22. ?>  
-----Lines 23 through 25 Omitted-----
```

Code Explanation

The screenshot below shows the result:



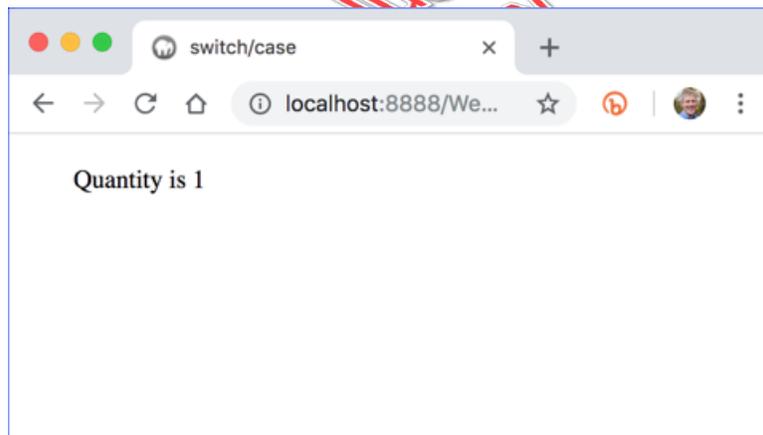
Notice that, once a match is found, all remaining echo statements are output. The following example shows how this can be fixed by adding break statements:

Demo 2.5: Conditionals/Demos/switch-with-break.php

```
-----Lines 1 through 11 Omitted-----
12. <?php
13. $quantity = 1;
14. switch ($quantity) {
15.     case 1 :
16.         echo 'Quantity is 1';
17.         break;
18.     case 2 :
19.         echo 'Quantity is 2';
20.         break;
21.     default :
22.         echo 'Quantity is not 1 or 2';
23. }
24. ?>
-----Lines 25 through 27 Omitted-----
```

Code Explanation

This time, only the first statement is output:



Use Case for switch without break

In most cases, you will include `break` statements in your `switch` conditions; however, there are cases when it makes sense to continue to execute all the subsequent statements in a `switch` condition after a match has been found. Consider the following, in which permissions are being added to a list based on a user's role:

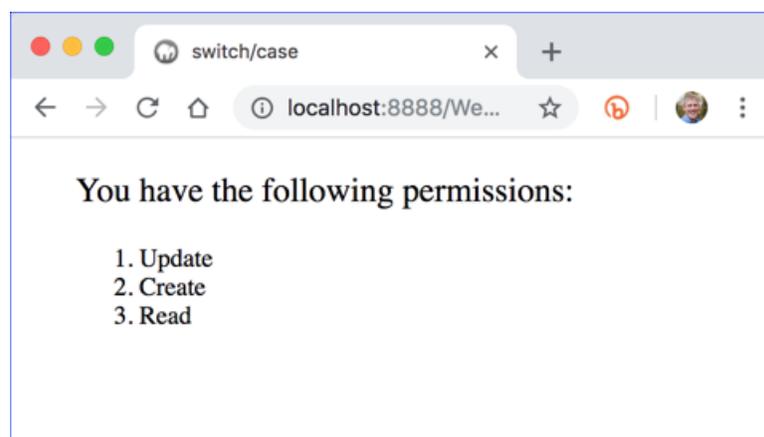
Demo 2.6: Conditionals/Demos/switch-no-break-use-case.php

```
-----Lines 1 through 11 Omitted-----
12. <p>You have the following permissions:</p>
13. <ol>
14. <?php
15. $role = 'Admin';
16. switch ($role) {
17.     case 'SuperAdmin' :
18.         echo '<li>Delete</li>';
19.     case 'Admin' :
20.         echo '<li>Update</li>';
21.     case 'Contributor' :
22.         echo '<li>Create</li>';
23.     default :
24.         echo '<li>Read</li>';
25. }
26. ?>
-----Lines 27 through 30 Omitted-----
```

Code Explanation

- SuperAdmin will get all permissions.
- Admin will get *update*, *create*, and *read* permissions.
- Contributor will get *create* and *read* permissions.
- All others will only get *read* permissions.

As we have set `$role` to “Admin”, this will output the following:



Order of Conditions

In conditional statements, it's good practice to test for the most likely cases/matches first so PHP can find the correct code to execute more quickly.

Evaluation
Copy

Exercise 8: Working with Conditions

 20 to 30 minutes

In this exercise, you will create a page for handling a simple form submission.

1. Open `Conditionals/Exercises/greeting.html` in your editor and review the code. This is the form that will be submitted. Filled out, it looks like this:

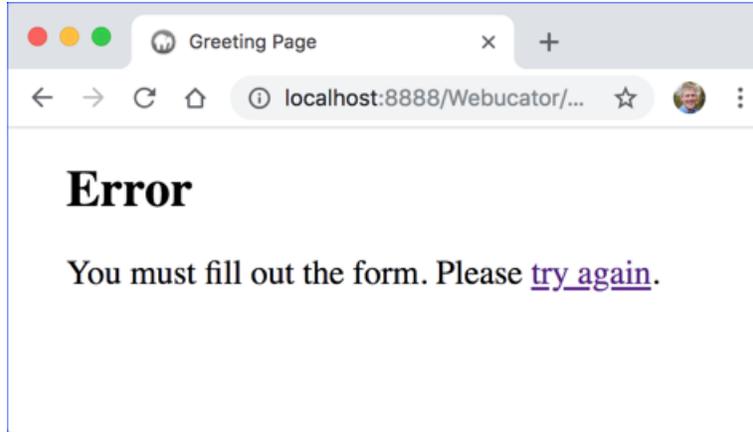


The screenshot shows a web browser window with the title 'Greeting'. The address bar shows the URL `localhost:8888/Webucator/php/Conditionals/Exerc...`. The form contains the following elements:

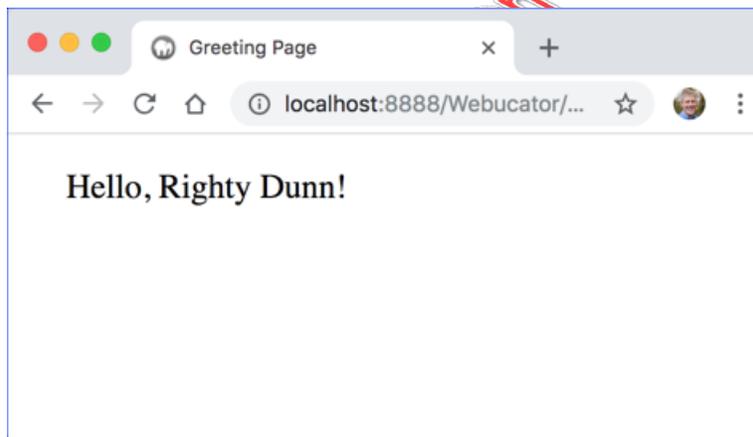
- A label 'Last Name:' followed by a text input field containing the value 'Dunn'.
- A label 'Are you left- or right-handed?' followed by a radio button group with two options: 'Left' (unselected) and 'Right' (selected).
- A button labeled 'Greet User'.

This form is submitted using the `get` method, which means the form entries will be appended to the query string and will be accessible in the `$_GET` array.

2. Open `Conditionals/Exercises/greeting.php` in your editor.
3. Insert a PHP block that checks to see if the user filled out both the `last-name` and the `dominant-hand` fields in the form.
 - If the user failed to fill out either one of the fields, write out an error message to the screen:



- If the user filled out both fields, return an appropriate greeting such as "Hello, Lefty Dunn!" or "Hello, Righty Dunn!"



4. Try to use both an if condition and a switch statement in this exercise.
5. Visit <http://localhost:8888/Webucator/php/Conditionals/Exercises/greeting.html> to test your solution in your browser. Submit the form with and without data.

Solution: Conditionals/Solutions/greeting.php

```
1.  <!DOCTYPE HTML>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>Greeting Page</title>
9.  </head>
10. <body>
11. <main>
12. <?php
13.     if (empty($_GET['last-name']) || empty($_GET['dominant-hand'])) {
14.         echo '<h1>Error</h1>';
15.         <p>You must fill out the form.
16.         Please <a href="greeting.html">try again</a>.</p>;
17.     } else {
18.         $lastName = $_GET['last-name'];
19.         $dominantHand = $_GET['dominant-hand'];
20.         switch($dominantHand) {
21.             case 'left' :
22.                 echo "<p>Hello, Lefty $lastName!</p>";
23.                 break;
24.             default :
25.                 echo "<p>Hello, Righty $lastName!</p>";
26.         }
27.     }
28. ?>
29. </main>
30. </body>
31. </html>
```



2.5. Ternary Operator

The ternary operator provides a shortcut for if conditions. The syntax is as follows:

```
$varName = (expression) ? $valueIfTrue : $valueIfFalse;
```

If the expression is true, `$varName` will be assigned `$valueIfTrue`. Otherwise, it will be assigned `$valueIfFalse`.

To illustrate, consider this if condition:

```
if (empty($_GET['greeting'])) {  
    $greeting = 'Hello';  
} else {  
    $greeting = $_GET['greeting'];  
}
```

Using the ternary operator, it could be rewritten like this:

```
$greeting = empty($_GET['greeting']) ? 'Hello' : $_GET['greeting'];
```

For longer conditions, it can be more readable to break this across lines:

```
$greeting = empty($_GET['greeting'])  
    ? 'Hello'  
    : $_GET['greeting'];
```

Open `Conditionals/Demos/hello-hi-ternary.php` in your editor to see this code in a file.

Exercise 9: The Ternary Operator

 10 to 15 minutes

In this exercise, you will modify the `$pageTitle` check in `header.php` to use the ternary operator instead of an `if` condition.

1. Open `Conditionals/Exercises/phppoetry.com/includes/header.php` in your editor.
2. Change the `if` condition that you wrote in the earlier exercise to use the ternary operator instead.

Solution:

Conditionals/Solutions/phppoetry.com/includes/header-ternary.php

```
1. <?php
2.     require_once 'utilities.php';
3.     if (isDebugMode()) {
4.         ini_set('display_errors', '1');
5.     }
6.
7.     $pageTitleTag = empty($pageTitle)
8.         ? 'The Poet Tree Club'
9.         : $pageTitle . ' | The Poet Tree Club';
10. ?>
-----Lines 11 through 46 Omitted-----
```

Evaluation
*
Copy

2.6. Null Coalescing Operator

PHP 7 introduced the *Null Coalescing Operator*, the double question mark (??), which can be used to set a default value for a variable. It is particularly useful when you don't know if a superglobal variable is set. Consider the following code:

```
$greeting = $_GET['greeting'] ?? 'Hello';
```

If `greeting` is passed in on the query string, then `$greeting` will be set to its value. However, if it is not passed in, `$greeting` will get 'Hello'. This is shown in the following file:

Demo 2.7: Conditionals/Demos/hello-hi-null-coalescing.php

```
1.  <?php
2.      $greeting = $_GET['greeting'] ?? 'Hello';
3.  ?>
4.  <!DOCTYPE html>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <link rel="stylesheet" href="../../static/styles/normalize.css">
10. <link rel="stylesheet" href="../../static/styles/styles.css">
11. <title><?= $greeting ?>, World!</title>
12. </head>
13. <body>
14. <main>
15. <?php
16.     echo "<p>$greeting, World!</p>";
17. ?>
18. </main>
19. </body>
20. </html>
```



Conclusion

In this lesson, you have learned to write if - elseif - else conditions, switch/case statements, ternary statements, and to use the null coalescing operator.

LESSON 3

Arithmetic Operators and Loops

Topics Covered

- PHP operators.
- The modulus operator.
- Loops in PHP.

Introduction

Arithmetic Operators are used to perform math operations in PHP. Loops are used to run the same code a number of times until some condition is met. In this lesson, you will learn about both.

*Evaluation
Copy*

3.1. Arithmetic Operators

Arithmetic operators in PHP are similar to those found in many modern C-like programming languages.

Arithmetic Operators

Operator	Name	Example
+	Addition	$\$a + \b
-	Subtraction	$\$a - \b
*	Multiplication	$\$a * \b
/	Division	$\$a / \b
%	Modulus (Remainder)	$\$a \% \b

Assignment Operators

Operator	Name	Example
=	Assignment	<code>\$a = 1;</code> <code>\$c = 'Hello' . ' world!';</code>
<code>+=</code> <code>-=</code> <code>*=</code> <code>/=</code> <code>%=</code> <code>.=</code>	Combination Assignment (<code>\$a+=3</code> is the same as <code>\$a=\$a+3</code>)	<code>\$a += 3;</code> <code>\$a -= 4;</code> <code>\$a *= 5;</code> <code>\$a /= 2;</code> <code>\$a %= 2;</code>
<code>++</code>	Increment By One (<code>\$a++</code> is the same as <code>\$a=\$a+1</code> or <code>\$a+=1</code>)	<code>\$a++;</code> <code>++\$a;</code>
<code>--</code>	Decrement By One (<code>\$a--</code> is the same as <code>\$a=\$a-1</code> or <code>\$a-=1</code>)	<code>\$a--;</code> <code>--\$a;</code>



3.2. The Modulus Operator

The *modulus* operator (%) is used to find the remainder after division:

```
5 % 2 // returns 1
11 % 3 // returns 2
22 % 4 // returns 2
22 % 3 // returns 1
10934 % 324 // returns 242
```

The modulus operator is useful for determining whether a number is even or odd:

```
1 % 2 // returns 1: odd
2 % 2 // returns 0: even
3 % 2 // returns 1: odd
4 % 2 // returns 0: even
5 % 2 // returns 1: odd
6 % 2 // returns 0: even
```



3.3. Loops

As the name implies, loops are used to loop (or iterate) over code blocks. The following section shows the syntax for different types of loops. Each loop will return “12345”.

There are several types of loops in PHP:

- while
- do...while
- for
- foreach⁹

❖ 3.3.1. while

while loops are used to execute a block of code repeatedly *while* one or more conditions is true.

```
$a=1;
while ($a < 6) {
    echo $a;
    $a++;
}
```

Evaluation
Copy

❖ 3.3.2. do...while

do...while loops are used to execute a block of code repeatedly until one or more conditions is found to be *false*. The difference between while loops and do...while loops is that the condition is checked *after the code block is executed*. This means that, in a do...while loop, the code block will always be executed at least once.

```
$a=1;
do {
    echo $a;
    $a++;
}
while ($a < 6);
```

9. foreach loops will be covered when we cover arrays.

❖ 3.3.3. for

A for loop takes three expressions separated by semi-colons and grouped in parentheses before the block to be iterated through.

1. The first expression is executed once before the loop starts. It is usually used to initialize the conditions.
2. The second expression is evaluated before each iteration through the loop. If it evaluates to `false`, the loop ends.
3. The third expression is executed at the end of each iteration through the loop. It is usually used to make changes that can affect the second expression.

```
for ($a=1; $a < 6; $a++) {  
    echo $a;  
}
```

❖ 3.3.4. break and continue

To break out of a loop, insert a `break` statement.

```
for ($a=1; $a < 6; $a++) {  
    echo $a;  
    if ($a > 3) {  
        break;  
    }  
}
```

This will output: 123. Then it will break out of the loop.

To jump to the next iteration of a loop without executing the remaining statements in the block, insert a `continue` statement.

```
for ($a=1; $a < 6; $a++) {  
    if ($a === 3) {  
        continue;  
    }  
    echo $a;  
}
```

This will output 1245. It skips 3.



Exercise 10: Working with Loops

 10 to 15 minutes

1. Open `OperatorsAndLoops/Exercises/loops.php` in your editor.
2. Under the **while** header, use a `while` loop to output all the even numbers that are less than or equal to 100.
3. Under the **for** header, use a `for` loop to output all the odd numbers that are less than or equal to 100.

Challenge

In both loops, skip all numbers that are divisible by 3. Be careful not to get caught in an infinite loop.

Solution: OperatorsAndLoops/Solutions/loops.php

```
1. <!DOCTYPE HTML>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Loops</title>
9. </head>
10. <body>
11. <main>
12. <h2>while</h2>
13. <ul>
14. <?php
15.     $a=2;
16.     while ($a <= 100) {
17.         echo "<li>$a</li>";
18.         $a+=2;
19.     }
20. ?>
21. </ul>
22.
23. <h2>for</h2>
24. <ul>
25. <?php
26.     for ($a=1; $a <= 100; $a+=2) {
27.         echo "<li>$a</li>";
28.     }
29. ?>
30. </ul>
31. </main>
32. </body>
33. </html>
```

Evaluation
Copy

Challenge Solution: OperatorsAndLoops/Solutions/loops-challenge.php

```
-----Lines 1 through 11 Omitted-----
12. <h2>while</h2>
13. <ul>
14. <?php
15.     $a=2;
16.     while ($a <= 100) {
17.         if ($a % 3 === 0) {
18.             $a+=2;
19.             continue;
20.         }
21.         echo "<li>$a</li>";
22.         $a+=2;
23.     }
24. ?>
25. </ul>
26.
27. <h2>for</h2>
28. <ul>
29. <?php
30.     for ($a=1; $a <= 100; $a+=2) {
31.         if ($a % 3 === 0) {
32.             continue;
33.         }
34.         echo "<li>$a</li>";
35.     }
36. ?>
37. </ul>
-----Lines 38 through 40 Omitted-----
```

Evaluation
Copy

Conclusion

In this lesson, you have learned to work with arithmetic operators and several different types of loops.

LESSON 4

Arrays

Topics Covered

- ☑ Indexed arrays.
- ☑ Associative arrays.
- ☑ Two-dimensional arrays.
- ☑ Array-manipulation functions.

Introduction

Up to this point, we have dealt only with variables that store single values, called scalar variables. In this lesson, we will be covering arrays. Arrays are variables that store sets of values.

Evaluation
*
Copy

4.1. Indexed Arrays

Indexed arrays are similar to tables with a single column. An indexed array can contain zero or more elements. In PHP, like in many programming languages, the first element of an array is in the “zeroeth” position. An array with no elements has a zero length.

❖ 4.1.1. Initializing Arrays

Arrays are initialized with the `array()` function, which can take a list of comma-delimited values that become the elements in the new array. The following initializes a zero-length array and then adds four elements to the array:

```
$beatles = array();  
$beatles[0] = 'John';  
$beatles[1] = 'Paul';  
$beatles[2] = 'George';  
$beatles[3] = 'Ringo';
```

The `$beatles` array could also be created in a single line by passing values into the `array()` function:

```
$beatles = array('John', 'Paul', 'George', 'Ringo');
```

But perhaps the simplest way of creating an array is to use square brackets. This is known as the *short array syntax*:

```
$beatles = ['John', 'Paul', 'George', 'Ringo'];
```

❖ 4.1.2. Appending to an Array

If you know how many elements are in an array, you can append to the array by specifying the index. For example, you could append a fifth Beatle to the `$beatles` array like this:

```
$beatles[4] = 'Nat';
```

However, often you won't know how many elements are in an array. Although you can easily figure this out, doing so requires an extra step. PHP provides an easy way of appending to an array of any length. Simply leave out the index. You could add a sixth Beatle to the `$beatles` array like this:

```
$beatles[] = 'Connie';
```

Now, `$beatles[5]` will contain 'Connie'.

❖ 4.1.3. Reading from Arrays

Reading from arrays is just a matter of pointing to a specific index or key.

```
echo $beatles[2]; // outputs George
```

❖ 4.1.4. Looping through Arrays

The following code will loop through the entire `$beatles` array outputting each element to the browser.

```
foreach ($beatles as $beatle) {
    echo "$beatle<br>";
}
```

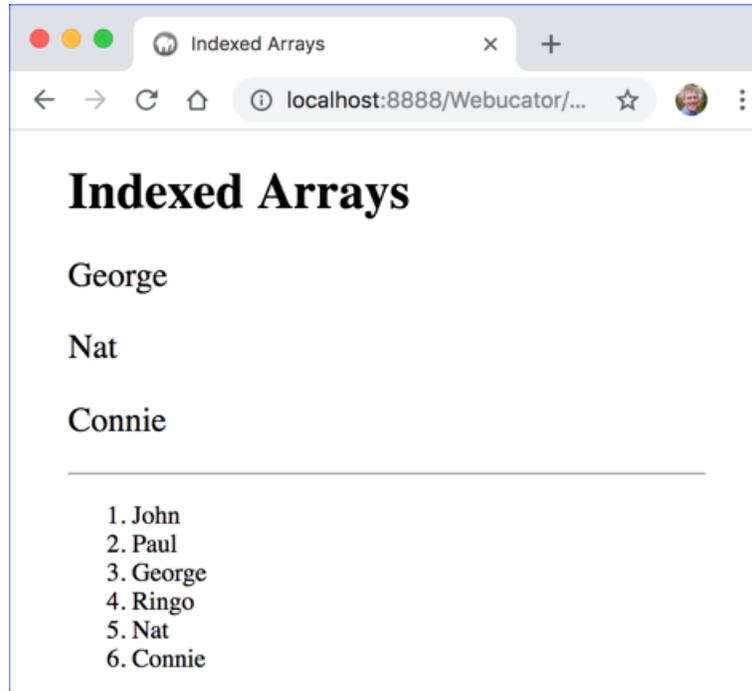
The above code snippets are combined in the following example:

Demo 4.1: Arrays/Demos/indexed-arrays.php

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Indexed Arrays</title>
9. </head>
10. <body>
11. <main>
12. <h1>Indexed Arrays</h1>
13. <?php
14.     $beatles = ['John', 'Paul', 'George', 'Ringo'];
15.
16.     echo "<p>$beatles[2]</p>"; //outputs George
17.
18.     $beatles[4] = 'Nat';
19.     echo "<p>$beatles[4]</p>"; //outputs Nat
20.
21.     $beatles[] = 'Connie';
22.     echo "<p>$beatles[5]</p>"; //outputs Connie
23. ?>
24. <hr>
25. <ol>
26. <?php
27.     foreach ($beatles as $beatle) {
28.         echo "<li>$beatle</li>";
29.     }
30. ?>
31. </ol>
32. </main>
33. </body>
34. </html>
```

Code Explanation

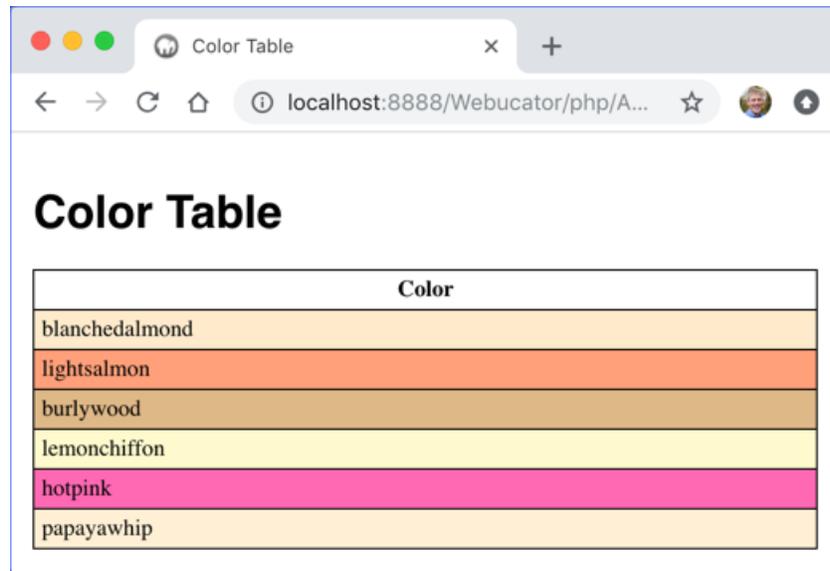
Navigate to <http://localhost:8888/Webucator/php/Arrays/Demos/indexed-arrays.php> to see this page in your browser:



Exercise 11: Working with Indexed Arrays

🕒 10 to 15 minutes

In this exercise, you will use arrays to create a table with a single column that lists all your favorite colors. As shown in the screenshot below, the background of each table row should be the same as the color named in the row:



Color Names

See https://developer.mozilla.org/en-US/docs/Web/CSS/color_value for a list of color names.

1. Open `Arrays/Exercises/color-table.php` for editing.
2. Create an array that holds your favorite colors.
3. Inside of the open and close `<tbody>` tags, loop through the array outputting a table row for each element.
4. Test your solution in a browser.

Solution: Arrays/Solutions/color-table.php

```
-----Lines 1 through 11 Omitted-----
12. <h1>Color Table</h1>
13. <?php
14.     $favColors = ['blanchedalmond',
15.                  'lightsalmon',
16.                  'burlywood',
17.                  'lemonchiffon',
18.                  'hotpink',
19.                  'papayawhip'];
20. ?>
21. <table>
22.     <thead>
23.         <tr>
24.             <th>Color</th>
25.         </tr>
26.     </thead>
27.     <tbody>
28.         <?php
29.             foreach ($favColors as $color) {
30.                 echo "<tr style='background-color:$color'>
31.                     <td>$color</td>
32.                 </tr>";
33.             }
34.         ?>
35.     </tbody>
36. </table>
-----Lines 37 through 39 Omitted-----
```



4.2. Associative Arrays

Whereas indexed arrays are indexed numerically, associative arrays are indexed using names. For example, instead of Ringo being indexed as 3, he could be indexed as “drummer”.

❖ 4.2.1. Initializing Associative Arrays

Like with indexed arrays, we can initialize a zero-length associative array and then add elements:

```
$beatles = array();
$beatles['singer1'] = 'Paul';
$beatles['singer2'] = 'John';
$beatles['guitarist'] = 'George';
$beatles['drummer'] = 'Ringo';
```

Or the array could be created using the *double-arrow* operator (`=>`) in a single step as follows:

```
$beatles = array('singer1' => 'John',
                'singer2' => 'Paul',
                'guitarist' => 'George',
                'drummer' => 'Ringo');
```

Like with indexed arrays, you can use the short array syntax to create associative arrays:

```
$beatles = ['singer1' => 'John',
            'singer2' => 'Paul',
            'guitarist' => 'George',
            'drummer' => 'Ringo'];
```

Evaluation
Copy

❖ 4.2.2. Reading from Associative Arrays

Reading from associative arrays is as simple as reading from indexed arrays:

```
echo $beatles['drummer']; // outputs Ringo
```

❖ 4.2.3. Looping through Associative Arrays

The following code will loop through the entire `$beatles` array outputting each element and its key to the browser:

```
foreach ($beatles as $key => $beatle) {
    echo "<strong>$key:</strong> $beatle<br>";
}
```

The above code snippets are combined in the following example:

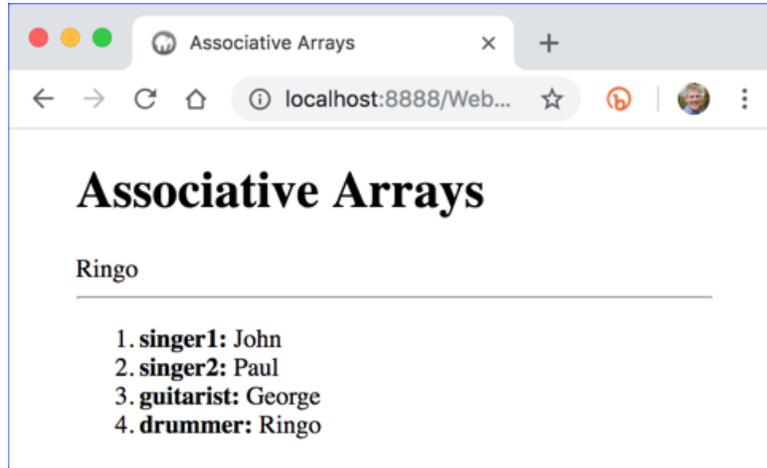
Demo 4.2: Arrays/Demos/associative-arrays.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>Associative Arrays</title>
9.  </head>
10. <body>
11. <main>
12.   <h1>Associative Arrays</h1>
13.   <?php
14.     $beatles = ['singer1' => 'John',
15.               'singer2' => 'Paul',
16.               'guitarist' => 'George',
17.               'drummer' => 'Ringo'];
18.
19.     echo $beatles['drummer']; // outputs Ringo
20.   ?>
21.   <hr>
22.   <ol>
23.     <?php
24.       foreach ($beatles as $key => $beatle) {
25.         echo "<li><strong>$key:</strong> $beatle</li>";
26.       }
27.     ?>
28.   </ol>
29. </main>
30. </body>
31. </html>
```



Code Explanation

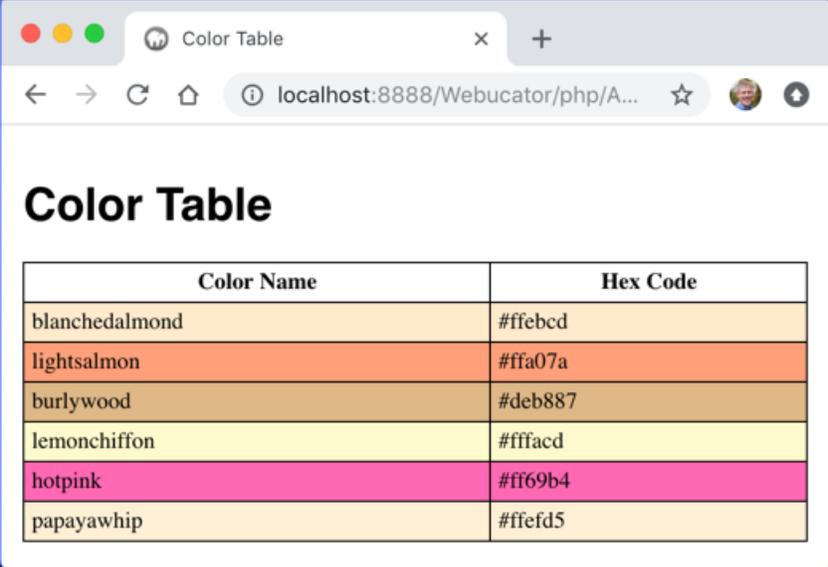
Navigate to <http://localhost:8888/Webucator/php/Arrays/Demos/associative-arrays.php> to see this page in your browser:



Exercise 12: Working with Associative Arrays

🕒 10 to 15 minutes

In this exercise, you will use arrays to create a table with two columns that lists all your favorite colors and their hexadecimal equivalents. The background of each table row should be the same as the color named in the row as shown in the screenshot below:



Color Name	Hex Code
blanchedalmond	#ffebed
lightsalmon	#ffa07a
burlywood	#deb887
lemonchiffon	#fffacd
hotpink	#ff69b4
papayawhip	#ffe5d5

1. Open `Arrays/Exercises/color-table-2.php` for editing.
2. Create an associative array that holds color hex codes indexed by their color names.
3. Inside of the open and close `<tbody>` tags, write code to loop through the array outputting a table row with two columns for each element in the array.
4. Test your solution in a browser.

Solution: Arrays/Solutions/color-table-2.php

```
-----Lines 1 through 11 Omitted-----
12. <h1>Color Table</h1>
13. <?php
14.     $favColors = [
15.         'blanchedalmond' => '#ffeacd',
16.         'lightsalmon' => '#ffa07a',
17.         'burlywood' => '#deb887',
18.         'lemonchiffon' => '#fffacd',
19.         'hotpink' => '#ff69b4',
20.         'papayawhip' => '#ffeefd5'
21.     ];
22. ?>
23.
24. <table>
25.     <thead>
26.         <tr>
27.             <th>Color Name</th>
28.             <th>Hex Code</th>
29.         </tr>
30.     </thead>
31.     <tbody>
32.     <?php
33.         foreach ($favColors as $key => $item) {
34.             echo "<tr style='background-color:$key'>
35.                 <td>$key</td>
36.                 <td>$item</td>
37.             </tr>";
38.         }
39.     ?>
40.     </tbody>
41. </table>
-----Lines 42 through 44 Omitted-----
```



4.3. Superglobal Arrays

The superglobal arrays (see page 23) are associative arrays. The file below outputs all the contents of the superglobal arrays using `foreach` loops. Don't worry about the `session_start()` statement at the top. We'll cover that in detail later in the course.

Demo 4.3: Arrays/Demos/superglobals.php

```
-----Lines 1 through 14 Omitted-----
15. <h1>Superglobal Arrays</h1>
16. <h2>$_COOKIE</h2>
17. <ol>
18.     <?php
19.         foreach ($_COOKIE as $key => $item) {
20.             echo "<li><strong>$key:</strong> $item</li>";
21.         }
22.     ?>
23. </ol>
24. <hr>
25. <h2>$_ENV</h2>
26. <ol>
27.     <?php
28.         foreach ($_ENV as $key => $item) {
29.             echo "<li><strong>$key:</strong> $item</li>";
30.         }
31.     ?>
32. </ol>
33. <hr>
34. <h2>$_FILES</h2>
35. <ol>
36.     <?php
37.         foreach ($_FILES as $key => $item) {
38.             echo "<li><strong>$key:</strong> $item</li>";
39.         }
40.     ?>
41. </ol>
42. <hr>
43. <h2>$_GET</h2>
44. <ol>
45.     <?php
46.         foreach ($_GET as $key => $item) {
47.             echo "<li><strong>$key:</strong> $item</li>";
48.         }
49.     ?>
50. </ol>
51. <hr>
52. <h2>$_POST</h2>
53. <ol>
54.     <?php
55.         foreach ($_POST as $key => $item) {
56.             echo "<li><strong>$key:</strong> $item</li>";
57.         }

```

Valuation
Copy

```

58.     ?>
59.     </ol>
60.     <hr>
61.     <h2>$_REQUEST</h2>
62.     <ol>
63.         <?php
64.             foreach ($_REQUEST as $key => $item) {
65.                 echo "<li><strong>$key:</strong> $item</li>";
66.             }
67.         ?>
68.     </ol>
69.     <hr>
70.     <h2>$_SESSION</h2>
71.     <ol>
72.         <?php
73.             foreach ($_SESSION as $key => $item) {
74.                 echo "<li><strong>$key:</strong> $item</li>";
75.             }
76.         ?>
77.     </ol>
78.     <hr>
79.     <h2>$_SERVER</h2>
80.     <ol>
81.         <?php
82.             foreach ($_SERVER as $key => $item) {
83.                 echo "<li><strong>$key:</strong> $item</li>";
84.             }
85.         ?>
86.     </ol>
-----Lines 87 through 89 Omitted-----

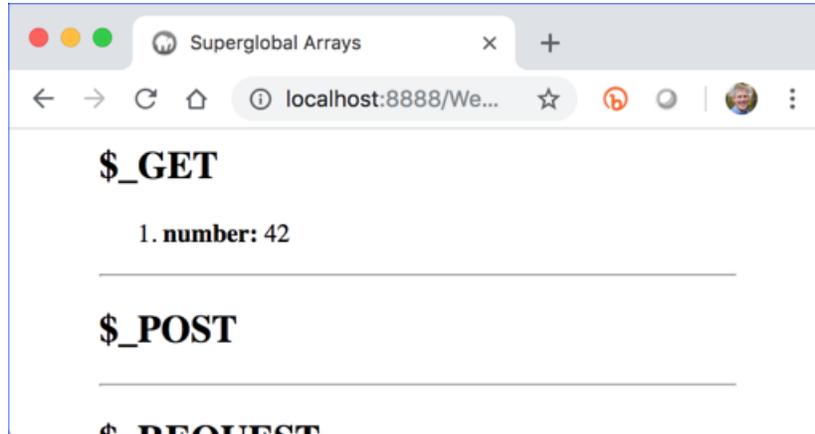
```

Code Explanation

Navigate to <http://localhost:8888/Webucator/php/Arrays/Demos/superglobals.php> to see this page in your browser. Note that many of these superglobal arrays will be empty. For example, the `$_POST` and `$_GET` arrays will be empty because no data has been posted via a form or sent via the query string. If you want to populate `$_GET` with some value, add a query string with a parameter, like this:

```
superglobals.php?number=42
```

Then you should see the following:



4.4. Multi-dimensional Arrays

In PHP, multi-dimensional arrays are arrays that contain arrays (that may contain arrays, and so on). You can think of a two-dimensional array as a table with the outer array containing the rows and the inner arrays containing the data cells in those rows. For example, a two-dimensional array called `$rockBands` could contain the names of the bands and some of the songs that they sing. Below is a grid that represents such a two-dimensional array:

Rock Band	Song1	Song2	Song3
Beatles	Love Me Do	Hey Jude	Helter Skelter
Rolling Stones	Waiting on a Friend	Angie	Yesterday's Papers
Eagles	Life in the Fast Lane	Hotel California	Best of My Love

The following code creates this two-dimensional array. The internal arrays are highlighted. Note that the header row is not included:

```
$rockBands = [  
  [  
    'Beatles',  
    'Love Me Do',  
    'Hey Jude',  
    'Helter Skelter'  
  ],  
  [  
    'Rolling Stones',  
    'Waiting on a Friend',  
    'Angie',  
    'Yesterday\'s Papers'  
  ],  
  [  
    'Eagles',  
    'Life in the Fast Lane',  
    'Hotel California',  
    'Best of My Love'  
  ]  
];
```

Evaluation
Copy

❖ 4.4.1. Reading from Two-dimensional Arrays

To read an element from a two-dimensional array, you must first identify the index of the “row” and then identify the index of the “column.” For example, the song “Waiting on a Friend” is in row 1, column 1, so it is identified as `$rockBands[1][1]`. Remember that the first row is row 0 and the first column is column 0.

❖ 4.4.2. Looping through Two-dimensional Arrays

To loop through a two-dimensional array, you need to nest one loop inside of another. The following code will create an HTML table from our two-dimensional array:

```
<table>
  <thead>
    <tr>
      <th>Rock Band</th>
      <th>Song 1</th>
      <th>Song 2</th>
      <th>Song 3</th>
    </tr>
  </thead>
  <tbody>
    <?php
      foreach($rockBands as $rockBand) {
        // $rockBand contains an inner array
        echo "<tr>";
        foreach($rockBand as $item) {
          echo "<td>$item</td>";
        }
        echo "</tr>";
      }
    ?>
  </tbody>
</table>
```

The above code snippets are combined in the following example to output a Rock Bands table:

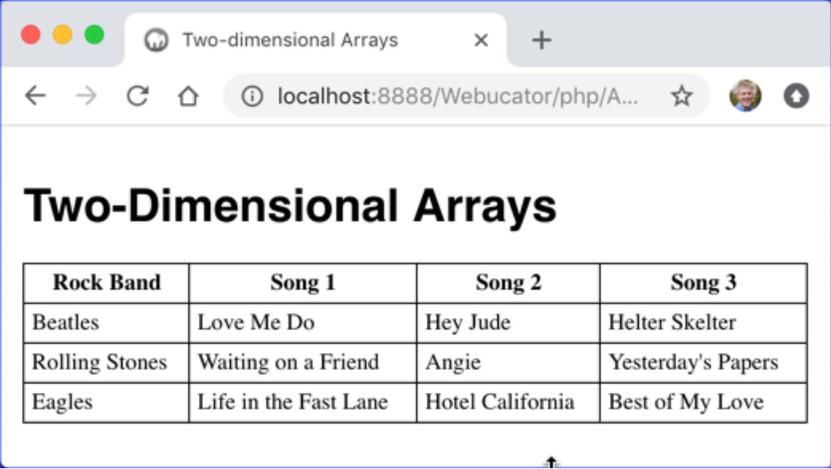
Demo 4.4: Arrays/Demos/two-dimensional-arrays.php

```
-----Lines 1 through 11 Omitted-----
12. <h1>Two-Dimensional Arrays</h1>
13. <?php
14. $rockBands = [
15.     ['Beatles',
16.         'Love Me Do',
17.         'Hey Jude',
18.         'Helter Skelter'],
19.     ['Rolling Stones',
20.         'Waiting on a Friend',
21.         'Angie',
22.         'Yesterday\'s Papers'],
23.     ['Eagles',
24.         'Life in the Fast Lane',
25.         'Hotel California',
26.         'Best of My Love']
27. ];
28. ?>
29. <table>
30.     <thead>
31.         <tr>
32.             <th>Rock Band</th>
33.             <th>Song 1</th>
34.             <th>Song 2</th>
35.             <th>Song 3</th>
36.         </tr>
37.     </thead>
38.     <tbody>
39.         <?php
40.         foreach($rockBands as $rockBand) {
41.             // $rockBand contains an inner array
42.             echo '<tr>';
43.             foreach($rockBand as $item) {
44.                 echo "<td>$item</td>";
45.             }
46.             echo '</tr>';
47.         }
48.         ?>
49.     </tbody>
50. </table>
-----Lines 51 through 53 Omitted-----
```

Evaluation
Copy

Code Explanation

Navigate to `http://localhost:8888/Webucator/php/Arrays/Demos/two-dimensional-arrays.php` to see this page in your browser:



The screenshot shows a web browser window with the title 'Two-dimensional Arrays'. The address bar displays 'localhost:8888/Webucator/php/A...'. The main content of the page is a table with the following data:

Rock Band	Song 1	Song 2	Song 3
Beatles	Love Me Do	Hey Jude	Helter Skelter
Rolling Stones	Waiting on a Friend	Angie	Yesterday's Papers
Eagles	Life in the Fast Lane	Hotel California	Best of My Love

❖ 4.4.3. Two-dimensional Associative Arrays

The two-dimensional array above would be better expressed as an associative array to clarify the relationship between the rock band and the songs. The following demo shows how to do this:

Demo 4.5: Arrays/Demos/two-dimensional-associative-arrays.php

```
-----Lines 1 through 11 Omitted-----
12.  <h1>Two-Dimensional Associative Arrays</h1>
13.  <?php
14.  $rockBands = [
15.      'Beatles' => ['Love Me Do',
16.                  'Hey Jude',
17.                  'Helter Skelter'],
18.      'Rolling Stones' => ['Waiting on a Friend',
19.                          'Angie',
20.                          'Yesterday\'s Papers'],
21.      'Eagles' => ['Life in the Fast Lane',
22.                 'Hotel California',
23.                 'Best of My Love']
24.  ];
25.  ?>
26.  <table>
27.      <thead>
28.          <tr>
29.              <th>Rock Band</th>
30.              <th>Song 1</th>
31.              <th>Song 2</th>
32.              <th>Song 3</th>
33.          </tr>
34.      </thead>
35.      <tbody>
36.          <?php
37.              foreach($rockBands as $rockBand => $songs) {
38.                  // $rockBand contains an inner array
39.                  echo "<tr><td><strong>$rockBand:</strong></td>";
40.                  foreach($songs as $song) {
41.                      echo "<td>$song</td>";
42.                  }
43.                  echo '</tr>';
44.              }
45.          ?>
46.      </tbody>
47.  </table>
-----Lines 48 through 50 Omitted-----
```



Code Explanation

Navigate to <http://localhost:8888/Webucator/php/Arrays/Demos/two-dimensional-associative-arrays.php> to see this page in your browser:



Rock Band	Song 1	Song 2	Song 3
Beatles:	Love Me Do	Hey Jude	Helter Skelter
Rolling Stones:	Waiting on a Friend	Angie	Yesterday's Papers
Eagles:	Life in the Fast Lane	Hotel California	Best of My Love

❖ 4.4.4. Non-tabular Multi-dimensional Arrays

While it is helpful to think of a two-dimensional array as a table, you should be aware that an array can contain subarrays of different lengths and even contain items of different types. To illustrate, let's look again at our \$beatles array:

```
$beatles = ['singer1' => 'John',
           'singer2' => 'Paul',
           'guitarist' => 'George',
           'drummer' => 'Ringo'];
```

The `singer1` and `singer2` keys are problematic. Imagine we were storing data on a lot of rock bands, some of which only had one singer and others which had several. Imagine then that you are tasked with writing code to list all of the singers from the bands. You wouldn't know which bands had keys for 'singer2', 'singer3', etc.

Now take a look at a better way of storing our Beatles data:

```
$beatles = [
  'singers' => ['Paul', 'John'],
  'guitarist' => 'George',
  'drummer' => 'Ringo'
];
```

With this code, you can easily list all the singers, simply by looping through the `$beatles['singers']` array:

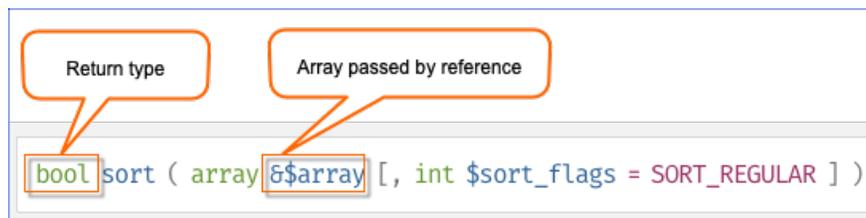
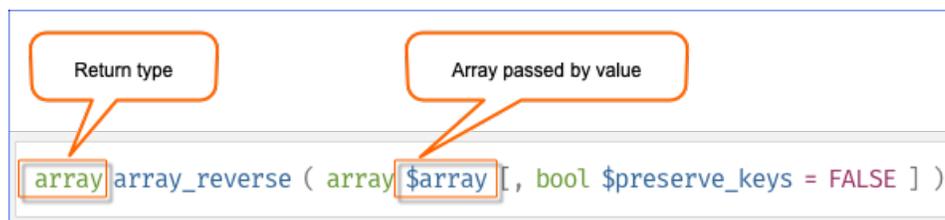
```
foreach ($beatles['singers'] as $singer) {  
    echo "$singer<br>";  
}
```



4.5. Array Manipulation Functions

There are many built-in array-manipulation functions in PHP. The full list is at <https://www.php.net/manual/en/ref.array.php>.

In an upcoming exercise, you will practice working with these functions. In doing so, you will need to read the documentation at [php.net](https://www.php.net). One important thing to know about a function is whether it modifies the original array that is passed in *in place* or creates and returns a new array (or some other object). If the parameter is passed in *by reference*, then it will modify the passed-in array. If it is passed in *by value*, then it will not modify the passed-in array. The documentation will generally explicitly tell you which it will do, but you can also tell from the function signature. The following two signatures illustrate the difference:



Notice that `array_reverse()` receives the array by value, meaning it won't change the original array, and returns a new array. The `sort()` function, on the other hand, receives

the array by reference (that's what the ampersand (&) does) and returns a boolean indicating whether or not it successfully modified the original array.



4.6. in_array() Function

The `in_array()` function is used to check if a value is found in an array. For example:

```
$fruit = ['apple', 'banana', 'pear'];
$foodItem = 'apple';
if (in_array($foodItem, $fruit)) {
    echo "$foodItem is a fruit.";
}
```

📄 Exercise 13: Array Practice

🕒 25 to 40 minutes

In this exercise, you will practice using PHP's built-in array functions and you will write a couple of your own. You will likely need to read the documentation at php.net to see how some of these functions work. You should test your solutions as you work.

1. Open `Arrays/Exercises/arrays.php` for editing.
2. The file contains the following code:
 - A. An `outputArray()` user-defined function that outputs an unordered list of the keys and values in an array.
 - B. A `$numbers` array.
 - C. A `$colors` array.
 - D. Beneath those array definitions, it outputs those arrays using the `outputArray()` function:

```
Original Arrays
$numbers


- 0: 5
- 1: 7
- 2: 11
- 3: 3
- 4: -5


$colors


- blanchedalmond: #ffebed
- lightsalmon: #ffa07a
- burlywood: #deb887
- lemonchiffon: #ffffac
- hotpink: #ff69b4
- papayawhip: #ffebed

```

Your job is to write code that uses array functions.

3. `array_reverse()` - Returns a new array in reverse order. Prove it with both arrays. **This one is done for you.**
4. `sort()` - Sorts an array in place and re-indexes. Prove it with the `$numbers` array. **This one is done for you.**

5. `asort()` - Sorts an associative array on its values, keeping the key-value pairs together. Prove it with the `$colors` array.
6. `krsort()` - Sorts an associative array on its keys, keeping the key-value pairs together. Prove it with the `$colors` array.
7. `rsort()` - Sorts an array in reverse order in place and re-indexes. Prove it with the `$numbers` array.
8. `arsort()` - Sorts an associative array on its values in reverse order, keeping the key-value pairs together. Prove it with the `$colors` array.
9. `krsort()` - Sorts an associative array on its keys in reverse order, keeping the key-value pairs together. Prove it with the `$colors` array.
10. `shuffle()` - Randomly shuffles an array and re-indexes. Prove it with the `$numbers` array.
11. `array_keys()` - Returns the keys of an associative array as an indexed array. Prove it with the `$colors` array.
12. `count()` - Counts the number of elements in an array. Prove it with both arrays.
13. `explode()` - Splits a string on a string to create an array. A string of spells is provided. Split it into an array on `,` (a comma followed by a space) and assign the result to `$spellsArray`.
14. `implode()` - Joins an array on a string to create a string. Join the array you created in the last step on `-` (a space, followed by a dash, followed by a space) and assign the result to `$spells`.
15. `is_array()` - Checks if the passed-in argument is an array.
 - A. Write a function called `outputIsArray()` that takes one parameter:
 - i. `$arr`
 - B. If `$arr` is an array, the function should output:

```
<p>That is an array:</p>
```

And then it should pass the array to `outputIsArray()`.
 - C. If `$arr` is not an array, the function should output:

```
<p>That is not an array.</p>
```

- D. Pass `$spellsArray` to `outputIsArray()`.
- E. Pass `$spells` to `outputIsArray()`.

16. `array_key_exists()` - Checks if an array contains a key.

- A. Write a function called `hasKey()` that takes two parameters:
 - i. `$key`
 - ii. `$arr`

- B. If `$key` is found in `$arr`, the function should output:

```
<p>$key is in that array.</p>
```

And then it should pass the array to `outputIsArray()`.

- C. If `$key` is not found in `$arr`, the function should output:

```
<p>$key is in not that array.</p>
```

- D. Use `hasKey()` to check if these colors are in the `$colors` array:
 - i. `hotpink`
 - ii. `olivedrab.`

17. `array_walk()` - Applies a function to every element in an array. There is already a `createButton()` function in the exercise file. Using that function and `array_walk()`, create buttons for all the colors in the `$colors` array.

Solution: Arrays/Solutions/arrays.php

```
-----Lines 1 through 11 Omitted-----
12. <h1>Practice with Arrays</h1>
13. <?php
14.     function outputArray($arr) {
15.         echo '<ul>';
16.         foreach ($arr as $key => $a) {
17.             echo "<li><strong>$key</strong>: $a</li>";
18.         }
19.         echo '</ul>';
20.         echo '<hr>';
21.     }
22.     $numbers = [5, 7, 11, 3, -5];
23.
24.     $colors = [
25.         'blanchedalmond' => '#ffeacd',
26.         'lightsalmon' => '#ffa07a',
27.         'burlywood' => '#deb887',
28.         'lemonchiffon' => '#fffacd',
29.         'hotpink' => '#ff69b4',
30.         'papayawhip' => '#ffeacd'
31.     ];
32.
33.     echo '<h2>Original Arrays</h2>';
34.     echo '<h3>$numbers</h3>';
35.     outputArray($numbers);
36.     echo '<h3>$colors</h3>';
37.     outputArray($colors);
38.
39.     echo '<h2>array_reverse()</h2>';
40.     echo '<p>Returns a new array in reverse order.</p>';
41.     $reversedNumbers = array_reverse($numbers);
42.     $reversedColors = array_reverse($colors);
43.     outputArray($reversedNumbers);
44.     outputArray($reversedColors);
45.
46.     echo '<h2>sort()</h2>';
47.     echo '<p>Sorts on value and re-indexes.</p>';
48.     sort($numbers);
49.     outputArray($numbers);
50.
51.     echo '<h2>asort()</h2>';
52.     echo '<p>Sorts on value. Keeps key-value pairs together.</p>';
53.     asort($colors);
54.     outputArray($colors);
```

