INTRO TO PHP 5 & MYSQL

STEP BY STEP TRAINING

Learn by doing step by step exercises.
Includes downloadable class files that work on Mac & PC.

EDITION 1.3

noble desktop

Adobe AUTHORIZED Training Center

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Published by:
Noble Desktop LLC
594 Broadway, Suite 1202
New York, NY 10012
www.nobledesktop.com

Copyright © 2010–2016 Noble Desktop LLC
Publish Date: 11-03-2016

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopy, recording, or otherwise without express written permission from the publisher. For information on reprint rights, please contact **educator-in-chief@nobledesktop.com**

The publisher makes no representations or warranties with respect to the accuracy or completeness of the contents of this work, and specifically disclaims any warranties. Noble Desktop shall not have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it. Further, readers should be aware that software updates can make some of the instructions obsolete, and that websites listed in this work may have changed or disappeared since publication.

Adobe, the Adobe Logo, Creative Cloud, InDesign, Illustrator, Photoshop, and Dreamweaver are trademarks of Adobe Systems Incorporated.

Apple and macOS are trademarks of Apple Inc. registered in the U.S. and other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the U.S. and other countries. All other trademarks are the property of their respective owners.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Table of Contents

SETUP & INTRODUCTION

Downloading the Class Files 9

Before You Begin 11
Topics: Recommended software
Installing packages into Sublime Text to boost functionality

Mac: Setting Up the Local Server 15
Topics: Differences between MAMP & MAMP PRO
Installing MAMP PRO
Installing MAMP

Windows: Setting Up the Local Server 17
Topics: Windows: installing XAMPP

INFO & EXERCISES

SECTION 1

Exercise 1A: Setting Up: Do This Before Other Exercises! 19
Topics: Setting up your class files
Setting MAMP PRO to display errors

Exercise 1B: Basic PHP Syntax 21
Topics: Echo, strings, & variables
Single quotes vs. double quotes
Escaping characters
Heredoc
Concatenation
Comments

Exercise 1C: Set Up Browser Preview in Sublime Text or Dreamweaver 27
Topics: Setting up Sublime Text
Setting up Dreamweaver

Exercise 1D: Working with Numbers 31
Topics: Arithmetic operators
Assignment operators
Table of arithmetic operators
Table of assignment operators

Table of Contents

Exercise 1E: Conditionals 35

Topics: If/Else statements
Elseif statements
Switch statements
Comparison operators
Logical operators
The difference between == & ===

SECTION 2

Exercise 2A: Arrays 43

Topics: Creating a simple array
Creating associative arrays using shorthand
Multidimensional arrays
Printing an entire array using print_r()

Exercise 2B: Loops 51

Topics: While loops
Do…while loops
For loops
Foreach loops
Breaking out of a loop
Continue statements

Exercise 2C: Working with Strings 61

Topics: Comparing strings
Converting to upper & lower case
Searching through strings
Case-sensitive & case-insensitive

Exercise 2D: Functions & Objects 67

Topics: Functions
Arguments
Objects & properties
Objects & methods
Private properties
Creating classes that extend classes

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Table of Contents

SECTION 3

Exercise 3A: Form Basics & Security 81
Topics: Post vs. get
Radios, checkboxes, & select fields
Magic quotes
Securing the page
Using functions

Exercise 3B: Sending Email 93
Topics: Installing MailHog
Setting up MAMP Pro
Setting up XAMPP
Sending a test email

Exercise 3C: Simple Form Validation and Email 99
Topics: Sanitizing input
Error checking
Displaying errors
Sending email
Adding a thank you page
Including files

SECTION 4

Exercise 4A: Cookies 109
Topics: Adding cookies
Tracking the number of visits
Sending an email with the cookie info

Exercise 4B: Sessions 119
Topics: Starting a session
Using session variables
Log in/log out
Destroying session variables

Exercise 4C: File Uploads 133
Topics: Making a file upload form
The $_FILES array
Uploading files
Basic security

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Table of Contents

SECTION 5

Exercise 5A: Creating a Database/MySQL/SELECT 147
Topics: Creating a new database
Connecting to the database
SQL basics
The SELECT statement
Display the number of rows returned
Making a reusable connection script
MySQL vs. MySQLi vs. PDO

Exercise 5B: Making a Reusable Connection Script 155
Topics: Error checking
Making an include
Sorting results

Exercise 5C: Prepared Statements 159
Topics: Selecting & filtering results
Preventing SQL injection attacks with prepared statements

SECTION 6

Exercise 6A: SQL: Insert 167
Topics: The INSERT statement
Using phpMyAdmin
Inserting information from a form

Exercise 6B: SQL: Update 177
Topics: The UPDATE statement
Update form
Display data in the update form
Display checkboxes
Hidden fields

Exercise 6C: SQL: Delete 187
Topics: The DELETE statement
Deleting rows from a database
Passing ID variables in a url

Exercise 6D: SQL: Search 191
Topics: Wildcard searches
Searching with a form

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Table of Contents

REFERENCE MATERIAL

Noble's Other Workbooks 193

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Downloading the Class Files

Thank You for Purchasing a Noble Desktop Course Workbook!

These instructions tell you how to install the class files you'll need to go through the exercises in this workbook.

Downloading & Installing Class Files

1. Navigate to the **Desktop**.
2. Create a **new folder** called **Class Files** (this is where you'll put the files after they have been downloaded).
3. Go to **nobledesktop.com/download**
4. Enter the code **php-1308-18**
5. If you haven't already, click **Start Download**.
6. After the **.zip** file has finished downloading, be sure to unzip the file if it hasn't been done for you. You should end up with a **phpclass** folder.
7. Drag the downloaded folder into the **Class Files** folder you just made. These are the files you will use while going through the workbook.
8. If you still have the downloaded .zip file, you can delete that. That's it! Enjoy.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Choosing a Code Editor to Work In

You probably already have a preferred code editor, such as **Sublime Text**, **Atom**, **Brackets**, etc. You can use whatever code editor you want, but if you don't have a preference we recommend **Sublime Text** (available for Mac and Windows). At **sublimetext.com** you can download a free trial. There is no time limit to the free trial, but if you like **Sublime Text**, you can purchase a copy for $70.

Recommended Software

Sublime Text is a great code editor for Mac and Windows. It has a free unlimited trial, which occasionally asks you to buy it. If you like it, you can buy it for $70.

Visit **sublimetext.com** and download the trial (or buy a copy) of **Sublime Text 3**.

We recommend installing some free packages (add-ons) that add great functionality. To make installing packages easier, you should first install **Package Control**.

Installing Package Control

1. Launch **Sublime Text**.
2. Go into the **Tools** menu and choose **Install Package Control**.
3. After a moment you should see a message telling you that Package Control was successfully installed. Click **OK**.

Installing the Emmet Package

Emmet offers shortcuts to make coding faster and easier. Here's how to install it:

1. After Package Control is installed, launch it as follows:
   - Mac: Go into the **Sublime Text** menu > **Preferences > Package Control**.
   - Windows: Go into the **Preferences** menu > **Package Control**.
2. Choose **Install Package**.
3. In the list that appears, start typing **emme** and **Emmet** should appear. Choose it.
4. A message will appear briefly in the bottom status bar to tell you it has been successfully installed.

Installing the SideBarEnhancements Package

The **SideBarEnhancements** package will allow you to do a bunch of useful things to Sublime Text's sidebar. Here's how to install it:

1. Open Package Control as follows:
   - Mac: Go into the **Sublime Text** menu > **Preferences > Package Control**.
   - Windows: Go into the **Preferences** menu > **Package Control**.
2. Choose **Install Package**.
3. In the list that appears, start typing **SideBarEnhancements** and choose it when it appears.
4. A message will appear briefly in the bottom status bar to tell you it has been successfully installed.

Setting Up F12 as a Shortcut for Preview in Browser

The SideBarEnhancements package, in addition to other things, lets you use F12 as a keyboard shortcut for testing a webpage in a browser. To take advantage of this, you will need to add a little code to your Sublime Text preferences.

1. Go to **nobledesktop.com/sublimetext-shortcuts**
2. Copy the following code:

```
[
   { "keys": ["f12"], "command": "side_bar_open_in_browser" , "args":{"paths":
[], "type":"testing", "browser":""}}
]
```

3. In Sublime Text, open the key binding preferences as follows:
   - Mac: Go into the **Sublime Text** menu > **Preferences > Key Bindings**.
   - Windows: Go into the **Preferences** menu > **Key Bindings**.

   A 2-column window will open. The **Default** key bindings are on the left, and **User** (your) key bindings are on the right.
4. In the User key bindings on the right, select and delete any code that's there.
5. In the User key bindings on the right, paste the code you just copied. We recommend previewing using Chrome because we like Chrome's Developer Tools. Add the following code highlighted in bold:

```
[
   { "keys": ["f12"], "command": "side_bar_open_in_browser" , "args":{"paths":
[], "type":"testing", "browser":"Chrome"}}
]
```

6. Save and close the file.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Installing the AutoFileName Package

By default Sublime Text does not suggest path and filenames. Manually typing these is tedious and it's easy to make typos. The **AutoFileName** package adds much needed code hints as you're typing.

1. Open Package Control as follows:
   - Mac: Go into the **Sublime Text** menu > **Preferences > Package Control**.
   - Windows: Go into the **Preferences** menu > **Package Control**.
2. Choose **Install Package**.
3. In the list that appears, start typing **AutoFileName** and choose it when it appears.
4. A message will appear briefly in the bottom status bar to tell you it has been successfully installed.

Restart Sublime Text

Some of the packages may require a restart. Quit and relaunch Sublime Text and you'll be all set!

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

In order to use PHP and MySQL locally on your computer, you first need to set up a local server. Luckily there is an application called MAMP that makes creating a local server easy. It is very important that you do this exercise, or you will be unable to follow along with the book!

If you are taking this class at Noble Desktop, you can skip this exercise. We've already done the setup for you!

MAMP vs. MAMP PRO

For the Mac there are two options for setting up a testing server: MAMP and MAMP PRO. MAMP is the free version of the program and can be used for this entire book —**except for sending and testing email**. MAMP can be used to send emails, but you'd need to manually setup a Postfix server in Apache which is beyond the scope of this book. MAMP PRO makes this easy via a GUI interface and it is the version we use for the class. MAMP PRO also has a few other nice features that make it worth considering if you do a lot of development work and don't want to take the time mucking about in the Terminal.

We've provided instructions on how to install MAMP and MAMP PRO, but keep in mind that if you want to send emails, you'll need MAMP PRO.

Installing MAMP PRO

If you'd like to send emails later in the book, proceed with the following instructions on installing MAMP PRO. If you prefer to use the free version, skip to the next section, Installing MAMP.

1. Launch Chrome.
2. Enter the following URL: **mamp.info/en**
3. Click the **BUY MAMP PRO** button.
4. Follow the instructions to purchase and download the application.
5. Once the download completes, double–click the downloaded file to unzip it.
6. Open the MAMP_MAMP_PRO.pkg file.
7. Follow the instructions for installing MAMP and MAMP PRO. When you're done, the applications will appear in your Applications folder.
8. Open the **MAMP PRO** application. (Go to Hard Drive > Applications > MAMP PRO and open MAMP PRO.app.)

9. The servers should start automatically and a MAMP start page will open up in your web browser. If not, click the **Start** button at the top to start the servers. Then, click the **WebStart** button to open the MAMP start page.

   NOTE: You can get back to this page by clicking the **WebStart** button in the MAMP application, or by going to **localhost:8888/MAMP** in a browser.

10. If you'd like to use the free version and don't want to send emails or want to set up Postfix manually, use MAMP (instructions below).

Installing MAMP

1. Launch Chrome.
2. Enter the following URL: **mamp.info/en**
3. Click the **Download** button underneath **MAMP: One-click-solution for setting up your personal webserver**.
4. Once the download completes, double–click the downloaded file to unzip it.
5. Open the MAMP pkg file.
6. Follow the instructions for installing MAMP. When you're done, the application will appear in your Applications folder.
7. Open the **MAMP** application. (Go to Hard Drive > Applications > MAMP and open MAMP.app.)
8. The MAMP start page will open up in your web browser. If not, click the **Start Servers** button.

   NOTE: You can get back to this page by clicking **Open WebStart page** in the MAMP application, or by going to **localhost:8888/MAMP** in a browser.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Windows: Setting Up the Local Server

Exercise Overview

In order to use PHP and MySQL locally on your computer, you first need to set up a local server. Luckily there is an application called XAMPP that makes creating a local server easy. It is very important that you do this exercise, or you will be unable to follow along with the book!

If you are taking this class at Noble Desktop, you can skip this exercise. We've already done the setup for you!

Installing XAMPP

1. Launch **Chrome**.
2. Enter the following URL: **tinyurl.com/xampp56**

   The download should start automatically (save the file on your desktop if it asks).
3. Once the download completes, double–click the file, then click **Run** to start the installation process.
4. If the installer warns that you have antivirus software running that may interfere with the installer, click **Yes** to continue the installation.
5. If it warns about UAC just click **OK**.
6. Click **Next** to start.
7. Under **Select Components**, make sure all of the options are checked, then click **Next**.
8. **C:\xampp** should already be set for the Installation folder. Click **Next**.
9. UNcheck **Learn more about Bitnami for XAMPP**, then click **Next**.
10. Click **Next** one more time to start installing XAMPP.
11. It should go though the install. When it's done, make sure **Do you want to start the Control Panel now?** is checked, then click **Finish**.

    The XAMPP Control Panel will open.
12. Press **Start** next to **Apache** and **MySQL** to start those services.
13. If you get a warning that Windows Firewall has blocked some features, click **Allow access**.
14. Open Chrome and go to this address: **localhost**
15. Click **English** as your preferred language to finish the installation.

    XAMPP is now installed and running.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Copying the Class Files into the Web Server Folder

Throughout this workbook you will be editing class files that we have prepared for you. In order for these PHP files to run, they need to be located in the web server folder that you installed during setup.

1. If you haven't already, download the class files for this book. Refer to the **Downloading the Class Files** chapter for instructions on how to download them.
2. Go to **Desktop > Class Files**.
3. Click once on the **phpclass** folder to select it.
4. Hit **Cmd–C** (Mac) or **Ctrl–C** (Windows) to copy it.
5. Follow the appropriate Mac or Windows instructions below:
   - Mac: Go into **Hard Drive > Applications > MAMP > htdocs**
   - Windows: Go into **My Computer > C: > xampp > htdocs**
6. Press **Cmd–V** (Mac) or **Ctrl–V** (Windows) to paste the class files.