```

55.
56.     echo '<h2>ksort()</h2>';
57.     echo '<p>Sorts on key. Keeps key-value pairs together.</p>';
58.     ksort($colors);
59.     outputArray($colors);
60.
61.     echo '<h2>rsort()</h2>';
62.     echo '<p>Sorts in reverse on value and re-indexes.</p>';
63.     rsort($numbers);
64.     outputArray($numbers);
65.
66.     echo '<h2>arsort()</h2>';
67.     echo '<p>Sorts in reverse on value. Keeps key-value pairs.</p>';
68.     arsort($colors);
69.     outputArray($colors);
70.
71.     echo '<h2>krsort()</h2>';
72.     echo '<p>Sorts in reverse on key. Keeps key-value pairs.</p>';
73.     krsort($colors);
74.     outputArray($colors);
75.
76.     echo '<h2>shuffle()</h2>';
77.     echo '<p>Randomly shuffles array and re-indexes. ' ;
78.     shuffle($numbers);
79.     outputArray($numbers);
80.
81.     echo '<h2>array_keys()</h2>';
82.     echo '<p>Returns keys as indexed array.</p>';
83.     $colorKeys = array_keys($colors);
84.     outputArray($colorKeys);
85.
86.     echo '<h2>count()</h2>';
87.     echo '<p>Counts the number of elements in an array.</p>';
88.     echo 'Number of numbers: ' . count($numbers);
89.     echo '<br>';
90.     echo 'Number of colors: ' . count($colors);
91.
92.     echo '<h2>explode()</h2>';
93.     echo '<p>Splits a string on a string to create an array.';
94.     $spells = 'Riddikulus, Obliviate, Avada Kedavra, Expelliarmus';
95.     $spellsArray = explode(' ', $spells);
96.     outputArray($spellsArray);
97.
98.     echo '<h2>implode()</h2>';
99.     echo '<p>Joins an array on a string to create a string.</p>';

```

```

100.     $spells = implode(' - ', $spellsArray);
101.     echo $spells;
102.
103.     echo '<h2>is_array()</h2>';
104.     echo '<p>Checks if a passed-in argument is an array.</p>';
105.
106.     function outputIsArray($arr) {
107.         if (is_array($arr)) {
108.             echo '<p>That is an array.</p>';
109.             outputArray($arr);
110.         } else {
111.             echo '<p>That is not an array.</p>';
112.         }
113.     }
114.
115.     echo outputIsArray($spellsArray);
116.     echo outputIsArray($spells);
117.
118.     echo '<h2>array_key_exists()</h2>';
119.     echo '<p>Checks if an array contains a key.</p>';
120.
121.     function hasKey($key, $arr) {
122.         if (array_key_exists($key, $arr)) {
123.             echo "<p>$key is in that array.</p>";
124.         } else {
125.             echo "<p>$key is not in that array.</p>";
126.         }
127.     }
128.
129.     echo hasKey('hotpink', $colors);
130.     echo hasKey('olivedrab', $spellsArray);
131.
132.
133.     echo '<h2>array_walk()</h2>';
134.
135.     function createButton($color, $text) {
136.         echo "<button style='background-color:$color'
137.             onclick='document.body.style.backgroundColor=\"\$color\";'>
138.             $text</button>";
139.     }
140.
141.     array_walk($colors, createButton);
142.     ?>
-----Lines 143 through 145 Omitted-----

```

Conclusion

Arrays are an important feature of many modern programming languages. In this lesson, we have covered the most common uses of arrays in PHP.

LESSON 5

Working with Databases

Topics Covered

- Object-oriented PHP.
- Database connections.
- phpMyAdmin.
- Querying a database.
- Displaying database records.
- Pagination.
- Sorting records.
- Filtering records.

Evaluation
Copy

Introduction

There are a few different ways of working with databases in PHP, but the most common two are through extensions that ship as part of PHP: the MySQL Improved (*mysqli*) extension and the PHP Data Objects (*PDO*) extension. They work in much the same way. In this course, we will teach the PDO extension as it can be used with any database; whereas *mysqli* just works with MySQL. We will start the lesson with an overview of objects and classes as it's important to understand a bit about object-oriented programming to work with PDO.



5.1. Objects

An object is something that has attributes (properties) and/or behaviors (methods), meaning it *is* certain ways and *does* certain things. In the real world, everything could be considered

an object. Some objects are tangible, like rocks, trees, tennis racquets, and tennis players. And some objects are intangible, like words, colors, tennis swings, and tennis matches.



5.2. Attributes / Properties

If you can say “x is y” or “x has y,” then y is an attribute of x. Some examples:

1. *The rock that he is holding is heavy.* Heavy is an attribute of the specific rock he is holding. More generally, rocks have weight.
2. *The apple tree in our back yard has four branches.* The four branches are attributes of that specific tree. More generally, trees have branches.
3. *Venus Williams’ swing is strong.* Strong is an attribute of Venus Williams’ swing. More generally, tennis swings have a strength.
4. *The final match had three sets.* The three sets are attributes of the specific match. More generally, matches have sets.
5. *Serena Williams has a first-serve percentage of 57.2%.* A 57.2% first-serve percentage is an attribute of Serena Williams. More generally, tennis players have a first-serve percentage.

Attributes are generally nouns (e.g., branches) or adjectives (e.g., heavy). Attributes of objects are called *properties*. In PHP, we could write the statements above like this:

```
$rockHeHolds->weight = 'heavy';  
$backyardAppleTree->branches = [branch1, branch2, branch3, branch4];  
$venus->swing = 'strong';  
$finalMatch->sets = [set1, set2, set3];  
$serena->serve1 = .572;
```



5.3. Behaviors / Methods

If you can say “x does” then does is a behavior of x. Some examples of behaviors:

1. *The rock falls fast.* More generally, rocks can fall.

2. *The apple tree in our back yard* first bore fruit on August 23, 1961. More generally, trees can bear fruit.
3. Venus Williams' *swing hit* the ball. More generally, tennis swings can hit things.
4. The *rocket landed* at 8:17PM, July 20, 1969. More generally, rockets can land.
5. *Serena Williams served*. More generally, tennis players can serve.

Behaviors are verbs and behaviors of objects are called *methods*, which are simply functions defined within a class definition. In PHP, we could write the statements above like this:

```
$rockHeHolds->fall('fast');  
$backyardAppleTree->bearFruit(mktime(0, 0, 0, 8, 23, 1961));  
$venus->swing->hit(ball);  
$rocket->land(mktime(0, 20, 17, 7, 20, 1969));  
$serena->serve();
```



5.4. Classes vs. Objects

A class is a template for an object. An object is an *instance* of a class. When we say Serena Williams is a tennis player, we are saying that Serena Williams is an object of the Tennis Player class. There are other tennis players who have the same attributes and behaviors as Serena Williams, but not in the same way. For example, Serena has a winning percentage. Her sister Venus also has a winning percentage. So do Roger Federer and Rafael Nadal. But their winning percentages are all different. They also all have backhands, but they don't all have the same backhand. Roger Federer has a one-handed backhand, while the others all have two-handed backhands. If you needed to express that in PHP, you could do it this way:

```
$serena->twoHandedBackhand = true;  
$venus->twoHandedBackhand = true;  
$roger->twoHandedBackhand = false;  
$rafa->twoHandedBackhand = true;
```

New objects are created from classes using the `new` keyword, like this:

```
$serena = new TennisPlayer();
```

In the code above, `$serena` would be a new object of the `TennisPlayer` class. `TennisPlayer()` is a *constructor* for initializing new objects. Constructors are similar to methods in that they can take arguments. We will see this with the PDO class.



5.5. Connecting to a Database with PDO

The first step of working with a database is to make the connection. With PDO, the connection is made by initializing a PDO object:

```
$db = new PDO($dsn, $username, $password);
```

1. `$dsn` - the DSN or database source name. Our DSN will be `'mysql:host=localhost;dbname=poetree'`.
2. `$username` - the username. We will be using `'root'`.
3. `$password` - the password. We will be using `'pwdpwd'`. In the real world, you should use a much more secure password.

Given the values we will be using, our code for connecting to the database will be:

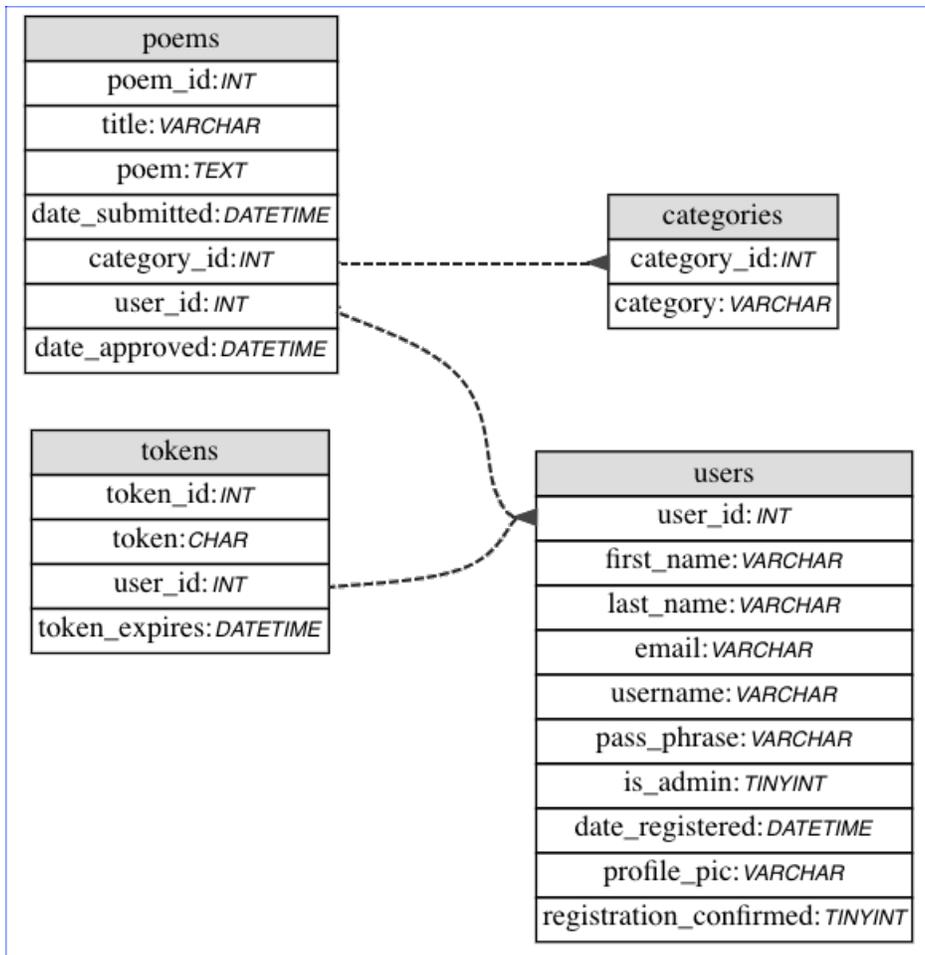
```
$dsn = 'mysql:host=localhost;dbname=poetree';  
$username = 'root';  
$password = 'pwdpwd';  
$db = new PDO($dsn, $username, $password);
```

“localhost” is the name of our local server and “poetree” is the name of the database we’ll be using in class. Before we learn more about PDO, let’s take a look at our database.



5.6. Introducing the Poetree Database

Before we begin querying our database, let’s take a look at how it is structured:



As you can see, the Poetree database consists of four tables:

1. poems
2. categories
3. users
4. tokens

There are three relationships between the tables:

1. Each poem is associated with a category. A category can have 0 or more poems associated with it.
2. Each poem is associated with a user. A user can have 0 or more poems associated with it.

- Each token is associated with a user. A user can have 0 or more tokens associated with it.



5.7. phpMyAdmin

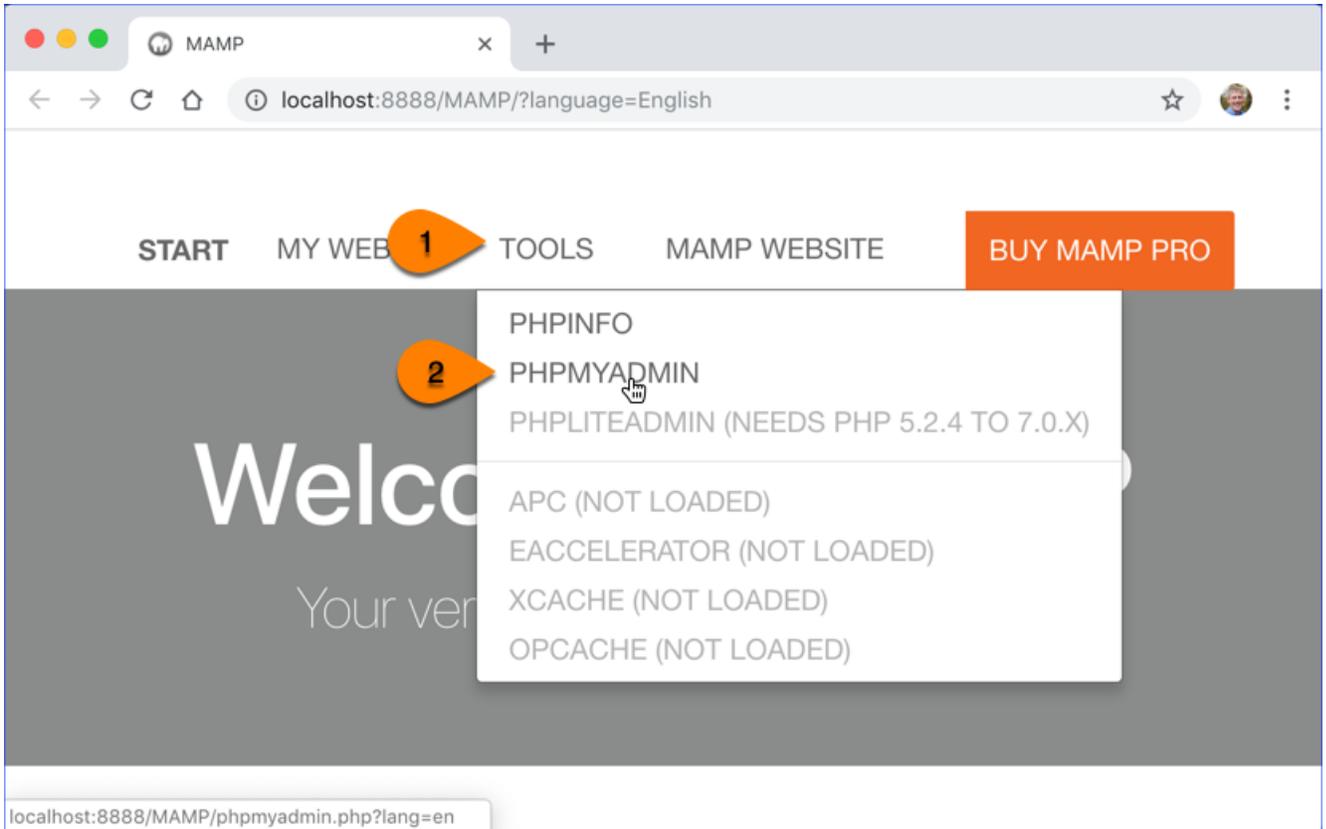
phpMyAdmin¹⁰ is a web-based tool, which comes bundled with MAMP for managing MySQL databases. It is useful for testing and debugging queries. Follow these steps to open phpMyAdmin:

- In MAMP, click **Open WebStart Page**:

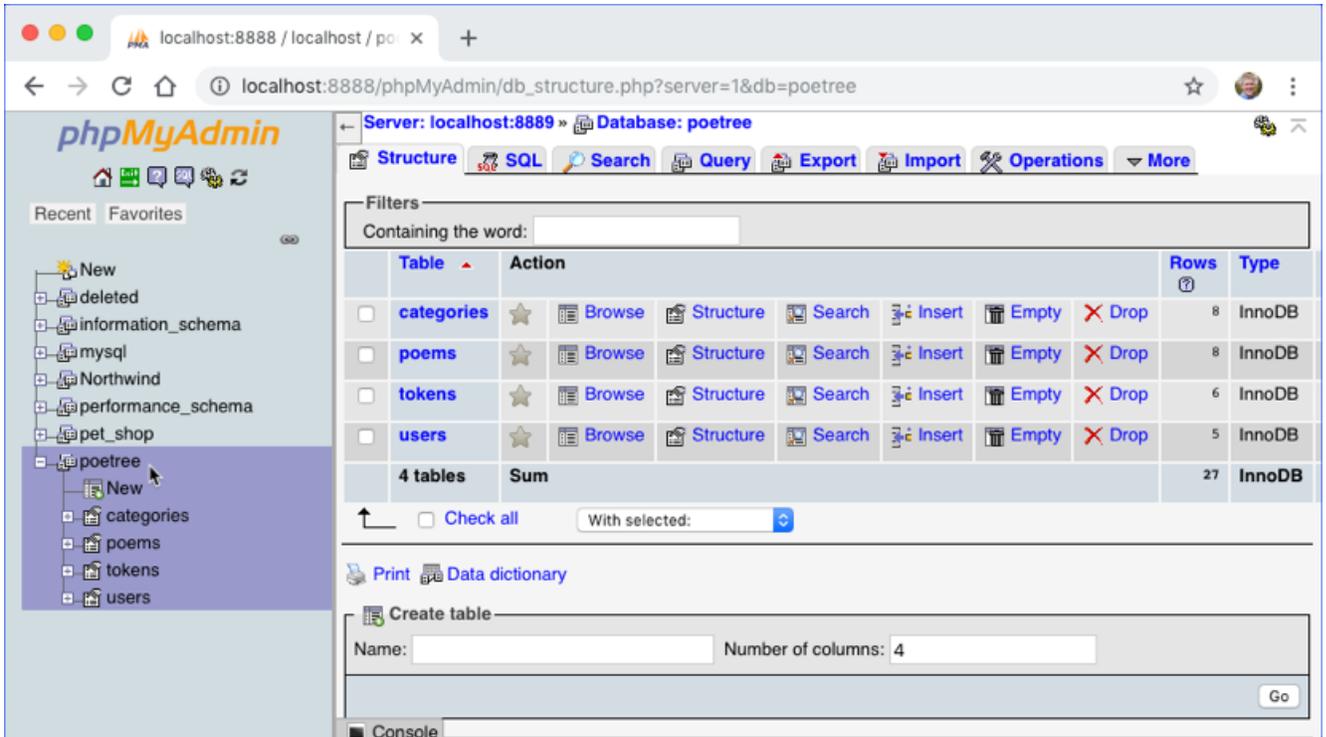


- That will launch the MAMP WebStart page. Click the **PHPMYADMIN** link in the **TOOLS** menu:

10. <https://www.phpmyadmin.net/>

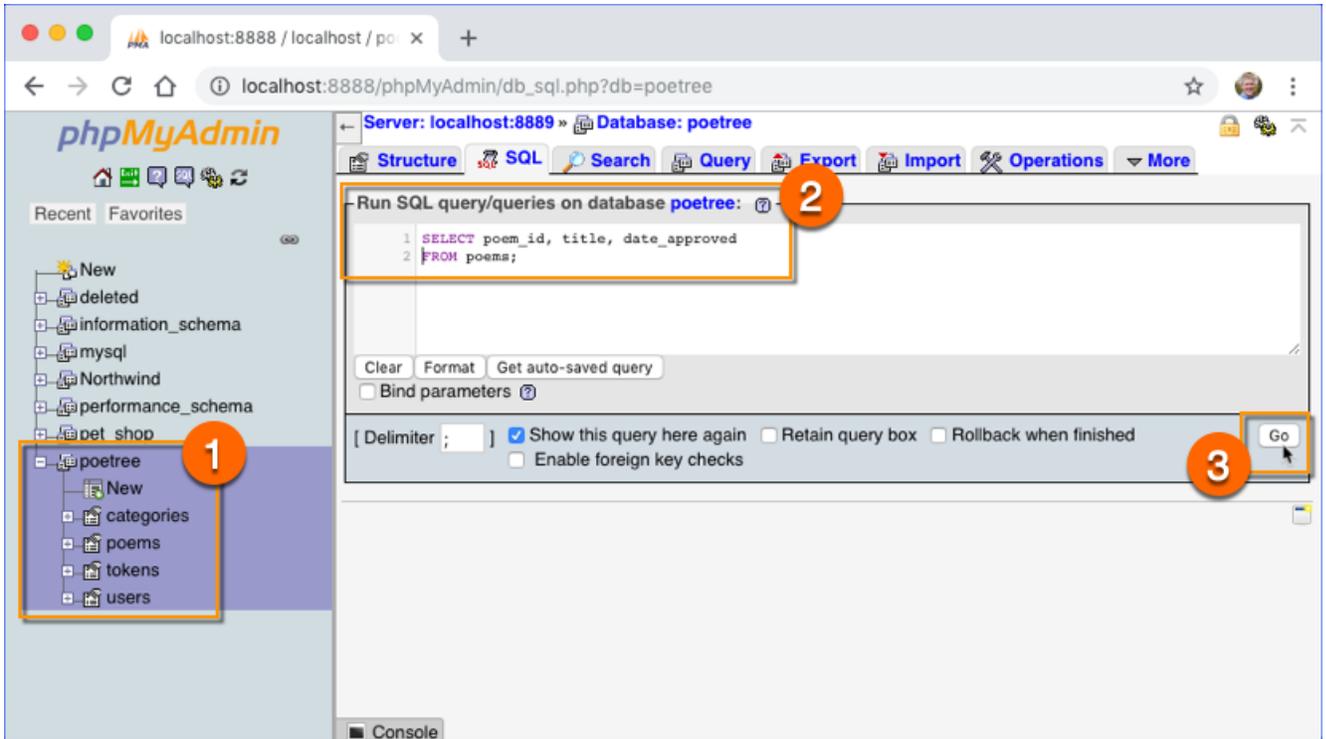


3. The left navigation in phpMyAdmin lists the MySQL databases installed. Click the **poetree** database:



- To run SQL code, click the **SQL** tab, enter your SQL statement(s), and click the **Go** button. For example, the screenshot below shows how to select all records from the **poems** table using the following query:

```
SELECT poem_id, title, date_approved
FROM poems;
```



5. This should return results similar to the following:

The screenshot shows the query results in phpMyAdmin. The table has 8 rows of data. The columns are 'poem_id', 'title', and 'date_approved'. The results are displayed in a table format with options to edit, copy, or delete each row.

poem_id	title	date_approved
1	Banana Duet	2018-01-01 17:25:19
2	The Aristocratic Apple	2018-11-20 15:01:19
3	Carrots and Camels	2019-01-11 18:37:19
4	The Geriatric General	2019-01-11 13:49:19
5	Dancing Dogs in Dungarees	2019-01-11 15:01:19
6	My Innocent Tulip	2018-11-03 22:37:19
7	Harry's Torment - The Villanelle Of The Cocoa	2018-10-11 08:13:19
8	A Bloody Good Goodbye	2018-10-27 15:25:19

While it isn't necessary to use phpMyAdmin to do PHP development, you may find it useful for testing SQL queries and managing data. As you can see from the **Edit**, **Copy**, and

Delete links, in addition to being able to run SQL statements, phpMyAdmin provides a GUI (Graphic User Interface) for managing the data.



5.8. Querying Records with PHP

We have already seen how to connect to the database. Now let's see how we can get data from a table. We'll start with the query we ran in phpMyAdmin to get all the records from the **poems** table:

```
SELECT poem_id, title, date_approved
FROM poems;
```

There are two approaches to running queries with PDO:

1. Use the `query()` method to execute the query directly. An example is below:

```
$query = 'SELECT poem_id, title, date_approved
FROM poems';
$stmt = $db->query($query);
```

This will return a `PDOStatement` object.

2. Use the `prepare()` method to prepare the statement for executing. This returns a new `PDOStatement` object, which you can then execute with the new object's `execute()` method. An example is below:

```
$query = 'SELECT poem_id, title, date_approved
FROM poems';
$stmt = $db->prepare($query);
$stmt->execute();
```

Both the `query()` and the `prepare()` methods return a `PDOStatement` object.

While using `query()` is simpler, using `prepare()` has at least two major advantages:

1. If you decide to re-run the query, it will run faster the second time.

2. You do not have to worry about character escaping or SQL Injection attacks, in which hackers try to attack your database content by passing partial or complete SQL statements through your GET and POST variables.

The second advantage is key and is the reason we recommend using `prepare()`. To illustrate how this works, consider a search form on a website in which a user enters a name to search poem authors. The search input field might be called “title”. And your PHP code to create the query might look like this:

```
$title = $_GET['title'];  
$query = "SELECT title, poem  
        FROM poems  
        WHERE title = '$title'";  
$stmt = $db->query($query);
```

If the searcher enters *Carrots and Camels* or *The Geriatric General*, this will be fine. The resulting query would look something like this:

```
SELECT title, poem  
FROM poems  
WHERE title = 'Carrots and Camels'
```

But if the searcher enters *Harry's Torment* and you don't properly escape the string, the resulting query will look like this:

```
SELECT title, poem  
FROM poems  
WHERE title = 'Harry's Torment'
```

And that apostrophe in *Harry's Torment* will cause the query to fail when we try to execute it.

With `prepare()`, you do not have to worry about escaping the string. You simply replace the unknown value in the query with a question mark, like this:

```
$title = $_GET['title'];  
$query = 'SELECT title, poem  
        FROM poems  
        WHERE title = ?';  
$stmt = $db->prepare($query);
```

And then when you execute the query, you pass in an array of values to replace any question marks in the query. In our example, we only have one:

```
$stmt->execute([$title]);
```

Notice that we do not need to (in fact, we cannot) put the question mark in single quotes. Single quotes are added behind the scenes if necessary based on the data type of the field.

Named Parameters

The query above uses a question mark (?) as a placeholder for the poem title. Each question mark is an *ordered parameter* to be replaced with a value when the query is executed. When we call the `execute()` method, we pass it an array containing one value for each question mark in the order those question marks appear. The query below has two ordered parameters:

```
$query = 'SELECT first_name, last_name
FROM users
WHERE username = ? AND email = ?';
$username = 'HugHerHeart';
$email = 'herheart@phppoetry.com';
$stmt = $db->prepare($query);
$stmt->execute([$username, $email]);
```

Evaluation
Copy

We can also write the query with named parameters by prefixing each with a colon, like this:

```
$query = 'SELECT first_name, last_name
FROM users
WHERE username = :un AND email = :em';
$username = 'HugHerHeart';
$email = 'herheart@phppoetry.com';
$stmt = $db->prepare($query);
```

When we execute this statement, we pass in an associative array with key-value pairs:

```
$stmt->execute(['un'=>$username, 'em'=>$email]);
```

There is no functional difference between using ordered and named parameters. Use whichever you are more comfortable with.

Binding Parameters

An alternative to passing an array of parameters to the statement's `execute()` method is binding the parameters to the statement. This can result in cleaner code, especially when there are many parameters. It works like this:

```
$query = 'SELECT first_name, last_name
FROM users
WHERE username = :un AND email = :em';
$username = 'HugHerHeart';
$email = 'herheart@phppoetry.com';
$stmt = $db->prepare($query);
$stmt->bindParam(':un', $username);
$stmt->bindParam(':em', $email);

$stmt->execute();
```

Notice that in this case nothing is passed to the `execute()` method. For more on `bindParam()` see <https://www.php.net/manual/pdostatement.bindparam.php>.

Fetching the Records

Once you have executed the statement, either using `$db->query()` or `$stmt->execute()`, you can fetch the next row (starting with the first) using the `$stmt->fetch()` method, like this:

```
$title = $_GET['title'];
$query = 'SELECT title, poem
FROM poems
WHERE title = ?';
$stmt = $db->prepare($query);
$stmt->execute([$title]);

$row = $stmt->fetch();
```

`$row` will then contain an array holding values for the keys 'title' and 'poem'. We can output them as part of the HTML like this:

```
<h1><?= $row['title'] ?></h1>
<div><?= nl2br($row['poem']) ?></div>
```

nl2br()

Note that we pass `$row['poem']` to the built-in `nl2br()` function, which replaces all newline characters with `
` tags.

See <https://www.php.net/nl2br> for documentation.

If `$stmt->fetch()` doesn't return any rows (e.g., because the query didn't return any results) then it will return `false` to `$row`. If you then attempt to treat `$row` as an array (e.g., `$row['title']`), you will get an error. To prevent this, you should do a check. For example:

```
<?php if ($row) { ?>
  <h1><?= $row['title'] ?></h1>
  <div><?= nl2br($row['poem']) ?></div>
<?php } else { ?>
  <h1>No Results</h1>
  <p>Sorry, we couldn't find a poem by that name.</p>
<?php } ?>
```

The following demos show how to put all this together. First, an HTML search form:

Demo 5.1: Database/Demos/poem-search.html

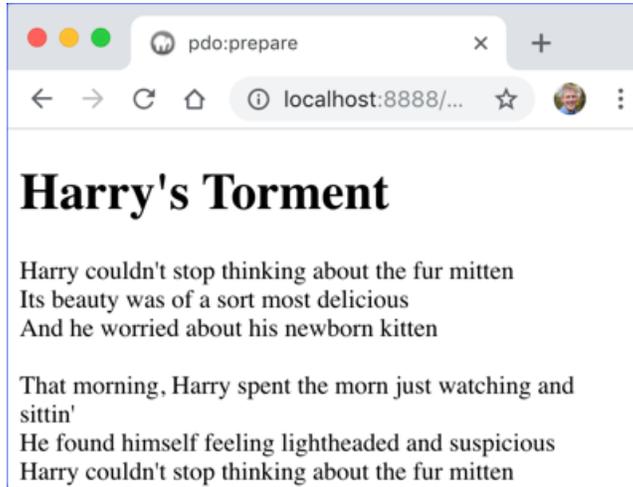
```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Search Poem</title>
9. </head>
10. <body>
11. <main>
12. <form method="get" action="pdo-prepare.php">
13.   <label for="title">Poem Title:</label>
14.   <input type="search" name="title" id="title">
15.   <button class="wide">Search</button>
16. </form>
17. </main>
18. </body>
19. </html>
```

Demo 5.2: Database/Demos/pdo-prepare.php

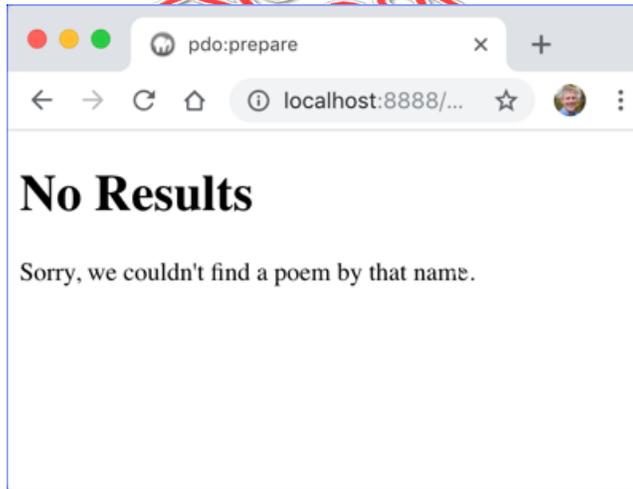
```
1.  <?php
2.    $dsn = 'mysql:host=localhost;dbname=poetree';
3.    $username = 'root';
4.    $password = 'pwdpwd';
5.    $db = new PDO($dsn, $username, $password);
6.    $title = $_GET['title'];
7.    $query = "SELECT title, poem
8.             FROM poems
9.             WHERE title = ?";
10.   $stmt = $db->prepare($query);
11.   $stmt->execute([$title]);
12.   $row = $stmt->fetch();
13.   ?>
14.   <!DOCTYPE html>
15.   <html lang="en">
16.   <head>
17.   <meta charset="UTF-8">
18.   <meta name="viewport" content="width=device-width,initial-scale=1">
19.   <link rel="stylesheet" href="../../static/styles/normalize.css">
20.   <link rel="stylesheet" href="../../static/styles/styles.css">
21.   <title><?= $row['title'] ?></title>
22.   </head>
23.   <body>
24.   <main>
25.   <?php if ($row) { ?>
26.     <h1><?= $row['title'] ?></h1>
27.     <div><?= nl2br($row['poem']) ?></div>
28.   <?php } else { ?>
29.     <h1>No Results</h1>
30.     <p>Sorry, we couldn't find a poem by that name.</p>
31.   <?php } ?>
32.   </main>
33.   </body>
34.   </html>
```

Code Explanation

Open <http://localhost:8888/Webucator/php/Database/Demos/poem-search.html> in your browser and search for *Harry's Torment*. You should get the following result:



Now go back to the form and search on some random string (e.g., *foobar*). You should get the following result:



Exercise 14: Creating a Single Poem Page

 20 to 30 minutes

In this exercise, you will create the `poem.php` page of the PHP Poetry website.

1. Open `Database/Exercises/phppoetry.com/poem.php` in your editor. Currently, the page always outputs the same poem.
2. Edit the page so that it expects a `poem-id` parameter to be passed on the query string and uses it to get the poem data from the database. You will need to get the following fields from the database:
 - A. `poems.title`
 - B. `poems.poem`
 - C. `poems.date_submitted`
 - D. `poems.date_approved`
 - E. `users.username`
3. Using the data returned from the database, edit the page so that the content highlighted below is dynamic (i.e., based on the `poem-id`):

The screenshot shows a web browser window with the address bar displaying `localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poem.php`. The website header features the logo "The Poet Tree Club" with the tagline "Set your poems free..." and navigation links for "Home", "Poems", "Submit Poem", "My Account", and "Contact us". The main content area displays the poem "Dancing Dogs in Dungarees" by LimerickMan, submitted on 01/11/2019. The poem text is: "A dozen dancing dogs in dungarees / All of the most distinguished pedigrees / Held their heads high by day / By night, they'd bow them and pray / For freedom from filthy frolicking fleas". Below the poem are three links: "More Funny Poems", "More Poems by LimerickMan", and "All Poems". The footer contains copyright information for 2019 and links for "Log out", "Admin", and "About us".

4. Visit `http://localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poem.php?poem-id=1` to test your solution. Then try passing in other values for `poem-id`.

Look in `Database/Exercises/phppoetry.com/sql.txt` if you need help with the SQL query.

Making use of php.net

To complete this exercise, you will likely need to review two functions on `https://www.php.net`:

1. `date()` - This function is used to format a date or a time.

2. `strtotime()` - This function converts a string (like the ones returned from the database for the `date_published` and `date_approved` fields) to an integer representing the number of seconds since the epoch.

Solution: Database/Solutions/phppoetry.com/poem.php

```
1.  <?php
2.      $dsn = 'mysql:host=localhost;dbname=poetree';
3.      $username = 'root';
4.      $password = 'pwdpwd';
5.      $db = new PDO($dsn, $username, $password);
6.      $poemId = $_GET['poem-id'];
7.      $query = "SELECT u.username,
8.          p.title, p.poem, p.date_submitted, p.date_approved
9.          FROM users u
10.         JOIN poems p ON u.user_id = p.user_id
11.         WHERE p.poem_id = ?";
12.      $stmt = $db->prepare($query);
13.      $stmt->execute([$poemId]);
14.      $row = $stmt->fetch();
15.
16.      if ($row) {
17.          $title = $row['title'];
18.          $authorUserName = $row['username'];
19.          $dateSubmitted = $row['date_submitted'];
20.          $dateApproved = $row['date_approved'];
21.          $poem = $row['poem'];
22.      } else {
23.          $title = 'Poem Not Found';
24.      }
25.
26.      $pageTitle = $title;
27.      require 'includes/header.php';
28.  ?>
29.  <main id="poem">
30.      <h1><?= $title ?></h1>
31.      <?php if ($row) { ?>
32.          <div id="submission-status">
33.              Submitted on <?= date('m/d/Y', strtotime($dateSubmitted)) ?>
34.              at <?= date('g:iA', strtotime($dateSubmitted)) ?>
35.              by <?= $authorUserName ?>
36.              <a href="#">Edit</a>
37.              <a href="#">Delete</a>
38.          </div>
39.          <div id="approval-status">
40.              Approved: <?= date('m/d/Y', strtotime($dateApproved)) ?>
41.          </div>
42.          <article class="poem">
43.              <?= nl2br($poem) ?>
44.          </article>
```

```
45.     <?php } else { ?>
46.         <p>Sorry, we couldn't find the poem you're looking for.</p>
47.     <?php } ?>
-----Lines 48 through 67 Omitted-----
```

Code Explanation

After fetching `$row`, we check to see if it returned a truthy value (an array). If it did, we set our variables accordingly. If it did not, we only set the `$title` variable, which we set to 'Poem Not Found'.

After the if condition, we set `$pageTitle`, which is used in `header.php` as part of the HTML title.

In the `body`, we again check `$row`, and based on its value, either output the poem data or a friendly message saying we couldn't find that poem.

*

5.9. Queries Returning Multiple Rows

When we queried the **poems** table using a `poem_id`, we knew that we would get at most one row, because `poem_ids` are unique. Now we will look at how we can get and output multiple poems. The following query, for example, will return all poem titles and their categories:

```
SELECT p.title, c.category
FROM poems p
JOIN categories c ON c.category_id = p.category_id
```

Assuming the query returns results, the `PDOStatement` object's `fetch()` method returns a row as an array with keys 'title' and 'category'. Each subsequent time `fetch()` is called, it gets the next row until there are no rows left, at which point it returns `false`.

Because `fetch()` returns `false` when there are no more rows to fetch, we can use a while loop to iterate through the results, checking the value returned from `fetch()` at the same time as we assign that value to `$row`, like this:

```
while ($row = $stmt->fetch()) {
    echo $row['title'] . ': ' . $row['category'] . '<br>';
}
```

The following demo puts this together:

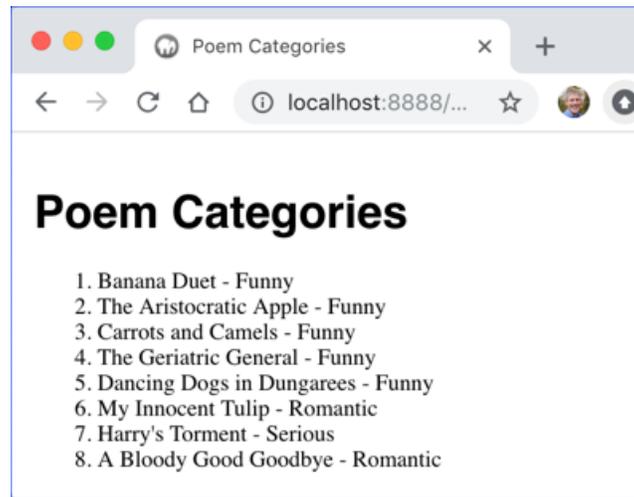
Demo 5.3: Database/Demos/poem-categories.php

```
1.  <?php
2.      $dsn = 'mysql:host=localhost;dbname=poetree';
3.      $username = 'root';
4.      $password = 'pwdpwd';
5.      $db = new PDO($dsn, $username, $password);
6.      $query = "SELECT p.title, c.category
7.          FROM poems p
8.          JOIN categories c ON c.category_id = p.category_id";
9.      $stmt = $db->prepare($query);
10.     $stmt->execute();
11.     ?>
12.     <!DOCTYPE html>
13.     <html lang="en">
14.     <head>
15.     <meta charset="UTF-8">
16.     <meta name="viewport" content="width=device-width,initial-scale=1">
17.     <link rel="stylesheet" href="../../static/styles/normalize.css">
18.     <link rel="stylesheet" href="../../static/styles/styles.css">
19.     <title>Poem Categories</title>
20.     </head>
21.     <body>
22.     <main>
23.         <h1>Poem Categories</h1>
24.         <ol>
25.             <?php
26.                 while ($row = $stmt->fetch()) {
27.                     echo '<li>' . $row['title'] . ' - '
28.                         . $row['category'] . '</li>';
29.                 }
30.             ?>
31.         </ol>
32.     </main>
33. </body>
34. </html>
```



Code Explanation

Run this code and you should see the following:



You've Got Skills!

You have now learned the basic PHP skills for building database-driven applications. In the following series of exercises, you will create a sortable table of poems that you can filter and paginate through.



Exercise 15: Creating the Poems Listings

🕒 25 to 40 minutes

In this exercise, you will create the poem listing on the home page and the `poems.php` page of the PHP Poetry website.

1. Open `Database/Exercises/phppoetry.com/poems.php` in your editor. Currently, the page outputs a table with two rows of static data.
2. Edit the page so that it outputs all poems that have been approved, beginning with those most recently approved. You will need to get the following fields from the database:
 - A. `poems.poem_id` - used to pass a value to `poem.php` when the poem title is clicked.
 - B. `poems.title` - the poem title.
 - C. `poems.date_approved` - the published date.
 - D. `categories.category` - the category.
 - E. `users.username` - the author's username.
3. Be sure to make the links work, so that `poem.php` shows the correct poem.
4. Visit `http://localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poems.php` when you are done. It should look something like this:

Poems | The Poet Tree Club

localhost:8888/Webucator/php/Database/Solutions/phppoetry.com/poems.php

[Home](#) [Poems](#) [Submit Poem](#) [My Account](#) [Contact us](#)

The Poet Tree Club

Set your poems free...



Poems

Total Poems: 8

Poem	Category	Author	Published
Carrots and Camels	Funny	LimerickMan	01/11/2019
Dancing Dogs in Dungarees	Funny	LimerickMan	01/11/2019
The Geriatric General	Funny	LimerickMan	01/11/2019
Harry's Torment	Serious	Dawnable	11/26/2018
My Innocent Tulip	Romantic	HugHerHeart	11/03/2018
A Bloody Good Goodbye	Romantic	HugHerHeart	10/27/2018
Banana Duet	Funny	LimerickMan	01/01/2018

Previous Next

- Be sure to test the poem title links.
- Now open `Database/Exercises/phppoetry.com/index.php` in your editor. Modify this table so that it gets and shows the latest three poems published. To limit the records returned in a MySQL query, add the following clause after the `ORDER BY` clause:

```
LIMIT 0, 3
```

- `0` is the *offset*, meaning the number of records to skip. In this case, we are starting with the first record.
- `3` is the number of rows to return.

Look in `Database/Exercises/phppoetry.com/sql.txt` if you need help with the SQL query.

Challenge

The number of total poems shown is static. You can get the actual number by writing a new query using the same WHERE conditions, but selecting a COUNT. See if you can make the page show the actual count.

Solution: Database/Solutions/phppoetry.com/poems.php

```
1. <?php
2.     $pageTitle = 'Poems';
3.     require 'includes/header.php';
4.
5.     $dsn = 'mysql:host=localhost;dbname=poetree';
6.     $username = 'root';
7.     $password = 'pwdpwd';
8.     $db = new PDO($dsn, $username, $password);
9.     $query = "SELECT p.poem_id, p.title, p.date_approved,
10.    c.category, u.username
11.           FROM poems p
12.           JOIN categories c ON c.category_id = p.category_id
13.           JOIN users u ON u.user_id = p.user_id
14.           WHERE p.date_approved IS NOT NULL
15.           ORDER BY p.date_approved DESC";
16.     $stmt = $db->prepare($query);
17.     $stmt->execute();
18.     ?>
19. <main id="poems">
20.     <h1><?= $pageTitle ?></h1>
21.     <table>
22.         <caption>Total Poems: 8</caption>
23.         <thead>
24.             <tr>
25.                 <th>Poem</th>
26.                 <th>Category</th>
27.                 <th>Author</th>
28.                 <th>Published</th>
29.             </tr>
30.         </thead>
31.         <tbody>
32.             <?php
33.                 while ($row = $stmt->fetch()) {
34.                     $approved = strtotime($row['date_approved']);
35.                     $published = date('m/d/Y', $approved);
36.                 ?>
37.                 <tr class="normal">
38.                     <td>
39.                         <a href="poem.php?poem-id=<?= $row['poem_id'] ?>">
40.                             <?= $row['title'] ?>
41.                         </a>
42.                     </td>
43.                     <td><?= $row['category'] ?></td>
44.                     <td><?= $row['username'] ?></td>
```

Evaluation
Copy

```
45.         <td><?= $published ?></td>
46.         </tr>
47.         <?php } ?>
48.     </tbody>
-----Lines 49 through 78 Omitted-----
```

Code Explanation

Notice that we chose to put the start of the while loop in one PHP block and the end in a separate PHP block. If we had put them in the same PHP block, we would have had to create the table row using a lot of concatenation. While that's perfectly valid, it can get a little messy.

Solution: Database/Solutions/phppoetry.com/index.php

```
1.  <?php
2.      require 'includes/header.php';
3.
4.      $dsn = 'mysql:host=localhost;dbname=poetree';
5.      $username = 'root';
6.      $password = 'pwdpwd';
7.      $db = new PDO($dsn, $username, $password);
8.      $query = "SELECT p.poem_id, p.title, p.date_approved,
9.      c.category, u.username
10.         FROM poems p
11.         JOIN categories c ON c.category_id = p.category_id
12.         JOIN users u ON u.user_id = p.user_id
13.         WHERE p.date_approved IS NOT NULL
14.         ORDER BY p.date_approved DESC
15.         LIMIT 0, 3";
16.      $stmt = $db->prepare($query);
17.      $stmt->execute();
18.  ?>
19.  <main>
20.  <h1>Latest Poems</h1>
21.  <table>
22.      <thead>
23.          <tr>
24.              <th>Poem</th>
25.              <th>Category</th>
26.              <th>Author</th>
27.              <th>Published</th>
28.          </tr>
29.      </thead>
30.      <tbody>
31.          <?php
32.              while ($row = $stmt->fetch()) {
33.                  $approved = strtotime($row['date_approved']);
34.                  $published = date('m/d/Y', $approved);
35.          ?>
36.          <tr>
37.              <td>
38.                  <a href="poem.php?poem-id=?= $row['poem_id'] ?>">
39.                      <?= $row['title'] ?>
40.                  </a>
41.              </td>
42.              <td><?= $row['category'] ?></td>
43.              <td><?= $row['username'] ?></td>
44.              <td><?= $published ?></td>
```

Evaluation
Copy

```
45.         </tr>
46.         <?php } ?>
47.     </tbody>
-----Lines 48 through 57 Omitted-----
```

Challenge Solution:

Database/Solutions/phppoetry.com/poems-with-count.php

```
1.  <?php
2.      $pageTitle = 'Poems';
3.      require 'includes/header.php';
4.
5.      $dsn = 'mysql:host=localhost;dbname=poetree';
6.      $username = 'root';
7.      $password = 'pwdpwd';
8.      $db = new PDO($dsn, $username, $password);
9.      $query = "SELECT p.poem_id, p.title, p.date_approved,
10.     c.category, u.username
11.         FROM poems p
12.         JOIN categories c ON c.category_id = p.category_id
13.         JOIN users u ON u.user_id = p.user_id
14.         WHERE p.date_approved IS NOT NULL
15.         ORDER BY p.date_approved DESC";
16.     $stmt = $db->prepare($query);
17.     $stmt->execute();
18.
19.     $qPoemCount = "SELECT COUNT(p.poem_id) AS num
20.     FROM poems p
21.         JOIN categories c ON c.category_id = p.category_id
22.         JOIN users u ON u.user_id = p.user_id
23.         WHERE p.date_approved IS NOT NULL";
24.
25.     $stmtPoemCount = $db->prepare($qPoemCount);
26.     $stmtPoemCount->execute();
27.     $poemCount = $stmtPoemCount->fetch()['num'];
28.     ?>
29.     <main id="poems">
30.         <h1><?= $pageTitle ?></h1>
31.         <table>
32.             <caption>Total Poems: <?= $poemCount ?></caption>
-----Lines 33 through 88 Omitted-----
```

Code Explanation

Note the following line of code:

```
$poemCount = $stmtPoemCount->fetch()['num'];
```

We could break that into two lines for clarity:

```
$poemCountRow = $stmtPoemCount->fetch();  
$poemCount = $poemCountRow['num'];
```

But when a method returns an array that is only going to be used one time, it is common to *chain* the lookup onto the method.

Exercise 16: Adding Pagination

 40 to 60 minutes

In this exercise, you will limit the number of poems shown at one time and make it possible to *paginate* through the results with **Previous** and **Next** links.

1. Open `Database/Exercises/phppoetry.com/poems.php` in your editor if it isn't still open. If you didn't do the challenge in the last exercise to get the correct poem count, you should replace your code with the code in `Database/Solutions/phppoetry.com/poems-with-count.php`.
2. Notice that there are "Previous" and "Next" placeholders in table data cells within `tfoot`.
3. Edit the page so that only two poems show up in the table. Normally, you would show a larger number (e.g., 10), but we will use two to demonstrate.
4. Unless the last poem is already showing up, make the word "Next" link to the same page, but show the next two poems by passing a new **offset** value on the query string. You will need to create a new variable to hold the next offset value, which should equal the current offset value **plus** the number of rows being shown.

- A. If the last poem is already showing up, the word "Next" should not be linked and the table data cell should get the "disabled" class, like this:

```
<td class="disabled">Next</td>
```

5. Unless the first poem is already showing up, make the word "Previous" link to the same page, but show the previous two poems by passing a new **offset** value on the query string. You will need to create a new variable to hold the previous offset value, which should equal the current offset value **minus** the number of rows being shown.

- A. If the first poem is already showing up, the word "Previous" should not be linked and the table data cell should get the "disabled" class, like this:

```
<td class="disabled">Previous</td>
```

6. Visit `http://localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poems.php` when you are done. On first loading, the page should look like this:

The Poet Tree Club
Set your poems free...

Home Poems Submit Poem My Account Contact us

Poems

Total Poems: 7

Poem	Category	Author	Published
Carrots and Camels	Funny	LimerickMan	01/11/2019
Dancing Dogs in Dungarees	Funny	LimerickMan	01/11/2019

Previous [Next](#)

Notice “Previous” is not linked and “Next” is.

7. Clicking the **Next** link should take you to `poems.php?offset=2`, which will show two new poems. On that page, both the **Previous** and **Next** links should be active.
8. Clicking the **Previous** link should take you to `poems.php?offset=0`. As we are again at the start of the poems, only the **Next** link should be active.
9. Clicking **Next** several times should take you to the end, at which point only the **Previous** link should be active.

Look in `Database/Exercises/phppoetry.com/sql.txt` if you need help with the SQL query.

Solution: Database/Solutions/phppoetry.com/poems-pagination.php

```
1. <?php
2.     $pageTitle = 'Poems';
3.     require 'includes/header.php';
4.
5.     $offset = $_GET['offset'] ?? 0;
6.     $offset = (int) $offset; // So we can use === later in the code
7.     $rowsToShow = 2;
8.     $dsn = 'mysql:host=localhost;dbname=poetree';
9.     $username = 'root';
10.    $password = 'pwdpwd';
11.    $db = new PDO($dsn, $username, $password);
12.    $query = "SELECT p.poem_id, p.title, p.date_approved,
13.    c.category, u.username
14.    FROM poems p
15.    JOIN categories c ON c.category_id = p.category_id
16.    JOIN users u ON u.user_id = p.user_id
17.    WHERE p.date_approved IS NOT NULL
18.    ORDER BY p.date_approved DESC
19.    LIMIT $offset, $rowsToShow";
20.    $stmt = $db->prepare($query);
21.    $stmt->execute();
22.
23.    $qPoemCount = "SELECT COUNT(p.poem_id) AS num
24.    FROM poems p
25.    JOIN categories c ON c.category_id = p.category_id
26.    JOIN users u ON u.user_id = p.user_id
27.    WHERE p.date_approved IS NOT NULL";
28.
29.    $stmtPoemCount = $db->prepare($qPoemCount);
30.    $stmtPoemCount->execute();
31.    $poemCount = $stmtPoemCount->fetch()['num'];
32.
33.    $prevOffset = max($offset - $rowsToShow, 0);
34.    $nextOffset = $offset + $rowsToShow;
35.
36.    $href = "poems-pagination.php?"; // Will be poems.php? for you.
37.    $prev = $href . "offset=$prevOffset";
38.    $next = $href . "offset=$nextOffset";
39.    ?>
40.    <main id="poems">
41.        <h1><?= $pageTitle ?></h1>
42.        <table>
43.            -----Lines 43 through 69 Omitted-----
44.
45.        <tfoot class="pagination">
```

```

71.     <tr>
72.         <?php
73.             if ($offset === 0) {
74.                 echo "<td class='disabled'>Previous</td>";
75.             } else {
76.                 echo "<td><a href='$prev'>Previous</a></td>";
77.             }
78.         ?>
79.     <td colspan="2"></td>
80.     <?php
81.         if ($nextOffset >= $poemCount) {
82.             echo "<td class='disabled'>Next</td>";
83.         } else {
84.             echo "<td><a href='$next'>Next</a></td>";
85.         }
86.     ?>
87. </tr>
88. </tfoot>
89. </table>

```

-----Lines 90 through 111 Omitted-----

Code Explanation

Things to notice:

1. We use the null coalescing operator to set `$offset` to the value of the `offset` parameter on the query string, or to `0` if that `offset` parameter is not set.
 2. We cast `$offset` to an integer as values coming in on the query string are always strings.
 3. We set `$rowsToShow` to 2. Try changing that to see how it works with different numbers of rows.
 4. We add a `LIMIT` clause to our `SELECT` statement using `$offset` and `$rowsToShow`.
 5. We set values for `$prevOffset` and `$nextOffset` and use them to create the `$prev` and `$next` variables, which hold the URLs for the **Previous** and **Next** links.
 6. If the current value of `$offset` is `0` then we disable the **Previous** link.
 7. If the value of `$nextOffset` is greater than the number of poems (`$poemCount`) then we disable the **Next** link.
-

Exercise 17: Sorting

 45 to 75 minutes

In this exercise, you will make the table sortable by turning the four table headers (**Poem**, **Category**, **Author**, and **Published**) into links. When the user clicks one of these headings, the results should sort on that header. The next time the user clicks a header, the results should sort in reverse order.

Warning: This exercise is much more complex than anything we have done thus far, but it doesn't require any knowledge of PHP code beyond what you have learned. Take your time and move through each step slowly. If you get stuck, you are welcome to peek at the solution for help.

1. Open `Database/Exercises/phppoetry.com/poems.php` in your editor if it isn't still open.
2. You are going to have to use variables in the `ORDER BY` clause of the SQL query for both the field you want to sort on and the direction of the sort (i.e., 'asc' or 'desc'). Name those variables `$order` and `$dir`. Values for these can be passed in on the query string (call those URL parameters `order` and `dir`), but they might not be. You should start by writing code that:
 - A. Sets `$order` to the value of the `order` URL parameter if it is set, and otherwise sets it to 'date_approved'.
 - B. Sets `$dir` to the value of the `dir` URL parameter if it is set, and otherwise sets it to 'desc'.
3. Next, change the `ORDER BY` clause to use the new `$order` and `$dir` variables.
4. When users click the **Previous** and **Next** links, you will need to keep track of the sorting, so, in addition to the `offset` parameter, which is already being passed, pass values for the `order` and `dir` parameters on the query string for `$prev` and `$next`.
5. Next you will need to construct the links for the headers. First consider the logic. If the poems are already sorted by `date_approved asc` and the user clicks the **Published** heading, you want to switch the sort to `date_approved desc`. But if the user clicks any other header, you will sort in ascending order by that header. So, the first step is to set variables for the direction for each of the four headers with a default of 'asc'. Good variable names would be `$dirTitle`, `$dirCategory`, etc.

6. If the last sort was done in *descending* order, then the next sort will be in *ascending* order, no matter which header is clicked. But if the last sort was done in *ascending* order, then the next sort will be done in *descending* order if the same header is clicked again. So, you need to check if the last sort was done in ascending order. If it was, you need to find out which header was clicked and change the value of the direction variable for that header (e.g., `$dirTitle`) to 'desc'.
7. Next you need to construct the href values for the headers and assign them to variables. Good variable names would be `$linkTitle`, `$linkCategory`, etc. Remember that you already have declared a `$href` variable to hold "poems.php?" You just need to append the query string on to that to pass the order field and direction for each header.
8. Finally, you need to turn the headers into links using the href values you constructed in the last step.
9. Visit <http://localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poems.php> when you are done to test your solution.

Solution: Database/Solutions/phppoetry.com/poems-sorting.php

```
-----Lines 1 through 13 Omitted-----
14. // Set defaults for $order and $dir
15. $order = $_GET['order'] ?? 'date_approved';
16. $dir = $_GET['dir'] ?? 'desc';
17. $query = "SELECT p.poem_id, p.title, p.date_approved,
18. c.category, u.username
19. FROM poems p
20. JOIN categories c ON c.category_id = p.category_id
21. JOIN users u ON u.user_id = p.user_id
22. WHERE p.date_approved IS NOT NULL
23. ORDER BY $order $dir
24. LIMIT $offset, $rowsToShow";
25. $stmt = $db->prepare($query);
26. $stmt->execute();

-----Lines 27 through 40 Omitted-----
41. $href = "poems-sorting.php?"; // Will be poems.php? for you.
42. $prev = $href . "offset=$prevOffset&order=$order&dir=$dir";
43. $next = $href . "offset=$nextOffset&order=$order&dir=$dir";
44.
45. /* CONSTRUCT THE LINKS FOR THE HEADERS */
46.
47. // Default all directions to ascending
48. $dirTitle = 'asc';
49. $dirCategory = 'asc';
50. $dirUsername = 'asc';
51. $dirPublished = 'asc';
52.
53. // If the current direction is 'asc', switch the direction
54. // for the header that is currently being sorted on
55. if ($dir === 'asc') {
56.     switch ($order) {
57.         case 'title':
58.             $dirTitle = 'desc';
59.             break;
60.         case 'category':
61.             $dirCategory = 'desc';
62.             break;
63.         case 'username':
64.             $dirUsername = 'desc';
65.             break;
66.         case 'date_approved':
67.             $dirPublished = 'desc';
68.             break;

```

```

69.     }
70.   }
71.
72.   $titleLink = $href . "order=title&dir=$dirTitle";
73.   $categoryLink = $href . "order=category&dir=$dirCategory";
74.   $usernameLink = $href . "order=username&dir=$dirUsername";
75.   $publishedLink = $href . "order=date_approved&dir=$dirPublished";
76. ?>
77. <main id="poems">
78.   <h1><?= $pageTitle ?></h1>
79.   <table>
80.     <caption>Total Poems: <?= $poemCount ?></caption>
81.     <thead>
82.       <tr>
83.         <th>
84.           <a href="<?= $titleLink ?>">Poem</a>
85.         </th>
86.         <th>
87.           <a href="<?= $categoryLink ?>">Category</a>
88.         </th>
89.         <th>
90.           <a href="<?= $usernameLink ?>">Author</a>
91.         </th>
92.         <th>
93.           <a href="<?= $publishedLink ?>">Published</a>
94.         </th>
95.       </tr>
96.     </thead>

```

-----Lines 97 through 156 Omitted-----



5.10. Anticipating Foul Play

Web developers always need to be aware that users won't always do what they expect them to do, and sometimes they will intentionally try to break things. One method of doing that is to pass in unexpected URL or POST parameters. We can prevent this from doing any harm by creating an array of acceptable values and checking to make sure the passed-in value is one of those values. Examine the code below:

Demo 5.4: Database/Solutions/phppoetry.com/poems-sorting-2.php

```
-----Lines 1 through 8 Omitted-----
9.     $order = $_GET['order'] ?? 'date_approved';
10.    $orderAllowed = ['date_approved',
11.                    'title',
12.                    'category',
13.                    'username'];
14.    if (!in_array($order, $orderAllowed)) {
15.        $order = 'date_approved';
16.    }
17.
18.    $dir = $_GET['dir'] ?? 'desc';
19.    $dirAllowed = ['asc', 'desc'];
20.    if (!in_array($dir, $dirAllowed)) {
21.        $dir = 'asc';
22.    }
-----Lines 23 through 167 Omitted-----
```



Code Explanation

By adding the code shown above checking that the passed-in values on the `order` and `dir` URL parameters are in our list of acceptable values, we ensure that the values of `$order` and `$dir` are valid. Take a moment to add this code into your `poems.php` file.

Exercise 18: Filtering

 45 to 75 minutes

In this exercise, you will add filtering to the results.

1. Open `Database/Exercises/phppoetry.com/poems.php` in your editor if it isn't still open.
2. We will add the ability to filter on two fields: **category** and **username**. The first step is to populate the form correctly. The form will submit to the same page (`poems.php`) using the GET method, so the form name-value pairs will be passed on the query string, just as occurs with the **Previous** and **Next** links.
 - A. We must remember the `order` and `dir` values, which may have changed from the defaults based on any sorting the user has done. Add two hidden fields to the form that hold those values.
 - B. Currently, the options for the `cat` (category) and `user` (username) select fields are hard coded. Those should be populated based on data from the database so that if the categories change or if a new user publishes a poem, the form reflects that. Write code to retrieve from the database:
 - i. The categories, category ids, and number of published poems in that category.
 - ii. The usernames and user ids of people who have published poems, and the number of poems each user has published.
 - C. Use those results to dynamically populate the `cat` and `user` options.
 - D. If the form had previously been submitted, then you want the category and/or username that was submitted the previous time to be pre-selected. Add the `selected` attribute to the category and/or username options that were submitted via the form.
3. Before moving on to the next step, test this to make sure it's working by visiting `http://localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poems.php`.

- A. The categories and username fields should be populated. View the source of the page. The form code should look something like this:

```
<form method="get" action="poems-filtering.php">
  <input type="hidden" name="order" value="date_approved">
  <input type="hidden" name="dir" value="desc">
  <label for="cat">Category:</label>
  <select name="cat" id="cat">
    <option value="0">All</option>
    <option value='2'>Funny (4)</option>
    <option value='1'>Romantic (2)</option>
    <option value='4'>Serious (1)</option>
  </select>
  <label for="user">Author:</label>
  <select name="user" id="user">
    <option value="0">All</option>
    <option value='3'>Dawnable (1)</option>
    <option value='2'>HugHerHeart (2)</option>
    <option value='1'>LimerickMan (5)</option>
  </select>
  <button name="filter" class="wide">Filter</button>
</form>
```

Don't worry if the format of your code is a little messy. Dynamically generated code is generally messy.

- B. Select a category and submit. The records won't change yet, but the category you selected should still be selected. If it is not, go back to your code and fix it.
- C. Select a username and submit. The records won't change yet, but the username you selected should still be selected. If it is not, go back to your code and fix it.
4. Next, we need to modify the WHERE clause of the poems query to take into account the filtering.

- A. Break \$query into two parts like this:

```
$query = "SELECT p.poem_id, p.title, p.date_approved,  
c.category, u.username  
FROM poems p  
JOIN categories c ON c.category_id = p.category_id  
JOIN users u ON u.user_id = p.user_id  
WHERE p.date_approved IS NOT NULL";  
  
/*  
  You will add code here to append to the WHERE clause  
  if a category and/or username has been selected.  
*/  
  
// Concatenate on the rest of the query  
$query .= " ORDER BY $order $dir  
LIMIT $offset, $rowsToShow";
```

- B. If the user selected a category and/or username, we need to get the selected category ID and/or selected user id. Declare two variables: \$selCatId and \$selUserId, and set them to the values of the cat and user parameters passed on the query string, if they are passed, or to 0 if they are not passed. Hint: this is a good opportunity to use the null coalescing operator.
- C. We now need to create the additional conditions to append to the WHERE clause. We will do this by creating two arrays: \$whereConditions and \$params. The items in \$whereConditions will be strings structured like this: "field_name = ?". The items in \$params will be the values that will replace the question mark placeholders in the where conditions.
- i. Create a new variable called \$whereConditions and assign it an empty array.
 - ii. Create a new variable called \$params and assign it an empty array.
 - iii. If a category was selected (i.e., \$selCatId is not 0), append "c.category_id = ?" to \$whereConditions and append \$selCatId to \$params.
 - iv. If a username was selected (i.e., \$selUserId is not 0), append "u.user_id = ?" to \$whereConditions and append \$selUserId to \$params.

- v. Now, if and only if `$whereConditions` contains items, we need to append the where conditions on to the end of the `WHERE` clause. To do that, we need to join them on the string “ AND ” and then concatenate the result to `$query` right before the `ORDER BY` clause is concatenated on. To do this, use the `implode()` function. If you need help with this part, open `Database/Demos/where-conditions.php` in your editor and study it. Then open the same file in your browser¹¹ to see the output.
- vi. When executing the statement with this query, pass in `$params`.
- vii. Finally, you will need to add the additional where conditions to `$qPoemCount` as well so that the poem count is correct. Do this in the same way. Note that you do not need to recreate the `$whereConditions` and `$params` arrays. Reuse the ones you have already created.

5. Visit <http://localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poems.php> when you are done to test your solution.

Look in `Database/Exercises/phppoetry.com/sql.txt` if you need help with the SQL query.

11. <http://localhost:8888/Webucator/php/Database/Demos/where-conditions.php>

Solution: Database/Solutions/phppoetry.com/poems-filtering.php

```
-----Lines 1 through 27 Omitted-----
28.     $query = "SELECT p.poem_id, p.title, p.date_approved,
29.     c.category, u.username
30.         FROM poems p
31.         JOIN categories c ON c.category_id = p.category_id
32.         JOIN users u ON u.user_id = p.user_id
33.         WHERE p.date_approved IS NOT NULL";
34.
35.
36.     $selCatId = $_GET['cat'] ?? 0; // category_id
37.     $selUserId = $_GET['user'] ?? 0; // user_id
38.     $whereConditions = [];
39.     $params = [];
40.
41.     if ($selCatId) {
42.         $whereConditions[] = "c.category_id = ?";
43.         $params[] = $selCatId;
44.     }
45.
46.     if ($selUserId) {
47.         $whereConditions[] = "u.user_id = ?";
48.         $params[] = $selUserId;
49.     }
50.
51.     if ($whereConditions) {
52.         $where = implode(' AND ', $whereConditions);
53.         $query .= ' AND ' . $where;
54.     }
55.
56.     $query .= " ORDER BY $order $dir
57.         LIMIT $offset, $rowsToShow";
58.
59.     $stmt = $db->prepare($query);
60.     $stmt->execute($params);
61.
62.     $qPoemCount = "SELECT COUNT(p.poem_id) AS num
63.     FROM poems p
64.         JOIN categories c ON c.category_id = p.category_id
65.         JOIN users u ON u.user_id = p.user_id
66.     WHERE p.date_approved IS NOT NULL";
67.
68.     if ($whereConditions) {
69.         $where = implode(' AND ', $whereConditions);
70.         $qPoemCount .= ' AND ' . $where;
```

Evaluation
Copy

```

71.     }
72.
73.     $stmtPoemCount = $db->prepare($qPoemCount);
74.     $stmtPoemCount->execute($params);
75.     $poemCount = $stmtPoemCount->fetch()['num'];
76.
77.     $prevOffset = max($offset - $rowsToShow, 0);
78.     $nextOffset = $offset + $rowsToShow;
79.
80.     $qCategories = "SELECT c.category_id, c.category,
81.         COUNT(p.poem_id) AS num_poems
82.     FROM categories c
83.         JOIN poems p ON c.category_id = p.category_id
84.     WHERE p.date_approved IS NOT NULL
85.     GROUP BY c.category_id
86.     ORDER BY c.category";
87.
88.     $stmtCats = $db->prepare($qCategories);
89.     $stmtCats->execute();
90.
91.     $qUsers = "SELECT u.user_id, u.username,
92.         COUNT(p.poem_id) AS num_poems
93.     FROM users u
94.         JOIN poems p ON u.user_id = p.user_id
95.     WHERE p.date_approved IS NOT NULL
96.     GROUP BY u.user_id
97.     ORDER BY u.username";
98.
99.     $stmtUsers = $db->prepare($qUsers);
100.    $stmtUsers->execute();
101.
102.    $href = "poems-filtering.php?cat=$selCatId&user=$selUserId&";
103.    // "poems.php?cat=$selCatId&user=$selUserId&" for you.
-----Lines 104 through 196 Omitted-----
197.    <h2>Filtering</h2>
198.    <!--The form action will be poems.php for you.-->
199.    <form method="get" action="poems-filtering.php">
200.        <input type="hidden" name="order" value="<?= $order ?>">
201.        <input type="hidden" name="dir" value="<?= $dir ?>">
202.        <label for="cat">Category:</label>
203.        <select name="cat" id="cat">
204.            <option value="0">All</option>
205.        <?php
206.            while ($row = $stmtCats->fetch()) {
207.                $category = $row['category'];

```

```

208.         $numPoems = $row['num_poems'];
209.         $categoryId = $row['category_id'];
210.         $selected = $categoryId === $selCatId ? 'selected' : '';
211.         echo "<option value='$categoryId' $selected>
212.             $category ($numPoems)
213.         </option>";
214.     }
215.     ?>
216. </select>
217. <label for="user">Author:</label>
218. <select name="user" id="user">
219.     <option value="0">All</option>
220.     <?php
221.         while ($row = $stmtUsers->fetch()) {
222.             $username = $row['username'];
223.             $userId = $row['user_id'];
224.             $numPoems = $row['num_poems'];
225.             $selected = $userId === $selUserId ? 'selected' : '';
226.             echo "<option value='$userId' $selected>
227.                 $username ($numPoems)
228.             </option>";
229.         }
230.     ?>
231. </select>
232. <button name="filter" class="wide">Filter</button>
233. </form>
234. </main>
235. <?php
236.     require 'includes/footer.php';
237. ?>

```

Code Explanation

When reviewing this code, you may find it helpful to go back through the exercise instructions and match up each instruction with the relevant solution code.

Exercise 19: Adding Filtering Links to the Single Poem Page

 20 to 30 minutes

Currently, on `poem.php`, there are a couple of links under the poem that don't go anywhere and have static text that isn't always relevant:

```
<li>
  <i class="fas fa-circle"></i>
  <a href="#">More Funny Poems</a>
</li>
<li>
  <i class="fas fa-circle"></i>
  <a href="#">More Poems by LimerickMan</a>
</li>
```

In this exercise, you will make these links work and be related to the poem on the page.

1. Open `Database/Exercises/phppoetry.com/poem.php` in your editor.
2. Modify the query so that:
 - A. It gets the `user_id` from the `users` table.
 - B. It joins on the `categories` table as well and gets the `category_id` and the `category` fields from that table.
3. Using the `user_id`, `category_id`, and `category` returned from the query, edit the page so that the links below the poem have text relevant to the poem and work correctly.
4. Visit `http://localhost:8888/Webucator/php/Database/Exercises/phppoetry.com/poems.php` to test your solution. Click poems in different categories or by different authors. Check the links below the poem to make sure they update correctly.

Look in `Database/Exercises/phppoetry.com/sql.txt` if you need help with the SQL query.

Solution: Database/Solutions/phppoetry.com/poem-2.php

```
-----Lines 1 through 6 Omitted-----
7.     $query = "SELECT u.username, u.user_id,
8.         p.title, p.poem, p.date_submitted, p.date_approved,
9.         c.category_id, c.category
10.    FROM users u
11.        JOIN poems p ON u.user_id = p.user_id
12.        JOIN categories c ON c.category_id = p.category_id
13.    WHERE p.poem_id = ?";
14.    $stmt = $db->prepare($query);
15.    $stmt->execute([$poemId]);
16.    $row = $stmt->fetch();
17.
18.    if ($row) {
19.        $title = $row['title'];
20.        $authorUserId = $row['user_id'];
21.        $authorUserName = $row['username'];
22.        $dateSubmitted = $row['date_submitted'];
23.        $dateApproved = $row['date_approved'];
24.        $poem = $row['poem'];
25.        $categoryId = $row['category_id'];
26.        $category = $row['category'];
27.    } else {
28.        $title = 'Poem Not Found';
29.    }
-----Lines 30 through 54 Omitted-----
55.    <?php if ($row) { ?>
56.        <li>
57.            <i class="fas fa-circle"></i>
58.            <a href="poems.php?cat=<?= $categoryId ?>">
59.                More <?= $category ?> Poems
60.            </a>
61.        </li>
62.        <li>
63.            <i class="fas fa-circle"></i>
64.            <a href="poems.php?user=<?= $authorUserId ?>">
65.                More Poems by <?= $authorUserName ?>
66.            </a>
67.        </li>
68.    <?php } ?>
-----Lines 69 through 78 Omitted-----
```

Conclusion

In this lesson, you have learned how to connect to a database, query it, and use the results to display, sort, paginate through, and filter records.

LESSON 6

Exception Handling

Topics Covered

- Exceptions.
- Logging errors.

Introduction

Like most programming languages, PHP throws exceptions (i.e., reports an error with detailed information) when something goes wrong. The programmer can anticipate, catch, and handle those exceptions in the code.



6.1. Uncaught Exceptions

If an exception is thrown by PHP and there is no code in place to handle the exception, then PHP will log the exception in the `php_error.log` and, if the `display_errors` directive is on, send the error to the browser to display. If the error is fatal, no further code will be processed.

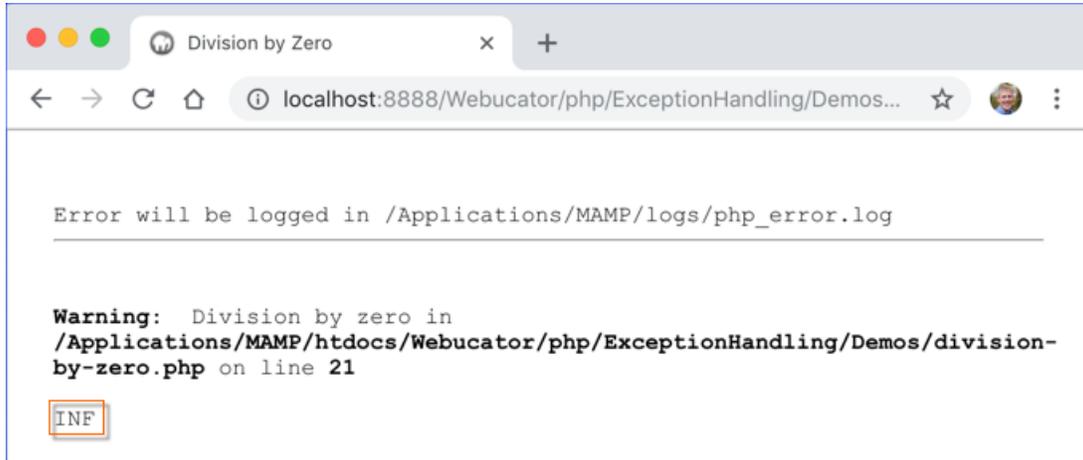
The following example shows how PHP handles division by zero, which generates a *warning*, the lowest level type of error. This does not stop execution of the code.

Demo 6.1: ExceptionHandling/Demos/division-by-zero.php

```
1.  <?php
2.      ini_set('display_errors', '1');
3.  ?>
4.  <!DOCTYPE html>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <link rel="stylesheet" href="../../static/styles/normalize.css">
10. <link rel="stylesheet" href="../../static/styles/styles.css">
11. <title>Division by Zero</title>
12. </head>
13. <body>
14. <main class="pre">
15. <?php
16.     echo 'Error will be logged in ' . ini_get('error_log');
17.     echo '<hr>';
18.
19.     $num = 5;
20.     $den = 0;
21.     $result = $num / $den;
22.     echo $result;
23. ?>
24. </main>
25. </body>
26. </html>
```

Code Explanation

Visit <http://localhost:8888/Webucator/php/ExceptionHandling/Demos/division-by-zero.php> to run this file in your browser. You should see something like this:



Notice that the code continues to run after the error is output. INF (for *infinity*) is the result of division by zero.

We have used `ini_get('error_log')` to find out the location of `php_error.log`: `/Applications/MAMP/logs/php_error.log`. The location may be different on your computer. Open that file in any text editor and you should see the error reported:

```
[29-Jan-2019 19:07:09 UTC] PHP Warning: Division by zero in /Applications/MAMP/htdocs/Webucator/php/ExceptionHandling/Demos/division-by-zero.php on line 21
```

You may want to keep `php_error.log` open for the remainder of this lesson as we will be referring to it again.



6.2. Throwing Your Own Exceptions

It is possible to throw your own exceptions using `throw`. While it's likely you won't have to do this unless you are creating PHP libraries that are used by other developers, it is worth seeing how it is done as you will certainly be catching exceptions thrown by libraries and PHP extensions that you use in your code.

In the following example, we create our own `divide()` function which throws an exception if the denominator is 0:

Demo 6.2: ExceptionHandling/Demos/uncaught-exception.php

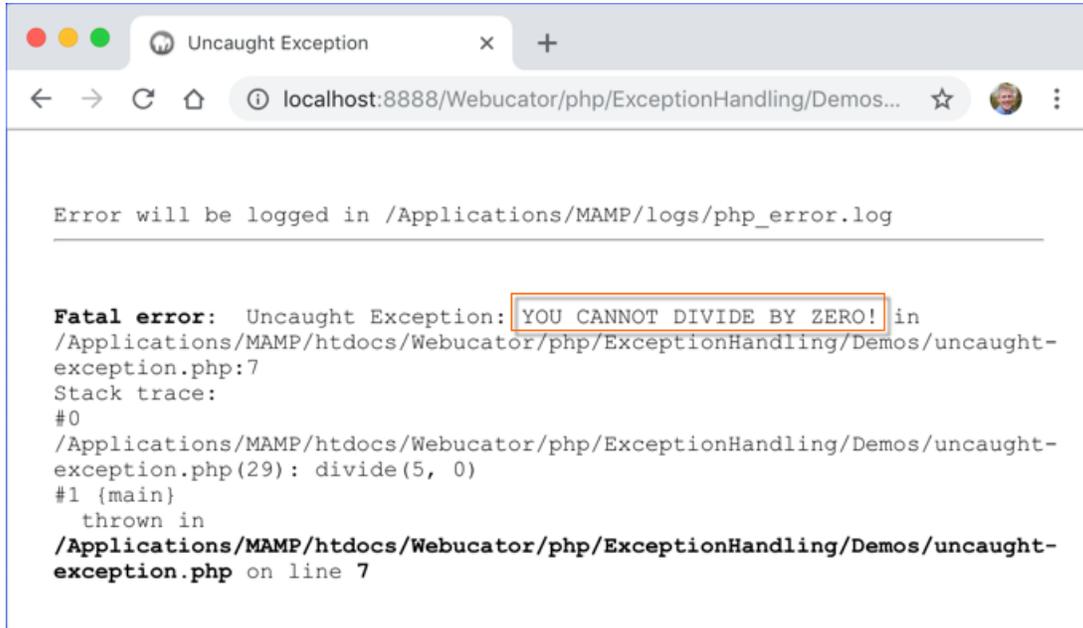
```
1.  <?php
2.      ini_set('display_errors', '1');
3.      function divide($numerator, $denominator) {
4.          $numerator = (int) $numerator;
5.          $denominator = (int) $denominator;
6.          if ($denominator === 0) {
7.              throw new Exception('YOU CANNOT DIVIDE BY ZERO!');
8.          }
9.          return $numerator / $denominator;
10.     }
11.  ?>
12.  <!DOCTYPE html>
13.  <html lang="en">
14.  <head>
15.  <meta charset="UTF-8">
16.  <meta name="viewport" content="width=device-width,initial-scale=1">
17.  <link rel="stylesheet" href="../../static/styles/normalize.css">
18.  <link rel="stylesheet" href="../../static/styles/styles.css">
19.  <title>Uncaught Exception</title>
20.  </head>
21.  <body>
22.  <main class="pre">
23.  <?php
24.      echo 'Error will be logged in ' . ini_get('error_log');
25.      echo '<hr>';
26.
27.      $num = 5;
28.      $den = 0;
29.      $result = divide($num, $den);
30.      echo $result;
31.  ?>
32.  </main>
33.  </body>
34.  </html>
```



Code Explanation

Notice that we cast `$numerator` and `$denominator` as integers. We do this in case someone passes in a string (e.g., '0'), which could be the case if they are getting the number from a form submitted by a user.

When you run this file in your browser, you should see something like this:

A screenshot of a web browser window titled "Uncaught Exception". The address bar shows "localhost:8888/Webucator/php/ExceptionHandling/Demos...". The main content area displays an error report. At the top, it says "Error will be logged in /Applications/MAMP/logs/php_error.log". Below that, the error message is "Fatal error: Uncaught Exception: YOU CANNOT DIVIDE BY ZERO! in /Applications/MAMP/htdocs/Webucator/php/ExceptionHandling/Demos/uncaught-exception.php:7". A stack trace follows, showing the error occurred in the file "uncaught-exception.php" at line 29, specifically in the "divide(5, 0)" function call. The error was thrown in the main script "uncaught-exception.php" on line 7.

```
Error will be logged in /Applications/MAMP/logs/php_error.log

Fatal error: Uncaught Exception: YOU CANNOT DIVIDE BY ZERO! in
/Applications/MAMP/htdocs/Webucator/php/ExceptionHandling/Demos/uncaught-
exception.php:7
Stack trace:
#0
/Applications/MAMP/htdocs/Webucator/php/ExceptionHandling/Demos/uncaught-
exception.php(29): divide(5, 0)
#1 {main}
   thrown in
/Applications/MAMP/htdocs/Webucator/php/ExceptionHandling/Demos/uncaught-
exception.php on line 7
```

Notice that the message we threw in our code is shown in the error report. The same error will be reported in `php_error.log`.

Also notice that the code does **not** continue to run after the error is output. Exceptions that are thrown this way are fatal. To prevent them from stopping the program, you have to catch them.



6.3. Catching Exceptions

❖ 6.3.1. Getting Information about Exceptions

Exception objects have methods for getting information about the exception. Three useful methods are:

1. `getMessage()` - returns the exception's message.
2. `getFile()` - returns the path to the file in which the exception occurred.
3. `getLine()` - returns the line number on which the exception occurred.

To catch an exception, you have to anticipate when it might occur. We know that the `divide()` function can throw an exception, so we should be prepared for this possibility. The following code *tries* to pass `divide()` a value of `0` for the denominator and then *catches* the exception that `divide()` throws:

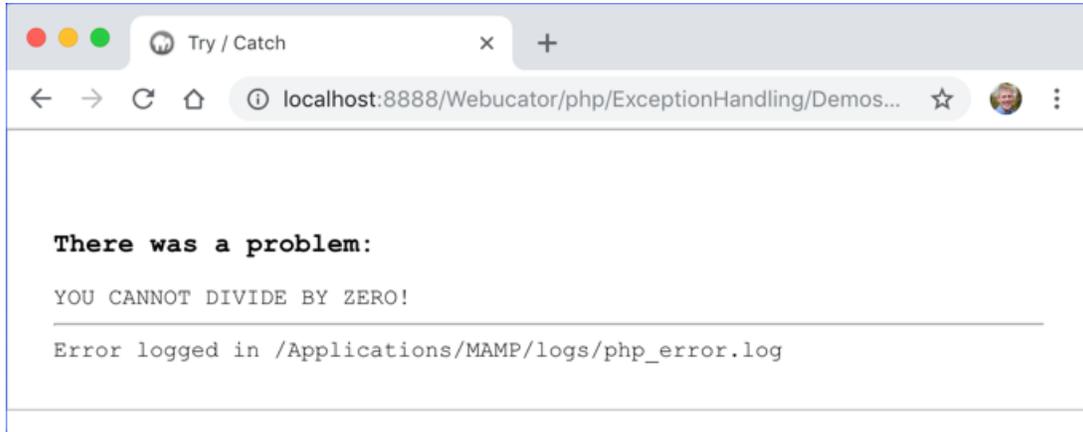
Demo 6.3: ExceptionHandling/Demos/try-catch.php

```
1.  <?php
2.      ini_set('display_errors', '1');
3.      function divide($numerator, $denominator) {
4.          $numerator = (int) $numerator;
5.          $denominator = (int) $denominator;
6.          if ($denominator === 0) {
7.              throw new Exception('YOU CANNOT DIVIDE BY ZERO!');
8.          }
9.          return $numerator / $denominator;
10.     }
11.     ?>
-----Lines 12 through 22 Omitted-----
23. <?php
24.     $num = 5;
25.     $den = 0;
26.
27.     try {
28.         $result = divide($num, $den);
29.         echo $result;
30.     } catch (Exception $e) {
31.         echo '<h3>There was a problem:</h3>';
32.         $errorMsg = $e->getMessage();
33.         echo $errorMsg;
34.         error_log( $errorMsg . ' in ' . $e->getFile() .
35.             ' on line ' . $e->getLine() );
36.         echo '<hr>';
37.         echo 'Error logged in ' . ini_get('error_log');
38.     }
39.     ?>
-----Lines 40 through 42 Omitted-----
```



Code Explanation

When you run this file in your browser, you should see something like this:



Notice that the `try / catch` syntax is similar to the `if / else` syntax, but unlike `if / else`, in `try / catch` code, the `catch` block is required.

You will also see the error reported in `php_error.log`, but only because we explicitly logged it using the built-in `error_log()` function. This code:

```
error_log( $errorMsg . ' in ' .  
          $e->getFile() . ' on line ' . $e->getLine() );
```

... resulted in the following logged error:

```
[29-Jan-2019 19:12:21 UTC] YOU CANNOT DIVIDE BY ZERO! in /Applications/MAMP/htdocs/Webu  
cator/php/ExceptionHandling/Demos/try-catch.php on line 7
```

Exercise 20: Division Form

 10 to 15 minutes

In this exercise, you will process data passed in through a form to either return the result of a division equation or an error message. The starting code is shown below:

Exercise Code 20.1: ExceptionHandling/Exercises/division.php

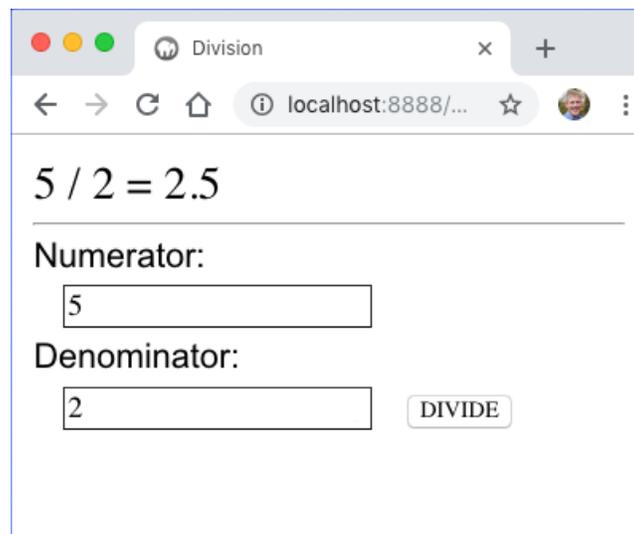
```
1.  <?php
2.      ini_set('display_errors', '1');
3.      function divide($numerator, $denominator) {
4.          $numerator = (int) $numerator;
5.          $denominator = (int) $denominator;
6.          if ($denominator === 0) {
7.              throw new Exception('YOU CANNOT DIVIDE BY ZERO!');
8.          }
9.          return $numerator / $denominator;
10.     }
11.  ?>
12.  <!DOCTYPE html>
13.  <html lang="en">
14.  <head>
15.  <meta charset="UTF-8">
16.  <meta name="viewport" content="width=device-width,initial-scale=1">
17.  <link rel="stylesheet" href="../../static/styles/normalize.css">
18.  <link rel="stylesheet" href="../../static/styles/styles.css">
19.  <title>Division</title>
20.  </head>
21.  <body>
22.  <main>
23.      <?php
24.          $num = $_GET['num'] ?? '';
25.          $den = $_GET['den'] ?? '';
26.          if (is_numeric($num) && is_numeric($den)) {
27.              // Your code here
28.          }
29.      ?>
30.      <form method="get" action="division.php">
31.          <label for="den">Numerator:</label>
32.          <input id="num" name="num" type="number" value="<?= $num ?>">
33.          <label for="den">Denominator:</label>
34.          <input id="den" name="den" type="number" value="<?= $den ?>">
35.          <button>DIVIDE</button>
36.      </form>
37.  </main>
38.  </body>
39.  </html>
```

Code Explanation

Notice that we check if `$num` and `$den` are numeric¹². If they are not, it is likely because the form hasn't been submitted and they are both empty strings, so we won't try to do any division.

1. Open `ExceptionHandler/Exercises/division.php` in your editor.
2. Add a `try / catch` block:
 - A. Attempt to divide using the numbers the user entered in the form. If all goes well, output something like:

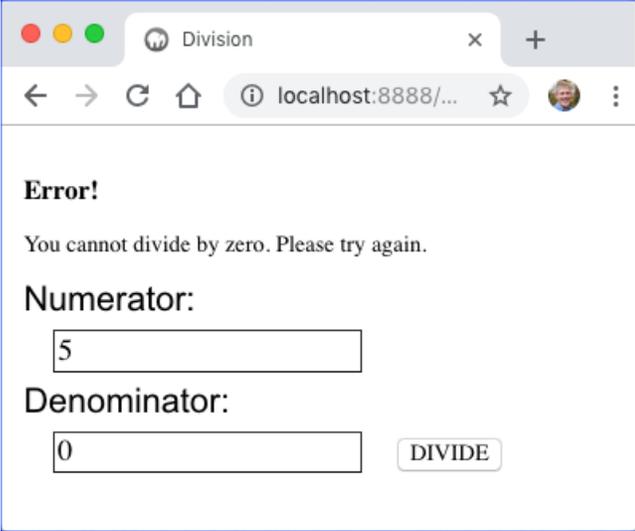
```
<output>5 / 2 = 2.5</output><hr>
```



- B. If the user enters `0` for the denominator, output an error similar to:

```
<h3>Error!</h3>
<p>You cannot divide by zero. Please try again.</p>
```

12. See https://www.php.net/is_numeric for details on the `is_numeric()` function.



Solution: ExceptionHandling/Solutions/division.php

```
-----Lines 1 through 22 Omitted-----
23. <?php
24.     $num = $_GET['num'] ?? '';
25.     $den = $_GET['den'] ?? '';
26.     if (is_numeric($num) && is_numeric($den)) {
27.         try {
28.             $result = divide($num, $den);
29.             echo '<output>' . $num . ' / ' . $den .
30.                 ' = ' . $result. '</output><hr>';
31.         } catch (Exception $e) {
32.             echo '<h3>Error!</h3>
33.                 <p>You cannot divide by zero.</p>';
34.         }
35.     }
36.     ?>
-----Lines 37 through 46 Omitted-----
```



6.4. PDOExceptions

We have been using the PDO extension to connect to our database. Generally, extensions like this will have specific types of exceptions that are extended from PHP's built-in exceptions. When your PDO code errors, it will throw a PDOException. So, try / catch blocks will be structured like this:

```
try {
    // PDO code here
} catch (PDOException $e) {
    // Handle error
}
```

When working with external systems, like databases, email, the file system, etc., a lot can go wrong that is outside the PHP developer's control. For example, a database can go down, or the sign-in credentials can change, or table names can be changed. Because of these possibilities, you should write your code expecting things to go wrong, so your website doesn't just stop working without any explanation to the user.

Now you will have the chance to try this out on the Poetry website with a few exercises:

1. First, you will do an exercise in which you create a `logError()` function in `utilities.php` that handles how errors are logged.
2. Second, you will do an exercise in which you create a `dbConnect()` function in `utilities.php` that connects to the database and returns the connection. If there is an error, this function will log the error and return `false`.
3. Third, you will modify the site's PHP files to make use of this new `dbConnect()` function and you will add code to properly handle errors when preparing and executing queries.

Exercise 21: Logging Errors

 20 to 30 minutes

In this exercise, you will create a `logError()` function in `utilities.php` that handles how errors are logged.

1. Open `ExceptionHandler/Exercises/phppoetry.com/includes/utilities.php` in your editor.
2. Add a function with the following signature:

```
void logError(mixed $e, [bool $redirect=false])
```

- A. `void` means the function does not return anything.
 - B. `mixed` means that `$e` could be more than one type.
 - C. The `$redirect` parameter is an optional boolean value, which defaults to `false`.
3. Create a `$msg` variable to hold the error message to be logged. Use the `gettype()` function¹³ and an `if - else` block or a `switch / case` block to check the type of `$e`. If the type of `$e` is a string, then `$e` should be assigned to `$msg`. Otherwise, the function should assume the type is an `Exception` object and it should construct the error message from `$e->getMessage()`, `$e->getFile()`, and `$e->getLine()` and assign the result to `$msg`.
 4. Log `$msg` to the error log.
 5. If in debug mode, output the following to the page:

```
<h3 class='error'>For Developers' Eyes Only</h3>
<div class='error'>$msg</div>
```

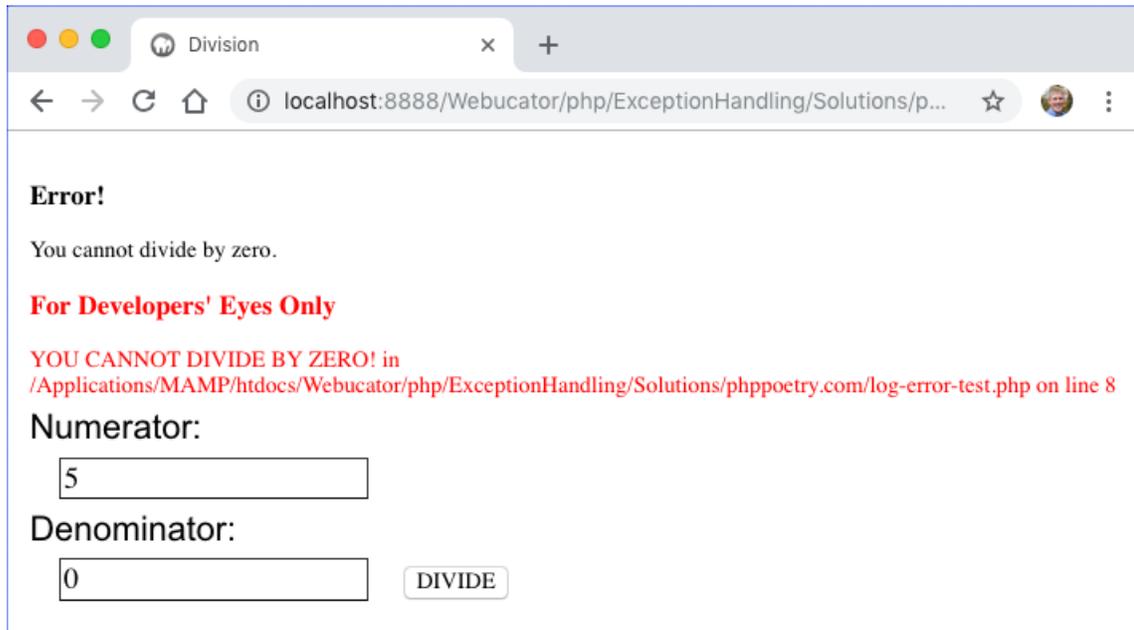
6. If `$redirect` is `true` and not in debug mode, the function should redirect to an error page using the following line of code:

```
header("Location: error-page.php");
```

13. See <https://www.php.net/gettype> for details on the `gettype()` function.

This tells the browser not to show the page with the error, but to instead redirect to `error-page.php`.

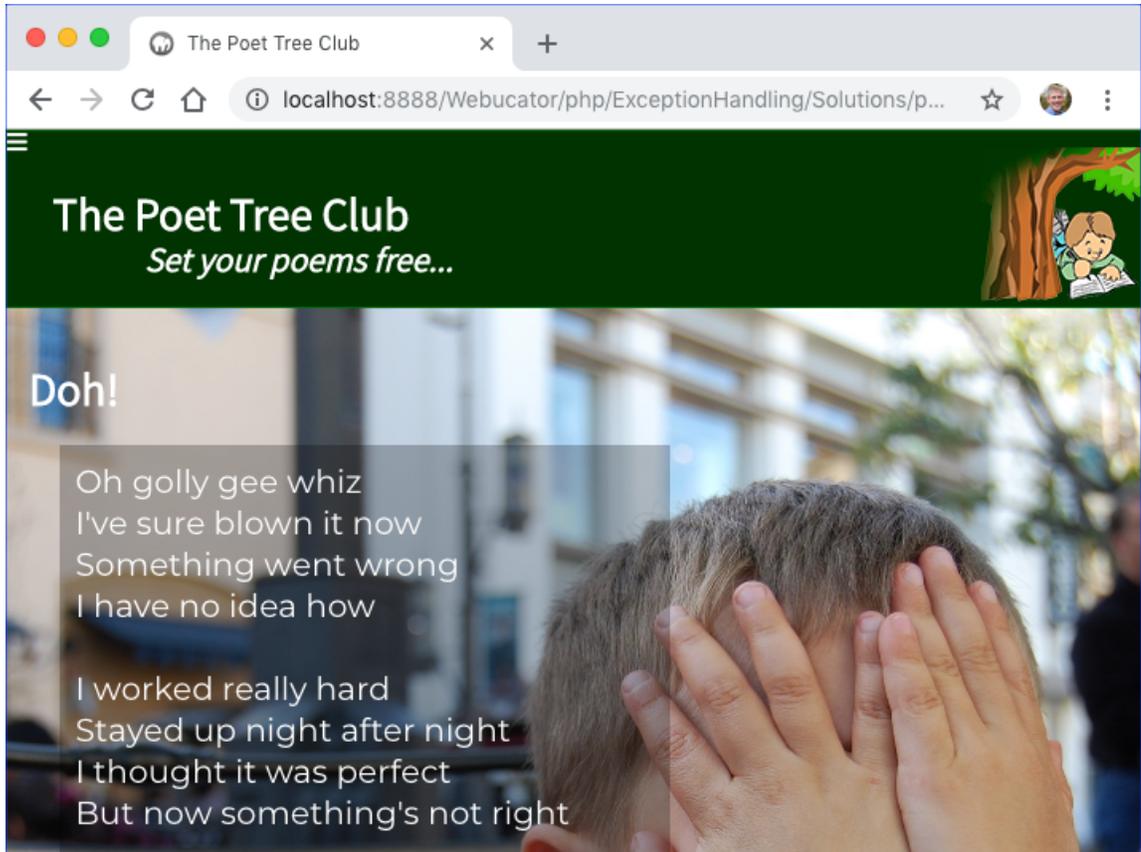
7. Visit `http://localhost:8888/Webucator/php/ExceptionHandling/Exercises/php-poetry.com/log-error-test.php` to test your code. Enter any number for the numerator and `0` for the denominator and press the **Divide** button. You should see a result like this:



If this doesn't work, fix your code and retest. If the page doesn't work at all, check `php_error.log` to see what went wrong.

- A. To see how the error would work on production, open `utilities.php` in your editor and change the `isProduction()` function to return `true` instead of `false`. Then run the file again. This time, you should get redirected to an error page:¹⁴

14. The <https://pixabay.com/en/boy-facepalm-child-youth-666803/> image is used under the terms of Pixabay License (<https://pixabay.com/service/license/>).



B. Be sure to change the `isProduction()` function back to returning `false`.

Solution

The `logError()` function should look like this:

```
function logError($e, $redirect=false) {
    $errorType = gettype($e);
    switch ($errorType) {
        case 'string':
            $msg = $e;
            break;
        default:
            $msg = $e->getMessage() . ' in ' . $e->getFile() .
                ' on line ' . $e->getLine();
    }
    error_log($msg); // php_error.log

    if (isDebugMode()) {
        echo "<h3 class='error'>For Developers' Eyes Only</h3>
            <div class='error'>$msg</div>";
    }

    if ($redirect && !isDebugMode()) {
        // Redirect to error page
        header("Location: error-page.php");
    }
}
```

Exercise 22: The dbConnect() Function

 20 to 30 minutes

In this exercise, you will create a `dbConnect()` function in `utilities.php` that connects to the database and returns the connection. If there is an error, this function will log the error and return `false`. The function signature is as follows:

```
mixed dbConnect()
```

`mixed` means the function can return different types. Often, when a function can return different types, it returns the expected object if all goes well and returns `false` if something goes wrong. This is the case for the `dbConnect()` function you are about to create.

1. Open `ExceptionHandler/Exercises/phppoetry.com/includes/utilities.php` in your editor.
2. Add a line at the top to include `config.php`, which we placed in the `includes` directory outside of the web root earlier in the course. That file contains a `getDbConfig()` function, which returns an array with keys for `'dsn'`, `'un'`, and `'pw'`.
3. Create a new function called `dbConnect()` that:
 - A. Sets `$dbConfig` to what `getDbConfig()` returns.
 - B. Sets `$dsn` to `$dbConfig['dsn']`.
 - C. Sets `$username` to `$dbConfig['un']`.
 - D. Sets `$password` to `$dbConfig['pw']`.
 - E. Attempts to create and return a database connection.
 - F. On failure, it should return `false` after calling `logError()` and passing it the exception and `true`, meaning it should redirect to the error page when not in debug mode.
4. Visit <http://localhost:8888/Webucator/php/ExceptionHandler/Exercises/php-poetry.com/test-db-connect.php> to test your code. You should get a “Success” message if the connection succeeds.

Solution:

ExceptionHandling/Solutions/phppoetry.com/includes/utilities.php

```
1. <?php
2.     require_once 'config.php';
3.
4.     function isProduction() {
5.         // Provide way of knowing if the code is on production server
6.         return false;
7.     }
8.
9.     function isDebugMode() {
10.        // You may want to provide other ways for setting debug mode
11.        return !isProduction();
12.    }
13.
14.    function dbConnect() {
15.        $dbConfig = getDbConfig();
16.        $dsn = $dbConfig['dsn'];
17.        $username = $dbConfig['un'];
18.        $password = $dbConfig['pw'];
19.
20.        try {
21.            $db = new PDO($dsn, $username, $password);
22.            return $db;
23.        } catch (PDOException $e) {
24.            // log error
25.            logError($e, true);
26.            return false;
27.        }
28.    }
29.
30.    function logError($e, $redirect=false) {
31.        $errorType = gettype($e);
32.        switch ($errorType) {
33.            case 'string':
34.                $msg = $e;
35.                break;
36.            default:
37.                $msg = $e->getMessage() . ' in ' . $e->getFile() .
38.                    ' on line ' . $e->getLine();
39.        }
40.        error_log($msg); // php_error.log
41.
42.        if (isDebugMode()) {
43.            echo "<h3 class='error'>For Developers' Eyes Only</h3>
```