   Now the class files are in the root directory of our active web server.

MAMP PRO: Displaying Errors

By default MAMP PRO does not display PHP errors on the page; instead it just shows a blank document. This is a good secure practice for a production site, but for local development it makes it very hard to troubleshoot your code.

1. Launch **MAMP PRO**.
2. Click the **PHP** tab.
3. At the bottom, next to **Log errors**, check **to screen**.
4. Click **Save**.
5. If it asks to restart the server, click **Yes**.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

This exercise gets you started with the basics of PHP syntax. If you're familiar with a language like JavaScript, a lot of this will look familiar to you, but even if you do not have previous programming experience, you'll find that PHP is pretty simple to learn.

Mac: Launching MAMP PRO

1. If MAMP PRO is not already running, go to **Hard Drive > Applications > MAMP PRO** and double–click **MAMP PRO.app**.
2. MAMP PRO will launch and it may automatically open the MAMP start page in your default web browser.

   If it does not automatically open, you can click the **WebStart** button in the MAMP PRO application (you may need to click the **Start** button first), or in a browser go to **localhost:8888/MAMP**

Windows: Launching XAMPP

1. If XAMPP is not already running, navigate to the **C: > xampp** folder, then double–click **xampp-control.exe**.
2. Next to **Apache** click **Start**.
3. Next to **MySQL** click **Start**.

Echo, Strings, & Variables

1. In your code editor, open **first.php** from the **phpclass** folder. This folder is located in your **htdocs** folder here:
   - Mac: **Hard Drive > Applications > MAMP > htdocs**
   - Windows: **C: > xampp > htdocs**

2. In between the **<body>** tags, add the following bold code:

```
<body>
<?php
   echo "Hello Universe";
?>
</body>
```

   Let's break down this very basic piece of code. We use **<?php** and **?>** to delineate the PHP code from the rest of the page. Everything inside will be PHP, and everything outside will be plain HTML.

   The **echo** command simply prints something to the page. Here we have put a string in double quotes, so it just outputs that string.

   The **;** signifies the end of the PHP statement. It is very important not to forget the semi-colon or you may get errors in your code.

3. Save the page.

4. Open a browser and go to:

   - Mac: **localhost:8888/phpclass/first.php**
   - Windows: **localhost/phpclass/first.php**

   The page should read:

   Hello Universe

5. The echo command will also do simple math. Switch back to your code editor.

6. Delete everything between the **<?php ?>** tags and add the following bold code:

```
<?php
   echo 2 + 2;
?>
```

   Notice that we did not put quotes around **2 + 2**. This tells PHP to evaluate the numbers as an expression, rather than a string.

7. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/first.php**
   - Windows: **localhost/phpclass/first.php**

   The page should read:

   4

8. Leave the page open in the browser so we can come back to it after making changes.

9. Let's see what happens when we add quotes. Switch back to your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

10. Add the quotes as shown in bold:

```
<?php
   echo "2 + 2";
?>
```

11. Save the page, go to the browser, and reload **first.php**.

    The page should read:

    2 + 2

    The quotes tell PHP to evaluate it as a string of text.

12. Now let's make a variable. Switch back to your code editor.

13. Select everything between the **`<?php ?>`** tags and replace it with the code shown in bold:

```
<?php
   $myMessage = "Hello Universe";
   echo $myMessage;
?>
```

    Don't forget the semi-colons at the end of each line!

14. Save the page, go back to the browser, and reload **first.php**.

    The page should read:

    Hello Universe

    You just created a variable called `$myMessage` and printed it to the page. All PHP variables start with a **$** sign.

    NOTE: Variables are also case-sensitive, so `$mymessage` and `$myMessage` would be totally unrelated!

Single Quotes vs. Double Quotes

You may have noticed that we have been using double quotes for our strings. We could also use single quotes, but there are a few subtle differences.

1. Switch back to your code editor.

2. Add **single** quotes around $myMessage as shown in bold:

```
$myMessage = "Hello Universe";
echo '$myMessage';
```

3. Save the page, go to the browser, and reload **first.php**.

   The page should read:

   $myMessage

   The single quotes tell PHP to evaluate everything inside them as a literal string of characters.

4. Now let's try the same thing but with **double** quotes. Switch back to your code editor.

5. Add **double** quotes around $myMessage as shown in bold:

```
$myMessage = "Hello Universe";
echo "$myMessage";
```

6. Save the page, go to the browser, and reload **first.php**.

   The page should read:

   Hello Universe

   Wow, it evaluated the variable. **Double** quotes are a bit more flexible and will allow you to mix variables and strings all at once. Let's try it out.

7. Switch back to your code editor.

8. Add the following code shown in bold:

```
$myMessage = "Hello Universe";
echo "$myMessage, nice to meet you.";
```

9. Save the page, go to the browser, and reload **first.php**.

   The page should read:

   Hello Universe, nice to meet you.

Escaping Characters

1. Switch back to your code editor.

2. Select everything between the **<?php ?>** tags and replace it with the following bold code:

```
<?php
   echo 'It's nice to meet you.';
?>
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Save the page, go to the browser, and reload **first.php**.

   The page will error out and not work. Why? PHP sees the single quote character in **It's** as the end of the string, and then it gets confused by all the characters after it. We must instead escape the single quote.

4. Switch back to your code editor.

5. Add the following **** shown in bold:

```
echo 'It\'s nice to meet you.';
```

6. Save the page, go to the browser, and reload **first.php**.

   The page should read:

   It's nice to meet you.

7. We can do the same thing with double quotes. Switch back to your code editor.

8. Edit the line so it reads:

```
echo "Tell him I said \"Hi\".";
```

9. Save the page, go to the browser, and reload **first.php**.

   It should read:

   Tell him I said "Hi".

   There are other escape characters you can use, such as **** for a backslash itself, **\t** for a tab, **\n** for a new line, and **\r** for a carriage return.

Concatenation

There are a few ways to concatenate text and variables. (Concatenate means to "link things together in a chain or series.")

1. Switch back to your code editor.

2. Select everything between the **<?php ?>** tags and replace it with the following bold code:

```
<?php
   echo "This ". "is ". "my string.";
?>
```

   Pay careful attention to the spaces when you type this!

3. Save the page, go to the browser, and reload **first.php**.

   It should read:

   This is my string.

4. The period syntax is a little bit more flexible. With it we can append something directly to a variable. Switch back to your code editor.

5. Select everything between the **`<?php ?>`** tags and replace it with the following bold code:

```
<?php
   $msg = "I like carrots.";
   $msg .= " And broccoli.";
   echo $msg;
?>
```

6. Save the page, go to the browser, and reload **first.php**.

   It should read:

   I like carrots. And broccoli.

   The **.** appends the string to the variable. This is a useful shorthand and you will see it often.

Comments

1. Switch back to your code editor.

2. To comment out a single line of code, add **//** to the beginning of the line, as shown below in bold:

```
$msg = "I like carrots.";
//$msg .= " And broccoli.";
echo $msg;
```

   In most code editors, the middle line will gray out, indicating that it is a comment and will no longer be executed.

3. To comment out multiple lines, use **/*** and ***/** as shown in bold below:

```
/*$msg = "I like carrots.";
//$msg .= " And broccoli.";
echo $msg;*/
```

   Now the entire block of code will gray out indicating it is one big comment.

4. Close the file, saving changes if you wish. We're done with it.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

In this exercise you'll learn how to set up Sublime Text or Dreamweaver to quickly preview pages without having to type a long localhost URL into the browser every time. If you use a different code editor, you can skip this exercise.

If You've Done Other Noble Desktop Coding Books

If you've setup Sublime Text in other Noble Desktop workbooks, you may have installed the **View in Browser** package. For this workbook, you will need to use the **SideBarEnhancements** package instead.

1. To see if you have SideBarEnhancements installed, launch Package Control as follows:
   - Mac: Go into the **Sublime Text** menu > **Preferences > Package Control**.
   - Windows: Go into the **Preferences** menu > **Package Control**.
2. Choose **Package Control: List Packages** to see a list of all installed packages.
3. If **SideBarEnhancements** is not listed, go to the **Before You Begin** instructions at the beginning of this book and do the following sections:
   - Installing the SideBarEnhancements Package
   - Setting Up F12 as a Shortcut for Preview in Browser
4. Continue with the Sublime Text instructions below.

Sublime Text Users Only

If you're using Sublime Text, and have the **SideBarEnhancements** package installed, you can set up the F12 key to preview the page instead of typing out the localhost URL in a browser.

1. In Sublime Text, close any files or windows you may have open.
2. In Sublime Text, go to **File > New File**.
3. Go to **Project > Add Folder to Project** and:
   - Mac: Navigate to **Hard Drive > Applications > MAMP > htdocs**. Select the **phpclass** folder and click **Open**.
   - Windows: Navigate to **C: > xampp > htdocs**. Select the **phpclass** folder and click the **Select Folder** button.

4. A **phpclass** folder has been added in the sidebar on the left. **Ctrl–click** (Mac) or **Right–click** (Windows) on the folder, and in the menu that opens choose **Project > Edit Preview URLs**.

5. A SideBarEnhancements.json file will open. We've prepared a file with the code needed to set up the preview URLs for this workbook. In the sidebar on the left, inside the **phpclass** folder (should be expanded) click on **sbe-code-mac.txt** (Mac) or **sbe-code-win.txt** (Windows).

   We got this code from the SideBarEnhancements package website. The first part of the code specifies the path for the folder of the site on your computer. **url_testing** allows you to set the URL of your local server, opened via F12.

6. Select all the code and copy it.

7. Close the file. You should be back in SideBarEnhancements.json.

8. Paste the code into the SideBarEnhancements.json file.

9. Save and close the file.

   You will now be able to preview files in the **phpclass** folder by hitting **F12** (or **fn–F12** depending on your keyboard settings) instead of typing out the URL in a browser.

   NOTE: F12 may not work on a Mac unless you change or disable the **Show Dashboard** keyboard shortcut in **System Preferences > Keyboard** (or **Mission Control**).

Dreamweaver Users Only

While it's not required, if you are using Dreamweaver, you can define a site and testing server to make previewing easier. It will also keep all the files organized.

1. Launch **Dreamweaver**.

2. In Dreamweaver, choose **Site > New Site**.

3. In the dialog that opens next to **Site Name**, enter **PHP Class**

4. Next to **Local Site Folder**, click the folder icon and:

   - Mac: Go to **Hard Drive > Applications > MAMP > htdocs**. Select the **phpclass** folder and click **Choose**.

   - Windows: Go to **C: > xampp > htdocs**. Double–click on the subfolder **phpclass** and click the **Select** button.

5. On the left, click **Servers**.

6. At the bottom, click the **Add new Server** button .

7. Next to **Server Name** enter **php**.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

8. Next to **Connect using**, choose **Local/Network**.
9. Next to **Server Folder**, click the folder icon and:
   - Mac: Go to **Hard Drive > Applications > MAMP > htdocs**. Select the **phpclass** folder and click **Choose**.
   - Windows: Go to **C: > xampp > htdocs**. Double–click the subfolder **phpclass** and click the **Select** button.
10. For **Web URL** enter:
    - Mac: **`http://localhost:8888/phpclass`**
    - Windows: **`http://localhost/phpclass`**
11. Click **Save**.
12. You should now see the server listed. On the right, uncheck **Remote**, and check on **Testing**, as shown below.

Name	Address	Connection	Remote	Testing
php	ChancoPro 750/Applica...	Local/Net...	☐	☑

13. Click **Save**.
14. Now you can go to **File > Real-time Preview** (or **File > Preview** depending on your version of Dreamweaver) instead of having to type a localhost URL into the browser every time.

    Also, you should see the **Files** panel (**Window > Files**) that lists all the files in the site.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Math! Who doesn't love it? OK, maybe not everyone loves math, but as programmers it's something we end up doing quite a bit of. Here we'll explore how PHP works with numbers.

Arithmetic Operators

1. In your code editor, open **math.php** from the **phpclass** folder.
2. In between the **<body>** tags add the following bold code:

```
<?php
   $myVar = 2;
   $myVar = $myVar + 1;
   echo $myVar;
?>
```

   This code makes a new variable called `$myVar` with the value of **2**, adds **1** to it, and then displays the new value.

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/math.php**
   - Windows: **localhost/phpclass/math.php**

   The page should read:

   3

4. Switch back to your code editor.
5. We can also **increment** the variable by 1. Select the **last two** lines, then replace them with the following bold code:

```
<?php
   $myVar = 2;
   $newVar = $myVar++;
   echo '$myVar: ' . $myVar;
?>
```

   This increments the value of **`$myVar`** by one.

6. Save the page, go to the browser, and reload **math.php**.

   It should now say:

   $myVar: 3

Assignment Operators

What if you wanted to increment the value of a variable directly and by a value other than one? You'd use what is called an **Assignment Operator**.

1. Switch back to your code editor.
2. Replace the last two lines with the following bold code:

```
$myVar = 2;
$myVar += 5;
echo $myVar;
```

This makes a variable called $myVar with the value of **2**, adds **5** to it, and then displays the new value.

`$myVar += 5` is equivalent to **`$myVar = $myVar + 5`**

3. Save the page, go to the browser, and reload **math.php**.

The screen should now say **7**.

4. Close the file. We're done with it.

Table of Arithmetic Operators

Operator	Meaning	Example
+	Addition	`$n + 1`
-	Subtraction	`$n - 5`
*	Multiplication	`$n * 20`
/	Division	`$n / 3`
%	Modulus (remainder)	`$n % 6`
++	Increment	`++$n`
--	Decrement	`--$n`

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Table of Assignment Operators

Example	Meaning	Alternative Longhand Syntax
`$n = 3`	`$n equals 3`	`$n = 3`
`$n += 3`	`$n equals $n plus 3`	`$n = $n + 3`
`$n -= 3`	`$n equals $n minus 3`	`$n = $n - 3`
`$n *= 3`	`$n equals $n times 3`	`$n = $n * 3`
`$n /= 3`	`$n equals $n divided by 3`	`$n = $n / 3`
`$n %= 3`	`$n equals the remainder of $n divided by 3`	`$n = $n % 3`
`$n .= 3`	`$n equals $n concatenated with 3`	`$n = $n . 3`

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Conditional operators will be one of the most-used elements of your programming life. Simply put, conditional operators are a way to choose when certain things happen. For example, **if** something is true, **then** do something. If it is **not** true, do something else.

If/Else Statements

An if/else statement is one of the simplest and most-used pieces of logic you'll find in any programming language. Let's explore how they work in PHP.

The basic syntax for a PHP if statement is as follows:

```
if (something is true) {
   do something
}
```

1. In your code editor, open **ifelse.php** from the **phpclass** folder.
2. Let's create two simple variables and see if they are equal. In between the **<body>** tags add the following bold code:

```
<?php
   $a = 8;
   $b = 8;

   if ($a == $b) {
      echo "a is equal to b";
   }
?>
```

First we set two variables, both equal to 8. Then we test to see if they are equal. Note that to **set** a variable you use **one** = sign, and to **test** if they are equal, you use **two** ==. (You can also use three equal signs === to test if they are equal and of the same type. We'll go into this more later.)

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   The page should say **a is equal to b**.
4. Let's see what happens if we make it so they are not equal. Switch back to your code editor.

5. Set $a to 20 as shown in bold:

```
$a = 20;
$b = 8;
```

6. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   The page will be blank. Because $a is now larger than $b, the code between the {} does not execute at all.

7. Switch back to your code editor.

8. Now we can add an **else** statement. This will execute if the if statement is false. Add the following bold code:

```
if ($a == $b) {
   echo "a is equal to b";
}
else {
   echo "a does not equal b";
}
```

9. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   Now the **else** statement will run and your page should say **a does not equal b**.

Elseif Statements

But what if we want to display a message if $a is GREATER than $b? We can use an **elseif** statement.

1. Switch back to your code editor.

2. In between the if and the else statements add an **elseif** as shown in bold:

```
if ($a == $b) {
   echo "a is equal to b";
}
elseif ($a > $b) {
   echo "a is greater than b";
}
else {
   echo "a does not equal b";
}
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   Now the **elseif** statement will run and your page should say **a is greater than b**.

Switch Statements

Sometimes you need to execute different pieces of code depending on what a particular variable (or expression) is equal to. Using multiple elseif statements could get tedious and hard to read. That's where the **switch** statement comes in.

For this example pretend the user is selecting different pages for navigation on a website. Depending on which one the user chooses, different code will execute.

1. Switch back to your code editor.
2. Delete everything between the **`<?php ?>`** tags, then enter the following bold code:

```
$nav = "home";
switch ($nav)
{
   case "home":
      echo "Take me to the home page.";
      break;
   case "news":
      echo "Take me to the news section.";
      break;
   case "about":
      echo "Take me to the about page.";
      break;
}
```

Here's what's going on with this code:

- First we set the variable **`$nav`** equal to **"home"**.
- **`switch ($nav)`** indicates that different code will execute depending on what **`$nav`** is equal to.
- **`case "home":`** if the **`$nav`** variable is equal to **`"home"`** then do the code in that section.
- **`break;`** this tells PHP to stop executing the section. **If you did not include the break commands PHP would run every case!**

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   The page should say **Take me to the home page.**
4. Switch back to your code editor.
5. Change the **$nav** variable to equal **news**:

   ```
   $nav = "news";
   ```
6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   The page should now say **Take me to the news section.**
7. Switch back to your code editor.
8. Try changing **$nav** to **sports**:

   ```
   $nav = "sports";
   ```
9. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   The page will be blank. Let's set a default value in case the user doesn't choose anything.
10. Switch back to your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

11. Enter the following bold code:

```
switch ($nav)
{
   case "home":
      echo "Take me to the home page.";
      break;
   case "news":
      echo "Take me to the news section.";
      break;
   case "about":
      echo "Take me to the about page.";
      break;
   default:
      echo "Please choose a page.";
      break;
}
```

12. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/ifelse.php**
    - Windows: **localhost/phpclass/ifelse.php**

    It will now say **Please choose a page.**

More Comparison Operators

To test if something is not equal, we use the **!=** sign.

1. Switch back to your code editor.
2. Delete everything between the **<?php ?>** tags, then enter the following bold code:

```
$a = 3;
$b = 4;

if ($a != $b) echo "a does not equal b.";
```

   This tests to see if $a is equal to $b. You'll also notice that we are using a shorthand of the if statement. If the statement is true, PHP will execute the code after the (). No need for brackets in this simple case!

3. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/ifelse.php**
    - Windows: **localhost/phpclass/ifelse.php**

    As predicted, $a does not equal $b.

Logical Operators

Let's take a quick look at some different comparison operators such as AND and OR.

1. Switch back to your code editor.
2. Delete the **if** statement and replace it with the following bold code:

```
$a = 3;
$b = 4;

if ($a == 3 || $b == 3) echo "One of these is 3.";
```

This tests to see if either $a or $b is equal to 3.

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   The page should say **One of these is 3.**
4. Switch back to your code editor.
5. Change the **||** to an **&&**, to test if they are BOTH equal to 3. Enter the following bold code:

```
if ($a == 3 && $b == 3) echo "Both of these equal 3.";
```

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   You'll see that the code does not fire, because it is not true.

The Difference Between == & ===

Double equal signs will test for equality, but are a bit more loose about the data type that they test for. For example, if you have an integer set to **3**, and a string set to the character **'3'**, the double equal sign would say that they are equal. However, if you have a **triple** equal sign, they would NOT be equal because they are different types (one is a numeric value, and the other is a string).

1. Switch back to your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

2. Delete everything between the **`<?php ?>`** tags, then enter the following bold code:

```
$a = 3;
$b = '3';

if ($a == $b) echo "a equals b.";
if ($a === $b) echo "a is the same type as b.";
```

3. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/ifelse.php**
   - Windows: **localhost/phpclass/ifelse.php**

   You'll see that only the first echo statement is run because the two variables are not of the same type.

4. Close the file. We're done with it.

Comparison Operators

Example	Description
`$a == $b`	True if $a equals $b
`$a != $b`	True if $a does not equal $b
`$a === $b`	True if $a equals $b and is of the same type
`$a !== $b`	True if $a does not equal $b or they are not the same type
`$a > $b`	True if $a is greater than $b
`$a < $b`	True if $a is less than $b
`$a <> $b`	True if $a does not equal $b
`$a <= $b`	True if $a is less than or equal to $b
`$a >= $b`	True if $a is greater than or equal to $b

Logical Operators

Example	Description
`$a and $b`	True if $a and $b are both true
`$a or $b`	True if $a or $b is true
`$a xor $b`	True if $a or $b is true, but not if both are true
`!$a`	True if $a is not true
`$a && $b`	True if $a and $b are both true
`$a || $b`	True if $a or $b is true

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

One of the most common and powerful types of variables is called an **array**. Simply put, an array holds lots of information that is grouped together. There are many ways you can create, output, and manipulate arrays, but here we'll just focus on some of the most commonly-used basics.

Creating a Simple Array

1. In your code editor, open **array.php** from the **phpclass** folder.
2. In between the **<body>** tags add the following bold code:

```
<?php

   $customer['firstName'] = 'Jeremy';
   $customer['lastName'] = 'Kay';
   $customer['city'] = 'New York';
   $customer['state'] = 'NY';

?>
```

   You've just created an array, **$customer**. The array contains various information about an imaginary customer. This type of array is called an **associative** array.

3. Outputting any value from this array is simple. Below the array you just made, add the following bold code:

```
$customer['firstName'] = 'Jeremy';
$customer['lastName'] = 'Kay';
$customer['city'] = 'New York';
$customer['state'] = 'NY';

echo "My first name is: " .$customer['firstName'];
```

   This is pretty straightforward, but notice that you have to add a **period** before the array variable in order to concatenate it to the string. If you don't, you'll get an error.

4. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/array.php**
   - Windows: **localhost/phpclass/array.php**

   Wow. It's a first name!

5. Switch back to your code editor.
6. Comment out the echo line as shown in bold:

```
//echo "My first name is: " .$customer['firstName'];
```

7. Another type of array is called an **indexed** array. Instead of using words to categorize it we use numbers. Below the echo line, add the following bold code:

```
$cars[0] = 'Ferrari';
$cars[1] = 'Porsche';
$cars[2] = 'Mustang';
```

   This is an array of our fabulous car collection. One thing to notice is that the first item starts with **zero**. PHP starts all of its elements as **0**, so that is something to keep in mind (`$cars[1]` is actually the **second** item in the array).

8. Echoing an element from this array is a bit easier. Add the following bold code:

```
$cars[0] = 'Ferrari';
$cars[1] = 'Porsche';
$cars[2] = 'Mustang';

echo "My other car is a $cars[0]";
```

9. Save the page and then in a browser go to (or reload the page if it's already open):

   - Mac: **localhost:8888/phpclass/array.php**
   - Windows: **localhost/phpclass/array.php**

   The page will read: **My other car is a Ferrari**

10. Switch back to your code editor.

11. Comment out the echo line as shown in bold:

```
//echo "My other car is a $cars[0]";
```

Creating Associative Arrays Using Shorthand

In PHP 5.4 and later, a simplified syntax for arrays has been introduced. In previous versions, the **array()** function was used as a common shorthand for building associative arrays. The current shorthand is even simpler, using just square brackets (**[ ]**)! If you're not sure which version of PHP you're using, do the appropriate sidebar as follows.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Checking PHP Version on MAC

1. Launch **MAMP PRO**.
2. In the main window, go to the **PHP** tab.
3. Next to **Default version** make sure the dropdown menu is set to **5.4** or later.
4. If you've made any changes, click **Save**.
5. Return to your code editor.

Checking PHP Version on PC

1. Launch the **XAMPP Control Panel**.
2. To the right of the **Apache** module, click the **Admin** button to open a webpage.
3. At the top of the orange sidebar on the left, it says **XAMPP** followed by a version number, such as [PHP: 5.6.3]. Make sure it says **5.4** or later.
4. If you don't have 5.4 or later, you can go to **apachefriends.org** to download a newer version of XAMPP.
5. Return to your code editor.

1. Below the bottom echo line, add the following bold code:

```
$restaurant = [
   'name' => 'Nobu',
   'type' => 'Sushi',
   'price' => 'Expensive'
];
```

Here we have created a new array called **`$restaurant`** and given it some values. Notice everything goes in between the brackets. The item on the left is called a **key**, and the element on the right of the **=>** is its **value**.

2. Let's test outputting part of this array. Add the following bold code:

```
$restaurant = [
   'name' => 'Nobu',
   'type' => 'Sushi',
   'price' => 'Expensive'
];

echo $restaurant['name']. ' is very ' .$restaurant['price'];
```

   Here we simply output the variables and again use the **period** to concatenate them to our string.

3. Save the page and then in a browser go to (or reload):

   - Mac: **localhost:8888/phpclass/array.php**
   - Windows: **localhost/phpclass/array.php**

   The page will read: **Nobu is very Expensive**

Multidimensional Arrays

What if you wanted to store a list of many restaurants in one array? We could use what is called a multidimensional array.

1. Switch back to your code editor.

2. Delete all the code that has your restaurant info in it (not the echo statement). It's probably a bit easier just to retype this from scratch. Add the following bold code:

```
$restaurant = [

];
```

   Here we have an empty array awaiting our glory.

3. In between the empty brackets add the following bold code:

```
$restaurant = [
   [
      'name' => 'Nobu',
      'type' => 'Sushi',
      'price' => 'Expensive'
   ]
];
```

   Notice we have nested one array inside of another.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

4. Next, add a **second** element to the main **`$restaurant`** array. Add the following bold code:

```
$restaurant = [
   [
      'name' => 'Nobu',
      'type' => 'Sushi',
      'price' => 'Expensive'
   ],
   [
      'name' => 'Burger King',
      'type' => 'Fast food',
      'price' => 'Cheap'
   ]
];
```

   Be sure to add a **comma** after the first array! Now we have a second array added to the main **`$restaurant`** array.

5. How would we output this? Let's check it out. Below the array code modify the **echo** statement as shown in bold:

```
echo $restaurant[0]['name']. ' is very ' .$restaurant[0]['price'];
```

   The **[0]** in each array element tells PHP to access the first element in the array, which in this case is Nobu.

6. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/array.php**
   - Windows: **localhost/phpclass/array.php**

   The page will read: **Nobu is very Expensive**

7. Switch back to your code editor.

   Let's add a line break, then output the second element in the array.

8. Add the following bold code:

```
echo $restaurant[0]['name']. ' is very ' .$restaurant[0]['price'];
echo '<br>';
```

9. **Copy** the first **echo** command, and then **paste** it below the `<br>` tag as shown in bold:

```
echo $restaurant[0]['name']. ' is very ' .$restaurant[0]['price'];
echo '<br>';
echo $restaurant[0]['name']. ' is very ' .$restaurant[0]['price'];
```

10. Change two instances of **0** to **1** as shown in bold:

```
echo $restaurant[0]['name']. ' is very ' .$restaurant[0]['price'];
echo '<br>';
echo $restaurant[1]['name']. ' is very ' .$restaurant[1]['price'];
```

11. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/array.php**
    - Windows: **localhost/phpclass/array.php**

    The page will read:

    Nobu is very Expensive
    Burger King is very Cheap

Printing an Entire Array Using print_r()

During the development of a site, it is often very useful to quickly print out the entire contents of an array.