```
44.         <div class='error'>$msg</div>";
45.     }
46.
47.     if ($redirect && !isDebugMode()) {
48.         // Redirect to error page
49.         header("Location: error-page.php");
50.     }
51. }
52. ?>
```



6.5. When Queries Fail to Execute

The `PDOStatement`'s `execute()` method returns `true` if the database query succeeds and `false` if the query fails. When a query fails, you can get information on the cause of the failure by calling the `PDOStatement`'s `errorInfo()` method, which returns an array containing:

1. A `SQLSTATE` error code.
2. A driver-specific error code.
3. A driver-specific error message.

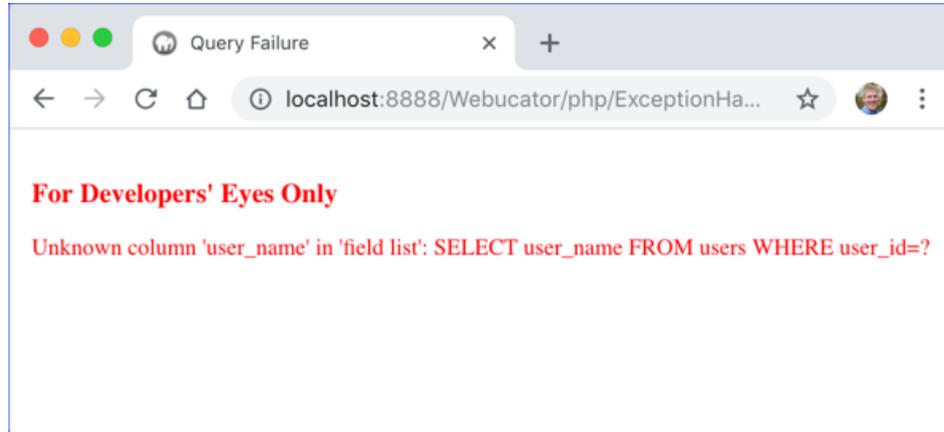
The last of these generally gives you the information you need to debug your query. Take a look at the following code:

Demo 6.4: ExceptionHandling/Demos/query-failure.php

```
1.  <?php
2.      require_once '../Solutions/phppoetry.com/includes/utilities.php';
3.  ?>
4.  <!DOCTYPE html>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <link rel="stylesheet" href="../../static/styles/normalize.css">
10. <link rel="stylesheet" href="../../static/styles/styles.css">
11. <title>Query Failure</title>
12. </head>
13. <body>
14. <main>
15. <?php
16.     $db = dbConnect();
17.     try {
18.         // users table doesn't have user_name field
19.         $query = 'SELECT user_name FROM users';
20.         $stmt = $db->prepare($query);
21.         if (!$stmt->execute()) {
22.             // The query failed
23.             $errorMsg = $stmt->errorInfo()[2] . ": $query";
24.             logError($errorMsg, true);
25.         }
26.     } catch (PDOException $e) {
27.         // Some error occurred with our database communication
28.         logError($e, true);
29.     }
30.
31.     // If we get here without an error,
32.     // we can safely fetch data from $stmt
33. ?>
34. </main>
35. </body>
36. </html>
```

Code Explanation

Visit <http://localhost:8888/Webucator/php/ExceptionHandling/Demos/query-failure.php> to test this page. It should look like this:



Things to note:

1. We are intentionally trying a query that will break: there is no `user_name` field in the `users` table.
2. If the query fails (and it will), `$stmt->execute()` will return `false`, so `!$stmt->execute()` will return `true`. This is *not an exception in our PHP code*. It is a *database error*; it is MySQL telling PHP that the query is bad. We log `$stmt->errorInfo()[2]` to provide error information for the developer.
3. It is also possible that something goes wrong with the communication between PHP and the database. For example, the database could go down between the time that we connected to it and the time we ran this query. In this case, a `PDOException` will occur. That's why we put the whole thing in a `try/catch` block.
4. If no `PDOException` occurs and the query succeeds, the `PDOStatement` object will then have the data from the database and we can safely fetch it.



Exercise 23: Catching Errors in the PHP Poetry Website

⌚ 40 to 60 minutes

In this exercise, you will change the code to use the `dbConnect()` function in `utilities.php`. You will also wrap all the code that prepares and executes SQL statements in error-catching code.

1. Add code to `includes/header.php` to connect to the database and store the connection in `$db` **unless \$db has already been set**.
2. Open `index.php` in your editor and review the code. Notice that we have removed the database connection code (shown below) and that we have wrapped the `prepare()` and `execute()` calls in a `try/catch` block. You must do the same in `poems.php` and `poem.php`:

```
$dsn = 'mysql:host=localhost;dbname=poetree';  
$username = 'root';  
$password = 'pwdpwd';  
$db = new PDO($dsn, $username, $password);
```

3. In `poem.php`, we need to connect to and query the database before including `header.php`, so that we can include the name of the poem in the title of the page. Set `$db` in `poem.php`. Note that this is why in `header.php` we only set `$db` if it hasn't already been set.
4. Visit <http://localhost:8888/Webucator/php/ExceptionHandling/Exercises/php-poetry.com/index.php> and navigate around to test the site. If you get any errors, they should be reported to the browser in a way that helps you fix them.

Solution:

ExceptionHandling/Solutions/phppoetry.com/includes/header.php

```
1.  <?php
2.      require_once 'config.php';
3.      require_once 'utilities.php';
4.      if (isDebugMode()) {
5.          ini_set('display_errors', '1');
6.      }
7.
8.      // If $db isn't already set, set it.
9.      if (!isset($db)) {
10.         $db = dbConnect();
11.     }
12.
13.     $pageTitleTag = empty($pageTitle)
14.         ? 'The Poet Tree Club'
15.         : $pageTitle . ' | The Poet Tree Club';
16. ?>
-----Lines 17 through 52 Omitted-----
```



Solution: ExceptionHandling/Solutions/phppoetry.com/index.php

```
1. <?php
2.     require 'includes/header.php';
3.
4.     $query = "SELECT p.poem_id, p.title, p.date_approved,
5.     c.category, u.username
6.             FROM poems p
7.             JOIN categories c ON c.category_id = p.category_id
8.             JOIN users u ON u.user_id = p.user_id
9.             WHERE p.date_approved IS NOT NULL
10.            ORDER BY p.date_approved DESC
11.            LIMIT 0, 3";
12.
13.     try {
14.         $stmt = $db->prepare($query);
15.         if (!$stmt->execute()) {
16.             $errorMsg = $stmt->errorInfo()[2] . ": $query";
17.             logError($errorMsg);
18.         }
19.     } catch (PDOException $e) {
20.         logError($e, true);
21.     }
22.     ?>
-----Lines 23 through 61 Omitted-----
```

Solution: ExceptionHandling/Solutions/phppoetry.com/poems.php

```
-----Lines 1 through 54 Omitted-----
55.     try {
56.         $stmt = $db->prepare($query);
57.         if (!$stmt->execute($params)) {
58.             $errorMsg = $stmt->errorInfo()[2] . ": $query";
59.             logError($errorMsg);
60.         }
61.     } catch (PDOException $e) {
62.         logError($e, true);
63.     }
64.
65.     $qPoemCount = "SELECT COUNT(p.poem_id) AS num
66. FROM poems p
67.     JOIN categories c ON c.category_id = p.category_id
68.     JOIN users u ON u.user_id = p.user_id
69. WHERE p.date_approved IS NOT NULL";
70.
71.     if ($whereConditions) {
72.         $where = implode(' AND ', $whereConditions);
73.         $qPoemCount .= ' AND ' . $where;
74.     }
75.
76.     try {
77.         $stmtPoemCount = $db->prepare($qPoemCount);
78.         if (!$stmtPoemCount->execute($params)) {
79.             $errorMsg = $stmtPoemCount->errorInfo()[2] . ": $query";
80.             logError($errorMsg);
81.         }
82.     } catch (PDOException $e) {
83.         logError($e, true);
84.     }
85.     $poemCount = $stmtPoemCount->fetch()['num'];
86.
87.     $prevOffset = max($offset - $rowsToShow, 0);
88.     $nextOffset = $offset + $rowsToShow;
89.
90.     $qCategories = "SELECT c.category_id, c.category,
91.         COUNT(p.poem_id) AS num_poems
92. FROM categories c
93.     JOIN poems p ON c.category_id = p.category_id
94. WHERE p.date_approved IS NOT NULL
95. GROUP BY c.category_id
96. ORDER BY c.category";
97.
```

```
98.     try {
99.         $stmtCats = $db->prepare($qCategories);
100.        if (!$stmtCats->execute()) {
101.            $errorMsg = $stmtCats->errorInfo()[2] . ": $query";
102.            logError($errorMsg);
103.        }
104.    } catch (PDOException $e) {
105.        logError($e, true);
106.    }
107.
108.    $qUsers = "SELECT u.user_id, u.username,
109.             COUNT(p.poem_id) AS num_poems
110.             FROM users u
111.             JOIN poems p ON u.user_id = p.user_id
112.             WHERE p.date_approved IS NOT NULL
113.             GROUP BY u.user_id
114.             ORDER BY u.username";
115.
116.    try {
117.        $stmtUsers = $db->prepare($qUsers);
118.        if (!$stmtUsers->execute()) {
119.            $errorMsg = $stmtUsers->errorInfo()[2] . ": $query";
120.            logError($errorMsg);
121.        }
122.    } catch (PDOException $e) {
123.        logError($e, true);
124.    }
-----Lines 125 through 259 Omitted-----
```

Evaluation
Copy

Solution: ExceptionHandling/Solutions/phppoetry.com/poem.php

```
1. <?php
2.     require_once 'config.php';
3.     require_once 'includes/utilities.php';
4.     $db = dbConnect();
5.     $poemId = $_GET['poem-id'];
6.     $query = "SELECT u.username, u.user_id,
7.         p.title, p.poem, p.date_submitted, p.date_approved,
8.         c.category_id, c.category
9.     FROM users u
10.        JOIN poems p ON u.user_id = p.user_id
11.        JOIN categories c ON c.category_id = p.category_id
12.        WHERE p.poem_id = ?";
13.
14.     try {
15.         $stmt = $db->prepare($query);
16.         if (!$stmt->execute([$poemId])) {
17.             $errorMsg = $stmt->errorInfo()[2] . ": $query";
18.             logError($errorMsg);
19.         }
20.     } catch (PDOException $e) {
21.         logError($e, true);
22.     }
23.     $row = $stmt->fetch();
    -----Lines 24 through 85 Omitted-----
```

Conclusion

In this lesson, you have learned to catch PHP exceptions, and specifically, to write robust and safe PHP code for connecting with and querying a database.

LESSON 7

PHP and HTML Forms

Topics Covered

- Processing form data.

Introduction

In this lesson, you will learn to process form data with PHP.



7.1. HTML Forms

❖ 7.1.1. How HTML Forms Work

A common way to pass data from one page to another is through HTML forms. There are two methods of submitting data through a form: the *get* method and the *post* method. The method used is determined by the value of the `method` attribute of the `form` tag. The default method is `get`.

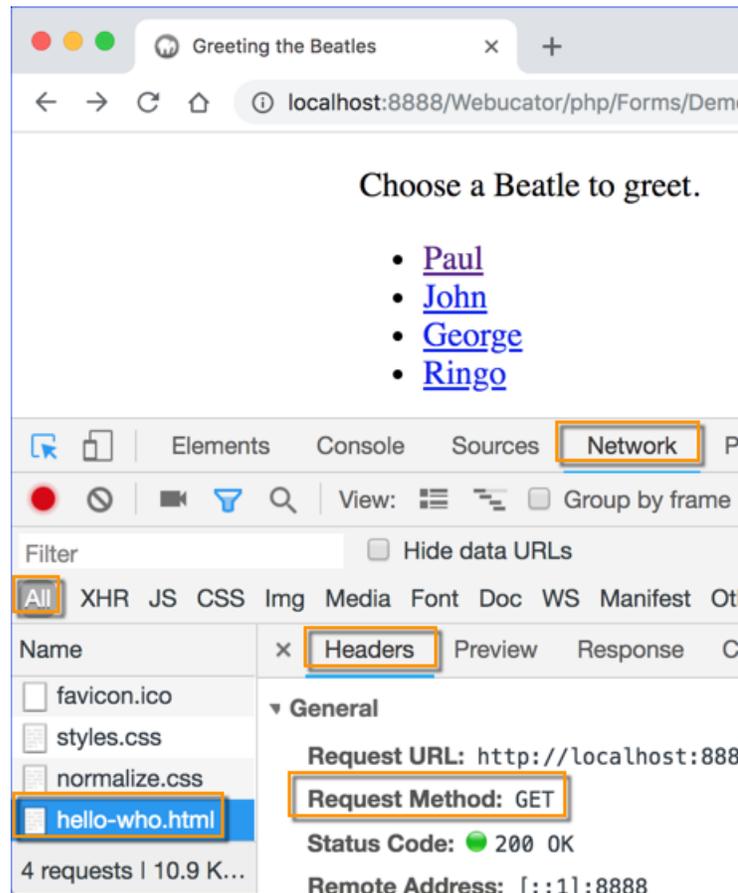
Get Method

When the `get` method is used, data is sent to the server in name-value pairs as part of the query string. The `get` method is most commonly used by search pages and is useful when it is important to be able to bookmark the resulting page (i.e, the page that is returned after the form is submitted).

The `get` method is the default method for delivering all pages, not just action pages of forms. To see this, do the following:

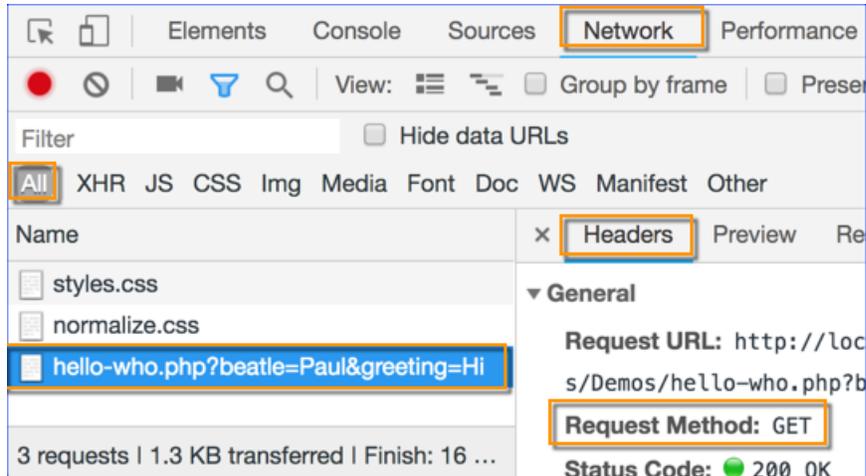
1. Navigate to `http://localhost:8888/Webucator/php/Forms/Demos/` in Google Chrome.
2. Open Chrome DevTools (see page 4) and make sure the **Network** tab is selected.
3. Click `hello-who.html`.

4. In the bottom left of DevTools, click `hello-who.html` and look at the **Headers** on the right side:

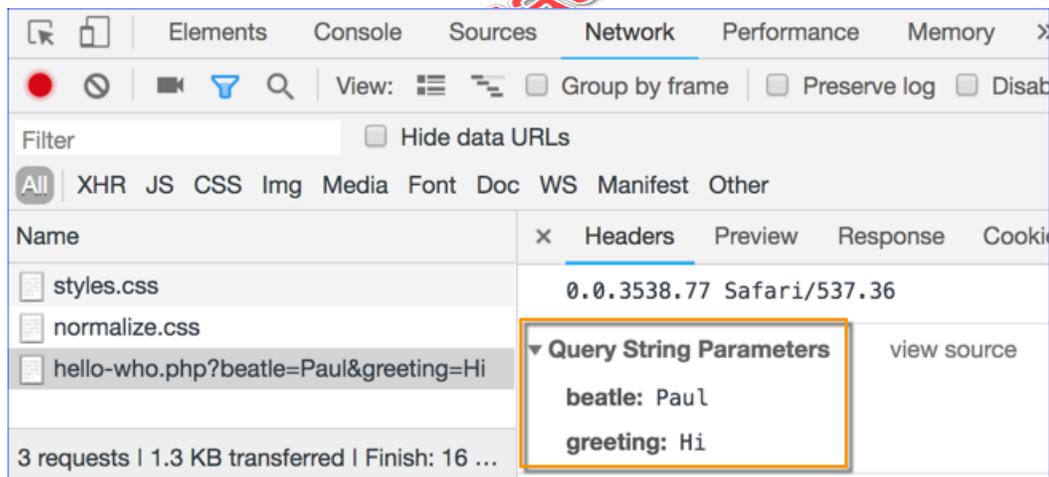


Notice the **Request Method** is GET.

5. Now click one of the links on the page and notice that the **Request Method** for that page is also GET:



- Finally, in the **Headers** section in the bottom right, scroll down to the **Query String Parameters** section and notice that it tells you what parameters are being passed in with the request:



- As we have already seen, we can access those parameters in PHP with `$_GET['beatle']` and `$_GET['greeting']`.

Now take a look at the following code sample:

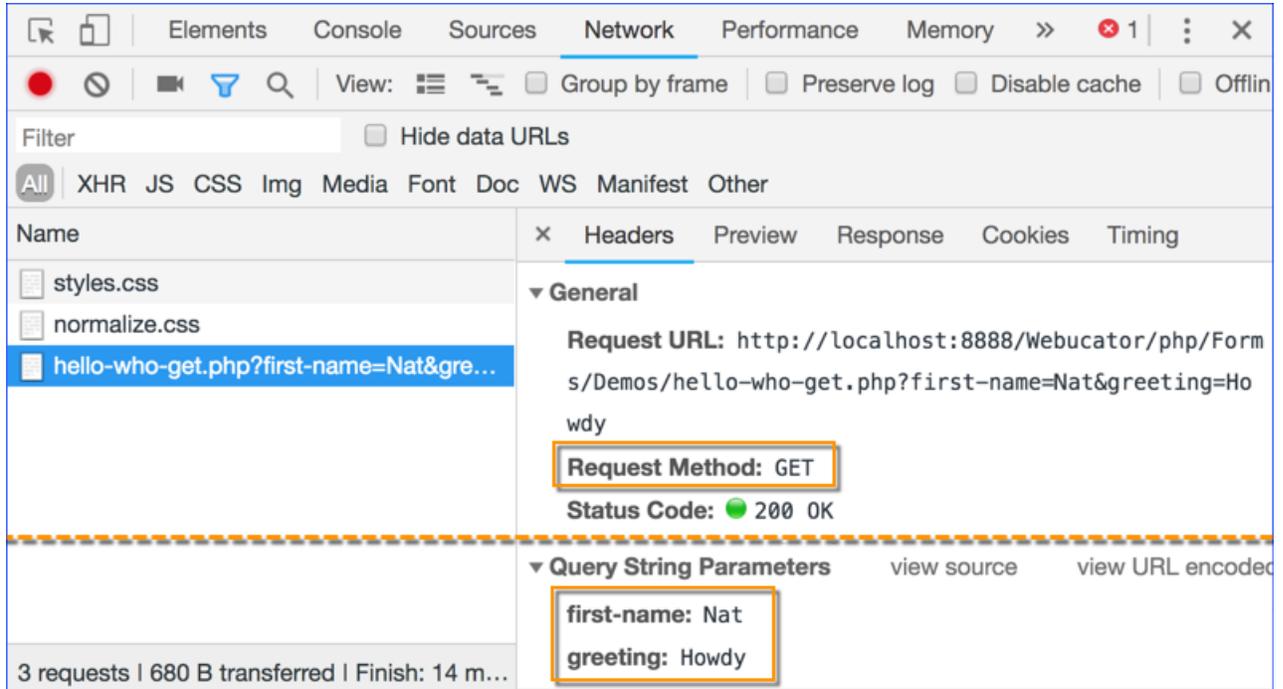
Demo 7.1: Forms/Demos/hello-who-get.html

```
1. <!DOCTYPE HTML>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Greeting Form - GET Method</title>
9. </head>
10. <body>
11. <main>
12. <p>Please fill out the form below.</p>
13. <form method="get" action="hello-who-get.php">
14. <label for="first-name">First Name:</label>
15. <input name="first-name" id="first-name">
16. <label for="greeting">Greeting:</label>
17. <input name="greeting" id="greeting">
18. <button class="wide">Submit</button>
19. </form>
20. </main>
21. </body>
22. </html>
```



Code Explanation

1. Notice that the form uses the `get` method.
2. Navigate to <http://localhost:8888/Webucator/php/Forms/Demos/hello-who-get.html> in Google Chrome.
3. Make sure Chrome DevTools is open to the **Network** tab.
4. Fill out and submit the form.
5. Notice in the **Network** tab that the **Request Method** is GET and that you can see the **Query String Parameters** at the bottom of the **Headers** section:



The important takeaway here is that PHP doesn't know that `hello-who-get.php` was requested via a form. It would return the exact same page if someone entered the following URL directly in the location bar of the browser:

```
http://localhost:8888/Webucator/php/Forms/Demos/hello-who-get.php?first-name=Nat&greeting=Howdy
```

And that is why these pages are bookmarkable and shareable.

Post Method

When the `post` method is used, data is sent to the server in name-value pairs behind the scenes. Take a look at the code below:

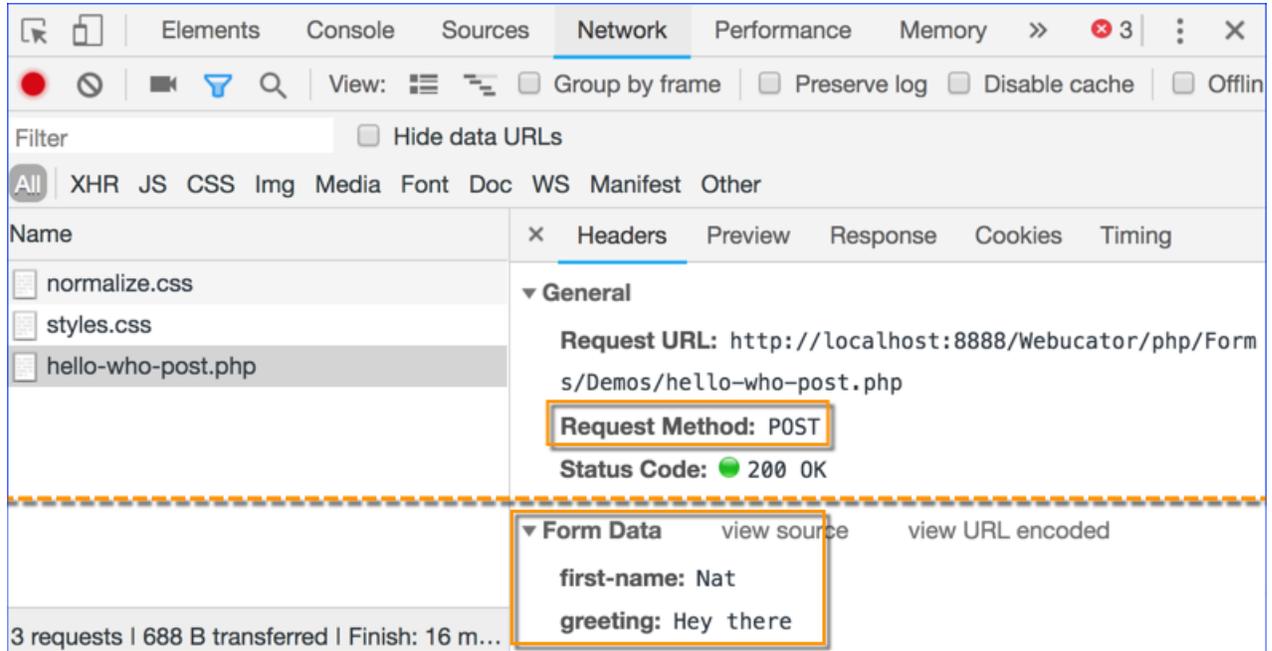
Demo 7.2: Forms/Demos/hello-who-post.html

```
1. <!DOCTYPE HTML>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Greeting Form - POST Method</title>
9. </head>
10. <body>
11. <main>
12. <p>Please fill out the form below.</p>
13. <form method="post" action="hello-who-post.php">
14. <label for="first-name">First Name:</label>
15. <input name="first-name" id="first-name">
16. <label for="greeting">Greeting:</label>
17. <input name="greeting" id="greeting">
18. <button class="wide">Submit</button>
19. </form>
20. </main>
21. </body>
22. </html>
```



Code Explanation

1. The only difference between this form and the last one is that this one uses the post method.
2. Navigate to <http://localhost:8888/Webucator/php/Forms/Demos/hello-who-post.html> in Google Chrome.
3. Make sure Chrome DevTools is open to the **Network** tab.
4. Fill out and submit the form.
5. Notice in the **Network** tab that the **Request Method** is POST and that you can see the **Form Data** at the bottom of the **Headers** section:



The action pages for the get and post forms are almost identical. The only difference is that one gets the `$_GET` parameters and the other gets the `$_POST` parameters:

Demo 7.3: Forms/Demos/hello-who-get.php

```
1.  <?php
2.    $firstName = $_GET['first-name'];
3.    $greeting = $_GET['greeting'];
4.  ?>
5.  <!DOCTYPE HTML>
6.  <html lang="en">
7.  <head>
8.    <meta charset="UTF-8">
9.    <meta name="viewport" content="width=device-width,initial-scale=1">
10.  <link rel="stylesheet" href="../../static/styles/normalize.css">
11.  <link rel="stylesheet" href="../../static/styles/styles.css">
12.  <title><?=$greeting . ', ' $firstName ?></title>
13. </head>
14. <body>
15. <main>
16. <?php
17.   echo "<p>$greeting, $firstName!</p>";
18. ?>
19. </main>
20. </body>
21. </html>
```

Demo 7.4: Forms/Demos/hello-who-post.php

```
1.  <?php
2.    $firstName = $_POST['first-name'];
3.    $greeting = $_POST['greeting'];
4.  ?>
5.  <!DOCTYPE HTML>
6.  <html lang="en">
7.  <head>
8.    <meta charset="UTF-8">
9.    <meta name="viewport" content="width=device-width,initial-scale=1">
10.  <link rel="stylesheet" href="../../static/styles/normalize.css">
11.  <link rel="stylesheet" href="../../static/styles/styles.css">
12.  <title><?= $greeting . ', ' . $firstName ?></title>
13. </head>
14. <body>
15. <main>
16. <?php
17.   echo "<p>$greeting, $firstName!</p>";
18. ?>
19. </main>
20. </body>
21. </html>
```

Use Post for Most Forms

The two major advantages of the `post` method are:

- The name-value pairs are not visible in the location bar, so sensitive data such as passwords are not displayed on the screen.
- Files, such as images and Office documents, can be uploaded via the form.

The major disadvantage is that the resulting page cannot be bookmarked.

As a general rule, you should use `post` unless you want the user to be able to bookmark or share (e.g., via email) the resulting web page.

A Note for Node Users

If you have used Node, you might be used to a routing system that doesn't have a one-to-one relationship between files and URLs (not including the query string). For

example, you might navigate to `http://localhost:8080/hello-world.html` and see a web page, even though there is no `hello-world.html` file on the server. Likewise, you could have a form that submits to `/process`, without any corresponding file named `process` on the server.

While it's possible to set your web server up to behave in a similar way for sites running on PHP, it is not the norm. Typically, with sites running PHP, when you make requests for pages (e.g., `hello-world.html` or `process.php`), there are actual files with those names on the server.



7.2. Form Submissions

The table below describes how data from different types of form fields are sent to the server when a form is submitted. Note that the table assumes that all the fields are enabled. Disabled form fields will not get sent to the server.

How HTML Form Data is Submitted

Field	On Form Submission
text-like and <code>textarea</code>	Always sent. If left blank, an empty string is sent.
<code>select</code>	Always sent.
radio buttons	Only sent if a selection is made.
checkbox	If checked, the default value of <code>on</code> is sent unless otherwise set in the HTML. If not checked, variable is not sent.
submit button	This applies to both <code>input</code> elements of type <code>submit</code> and <code>button</code> elements of type <code>submit</code> (the default). If a submit button has a name, it will be sent along with its <code>value</code> . Note that if there are multiple submit buttons, only the one that is clicked will get sent. If the user submits the form by pressing the Enter key while focus is on a text-like field, the first submit button in the form will be used to submit the form.



7.3. Sanitizing Form Data

When form data is submitted, the data needs to be *sanitized* (cleaned).

The first step to sanitize text entered through a form is to remove any unwanted whitespace before and after the entry. This is done with the `trim()` function. For example:

```
$firstName = trim($_POST['first-name']);
```

This way, if the user enters ' Bruce ' with spaces before and after the name, `$firstName` will get 'Bruce' without the spaces.

Depending on what you plan to do with the data, that may be all you need to do. However, if you plan to output the data to the browser, you must further sanitize it first. The following demo illustrates why:

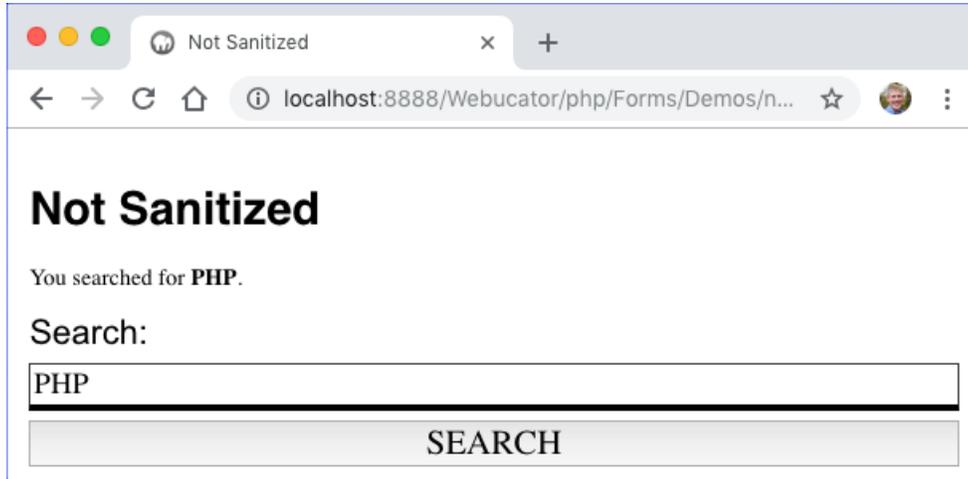
Demo 7.5: Forms/Demos/not-sanitized.php

```
1. <!DOCTYPE HTML>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width,initial-scale=1">
6. <link rel="stylesheet" href="../../static/styles/normalize.css">
7. <link rel="stylesheet" href="../../static/styles/styles.css">
8. <title>Not Sanitized</title>
9. </head>
10. <body>
11. <main>
12. <h1>Not Sanitized</h1>
13. <?php
14.     $q = $_GET['q'] ?? '';
15.     if ($q) {
16.         echo "<p>You searched for <strong>$q</strong>.</p>";
17.     }
18.     ?>
19. <form>
20.     <label for="q">Search:</label>
21.     <input name="q" id="q" value="<?= $q ?>">
22.     <button class="wide">SEARCH</button>
23. </form>
24. </main>
25. </body>
26. </html>
```

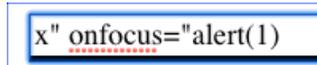
Code Explanation

This page simply outputs the user input back to the text field. This is a common practice when the user submission contains errors. Rather than making the user fill out the form from scratch, we put the user's original entries back into the form. To see this:

1. Navigate to <http://localhost:8888/Webucator/php/Forms/Demos/not-sanitized.php>.
2. Enter a search query and press **SEARCH**.
3. Your query will show up in the search box, like this:



4. That looks fine. Now enter `x" onfocus="alert(1)` like this:



And press **SEARCH**.

5. The PHP page will process the following:

```
<input name="q" id="q" value="<?= $q ?>">
```

... resulting in:

```
<input name="q" id="q" value="x" onfocus="alert(1)">
```

Notice that JavaScript has been written into the page. Although this JavaScript is relatively harmless, more nefarious hackers could attempt to inject something more dangerous that uses cross-site scripting (XSS) to do something more dangerous.

6. Some modern browsers, like Google Chrome, will refuse to show the resulting page. They will recognize that unsafe data from the form is being used in the code. Safari will show the page, but will not execute the JavaScript. But you, as the developer, should not rely on the browsers to catch these attempts for you. You need to sanitize your data on the server-side to prevent that code getting returned to the browser.

PHP provides many options for sanitizing data. A few of the most useful are listed below:

❖ 7.3.1. htmlspecialchars()

The `htmlspecialchars()` function converts `<`, `>`, `&`, and `"` to `<`, `>`, `&`, and `"`, respectively. Notably, it does not convert an apostrophe (`'`) to an HTML entity by default. However, you can force it to do so by passing in the built-in constant `ENT_QUOTES` as a second argument:

```
htmlspecialchars("'Hello, world!'", ENT_QUOTES);  
// &#039;Hello, world!&#039;
```

❖ 7.3.2. htmlentities()

The `htmlentities()` function is the same as `htmlspecialchars()` except that it also converts characters that have equivalent HTML entities to their entity equivalents. For example, it converts `©` to `©`. Like with `htmlspecialchars()`, `htmlentities()` does not convert an apostrophe (`'`) to an HTML entity by default. However, you can force it to do so by passing in the built-in constant `ENT_QUOTES` as a second argument:

```
htmlentities("'Hello, world!'", ENT_QUOTES);  
// &#039;Hello, world!&#039;
```

❖ 7.3.3. filter_var()

The `filter_var()` function is used to sanitize and validate data. There are many built-in constants¹⁵ for sanitizing and validating different types of data. For making strings safe to output to the browser, the most useful of these is `FILTER_SANITIZE_STRING`. Note that this will completely remove less than and greater than signs, which may or may not be what you want.

```
$q = filter_var($_GET['q'], FILTER_SANITIZE_STRING);
```

If `q` does not exist in the `$_GET` array, the code above will generate an `Undefined index` notice. This won't break anything and, depending on your error-display settings, may go completely unnoticed, but it's better practice not to reference variables unless they have been defined. As such, you may want to nest this in a condition:

15. <https://www.php.net/manual/en/filter.filters.php>

```

if (isset($_GET['q'])) {
    $q = filter_var($_GET['q'], FILTER_SANITIZE_STRING);
}

```

❖ 7.3.4. filter_input()

The `filter_input()` function is similar to `filter_var()`, but it is specifically used with the built-in superglobals, most commonly with `$_GET` and `$_POST`. For example, the code below would assign a sanitized value of `$_GET['q']` to `$q`. Like `filter_var()`, this will completely remove less than and greater than signs, which may or may not be what you want.

```
$q = filter_input(INPUT_GET, 'q', FILTER_SANITIZE_STRING);
```

Unlike `filter_var()`, if `q` does not exist in the `$_GET` array, `filter_input()` will not generate a notice. Instead it will return `NULL`, which conveniently converts to an empty string when treated as a string.

The table below shows what these functions will output if passed `$_GET['q']` when it contains the following string:

```
© " ' < >
```

Function	Output
<code>htmlspecialchars(\$_GET['q'])</code>	© " ' < >
<code>htmlentities(\$_GET['q'])</code>	© " ' < >
<code>htmlentities(\$_GET['q'], ENT_QUOTES)</code>	© " ' < >
<code>filter_var(\$_GET['q'], FILTER_SANITIZE_STRING)</code>	© " '
<code>filter_input(INPUT_GET, 'q', FILTER_SANITIZE_STRING)</code>	© " '

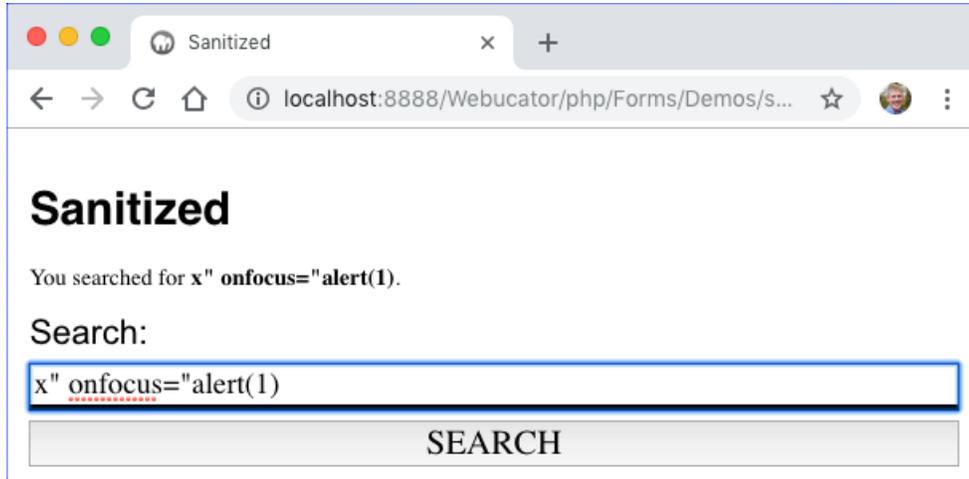
The option you choose depends on what you intend to do with the sanitized data. If you're just outputting the data to the form fields, `htmlentities($_GET['q'], ENT_QUOTES)` is a good option. Here is the code we saw earlier, but this time we sanitize the submitted value:

Demo 7.6: Forms/Demos/sanitized.php

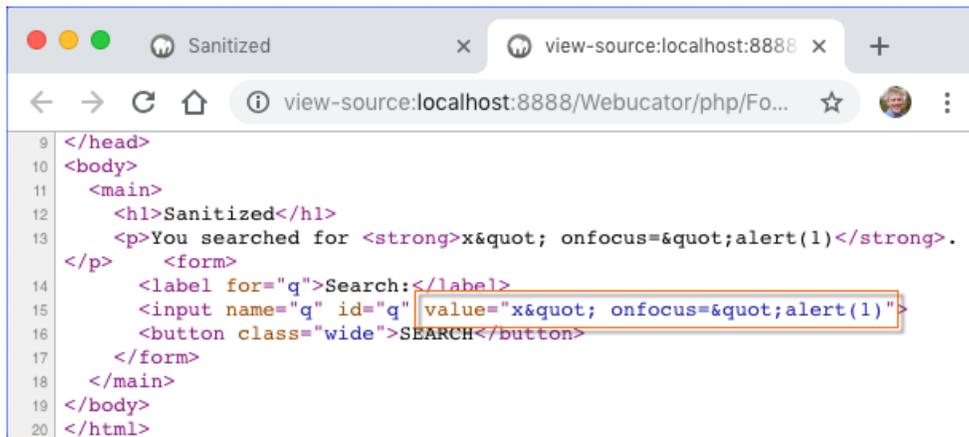
```
1.  <!DOCTYPE HTML>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>Sanitized</title>
9.  </head>
10. <body>
11.   <main>
12.     <h1>Sanitized</h1>
13.     <?php
14.       $q = $_GET['q'] ?? '';
15.       $q = htmlentities($q, ENT_QUOTES);
16.       if ($q) {
17.         echo "<p>You searched for <strong>$q</strong>.</p>";
18.       }
19.     ?>
20.     <form>
21.       <label for="q">Search:</label>
22.       <input name="q" id="q" value="<?= $q ?>">
23.       <button class="wide">SEARCH</button>
24.     </form>
25.   </main>
26. </body>
27. </html>
```

Code Explanation

Visit <http://localhost:8888/Webucator/php/Forms/Demos/sanitized.php> and search for `x" onfocus="alert(1) again:`



This time, it replaces the quotation marks with the " ; entity. This is easier to see if you view the source of the page:



7.4. Validating Form Data

After sanitizing the form data, the next step is to validate it. A common practice is to validate all the fields that require validation and store any errors found in an array. To do so, start by creating an empty \$errors array:

```
$errors = [];
```

❖ 7.4.1. Was the Field Filled In?

Often you will simply want to validate that the user entered a value. There are a variety of ways to do this.

Treating the Variable as a Boolean

If you know the variable exists, you can simply treat it as a boolean:

```
if (!$q) {...
```

If you have already assigned a value to `$q` then this technique is fine. One approach, which we will use in the upcoming exercise, is to store sanitized values submitted via the form in a new array, like this:

```
$f = []; // f for form
$f['q'] = trim($_POST['q'] ?? '');
```

Notice that this uses the null coalescing operator (`??`) to assign an empty string to `$f['q']` if `$_POST['q']` does not exist.

Once we have done this, we can then safely treat `$f['q']` as a boolean:

```
if (!$f['q']) {
    $errors[] = 'You must include a search value.';
}
```

When finished checking all the fields, you can then check to see if there are any errors by treating `$errors` as a boolean. Empty arrays are falsy:

```
if ($errors) {
    // Show form errors
}
```

❖ 7.4.2. Is the Entered Value an Integer?

In addition to sanitizing data the `filter_var()` and `filter_input()` functions can validate data using the `FILTER_VALIDATE` constants. Both functions will return `false` if the validation fails and will return the filtered value of the variable if it succeeds.

The `FILTER_VALIDATE_INT` filter will check to see if the variable represents an integer (e.g., "1", which is a string, but can be converted to an integer). If it does represent an integer, it will convert the value to an integer type and return it. If it is not an integer, it will return `false`, which, like `NULL`, conveniently converts to an empty string when treated as a string.

```
$f['age'] = filter_input(INPUT_POST, 'age', FILTER_VALIDATE_INT);
```

After running the code above, you can just treat `$f['age']` as a boolean:

```
if (!$f['age']) {  
    $errors[] = 'You must include a valid age.';  
}
```

Evaluation
Copy

There is a gotcha here though! Remember that `0` is *falsey*, meaning that it converts to `false` when treated as a boolean. If `0` is an acceptable value for `$age`, the code above will not work correctly. Also, presumably negative ages would not be valid. As such, you should do something like this:¹⁶

```
if (!is_int($i) || $i < 0) {  
    $errors[] = 'You must include a valid age.';  
}
```

❖ 7.4.3. Is it an Email?

We can use the `filter_var()` and `filter_input()` functions to check for a valid email just as we did to check for a valid integer above:

```
$f['email'] = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
```

16. See https://www.php.net/is_int for details on the `is_int()` function.

However, if you want to return one error if the email is missing and a different error if the email is present but invalid, you can take the following approach:

1. Save the submitted email in the `$f` array:

```
$f['email'] = trim($_POST['email'] ?? '');
```

2. Check if the email is *falsey*, most likely meaning that nothing was entered in the form field, and then check if it the email is valid:

```
if (!$f['email']) {  
    $errors[] = 'Email is required.';  
} elseif (!filter_var($f['email'], FILTER_VALIDATE_EMAIL)) {  
    $errors[] = 'Email is not valid.';  
}
```

❖ 7.4.4. Is it a Valid Password and Do the Passwords Match?

You should not trim or filter passwords. You should use exactly what the user entered. And, for security reasons, you should never return the password to the browser. Websites use all sorts of different algorithms for checking passwords, but for starters, you want to make sure the password is a minimum length and that the two passwords entered when registering for a site are equal. The following check will do the trick:

```
if (!$_POST['password-1']) {  
    $errors[] = 'Password is required.';  
} elseif (strlen($_POST['password-1']) < 8) {  
    $errors[] = 'Password must be at least 8 characters.';  
} elseif ($_POST['password-1'] !== $_POST['password-2']) {  
    $errors[] = 'Passwords do not match.';  
}
```

❖ 7.4.5. Do the Combined Values Create a Valid Date?

Often forms use separate fields for month, day, and year values. In this case, each should be an integer and should be checked separately using the technique we described above. After checking that they are all integers, you can also use PHP's `checkdate()` function to see if they combine to make a valid date:

```
if (!is_int($f['birth-day'])
    || !is_int($f['birth-month'])
    || !is_int($f['birth-year'])) {
    $errors[] = 'A full birth date is required.';
} elseif ( !checkdate($f['birth-month'],
                    $f['birth-day'],
                    $f['birth-year']) ) {
    $errors[] = 'The birth date is not valid.';
}
```

❖ 7.4.6. Did the User Check the Box?

The name-value pair associated with a checkbox only gets submitted if the checkbox is checked. So, to check if a user checked a checkbox, you just need to check whether the key exists in `$_POST`:

```
if (!isset($_POST['terms'])) {
    errors[] = 'You must agree to our terms of use.';
}
```

Exercise 24: Processing Form Input

 60 to 90 minutes

In this exercise, you will create a page that sanitizes and validates form data. The form entry code is already complete:

Exercise Code 24.1: Forms/Exercises/add-employee.php

```
1.  <?php
2.    ini_set('display_errors', '1');
3.
4.    // Used to populate birth-month and hire-month fields
5.    $months = [
6.        'January', 'February', 'March', 'April', 'May', 'June', 'July',
7.        'August', 'September', 'October', 'November', 'December'
8.    ];
9.
10.   // Used to populate courtesy-title field
11.   $courtesyTitles = [ 'Dr.', 'Mr.', 'Mrs.', 'Ms.' ];
12.   ?>
13.   <!DOCTYPE HTML>
14.   <html lang="en">
15.   <head>
16.   <meta charset="UTF-8">
17.   <meta name="viewport" content="width=device-width,initial-scale=1">
18.   <link rel="stylesheet" href="../../static/styles/normalize.css">
19.   <link rel="stylesheet" href="../../static/styles/styles.css">
20.   <title>Add Employee</title>
21.   </head>
22.   <body>
23.   <main>
24.     <h1>Add Employee</h1>
25.     <form method="post" action="add-employee.php" novalidate>
26.       <label for="first-name">First Name:</label>
27.       <input name="first-name" id="first-name">
28.       <label for="last-name">Last Name:</label>
29.       <input name="last-name" id="last-name">
30.       <label for="email">Email:</label>
31.       <input type="email" name="email" id="email">
32.       <fieldset>
33.         <legend>Password:</legend>
34.         <input type="password" placeholder="Password"
35.             name="password-1" id="password-1">
36.         <input type="password" placeholder="Repeat Password"
37.             name="password-2" id="password-2">
38.       </fieldset>
39.       <label for="title">Title:</label>
40.       <input name="title" id="title">
41.       <fieldset>
42.         <legend>Title of Courtesy:</legend>
43.         <?php
44.           foreach ($courtesyTitles as $title) {
```

Evaluation
Copy

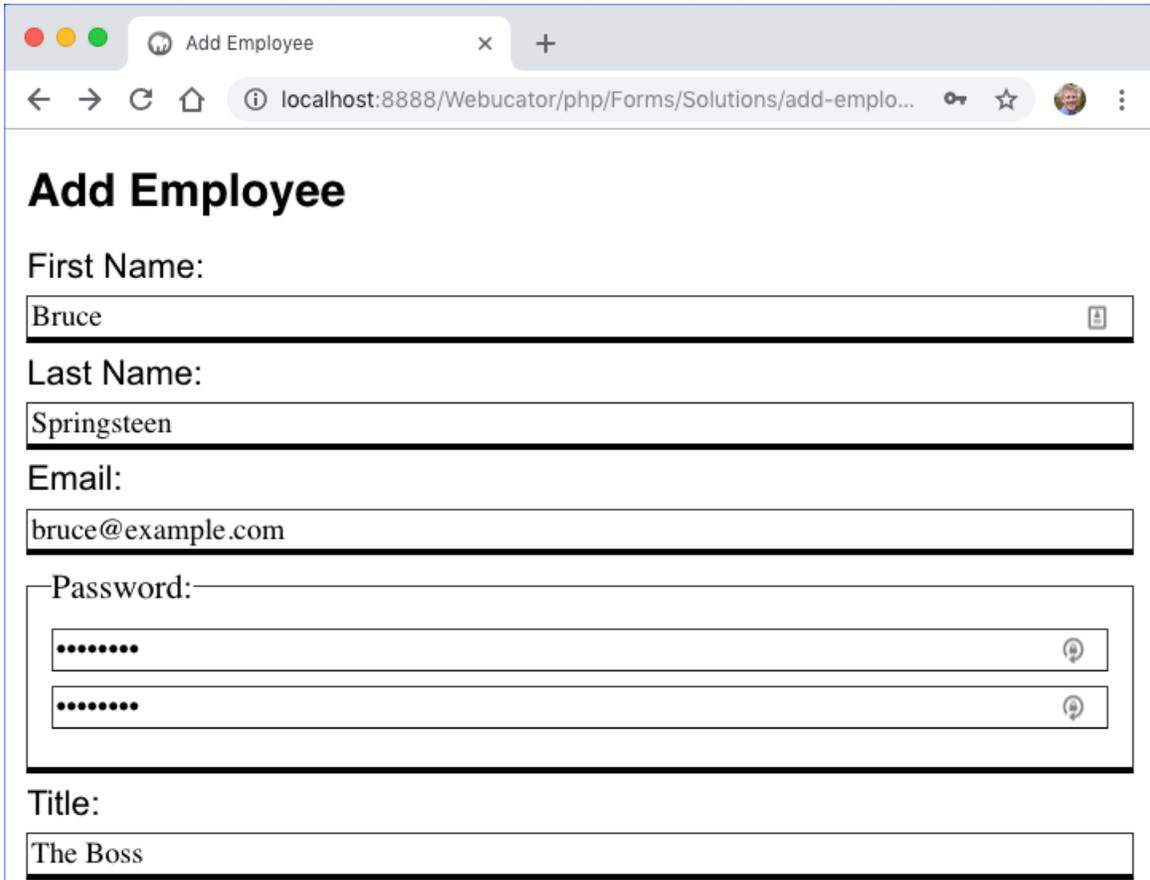
```