1. Switch back to your code editor.
2. After the last echo statement, add the following bold code:

```
print_r($restaurant);
```

3. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/array.php**
    - Windows: **localhost/phpclass/array.php**

    The page will output the entire array (along with the previous statements):

```
Array ( [0] => Array ( [name] => Nobu [type] => Sushi [price] => Expensive )
[1] => Array ( [name] => Burger King [type] => Fast food [price] => Cheap ) )
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

4. If this code is a bit hard to read, you can view the source in your browser and the formatting will be a lot better:

```
Array
(
   [0] => Array
      (
         [name] => Nobu
         [type] => Sushi
         [price] => Expensive
      )

   [1] => Array
      (
         [name] => Burger King
         [type] => Fast food
         [price] => Cheap
      )

)
```

   The source looks better because PHP adds spacing and line breaks, but remember that a browser will ignore whitespace.

5. Switch back to your code editor and close the file. We're done with it.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Loops are an incredibly important and often-used element of your programming tool belt. Here we'll explore the many kinds of loops PHP has to offer.

While Loops

One of the simplest loops is the while loop. The basic idea of this loop is that **while** an expression is true, **do** something. For example:

```
while (I have more than $1) {
   Keep shopping!
}
```

In this statement you'll keep shopping as long as you have more than $1. Perhaps it's not the wisest savings strategy, but it sure is fun!

1. In your code editor, open **loops.php** from the **phpclass** folder.
2. Here's how to write a while loop in PHP. In between the **<body>** tags add the following bold code:

```
<?php

   $money = 100;

   while ($money > 1) {
      --$money;
      echo "I just bought something! I have $money dollars left now.<br>";
   }

?>
```

What's going on here?

- First, we set **$money** to **100**. So we're starting off with 100 dollars.
- Next, we say do something while **$money** is **greater** than **1**.
- **--$money** will decrement our money by **1**. So every item we are buying costs 1 dollar.
- Then we **echo** on the page that we just bought something and have xx dollars left. By putting the **$money** variable inside of our double quotes it will print the value of the variable to the screen.
- Also note that we have to manually insert a **
** tag to make a line break.

3. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/loops.php**
   - Windows: **localhost/phpclass/loops.php**

   Wow, that's a lot of shopping! Notice that we start with 100 dollars, but we do a little shopping straight away so our first echo statement says we have 99 dollars left.

4. Switch back to your code editor.

5. What would happen if we started off with only one dollar? Change the **$money** variable to **1** as shown in bold below:

```
$money = 1;

   while ($money > 1) {
      --$money;
      echo "I just bought something! I have $money dollars left now.<br>";
   }
```

6. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/loops.php**
   - Windows: **localhost/phpclass/loops.php**

   Nothing happens! Why is this? Because we only start off with 1 dollar and we will only go shopping if we have **more** than 1 dollar, the while statement will not execute.

Do...while Loops

The do…while statement is a slight variation of while. The do…while statement guarantees that your code is executed at least once at the beginning and then will execute more times if the while conditional statement is true. For example:

```
do {
    do something
} while (you can keep doing it again if this is true)
```

1. Switch back to your code editor.

2. Delete everything between the **<?php ?>** tags, then enter the following bold code:

```
$money = 1;
do {
   --$money;
   echo "I just bought something! I have $money dollars left now.<br>";
} while ($money > 1);
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/loops.php**
   - Windows: **localhost/phpclass/loops.php**

   Now you get your shopping in. The **do** code will fire the first time no matter what, and may or may not fire after that depending on whether the **while** conditional is true.

For Loops

The **for** loop is a bit more complicated than the **while** loop. Its syntax is as follows:

```
for (expression 1; expression 2; expression 3) {
    do something
}
```

- **Expression 1** is run **once** at the **beginning** of the loop.
- **Expression 2** is evaluated at the beginning of each loop and if it is **true** the code will continue to loop.
- **Expression 3** is run at the **end** of each loop.

1. Switch back to your code editor.
2. Delete everything between the **`<?php ?>`** tags, then enter the following bold code:

```
for ($money = 100; $money > 1; --$money)
{
    echo "I have $money dollars, so I can still shop!<br>";
}
```

   - The **first** expression sets the **`$money`** variable to **100**.
   - The **second** expression is similar to the conditional in the if statements we've seen before. If this second expression is true, then the loop will run. In this case the loop will run as long as $money is greater than 1.
   - The **third** expression will run at the end of each loop, decrementing $money by 1.

   You can see that here the **for** loop acts like a shorthand to the while statements we wrote earlier. Note that the three expressions are separated by semi-colons!

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/loops.php**
   - Windows: **localhost/phpclass/loops.php**

   You'll see that we start off with 100 dollars and each time the loop runs we have one less.

Foreach Loops

This loop is typically used to loop through an array. The basic syntax is as follows:

```
foreach (someArray as someTemporaryValue) {
    do something, most typically you would output someTemporaryValue
}
```

So, for each element in the array, we set the value of the array item to be **someTemporaryValue**. We can do whatever we want between the brackets but most often we just want to display the value of the array.

1. We've created an array for you. Open **foreach.php** from the **phpclass** folder.
2. Take a moment to look through the file. We have **three** different types of arrays:
   - **`$movies`**: A simple indexed array containing the names of some movies.
   - **`$customer`**: A simple associative array containing some customer information.
   - **`$cars`**: A multidimensional array containing the info on some different cars.

   The **foreach** loop will allow us to easily display all the information contained in these arrays.
3. Scroll to the bottom of the page and below the closing **`?>`** tag (around line 77) make another set of **`<?php ?>`** tags. We don't need to do this, but it visually helps to separate the code we will be writing now from the code that was provided for you.
4. First let's loop over the **`$movies`** array. In between the **`<?php ?>`** tags you just added, enter the following bold code:

   ```
   foreach ($movies as $value) {
      echo $value;
      echo '<br>';
   }
   ```

   This takes the array **`$movies`** and puts its value into the **`$value`** variable (you can call this variable anything you want). Then it echoes the **`$value`** and adds a **`<br>`** to make the output easier to read.
5. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/foreach.php**
   - Windows: **localhost/phpclass/foreach.php**

   You'll see a list of all the movies.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

6. Next let's output the values of the **$customer** array. Change the **$movies** array to **$customer** as shown in bold:

```
foreach ($customer as $value) {
   echo $value;
   echo '<br>';
}
```

7. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/foreach.php**
   - Windows: **localhost/phpclass/foreach.php**

   Now you'll see all the customer information.

8. Switch back to your code editor.

   What if we wanted to include the text of the labels as well? We want to output something like:

   firstName; Jeremy
   lastName; Kay
   etc..

   To do this we use a **$key $value** pair. This will pass both the name of the key as well as the value of the key.

9. Modify the code as shown in bold:

```
foreach ($customer as $key => $value) {
   echo "$key: $value";
   echo '<br>';
}
```

   $key in this case becomes the **name** of the variable in the associative array and **$value** is the **value** of the variable.

10. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/foreach.php**
    - Windows: **localhost/phpclass/foreach.php**

    You'll see the following:

    firstName: Jeremy
    lastName: Kay
    ID: 78
    email: jeremy@jeremy.com

11. What if we wanted to loop through the entire **$cars** multidimensional array? We would simply put one foreach loop inside another. Switch back to your code editor.

12. Take a look at the multidimensional array **`$cars`**. It contains the details of a number of nice cars. It is essentially one array that contains a bunch of smaller arrays.

13. Delete the foreach loop and replace it with the following bold code:

```
foreach ($cars as $i) {
   echo $i;
}
```

14. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/foreach.php**
    - Windows: **localhost/phpclass/foreach.php**

    You'll see the following output repeated five times:

    Notice: Array to string conversion in .../htdocs/phpclass/foreach.php on line 80
    Array

    A notice tells you that you're doing something not totally correct, but it is correct enough to let the page still execute. In this case, we are echoing `$i` which contains the value of the associative array that contains the individual car details. It converts the associative array to a string, which is why you see Array outputted after the notice. What we need to do now is loop through **`$i`** to output the details.

15. Switch back to your code editor.

16. Delete the **`echo $i;`** line and replace it with another **foreach** loop as shown in **bold**:

```
foreach ($cars as $i) {
   foreach ($i as $key => $value) {
      echo "$key: $value";
   }
}
```

    This loops through the **`$i`** array and outputs its key and value.

17. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/foreach.php**
    - Windows: **localhost/phpclass/foreach.php**

    Wow, that's a mess but at least we know it worked.

18. Switch back to your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

19. Let's add some HTML formatting so it's a bit easier to read. Add the bold code:

```
foreach ($cars as $i) {
   echo '<p>';
     foreach ($i as $key => $value) {
        echo "$key: $value";
        echo '<br>';
     }
   echo '</p>';
}
```

This puts each car in its own <p> tag and separates each detail with a line break.

20. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/foreach.php**
- Windows: **localhost/phpclass/foreach.php**

Much better.

Breaking Out of the Loop

Every now and then you may want to break out of a loop. To do that, simply put the **break** command in the loop.

1. Switch back to your code editor.
2. Open **break.php** from the phpclass folder.
3. Let's start by making a simple loop that counts down from 50. In between the **<body>** tags add the following code:

```
<?php
   for ($count = 50; $count > 1; --$count) {
      echo "$count <br>";
   }
?>
```

This is a simple **for** loop. First it sets **$count** to 50, then says to run as long as **$count** is greater than 1. At the end of each loop it decrements **$count** by 1.

4. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/break.php**
- Windows: **localhost/phpclass/break.php**

You'll see a countdown from 50.

5. Switch back to your code editor.

6. We want to stop the loop when **$count** equals 25. Add the following bold code:

```
for ($count = 50; $count > 1; --$count) {
   echo "$count <br>";
   if ($count == 25) {
      break;
   }
}
```

This says, "if $count equals 25, stop the loop".

7. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/break.php**
- Windows: **localhost/phpclass/break.php**

The loop stops at 25. Simple.

Continue Statements

A **continue** statement tells PHP to stop the execution of the current loop and continue immediately to the next iteration of the loop. So it doesn't stop the entire loop, but just the current iteration of the loop.

As an example let's have our loop only output even numbers.

1. Switch back to your code editor.
2. First delete the current **if** statement. The code should look as follows:

```
for ($count = 50; $count > 1; --$count) {
   echo "$count <br>";
}
```

3. To test whether a number is even or odd we can use the **%** (modulus) command. Modulus is the remainder after division. If you divide an even number by 2 then the remainder is 0. If you divide an odd number by 2 then the remainder will be 1. Add the following bold code:

```
for ($count = 50; $count > 1; --$count) {
   if ($count % 2 == 1) {
      continue;
   }
   echo "$count <br>";
}
```

This says: "Divide $count by 2 and if the remainder is 1, then stop the current iteration of the loop and continue to the next iteration." So if the code encounters an odd number, it won't get to the echo statement.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

4. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/break.php**
   - Windows: **localhost/phpclass/break.php**

   You'll see a list of only even numbers.
5. Switch back to your code editor.
6. Close any files you may have open. We're done with them.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Let's explore some basic string functions.

Comparing Two Strings

One way to compare strings is to use the **`strcmp()`** function. This will perform a **case-sensitive** comparison. If the strings are equal it will output **0**. If the first string is greater than the second string it will return a positive value, and if the second string is greater it will return a negative value.

1. In your code editor, open **strings.php** from the **phpclass** folder.
2. To test it out we'll write a simple if/else statement to see if two strings match. Notice that the second string is capitalized. In between the **`<body>`** tags add the following bold code:

```
<?php

   $var1 = 'noble';
   $var2 = 'Noble';

   if (strcmp($var1,$var2) == 0) {
      echo 'The strings match.';
   }
   else {
      echo 'The strings do not match.';
   }

?>
```

   If the strings exactly match **`strcmp()`** will output **0**. Otherwise, the strings do not match.

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/strings.php**
   - Windows: **localhost/phpclass/strings.php**

   As you can see, the strings do not match.
4. Switch back to your code editor.

5. Next let's try a **case-insensitive** search. To do that we'll use **`strcasecmp()`**. Change the code as shown in bold:

```
if (strcasecmp($var1,$var2) == 0) {
      echo 'The strings match.';
   }
else {
   echo 'The strings do not match.';
}
```

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/strings.php**
   - Windows: **localhost/phpclass/strings.php**

   Now the strings match.

Converting to Upper & Lower Case

1. Switch back to your code editor.
2. To convert a string to lower case we use the **`strtolower()`** function. Delete the code between the **`<?php ?>`** tags and replace it with the following bold code:

```
$email = 'MYEMAIL@MYURL.COM';

echo strtolower($email);
```

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/strings.php**
   - Windows: **localhost/phpclass/strings.php**

   The string is converted to lower case.
4. Switch back to your code editor.
5. Converting to upper case is exactly the same with the **`strtoupper()`** function. Let's test it. Delete the code between the **`<?php ?>`** tags and replace it with the following bold code:

```
$var = 'why are you yelling?';

echo strtoupper($var);
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/strings.php**
   - Windows: **localhost/phpclass/strings.php**

   The page will display: WHY ARE YOU YELLING?
7. Switch back to your code editor.
8. A more useful function only capitalizes the first word in a sentence. It also leaves any capitalization that is already there alone.
9. Delete the code between the **`<?php ?>`** tags and replace it with the following bold code:

```
$var = "don't you think that PHP is great?";

echo ucfirst($var);
```

10. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/strings.php**
    - Windows: **localhost/phpclass/strings.php**

    Don't you think that PHP is great?

Searching Through Strings

To search through a string we could use the **`strpos()`** function. The syntax would look like:

```
strpos($haystack, $needle)
```

This says to search through the "haystack" (typically the larger string that we are searching in) for the "needle" (the smaller substring we are searching for). It returns a numerical value with the location of the "needle." If the needle is found at the very start of the haystack it will return 0. Keep this in mind when evaluating whether the statement is true or not. Also note that this is a case-sensitive search.

1. Switch back to your code editor.

2. Delete the code between the **<?php ?>** tags and replace it with the following bold code:

```
$haystack = 'abcdefg';
$needle = 'b';

echo strpos($haystack, $needle);
```

   This searches the haystack "abcdef" for the needle "b." It then outputs the numeric position.

3. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/strings.php**
   - Windows: **localhost/phpclass/strings.php**

   It will say **1**. This is the position of the sub-string (needle) in the string (haystack).

4. Switch back to your code editor.

5. Change the `$needle` to **'a'** as shown in bold:

```
$haystack = 'abcdefg';
$needle = 'a';

echo strpos($haystack, $needle);
```

6. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/strings.php**
   - Windows: **localhost/phpclass/strings.php**

   It will say **0**. It is very important to keep this in mind when trying to find out if a string exists.

7. Switch back to your code editor.

8. Delete the echo then add the following bold code:

```
if ( strpos($haystack, $needle) != false ) {
      echo 'I found my needle!';
}
```

   This says if `strpos()` is not equal to false (in other words if it is found), echo that the needle was found.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

9. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/strings.php**
   - Windows: **localhost/phpclass/strings.php**

   The page will be blank. Why is this? Because **'a'** is the very first element of the string, its location is 0. In the if statement we test to see if it is false, but because we didn't use strict typing it evaluated 0 as false. Typically in true/false statements 1=true and 0=false.

10. Switch back to your code editor.
11. Make it more strict by adding an extra = sign as shown in bold:

```
if ( strpos($haystack, $needle) !== false ) {
        echo 'I found my needle!';
}
```

12. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/strings.php**
    - Windows: **localhost/phpclass/strings.php**

    It should now say, "I found my needle!"

13. Switch back to your code editor.
14. Capitalize the **$needle** variable as shown in bold:

```
$haystack = 'abcdefg';
$needle = 'A';
```

15. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/strings.php**
    - Windows: **localhost/phpclass/strings.php**

    The page will be blank because it is a case-sensitive search.

16. Switch back to your code editor.
17. To make it case-insensitive, change the function to **stripos()** as shown in bold:

```
if ( stripos($haystack, $needle) !== false ) {
      echo 'I found my needle!';
}
```

18. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/strings.php**
    - Windows: **localhost/phpclass/strings.php**

    It will now say, "I found my needle!" again.

19. Switch back to your code editor.
20. Close the file. We're done with it.

    There are many more string functions that are explained in detail in the PHP manual: **php.net/manual/en/ref.strings.php**

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

In this exercise, we'll learn the basics of **functions** and how to use **arguments** within them. We'll go over how to create **objects** with **properties** and **methods**. We'll explore the differences between **public** and **private** properties, and additionally how to **extend** classes' functionality.

Functions

Functions in PHP are code routines that can be called at any time.

1. In your code editor, open **objects.php** from the **phpclass** folder. Notice this has the same basic HTML structure as other PHP files we've worked with.
2. In between the **`<body>`** tags, add PHP tags, as shown:

```
<?php

?>
```

3. Say we've been working with a lawyer and want to generate a bill for their fees. We can create a function to do this. Add the following bold code to define a function:

```
<?php

   function get_lawyer_bill()

?>
```

To define a function, you start with the word **`function`**, followed by the name of the function, followed by parentheses **()**.

There are some rules for naming functions:

- They can include alphabetical or numerical characters.
- They cannot begin with a number.
- They cannot include spaces, but can use underscores.

4. Add variables to the function, as shown in bold:

```
<?php

   function get_lawyer_bill() {
      $hourly_rate = 100;
      $hours = 3;
   }

?>
```

Similarly to if/else statements, any time functions are called, the code between the curly braces {} is executed.

5. Add the following bold code:

```
<?php

   function get_lawyer_bill() {
      $hourly_rate = 100;
      $hours = 3;
      return $hourly_rate * $hours;
   }

?>
```

The keyword **`return`** specifies the value to be passed out of the function (printed on-screen, stored in a database, etc.). In this case, we want to return the hourly rate multiplied by the hours.

6. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/objects.php**
   - Windows: **localhost/phpclass/objects.php**

   The page is blank because functions don't do anything unless they're called. By defining the function we've made it possible to get the lawyer bill, but if we want to see anything, we need to call it.

7. Switch back to your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

8. Add the following bold code:

```
<?php

   function get_lawyer_bill() {
      $hourly_rate = 100;
      $hours = 3;
      return $hourly_rate * $hours;
   }

   echo get_lawyer_bill();

?>
```

   Here, we've called the function by name. The keyword **echo** will print it to the screen.

9. Save the page, then go back to the browser and reload **objects.php**. Now you should see the value **300**.

10. Return to your code editor.

11. One cool thing about functions is you can call them as many times as you want. To see this in action, add the bold code:

```
<?php

   function get_lawyer_bill() {
      $hourly_rate = 100;
      $hours = 3;
      return $hourly_rate * $hours;
   }

   echo get_lawyer_bill();
   echo '<br>';
   echo get_lawyer_bill();
   echo '<br>';
   echo get_lawyer_bill();
   echo '<br>';

?>
```

   This will call the function three times and put a line break between each time.

12. Save the file, go to the browser, and reload **objects.php**. Now you should see **300** print three times.

Arguments

Functions become more useful when arguments are passed in. Arguments are outside values that are sent into a function when it's called. Arguments can vary each time a function is called. Arguments are placed between parentheses after the function name when defining a function.

1. Let's make our `get_lawyer_bill` function more flexible by making the hourly rate and hours variable. Add the following bold code:

```
function get_lawyer_bill($hourly_rate, $hours) {
   $hourly_rate = 100;
   $hours = 3;
   return $hourly_rate * $hours;
}
```

2. We no longer need the code defining the variables within the function. Delete them so that you are left with the code shown:

```
function get_lawyer_bill($hourly_rate, $hours) {
   return $hourly_rate * $hours;
}
```

3. Now we need to return the product of the two variables. First off, we don't need that third function call. Select it, as shown in bold, and **delete** it:

```
echo get_lawyer_bill();
echo '<br>';
echo get_lawyer_bill();
echo '<br>';
echo get_lawyer_bill();
echo '<br>';
```

4. Let's try passing different arguments into the function at the same time. Say we have two lawyers, each with different hourly rates. Add the bold code:

```
echo get_lawyer_bill(100, 3);
echo '<br>';
echo get_lawyer_bill(120, 2);
```

5. Save the file, then go back to the browser and reload **objects.php**.

6. Check out the results of each lawyer's bill (**300** and **240**)! This makes our code much more reusable and versatile. As you can see, arguments allow for a lot of flexibility and help you reuse important logic without having to rewrite code multiple times.

7. Return to your code editor.

Arguments can be provided with default values by including an equals sign (**=**). In the case that the argument is omitted, the default will be used.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

8. Add the code shown in bold to make the default number of hours equal to 1:

```
function get_lawyer_bill($hourly_rate, $hours = 1) {
   return $hourly_rate * $hours;
}
```

9. Edit the arguments as shown:

```
function get_lawyer_bill($hourly_rate, $hours = 1) {
   return $hourly_rate * $hours;
}
echo get_lawyer_bill(200);
echo '<br>';
echo get_lawyer_bill(200, 2);
```

10. Save the file, go to the browser, and reload **objects.php**. The first number that prints is **200**. Because we didn't specify the hours, it defaulted to 1 hour. The second number, **400**, is the result of 200 x 2.

Objects & Properties

PHP objects are code constructs that can contain information, similarly to how multidimensional arrays do. Let's start by creating a simple object to see how it works.

1. In your code editor, **delete** the entire function so that you're left with empty PHP tags:

```
<?php

?>
```

2. The most basic object type in PHP is called **stdClass** (standard class). Type the bold code:

```
<?php

   $lawyer = new stdClass();

?>
```

This creates a new `stdClass` basic object called `lawyer`. The keyword, **`new`**, is always used to initialize a new PHP object of any kind.

Notice the way **`stdClass`** is written. This format of stringing words together while capitalizing each word is called CamelCase. PHP class names are written in CamelCase. Note that the first letter may be capital or lowercase.

3. Now that we've created a new `lawyer` object, let's give it a name. Add:

```
<?php

   $lawyer = new stdClass();
   $lawyer->name = 'Juanita Jones';

?>
```

Variables attached to objects are called **properties**. The **->** is used to specify that **name** is a property of our `lawyer` object.

4. Let's assign properties for Juanita's hourly rate and law specialties:

```
<?php

   $lawyer = new stdClass();
   $lawyer->name = 'Juanita Jones';
   $lawyer->hourly_rate = 100;
   $lawyer->specialties = ['Family Law', 'Tax Law'];

?>
```

Anything can become a property of an object. As you can see, we've just created a string, an integer, and an array as properties.

5. Call the results by adding the bold code:

```
<?php

   $lawyer = new stdClass();
   $lawyer->name = 'Juanita Jones';
   $lawyer->hourly_rate = 100;
   $lawyer->specialties = ['Family Law', 'Tax Law'];

   echo $lawyer->name . ' charges $' . $lawyer->hourly_rate . ' per hour.';

   echo '<br><br>';

?>
```

This will print that the lawyer object charges the hourly rate per hour followed by a couple line breaks.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

6. Use `print_r()` to examine the entire object by adding:

```
<?php

   $lawyer = new stdClass();
   $lawyer->name = 'Juanita Jones';
   $lawyer->hourly_rate = 100;
   $lawyer->specialties = ['Family Law', 'Tax Law'];

   echo $lawyer->name . ' charges $' . $lawyer->hourly_rate . ' per hour.';

   echo '<br><br>';

   echo "<pre>";
   print_r($lawyer);
   echo "</pre>";

?>
```

Remember in the Arrays exercise, with the multidimensional array, print_r() dumped the entire array on screen. Above and below the print_r() function, we added `<pre>` tags so that the printed results will be preformatted (including the line breaks, tabs, etc. in the source code) and easier to read.

7. Save the file, go to the browser, and reload **objects.php**. You should see **Juanita Jones charges $100 per hour.** followed by the code for the object and its properties.

Objects & Methods

At this point, the functionality we've explored is very similar to that of associative arrays. However, you should know that objects can do much more than what we've done so far! Objects can contain **methods**, which are functions inside of a class. Using methods within a class can make objects very powerful.

1. Go back to your code editor.

2. **Delete** all the `lawyer` object code so that you're again left with empty PHP tags:

```
<?php

?>
```

3. Time to define our own class. Add a new Lawyer class, as shown in bold:

```
<?php

   class Lawyer {
   }

?>
```

Class declarations start with the keyword, **`class`**, followed by the class name. Unlike most variables in PHP, these are in CamelCase and will often start with a capital letter.

4. Add some properties to the class:

```
<?php

   class Lawyer {
      public $name, $hourly_rate;
   }

?>
```

We declared **`public`** so that these properties can be easily accessed, set, and retrieved (like with stdClass objects).

5. Add a method to get the lawyer bill:

```
<?php

   class Lawyer {
      public $name, $hourly_rate;

      function get_bill($hours) {

      }
   }

?>
```

This creates a `get_bill` function with an `$hours` argument. The `$hourly_rate` will be a property of the `Lawyer` object, so it's already known to us within the function, which is why we didn't need to specify it as an argument.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

6. Now we need to compute the total. In order to get the `$hourly_rate`, we'll need to refer to the object internally. This is because we want to refer to this particular instance of the `Lawyer` object from within the `Lawyer` class. To accomplish our goal, we can use the special variable, **`$this`**. Add the bold code:

```
class Lawyer {
   public $name, $hourly_rate;

   function get_bill($hours) {
      $total = $this->hourly_rate * $hours;
   }
}
```

`$this` lets PHP know we're referring to one specific instance of the `Lawyer` class. **`$this->hourly_rate`** gets the hourly rate of this specific lawyer.

7. Add the following code to return a string saying how much and to whom you owe money:

```
class Lawyer {
   public $name, $hourly_rate;

   function get_bill($hours) {
      $total = $this->hourly_rate * $hours;
      return "You owe " . $this->name . ' $' . $total . '.';
   }
}
```

8. Just like when we declared a function earlier, declaring a class doesn't make anything happen until we instantiate an instance of that class. In order to create a new lawyer, we need to create a variable using the `new` keyword. Create a new Lawyer by adding the bold code:

```
   function get_bill($hours) {
      $total = $this->hourly_rate * $hours;
      return "You owe " . $this->name . ' $' . $total . '.';
   }
}
$alfred = new Lawyer();
```

9. Let's give our new lawyer a name and hourly rate:

```
   function get_bill($hours) {
      $total = $this->hourly_rate * $hours;
      return "You owe " . $this->name . ' $' . $total . '.';
   }
}
$alfred = new Lawyer();
$alfred->name = 'Alfred Frank';
$alfred->hourly_rate = 50;
```

10. Call the results by adding:

```
$alfred = new Lawyer();
$alfred->name = 'Alfred Frank';
$alfred->hourly_rate = 50;

echo $alfred->get_bill(2);
```

This is how we call a method on an object. First we create the new Lawyer (`$alfred`) and set up some properties for it. Then we call the `get_bill` method within the class and pass it an argument (the number of hours).

11. Save the file, go to the browser, and reload **objects.php**. You should see **You owe Alfred Frank $100.**

12. Return to your code editor.

13. One really powerful thing we can do is have multiple instances of the same class going at once, all with different properties. Add a new Lawyer preceded by a line break:

```
$alfred = new Lawyer();
$alfred->name = 'Alfred Frank';
$alfred->hourly_rate = 50;

echo $alfred->get_bill(2);

echo '<br>';

$juanita = new Lawyer();
```

This creates a new instance of the Lawyer class, which can have completely different properties.

14. Let's add properties and get the bill for the new lawyer:

```
echo '<br>';

$juanita = new Lawyer();
$juanita->name = 'Juanita Jones';
$juanita->hourly_rate = 150;
echo $juanita->get_bill(5);
```

15. Save the file, go to the browser, and reload. You should see **You owe Juanita Jones $750.** Now we're writing some really dynamic code! We've got two different objects with different properties, and our code is really efficient because it's all running through the same functions and the same class.

16. Return to your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Private Properties

So far we've only dealt with public properties, but we should also discuss private properties. Most classes in PHP source code use private properties, which as you might guess, are not as easily accessed as public properties.

1. Around line 12, change **public** to **private**:

   ```
   private $name, $hourly_rate;
   ```

2. Save the file, go to the browser, and reload. You will either see a blank page or an error, which says, **Cannot access private property Lawyer::$name**.

3. Return to your code editor and around line 20, notice this is where we tried to set the Lawyer name.

   With public properties using the stdClass object, we could set a property by simply making it equal to something. With private properties, we'll have to take a different approach. Usually private properties are set through methods. Commonly, important properties are set when an object is first created.

4. It'll help to clean up our code so we can start fresh. Delete lines 20–30, so that section of code looks like so:

   ```
   $alfred = new Lawyer();

   ?>
   ```

5. Create a new function around line 14:

   ```
   class Lawyer {
      private $name, $hourly_rate;

      function __construct()

      function get_bill($hours) {
   ```

   Make sure you type two underscores (**__**) before `construct`. `__construct` is a special method for any class you create. This method is called whenever your class is initialized with that new keyword.

6. Between the parentheses, you can accept arguments and use them to do things with your class. For example, setting private properties when a class instance is created. Add the bold code:

   ```
   function __construct($name, $hourly_rate)
   ```

7. To set these properties permanently on the object, add:

```
function __construct($name, $hourly_rate) {
   $this->name = $name;
   $this->hourly_rate = $hourly_rate;
}
```

8. Around line 24, add our new Lawyer's name and hourly rate:

```
   $alfred = new Lawyer('Alfred Frank', 50);

?>
```

Here, we are passing our new Lawyer's name and hourly rate within that call to `new Lawyer`. Those two values will get sent to the `__construct` function and used to set up the Lawyer's private properties.

9. Let's call the results for a bill of 10 hours. Add:

```
   $alfred = new Lawyer('Alfred Frank', 50);
   echo $alfred->get_bill(10);

?>
```

10. Save the file, go to the browser, and reload. You should see **You owe Alfred Frank $500.** Great, things are working again!

Creating Classes that Extend Classes

New classes can be created to extend (or add additional functionality to) existing classes.

1. In your code, delete lines 24–25 and replace them with the following bold code:

```
   class BillboardLawyer extends Lawyer {
   }

?>
```

This tells PHP that this class is going to build on the Lawyer class. So we don't have to redefine anything that's already there (for example, we don't need to add a private name or hourly rate or redo the `__construct` function).

2. We do however, want the `get_bill` function to work a little differently here. Add the bold code to start redefining the function:

```
class BillboardLawyer extends Lawyer {
   function get_bill($hours) {
   }
}
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Our goal is for the original bill to print along with some additional text. First, let's make sure we get the original bill by typing:

```
class BillboardLawyer extends Lawyer {
   function get_bill($hours) {
      $original = parent::get_bill($hours);
   }
}
```

   This says to go to the `parent` class (`Lawyer`) and execute the function `get_bill` there using the value `$hours`.