45.         echo "<label>
46.         <input type='radio' name='courtesy-title' value='$title'>
47.         $title</label>";
48.     }
49.     ?>
50. </fieldset>
51. <fieldset>
52.     <legend>Birth date:</legend>
53.     <select name="birth-month" id="birth-month">
54.         <option value="0">--Select Month--</option>
55.         <?php
56.             for ($i=1; $i<=12; $i++) {
57.                 echo "<option value='$i'>" . $months[$i-1] . "</option>";
58.             }
59.         ?>
60.     </select>
61.     <input name="birth-day" type="number" min="1" max="31"
62.         placeholder="day">
63.     <input name="birth-year" type="number" placeholder="year">
64. </fieldset>
65. <fieldset>
66.     <legend>Hire date:</legend>
67.     <select name="hire-month">
68.         <option value="0">--Select Month--</option>
69.         <?php
70.             for ($i=1; $i<=12; $i++) {
71.                 echo "<option value='$i'>" . $months[$i-1] . "</option>";
72.             }
73.         ?>
74.     </select>
75.     <input name="hire-day" type="number" min="1" max="31"
76.         placeholder="day">
77.     <input name="hire-year" type="number" placeholder="year">
78. </fieldset>
79. <label for="cell-phone">Cell Phone:</label>
80. <input type="tel" name="cell-phone" id="cell-phone">
81. <label for="notes">Notes:</label>
82. <textarea name="notes" id="notes"></textarea>
83. <button name="add-employee" class="wide">Add Employee</button>
84. </form>
85. </main>
86. </body>
87. </html>

```

Code Explanation

The code outputs an HTML form for adding a new employee. Its action page is `add-employee.php`, which means the page will submit to itself. The filled out form looks like this (broken into two parts):



A screenshot of a web browser window titled "Add Employee". The address bar shows the URL `localhost:8888/Webucator/php/Forms/Solutions/add-emplo...`. The form content is as follows:

Add Employee

First Name:

Last Name:

Email:

Password:

Title:

Title of Courtesy: Dr. Mr. Mrs. Ms.

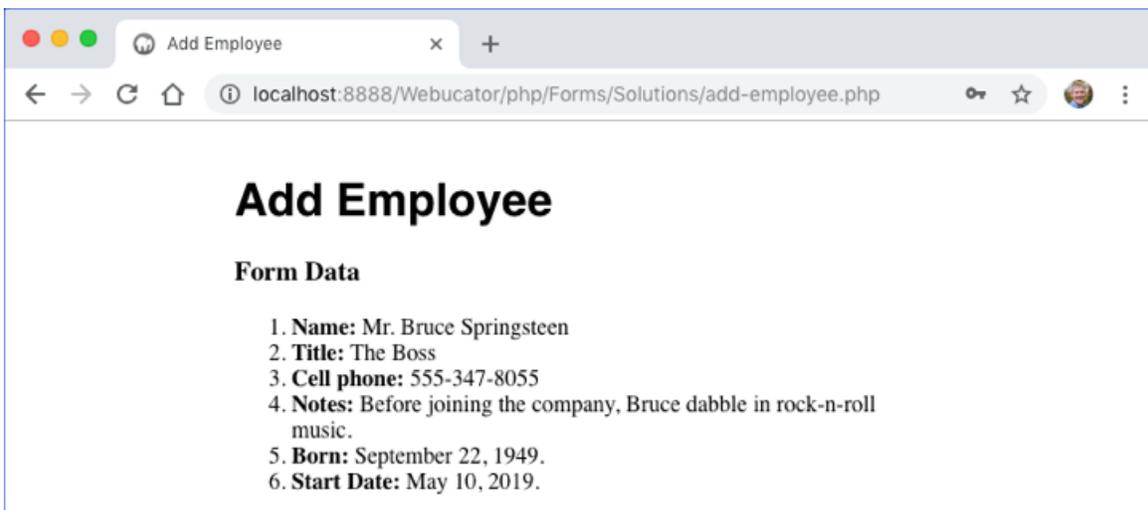
Birth date:

Hire date:

Cell Phone:

Notes:

If everything is filled out correctly, the page should display as follows:



If all fields are left blank, errors should show up at the top of the page:

Add Employee

Please correct the following errors:

1. Email is required.
2. Password is required.
3. First name is required.
4. Last name is required.
5. Title is required.
6. Title of Courtesy is required.
7. A full birth date is required.
8. A full hire date is required.
9. Cell phone is required.

First Name:

Last Name:

Email:

1. Open Forms/Exercises/add-employee.php in your editor.
2. Below the declaration of \$courtesyTitles, declare \$f and assign it an empty array. This will hold our cleaned-up form variables.
3. Trim and assign all the text-like form entries to values within the \$f array. Use the null coalescing operator to assign an empty string if the form variable doesn't exist. For example:

```
$f['first-name'] = trim($_POST['first-name'] ?? '');
```

4. Use filter_input() to validate that all the date parts represent integers and to convert them from strings to integers. For example:

```
$f['birth-day'] = filter_input(INPUT_POST, 'birth-day',  
    FILTER_VALIDATE_INT);
```

5. If the form has been submitted, add-employee (the name of the submit button) will be present in the \$_POST array. Write an if condition to check for this. Within this if block, you'll do your form validation.

- A. Declare `$errors` and assign it an empty array and populate `$errors` with error messages:
- B. If the email isn't filled out, add: 'Email is required.'
- C. If the email isn't valid, add: 'Email is not valid.'
- D. If the password isn't filled out, add: 'Password is required.'
- E. If the password is shorter than 8 characters, add: 'Password must be at least 8 characters.'
- F. If the passwords do not match, add: 'Passwords do not match.'
- G. For each of the following fields that is not filled out, add 'XYZ is required.' For example, 'First name is required.'
 - i. First name
 - ii. Last name
 - iii. Title
 - iv. Title of courtesy
 - v. Cell phone
- H. Add code to check that the birth date and hire date fields are all filled out and create valid dates.
- I. If the notes field is longer than 100 characters, add 'Notes cannot be longer than 100 characters.'

6. Below the "Add Employee" h1 element, create a PHP block.

- A. If there are any errors, output `<h3>Please correct the following errors:</h3>` followed by an ordered list of error messages:

Please correct the following errors:

1. Email is required.
2. Password is required.
3. First name is required.
4. Last name is required.
5. Title is required.
6. Title of Courtesy is required.
7. A full birth date is required.
8. A full hire date is required.
9. Cell phone is required.

- B. If there are no errors and the form has been submitted, output `<h3>Form Data</h3>` followed by an ordered list of the data:

Form Data

1. **Name:** Mr. Bruce Springsteen
2. **Title:** The Boss
3. **Cell phone:** 555-347-8055
4. **Notes:** Before joining the company, Bruce dabbled in rock-n-roll music.
5. **Born:** September 22, 1949.
6. **Start Date:** May 10, 2019.

We would normally insert the data into a database, but for our purposes right now displaying the data in the browser is enough.

- C. If there are no errors and the form has been submitted, output the form. You should end your PHP block with the if statement and then add another PHP block after the form to close the if statement:

```
<?php
}
?>
```

7. Add code to all the text-like fields so that they “remember” values submitted through the form. For example:

```
<input name="first-name" id="first-name"
value="<?= $f['first-name'] ?>">
```

8. Add code to the for loops for the select menus (using the selected attribute) and radio buttons (using the checked attribute) to “remember” values submitted through the form.
9. Navigate to <http://localhost:8888/Webucator/php/Forms/Exercises/add-employee.php> in the browser to test your solution:
 - A. Submit it without entering any data.
 - B. Submit it with invalid data (e.g., an invalid email address, passwords that are too short or don’t match, a date that isn’t real like February 31).
 - C. Submit it with valid data.

Solution: Forms/Solutions/add-employee.php

```
1.  <?php
2.    ini_set('display_errors', '1');
3.
4.    // Used to populate birth-month and hire-month fields
5.    $months = [
6.        'January', 'February', 'March', 'April', 'May', 'June', 'July',
7.        'August', 'September', 'October', 'November', 'December'
8.    ];
9.
10.   // Used to populate courtesy-title field
11.   $courtesyTitles = [ 'Dr.', 'Mr.', 'Mrs.', 'Ms.' ];
12.
13.   $f = [];
14.
15.   // Trim and Assign Form Entries
16.   $f['first-name'] = trim($_POST['first-name'] ?? '');
17.   $f['last-name'] = trim($_POST['last-name'] ?? '');
18.   $f['email'] = trim($_POST['email'] ?? '');
19.   $f['title'] = trim($_POST['title'] ?? '');
20.   $f['courtesy-title'] = trim($_POST['courtesy-title'] ?? '');
21.   $f['birth-month'] = filter_input(INPUT_POST, 'birth-month',
22.       FILTER_VALIDATE_INT);
23.   $f['birth-day'] = filter_input(INPUT_POST, 'birth-day',
24.       FILTER_VALIDATE_INT);
25.   $f['birth-year'] = filter_input(INPUT_POST, 'birth-year',
26.       FILTER_VALIDATE_INT);
27.   $f['hire-month'] = filter_input(INPUT_POST, 'hire-month',
28.       FILTER_VALIDATE_INT);
29.   $f['hire-day'] = filter_input(INPUT_POST, 'hire-day',
30.       FILTER_VALIDATE_INT);
31.   $f['hire-year'] = filter_input(INPUT_POST, 'hire-year',
32.       FILTER_VALIDATE_INT);
33.   $f['cell-phone'] = trim($_POST['cell-phone'] ?? '');
34.   $f['notes'] = trim($_POST['notes'] ?? '');
35.
36.   if (isset( $_POST['add-employee'])) {
37.       $errors = [];
38.
39.       if (!$f['email']) {
40.           $errors[] = 'Email is required.';
41.       } elseif (!filter_var($f['email'], FILTER_VALIDATE_EMAIL)) {
42.           $errors[] = 'Email is not valid.';
43.       }
44.   }
```

```

45.     if (!$_POST['password-1']) {
46.         $errors[] = 'Password is required.';
47.     } elseif (strlen($_POST['password-1']) < 8) {
48.         $errors[] = 'Password must be at least 8 characters.';
49.     } elseif ($_POST['password-1'] != $_POST['password-2']) {
50.         $errors[] = 'Passwords do not match.';
51.     }
52.
53.     if (!$f['first-name']) {
54.         $errors[] = 'First name is required.';
55.     }
56.
57.     if (!$f['last-name']) {
58.         $errors[] = 'Last name is required.';
59.     }
60.
61.     if (!$f['title']) {
62.         $errors[] = 'Title is required.';
63.     }
64.
65.     if (!$f['courtesy-title']) {
66.         $errors[] = 'Title of Courtesy is required.';
67.     }
68.
69.     if (!$f['birth-day'] || !$f['birth-month'] || !$f['birth-year']) {
70.         $errors[] = 'A full birth date is required.';
71.     } elseif ( !checkdate($f['birth-month'],
72.                            $f['birth-day'],
73.                            $f['birth-year']) ) {
74.         $errors[] = 'The birth date must be a valid date.';
75.     }
76.
77.     if (!$f['hire-day'] || !$f['hire-month'] || !$f['hire-year']) {
78.         $errors[] = 'A full hire date is required.';
79.     } elseif ( !checkdate($f['hire-month'],
80.                            $f['hire-day'],
81.                            $f['hire-year']) ) {
82.         $errors[] = 'The hire date must be a valid date.';
83.     }
84.
85.     if (!$f['cell-phone']) {
86.         $errors[] = 'Cell phone is required.';
87.     }
88.
89.     if (strlen($f['notes']) > 100) {

```

```

90.     $errors[] = 'Notes cannot be longer than 100 characters.';
91.     }
92.     }
93.     ?>
94.     <!DOCTYPE HTML>
95.     <html lang="en">
96.     <head>
97.     <meta charset="UTF-8">
98.     <meta name="viewport" content="width=device-width,initial-scale=1">
99.     <link rel="stylesheet" href="../../static/styles/normalize.css">
100.    <link rel="stylesheet" href="../../static/styles/styles.css">
101.    <title>Add Employee</title>
102.    </head>
103.    <body>
104.    <main>
105.        <h1>Add Employee</h1>
106.        <?php
107.            if (!empty($errors)) {
108.                // Show form errors
109.                echo '<h3>Please correct the following errors:</h3>';
110.                <ol class="error">;
111.                foreach ($errors as $error) {
112.                    echo "<li>$error</li>";
113.                }
114.                echo '</ol>';
115.            } elseif (isset( $_POST['add-employee'])) {
116.                // We'd normally insert the data into a database here,
117.                // but we will just show the form data.
118.                echo '<h3>Form Data</h3>';
119.                echo '<ol>';
120.                echo '<li><strong>Name:</strong> ' . $f['courtesy-title'] .
121.                    ' ' . $f['first-name'] . ' ' . $f['last-name'] . '</li>';
122.                echo '<li><strong>Title:</strong> ' . $f['title'] . '</li>';
123.                echo '<li><strong>Cell phone:</strong> ' .
124.                    $f['cell-phone'] . '</li>';
125.                echo '<li><strong>Notes:</strong> ' . $f['notes'] . '</li>';
126.
127.                $birthDate = mktime(0, 0, 0, $f['birth-month'],
128.                    $f['birth-day'], $f['birth-year']);
129.                $sBirthDate = date("F j, Y", $birthDate);
130.                echo "<li><strong>Born:</strong> $sBirthDate.</li>";
131.
132.                $hireDate = mktime(0, 0, 0, $f['hire-month'],
133.                    $f['hire-day'], $f['hire-year']);
134.                $sHireDate = date("F j, Y", $hireDate);

```

Evaluation Copy

```

135.     echo "<li><strong>Start Date:</strong> $sHireDate.</li>";
136.     echo '</ol>';
137. }
138.
139.     if (!empty($errors) || !isset( $_POST['add-employee'])) {
140.         // Show form
141.     ?>
142. <form method="post" action="add-employee.php" novalidate>
143.     <label for="first-name">First Name:</label>
144.     <input name="first-name" id="first-name"
145.         value="<?= $f['first-name'] ?>">
146.     <label for="last-name">Last Name:</label>
147.     <input name="last-name" id="last-name"
148.         value="<?= $f['last-name'] ?>">
149.     <label for="email">Email:</label>
150.     <input type="email" name="email" id="email"
151.         value="<?= $f['email'] ?>">
152.     <fieldset>
153.         <legend>Password:</legend>
154.         <input type="password" placeholder="Password"
155.             name="password-1" id="password-1">
156.         <input type="password" placeholder="Repeat Password"
157.             name="password-2" id="password-2">
158.     </fieldset>
159.     <label for="title">Title:</label>
160.     <input name="title" id="title" value="<?= $f['title'] ?>">
161.     <fieldset>
162.         <legend>Title of Courtesy:</legend>
163.         <?php
164.             foreach ($courtesyTitles as $cTitle) {
165.                 echo "<label>
166.                     <input type='radio' name='courtesy-title' value='$cTitle'";
167.                     if ($cTitle === $f['courtesy-title']) {
168.                         echo ' checked';
169.                     }
170.                     echo ">$cTitle</label>";
171.                 }
172.         ?>
173.     </fieldset>
174.     <fieldset>
175.         <legend>Birth date:</legend>
176.         <select name="birth-month" id="birth-month">
177.             <option value="0">--Select Month--</option>
178.             <?php
179.                 for ($i=1; $i<=12; $i++) {

```

```

180.         echo "<option value='\$i'";
181.         if ( \$f['birth-month'] === \$i) {
182.             echo " selected";
183.         }
184.         echo ">" . \$months[\$i-1] . "</option>";
185.     }
186.     ?>
187. </select>
188. <input name="birth-day" type="number" min="1" max="31"
189.     placeholder="day" value="<?=\$f['birth-day'] ?>">
190. <input name="birth-year" type="number"
191.     placeholder="year" value="<?=\$f['birth-year'] ?>">
192. </fieldset>
193. <fieldset>
194.     <legend>Hire date:</legend>
195.     <select name="hire-month">
196.         <option value="0">--Select Month--</option>
197.         <?php
198.             for (\$i=1; \$i<=12; \$i++) {
199.                 echo "<option value='\$i'";
200.                 if ( \$f['hire-month'] === \$i) {
201.                     echo " selected";
202.                 }
203.                 echo ">" . \$months[\$i-1] . "</option>";
204.             }
205.             ?>
206.     </select>
207.     <input name="hire-day" type="number" min="1" max="31"
208.     placeholder="day" value="<?=\$f['hire-day'] ?>">
209.     <input name="hire-year" type="number"
210.     placeholder="year" value="<?=\$f['hire-year'] ?>">
211. </fieldset>
212. <label for="cell-phone">Cell Phone:</label>
213. <input type="tel" name="cell-phone" id="cell-phone"
214.     value="<?=\$f['cell-phone'] ?>">
215. <label for="notes">Notes:</label>
216. <textarea name="notes" id="notes"><?=\$f['notes'] ?></textarea>
217. <button name="add-employee" class="wide">Add Employee</button>
218. </form>
219. <?php
220.     }
221.     ?>
222. </main>
223. </body>
224. </html>

```

Code Explanation

There is a significant amount of code here. If you're having trouble understanding any of it, go back through the exercise instructions slowly and methodically, matching each instruction with the relevant code in the solution.

Evaluation
Copy

Conclusion

In this lesson, you have learned how to process and validate forms.

LESSON 8

Sending Email with PHP

Topics Covered

- PHP's built-in `mail()` function.
- PHPMailer

Introduction

In this lesson, you will learn to send emails using PHP's built-in `mail()` function, and to send email using PHPMailer, a PHP extension with more features than `mail()`.



8.1. mail()

PHP has a built-in `mail()` function, which returns `true` on success and `false` on failure. Note that the function can't actually know if the email was sent. It only knows if it was successfully able to hand off the task to the mail server. Also note that for `mail()` to work, you need to have a mail server configured.

mail() Parameters

Parameter	Description
<code>\$to</code>	The address to send the email to.
<code>\$subject</code>	The email's subject.
<code>\$message</code>	The body of the email.
<code>\$additional_headers</code>	Optional. An array of additional headers (e.g, From, Reply-To)
<code>\$additional_parameters</code>	Optional. An array of any additional parameters you may want to send to your mail server.

Demo 8.1: Email/Demos/mail.php

```
1.  <?php
2.      ini_set('display_errors', '1');
3.  ?>
4.  <!DOCTYPE html>
5.  <html lang="en">
6.  <head>
7.  <meta charset="UTF-8">
8.  <meta name="viewport" content="width=device-width,initial-scale=1">
9.  <link rel="stylesheet" href="../../static/styles/normalize.css">
10. <link rel="stylesheet" href="../../static/styles/styles.css">
11. <title>Mail()</title>
12. </head>
13. <body>
14. <main>
15. <?php
16.     $to = 'somebody@example.com';
17.     $headers = array('From' => 'you@youremail.com');
18.     $subject = 'Test Email';
19.     $message = 'Test Message';
20.
21.     if(mail($to,$subject,$message,$headers)) {
22.         echo "Message Sent (Maybe)";
23.     } else {
24.         echo "Message Not Sent";
25.     }
26. ?>
27. </main>
28. </body>
29. </html>
```



Code Explanation

Visit <http://localhost:8888/Webucator/php/Email/Demos/mail.php> to run this page. For this example to work, you will need to have a mail server configured.

This probably won't actually send an email. Don't worry if it doesn't. You may be able to use this when you are using a production server, but you're probably going to want to do it differently. We're going to show you a better way to send emails next.

❖ 8.1.1. Shortcomings of mail()

The `mail()` function has many limitations.

- No support for SMTP authentication.
- Difficult to send HTML-formatted emails.
- Difficult to add attachments.

Luckily, there are extensions that do provide these features.

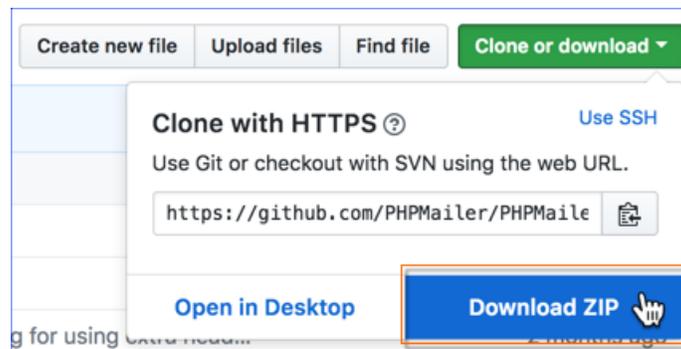


8.2. Setting Up PHPMailer

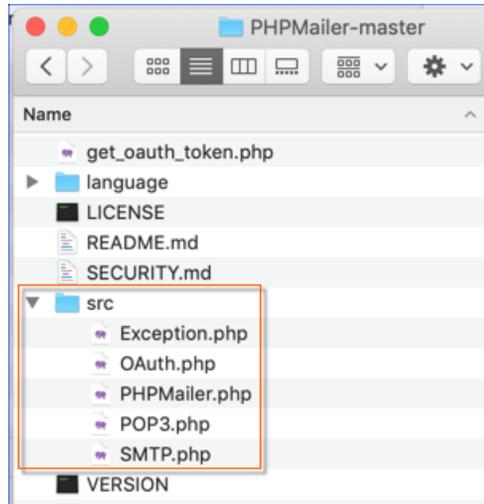
A very good email extension is *PHPMailer*, which is available for free at <https://github.com/PHPMailer/PHPMailer>.

❖ 8.2.1. Get and Install the Latest Version of PHPMailer

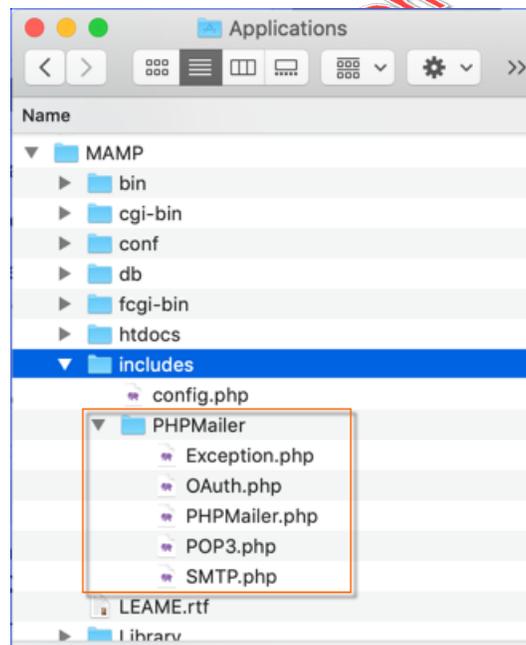
1. In your browser, navigate to <https://github.com/PHPMailer/PHPMailer>, click the **Clone or download** button and then click **Download ZIP**.



2. Download and unzip the file and open the resulting `PHPMailer-master` folder:

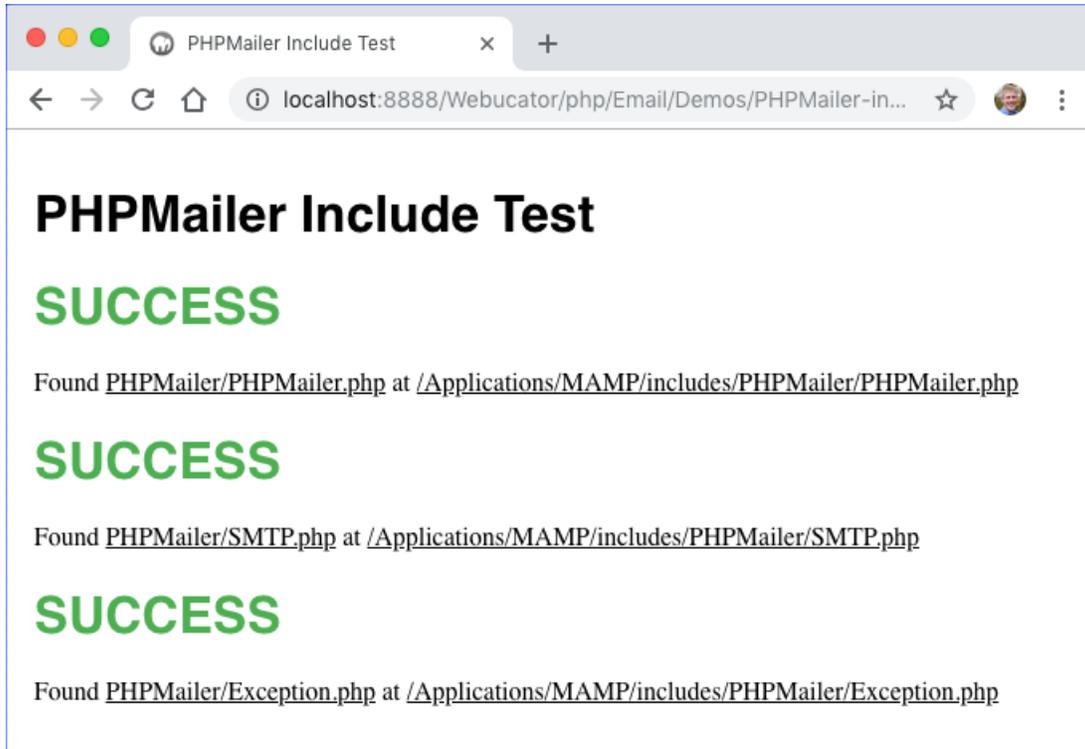


3. Copy the `src` directory and paste it in the `includes` folder that you created (or found) outside of the web root.¹⁷ Rename the `src` directory “PHPMailer”:



Run `http://localhost:8888/Webucator/php/Email/Demos/PHPMailer-include-test.php` in your browser to test that you are able to include PHPMailer. It should look like this:

17. Open `http://localhost:8888/Webucator/php/PhpBasics/Demos/find-document-root.php` to find the location of the web (or document) root. If you have taken this course straight through and haven't changed your setup, the `includes` folder should be next to the `htdocs` folder and have a file called `config.php` in it.



8.3. Mail Server

To send email, you need access to a mail server. While it is possible to set up a mail server on your own computer, in this course, we assume you are sending email using a hosted email service, such as **hotmail.com**.

Note

❖ 8.3.1. Be Careful with your Password!

Storing your personal password in a file is risky. If you choose to do it for this course, you should be sure to remove it once you are done. A safer alternative is to create another free email account with a different password for testing.

❖ 8.3.1. A Note on Hosted Email Services

While it is possible to send email with PHP and hosted email services other than hotmail.com, most of them will make you enable or disable specific settings and possibly jump through other hoops to get it working. We recommend you work with a hotmail.com account as that is currently the easiest option.

For production websites, you will either have your own mail server or use a hosted email service that is specifically designed for third-party apps, such as Amazon's Simple Email Service (<https://aws.amazon.com/ses/>).

We need to let PHPMailer know what mail server we are using, how to log in, and from whom to send emails. We will use a function for creating a new reusable, pre-configured PHPMailer object in the mail_config.php file in our includes directory outside of the web root.

Exercise 25: Including a Mail Configuration File

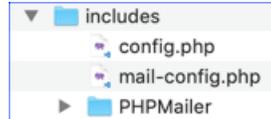
 10 to 15 minutes

In this exercise, you will add the configuration file shown below to your `includes` folder outside of your web root for sending emails.

Exercise Code 25.1: Email/Exercises/mail-config-sample.php

```
1.  <?php
2.      /*
3.          This config file is specific to the development machine.
4.          The config file on production will have the same
5.          functions, but they will return different values.
6.      */
7.      //Import PHPMailer classes into the global namespace
8.      use PHPMailer\PHPMailer\PHPMailer;
9.      use PHPMailer\PHPMailer\Exception;
10.
11.     require_once "PHPMailer/PHPMailer.php";
12.     require_once "PHPMailer/SMTP.php";
13.     require_once "PHPMailer/Exception.php";
14.
15.     function createMailer($debug=false) {
16.         $debugMode = $debug ? 2 : 0;
17.         $mail = new PHPMailer(true);
18.
19.         // Change these values to match your settings
20.         $mail->Host = "smtp.live.com"; // hotmail.com or outlook.com
21.         $mail->Port = 587;
22.         $mail->Username = 'you@hotmail.com'; // SMTP account username
23.         $mail->Password = 'yourP@ssw0rd'; // SMTP account password
24.         $mail->setFrom('you@hotmail.com', 'Your Name');
25.
26.         // Uncomment the next line and change the email address
27.         // if you don't want replies to go to the from address
28.         // $mail->addReplyTo('donotreply@example.com');
29.
30.         // Don't change values below this
31.         $mail->IsSMTP(); // use Simple Mail Transfer Protocol
32.         $mail->SMTPAuth = true; // enable SMTP authentication
33.         $mail->SMTPSecure = 'tls'; // use Transport Layer Security
34.         $mail->isHTML(true); // send as HTML
35.         $mail->SMTPDebug = $debugMode; // Debugging. 0 = no debug output
36.
37.         return $mail;
38.     }
39.     ?>
```

1. In Email/Exercises, you will find the mail-config-sample.php file shown above. Copy and paste that file into the includes folder outside of your web root and then rename it mail-config.php:



2. In your editor, open `mail-config.php` from the `includes` directory.
3. Note the first few lines:

```
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require_once "PHPMailer/PHPMailer.php";
require_once "PHPMailer/SMTP.php";
require_once "PHPMailer/Exception.php";
```

These are required to send email with PHPMailer. The two `use` statements bring PHPMailer's `PHPMailer` and `Exception` classes into the global namespace. Namespaces are beyond the scope of this course. It suffices to know that these lines need to be there so that we can use PHPMailer to send email and its related `Exception` class to catch PHPMailer-specific exceptions. The `require_once` statements bring in the PHPMailer files you have placed in the `includes` folder.

4. Below the `require_once` statements, you will see the `createMailer()` function, which creates and returns a configured PHPMailer object that you can use to send email, but first you will have to change the configuration.
5. Leave the `$debugMode` line as is. This is used to set the debug mode of PHPMailer. If you pass `true` to the `createMailer()` function, debug mode will be turned on via the `SMTPDebug` property.
6. Change the values of `$mail->Host` to the host of your email account:
 - A. For `hotmail.com` and `outlook.com` email addresses, the value should be `'smtp.live.com'`.
 - B. For `gmail.com`, the value should be `'smtp.gmail.com'`.
 - C. For `yahoo.com`, the value should be `'smtp.mail.yahoo.com'`.
 - D. If you don't have one of the above email addresses, you will need to either create one or look up the settings for the service you have.
7. You probably won't need to change the value of `$mail->Port` as most services use 587.

8. Change the values of the `$mail->Username` and `$mail->Password` to the username (email address) and password that you use to log in to your email account.
9. Change the values passed in to `$mail->setFrom` to the email address from which you want to send emails (most likely the same email you used for your username) and to your name, respectively.
10. By default, replies will go to the sender's email address. If you want them to go to a different address, uncomment the `$mail->addReplyTo()` line and replace the email address with the one you want replies to go to.
11. Save `mail-config.php`.
12. Run `http://localhost:8888/Webucator/php/Email/Exercises/send-email-test.php` in your browser to test that you are able to send email. Enter your email in the form field and click **Send**. It should output some debugging information followed by a success message:



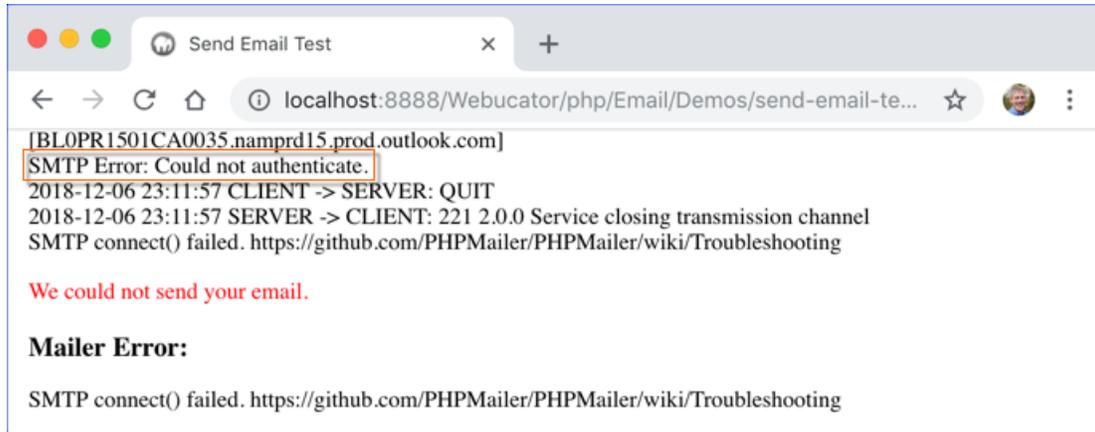
The screenshot shows a web browser window with the title "Send Email Test". The address bar displays "localhost:8888/Webucator/php/Email/Demos/send-email-te...". The main content area shows the following SMTP debugging output:

```
2018-12-06 23:01:51 CLIENT -> SERVER:
2018-12-06 23:01:51 CLIENT -> SERVER: .
2018-12-06 23:01:51 SERVER -> CLIENT: 250 2.0.0 OK
<CY1PR14MB0013B69541F766E38245E671D9A90@CY1PR14MB0013.namprd14.prod.outlook.com
[Hostname=CY1PR14MB0013.namprd14.prod.outlook.com]
2018-12-06 23:01:51 CLIENT -> SERVER: QUIT
2018-12-06 23:01:51 SERVER -> CLIENT: 221 2.0.0 Service closing transmission channel
```

Below the debugging output, the message "Test email sent." is displayed in green text.

Check your email. You should receive an email shortly.

- A. If the email fails to send, you should get debugging information followed by a failure message:



The reasons for failure will vary. In the screenshot above, the email failed because it couldn't authenticate. This probably means the username or password was wrong. If yours fails, read through the debugging information to see if you can figure out why it failed. If you have set up an email account, then it likely has something to do with your email configuration in `mail-config.php`.

- B. If you get a success message, but do not receive an email, check your spam. If you still don't find the email, log into your email account and look in your sent items to see if an email was sent. If it was not, then you'll have to do some more debugging to figure out what's going on. The reasons will vary and can be tricky to figure out.



8.4. Sending Email with PHPMailer

Now that you know that you are able to send email with PHPMailer, let's take a look at how `send-email-test.php` is written.

Demo 8.2: Email/Exercises/send-email-test.php

```
-----Lines 1 through 11 Omitted-----
12. <?php
13.     if (!isset($_POST['send'])) {
14.     ?>
15.         <form method="post" action="send-email-test.php">
16.             <label for="email">Email:</label>
17.             <input id="email" name="email">
18.             <button name="send">Send Test Email</button>
19.         </form>
20. <?php
21.     } else {
22.         require_once 'mail-config.php';
23.
24.         $now = date('r'); // Formatted date / time
25.
26.         try {
27.             $mail = createMailer(true);
28.             $mail->addAddress($_POST['email'], 'Webucator Student');
29.             $mail->Subject = 'Test Email';
30.             $mail->Body = "<p><strong>Email works!</strong> - $now</p>";
31.             $mail->AltBody = "Email works! - $now";
32.
33.             $mail->send();
34.             echo "<p class='success'>Test email sent.</p>";
35.         } catch (Exception $e) {
36.             echo "<p class='error'>We could not send your email.</p>";
37.             logError($e);
38.         }
39.     }
40. ?>
-----Lines 41 through 43 Omitted-----
```

Code Explanation

If the form is submitted, we attempt to send the email:

1. We include `mail-config.php` and set `$now` to hold a formatted date, so we can include that in the email message.
2. Within a try block, we...
 - A. Create a PHPMailer object, `$mail`, by calling `createMailer()`. We pass `true` to turn debugging on.

- B. We set the recipient's email and name using the `$mail->addAddress()` method.
 - C. We set the subject (`$mail->Subject`), HTML body (`$mail->Body`), and text body (`$mail->AltBody`). It's a good idea to send a text body in case the recipient's email client doesn't support HTML, though most do these days.
 - D. Finally, we try to send the email using `$mail->send()`. If this fails, an exception will occur, which will take us to the `catch` block.
3. In the `catch` block, we output the reason for the exception. We use `$mail->ErrorInfo` to output the error, which is useful during development. A common exception is a failure to connect to the SMTP server.



8.5. PHPMailer Methods and Properties

The following tables show some of the more common methods and properties of PHPMailer, many of which we have already seen. For complete documentation, see <http://phpmailer.github.io/PHPMailer/classes/PHPMailer.PHPMailer.PHPMailer.html>.

PHPMailer Methods

Method	Description
<code>addAddress()</code>	Adds a "To" address and name.
<code>addAttachment()</code>	Adds an attachment from a path on the file system.
<code>addBCC()</code>	Adds a "bcc" address.
<code>addCC()</code>	Adds a "cc" address.
<code>addReplyTo()</code>	Adds a "Reply-to" address. Without this, replies will go to the sender's email.
<code>isHTML()</code>	Sets message type to HTML.
<code>isSMTP()</code>	Sets Mailer to send message using Simple Mail Transfer Protocol (SMTP).
<code>send()</code>	Attempts to send message. If the message is not sent successfully then an exception occurs.
<code>setFrom()</code>	Adds a "From" address and name.

PHPMailer Properties

Property	Description
AltBody	Sets the text-only body of the message.
Body	Sets the Body of the message. This can be either an HTML or text body.
ErrorInfo	Holds the most recent mailer error message.
Host	Sets the SMTP hosts. All hosts must be separated by semicolons.
Password	Sets SMTP password.
Port	Sets the port. Usually 587.
SMTPAuth	Sets SMTP authentication. Utilizes the Username and Password properties.
SMTPDebug	SMTP class debug output mode.
SMTPSecure	Sets the encryption method. Usually set to 'tls'.
Subject	Sets the Subject of the message.
Username	Sets SMTP username.
WordWrap	Sets word wrapping on the body of the message to a given number of characters.

Exercise 26: Creating a Contact Form

 30 to 45 minutes

In this exercise, you will process the submission of the simple contact form created here:

Exercise Code 26.1: Email/Exercises/phppoetry.com/contact.php

```
1.  <?php
2.  ini_set('display_errors', '1');
3.  $pageTitle = 'Contact Us';
4.  require 'includes/header.php';
5.
6.  $f = [];
7.
8.  // Trim and Assign Form Entries
9.  $f['first-name'] = trim($_POST['first-name'] ?? '');
10. $f['last-name'] = trim($_POST['last-name'] ?? '');
11. $f['email'] = trim($_POST['email'] ?? '');
12. $f['message'] = trim($_POST['message'] ?? '');
13. $f['placeholder'] = 'Be it poetry. Be it prose.
14. This is where your message goes.';
15.
16. echo '<main id="contact-form">';
17. echo "<h1>$pageTitle</h1>";
18.
19. // You will do your work here.
20. ?>
21. <form method="post" action="contact.php" novalidate>
22.   <label for="first-name">First Name*:</label>
23.   <input name="first-name" id="first-name"
24.     value="<?= $f['first-name'] ?>" required>
25.   <label for="last-name">Last Name*:</label>
26.   <input name="last-name" id="last-name"
27.     value="<?= $f['last-name'] ?>" required>
28.   <label for="email">Email*:</label>
29.   <input type="email" name="email" id="email"
30.     value="<?= $f['email'] ?>" required>
31.   <label for="message">Message*:</label>
32.   <textarea placeholder="<?= $f['placeholder'] ?>" id="message"
33.     name="message"><?= $f['message'] ?></textarea>
34.   <button name="send" class="wide">Send</button>
35. </form>
36. </main>
37. <?php
38.   require 'includes/footer.php';
39. ?>
```

Code Explanation

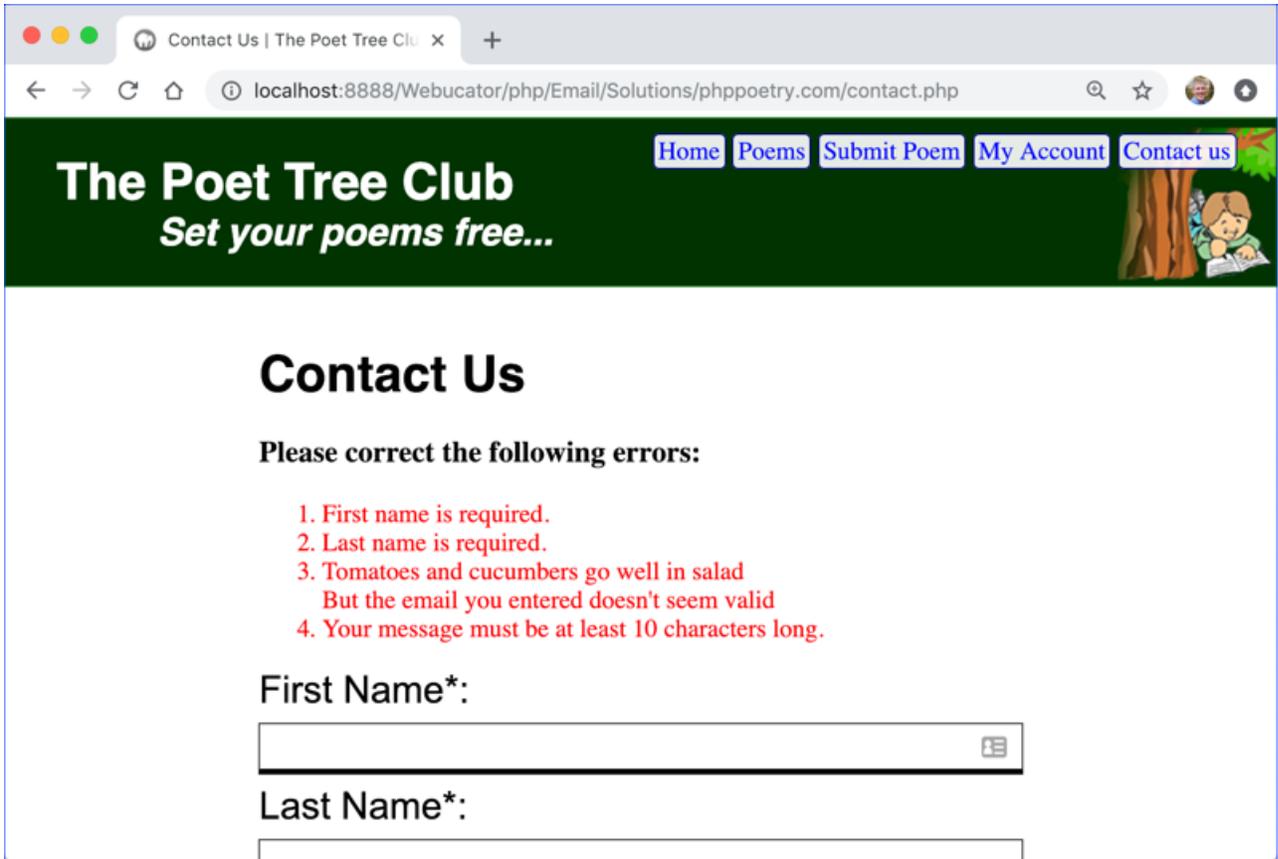
Navigate to `http://localhost:8888/Webucator/php/Email/Exercises/phppoetry.com/contact.php` to open the page in your browser. Submit the form and you will notice that the page simply refreshes. You need to write the processing code.

Things to notice:

1. The action of the form is “`contact.php`”. The page is submitting to itself. The processing code should only run if the form has been submitted.
2. We include the `novalidate` attribute in the `<form>` tag so that we can submit bad data to test our PHP validation. In a production environment, we would probably remove this attribute unless we were customizing the client-side validation with JavaScript.

Complete the exercise by following the instructions below:

1. Open `Email/Exercises/phppoetry.com/includes/constants.php` in your editor. This file contains many constants that we will use throughout the rest of the course. We will use the first two in this exercise: `POEM_MAIL_FAIL` and `POEM_MAIL_SUCCESS`.
2. Open `Email/Exercises/phppoetry.com/includes/header.php` in your editor. Notice that we now require `constants.php`.
3. Open `Email/Exercises/phppoetry.com/contact.php`. This is where you will process the form.
4. The form processing code should only run if the form has been submitted. Write an if condition checking to see if the form has been submitted by checking for the presence of ‘`send`’ in the `$_POST` array. There are a variety ways to do this. The rest of the code you write will go within this if block.
5. Include `mail-config.php`.
6. We will track any errors in the form submission in an `$errors` array. To start that, set an empty `$errors` array.
7. Check to make sure `$_POST['first-name']` and `$_POST['last-name']` have values, that `$_POST['email']` is a valid email address, and that `$_POST['message']` is at least 10 characters long. For the email error, you may want to use the `POEM_INVALID_EMAIL` constant.
8. If there are errors, report them in an ordered list with the class of “`error`”:

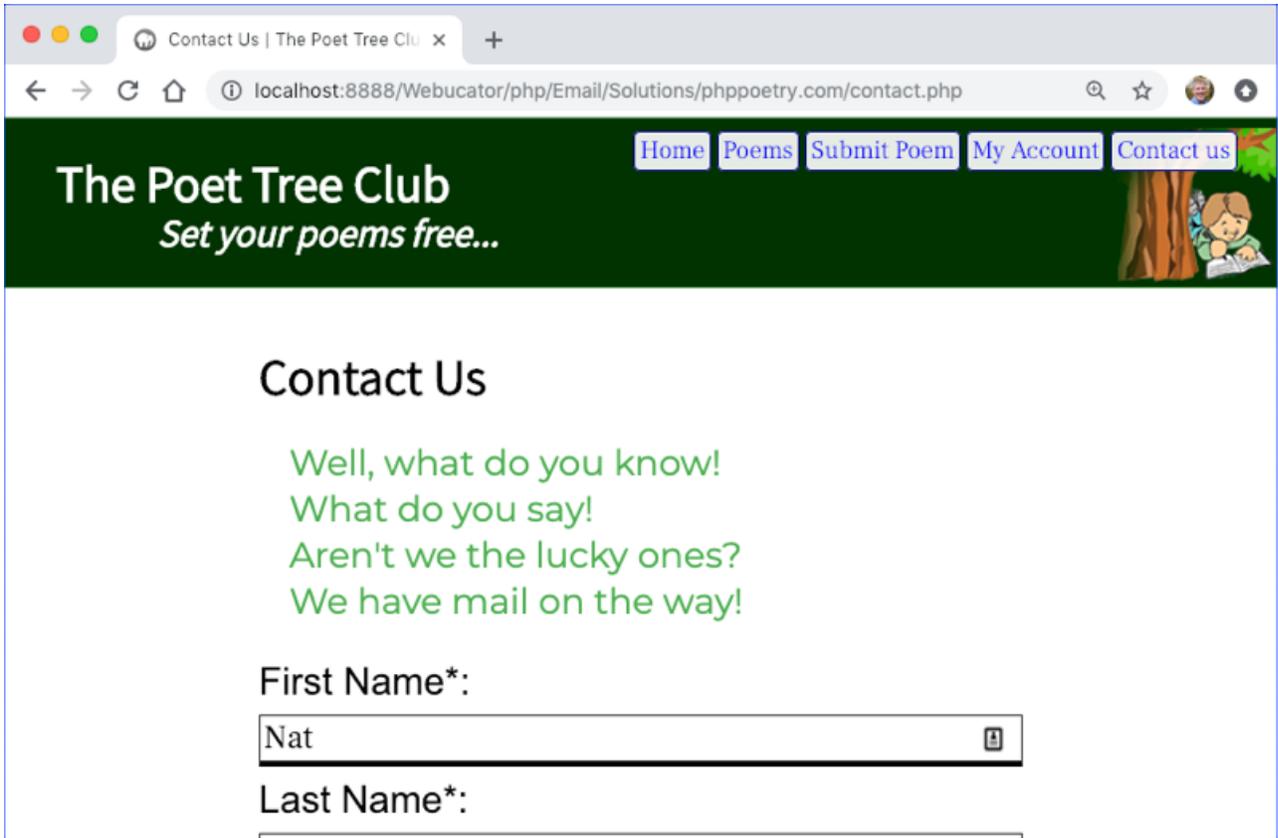


9. If there are no errors, attempt to send an email to the person who submitted the form. Include the data from the form in the body of the email. For example, your email body might read:

```
Oh, happy day, we are touched
You took the time to contact us
We will take note
Of what you wrote
Thank you very, very much!
```

```
Your Name: Nat Dunn
Your Email: nat@example.com
Your Message
I am writing to say hi.
```

10. If the email sends, output the `POEM_MAIL_SUCCESS` constant to the browser in a `<p>` tag with the class of "success":



11. If there is an exception, log the error in `utilities.php` using the `logError()` function:

Contact Us | The Poet Tree Club

localhost:8888/Webucator/php/Email/Solutions/phppoetry.com/contact.php

The Poet Tree Club

Set your poems free...

Contact Us

2019-01-30 23:06:12 SMTP ERROR: Failed to connect to server:
php_network_getaddresses: getaddrinfo failed: nodename nor servname
provided, or not known (0)
SMTP connect() failed.
<https://github.com/PHPMailer/PHPMailer/wiki/Troubleshooting>

Oh how we did try
And oh how we did fail
I am sorry to say
We could not send your mail

For Developers' Eyes Only

Mailer Exception:

SMTP connect() failed.
<https://github.com/PHPMailer/PHPMailer/wiki/Troubleshooting>

First Name*:

12. Test your solution in your browser.

Solution: Email/Solutions/phppoetry.com/contact.php

```
-----Lines 1 through 18 Omitted-----
19. if (isset($_POST['send'])) {
20.     require_once 'mail-config.php';
21.
22.     $errors = [];
23.
24.     if (!$f['first-name']) {
25.         $errors[] = 'First name is required.';
26.     }
27.
28.     if (!$f['last-name']) {
29.         $errors[] = 'Last name is required.';
30.     }
31.
32.     if (!$f['email']) {
33.         $errors[] = 'Email is required.';
34.     } elseif (!filter_var($f['email'], FILTER_VALIDATE_EMAIL)) {
35.         $errors[] = nl2br(POEM_INVALID_EMAIL);
36.     }
37.
38.     if (strlen($f['message']) < 10) {
39.         $errors[] = 'Your message must be at least 10 characters long.';
40.     }
41.
42.     if ($errors) {
43.         echo '<h3>Please correct the following errors:</h3>';
44.         <ol class="error">;
45.         foreach ($errors as $error) {
46.             echo "<li>$error</li>";
47.         }
48.         echo '</ol>';
49.     } else {
50.         $to = $f['email'];
51.         $toName = $f['first-name'] . ' ' . $f['last-name'];
52.         $subject = 'Contact Form Submission';
53.
54.         $html = "<p>Oh, happy day, we are touched<br>
55.         You took the time to contact us<br>
56.         We <em>will</em> take note<br>
57.         Of what you wrote<br>
58.         Thank you <em>very</em>, very much!</p>
59.         <p>Name: $toName</p>
60.         <p>Email: " . $f['email'] . "</p>
61.         <h3>Your Message</h3>" . nl2br($f['message']);
```

```
62.
63.     $text = "Oh, happy day, we are touched
64. You took the time to contact us
65. We will take note
66. Of what you wrote
67. Thank you very, very much!
68.
69. * Name: $toName
70. * Email: " . $f['email'] . "
71. * Your Message: " . $f['message'];
72.
73.     echo '<article class="poem">';
74.     try {
75.         // Pass true to createMailer() to enable debugMode
76.         $mail = createMailer();
77.         $mail->addAddress($to, $toName);
78.         // $mail->addBcc('you@example.com');
79.         $mail->Subject = $subject;
80.         $mail->Body = $html;
81.         $mail->AltBody = $text;
82.
83.         $mail->send();
84.         echo '<p class="success">' . nl2br(POEM_MAIL_SUCCESS) . '</p>';
85.     } catch (Exception $e) {
86.         echo '<p class="error">' . nl2br(POEM_MAIL_FAIL) . '</p>';
87.         logError($e);
88.     }
89.     echo '</article>';
90. }
91. }
92. ?>
-----Lines 93 through 111 Omitted-----
```

Conclusion

In this lesson, you have learned to send email messages with PHP's built-in `mail()` function and with PHPMailer.

LESSON 9

Authentication with PHP and SQL

Topics Covered

- Passwords and pass phrases.
- Tokens.
- Session variables.
- Cookies.
- Creating a login form.
- Creating a logout page.
- Creating a registration form.
- Creating a pass phrase reset form.

*Evaluation
Copy*

Introduction

In this lesson, you will learn to safely store passwords and pass phrases, to create and work with tokens, to manage session variables and cookies, to create a login form, a logout page, a registration form, and a pass phrase reset form.



9.1. The Registration Process

A good registration process works like this:

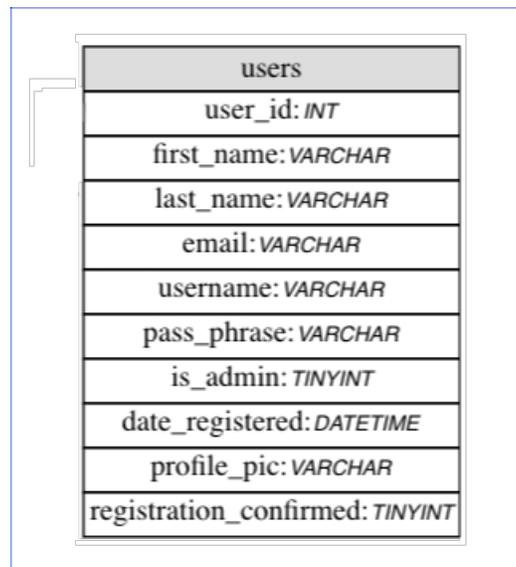
1. User registers choosing a username and password, which they must enter twice to make sure they didn't mistype it.
2. If registration is valid, the user's data gets stored in the database, but is not marked as confirmed.
3. An email is sent to the user asking them to confirm their registration. This is to prevent people from registering others without their knowledge.

4. The user clicks on a link in the email confirming their registration. This marks the user confirmed in the database and brings the user to a page with a login form or a link to a login page.
5. The user can then log in with their username and password.
6. If the user doesn't remember their password, they can click a link to reset it. This will generate an email with a password reset link, which leads to a form for resetting the password.
7. The user can also update their data, including their password, on their "my account" page.



9.2. Passwords and Pass Phrases

The **users** table in our poetree database looks like this:

A diagram showing the structure of the 'users' table. The table is represented as a vertical stack of rows, each containing a column name and its data type. The table is enclosed in a blue border with a small handle on the left side.

users
user_id: <i>INT</i>
first_name: <i>VARCHAR</i>
last_name: <i>VARCHAR</i>
email: <i>VARCHAR</i>
username: <i>VARCHAR</i>
pass_phrase: <i>VARCHAR</i>
is_admin: <i>TINYINT</i>
date_registered: <i>DATETIME</i>
profile_pic: <i>VARCHAR</i>
registration_confirmed: <i>TINYINT</i>

Notice that, instead of a traditional password, we are using a pass phrase (`pass_phrase`). Long pass phrases can be easier for users to remember and more difficult for hackers to discover than shorter complex passwords.

Look at the data in the database table and you'll see that the `pass_phrase` field contains values like:

```
$2y$10$GTRg30MHAX5KFXdcVwDaS.oCHHrh1U6BqFF0h5JTpyycDhojjguqW
```

While that may be difficult for a hacker to figure out, it would also be difficult for a user to remember! Luckily, that isn't the user's actual pass phrase. Rather, it is created using PHP's built-in `password_hash()` function, which generally takes two arguments:

1. The password (or phrase) to be hashed.
2. An integer representing the hashing algorithm. PHP provides several constants holding valid integers. You will most likely use the `PASSWORD_DEFAULT` algorithm, but you can check out other options at https://www.php.net/password_hash.

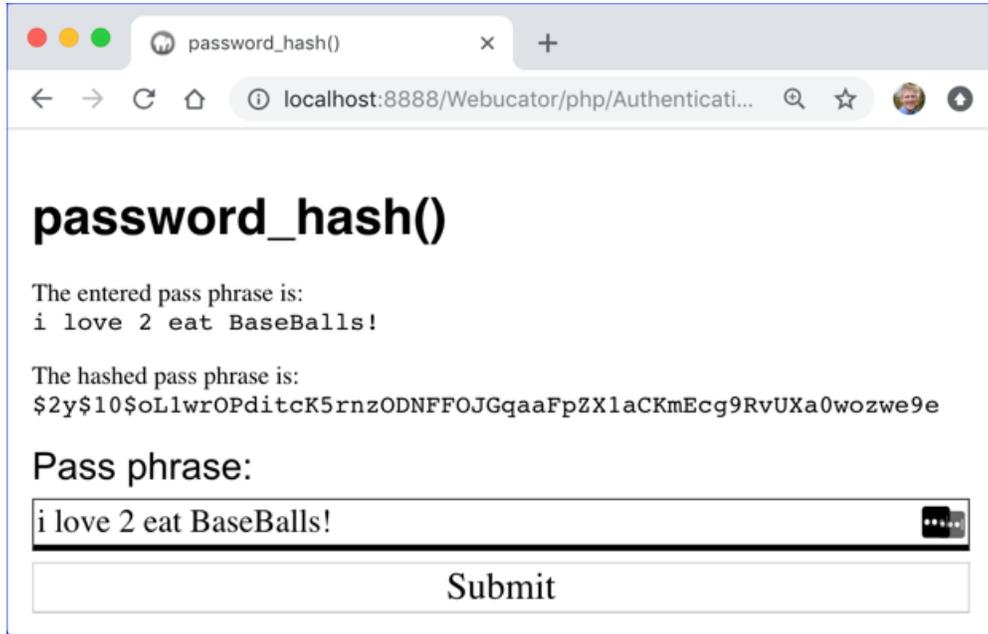
The following demo shows how the `password_hash()` function works.

Demo 9.1: Authentication/Demos/password-hash.php

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.  <meta charset="UTF-8">
5.  <meta name="viewport" content="width=device-width,initial-scale=1">
6.  <link rel="stylesheet" href="../../static/styles/normalize.css">
7.  <link rel="stylesheet" href="../../static/styles/styles.css">
8.  <title>password_hash()</title>
9.  </head>
10. <body>
11. <main>
12.   <h1>password_hash()</h1>
13.   <?php
14.     $passPhrase = $_POST['pass-phrase'] ?? '';
15.     if ($passPhrase) {
16.       $hashedPhrase = password_hash($passPhrase, PASSWORD_DEFAULT);
17.
18.       echo "<p>The entered pass phrase is:<br>
19.           <code>$passPhrase</code></p>";
20.       echo "<p>The hashed pass phrase is:<br>
21.           <code>$hashedPhrase</code></p>";
22.     }
23.   ?>
24.   <form method="post">
25.     <label for="pass-phrase">Pass phrase:</label>
26.     <input name="pass-phrase" id="pass-phrase"
27.       value="<?= $passPhrase ?>">
28.     <button class="wide">Submit</button>
29.   </form>
30. </main>
31. </body>
32. </html>
```

Code Explanation

Visit <http://localhost:8888/Webucator/php/Authentication/Demos/password-hash.php> and try submitting some phrases. You will see the original phrase and the hashed phrase returned by the `password_hash()` function:



Passwords and Security

For security reasons, you should never return the password to the browser, send it by email, or even store it in a database or a file in its raw form.