4. We want to return that original bill appended with a bit more text. Type:

```
class BillboardLawyer extends Lawyer {
   function get_bill($hours) {
      $original = parent::get_bill($hours);
      return $original . ' And you pay NOTHING until we recover for YOU!';
   }
}
```

5. Now let's make Alfred a billboard lawyer and get his bill for 20 hours. Type:

```
class BillboardLawyer extends Lawyer {
   function get_bill($hours) {
      $original = parent::get_bill($hours);
      return $original . ' And you pay NOTHING until we recover for YOU!';
   }
}

$alfred = new BillboardLawyer('Alfred Frank', 500);
echo $alfred->get_bill(20);
```

6. Save the file, go to the browser, and reload. You should see **You owe Alfred Frank $10000. And you pay NOTHING until we recover for YOU!**

   In summary, we're now using the same `__construct` method and private properties of the `Lawyer` class without having to do anything additional. Then we're overriding the `get_bill` function, calling the original `parent` function, and appending text onto the output before returning it.

7. Great! We're done for now. Feel free to close your files.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

In this exercise, we'll cover the basics of PHP form submission. We'll explore the difference between **POST** and **GET**, how to deal with radio, checkbox, and select fields, and how to secure your pages from XSS (cross-site scripting) attacks.

Setting Up a Basic Form

1. In your code editor, open **form.php** from the **form-basic** folder in the **phpclass** folder.

   This is a simple signup page that we need to prepare to use with PHP.

2. Around line **13** find the `<form>` tag:

   ```
   <form action="" method="get" name="signup" id="signup">
   ```

   Notice that the action is currently blank and the method is set to **get**. There are two **method** options: **get** and **post**. **Get** will put all of your form data into the URL and **Post** will put the data behind the scenes. We'll explore the differences between the two shortly.

3. First let's link the form to the action page. We've already started a basic action page called **form-action.php**. In the **`<form>`** tag, change the **action** to point to the new page, as shown in bold below:

   ```
   <form action="form-action.php" method="get" name="signup" id="signup">
   ```

4. Save the file.

5. Now let's add some code on our action page that will display the contents of the form. When you submit a form to PHP with the **get** method the values are stored in an array called **`$_GET`**.

   To access a specific field, such as email, you'd use: **`$_GET['email']`**

   If you submitted a form with **post**, PHP puts everything into the **`$_POST`** array. In that case, to access a specific field such as email, you'd use: **`$_POST['email']`**

6. Open **form-action.php** from the **form-basic** folder in the **phpclass** folder.

7. For now let's just display the entire **`$_GET`** array. Under the opening **`<body>`** tag add the following bold code:

   ```
   <?php

      print_r($_GET);

   ?>
   ```

8. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/form-basic/form.php**
   - Windows: **localhost/phpclass/form-basic/form.php**

   NOTE: We are previewing **form.php**, the file with our form. This is not the form-action.php file we just edited.

9. Fill out the form and submit it.

10. On the action page, you'll see something like:

```
Array ( [name] => Jeremy Kay [email] => jeremy@jeremy.com
[level] => expert [publications] => Elle [howoftentrack] =>
daily [comments] => This form is great! [submit] => Sign Me
Up! )
```

11. Look at the URL of the page, and you'll see that all of the contents of the form are there as well:

```
http://localhost:8888/phpclass/form-basic/form-action.php?name=Jeremy
+Kay&email=jeremy%40jeremy.com&level=expert&publications=Daily+Racing
+Form&publications=Elle&howoftentrack=daily&comments=This+form+is+great!
&submit=Sign+Me+Up!
```

   Whew, that's a mess. If we format it a bit more legibly, it looks like:

```
http://localhost:8888/phpclass/form-basic/form-action.php
?name=Jeremy+Kay
&email=jeremy%40jeremy.com
&level=expert
&publications=Daily+Racing+Form
&publications=Elle
&howoftentrack=daily
&comments=This+form+is+great%21
&submit=Sign+Me+Up%21
```

   Notice that the first portion is the URL of the server and page. The form portion is put after the **?** and each form field begins with a **&** followed by the name of the field. URLs cannot contain spaces or special characters so it is automatically URL encoded for us (notice all the spaces have become **+** signs).

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Post vs. Get

While the **get** method has its uses, it also has its drawbacks. First, it is obviously less secure because a user can see and manipulate the data however they want. Second, because the whole form is put into the URL, there are length limits. If you had a lot of fields or a very long comments section, it might get cut off because browsers put a limit on how long a URL can be. For these reasons, it's better to use **post** when submitting forms.

1. Switch back to your code editor.
2. Switch to **form.php** and change the method to **post** as shown in bold:

```
<form action="form-action.php" method="post" name="signup" id="signup">
```

3. Save the page.
4. Switch to **form-action.php**.
5. Let's display each form field individually. **Delete** the **`print_r()`** code and its surrounding php tags.
6. Then, next to the **Name** and **Email** add the following bold code:

```
<p><strong>Name:</strong> <?php echo $_POST['name']; ?></p>
<p><strong>Email:</strong> <?php echo $_POST['email']; ?></p>
```

7. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/form-basic/form.php**
   - Windows: **localhost/phpclass/form-basic/form.php**
8. Fill out the form and submit it. You'll see the name and email displayed. Excellent.

Radios, Checkboxes, & Select Fields

Working with radios, checkboxes, and select fields requires special care. First, if the checkbox or radio is not selected by the user, then the form does not send the input at all. What this means is that we have to check to see if the form field exists before we start to process it. Second, if they accept more than one input, such as a range of checkboxes, they must be sent to PHP as an array. Because it is sent as an array, we then need to process that form element differently.

First we need to set up the checkbox fields to be properly named for PHP. Any input that accepts multiple values must have brackets added after its name for PHP to properly process all the values. For example if a radio field is named **publications** you must change the **name** to **publications[]** to work with PHP. Be sure you change the **name** of the input and not the **id**.

1. Switch back to your code editor.

2. Switch to **form.php**.
3. Find the **What do you read?** checkbox fields around line **31**. Add brackets to the **name** as shown in bold below:

```
<label><input name="publications[]" type="checkbox" id="publications_drf"
value="Daily Racing Form"> Daily Racing Form</label>
<label><input name="publications[]" type="checkbox" id="publications_elle"
value="Elle"> Elle</label>
```

4. Save the page.
5. Switch to **form-action.php**.
6. Around line **11**, find **Level** and add php tags as shown below (add them between the p tags):

```
<p><strong>Level:</strong>
   <?php

   ?>
</p>
```

7. First we must check to see if the **level** form element has been set. Add the following bold code:

```
<p><strong>Level:</strong>
   <?php
      if ( isset($_POST['level']) ) {

      }
   ?>
</p>
```

We must check if the level element is set in case the user does not select a radio option. If we didn't check this PHP would error out.

8. Now output the level. Add the following bold code:

```
<p><strong>Level:</strong>
   <?php
      if ( isset($_POST['level']) ) {
         echo $_POST['level'];
      }
   ?>
</p>
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

9. Next let's take care of the publications checkbox. Around line **18** find **Publications** and add php tags as shown below (add them between the p tags):

```
<p><strong>Publications:</strong>
   <?php

   ?>
</p>
```

10. Add an if statement that checks to see if the publications checkbox has been checked. If we didn't write this if statement and the user did not check the checkbox, PHP would give an error. Add the following bold code:

```
<p><strong>Publications:</strong>
   <?php
      if ( isset($_POST['publications']) ) {

      }
   ?>
</p>
```

11. Because publications is an array, we have to process it in a special way (remember we added [] brackets after the name which makes it an array). An easy way to deal with the array is to simply turn it into a comma-delimited list. To do that, we can use the **`implode()`** function which, although not as exciting to watch as its name might have you believe, is quite useful in this case. Add the following bold code:

```
<p><strong>Publications:</strong>
   <?php
      if ( isset($_POST['publications']) ) {
         echo implode(', ', $_POST['publications']);
      }
   ?>
</p>
```

This says to turn the publications array into a list. **implode()** takes two parameters, the first of which is the **delimiter**. In our case we want a comma-delimited list followed by a space. The second parameter is the array variable we want to implode.

12. The next two fields are easy, as we can just output the strings to the page. Find the **How often track** paragraph and add the following bold code:

```
<p><strong>How often track:</strong> <?php echo $_POST['howoftentrack']; ?></p>
```

13. For the **Comments** paragraph, we'll output the variable in a separate paragraph tag to make it a bit neater. Add the bold code below the **Comments** paragraph:

```
<p><strong>Comments:</strong></p>
<p><?php echo $_POST['comments']; ?></p>
```

14. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/form-basic/form.php**
    - Windows: **localhost/phpclass/form-basic/form.php**
15. Fill out the form and test everything out. Everything should be outputting nicely.
16. Go back and fill out the form again, except this time in the **Comments** field, be sure to add a few line breaks such as:

    This is a great form!

    And the comments field is spectacular.
17. Submit the form and notice that the line breaks were not preserved. There is a function called **`nl2br()`** that will take care of this for us.
18. Switch back to your code editor. You should still be in **form-action.php**.
19. Wrap the **`nl2br()`** function around the **comments** post echo as shown in bold:

    ```
    <p><strong>Comments:</strong></p>
    <p><?php echo nl2br($_POST['comments']); ?></p>
    ```

    Don't forget the closing parenthesis!
20. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/form-basic/form.php**
    - Windows: **localhost/phpclass/form-basic/form.php**
21. Fill out the form and test out the comments field. The line breaks will now be preserved.

Magic Quotes

Magic Quotes are an old PHP security feature designed to help prevent injection attacks when querying a database. They work by automatically adding a backslash in front of every quote. Unfortunately this does not adequately protect the database, and has the annoying side effect of adding a bunch of slashes to your text! Because of this, Magic Quotes have been deprecated in newer versions of PHP. However, many ISPs still have Magic Quotes set to on. It is best practice to turn them off before developing your application.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Magic Quotes in Action

1. In a browser go to:
   - Mac: **localhost:8888/phpclass/form-basic/form.php**
   - Windows: **localhost/phpclass/form-basic/form.php**
2. For the name enter: **Peter O'Toole**.
3. Submit the form.
4. If magic quotes are turned on, it will be output as: **Peter O\'Toole**

   If there is no backslash, then you are using a newer version of PHP or Magic Quotes have been turned off and you can skip over the next section and continue with the **Securing the Page** section.

How to Tell if Magic Quotes Are On

1. Open **test-magic-quotes.php** from the **phpclass** folder.
2. Notice the first line of code reads:

   ```
   <?php phpinfo(); ?>
   ```

   phpinfo() is a function that displays all the info for your current PHP installation. You can also run this function on your remote server to find out more about what features your ISP has turned on and off.
3. Let's see this function in action. In a browser go to:
   - Mac: **localhost:8888/phpclass/test-magic-quotes.php**
   - Windows: **localhost/phpclass/test-magic-quotes.php**
4. Press **Cmd–F** (Mac) or **Ctrl–F** (Windows) and search for **magic_quotes_gpc**

   NOTE: If you can't find it on the page, then you are using a newer version of PHP where magic quotes have been deprecated.
5. Ideally, it will be set to Off. XAMPP by default has the feature turned off, however by default MAMP (if running an older version of PHP) has them turned on. If you're using XAMPP, you can skip to the next section.
6. Switch to **MAMP Pro**.
7. Click the **PHP** tab.
8. Under **PHP Version**, choose **PHP 5.6.x**. (We'd like to use the latest version, but you should check to see what version your ISP is running before developing your app. If your ISP is still running 5.2 then you should use that.)

9. Go to **File > Edit Template > PHP > PHP 5.6.x php.ini**.
10. If you get a warning, just click **OK**.
11. Scroll down to around line **381**.
12. Add the following bold code:

```
magic_quotes_gpc = Off
```

13. Close the window and save when asked.
14. Hit the **Stop** button.
15. Hit the **Start** button.
16. Go back to your browser and reload the `phpinfo()` page.
17. Search for **magic_quotes_gpc**. You'll see that it is now **Off**.

Securing the Page

We've learned the basics of form submission, however there is one giant problem: **the page we have now is totally insecure**. As a rule you should **NEVER** display any user input on the page directly without first sanitizing it. For example, a malicious user could insert any sort of JavaScript or HTML that could actually be run on the page.

Let's see what might happen if we do not first sanitize user input.

1. In a browser go to:
   - Mac: **localhost:8888/phpclass/form-basic/form.php**
   - Windows: **localhost/phpclass/form-basic/form.php**
2. In the **Name** field, enter the following:

```
<i>Your Name</i>
```

3. Submit the form. You'll see that your name has been *italicized*.

While this may not seem serious, an informed attacker could potentially wreak havoc on your site by running any JavaScript or HTML they desire.

There are a number of different ways to filter user input. You can choose to strip out all HTML characters so **`<i>Your Name</i>`** would become **Your Name**. Because we are outputting to an HTML page, the safest way to is automatically escape any HTML characters. We do this using the **`htmlentities()`** function.

`htmlentities()` will take **`<i>`** and convert it to: **`<i>`**

This way the browser does not render it as HTML (and will not run any script tags), but will display exactly as the user entered it.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

4. Switch back to your code editor.
5. Switch to **form-action.php**.
6. We are going to build a function to sanitize the input. At the very top of the document, above the **<!DOCTYPE HTML>** enter the following bold code:

```
<?php
   function sanitizeInput() {

   }
?>
```

You've just created a function called **sanitizeInput**. Now let's make it do something.

7. We want the function to accept a parameter (in this case our form input), do something to it, and then return the variable back to us. Add the following bold code:

```
function sanitizeInput($myInput) {
   //do something
   return $myInput;
}
```

The function now accepts one parameter (it goes in the parentheses), and returns the variable $myInput. The comment that says "do something" is where we will process the variable.

8. OK, now we can actually sanitize the input. First we want to trim away any whitespace the user may enter. Delete the comment and in its place add the following bold code:

```
function sanitizeInput($myInput) {
   $myInput = trim($myInput);
   return $myInput;
}
```

9. Now that the function is at least partly built, we can put it to use. To use it we'll wrap it around the **name** and **email** `$_POST` variables on our page. Add the bold code as shown below:

```
<p><strong>Name:</strong> <?php echo sanitizeInput($_POST['name']); ?></p>
<p><strong>Email:</strong> <?php echo sanitizeInput($_POST['email']); ?></p>
```

Don't forget the closing parentheses!

10. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/form-basic/form.php**
    - Windows: **localhost/phpclass/form-basic/form.php**

11. Add a bunch of spaces before your name and submit the form. You'll see that the space is automatically removed.
12. Switch back to your code editor.
13. Next we can add the **htmlentities()** function which will convert any HTML characters to harmless HTML entity equivalents. In the **sanitizeInput()** function add the following bold code:

```
function sanitizeInput($myInput) {
   $myInput = trim($myInput);
   $myInput = htmlentities($myInput, ENT_QUOTES, 'UTF-8');
   return $myInput;
}
```

The **htmlentities()** function takes three parameters:

- The string variable that we want to process.
- How it'll handle quotes. ENT_QUOTES encodes both double and single quotes. There are other options if you want to leave single quotes alone, but converting all of them is the safest option.
- The character encoding type. Most modern webpages use UTF-8 encoding which allows special characters such as a • or ñ character.

14. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/form-basic/form.php**
    - Windows: **localhost/phpclass/form-basic/form.php**
15. In the **Name** field enter the following:

```
<i>Your Name</i>
```

16. Submit the form. You'll see that the text is not italicized.
17. View the source of the page and you'll see that the HTML characters have been converted into their HTML entity equivalent.
18. Switch back to your code editor.
19. Switch back to **form-action.php**.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

20. Finish adding **sanitizeInput()** to the rest of the `$_POST` variables as shown in bold. For the comments field, be sure to add **`sanitizeInput()`** inside the **`nl2br()`** so that the line breaks are preserved.

```
<p><strong>Level:</strong>
   <?php
      if ( isset($_POST['level']) ) {
         echo sanitizeInput($_POST['level']);
      }
   ?>
</p>
<p><strong>Publications:</strong>
   <?php
      if ( isset($_POST['publications']) ) {
         echo sanitizeInput( implode(', ', $_POST['publications']) );
      }
   ?>
</p>
<p><strong>How often track:</strong>
   <?php echo sanitizeInput($_POST['howoftentrack']); ?></p>
<p><strong>Comments:</strong></p>
<p><?php echo nl2br( sanitizeInput($_POST['comments']) ); ?></p>
```

21. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/form-basic/form.php**
    - Windows: **localhost/phpclass/form-basic/form.php**
22. Test everything out and all the form inputs should now be securely output on the action page.
23. Switch back to your code editor.
24. Close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

One of the great things about PHP is how easy it is to send an email. However, setting up a mail server in a testing environment is not always so straightforward. In this exercise we'll show you how to set up MAMP Pro or XAMPP to send test emails, as well as how to actually send the email with PHP.

Getting Started with MailHog

MailHog is a mail catcher, which means it catches any mail sent to it, and displays it.

If you are in a Noble Desktop instructor-led class, we have already installed and configured MailHog for you. If you're going through this workbook on your own at your office or home, install MailHog using the sidebar for your operating system.

Installing MailHog on Mac

1. Open Terminal.
2. Install the Xcode command line developer tools by typing the following command and hitting Return:

   ```
   xcode-select --install
   ```

3. If you get a pop-up asking if you would like to install the tools now, click **Install**.
4. Click **Agree** to the terms of use. It may take a few minutes to install.
5. Next we need to install Homebrew, a command line tool that makes it easy to install popular software packages (like MailHog) on macOS. In your web browser, go to **http://brew.sh**
6. Copy the command prompt under **Install Homebrew**.
7. Paste the command into Terminal.
8. Hit Return and follow the prompts.
9. If you get a message that Homebrew is already installed, type the following command and hit Return:

   ```
   brew update
   ```

10. Install MailHog by typing the following prompt and hitting Return:

    ```
    brew install mailhog
    ```

Installing MailHog on Windows

1. In your browser go to **github.com/mailhog/MailHog**
2. Scroll down to **Getting started**.
3. Click on **Download the latest release**.
4. Click on the latest release number. As of the writing of this book, the newest release is v0.1.8.
5. Under **Downloads** click **MailHog_windows_amd64.exe** (64-bit system) or **MailHog_windows_386.exe** (32-bit system).
6. Save the file in your Program Files in a new folder called MailHog.

Running MailHog on Mac

If you're a Windows user, skip to the next section.

1. Open Terminal.
2. Type the following command:

```
mailhog start
```

3. Hit Return.
4. Leave Terminal open in the background.

Running MailHog on Windows

1. Navigate to **Program Files > MailHog** and find the .exe file that starts with **MailHog**.
2. Double–click it and hit **Run** to open it in the Command Prompt.
3. If you get an alert that Windows Firewall has blocked access to some features of this app, click **Allow access**.
4. Leave Command Prompt open in the background.

Setting Up Email: MAMP PRO

If you're a Windows user, skip to the next section.

1. Switch to **MAMP PRO**.
2. Click the **Postfix** tab.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Next to **Set domain of outgoing e-mails to**, type any domain you like, as long as it's syntactically valid. We'll use nobledesktop.com.
4. Check the box next to **Use a Smart host for routing**.
5. In the Server name field, type: **127.0.0.1:1025**
6. Click the **Use this data** button to apply it.

Setting Up Email: XAMPP

Before we can use MailHog, we'll have to edit two configuration files: **php.ini** and **sendmail.ini**

1. Go to the **XAMPP** control panel.
2. Next to **Apache**, click the **Config** button and from the menu choose **PHP (php.ini)**.
3. This will open up **php.ini** in your default text editor.
4. Press **Ctrl–F** to open up the **Find** window.
5. Next to **Find what**, type **sendmail** and click **Find Next**.
6. You should be taken to the following lines of code:

```
; For Win32 only.
; http://php.net/sendmail-from
;sendmail_from = postmaster@localhost
```

7. Scroll down a few lines until you see the following code:

```
; XAMPP: Comment out this if you want to work with fakemail for forwarding to
your mailbox (sendmail.exe in the sendmail folder)
;sendmail_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"
```

8. By default XAMPP writes our emails to the disk instead of sending out the email. Let's change that. As shown below, delete the **semi-colon (;)** before the **sendmail** line and add a **semi-colon (;)** before the **mailtodisk** line.

```
; XAMPP: Comment out this if you want to work with fakemail for forwarding to
your mailbox (sendmail.exe in the sendmail folder)
sendmail_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"

; XAMPP: Comment out this if you want to work with mailToDisk, It writes all
mails in the C:\xampp\mailoutput folder
;sendmail_path = "C:\xampp\mailtodisk\mailtodisk.exe"
```

9. Save the file.
10. Close the file. You should be back in the XAMPP Control Panel.
11. Switch back to your code editor.

12. Go to **File > Open File**.
13. Navigate to the **C: > xampp > sendmail** folder.
14. We need to edit **sendmail.ini**. If you see the file, double–click it to open it.

    If instead you see two sendmail files without extensions, find the **File name** field at the bottom of the Open window, type **sendmail.ini** then click **Open**.
15. Around line 14, find the following lines of code:

```
smtp_server=mail.mydomain.com

; smtp port (normally 25)

smtp_port=25
```

16. Edit the code as shown below:

```
smtp_server=localhost

; smtp port (normally 25)

smtp_port=1025
```

17. Save the file.
18. Close the file.
19. Switch back to the **XAMPP** control panel.

    We need to restart the Apache server for the change to take effect. If Apache is installed as a service (there is a green checkmark to the left of Apache), you will also need to restart your computer.
20. To the right of **Apache** click the **Stop** button.
21. To the right of **Apache** click the **Start** button.

    NOTE: If Apache does not start up (or it starts then immediately stops), you will need to restart your computer. After you restart your computer, relaunch XAMPP. The servers should start up without a problem.

Sending an Email

Now that we have a test mail server set up, we can actually go about sending an email. Sending a simple mail message with PHP is very easy: just use the **mail()** function.

The **mail()** function is formatted like this:

```
mail(to, subject, message, headers, additional parameters)
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

1. In your code editor, open **mail.php** from the **phpclass** folder.
2. Add the following at the top of the document:

```
<?php mail(
      "youremail@gmail.com",
      "Hello World",
      "Hello, this is a test msg!",
      "From:youremail@gmail.com\r\n"
   );
?>
```

   Note also the **\r\n** after the From address. This is a carriage return and a line feed required by many email servers. Your email may not get sent if you leave it out.

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/mail.php**
   - Windows: **localhost/phpclass/mail.php**

   If you pull up Terminal or Command Prompt, you can see the status of your email being sent.

4. Time to check our email! In your browser, go to **localhost:8025**
5. The MailHog interface will open, showing an inbox. You should see your new email listed in the inbox.
6. Close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

When dealing with user input it is vital that we error-check that input and display any errors to the user. You've seen it many times before in any form you've filled out: perhaps you've forgotten to enter your phone number, or you got your password wrong, etc. There are two ways of doing this error checking: on the client-side and on the server-side. While it is best practice to always do both, having at least server-side is critical. Client-side validation typically uses JavaScript or a JavaScript library such as jQuery, but it cannot be relied on because a small percentage of users will have JavaScript turned off or unavailable, or worse yet a hacker may be trying to inject faulty information to your system and will purposefully bypass your client-side security.

This exercise will focus on a very simple form validation script that checks for simple errors, sanitizes input, and sends an email. There are many solutions to this common problem, but here we'll focus on the basics so you can understand how the more robust systems are put together and perhaps adapt them to your own projects.

Getting Started

1. Open **form.php** from the **form-validate-simple** folder in the **phpclass** folder.
2. In a browser go to:
   - Mac: **localhost:8888/phpclass/form-validate-simple/form.php**
   - Windows: **localhost/phpclass/form-validate-simple/form.php**

   We have a very simple form consisting of name, email, and checkboxes for publications that they read.

   Here's what we'll do with the form:

   - Make the **name** and **email** required.
   - Make sure that the email is valid.
   - Sanitize the inputs.
   - Display errors if there are any.
   - Send an email.
   - Display a confirmation page when the email is sent.

   First we need to add brackets to the **publications** input because PHP needs to see it as an array.
3. Switch back to your code editor.

4. Find the **What do you read?** checkbox fields around line **25**. Add brackets to the **name** as shown in bold below:

```
<label><input name="publications[]" type="checkbox" id="publications_drf"
value="Daily Racing Form"> Daily Racing Form</label>
<label><input name="publications[]" type="checkbox" id="publications_elle"
value="Elle"> Elle</label>
```

5. Now let's set an action page. Around line **13** set **action** to **form-action.php** as shown in bold below:

```
<form action="form-action.php" method="post" name="signup" id="signup">
```

6. Save the page.

7. Create a new page called **form-action.php** and save it into the **form-validate-simple** folder in the **phpclass** folder.

Making a sanitizeInput() Function

We're going to make a `sanitizeInput()` function like before, except we are going to strip out HTML tags instead of just escaping them.

1. In **form-action.php**, add the following code to the page:

```
<?php
   function sanitizeInput() {
   }
?>
```

This is an empty function called `sanitizeInput()`.

2. Let's make it accept a variable called `$myInput`. Add the following bold code:

```
function sanitizeInput($myInput) {
   $myInput = trim($myInput);
   $myInput = strip_tags($myInput);
   return $myInput;
}
```

This takes $myInput and:

- Uses the **`trim()`** function to trim any extra whitespace.
- Uses the **`strip_tags()`** function to strip out any HTML tags.
- **Returns** the modified **`$myInput`** variable.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Set a new empty array called **`$errors`**. If there are any errors on the page we can put them in there. This also makes it easy to check if the page has errors or not. If the array is empty then we can send out our email. Below the `sanitizeInput()` function add the following bold code:

```
function sanitizeInput($myInput) {
   $myInput = trim($myInput);
   $myInput = strip_tags($myInput);
   return $myInput;
}

$errors = array();
```

4. Add the following bold code:

```
$errors = array();

$name = sanitizeInput($_POST['name']);
$email = sanitizeInput($_POST['email']);
```

This sanitizes the name and email inputs, and puts them into new variables that will make it easy to build the mail() script later.

5. That works fine for the text inputs, but the checkbox fields may or may not be returned. Because of this we will first see if `$_POST['publications']` has been set. If it was set we'll `implode()` it, then sanitize the string. To start, add the following bold code:

```
$name = sanitizeInput($_POST['name']);
$email = sanitizeInput($_POST['email']);

if ( isset($_POST['publications']) ) {

}
```

6. Add the following bold code:

```
if ( isset($_POST['publications']) ) {
    $publications = sanitizeInput( implode(', ',$_POST['publications']) );
}
```

This sets a new variable called **$publications**, uses `implode()` to turn the array into a comma-delimited list, then runs the `sanitizeInput()` function.

7. If `$_POST['publications']` has not been set (the user didn't choose any checkbox) then we simply set **$publication** to be an empty string. Add the following bold code:

```
if ( isset($_POST['publications']) ) {
   $publications = sanitizeInput( implode(', ',$_POST['publications']) );
}
else {
   $publications = '';
}
```

Error Checking

Now that all the form fields have been sanitized and assigned to variables, we can get to the error checking.

1. The first thing to do is make sure that users enter a name. If they didn't we'll add an error to the $error array. Below the if/else statement you just entered, add the following bold code:

```
if ( $name  == ''  ) {
   $errors[] = "You must enter a name.";
}
```

2. The same can be done for the email input. Add the following bold code:

```
if ( $email == ''  ) {
   $errors[] = "You must enter an email.";
}
```

It is vital that the user enters a valid email. This is a common problem and luckily there is a common solution. PHP 5 includes the **`filter_var()`** function which can both sanitize and validate a wide range of objects, including email addresses, URLs, integers, and more. For a complete list of filter options, see the PHP manual: **php.net/manual/en/filter.filters.php**

`filter_var()` takes two parameters: the first is the data to check, and the second is the type of check to run. To validate an email it would be formatted as such:

```
filter_var($email, FILTER_VALIDATE_EMAIL)
```

`filter_var()` returns **true** if it is valid, and **false** if it is not. Note that it is a basic formatting check (makes sure there is an @ sign, etc.) and doesn't actually ping the mail server to see if the address exists.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. If the user did enter an email we'll want to check if it is valid. Add the following bold code:

```
if ( $email == ''  ) {
   $errors[] = "You must enter an email.";
}
elseif ( !filter_var($email, FILTER_VALIDATE_EMAIL) ) {
   $errors[] = "That email is not valid.";
}
```

Displaying Errors

If there are errors we need to display them to the user. We've already made a simple error page for you and all we need to do is loop through the $errors array and output the errors.