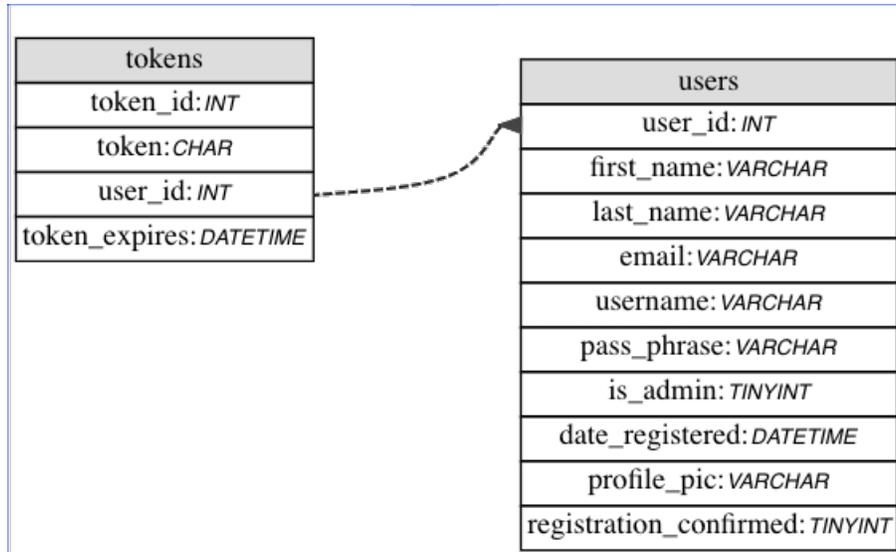


9.3. Registration with Tokens

In the registration process we laid out above, we included:

1. If registration is valid, the user's data gets stored in the database, *but is not marked as confirmed*.
2. An email is sent to the user asking them to confirm their registration.
3. The user clicks on a link in the email confirming their registration. This *marks the user confirmed* in the database.

Here we will look more deeply at the confirmation process, which is done with tokens, which are stored in the database. The diagram below show the relationship between users and tokens:



Things to note:

1. A token is associated with a user through the `user_id` field. The `token_expires` field will hold the date and time that the token expires.
2. The token field has a *unique* constraint, so we can be sure that searching for a specific token will only return 0 or 1 records.
3. The **users** table includes a `registration_confirmed` field, which will default to 0, indicating that the user has not confirmed their registration.

The confirmation process works as follows:

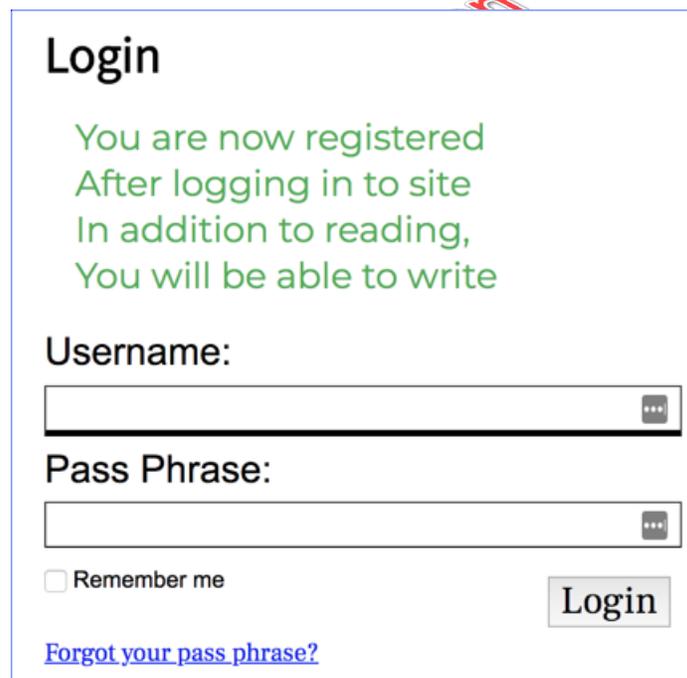
1. When the user registers, the user data is entered into the **users** table and a random token is generated and stored in the **tokens** table with the `user_id` set to the new user's `user_id` and the `token_expires` field set to a date and time by which the user must confirm the registration.
2. An email is sent to the user with a link to a URL like the one below:

`registration-confirm.php?token=823814931337503d622eb2967a7311f3db177bd6afa2db73508535e8fdec1d10`

3. When the user clicks on that link, the page will run the following query:

```
UPDATE users
SET registration_confirmed = 1
WHERE user_id = (SELECT user_id
                 FROM tokens
                 WHERE token = ?
                 AND token_expires > now() );
```

- A. The ? will be replaced with the token passed in on the URL.
- B. The subquery gets the `user_id` from the **tokens** table for the record with that token that has not yet expired, if such a record exists.
4. We will redirect the user to `login.php?just-registered=1` if the query successfully updates a record, and use the `just-registered` flag to output a success message:



The screenshot shows a login form titled "Login". At the top, there is a green success message: "You are now registered After logging in to site In addition to reading, You will be able to write". Below this, there are two input fields: "Username:" and "Pass Phrase:". Each field has a small "..." icon on the right side. Below the "Pass Phrase:" field, there is a checkbox labeled "Remember me". To the right of the checkbox is a "Login" button. At the bottom left, there is a blue link: "Forgot your pass phrase?".

Exercise 27: Creating a Registration Form

 45 to 60 minutes

In this exercise, you will process a registration form. First, let's take a look at the files involved:

1. `utilities.php` - This contains our utility functions. We have added a few:
 - A. `isAuthenticated()` - We will look at this soon.
 - B. `generateToken()` - This generates a random string of a specified even length defaulting to 64.
 - C. `getFullPath()` - This is a utility function that creates a full absolute URL from a relative path.
 - D. `logout()` - We will look at this soon.
2. `registration-confirm.php` - This is the page the user visits to confirm their registration. It has been done for you. Review it thoroughly.
3. `register.php` - This is the page you will be working on.

The three pages are shown below:

Exercise Code 27.1:

Authentication/Exercises/phppoetry.com/includes/utilities.php

```
-----Lines 1 through 19 Omitted-----
20.     function isAuthenticated() {
21.         return isset( $_SESSION['user-id'] );
22.     }

-----Lines 23 through 33 Omitted-----
34.     function generateToken($length = 64) {
35.         // generate random token
36.         if ($length % 2 !== 0) {
37.             throw new Exception('$length must be even. ');
38.             return false;
39.         }
40.         return bin2hex(random_bytes($length/2));
41.     }
42.
43.     function getFullPath($relativePath) {
44.         $protocol = ( !empty($_SERVER['HTTPS']) &&
45.             $_SERVER['HTTPS'] !== 'off' || $_SERVER['SERVER_PORT'] == 443
46.             ) ? "https://" : "http://";
47.         $domainName = $_SERVER['HTTP_HOST'];
48.         $relPathSplit = explode('/', $relativePath);
49.         $pathFromHost = dirname($_SERVER['REQUEST_URI']);
50.         $pathFromHostSplit = explode('/', $pathFromHost);
51.         while ($relPathSplit[0] === '..') {
52.             array_shift($relPathSplit);
53.             array_pop($pathFromHostSplit);
54.         }
55.         return $protocol.$domainName . implode('/', $pathFromHostSplit) .
56.             '/' . implode('/', $relPathSplit);
57.     }
58.
-----Lines 59 through 81 Omitted-----
82.     function logout() {
83.         unset($_SESSION['user-id']);
84.         unset($_COOKIE['token']); // unset on server
85.         setcookie('token', '', 0); // unset on client
86.     }
87.     ?>
```

Next is the page that the user is directed to to confirm their registration. The code is already complete. Review it.

Exercise Code 27.2:

Authentication/Exercises/phppoetry.com/registration-confirm.php

```
1.  <?php
2.    if (!isset($_GET['token'])) {
3.        // How did you get here?
4.        header("Location: index.php");
5.    }
6.
7.    $pageTitle = 'Registration Confirmation';
8.    require 'includes/header.php';
9.    logout(); // In case a different user has logged in
10.
11.    $token = $_GET['token'];
12.    ?>
13.
14.  <?php
15.    $sqlUpdate = "UPDATE users
16.    SET registration_confirmed = 1
17.    WHERE user_id = (SELECT user_id
18.    FROM tokens
19.    WHERE token = ?
20.    AND token_expires > now() );";
21.
22.    try {
23.        $stmtUpdate = $db->prepare($sqlUpdate);
24.
25.        if (!$stmtUpdate->execute( [$token] )) {
26.            // Query failed to execute
27.            logError($stmtUpdate->errorInfo()[2], true);
28.        } elseif ($stmtUpdate->rowCount()) {
29.            // Redirect user to login page
30.            header("Location: login.php?just-registered=1");
31.        } // Else no rows were updated. Continue to error message.
32.    } catch (PDOException $e) {
33.        logError($e);
34.    }
35.    ?>
36.  <!-- Won't get her unless something went wrong -->
37.  <main class="narrow">
38.    <h1><?= $pageTitle ?></h1>
39.    <article class='poem error'>
40.        <?= nl2br(POEM_INVALID_TOKEN_REGISTRATION) ?>
41.    </article>
42.  </main>
43.  <?php
```

```
44.     require 'includes/footer.php' ;
45.     ?>
```

Code Explanation

Things to notice:

1. The first thing it does is check for the existence of `$_GET['token']`. The only proper way to get to this page is through the confirmation email we sent, so if the token doesn't exist, it means the user got here improperly, so we redirect to the home page.
2. After setting `$pageTitle` and including `header.php`, we call `logout()` to force a logout. This is a precaution. It's possible that between the time a visitor registers on the site and confirms the registration, another user has logged in. We will look at the `logout()` function soon.
3. We then assign `$_GET['token']` to `$token`.
4. We then run the update:
 - A. If `$stmtUpdate->execute([$token])` returns false, the query didn't execute properly (maybe a syntax error?). In this case, we log the error and redirect to the error page.
 - B. If `$stmtUpdate->rowCount()` returns 1 (true), we redirect to the login page, passing the `just-registered=1` flag.
 - C. Otherwise, no rows were updated, meaning the token either didn't exist or was expired. In this case, we remain on the page and output an error letting them know that something went wrong:

Registration Confirmation

Oh no, something went wrong
I'm sorry, my friend
You may have waited too long
Please, [do try again](#)

Finally, we have the file in which you will be doing your work: the registration page:

Exercise Code 27.3:

Authentication/Exercises/phppoetry.com/register.php

```
-----Lines 1 through 47 Omitted-----
48.     // Check if username exists
49.     $qUsernameCheck = "SELECT user_id
50.         FROM users
51.         WHERE username = ?";
52.
53.     try {
54.         $stmtUsername = $db->prepare($qUsernameCheck);
55.         $stmtUsername->execute([ $f['username'] ]);
56.
57.         if ($stmtUsername->fetch()) {
58.             $errors[] = 'That username is already taken.<br>
59.                 Please try a different one.';
60.         }
61.     } catch (PDOException $e) {
62.         logError($e);
63.         $errors[] = 'Oops! Our bad. We cannot register you right now.';
64.     }
65.
66.     // TODO: Check if email exists
67.
68.     if (!$errors) {
69.         // TODO: Prepare to insert user by creating the $hashedPhrase,
70.         //         $token, and $qInserts variables
71.
72.         try {
73.             // TODO: Insert the user
74.         } catch (PDOException $e) {
75.             logError($e);
76.             $errors[] = 'Registration failed. Please try again.';
77.         }
-----Lines 78 through 166 Omitted-----
```

Code Explanation

This is where you will do your work.

Follow the instructions below to complete the exercise:

1. Open `Authentication/Exercises/phppoetry.com/register.php` in your editor.
2. Much of this code is already done for you. Study the code and make sure you understand everything that is already done.
3. When the user submits the form, we check to make sure the data entered is valid and add error messages to the `$errors` array as appropriate.
4. We then check to see if the username is already taken. If it is, we add an error message to the `$errors` array.
5. Below the username check, write code checking to see if a user with that email already exists. If one does, add the following error to the `$errors` array:

```
We recognize that email.<br>
Did you <a href="pass-phrase-reset.php">forget your pass phrase</a>?
```

6. If, after validating the data, there are no errors, insert the user and a user token into the database:
 - A. Set `$hashedPhrase` to the hashed user's pass phrase.
 - B. Create a 64-character token using the `generateToken()` method in `utilities.php`. Name the variable `$token`.
 - C. Set `$qInserts` to:

```
INSERT INTO users
(first_name, last_name, email, username, pass_phrase)
VALUES (:first_name, :last_name, :email,
        :username, '$hashedPhrase');
```

```
INSERT INTO tokens
(token, user_id, token_expires)
VALUES (:token, LAST_INSERT_ID(),
        DATE_ADD(now(), INTERVAL 1 HOUR));
```

7. Within the `try` block:
 - A. Prepare a `PDOStatement` using `$qInserts` and assign the result to `$stmtInserts`.

- B. Use the `bindParam()` method of `$stmtInserts` to bind variables to the named parameters. The first one will look like this:

```
$stmtInserts->bindParam(':first_name', $f['first-name']);
```

- C. Attempt to execute `$stmtInserts`. If it fails, log the error and add the following error to `$errors`:

```
Registration failed. Please try again.
```

8. The rest of the code is written.
9. If there are still no errors:
- We build a query string using PHP's built-in `http_build_query()` function. This function will encode the query string to make it safe for URLs.
 - With that query string and the `getFullPath()` function from `utilities.php`, we create the URL we will send to the user to confirm the registration.
 - We attempt to send the user the email.
10. If the email was sent, we output a success message: "We have sent you an email with instructions. Check your email."
11. If it was not, we show any errors followed by the form.

Note that the first time this page is visited, `$registrationMailSent` and `$errors` will both be empty, because those variables are created within the `if` block that checks if the form has been submitted. Because they are both empty, only the registration form will show up on the page.

To test your solution:

- Visit <http://localhost:8888/Webucator/php/Authentication/Exercises/phppeetry.com/register.php>.
- Register for a new account. Remember your pass phrase!
- You should get an email asking you to confirm your registration. Click the link and confirm the registration.
- Look at the **users** table in PHPMyAdmin. Are you in there?
- If you get any errors, go back and fix your code.

6. If the registration succeeds, try registering again, but intentionally enter invalid data. Do you get helpful errors?

Valuable
Copy

Solution: Authentication/Solutions/phppoetry.com/register.php

```
-----Lines 1 through 65 Omitted-----
66.     // Check if email exists
67.     $qEmailCheck = "SELECT user_id
68.         FROM users
69.         WHERE email = ?";
70.
71.     try {
72.         $stmtEmail = $db->prepare($qEmailCheck);
73.         $stmtEmail->execute([ $f['email'] ]);
74.
75.         if ($stmtEmail->fetch()) {
76.             $errors[] = 'We recognize that email.<br>
77.                 Did you <a href="pass-phrase-reset.php">forget your
78.                 pass phrase</a>?';
79.         }
80.     } catch (PDOException $e) {
81.         logError($e);
82.         $errors[] = 'Oops! Our bad. We cannot register you right now.';
83.     }
84.
85.     if (!$errors) {
86.         // Insert user
87.         $hashedPhrase = password_hash($passPhrase1, PASSWORD_DEFAULT);
88.         $token = generateToken();
89.         $qInserts = "INSERT INTO users
90.             (first_name, last_name, email, username, pass_phrase)
91.             VALUES (:first_name, :last_name, :email,
92.                 :username, '$hashedPhrase');
93.
94.         INSERT INTO tokens
95.             (token, user_id, token_expires)
96.             VALUES (:token, LAST_INSERT_ID(),
97.                 DATE_ADD(now(), INTERVAL 1 HOUR));";
98.
99.         try {
100.            $stmtInserts = $db->prepare($qInserts);
101.            $stmtInserts->bindParam(':first_name', $f['first-name']);
102.            $stmtInserts->bindParam(':last_name', $f['last-name']);
103.            $stmtInserts->bindParam(':email', $f['email']);
104.            $stmtInserts->bindParam(':username', $f['username']);
105.            $stmtInserts->bindParam(':token', $token);
106.            if (!$stmtInserts->execute()) {
107.                logError($stmtInserts->errorInfo()[2]);
108.                $errors[] = 'Registration failed. Please try again.';

```

```
109.     }
110.     } catch (PDOException $e) {
111.         logError($e);
112.         $errors[] = 'Registration failed. Please try again.';
113.     }
```

-----Lines 114 through 202 Omitted-----



9.4. Sessions

When a user logs into a website, you need to keep track of them throughout their visit. This is done using sessions.

A session begins when a visiting client identifies itself to the web server. The web server assigns the client a unique session id, which the client uses to re-identify itself as it moves from page to page on the website. Most of the time, these unique ids are stored in session cookies that expire after the client hasn't interacted with the server for some amount of time. The amount of time varies depending on the web application. For example, an online investment site might have very short sessions, so that if a user leaves their computer without logging out, another user who sits down at the same computer several minutes later cannot continue with the first user's session.

The default session length in PHP is 1440 seconds (24 minutes), but you can change this using the `session.gc_maxlifetime` variable in the `php.ini` file.

❖ 9.4.1. Session Variables

Before setting, reading, or unsetting session variables, you need to start the session. This is done using the `session_start()` function. If a session has already been started on a previously visited page, `session_start()` will continue with that session.

The page below illustrates how session variables are used. Notice the following:

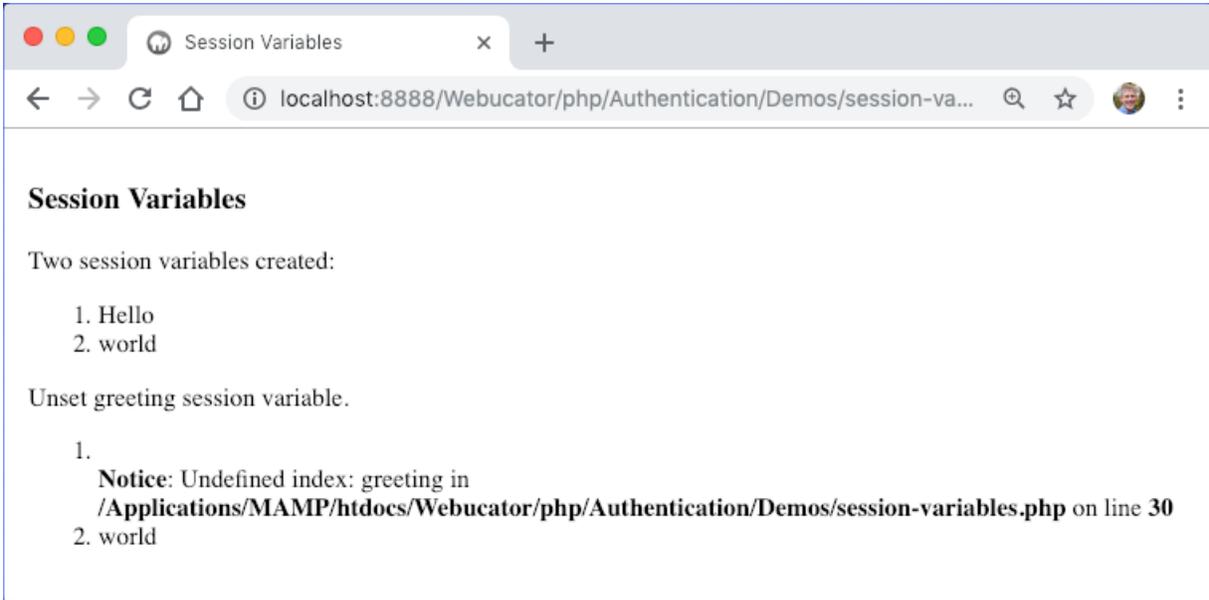
1. Pages that are part of the session should begin with a call to `session_start()`.
2. Session variables are created in the `$_SESSION` superglobal array.
3. Session variables are deleted in the same way as other variables - using the `unset()` function.

Demo 9.2: Authentication/Demos/session-variables.php

```
1.  <?php
2.    ini_set('display_errors', '1');
3.
4.    session_start();
5.    $_SESSION['greeting'] = 'Hello';
6.    $_SESSION['name'] = 'world';
7.  ?>
8.  <!DOCTYPE html>
9.  <html lang="en">
10. <head>
11. <meta charset="UTF-8">
12. <meta name="viewport" content="width=device-width,initial-scale=1">
13. <link rel="stylesheet" href="../../static/styles/normalize.css">
14. <link rel="stylesheet" href="../../static/styles/styles.css">
15. <title>Session Variables</title>
16. </head>
17. <body>
18. <main>
19.   <h3>Session Variables</h3>
20.   <p>Two session variables created:</p>
21.   <ol>
22.     <li><?= $_SESSION['greeting'] ?></li>
23.     <li><?= $_SESSION['name'] ?></li>
24.   </ol>
25.   <p>Unset greeting session variable.</p>
26.   <?php
27.     unset($_SESSION['greeting']);
28.   ?>
29.   <ol>
30.     <li><?= $_SESSION['greeting'] ?></li>
31.     <li><?= $_SESSION['name'] ?></li>
32.   </ol>
33. </main>
34. </body>
35. </html>
```

Code Explanation

In this file, we start the session, create and display two session variables, and then unset one of those variables:



session_unset() and session_destroy()

There are two session functions that may be tempting to use, but should be avoided: `session_unset()` and `session_destroy()`. It is better and safer to clean up unneeded or unwanted session variables using `unset()`. For example:

```
unset($_SESSION['foo']);
```



9.5. Cookies

When a user leaves your website and then returns at a later time or date, you may want to remember them from their previous visit. This is done using cookies.

Cookies are stored by the browser on the client machine. Web pages with the right permissions can set, read, and delete cookies. Cookies are generally used to track user information between visits.

In PHP, cookies are set with the `setcookie()` function, which can take several parameters including:

1. The cookie's name (required).
2. The cookie's value.
3. The cookie's expiration date (if this isn't set, the cookie will expire when the browser window is closed).
4. The directory path on the server that can read the cookie.
5. The domain name that can read the cookie.
6. A flag indicating whether the cookie should only be read over HTTPS.

The following code will set a cookie that expires in one week.

```
setcookie('flavor','chocolate chip', time()+60*60*24*7);
```

There is no `deletecookie()` function. To delete a cookie on the client, set the value to an empty string and set the expiration date to sometime in the past, like this (0 is the epoch):

```
setcookie('flavor','', 0);
```

Note that setting the cookie's expiration date to a date in the past only deletes the cookie on the client. It will not delete it on the server.

Cookies are set in the HTTP header, so they must be set before any HTML code is passed back to the browser.

Together, the files below illustrate how cookies work:

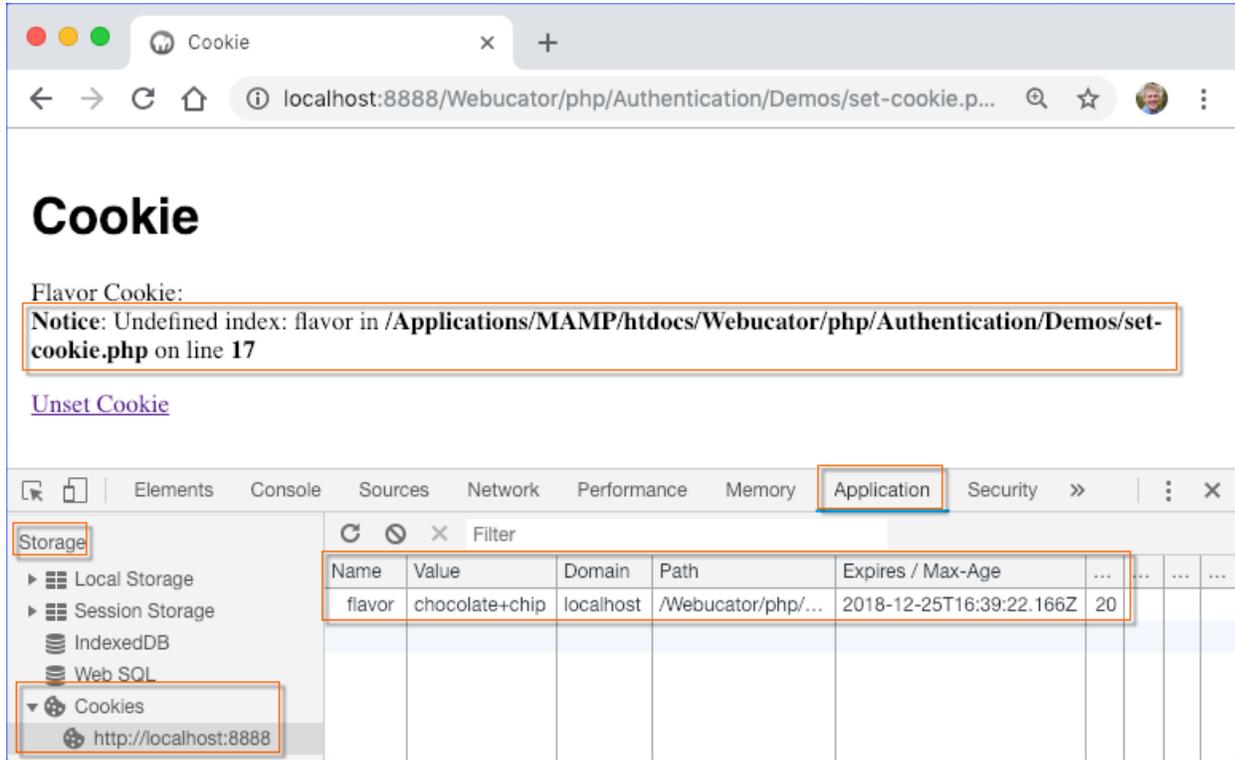
Demo 9.3: Authentication/Demos/set-cookie.php

```
1.  <?php
2.    ini_set('display_errors', '1');
3.
4.    setcookie('flavor','chocolate chip', time()+60*60*24*7);
5.  ?>
6.  <!DOCTYPE html>
7.  <html lang="en">
8.  <head>
9.  <meta charset="UTF-8">
10. <meta name="viewport" content="width=device-width,initial-scale=1">
11. <link rel="stylesheet" href="../../static/styles/normalize.css">
12. <link rel="stylesheet" href="../../static/styles/styles.css">
13. <title>Cookie</title>
14. </head>
15. <body>
16. <main>
17.   <h1>Cookie</h1>
18.   <p>Flavor Cookie: <?= $_COOKIE['flavor'] ?></p>
19.   <a href="unset-cookie.php">Unset Cookie</a>
20. </main>
21. </body>
22. </html>
```

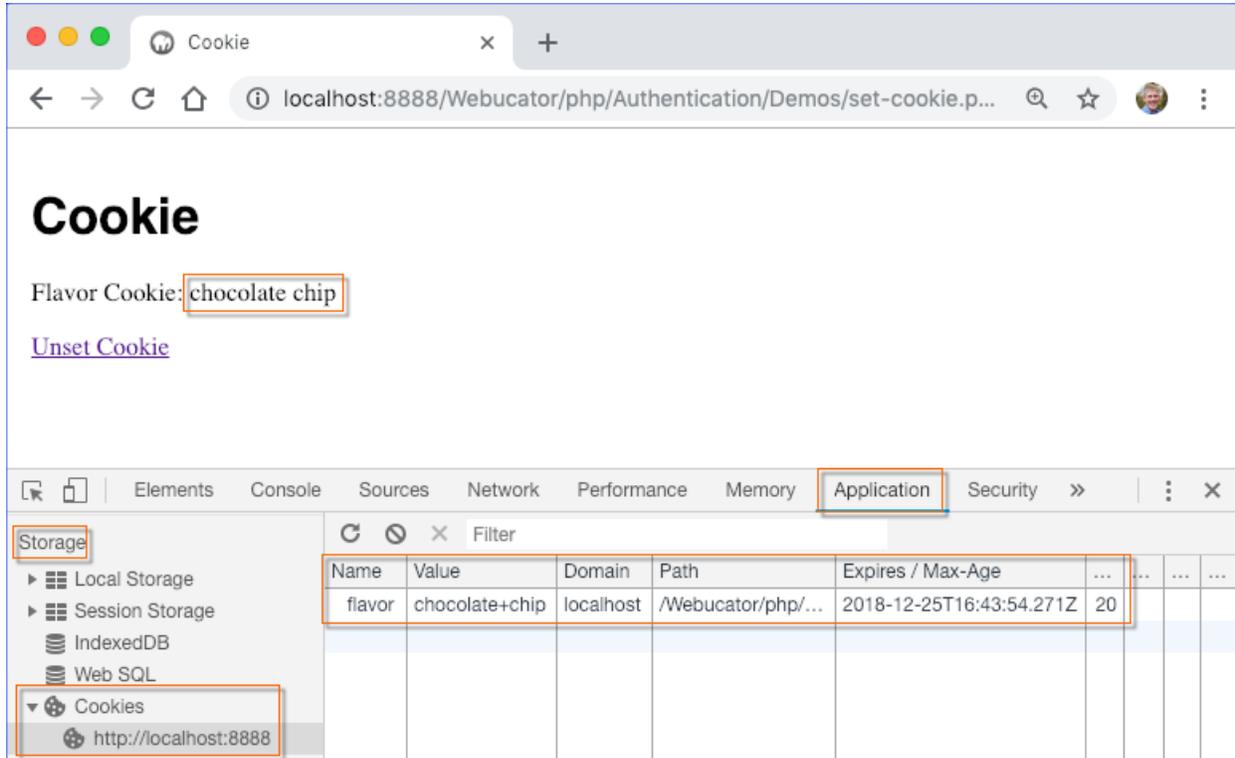
Code Explanation

Visit <http://localhost:8888/Webucator/php/Authentication/Demos/set-cookie.php> to open this page in your browser.

This sets the cookie, but note that it won't be available on the page in which it is initially set. That's because the `$_COOKIE` array is populated with cookies sent from the client with the page request. The `setcookie()` method does not itself add an item to the `$_COOKIE` array. So, you won't be able to read a cookie that you just set. The screenshot below shows that the cookie is set in Google Chrome, but the page did not have access to it at the time the page was created:



Refresh the page and the page can now read `$_COOKIE['flavor']`. That's because the browser sent the cookie with the page request:



As shown in the screenshots above, you can see the cookies for a page in Google Chrome DevTools. To do so:

1. Open Chrome DevTools.
2. Click the **Application** tab.
3. On the left side under **Storage**, expand **Cookies** and click the URL.
4. On the right, you'll see the **flavor** cookie. Notice the expiration date. The screenshot is from December 18. We set the cookie to expire in 7 days.

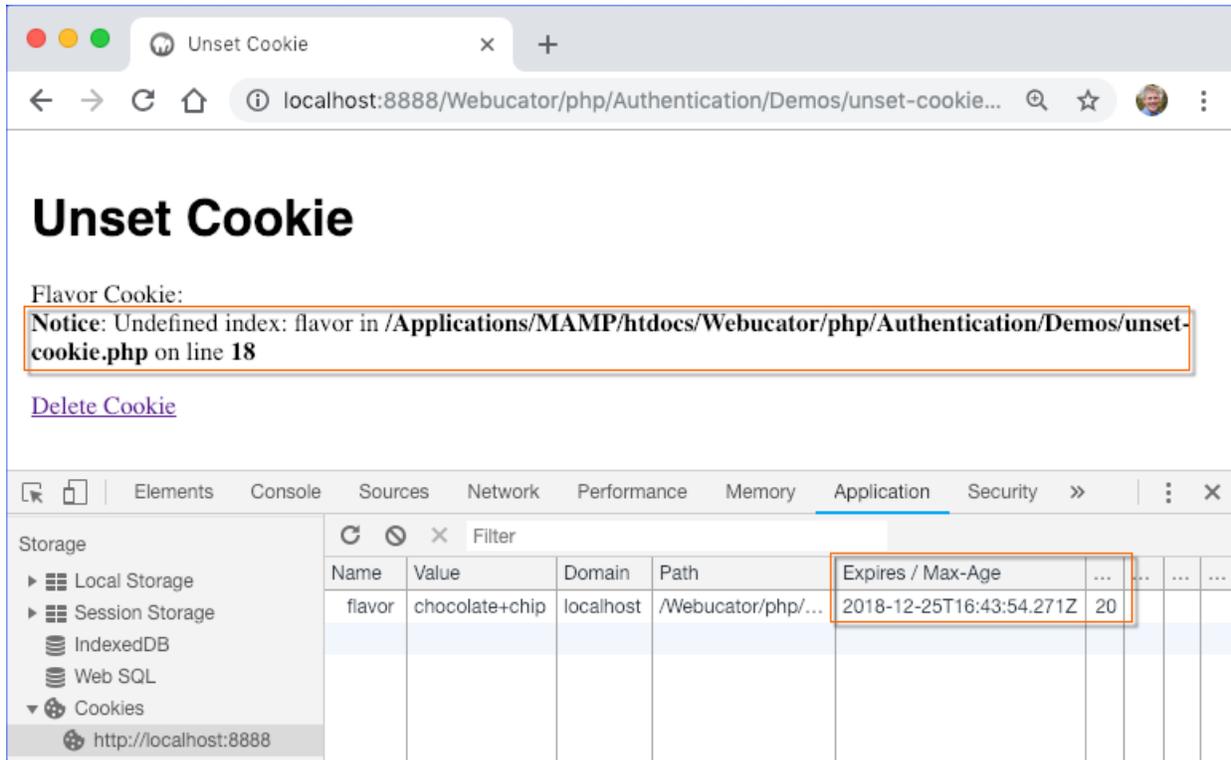
Click the **Unset Cookie** link to move to the next demo.

Demo 9.4: Authentication/Demos/unset-cookie.php

```
1. <?php
2.     ini_set('display_errors', '1');
3.
4.     unset($_COOKIE['flavor']);
5. ?>
-----Lines 6 through 22 Omitted-----
```

Code Explanation

This unsets the cookie on the server, but does not delete it on the client. When we try to read `$_COOKIE['flavor']` after unsetting it, we get a notice saying the index is undefined. Notice, however, that the expiration date for the cookie has not changed.



The screenshot shows a web browser window titled "Unset Cookie" at the URL `localhost:8888/Webucator/php/Authentication/Demos/unset-cookie...`. The page content includes the heading "Unset Cookie", the text "Flavor Cookie:", a notice box stating "Notice: Undefined index: flavor in /Applications/MAMP/htdocs/Webucator/php/Authentication/Demos/unset-cookie.php on line 18", and a link "Delete Cookie".

The browser's Application tab is open, showing a table of cookies for the domain `http://localhost:8888`. The table has columns for Name, Value, Domain, Path, Expires / Max-Age, and others. A row for the "flavor" cookie is highlighted, showing its value as "chocolate+chip" and its expiration date as "2018-12-25T16:43:54.271Z" with a max-age of "20".

Name	Value	Domain	Path	Expires / Max-Age
flavor	chocolate+chip	localhost	/Webucator/php/...	2018-12-25T16:43:54.271Z	20			

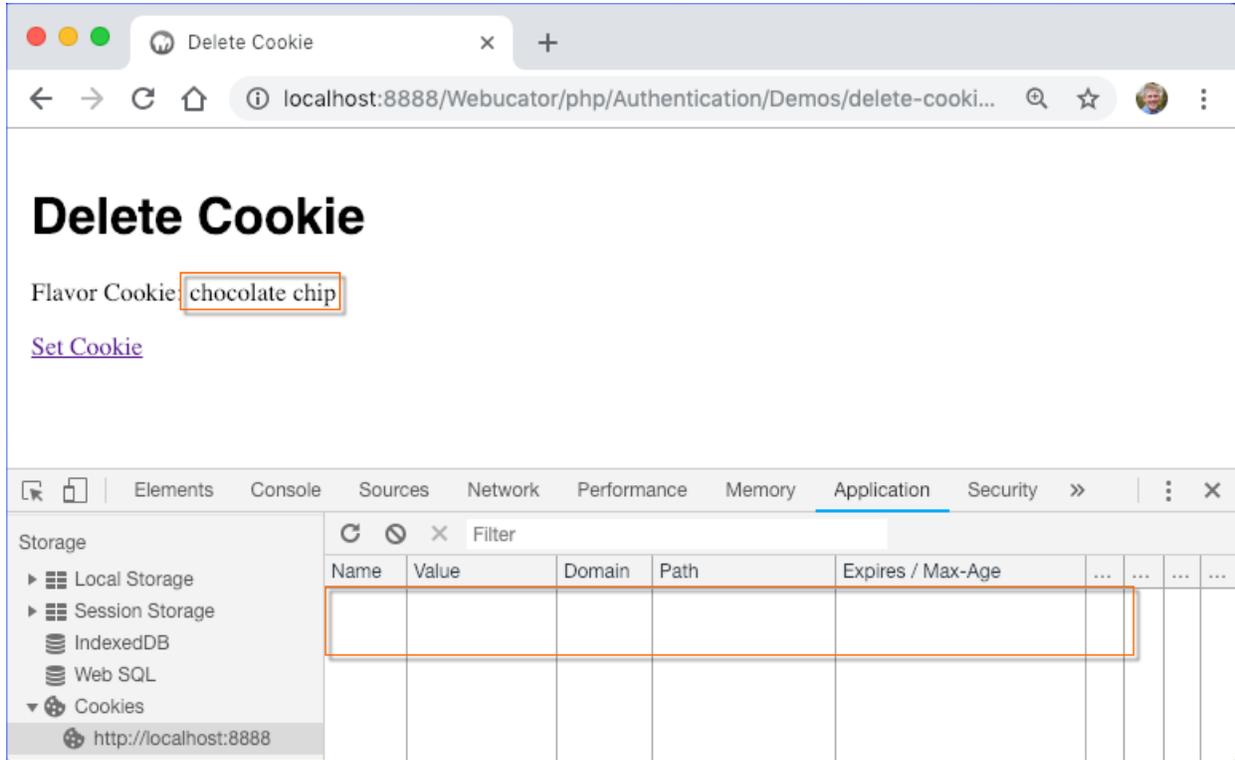
Click the [Delete Cookie](#) link to move to the next demo.

Demo 9.5: Authentication/Demos/delete-cookie.php

```
1. <?php
2.     ini_set('display_errors', '1');
3.
4.     setcookie('flavor', '', 0);
5.     ?>
   -----Lines 6 through 22 Omitted-----
```

Code Explanation

This deletes the cookie on the client, but not from the `$_COOKIES` array:



To delete the cookie on both the client and the server, navigate to `http://localhost:8888/Webucator/php/Authentication/Demos/delete-cookie-fully.php`:

Demo 9.6: Authentication/Demos/delete-cookie-fully.php

```
1. <?php
2.     ini_set('display_errors', '1');
3.
4.     unset($_COOKIE['flavor']);
5.     setcookie('flavor', '', 0);
6. ?>
-----Lines 7 through 22 Omitted-----
```

Code Explanation

This deletes the cookie on the client and on the server in the `$_COOKIES` array:

The screenshot shows a web browser window with the title "Fully Delete Cookie". The address bar shows the URL "localhost:8888/Webucator/php/Authentication/Demos/delete-cooki...". The main content area displays the heading "Fully Delete Cookie" and the text "Flavor Cookie:". Below this, a console error message is highlighted in a red box: "Notice: Undefined index: flavor in /Applications/MAMP/htdocs/Webucator/php/Authentication/Demos/delete-cookie-fully.php on line 19".

The bottom of the browser shows the "Application" tab, which contains a table of storage items. The "Cookies" section is expanded, showing a table for "http://localhost:8888". The table has columns for Name, Value, Domain, Path, Expires / Max-Age, and several empty columns. The table is currently empty.

Name	Value	Domain	Path	Expires / Max-Age

Exercise 28: Logging in

 30 to 40 minutes

In this exercise, you will create a page for logging the user in. A couple of things to note:

1. We will use a session variable that holds the user's user id to keep track of a logged-in user during a session. This is safe to do, because the user id is never shared with the browser.
2. We will use a cookie that holds a token to keep track of a user between sessions. We use a token instead of the user's id because this value is transferred between the server and client. For security reasons, we don't want to send the user's id to the client.

Exercise Code 28.1: Authentication/Exercises/phppoetry.com/login.php

```
1.  <?php
2.    $pageTitle = 'Login';
3.    require 'includes/header.php';
4.    logout(); // In case a different user has logged in
5.
6.    // TODO: Set $username and $passPhrase
7.    //     These should be set to the values posted in the form
8.    //     If the form has not been posted, they should default to
9.    //     empty strings.
10.
11.   if ($username && $passPhrase) {
12.
13.       $qLogin = "SELECT pass_phrase, registration_confirmed, user_id
14.                 FROM users
15.                 WHERE username = ?";
16.
17.       try {
18.           // TODO: Prepare and execute the $qLogin query
19.
20.           if (!$row = $stmt->fetch()) {
21.               // username doesn't exist
22.               $loginFailed = true;
23.               $failureMessage = nl2br(POEM_LOGIN_FAILED);
24.           } elseif ( !$row['registration_confirmed'] ) {
25.               // user never confirmed registration
26.               $loginFailed = true;
27.               $failureMessage = nl2br(POEM_REGISTRATION_UNCONFIRMED);
28.           } elseif (password_verify($passPhrase, $row['pass_phrase'])) {
29.               // TODO: log user in and redirect to home page
30.               //
31.               //     Set 'user-id' session variable to user_id returned by
32.               //     query.
33.
34.               //     If the user checked the remember-me checkbox, create a
35.               //     'token' cookie for 30 days (in minutes).
36.               //     To do so, follow these steps:
37.               //         Use the generateToken() utility function to
38.               //         generate the token.
39.               //         Insert a new row into the tokens table with values
40.               //         for token, user_id, and token_expires set to the
41.               //         generated token, the current user id, and a date
42.               //         30 days (in minutes) in the future.
43.               //         If the insert statement fails to execute, log the
44.               //         error and redirect to the error page.
```

```

45.         //      If the insert fails due to a PDOException, log
46.         //      the error and fail silently.
47.         //      If the insert is successful, attempt to set a
48.         //      'token' cookie that expires in 30
49.         //      days (in seconds). If this fails (i.e.,
50.         //      set_cookie() returns false), log the error.
51.         //
52.         //      After (and separate from) the cookie code,
53.         //      redirect to the home page
54.     } else {
55.         // bad password
56.         $loginFailed = true;
57.         $failureMessage = nl2br(POEM_LOGIN_FAILED);
58.     }
59. } catch (PDOException $e) {
60.     logError($e);
61.     $loginFailed = true;
62.     $failureMessage = nl2br(POEM_GENERIC_ERROR);
63. }
64. }
65. ?>
66. <main class="narrow">
67.     <h1><?= $pageTitle ?></h1>
68.     <?php
69.         if (isset($loginFailed)) {
70.             echo "<article class='poem error'>$failureMessage</article>";
71.         }
72.
73.         if (isset($_GET['just-registered'])) {
74.             echo '<article class="poem success">' .
75.                 nl2br(POEM_REGISTRATION_SUCCESS) .
76.                 '</article>';
77.         } else {
78.             echo '<p>Need an account?
79.                 <a href="register.php">Register</a></p>';
80.         }
81.     ?>
82.     <!-- novalidate set so that PHP validation can be tested -->
83.     <form method="post" action="login.php" novalidate>
84.         <label for="username">Username:</label>
85.         <input name="username" id="username" required
86.             value="<?= $username ?>">
87.         <label for="pass-phrase">Pass Phrase:</label>
88.         <input type="password" name="pass-phrase" id="pass-phrase"
89.             required>

```

```
90.     <input type="checkbox" name="remember-me" id="remember-me">
91.     <label for="remember-me" class="inline">Remember me</label>
92.     <button>Login</button>
93. </form>
94. <p class="clear">
95.     <a href="pass-phrase-reset.php">Forgot your pass phrase?</a>
96. </p>
97. </main>
98. <?php
99.     require 'includes/footer.php' ;
100. ?>
```

Code Explanation

To complete the login page:

1. Open `Authentication/Exercises/phppoetry.com/login.php` in your editor.
2. Much of this code is already done for you. Study the code and make sure you understand everything that is already done.
3. Beneath the call to `logout()`, set `$username` and `$passPhrase`.
 - A. If the form has been posted, these should be set to the values posted in the form.
 - B. If the form has not been posted, they should default to empty strings.
4. Within the `try` block after `$qLogin` is set, prepare and execute the `$qLogin` query.
5. Carefully review the `if` and first `elseif` blocks.
6. The second `elseif` statement checks if the password is correct. If it is, you will log the user in and redirect to the home page:
 - A. Set a 'user-id' session variable to the user id returned by the `$qLogin` query.
 - B. If the user checked the **remember-me** checkbox, create a 'token' cookie for 30 days (in minutes). To do so, follow these steps:
 - i. Use the `generateToken()` utility function to generate the token.
 - ii. Insert a new row into the `tokens` table with values for `token`, `user_id`, and `token_expires` set to the generated token, the current

user id, and a date 30 days (in minutes) in the future. The query should look like this:

```
INSERT INTO tokens
(token, user_id, token_expires)
VALUES ( '$token', ?, DATE_ADD(now(),
INTERVAL $interval MINUTE) )
```

- a. If the insert statement fails to execute, log the error and redirect to the error page.
- b. If the insert fails due to a PDOException, log the error and fail silently.
- c. If the insert is successful, attempt to set a 'token' cookie that expires in 30 days (in seconds). If this fails (i.e., set_cookie() returns false), log the error.

7. After (and separate from) the cookie code, redirect to the home page.
-

Exercise Code 28.2:

Authentication/Exercises/phppoetry.com/includes/header.php

```
1.  <?php
2.      // TODO: Start the session
3.      require_once 'config.php';
4.      require_once 'utilities.php';
5.      require_once 'constants.php';
6.      if ( isDebugMode() ) {
7.          ini_set('display_errors', '1');
8.      }
9.
10.     // If $db isn't already set, set it.
11.     if ( !isset($db) ) {
12.         $db = dbConnect();
13.     }
14.
15.     // TODO: Set a variable called $currentUserId, which
16.     //       will either contain the logged-in user's id or 0 (if no user
17.     //       is logged in)
18.
19.     // TODO: If 'user-id' doesn't exist in $_SESSION, check if there
20.     //       is a token cookie.
21.     //       If there is one, look for an unexpired match in the
22.     //       tokens table.
23.     //       If there is a match, set $_SESSION['user-id'] and
24.     //       $currentUserId to the user_id associated with that
25.     //       token.
26.     //       Don't forget to wrap the code connecting to the database in
27.     //       a try / catch block.
28.
29.     $pageTitleTag = empty($pageTitle)
30.         ? 'The Poet Tree Club'
31.         : $pageTitle . ' | The Poet Tree Club';
32.     ?>
-----Lines 33 through 49 Omitted-----
50. <header>
51.     <nav id="main-nav">
52.         <!-- Bar icon for mobile menu -->
53.         <div id="mobile-menu-icon">
54.             <i class="fa fa-bars"></i>
55.         </div>
56.         <ul>
57.             <li><a href="index.php">Home</a></li>
58.             <li><a href="poems.php">Poems</a></li>
```

```
59.     <li><a href="poem-submit.php">Submit Poem</a></li>
60.     <!-- TODO: If the user is logged in, include this link: -->
61.         <li><a href="my-account.php">My Account</a></li>
62.     <!-- OTHERWISE include this link: -->
63.         <li><a href="login.php">Log in / Register</a></li>
64.     <li><a href="contact.php">Contact us</a></li>
65. </ul>
66. </nav>
67. <h1>
68.     <a href="index.php">The Poet Tree Club</a>
69. </h1>
70. <h2>Set your poems free...</h2>
71. </header>
```

Code Explanation

To update header .php to keep track of logged-in users:

1. Open Authentication/Exercises/phppoetry.com/includes/header.php in your editor.
2. Much of this code is already done for you. Study the code and make sure you understand everything that is already done.
3. At the beginning of the file, write code to start or continue the session. As header.php is included in all of our files, we only have to do this in this one file.
4. Below the code that connects to the database, set a variable called `$currentUserId`, which will contain `0` if the user is not logged in and will contain the logged-in user's id if 'user-id' is set in `$_SESSION`. We will be able to use this throughout the site to check if the user is logged in.
5. If 'user-id' doesn't exist in `$_SESSION`, check if there is a token cookie.
 - A. If there is a token cookie, look for an unexpired match in the **tokens** table.
 - i. If there is a match, set `$_SESSION['user-id']` and `$currentUserId` to the `user_id` associated with that token.
 - B. Don't forget to wrap the code connecting to the database in a try / catch block.

Exercise Code 28.3: Authentication/Exercises/phppoetry.com/includes/footer.php

```
1. <footer>
2.   <span>Copyright &copy; 2019 The Poet Tree Club.</span>
3.   <nav>
4.     <!--TODO: Add the following link if the user is logged in:-->
5.     <a href="logout.php">Log out</a>
6.
7.     <a href="admin/index.php">Admin</a>
8.     <a href="about-us.php">About us</a>
9.   </nav>
10. </footer>
11. </body>
12. </html>
```

Code Explanation

To update the **logout** link in footer.php:

1. Open Authentication/Exercises/phppoetry.com/includes/footer.php in your editor.
2. Add the following link if the user is logged in:

```
<a href="logout.php">Log out</a>
```

To test your solution:

1. Navigate to `http://localhost:8888/Webucator/php/Authentication/Exercises/phppoetry.com/login.php` and log in with the wrong pass phrase. Do you get an appropriate error?
2. Now log in with the correct username and pass phrase. You should be redirected to the home page. The header should have a **My Account** link instead of **Log in / Register** link and the footer should have a **logout** link.

Solution: Authentication/Solutions/phppoetry.com/login.php

```
1. <?php
2.     $pageTitle = 'Login';
3.     require 'includes/header.php';
4.     logout(); // In case a different user has logged in
5.
6.     $username = trim($_POST['username'] ?? '');
7.     $passPhrase = $_POST['pass-phrase'] ?? '';
8.
9.     if ($username && $passPhrase) {
10.
11.         $sqlLogin = "SELECT pass_phrase, registration_confirmed, user_id
12.             FROM users
13.             WHERE username = ?";
14.
15.         try {
16.             $stmt = $db->prepare($sqlLogin);
17.             $stmt->execute([$username]);
18.
19.             if (!$row = $stmt->fetch()) {
20.                 // username doesn't exist
21.                 $loginFailed = true;
22.                 $failureMessage = nl2br(POEM_LOGIN_FAILED);
23.             } elseif ( !$row['registration_confirmed'] ) {
24.                 // user never confirmed registration
25.                 $loginFailed = true;
26.                 $failureMessage = nl2br(POEM_REGISTRATION_UNCONFIRMED);
27.             } elseif (password_verify($passPhrase, $row['pass_phrase'])) {
28.                 // log user in and redirect to home page
29.                 $_SESSION['user-id'] = $row['user_id'];
30.
31.                 if (!empty($_POST['remember-me'])) {
32.                     // Set cookie for 30 days
33.                     $interval = 30 * 24 * 60; // 30 days
34.                     $token = generateToken();
35.                     $sqlInsert = "INSERT INTO tokens
36.                         (token, user_id, token_expires)
37.                         VALUES ( '$token', ?, DATE_ADD(now(),
38.                             INTERVAL $interval MINUTE) )";
39.
40.                     try {
41.                         $stmtInsert = $db->prepare($sqlInsert);
42.
43.                         if ($stmtInsert->execute( [$_SESSION['user-id']] )) {
44.                             $expiration = time() + 60 * $interval;
```

```
45.         if (!setcookie('token', $token, $expiration)) {
46.             // Could not set cookie on browser. Fail silently.
47.             logError("Could not set cookie for $username.");
48.         }
49.     } else {
50.         // Likely SQL error. Log and redirect to error page.
51.         logError($stmtInsert->errorinfo()[2], true);
52.     }
53. } catch (PDOException $e) {
54.     // Could not insert cookie token. Fail silently.
55.     logError($e);
56. }
57. }
58. header("Location: index.php");
59. } else {
60.     // bad password
61.     $loginFailed = true;
62.     $failureMessage = nl2br(POEM_LOGIN_FAILED);
63. }
64. } catch (PDOException $e) {
65.     logError($e);
66.     $loginFailed = true;
67.     $failureMessage = nl2br(POEM_GENERIC_ERROR);
68. }
69. }
70. ?>
```

-----Lines 71 through 105 Omitted-----

Solution: Authentication/Solutions/phppoetry.com/includes/header.php

```
1.  <?php
2.      session_start();
3.      require_once 'config.php';
4.      require_once 'utilities.php';
5.      require_once 'constants.php';
6.      if (isDebugMode()) {
7.          ini_set('display_errors', '1');
8.      }
9.
10.     // If $db isn't already set, set it.
11.     if (!isset($db)) {
12.         $db = dbConnect();
13.     }
14.
15.     $currentUserId = $_SESSION['user-id'] ?? 0;
16.     if (!$currentUserId) {
17.         // Do we remember this user?
18.         if (isset($_COOKIE['token'])) {
19.             $qSelect = "SELECT user_id
20.             FROM tokens
21.             WHERE token = ? AND token_expires > now()";
22.
23.             try {
24.                 $stmt = $db->prepare($qSelect);
25.                 $stmt->execute([$_COOKIE['token']]);
26.
27.                 if ($row = $stmt->fetch()) {
28.                     // Found unexpired matching token
29.                     $_SESSION['user-id'] = $row['user_id'];
30.                     $currentUserId = $row['user_id'];
31.                 }
32.             } catch (PDOException $e) {
33.                 logError($e);
34.             }
35.         }
36.     }
37.
38.     $pageTitleTag = empty($pageTitle)
39.         ? 'The Poet Tree Club'
40.         : $pageTitle . ' | The Poet Tree Club';
41. ?>
-----Lines 42 through 58 Omitted-----
59. <header>
60.     <nav id="main-nav">
```

```

61.     <!-- Bar icon for mobile menu -->
62.     <div id="mobile-menu-icon">
63.         <i class="fa fa-bars"></i>
64.     </div>
65.     <ul>
66.         <li><a href="index.php">Home</a></li>
67.         <li><a href="poems.php">Poems</a></li>
68.         <li><a href="poem-submit.php">Submit Poem</a></li>
69.         <?php if ($currentUserId) { ?>
70.             <li><a href="my-account.php">My Account</a></li>
71.         <?php } else { ?>
72.             <li><a href="login.php">Log in / Register</a></li>
73.         <?php } ?>
74.         <li><a href="contact.php">Contact us</a></li>
75.     </ul>
76. </nav>
77. <h1>
78.     <a href="index.php">The Poet Tree Club</a>
79. </h1>
80. <h2>Set your poems free...</h2>
81. </header>

```

Evaluation
Copy

Solution: Authentication/Solutions/phppoetry.com/includes/footer.php

```

1. <footer>
2.     <span>Copyright &copy; 2019 The Poet Tree Club.</span>
3.     <nav>
4.         <?php
5.             if ($currentUserId) {
6.                 echo '<a href="logout.php">Log out</a>';
7.             }
8.         ?>
9.         <a href="admin/index.php">Admin</a>
10.        <a href="about-us.php">About us</a>
11.    </nav>
12. </footer>
13. </body>
14. </html>

```



9.6. Logging Out

The logout page is very simple. It makes use of the following `logout()` function from the `utilities.php` file:

```
function logout() {
    unset($_SESSION['user-id']);
    unset($_COOKIE['token']); // unset on server
    setcookie('token', '', 0); // unset on client
}
```

Here is the code:

Demo 9.7: Authentication/Exercises/phppoetry.com/logout.php

```
1. <?php
2.     require_once 'includes/utilities.php';
3.     session_start();
4.     logout();
5.     header("Location: index.php");
6.     ?>
```



To test logging out:

1. Log in at <http://localhost:8888/Webucator/php/Authentication/Exercises/phppoetry.com/login.php> using the account you created earlier.
2. Click the **Logout** link in the footer. You should be redirected to the home page and no longer be logged in.



9.7. \$_REQUEST Variables

All variables in the `$_GET`, `$_POST`, and `$_COOKIE` superglobals are combined into the `$_REQUEST` superglobal array. The following example illustrates this:

Demo 9.8: Authentication/Demos/request.php

```
-----Lines 1 through 13 Omitted-----
14. <main id="request-variables">
15.   <h4>REQUEST</h4>
16.   <?= $_REQUEST['greeting'] ?>
17.   <h4>GET</h4>
18.   <?= $_GET['greeting'] ?>
19.   <h4>POST</h4>
20.   <?= $_POST['greeting'] ?>
21.   <h3>Set Variables</h3>
22.   <div>
23.     <a href="request.php?greeting=Hello,+get!">Hello, get!</a>
24.   </div>
25.   <div>
26.     <form method="post" action="request.php">
27.       <button name="greeting" value="Hello, post!">
28.         Hello, post!
29.       </button>
30.     </form>
31.   </div>
32. </main>
33. </body>
34. </html>
```

Code Explanation

At the top of the page, we display `$_REQUEST['greeting']`, `$_GET['greeting']`, and `$_POST['greeting']`. And at the bottom of the page, we provide a link and a form button for setting greeting via the query string and a form post, respectively. To see how it works:

1. Open `http://localhost:8888/Webucator/php/Authentication/Demos/request.php` in your browser. The first time you visit, no variables exist in the `$_GET` or `$_POST` arrays, and so none exist in the `$_REQUEST` either:

Request Variables

localhost:8888/Webucator/php/Authentication/Demos/r...

REQUEST

Notice: Undefined index: greeting in /Applications/MAMP/htdocs/Webucator/php/Authentication/Demos/request.php on line 16

GET

Notice: Undefined index: greeting in /Applications/MAMP/htdocs/Webucator/php/Authentication/Demos/request.php on line 18

POST

Notice: Undefined index: greeting in /Applications/MAMP/htdocs/Webucator/php/Authentication/Demos/request.php on line 20

Set Variables

[Hello, get!](#)

2. Now click the **Hello, get!** link. Notice that “Hello, get!” shows up in `$_REQUEST` as well as `$_GET`.

Request Variables

localhost:8888/Webucator/php/Authentication/Demos/r...

REQUEST

Hello, get!

GET

Hello, get!

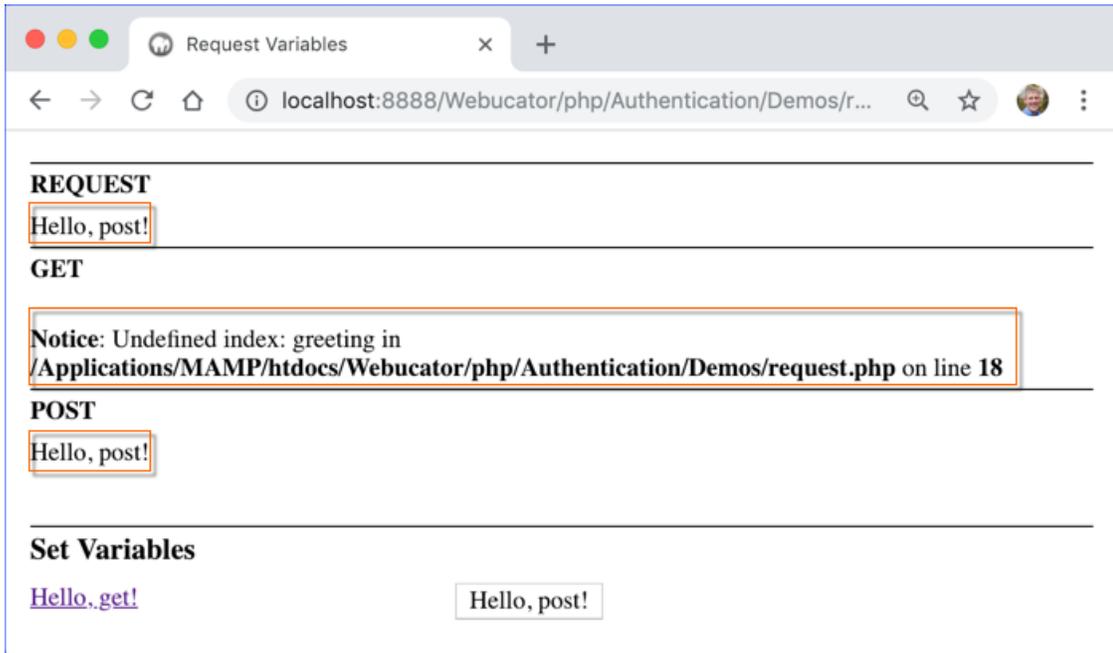
POST

Notice: Undefined index: greeting in /Applications/MAMP/htdocs/Webucator/php/Authentication/Demos/request.php on line 20

Set Variables

[Hello, get!](#)

- Now click the **Hello, post!** button. Notice that “Hello, post!” shows up in `$_REQUEST` as well as `$_POST`.



Exercise 29: Resetting the Pass Phrase

 30 to 45 minutes

In this exercise, you will review and comment pages for resetting the pass phrase. The process is similar to a new registration:

1. From the login page, the user clicks on the **Forgot your pass phrase?** link, taking them to `pass-phrase-reset.php`.
2. On `pass-phrase-reset.php`, the user enters their email and submits the form.
3. If the email is found, a token is generated and saved in the **tokens** table, and an email is sent to the user's email address with a link to `pass-phrase-reset-confirm.php` with the token on the query string.
4. On `pass-phrase-reset-confirm.php`, the user will be able to reset their pass phrase, unless the token has timed out.

The pages are already complete and functioning. Your job is to review them and add detailed comments explaining what each piece of code does.

Exercise Code 29.1:

Authentication/Exercises/phppoetry.com/pass-phrase-reset.php

```
1.  <?php
2.      require 'mail-config.php';
3.
4.      $pageTitle = 'Pass Phrase Reset';
5.      require 'includes/header.php';
6.      logout();
7.
8.      $errors = [];
9.
10.     $email = trim($_POST['email'] ?? '');
11.     ?>
12.     <main id="pass-phrase-reset" class="narrow">
13.     <?php
14.     if (isset($_POST['submit'])) {
15.         if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
16.             $errors[] = 'You must enter a valid email.';
17.         } else {
18.             $sqlUser = "SELECT first_name, last_name, user_id
19.                         FROM users WHERE email = ?";
20.
21.             try {
22.                 $stmt = $db->prepare($sqlUser);
23.                 $stmt->execute([$email]);
24.                 $row = $stmt->fetch();
25.             } catch (PDOException $e) {
26.                 logError($e);
27.                 $errors[] = nl2br(POEM_GENERIC_ERROR);
28.             }
29.
30.             if (!empty($row)) {
31.                 $userId = $row['user_id'];
32.                 $firstName = $row['first_name'];
33.                 $lastName = $row['last_name'];
34.
35.                 $token = generateToken();
36.
37.                 $sqlInsert = "INSERT INTO tokens
38.                               (token, user_id, token_expires)
39.                               VALUES ('$token', ?, DATE_ADD(now(), INTERVAL 1 HOUR))";
40.
41.                 try {
42.                     $stmtInsert = $db->prepare($sqlInsert);
43.
```

```

44.     if ($stmtInsert->execute( [$userId] )) {
45.         $qs = http_build_query(['token' => $token]);
46.         $confPath = getFullPath('pass-phrase-reset-confirm.php');
47.         $href = $confPath . '?' . $qs;
48.
49.         $to = $email;
50.         $toName = $firstName . ' ' . $lastName;
51.         $subject = 'Pass phrase reset';
52.
53.         $html = "<p>A reset-pass-phrase request has been made
54.         for your account for phppoetry.com. If you didn't make the
55.         request, you can ignore this email. If you did, reset your
56.         pass phrase by <a href='$href'>clicking here</a>.</p>";
57.
58.         $text = "A reset-pass-phrase request has been made
59.         for your account for phppoetry.com. If you didn't make the
60.         request, you can ignore this email. If you did, visit $href
61.         to reset your pass phrase.";
62.
63.         $mail = createMailer();
64.         $mail->addAddress($to, $toName);
65.         $mail->Subject = $subject;
66.         $mail->Body = $html;
67.         $mail->AltBody = $text;
68.
69.         if (!$mail->send()) {
70.             echo '<p class="error">
71.             We could not send you a pass-phrase-reset email.</p>';
72.             echo "Mailer Error: " . $mail->ErrorInfo;
73.         } else {
74.             echo '<p class="success">We have sent you an email with
75.             instructions. Check your email.</p>';
76.         }
77.     } else {
78.         logError($stmtInsert->errorinfo()[2], true);
79.     }
80. } catch (PDOException $e) {
81.     logError($e);
82.     $errors[] = nl2br(POEM_GENERIC_ERROR);
83. }
84. } else {
85.     $errors[] = "Sorry. We don't recognize that email.";
86. }
87. }
88. }

```



```
89.
90. if (!isset($_POST['submit']) || $errors) {
91.
92.     if ($errors) {
93.         echo '<h3>Uh oh!</h3>';
94.         foreach ($errors as $error) {
95.             echo "<p class='error'>$error</p>";
96.         }
97.     }
98.
99.     echo "<p>Use the form below to reset your pass phrase:</p>
100.     <form method='post' novalidate>
101.         <label for='email'>Email:</label>
102.         <input type='email' name='email' id='email'
103.             value='$email' required>
104.         <button name='submit' value='1' class='wide'>
105.             Reset Pass Phrase</button>
106.     </form>";
107. }
108. ?>
109. </main>
110. <?php
111.     require 'includes/footer.php';
112. ?>
```

Evaluation
Copy

Code Explanation

1. Open `Authentication/Exercises/phppoetry.com/pass-phrase-reset.php` in your editor.
 2. Review and add detailed comments to the page. The goal is to prove to yourself that you understand every line of code in the file by explaining what it does in a comment.
 3. Note that we check `$_REQUEST` for 'token'. We use `$_REQUEST` because 'token' will be in the `$_GET` array when the user visits from the link sent to their email and it will be in the `$_POST` array when the user submits the form to confirm their new pass phrase.
 4. If there is any line or block of code that you do not understand, add a comment trying to explain the point of confusion. Attempt to resolve it yourself through studying past demos and exercises and looking at the PHP documentation. If you still are not able to figure it out, review the solution.
-

Exercise Code 29.2:

Authentication/Exercises/phppoetry.com/pass-phrase-reset-confirm.php

```
1.  <?php
2.    if (!isset($_REQUEST['token'])) {
3.        header("Location: index.php");
4.    }
5.
6.    $pageTitle = 'Reset Password Form';
7.    require 'includes/header.php';
8.    logout();
9.
10.   $showForm = true;
11.   ?>
12.   <main class="narrow">
13.   <h1><?= $pageTitle ?></h1>
14.
15.   <?php
16.       if ($showForm && isset($_POST['token'])) {
17.           $token = $_POST['token'];
18.           $passPhrase1 = $_POST['pass-phrase-1'];
19.           $passPhrase2 = $_POST['pass-phrase-2'];
20.
21.           if (strlen($passPhrase1) < 20) {
22.               $error = 'Your pass phrase must be at least 20 characters.';
23.           } elseif ($passPhrase1 != $passPhrase2) {
24.               $error = 'Your pass phrases don\'t match.';
25.           } else {
26.               $showForm = false;
27.
28.               $hashedPhrase = password_hash($passPhrase1, PASSWORD_DEFAULT);
29.
30.               $qUpdate = "UPDATE users
31.                   SET pass_phrase = '$hashedPhrase',
32.                       registration_confirmed = 1
33.                   WHERE user_id = (SELECT user_id
34.                                   FROM tokens
35.                                   WHERE token = ?
36.                                   AND token_expires > now() );";
37.
38.               try {
39.                   $stmtUpdate = $db->prepare($qUpdate);
40.
41.                   if ($stmtUpdate->execute( [$token] )) {
42.                       echo "<span class='success'>Success.
43.                           <a href='login.php'>Login</a></span>";
```

```

44.         } else {
45.             logError($stmtUpdate->errorinfo()[2], true);
46.         }
47.     } catch (PDOException $e) {
48.         logError($e);
49.         $error = nl2br(POEM_GENERIC_ERROR) .
50.             '<p><a href="pass-phrase-reset.php">try again</a></p>';
51.     }
52. }
53. } elseif ($showForm) {
54.     $token = $_GET['token'];
55.
56.     $qSelect = "SELECT *
57.         FROM tokens
58.         WHERE token = ? AND token_expires > now()";
59.
60.     try {
61.         $stmt = $db->prepare($qSelect);
62.         $stmt->execute([$token]);
63.
64.         if (!$stmt->fetch()) {
65.             $error = nl2br(POEM_INVALID_TOKEN_PASS_PHRASE_RESET);
66.             $showForm = false;
67.         }
68.     } catch (PDOException $e) {
69.         logError($e);
70.         $error = nl2br(POEM_GENERIC_ERROR) .
71.             '<p><a href="pass-phrase-reset.php">try again</a></p>';
72.     }
73. }
74. ?>
75. <?php
76.     if (isset($error)) {
77.         echo "<article class='poem error'>$error</article>";
78.     }
79.
80.     if ($showForm) {
81.     ?>
82.     <form method="post" action="pass-phrase-reset-confirm.php" novalidate>
83.         <input type="hidden" name="token" value="<?= $token ?>">
84.         <fieldset>
85.             <legend>Pass Phrase:</legend>
86.             <em>A hard-to-guess phrase at least 20 characters long.</em>
87.             <input type="password" placeholder="Pass Phrase"
88.                 name="pass-phrase-1" id="pass-phrase-1"

```

```
89.         required minlength="20">
90.     <input type="password" placeholder="Repeat Pass Phrase"
91.         name="pass-phrase-2" id="pass-phrase-2"
92.         required minlength="20">
93.     </fieldset>
94.     <button>Change Pass Phrase</button>
95. </form>
96. <?php
97.     }
98.     echo '</main>';
99.     require 'includes/footer.php';
100. ?>
```

Code Explanation

To complete this page:

1. Open `Authentication/Exercises/phppoetry.com/pass-phrase-reset-confirm.php` in your editor.
 2. Review and add detailed comments to the page.
 3. If there is any line or block of code that you do not understand, add a comment trying to explain the point of confusion. Attempt to resolve it yourself through studying past demos and exercises and looking at the PHP documentation. If you still are not able to figure it out, review the solution.
-

Solution

To see the commented solutions, open the following files in your editor:

1. Authentication/Solutions/phppoetry.com/pass-phrase-reset.php
2. Authentication/Solutions/phppoetry.com/pass-phrase-reset-confirm.php

Evaluation
Copy

Conclusion

In this lesson, you have learned to create a full registration / authentication process and to track users using session and cookie variables.

LESSON 10

LAB: Inserting, Updating, and Deleting Poems

Topics Covered

- Inserting.
- Updating.
- Deleting.

Introduction

*Evaluation
Copy*

This lesson consists of four exercises:

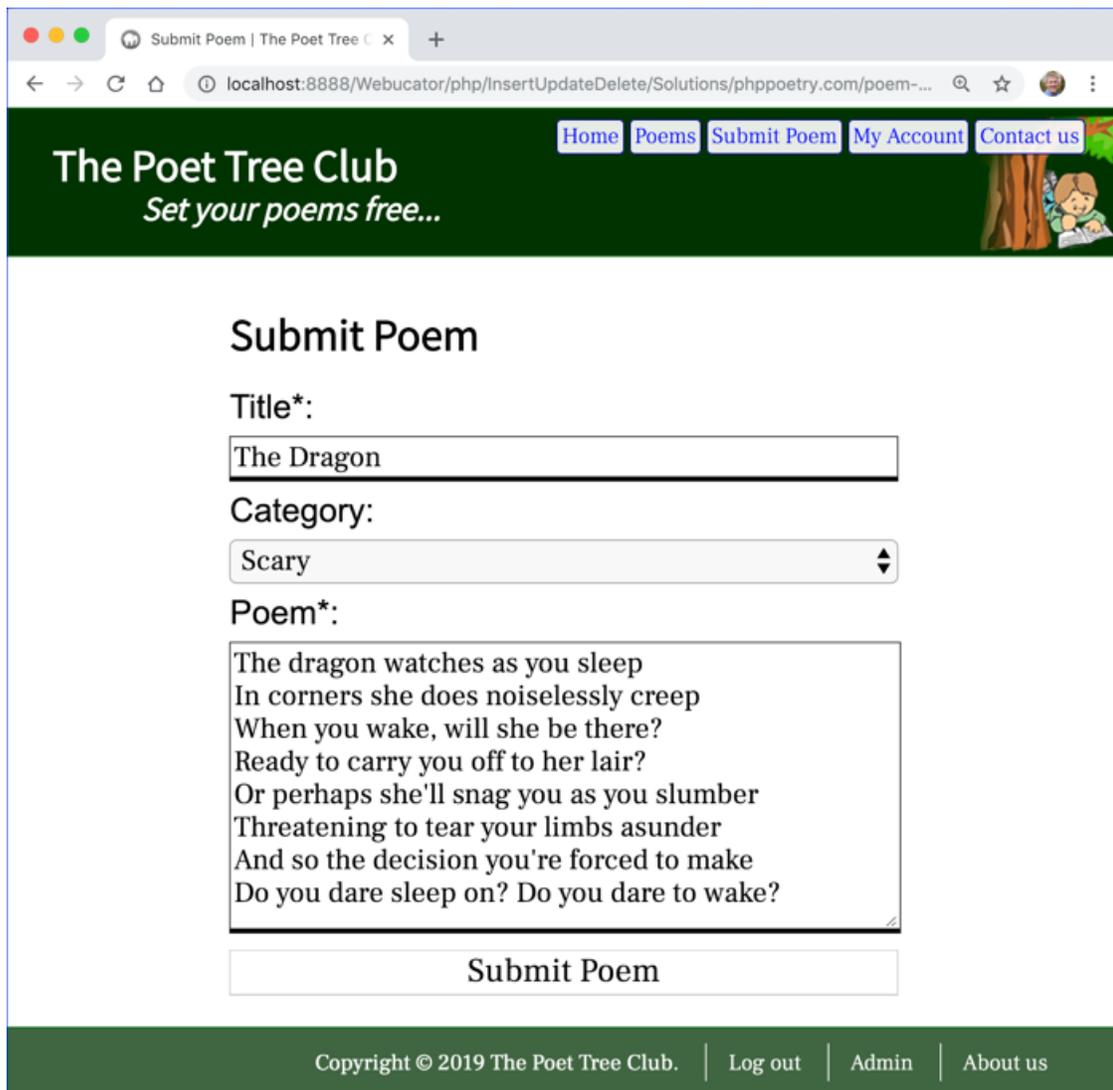
1. Creating a page to submit a new poem.
2. Modifying the poems page so that it shows all the current user's poems, even those not yet approved.
3. Creating a page to edit an existing poem.
4. Creating a page to delete a poem.

Exercise 30: Submitting a New Poem

🕒 45 to 90 minutes

In this exercise, you will write a PHP page from scratch for submitting a new poem.

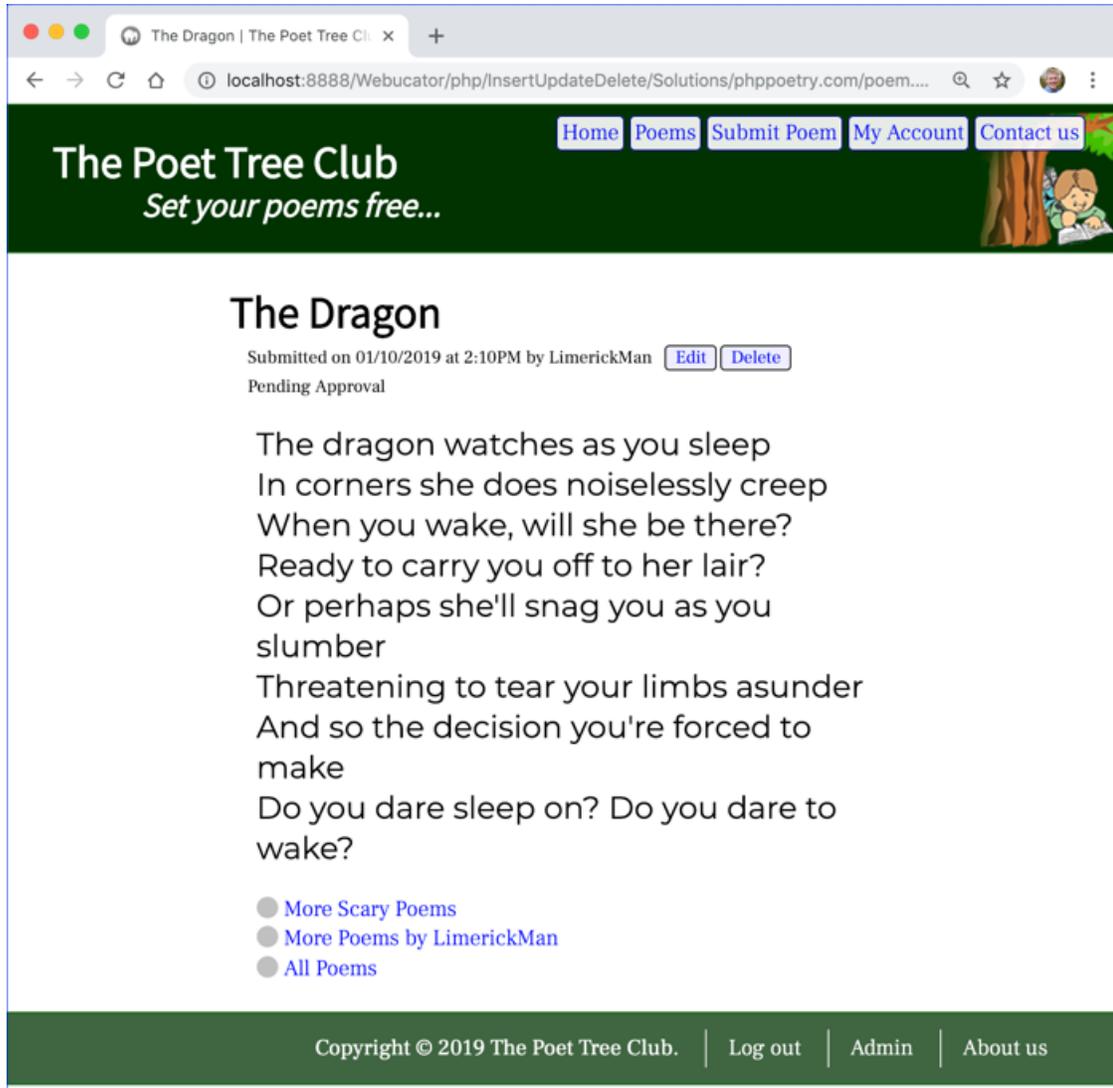
1. Open a new document and save it as `poem-submit.php` in the `InsertUpdateDelete/Exercises/phppoetry.com` folder.
2. Write code to create a page that looks like the one below:



The screenshot shows a web browser window with the following content:

- Browser tab: Submit Poem | The Poet Tree Club
- Address bar: localhost:8888/Webucator/php/InsertUpdateDelete/Solutions/phppoetry.com/poem-...
- Navigation menu: Home, Poems, Submit Poem, My Account, Contact us
- Header: The Poet Tree Club, Set your poems free...
- Form title: Submit Poem
- Form fields:
 - Title*: The Dragon
 - Category: Scary
 - Poem*: The dragon watches as you sleep
In corners she does noiselessly creep
When you wake, will she be there?
Ready to carry you off to her lair?
Or perhaps she'll snag you as you slumber
Threatening to tear your limbs asunder
And so the decision you're forced to make
Do you dare sleep on? Do you dare to wake?
- Submit button: Submit Poem
- Footer: Copyright © 2019 The Poet Tree Club. | Log out | Admin | About us

3. If the poem title is missing, or no category was selected, or the poem length is shorter than 10 characters, report an error and reshew the form with the user's entries still in it. Otherwise, insert the poem into the database and redirect to the appropriate poem page:



4. We have covered all the PHP you need to be able to do this. You will need two SQL queries:

A. One for getting the list of categories:

```
SELECT c.category_id, c.category
FROM categories c
LEFT JOIN poems p ON c.category_id = p.category_id
GROUP BY c.category_id
ORDER BY c.category
```

B. One for inserting the poem:

```
INSERT INTO poems
(title, poem, category_id, user_id, date_approved)
VALUES (:title, :poem, :category_id, :user_id,
        $dateApproved)
```

You will need to bind the parameters to values submitted through the form. `$dateApproved` should be set to 'null'. In the future, you might want to change this so that when an administrator submits a new poem it is approved immediately. Administrators will be able to approve new poems through an admin interface to be built later.

5. Be sure to write code to catch and log potential exceptions.
6. Note that only authenticated users should be able to reach this page. Any other user should be immediately redirected to the login page with a `no-access` URL parameter set to 1: `login.php?no-access=1`

Login

We're happy you're here
You're welcome, my friend
Log in or register
And then try it again

Username:

Pass Phrase:

Remember me

[Forgot your pass phrase?](#)

7. Open `login.php` in your editor. After the `if (isset($_GET['just-registered']))` check, add an `elseif` block that checks if `no-access` exists in the `$_GET` array. If it does, output the `POEM_ACCESS_DENIED` constant just as we did with the `POEM_REGISTRATION_SUCCESS` constant in the `if` block.
8. Open `poem.php` in your editor. Modify it so that it outputs “Pending Approval” instead of “Date Approved:” if the poem has not yet been approved.
9. Test your solution in a browser by navigating to Visit `http://localhost:8888/We bucator/php/InsertUpdateDelete/Exercises/phppoetry.com/poem-submit.php` and writing and submitting a new poem.

Solution: InsertUpdateDelete/Solutions/phppoetry.com/poem-submit.php

```
1. <?php
2.     $pageTitle = 'Submit Poem';
3.     require 'includes/header.php';
4.
5.     if (!isAuthenticated()) {
6.         header("Location: login.php?no-access=1");
7.     }
8.
9.     $errors = [];
10.    $f['category'] = $_POST['cat'] ?? 0;
11.    $f['title'] = trim($_POST['title'] ?? '');
12.    $f['poem'] = trim($_POST['poem'] ?? '');
13.
14.    $qCategories = "SELECT c.category_id, c.category
15.        FROM categories c
16.        LEFT JOIN poems p ON c.category_id = p.category_id
17.        GROUP BY c.category_id
18.        ORDER BY c.category";
19.
20.    try {
21.        $stmtCats = $db->prepare($qCategories);
22.        $stmtCats->execute();
23.    } catch (PDOException $e) {
24.        logError($e);
25.        $errors[] = 'Oops. Our bad. Cannot get categories.';
26.    }
27.
28.    if (!empty($_POST['submit'])) {
29.        // Validate Form Entries
30.        if (!$f['title']) {
31.            $errors[] = 'You must enter a poem title.';
32.        }
33.        if (!$f['category']) {
34.            $errors[] = 'You must select a category.';
35.        }
36.        if (!$f['poem'] || strlen($f['poem']) < 10) {
37.            $errors[] = 'Your poem must be at least 10 characters.';
38.        }
39.
40.        if (!$errors) {
41.            // Insert poem
42.            $dateApproved = 'null'; // For now
43.
44.            $qInsert = "INSERT INTO poems
```

```

45.     (title, poem, category_id, user_id, date_approved)
46.     VALUES (:title, :poem, :category_id, :user_id,
47.             $dateApproved);";
48.
49.     try {
50.         $stmtInsert = $db->prepare($qInsert);
51.         $stmtInsert->bindParam(':title', $f['title']);
52.         $stmtInsert->bindParam(':poem', $f['poem']);
53.         $stmtInsert->bindParam(':category_id', $f['category']);
54.         $stmtInsert->bindParam(':user_id', $currentUserId);
55.         $stmtInsert->execute();
56.
57.         $lastInsertId = $db->lastInsertId();
58.         header("Location: poem.php?poem-id=$lastInsertId");
59.     } catch (PDOException $e) {
60.         logError($e);
61.         $errors[] = 'Oops. Our bad. Cannot insert poem.';
62.     }
63. }
64. }
65. ?>
66. <main id="poem-submit">
67.     <h1><?= $pageTitle ?></h1>
68.     <?php
69.         if ($errors) {
70.             echo '<h3>Please correct the following errors:</h3>';
71.             <ol class="error">';
72.             foreach ($errors as $error) {
73.                 echo "<li>$error</li>";
74.             }
75.             echo '</ol>';
76.         }
77.     ?>
78.     <form method="post" action="poem-submit.php" novalidate>
79.         <label for="title">Title*:</label>
80.         <input name="title" id="title"
81.             value="<?= $f['title'] ?>" required>
82.         <label for="cat">Category:</label>
83.         <select name="cat" id="cat">
84.             <option value="0">--Choose a Category</option>
85.         <?php
86.             while ($row = $stmtCats->fetch()) {
87.                 $category = $row['category'];
88.                 $categoryId = $row['category_id'];
89.                 $selected = $categoryId === $f['category'] ?

```



```

90.         ' selected' : '';
91.         echo "<option value='$categoryId'$selected>
92.             $category
93.         </option>";
94.     }
95.     ?>
96. </select>
97. <label for="poem">Poem*:</label>
98. <textarea id="poem" name="poem"><?= $f['poem'] ?></textarea>
99. <button name="submit" value="1" class="wide">Submit Poem</button>
100. </form>
101. </main>
102. <?php
103.     require 'includes/footer.php';
104. ?>

```

Solution: InsertUpdateDelete/Solutions/phppoetry.com/poem-1.php

```

-----Lines 1 through 54 Omitted-----
55. <div id="approval-status">
56.     <?php
57.         if ($dateApproved) {
58.             echo 'Approved: ' . date('m/d/Y', strtotime($dateApproved));
59.         } else {
60.             echo 'Pending Approval';
61.         }
62.     ?>
63. </div>
-----Lines 64 through 95 Omitted-----

```

Solution: InsertUpdateDelete/Solutions/phppoetry.com/login.php

```
-----Lines 1 through 77 Omitted-----
78.     if (isset($_GET['just-registered'])) {
79.         echo '<article class="poem success">' .
80.             nl2br(POEM_REGISTRATION_SUCCESS) .
81.             '</article>';
82.     } elseif (isset($_GET['no-access'])) {
83.         echo '<article class="poem success">' .
84.             nl2br(POEM_ACCESS_DENIED) .
85.             '</article>';
86.     } else {
87.         echo '<p>Need an account?
88.             <a href="register.php">Register</a></p>';
89.     }
-----Lines 90 through 109 Omitted-----
```

Exercise 31: Showing All User's Poems in on the Poems Page

 25 to 40 minutes

In this exercise, you will modify `poems.php` so that users can see all their own poems, even the ones that have not yet been approved, in the poems table.

1. Change the WHERE clause in **both** the `$query` and `$qPoemCount` variables to:

```
WHERE (p.date_approved IS NOT NULL OR u.user_id = ?)
```

2. When first initiation `$params`, include `$currentUserId` in the array, like this:

```
$params = [$currentUserId];
```

This will result in an effective WHERE clause of:

```
WHERE (p.date_approved IS NOT NULL OR u.user_id = $currentUserId)
```

3. Within the `while` loop within the `tbody`, use the ternary operator to define a `$cls` variable. The value should be 'normal' if the `$row['date_approved']` is not null (remember that null is *falsey*). Otherwise, the value should be 'pending-approval'.
4. Currently, `$published` is getting a date based on the value of `$row['date_approved']`. But if `$row['date_approved']` is null, then `$published` should get "N/A", for not applicable.
5. Finally, change the open `<tr>` tag to use the `$cls` class.

Solution: InsertUpdateDelete/Solutions/phppoetry.com/poems.php

```
-----Lines 1 through 23 Omitted-----
24.     $query = "SELECT p.poem_id, p.title, p.date_approved,
25.     c.category, u.username
26.         FROM poems p
27.         JOIN categories c ON c.category_id = p.category_id
28.         JOIN users u ON u.user_id = p.user_id
29.         WHERE (p.date_approved IS NOT NULL
30.             OR u.user_id = ?)";
31.
32.
33.     $selCatId = $_GET['cat'] ?? 0; // category_id
34.     $selUserId = $_GET['user'] ?? 0; // user_id
35.     $whereConditions = [];
36.     $params = [$currentUserId];
37.
-----Lines 38 through 65 Omitted-----
66.     $qPoemCount = "SELECT COUNT(p.poem_id) AS num
67.     FROM poems p
68.         JOIN categories c ON c.category_id = p.category_id
69.         JOIN users u ON u.user_id = p.user_id
70.     WHERE (p.date_approved IS NOT NULL
71.         OR u.user_id = ?)";
-----Lines 72 through 188 Omitted-----
184.     <tbody>
185.     <?php
186.         while ($row = $stmt->fetch()) {
187.             $cls = ($row['date_approved']
188.                 ? 'normal'
189.                 : 'pending-approval';
190.             if ($row['date_approved']) {
191.                 $approved = strtotime($row['date_approved']);
192.                 $published = date('m/d/Y', $approved);
193.             } else {
194.                 $published = 'N/A';
195.             }
196.         ?>
197.         <tr class="<?=$cls ?>">
198.             <td>
199.                 <a href="poem.php?poem-id=<?=$row['poem_id'] ?>">
200.                     <?=$row['title'] ?>
201.                 </a>
202.             </td>
203.             <td><?=$row['category'] ?></td>
```

```
204.         <td><?= $row['username'] ?></td>
205.         <td><?= $published ?></td>
206.     </tr>
207.     <?php } ?>
208. </tbody>
```

-----Lines 209 through 268 omitted-----

**Evaluation
Copy**

Exercise 32: Editing an Existing Poem

 45 to 90 minutes

In this exercise, you will write a PHP page from scratch for editing an existing poem. Note that you will be making use of a new `isPoemAuthor()` function in `utilities.php`:

Exercise Code 32.1:

InsertUpdateDelete/Solutions/phppoetry.com/includes/utilities.php

```
-----Lines 1 through 28 Omitted-----
29.     function isPoemAuthor($poemId, $userId = null) {
30.         /*
31.          * Check if user is author of poem
32.          * $userId defaults to logged-in user id
33.          */
34.         $db = dbConnect();
35.         if (!$userId && !isAuthenticated()) {
36.             return false;
37.         }
38.         $userId = $userId ?? $_SESSION['user-id'];
39.         $q = 'SELECT user_id FROM poems WHERE poem_id = ?';
40.         try {
41.             $stmt = $db->prepare($q);
42.             $stmt->execute([$poemId]);
43.             $row = $stmt->fetch();
44.         } catch (PDOException $e) {
45.             logError($e);
46.             return false;
47.         }
48.
49.         return($row['user_id'] === $userId);
50.     }
-----Lines 51 through 121 Omitted-----
```

1. Open a new document and save it as `poem-edit.php` in the `InsertUpdateDelete/Exercises/phppoetry.com` folder.
2. Write code to create a page that looks like the one below:

The screenshot shows a web browser window with the following elements:

- Browser Tab:** Edit Poem | The Poet Tree Club
- Address Bar:** localhost:8888/Webucator/php/InsertUpdateDelete/Solutions/phppoetry.com/poem-edit.php?...
- Navigation Menu:** Home, Poems, Submit Poem, My Account, Contact us
- Header:** The Poet Tree Club, Set your poems free... (with a tree and girl illustration)
- Main Content:**
 - Title:** Edit Poem: Dancing Dogs in Dungarees
 - Metadata:** Submitted on 01/06/2019 at 4:00PM by LimerickMan (Delete button), Approved: 01/06/2019
 - Title*:** Text input field containing "Dancing Dogs in Dungarees"
 - Category:** Dropdown menu with "Funny" selected
 - Poem*:** Text area containing:
A dozen dancing dogs in dungarees
All of the most distinguished pedigrees
Held their heads high by day
By night, they'd bow them and pray
For freedom from filthy frolicking fleas
 - Update Poem:** A large button at the bottom of the form.
- Footer:** Copyright © 2019 The Poet Tree Club. | Log out | Admin | About us

3. The author should be able to update the title, category, and text of the poem.
4. When the poem is updated, the `date_approved` field in the **poems** table should be set to `null`.
5. When the user presses the **Update Poem** button, the page should submit to itself. The changes should take effect and there should be a message letting the user know the poem has been updated and will be reviewed:

Edit Poem: Dancing Dolls in Dreams

Submitted on 01/06/2019 at 4:00PM by LimerickMan [Delete](#)

Pending Approval

Poem updated.

We will review your updated poem soon.

Title*:

Dancing Dolls in Dreams

Category:

Romantic

Poem*:

A dozen dancing dolls in her dreams
Prancing through forests, prairies, and streams
How they did shine so much
It appeared they were touched
By one of the sun's sunny sunbeams

Update Poem

- The user could make additional updates and resubmit.
6. Be sure to write code to catch and log potential exceptions.
 7. Note that only the author of the poem should be able to reach this page. Any other user should be immediately redirected to the home page.
 8. If `poem-id` is not passed in on the URL then the user should be immediately redirected to the home page.
 9. Modify `poem.php` so that the **Edit** and **Delete** links show up only when the logged-in user is the author of the poem. Use `isPoemAuthor($poemId)` to check this. You will also need to modify those links to point to `poem-edit.php` and `poem-delete.php`, respectively, and to pass in the poem id.
 10. Test your solution in a browser by navigating to `http://localhost:8888/Webucator/php/InsertUpdateDelete/Exercises/phppoetry.com/poems.php`, clicking a

poem you wrote, clicking the **Edit** button, making changes, and submitting the form.

Valuable
Copy

Solution: InsertUpdateDelete/Solutions/phppoetry.com/poem-edit.php

```
1.  <?php
2.    $pageTitle = 'Edit Poem';
3.    require 'includes/header.php';
4.
5.    if (!isset( $_REQUEST['poem-id'] )) {
6.        header("Location: index.php");
7.    }
8.
9.    $poemId = $_REQUEST['poem-id'];
10.   if (!isPoemAuthor($poemId)) {
11.       header("Location: index.php");
12.   }
13.
14.   $errors = [];
15.   $f['category'] = $_POST['cat'] ?? 0;
16.   $f['title'] = trim($_POST['title'] ?? '');
17.   $f['poem'] = trim($_POST['poem'] ?? '');
18.
19.   if (!empty($_POST['update'])) {
20.
21.       // Validate Form Entries
22.       if (!$f['title']) {
23.           $errors[] = 'You must enter a poem title.';
24.       }
25.       if (!$f['poem'] || strlen($f['poem']) < 10) {
26.           $errors[] = 'Your poem must be at least 10 characters.';
27.       }
28.
29.       if (!$errors) {
30.           $dateApproved = 'null'; // For now
31.
32.           // Update poem
33.           $qUpdate = "UPDATE poems
34.               SET title = :title,
35.                   poem = :poem,
36.                   category_id = :category_id,
37.                   date_approved = $dateApproved
38.               WHERE poem_id = :poem_id";
39.
40.           try {
41.               $stmtUpdate = $db->prepare($qUpdate);
42.               $stmtUpdate->bindParam(':title', $f['title']);
43.               $stmtUpdate->bindParam(':poem', $f['poem']);
44.               $stmtUpdate->bindParam(':category_id', $f['category']);
```

```

45.         $stmtUpdate->bindParam(':poem_id', $poemId);
46.         $poemUpdated = $stmtUpdate->execute();
47.     } catch (PDOException $e) {
48.         $errors[] = 'Update failed. Please try again.';
49.         logError($e);
50.     }
51. }
52. }
53.
54. $qPoem = "SELECT p.poem_id, p.title, p.poem,
55.         p.date_submitted, p.date_approved,
56.         c.category_id, u.username
57.         FROM poems p
58.         JOIN users u ON u.user_id = p.user_id
59.         JOIN categories c ON c.category_id = p.category_id
60.         WHERE p.poem_id = ?"; // Pass SQL Param
61.
62. $qCategories = "SELECT c.category_id, c.category
63.               FROM categories c
64.               LEFT JOIN poems p ON c.category_id = p.category_id
65.               GROUP BY c.category_id
66.               ORDER BY c.category";
67.
68. try {
69.     $stmtPoem = $db->prepare($qPoem);
70.     $stmtPoem->execute([$poemId]);
71.     $row = $stmtPoem->fetch();
72.     $title = $row['title'];
73.     $poemCatId = $row['category_id'];
74.     $poem = $row['poem'];
75.     $authorUserName = $row['username'];
76.     $dateSubmitted = $row['date_submitted'];
77.     $dateApproved = $row['date_approved'];
78.     $approvedChecked = $dateApproved ? 'checked' : '';
79. } catch (PDOException $e) {
80.     logError($e, true);
81. }
82.
83. try {
84.     $stmtCategories = $db->prepare($qCategories);
85.     $stmtCategories->execute();
86. } catch (PDOException $e) {
87.     logError($e, true);
88. }
89. ?>

```

```

90. <main id="poem-edit">
91.   <h1>
92.     <?= $pageTitle ?>;
93.     <a href="poem.php?poem-id=<?= $poemId ?>"><?= $title ?></a>
94.   </h1>
95.   <div id="submission-status">
96.     Submitted on <?= date('m/d/Y', strtotime($dateSubmitted)) ?>
97.     at <?= date('g:iA', strtotime($dateSubmitted)) ?>
98.     by <?= $authorUserName ?>
99.     <?php
100.        if (isPoemAuthor($poemId)) {
101.            echo "<a href='poem-delete.php?poem-id=$poemId'>Delete</a>";
102.        }
103.    ?>
104. </div>
105. <div id="approval-status">
106. <?php
107.     if (isPoemAuthor($poemId)) {
108.         if ($dateApproved) {
109.             echo 'Approved: ' . date('m/d/Y', strtotime($dateApproved));
110.         } else {
111.             echo 'Pending Approval';
112.         }
113.     }
114. ?>
115. </div>
116. <?php
117.
118.     if (!empty($poemUpdated)) {
119.         echo '<p class="success">Poem updated.</p>';
120.         echo '<p>We will review your updated poem soon.</p>';
121.     }
122.
123.     if ($errors) {
124.         echo '<h3>Please correct the following errors:</h3>';
125.         <ol class="error">';
126.         foreach ($errors as $error) {
127.             echo "<li>$error</li>";
128.         }
129.         echo '</ol>';
130.     }
131. ?>
132. <form method="post" action="poem-edit.php" novalidate>
133.     <input type="hidden" name="poem-id" value="<?= $poemId?>">
134.     <label for="title">Title*:</label>

```



```
135. <input name="title" id="title"
136.   value="<?= $title ?>" required>
137. <label for="cat">Category:</label>
138. <select name="cat" id="cat">
139.   <?php
140.     while ($row = $stmtCategories->fetch()) {
141.       $category = $row['category'];
142.       $rowCatId = $row['category_id'];
143.       $selected = $poemCatId === $rowCatId ? ' selected' : '';
144.       echo "<option value='$rowCatId'$selected>
145.         $category
146.       </option>";
147.     }
148.   ?>
149. </select>
150. <label for="poem">Poem*:</label>
151. <textarea id="poem" name="poem"><?= $poem ?></textarea>
152. <button name="update" value="1" class="wide">Update Poem</button>
153. </form>
154. </main>
155. <?php
156.   require 'includes/footer.php';
157. ?>
```

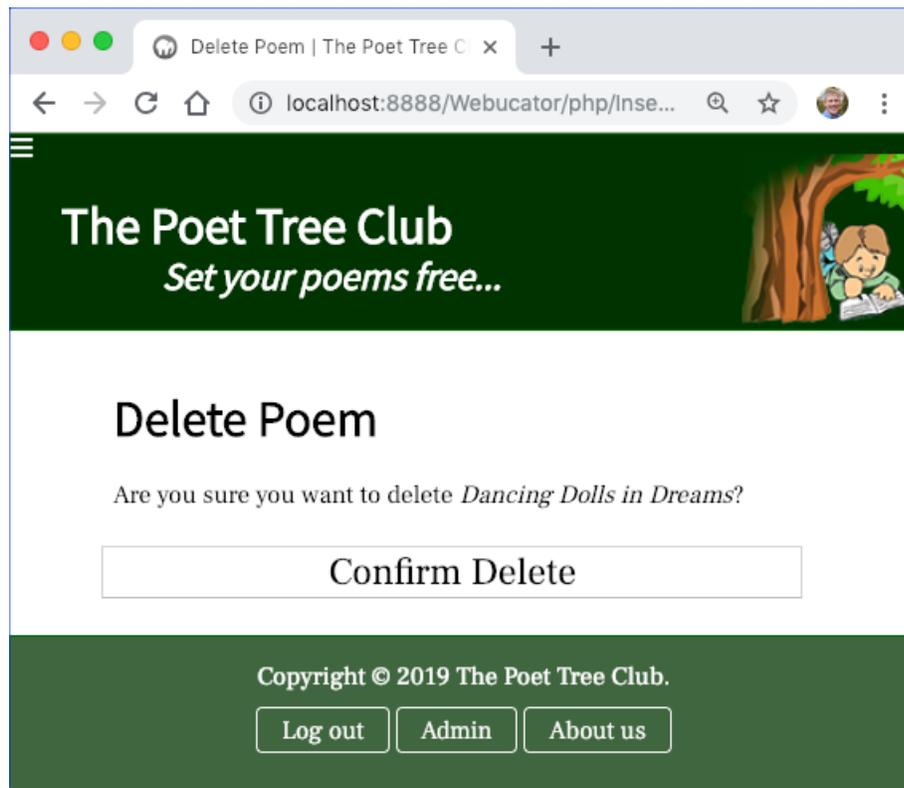
Evaluation
Copy

Exercise 33: Deleting a Poem

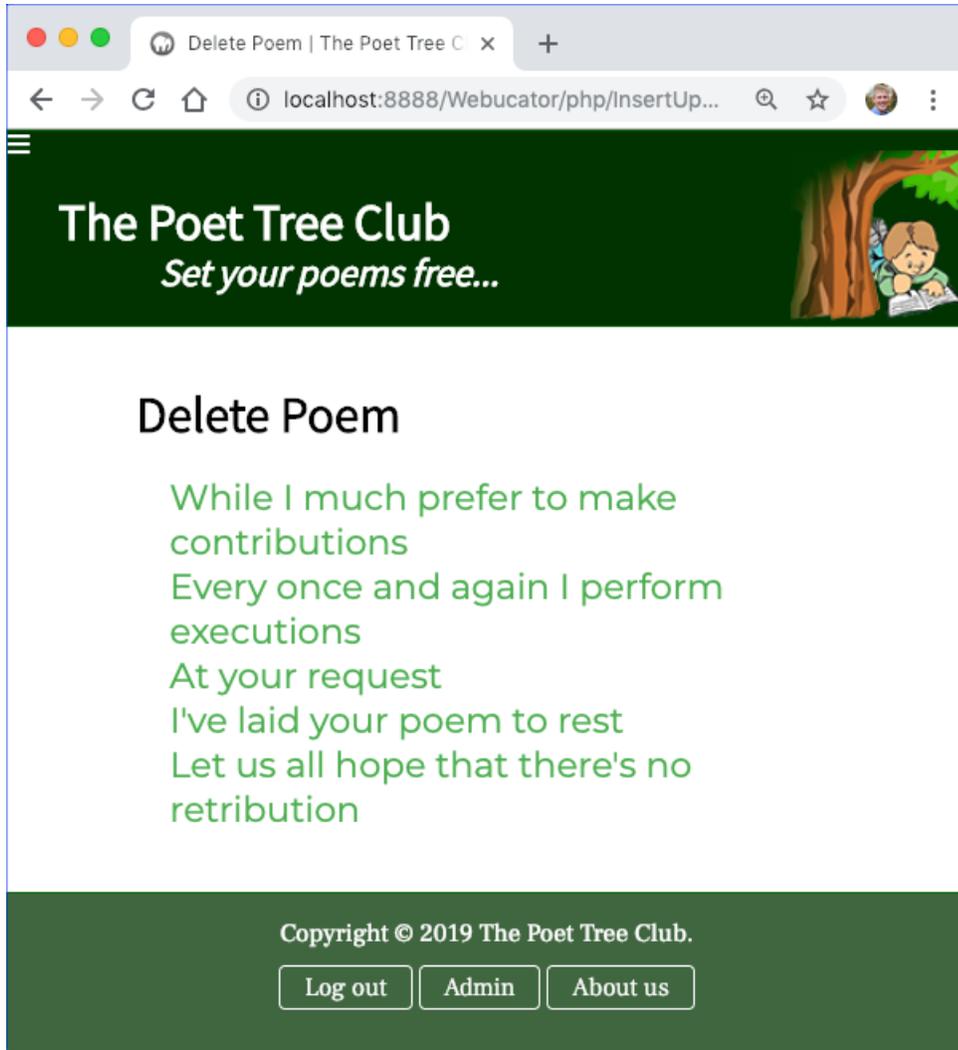
🕒 20 to 30 minutes

In this exercise, you will write a PHP page from scratch for deleting an existing poem.

1. Open a new document and save it as `poem-delete.php` in the `InsertUpdateDelete/Exercises/phppoetry.com` folder.
2. Write code to create a page that provides a **Confirm Delete** button when first visited:



3. When the Confirm Delete button is clicked, the poem should get deleted and a success message should appear:



You can use the `POEM_DELETE_SUCCESS` constant for the success message.

4. Be sure to write code to catch and log potential exceptions.
5. Note that only the author of the poem should be able to reach this page. Any other user should get an access denied.
6. Test your solution in a browser by navigating to `http://localhost:8888/Webucator/php/InsertUpdateDelete/Exercises/phppoetry.com/poems.php`, clicking a poem you wrote, clicking the **Delete** button and then the **Confirm Delete** button. Does the poem get deleted?

Solution: InsertUpdateDelete/Solutions/phppoetry.com/poem-delete.php

```
1.  <?php
2.  $pageTitle = 'Delete Poem';
3.  require 'includes/header.php';
4.
5.  if (!isset( $_REQUEST['poem-id'] )) {
6.      header("Location: index.php");
7.  }
8.
9.  $poemId = $_REQUEST['poem-id'];
10. if (!isPoemAuthor($poemId)) {
11.     header("Location: index.php");
12. }
13.
14. $confirmDelete = isset($_POST['poem-id']);
15. $errors = [];
16.
17. if ($confirmDelete) {
18.     $qDelete = 'DELETE FROM poems WHERE poem_id = ?';
19.     try {
20.         $stmt = $db->prepare($qDelete);
21.         if (!$stmt->execute( [$poemId] )) {
22.             $errorMsg = $stmt->errorInfo()[2];
23.             logError($errorMsg , true);
24.         }
25.         $deleteResult = 1;
26.     } catch (PDOException $e) {
27.         logError($e);
28.         $deleteResult = 0;
29.     }
30. } else {
31.     $qSelect = 'SELECT title FROM poems WHERE poem_id = ?';
32.     try {
33.         $stmt = $db->prepare($qSelect);
34.         if (!$stmt->execute( [$poemId] )) {
35.             $errorMsg = $stmtPoemCount->errorInfo()[2];
36.             logError($errorMsg , true );
37.         } else {
38.             $row = $stmt->fetch();
39.             $poemTitle = $row['title'];
40.         }
41.     } catch (PDOException $e) {
42.         logError( $e->getMessage(), true);
43.     }
44. }
```

```

45. ?>
46. <main id="poem-delete" class="narrow">
47.   <h1><?=$pageTitle ?></h1>
48.
49.   <?php
50.
51.     if (!empty($deleteResult)) {
52.       $deleteResultMsg = nl2br(POEM_DELETE_SUCCESS);
53.       $cls = 'success';
54.     } elseif (isset($deleteResult)) {
55.       logError("Failed to delete poem id $poemId using: $qDelete");
56.       $deleteResultMsg = nl2br(POEM_DELETE_FAIL);
57.       $cls = 'error';
58.     }
59.
60.     if (isset( $deleteResultMsg )) {
61.       // Output delete result
62.       ?>
63.       <article class="poem <?=$cls ?>">
64.         <?=$deleteResultMsg ?>
65.       </article>
66.     <?php
67.     } else {
68.       // Output delete form
69.       ?>
70.       <form method="post" action="poem-delete.php">
71.         <p>Are you sure you want to delete <em><?=$poemTitle ?></em>?</p>
72.         <button name="poem-id" value="<?=$poemId?>" class="wide">
73.           Confirm Delete
74.         </button>
75.       </form>
76.     <?php
77.     }
78.     ?>
79. </main>
80. <?php
81.   require 'includes/footer.php';
82. ?>

```

Conclusion

In this lesson, you have had the chance to practice the PHP skills you have learned thus far.

LESSON 11

Uploading Files

Topics Covered

- Image uploads.

Introduction

In some cases, you may need to collect files from website users. You can do this using the `file` input type. But you'll want to be careful not to just allow any type or any size file to be uploaded.



11.1. Enabling File Info Functions on Windows

Mac Users: move on...

If you are on a Mac, you can move on to the next activity. If you're on Windows, read on...

In this lesson, we make use of the built-in PHP function `mime_content_type()`. For this function to work on Windows, you must enable the **php_fileinfo.dll** extension, which is already included with MAMP, but is not enabled by default.

1. Open your `php.ini` file in your editor. If you don't remember where the `php.ini` file is, visit <http://localhost:8888/Webucator/php/PhpBasics/Demos/find-php.ini.php> in your browser.
2. Add the following line of code in the section on extensions:

```
extension=php_fileinfo.dll
```

It should look something like this:

```
669 extension=php_mbstring.dll
670 extension=php_exif.dll
671
672 ; Added to enable mime_content_type()
673 extension=php_fileinfo.dll
674
675 extension=php_mysql.dll
676 extension=php_mysqli.dll
```

3. Restart the MAMP Servers so that your new php.ini settings take effect.

Now you should be all set to continue with the lesson.



11.2. Uploading Images via an HTML Form

To upload files via an HTML form, the form tag's method must be set to "post" and the enctype attribute must be set to "multipart/form-data" as shown below:

```
<form method="post" enctype="multipart/form-data">
```

The following example demonstrates how to safely allow the user to upload an image to the server:

mime_content_type() Error

If you get a Call to undefined function mime_content_type() error while running the following demo, make sure that the php.ini file includes these lines:

```
; Added to enable mime_content_type()
extension=php_fileinfo.dll
```

The second line should **not** begin with a semi-colon, which is used to denote a comment in the php.ini file.

Demo 11.1: UploadingFiles/Demos/file-upload.php

```
-----Lines 1 through 11 Omitted-----
12. <?php
13.     if (empty($_POST['submitted'])) {
14.     ?>
15.     <h1>Upload Form</h1>
16.     <form method="post" action="file-upload.php"
17.         enctype="multipart/form-data">
18.         <label for="image-name">Image Name:</label>
19.         <input name="image-name" id="image-name">
20.         <label for="image">Image:</label>
21.         <input type="file" name="image" id="image" accept=".png,.jpg">
22.         <button name="submitted" value="1" class="wide">Submit</button>
23.     </form>
24. <?php
25. } else {
26.     //process the form
27.     $errors = [];
28.     if (!$_POST['image-name']) {
29.         $error = 'Image name is required.';
30.     } else {
31.         $imgFileName = $_FILES['image']['name'];
32.         $imgTmpLocation = $_FILES['image']['tmp_name'];
33.         $fileSize = $_FILES['image']['size'];
34.         $fileError = $_FILES['image']['error'];
35.
36.         $sxt = strtolower(end(explode('.', $imgFileName)));
37.         $time = time();
38.         $imgNewName=$_POST['image-name'] . '_' . $time . '.' . $sxt;
39.         $fileSavePath = 'uploaded-images/' . $imgNewName;
40.
41.         $allowedTypes = ['image/png','image/jpeg'];
42.         $mimeType = mime_content_type($imgTmpLocation);
43.
44.         if (!in_array($mimeType, $allowedTypes)) {
45.             $error = 'You uploaded a file of type ' . $mimeType .
46.                 ' Only png and jpg files are allowed.';
47.         } elseif ($fileError === UPLOAD_ERR_INI_SIZE) {
48.             $error = "The file is too big.";
49.         } elseif ($fileError) {
50.             $error = "The file could not be uploaded.";
51.         } elseif (!move_uploaded_file($imgTmpLocation, $fileSavePath)) {
52.             $error = "Could not save file.";
53.         }
54.     }
```

```
55.
56.     if ($error) {
57.         echo "<h2>Error</h2>
58.             <p>$error</p>";
59.     } else {
60.         echo "<h2>Success</h2>
61.             <p>Your image has been uploaded.</p>";
62.     }
63. }
64. ?>
```

-----Lines 65 through 67 Omitted-----

Code Explanation

1. The first thing to notice about this page is that it submits to itself. It uses `$_POST['submitted']` to know whether or not the form has been submitted. The first time the page is loaded, it will show the form. When the form is submitted, it will attempt to upload and save the uploaded picture.
2. The form includes an input field of type `file` that is used to browse for the file to upload. Notice the `accept` attribute. This is used to tell the browser what file extensions are accessible. This can take a comma-delimited list (e.g., `.png, .jpg`) to specify multiple acceptable extensions.
3. The form also includes an `image-name` field, which is combined with the value of `time()` and the extension of the uploaded file to name the uploaded file:

```
$ext = strtolower(end(explode('.', $imgFileName)));
$time = time();
$imgNewName=$_POST['image-name'] . '_' . $time . '.' . $ext;
$fileSavePath = 'uploaded-images/' . $imgNewName;
```

- The built-in `end()` function gets the last element of an array.
- Using the value of `time()` as part of the file name ensures that each file name will be unique.

4. We then get the mime type of the file using the built-in `mime_content_type()` function:

```
$allowedTypes = ['image/png','image/jpeg'];
$mimeType = mime_content_type($imgTmpLocation);

if (!in_array($mimeType, $allowedTypes)) {
    $error = 'You uploaded a file of type ' . $mimeType .
        ' Only png and jpg files are allowed.';
}
```

This is a secure way of checking that the file is the type of file you're expecting. You cannot rely on the extension of the file, as that can be modified.

5. Uploaded files are stored in the `$_FILES` superglobal array. Because our file input field is named "image", this file is stored by PHP as `$_FILES['image']`. The file includes the following keys:

- `tmp_name` - The temporary name of the uploaded file, which we will use when we move the file to the `uploaded-images` directory.
- `size` - The size of the uploaded file.
- `type` - The type of the uploaded file as reported by the browser. As this can easily be faked by hackers, we won't use this.
- `error` - Any error that prevented the file from being uploaded. You can check for specific errors using the constants documented at <https://www.php.net/manual/features.file-upload.errors.php>. Our code specifically checks for `UPLOAD_ERR_INI_SIZE`, so that we can give a relevant error message and then just checks for any other error to report a generic error message.

```
elseif ($fileError === UPLOAD_ERR_INI_SIZE) {
    $error = "The file is too big.";
} elseif ($fileError) {
    $error = "The file could not be uploaded.";
}
```

6. We then use the built-in `move_uploaded_file()` function to move the uploaded file to the `uploaded-images` folder:

```
elseif (!move_uploaded_file($imgTmpLocation, $fileSavePath)) {  
    $error = "Could not save file."  
}
```

7. Finally, we either report the error if there is one or success if there is not.

post_max_size and MAX_FILE_SIZE

PHP provides two related directives that can affect file uploads:

1. The `php.ini` file includes a related directive: `post_max_size`. This sets the maximum total size of all data, including any uploaded files, that can be sent to the server in a single post. If a post is larger than the value of `post_max_size`, no error will occur, but the `$_FILES` and `$_POST` arrays will both be emptied, so all the posted data will be lost. For more information on `post_max_size`, see <https://www.php.net/manual/ini.core.php#ini.post-max-size>.
2. PHP provides a way of setting the maximum file size allowed on a form-by-form basis through a hidden input field with the name "MAX_FILE_SIZE" and a value in bytes. For more information on `MAX_FILE_SIZE`, see <https://www.php.net/manual/features.file-upload.post-method.php>.



11.3. Resizing Images

There are three functions built in to PHP that take a path to an image and from it create an *image identifier*, which can then be used to manipulate and save over or create a new copy of the original image:

1. `imagecreatefromjpeg(filename)` - creates an image identifier for a JPEG image.
2. `imagecreatefrompng(filename)` - creates an image identifier for a PNG image.

3. `imagecreatefromgif(filename)` - creates an image identifier for a GIF image.

The following demo shows how to use the first two of these to take an image, resize it with the built-in `imagescale()` function, and then save it as a new image:

Demo 11.2: UploadingFiles/Demos/resize-image.php

```
-----Lines 1 through 11 Omitted-----
12. <h1>Resize Image</h1>
13. 
14. 
15. <form method="post" action="resize-image.php">
16.     <label for="width" class="inline">Width:</label>
17.     <input type="number" min="50" max="500" step="25"
18.         id="width" name="width" value="200">
19.     <label for="height" class="inline">Height:</label>
20.     <input type="number" min="50" max="500" step="25"
21.         id="height" name="height" value="300">
22.     <button name="resizing" value="1">Resize</button>
23. </form>
24. <?php
25.     if (isset($_POST['resizing'])) {
26.         $w = $_POST['width'];
27.         $h = $_POST['height'];
28.         echo '<h1>Resized Images</h1>';
29.         $poohOrigPath = 'images/pooh.jpg';
30.         $poohNewPath = "images/pooh-$w-$h.jpg";
31.         $bulbOrigPath = 'images/bulb.png';
32.         $bulbNewPath = "images/bulb-$w-$h.png";
33.
34.         if ($img = imagecreatefromjpeg($poohOrigPath)) {
35.             $newImg = imagescale($img, $w, $h);
36.             imagejpeg($newImg, $poohNewPath);
37.             imagedestroy($img);
38.             echo "<img src='$poohNewPath' alt='Winnie Resized'>";
39.         } else {
40.             echo 'imagecreatefromjpeg() failed: ' . $poohNewPath;
41.         }
42.
43.         if ($img = imagecreatefrompng($bulbOrigPath)) {
44.             $newImg = imagescale($img, $w, $h);
45.             imagesavealpha($newImg, true);
46.             imagepng($newImg, $bulbNewPath);
47.             imagedestroy($img);
48.             echo "<img src='$bulbNewPath' alt='Bulb Resized'>";
49.         } else {
50.             echo 'imagecreatefrompng() failed: ' . $bulbNewPath;
51.         }
52.     }
53.     ?>
```

Code Explanation

This page contains a form for submitting new height and width values. When the form is submitted it resizes `images/pooh.jpg`¹⁸ and `images/bulb.png`¹⁹ to the specified dimensions and creates new images with file names formatted as `images/pooh-$w-$h.jpg` and `images/bulb-$w-$h.png`.

Things to note:

1. We initially create the images with `imagecreatefromjpg()` and `imagecreatefrompng()`.
2. We resize both types of images using `imagescale($img, $w, $h)`.
3. For the PNG image, we use `imagesavealpha($newImg, true)` to keep the transparency. PHP includes many other built-in functions for working with color and transparency.
4. We save the new images with `imagejpeg($newImg, $poohNewPath)` and `imagepng($newImg, $bulbNewPath)`.
5. Finally, to free memory, we destroy the image identifiers with `imagedestroy($img)`. This does not delete the original image files.

18. The https://commons.wikimedia.org/wiki/File:R._John_Wright_Winnie_the_Pooh_Bear.jpg image is used under the terms of GNU Free Documentation License, version 1.2 (https://commons.wikimedia.org/wiki/Commons:GNU_Free_Documentation_License,_version_1.2).

19. The https://commons.wikimedia.org/wiki/File:Light_bulb_icon_tips.svg image is used under the terms of GNU Free Documentation License, version 1.2 (https://commons.wikimedia.org/wiki/Commons:GNU_Free_Documentation_License,_version_1.2).

Exercise 34: Uploading a Profile Picture

 45 to 60 minutes

In this exercise, you will write a new `uploadProfilePic()` function in `utilities.php`.

1. Open `utilities.php` from the `UploadingFiles/Exercises/phppoetry.com/includes` folder.
2. Write a new `uploadProfilePic()` function, which should do the following:
 - A. Take one parameter: `$img`, which takes an uploaded picture (e.g., `$_FILES['image']`).
 - B. Get the temporary location and save it as `$imgTmpLocation`.
 - C. Check to make sure the mime type of the image is either 'image/png' or 'image/jpeg'. You can use the `in_array()` function for this. If it is not one of these mime types, the function should return `false`.
 - D. If the passed-in image contains an error, log the error and return `false`.
 - E. Create a name and path for the new image as `profile-$time.$ext` and `../../../../static/images/profile-pics/profile-$time.$ext`, where `$ext` is "png" or "jpg" and `$time` is the current value of `time()`.
 - F. Try to create an image from the image at `$imgTmpLocation`. You will have to account for the type of image passed in (PNG or JPEG).
 - G. Return the new image file name.
3. To test your code, navigate to `http://localhost:8888/Webucator/php/UploadingFiles/Exercises/phppoetry.com/my-account.php`, which has been created for you and upload a profile picture. Note that you will need to be logged in.

Solution: UploadingFiles/Solutions/phppoetry.com/includes/utilities.php

```
-----Lines 1 through 121 Omitted-----
122.     function uploadProfilePic($img) {
123.         // Get the temporary location
124.         $imgTmpLocation = $img['tmp_name'];
125.
126.         // Check mime type and file error
127.         $fileError = $img['error'];
128.         $allowedTypes = ['image/png', 'image/jpeg'];
129.         $mimeType = mime_content_type($imgTmpLocation);
130.
131.         if (!in_array($mimeType, $allowedTypes)) {
132.             return false;
133.         } elseif ($fileError) {
134.             logError($fileError);
135.             return false;
136.         }
137.
138.         // Create new image name and path
139.         $ext = ($mimeType === 'image/png') ? 'png' : 'jpg';
140.         $time = time();
141.         $newImgFileName = "profile-$time.$ext";
142.         $fileSavePath = '../static/images/profile-pics/' .
143.             $newImgFileName;
144.
145.         // Try to move the passed-in image to the new path
146.         try {
147.             if ($mimeType === 'image/png') {
148.                 if ($img = imagecreatefrompng($imgTmpLocation)) {
149.                     $newImg = imagescale($img, 200, 200);
150.                     imagesavealpha($newImg, true);
151.                     imagepng($newImg, $fileSavePath);
152.                     imagedestroy($img);
153.                 } else {
154.                     logError('imagecreatefrompng() failed: ' . $fileSavePath);
155.                     return false;
156.                 }
157.             } else { // jpg
158.                 if ($img = imagecreatefromjpeg($imgTmpLocation)) {
159.                     $newImg = imagescale($img, 200, 200);
160.                     imagejpeg($newImg, $fileSavePath);
161.                     imagedestroy($img);
162.                 } else {
163.                     logError('imagecreatefromjpeg() failed: ' . $fileSavePath);
164.                     return false;

```

```
165.     }
166.     }
167.     return $newImgFileName;
168. } catch (PDOException $e) {
169.     logError($e);
170.     return false;
171. }
172. }
173. ?>
```

Be sure to review `my-account.php` in your editor to make sure you understand all the code. If there are functions that are new to you (e.g., `is_file()`), look them up on php.net.

Conclusion

In this lesson, you have learned to upload images, to create copies of images, and to resize images.

LESSON 12

Admin Site

Topics Covered

- Creating an admin site.

Introduction

This lesson is made up of several exercises, in which you create the admin site for php-poetry.com. Note that this lesson is not a required part of the course, but it gives you an opportunity to practice the PHP skills you have learned. At this point, if you don't have time to work through the exercises, you may just read through them and consider how you would go about creating the admin site.

Evaluation
Copy

Exercise 35: Adding the Admin Pages

 90 to 120 minutes

In this exercise, you will begin the admin site for `phppoetry.com` by adding the admin pages and modifying the header and footer includes so that the pages display correctly and the navigation links work. You will be doing your work in `AdminSite/Exercises/phppoetry.com`.

1. Create a new folder called `admin`.
2. Within the `admin` folder, create another folder called `includes`.
3. Within the `includes` folder, create a single file: `nav.php` and add the following code to it:

```
<nav id="admin-nav">
  <ul>
    <li><a href="index.php">Home</a></li>
    <li><a href="users.php">Users</a></li>
    <li><a href="poems.php">Poems</a></li>
  </ul>
</nav>
```

4. You will be creating the following files in `admin`:
 - A. `index.php` - The admin home page.
 - B. `poems.php` - The listing of all poems, approved and unapproved.
 - C. `users.php` - A listing of all users.

Go ahead and create each of those files.

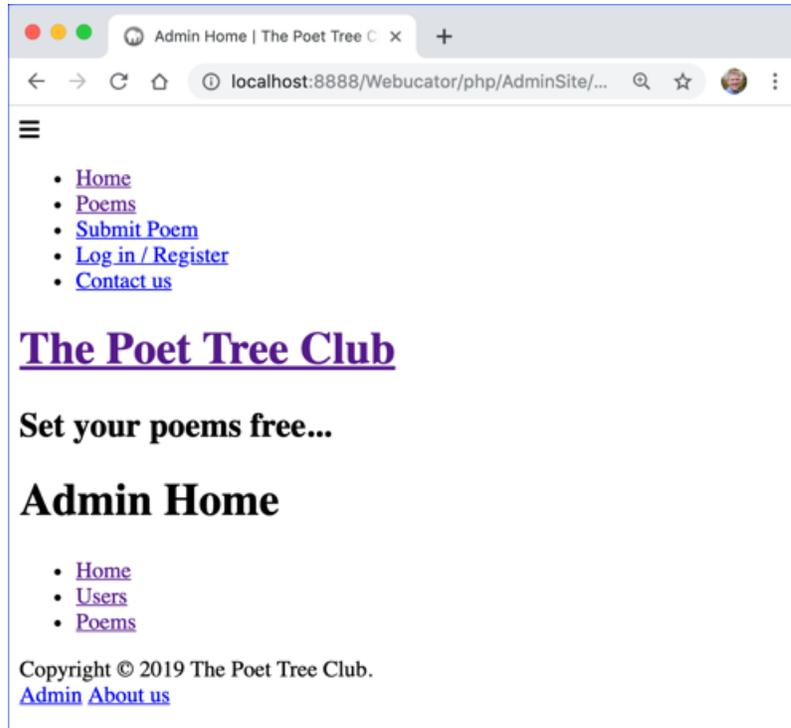
5. Enter the following code in `index.php`:

```
<?php
    $pageTitle = 'Admin Home';
    require '../includes/header.php';
?>
<main id="page-id" class="admin">
    <h1><?= $pageTitle ?></h1>
    <?php require 'includes/nav.php'; ?>

</main>
<?php
    require '../includes/footer.php';
?>
```

Evaluation
Copy

6. Enter the same code in `poems.php`, and `users.php`, but change the titles to **Admin Poems** and **Admin Users**, respectively.
7. In all three files, change the `main` element's `id` values from "page-id" to "admin", "poems", and "users".
8. You should now be able to navigate to `http://localhost:8888/Webucator/php/AdminSite/Exercises/phppoetry.com/admin/index.php`. The page will look something like this:



9. The page is unstyled because the paths to the CSS pages are incorrect. And those are incorrect, because the admin pages are in a directory below the other pages on the site. View the source of the `admin/index.php` and you will see that the `href` values of the link tags begin with `../../../../static/`. For the admin pages, they should need to go up an additional directory before going into `static`, so they should begin with `../../../../../static/`. We have to let the header include know this:

- A. Add the following line of code to each of the three main admin pages:

```
$pathStart = '../';
```

This code should come before requiring `header.php`.

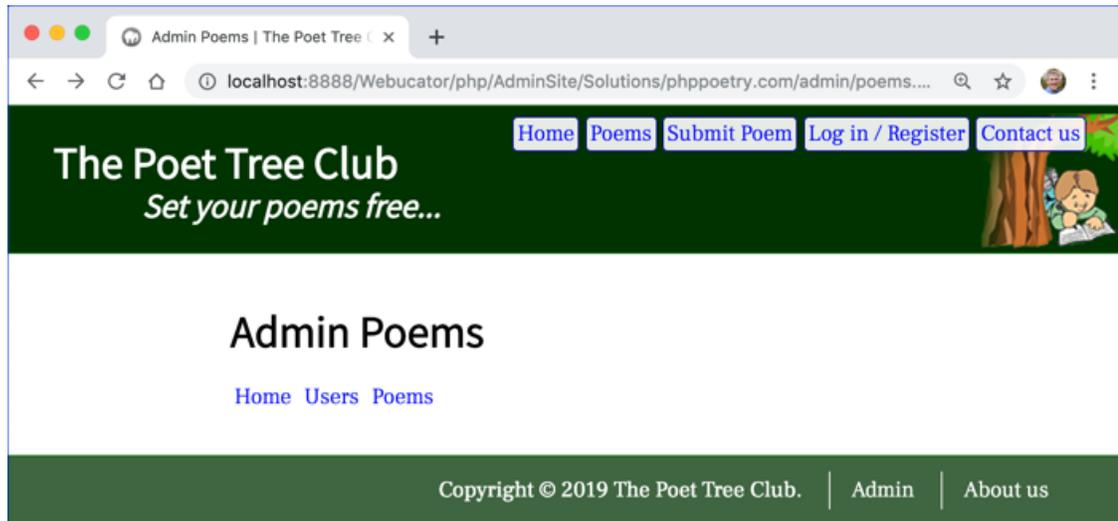
- B. Open `AdminSite/Exercises/phppoetry.com/includes/header.php` in your editor and add the following line of code somewhere in the initial PHP block:

```
$pathStart = $pathStart ?? '../';
```

- C. Now prepend `$pathStart` to the paths to `normalize.css`, `styles.css`, and `scripts.js`. For example:

```
<?= $pathStart ?>../../../../static/styles/normalize.css
```

- D. Do you understand what you done in these last three steps? Go back to the admin home page. It should now be styled like this:



- E. The links in the main navigation in the upper-right of the page will not work on the admin pages. Fix those using the same method that you fixed the CSS and JavaScript links above.
- F. The links in the footer will also not work on the admin pages. Fix those as well.
- G. Test all your header and footer links on both the non-admin and admin pages. Fix anything that doesn't work.

Exercise 36: Creating the isAdmin() Function

🕒 45 to 75 minutes

You will need to be able to identify whether a logged-in user is an administrator or not. To do so, you will create an `isAdmin()` function in the `utilities.php` include.

1. Open `AdminSite/Exercises/phppoetry.com/includes/utilities.php` in your editor. Add a function with the following signature:

```
bool isAdmin(int $userId)
```

This function should return `true` if the `is_admin` field in the **users** table has a value of 1. Otherwise, it should return `false`.

2. Open `AdminSite/Exercises/phppoetry.com/includes/footer.php` in your editor. Add code so that the **Admin** link only shows up if the user is logged in and is an admin.
3. Add code to all three main admin pages that redirects any non-admin user to the home page.
4. Test your code:
 - A. Log in to the site as **LimerickMan**. The **Admin** link should show up in the footer.
 - **Username:** LimerickMan
 - **Password:** i love 2 eat BaseBalls!
 - B. Log out. The **Admin** link should **not** show up in the footer.
 - C. Log in as **HugHerHeart** who is not an admin. The **Admin** link should **not** show up in the footer.
 - **Username:** HugHerHeart
 - **Password:** i love 2 eat BaseBalls!
5. While still logged in as **HugHerHeart**, navigate to `http://localhost:8888/Webucator/php/AdminSite/Exercises/phppoetry.com/admin/index.php`. You should be redirected to the main home page.

Exercise 37: Completing the Admin Home Page

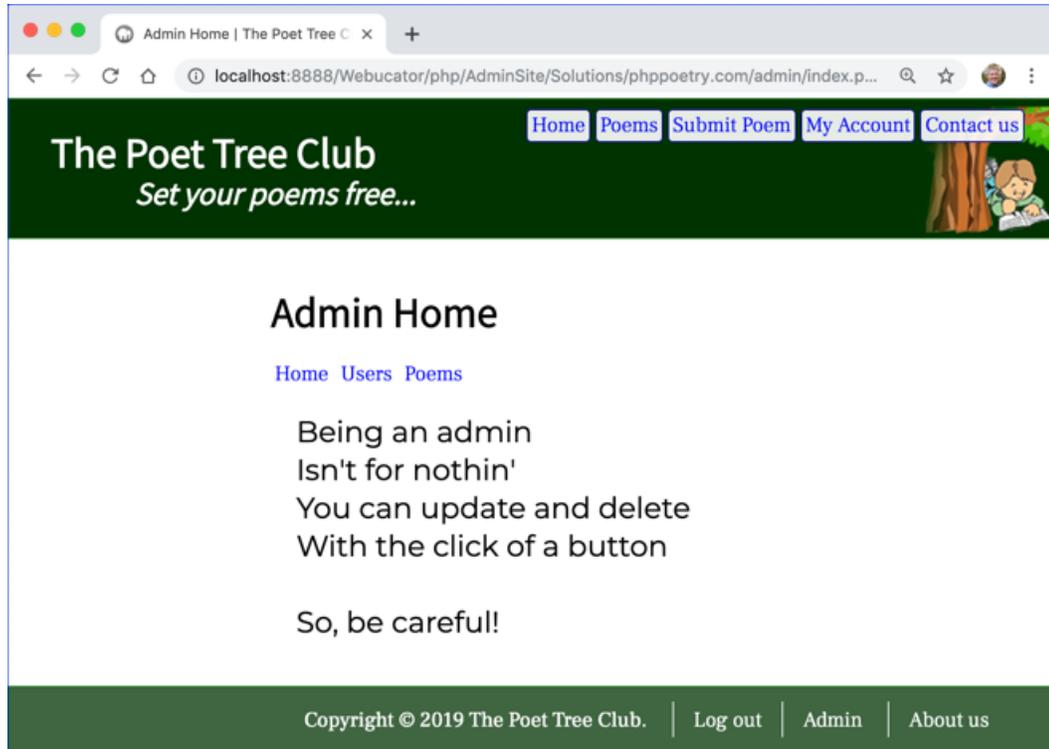
 10 to 15 minutes

The admin home page is very simple.

1. Open `AdminSite/Exercises/phppoetry.com/admin/index.php` in your editor.
2. Add a narrow class to the main element.
3. Add the following article element below where `nav.php` is included:

```
<article class="poem">
  Being an admin<br>
  Isn't for nothin'<br>
  You can update and delete<br>
  With the click of a button<br><br>
  So, be careful!
</article>
```

4. Log in as **LimerickMan** and go to the admin home page. It should look like this:



Exercise 38: Completing the Admin Poems Page

 15 to 20 minutes

The admin poems page will be similar to the regular poems page. It will include a sortable table of poems with pagination and a form for filtering the poems. The poem titles will link to the poem page.

The major differences are:

1. The admin poems table will include poems that have not yet been approved. Table rows containing poems that have not been approved will have the “pending-approval” class (e.g., `<tr class='pending-approval'>`). All other rows will have the “normal” class.
2. The default sorting should be `date-approved asc` so that the poems that have not been approved appear first.
3. The table should show five rows per page.

When first loaded, the page should look something like this:

The Poet Tree Club
Set your poems free...

Home Poems Submit Poem My Account Contact us

Admin Poems

[Home](#) [Users](#) [Poems](#)

Total Poems: 8

Poem	Category	Author	Submitted	Approved
The Aristocratic Apple	Funny	LimerickMan	12/23/2018	N/A
Banana Duet	Funny	LimerickMan	02/03/2018	02/03/2018
Harry's Torment	Serious	Dawnable	11/09/2018	11/12/2018
A Bloody Good Goodbye	Romantic	HugHerHeart	11/24/2018	11/29/2018
My Innocent Tulip	Romantic	HugHerHeart	12/02/2018	12/06/2018

[Previous](#) [Next](#)

Filtering

Category:

Author:

Copyright © 2019 The Poet Tree Club. | [Log out](#) | [Admin](#) | [About us](#)

1. Open AdminSite/Exercises/phppoetry.com/admin/poems.php in your editor.
2. Create the page described above using AdminSite/Exercises/phppoetry.com/poems.php as a guide.
3. Test your page by visiting http://localhost:8888/Webucator/php/AdminSite/Exercises/phppoetry.com/admin/poems.php while logged in as LimerickMan.

Exercise 39: Approving, Editing, and Deleting Poems

 45 to 60 minutes

In this exercise, you will modify `poem.php`, `poem-edit.php`, and `poem-delete.php` to make it possible for admins to approve, edit, and delete other users' poems. .

1. Open `AdminSite/Exercises/phppoetry.com/poem.php` in your editor. Currently, the links to `poem-edit.php` and `poem-delete.php` only show up for the logged-in user. Change the code so that they show up for admin users as well.
2. Open `AdminSite/Exercises/phppoetry.com/poem-delete.php` in your editor. Currently, any user who is not the author of the poem will be redirected to the home page. Allow admin users to stay on this page as well.
3. Open `AdminSite/Exercises/phppoetry.com/poem-edit.php` in your editor.
 - A. Currently, any user who is not the author of the poem will be redirected to the home page. Allow admin users to stay on this page as well.
 - B. Add a checkbox to the form with the name "approve". If, and only if, the poem has been approved already, the checkbox should be checked. This checkbox should only show up in the form for admins.
 - C. Currently, only the poem author can see if and when the poem was approved. Modify this so that admins also see this information.
 - D. Currently, when a user modifies a poem, we output "We will review your updated poem soon." Admins don't need to see this sentence. Hide it from them.
 - E. Currently, `$dateApproved` is always set to 'null'. Modify the code so that it is set to 'now()' if the "approve" checkbox was checked and the current user is an admin.
4. Test your page by visiting `http://localhost:8888/Webucator/php/AdminSite/Exercises/phppoetry.com/admin/poems.php` while logged in as Dawnable.
 - A. Submit a new poem.
 - B. Log out and log back in as LimerickMan.
 - C. Approve the poem that Dawnable just submitted.
 - D. Unapprove the poem.

E. Delete the poem.

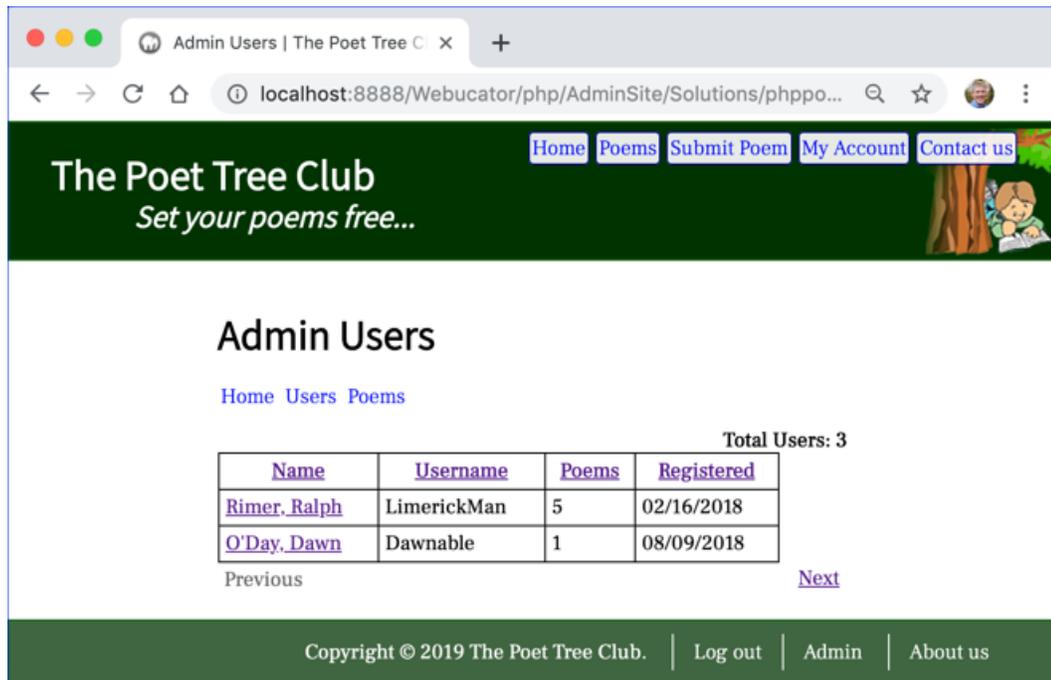
*Evaluation
Copy*

Exercise 40: Completing the Admin Users Page

60 to 90 minutes

In this exercise, you will create the admin users page, which is similar to the admin poems page, but it is simpler as there is not filtering.

1. Open `AdminSite/Exercises/phppoetry.com/admin/users.php` in your editor.
2. Create a page that looks like this:



The screenshot shows a web browser window with the URL `localhost:8888/Webucator/php/AdminSite/Solutions/phppoetry.com/admin/users.php`. The page title is "Admin Users | The Poet Tree Club". The header is green with the text "The Poet Tree Club" and "Set your poems free...". There are navigation buttons for "Home", "Poems", "Submit Poem", "My Account", and "Contact us". The main content area is titled "Admin Users" and has links for "Home", "Users", and "Poems". A table displays the following data:

Total Users: 3			
Name	Username	Poems	Registered
Rimer, Ralph	LimerickMan	5	02/16/2018
O'Day, Dawn	Dawnable	1	08/09/2018

There are "Previous" and "Next" links below the table. The footer contains "Copyright © 2019 The Poet Tree Club." and links for "Log out", "Admin", and "About us".

3. The query to get the users is:

```
SELECT u.user_id, u.username, u.first_name, u.last_name,  
       u.email, u.date_registered, u.registration_confirmed,  
       COUNT(p.poem_id) AS num_poems  
FROM users u  
     LEFT JOIN poems p ON u.user_id = p.user_id  
GROUP BY u.user_id, u.username, u.first_name, u.last_name,  
         u.email, u.date_registered, u.registration_confirmed  
ORDER BY $order $dir  
LIMIT $offset, $rowsToShow
```

We use a left join on **poems** so as not to exclude users who haven't written poems.

4. Because there are only three users in the database, only show two users at a time.
5. The sortable fields should be last_name, username, num_poems, and date_registered.
6. The user's names should link to my-account.php and pass in the user's user-id on the query string (e.g., my-account.php?user-id=1).
7. Test your page by visiting <http://localhost:8888/Webucator/php/AdminSite/Exercises/phppoetry.com/admin/users.php> while logged in as LimerickMan.

Exercise 41: Make the My Account Page Spoofable

 20 to 30 minutes

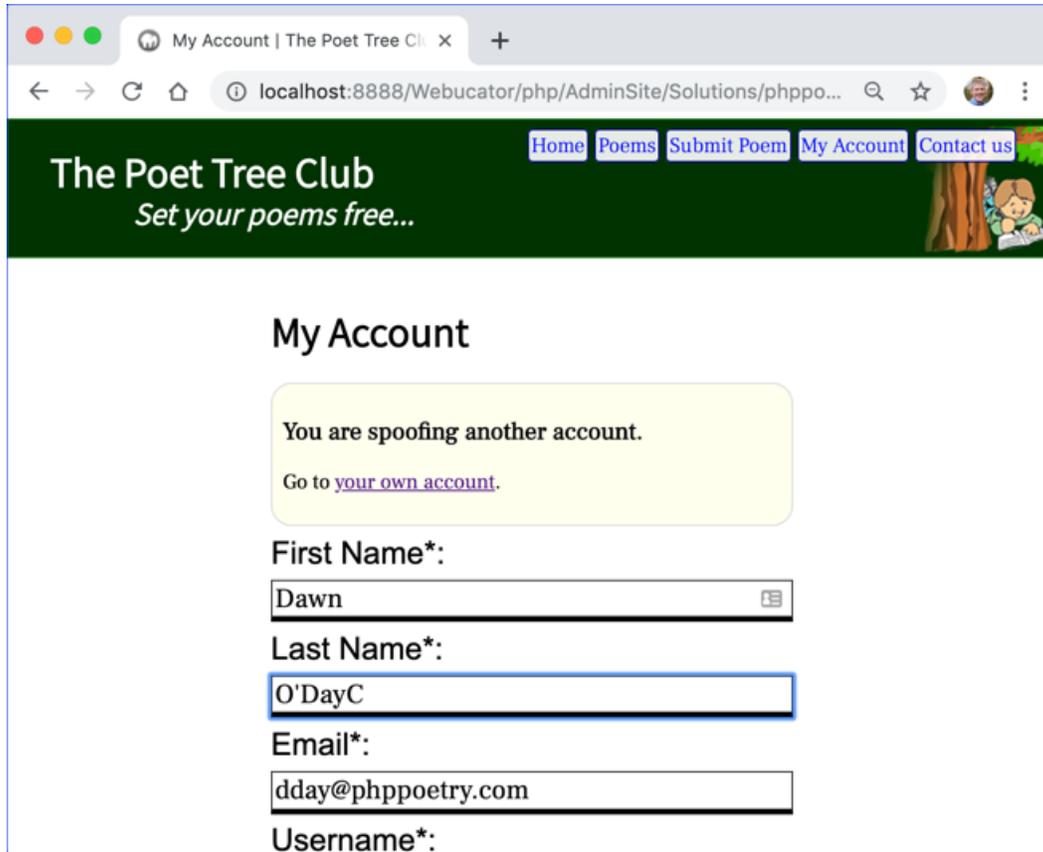
In this exercise, we will make it possible for admin users to see other user's **My Account** pages.

1. Open `AdminSite/Exercises/phppoetry.com/my-account.php` in your editor.
2. After the code that makes sure the user is authenticated, create a new `$userId` variable. If a `user-id` parameter is passed in over the query string or via a form (i.e., if it exists in the `$_REQUEST` superglobal array) then `$userId` should get the value of that parameter. Otherwise, it should get `$currentUserId`.
3. On the rest of the page, change all instances of `$currentUserId` to `$userId`.
4. Immediately after the `<h1>` tag, add the following code, which will output a message reminding the admin user that they are spoofing an account:

```
if ($userId !== $currentUserId) {  
    echo '<div class="callout">';  
    echo '<h3>You are spoofing another account.</h3>';  
    echo '<p>Go to <a href="my-account.php">your own account</a>.</p>';  
    echo '</div>';  
}
```

5. Add a hidden “user-id” field to the form that holds `$userId`.
6. Test your page by visiting `http://localhost:8888/Webucator/php/AdminSite/Exercises/phppoetry.com/admin/users.php` while logged in as LimerickMan. Then click a user and see if you can modify their account.

When spoofing another account, the page should look something like this:



Conclusion

The completed admin site is in AdminSite/Solutions/phppoetry.com. The files that have been modified are:

1. All the files in the admin folder.
2. includes/utilities.php
3. poem.php
4. poem-edit.php
5. poem-delete.php
6. my-account.php