1. Add the following bold code:

```
elseif ( !filter_var($email, FILTER_VALIDATE_EMAIL) ) {
   $errors[] = "That email is not valid.";
}

if ( empty($errors) ) {
   //send email and display confirm page
}
else {
   //display errors
}
```

This says if the $errors array is empty, send the email and display a thank you page. If it is NOT empty, display the errors.

2. We've made a page called **form-error.php** that will display errors to the user. We can include it in our script for if there is an error. Replace the **//display errors** comment with the following bold code:

```
if ( empty($errors) ) {
   //send email and display confirm page
}
else {
   require_once('form-error.php');
}
```

   The **require_once()** function includes a file into our script. You can think of it like PHP copying and pasting that code for us. This can be very useful if you have a bit of script that you'll use over and over in different parts of your application.

   We could have also used the **include()** function, but **require_once()** provides the following benefits:

   - Only includes a file a single time, in the event the statement gets called more than once in a single request
   - Errors out if the file cannot be found, making it easier to debug if the path is wrong

3. Save the page.

4. Open **form-error.php** from the **form-validate-simple** folder in the **phpclass** folder.

5. Around line **15** find the **Display errors here** comment and replace it with the following bold code:

```
<?php
   foreach($errors as $value) {
      echo '<li>';
      echo $value;
      echo '</li>';
   }
?>
```

   This wraps each error in an `<li>` tag (notice that the PHP code is already in an unordered list).

6. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/form-validate-simple/form.php**
   - Windows: **localhost/phpclass/form-validate-simple/form.php**

7. Try leaving the name blank, or entering an incorrect email. Your errors should be correctly displayed!

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Sending the Email

1. Switch back to your code editor.
2. Switch to **form-action.php**.
3. Replace the **//send email and display confirm page** comment with an include as shown in bold:

```
if ( empty($errors) ) {
    require_once('form-send-email.php');
}
else {
    require_once('form-error.php');
}
```

4. Save the page.
5. Create a new page called **form-send-email.php** and save it in the **form-validate-simple** folder in the phpclass folder.
6. Add the following bold code to the new page (replacing youremail@gmail.com with your actual email address if on Mac):

   - Mac:

```
<?php
   $to = "youremail@gmail.com";
?>
```

   - Windows:

```
<?php
   $to = "newuser@localhost";
?>
```

7. Continue to build up the email:

```
<?php
   $to = "youremail@gmail.com";
   $subject = "You have new mail!";
   $message = "You have a new signup!\r\n\r\n";
   $headers = "From:newuser@localhost\r\n";
   mail($to, $subject, $message, $headers);
?>
```

   The above code builds up a simple email and sends it out.

8. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/form-validate-simple/form.php**
   - Windows: **localhost/phpclass/form-validate-simple/form.php**

9. Fill out and submit the form, and then check your email:
   - Mac: Log into your email account and enjoy your new message! If you don't see it, check your spam folders.
   - Windows: Switch to **Thunderbird** and click **Get Mail**. You'll see your email appear.

   Looking good so far; all we need to do is add the form information to the email.

10. Switch back to your code editor.
11. Make sure you are in **form-send-email.php**.
12. Add the following bold code:

```
<?php
   $to = "youremail@gmail.com";
   $subject = "You have new mail!";
   $message = "You have a new signup!\r\n\r\n";
   $message .= "Name: $name \r\n";
   $message .= "Email: $email \r\n";
   $message .= "Publications: $publications\r\n";
   $headers = "From:newuser@localhost\r\n";
   mail($to, $subject, $message, $headers);
?>
```

   The **.=** concatenates a string to the $message variable. The **\r\n** adds a line break to the email.

13. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/form-validate-simple/form.php**
    - Windows: **localhost/phpclass/form-validate-simple/form.php**
14. Try out the form again, then check your email. You'll see all the form data is now being sent in the email!

Adding a Thank You Page

It would be nice for users to see a thank you page when they have correctly filled out the form, instead of the blank page they see now.

1. Switch back to your code editor.
2. Switch to **form-action.php**.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Find the **`require_once('form-send-email.php')`** and right below it add the following bold code:

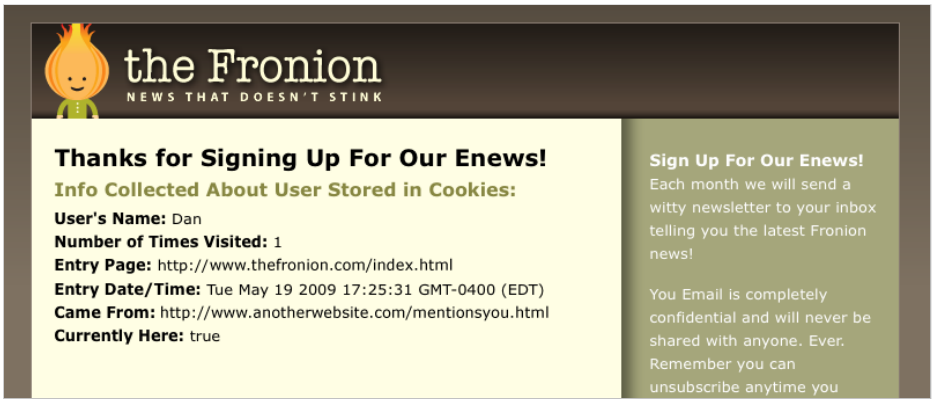
```
if ( empty($errors) ) {
    require_once('form-send-email.php');
    require_once('form-confirm.html');
}
else {
    require_once('form-error.php');
}
```

   We've already made the **form-confirm.html** page so you don't need to do anything else.

4. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/form-validate-simple/form.php**
   - Windows: **localhost/phpclass/form-validate-simple/form.php**
5. Try out the form again, then check your email. Savor the incredible rush of a job well done.
6. Close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Preview



Exercise Overview

Every website should use analytics to track where visitors come from, etc. That anonymous info is useful, but you can take things a step further if you have forms on your website that people fill out. PHP can find information such as their landing page, how many times they've visited, where they came from, etc., and store it in a cookie. Later when that user submits a form, their info is sent along (invisibly to them), offering you more insight into your customers.

Initial Setup Before Moving On

If you are doing this exercise in a classroom (or are doing it for a second time) you must complete the following steps before moving on. If you are doing this exercise at home/work for the first time, you can skip this section.

1. In your browser, go to:
   - Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/setUp.php**
   - Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/setUp.php**
2. This will clear your cookies so you have a fresh start for this exercise.
3. Close the page. You are done!

Coding the Page

1. In your code editor, open **form.php** from the **Tracking-Visitors-With-Cookies** folder in the **phpclass** folder.

   We're going to start out by creating several cookies for user tracking. Let's start by making a counter to track the number of times users have visited our site. We'll start off the counter at 1. We'll also give all these cookies a long expiration time because we never know how long it will be from someone's first visit to the time they submit a form.

   Setting a cookie with PHP is easy; just use the **`setcookie()`** function. **`setcookie()`** takes the following parameters:

   ```
   setcookie(
      'name',
      'value',
      'expire',
      'path',
      'domain',
      'secure',
      'httponly'
   )
   ```

   For this exercise we only need to worry about the name, value, and expiration. Note also that expiration is a value in **seconds** and must be a Unix timestamp. We can feed it a Unix timestamp by using the **time()** function.

2. Let's try it out. At the very top of the page, above the doctype, add the following bold code:

   ```
   <?php
       $expiration = 60*60*24*365*3;
       setcookie("visits",1,time()+$expiration);
   ?>
   <!DOCTYPE HTML>
   ```

   First we set a variable called expiration. Because a cookie's expiration is valued in **seconds**, we need to calculate the number of seconds we would like it to last. In this example, we'll make it last 3 years. We let PHP do the calculation for us, because it is a little easier to update/visualize than writing 94608000 seconds.

   Next, we set a cookie called **visits** with a value of **1** and make it expire to the value given. When you set a cookie expiration, always use: **time()+a number of seconds**

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. To create a cookie that saves the date and time users enter on, we'll use the **`date()`** function. We'd like to output a date that is formatted like so:

   October 21, 2012, 5:15 pm

   The **`date()`** function will allow us to format a date in almost any way. Let's play with it a bit. Add the bold code:

```
<?php
    $expiration = 60*60*24*365*3;
    setcookie("visits",1,time()+$expiration);
    echo date("M j, Y");
    exit();
?>
```

   M outputs a three letter version of the month, such as **Oct**
   j outputs the day of the month without leading zeros, such as **1** or 22
   Y outputs the full four-digit year, such as **2012**

   `exit()`stops the execution of the script. We put this in so we can just focus on this echo for now without seeing the rest of the page.

4. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/form.php**
   - Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/form.php**

   It should read something like: **Aug 23, 2014**

5. Switch back to your code editor.

6. Now let's have it format the date a bit differently. Change the echo statement as shown in bold:

```
setcookie("visits",1,time()+$expiration);
echo date("F j, Y, g:i a");
exit();
```

   Don't worry so much about what each letter means. You can always look up the date() function in the PHP manual: **php.net/manual/en/function.date.php**

7. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/form.php**
   - Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/form.php**

   It should read something like: **August 23, 2014, 5:35 pm**.

8. OK, now that we know how we want to format the date, let's actually add it to a cookie. Switch back to your code editor.

9. Delete the **echo** and **exit** lines and replace them as shown in bold:

```
setcookie("visits",1,time()+$expiration);
setcookie("entryDateTime",date("F j, Y, g:i a"),time()+$expiration);
```

10. Save the page.

11. Now let's write some code on our action page that will display the cookies. Open **thankyou.php** from the **Tracking-Visitors-With-Cookies** folder in the **phpclass** folder.

    This is our pre-made action page. For now we will just display the cookie values.

12. Around line **17** find the comment code that says **Display cookies here**.

13. Delete that comment and add the following bold code:

```
<?php
   echo $_COOKIE['visits'];
   echo '<br>';
   echo $_COOKIE['entryDateTime'];
?>
```

    To access a cookie's value, we use the code: **`$_COOKIE['myCookieName']`**
    You should remember this syntax as it is very similar to working with forms and `$_POST`.

14. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/form.php**
    - Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/form.php**

15. Click **Signup Now!** You don't have to bother filling out the form. If everything works you'll be taken to the thank you page and your cookies will be displayed.

    Next we'll create a cookie that saves the page a visitor enters on. There are many different ways for PHP to get the current page's URL. We've included one popular option that has already been written. While you could write your own method, the nice thing about PHP is that there are so many developers in the world that it is easy to find a robust solution to a common problem.

16. Open **entryURL.php** from the **Tracking-Visitors-With-Cookies** folder in the **phpclass** folder.

17. Select all the code on the page.

18. Copy it.

19. Close the file.

20. Switch to **form.php**.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

21. Just **below** the `$expiration` variable and **above** the first `setcookie()` function, paste the code:

```
$expiration = 60*60*24*365*3;

function currentPageURL() {
   $isHTTPS = (isset($_SERVER["HTTPS"]) && $_SERVER["HTTPS"] == "on");
   $port = (isset($_SERVER["SERVER_PORT"]) && ((!$isHTTPS &&
$_SERVER["SERVER_PORT"]
   $port = ($port) ? ':'.$_SERVER["SERVER_PORT"] : '';
   $url = ($isHTTPS ? 'https://' : 'http://').$_SERVER["SERVER_NAME"].$port.
$_SERVER[
   return $url;
}

$entryURL = currentPageURL();

setcookie("visits",1,time()+$expiration);
setcookie("entryDateTime",date("F j, Y, g:i a"),time()+$expiration);
```

This function uses the **`$_SERVER`** variable to display information about the page. It also takes into account whether the page is currently using a secure server (https) or a normal page (http), gets the port (such as :8888) if there is any, and also gets any URL variables. We run the function then save it into the variable `$entryURL`.

22. Luckily, finding out the referring URL (the page they came from) is a lot easier. We can just use **`$_SERVER['HTTP_REFERER']`**

However, this will not always be set so we have to check whether it is or not before putting it into a cookie. Below the code you just pasted, add the following bold code:

```
$cameFromURL = 'none';
if ( isset($_SERVER['HTTP_REFERER']) ) {
   $cameFromURL = $_SERVER['HTTP_REFERER'];
}
```

This sets a default of **none** for the variable **`$cameFromURL`** in case there is no referrer. If there is a referrer, it will set the variable to the referring page.

23. Now we can set the entry page and the came from cookies. Below the other `setcookie()` functions add the following bold code:

```
setcookie("visits",1,time()+$expiration);
setcookie("entryDateTime",date("F j, Y, g:i a"),time()+$expiration);
setcookie("entryPage",$entryURL,time()+$expiration);
setcookie("cameFrom",$cameFromURL,time()+$expiration);
```

24. Save the page and switch to **thankyou.php**.

25. Let's display these new cookies. Under the PHP code you already have add the following bold code:

```
echo $_COOKIE['visits'];
echo '<br>';
echo $_COOKIE['entryDateTime'];
echo '<br>';
echo $_COOKIE['entryPage'];
echo '<br>';
echo $_COOKIE['cameFrom'];
```

26. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/form.php**
    - Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/form.php**
27. Click **Signup Now!** (Remember, no need to fill out the form). You'll see your cookies displayed on the thank you page.

Tracking the Number of Visits

What we'd like to do is increase the **visits** cookie by 1 every time a user visits the site. We'd also like to only set the **entrypage** and other cookies the very first time the user visits, not every time.

1. Switch back to your code editor.
2. Switch to **form.php**.
3. First, let's make it so that our cookies are only set on the very first visit to the site. We can do this by checking if the **visits** cookie has been set. If it has NOT been set, then it is the first time they are coming to the site and we can set all our cookies. Just below where you set the $expiration var, wrap the rest of the code in an **if** statement as shown in bold (don't forget the closing curly brace!):

```
$expiration = 60*60*24*365*3;

if ( !isset($_COOKIE["visits"]) ) {
   function currentPageURL() {
   $isHTTPS = (isset($_SERVER["HTTPS"]) && $_SERVER["HTTPS"] == "on");
   CODE OMITTED TO SAVE SPACE
   setcookie("entryPage",$entryURL,time()+$expiration);
   setcookie("cameFrom",$cameFromURL,time()+$expiration);
}
```

This says: "If the visits cookie is NOT set, set all the cookies."

4. Save the page.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

5. The visits cookie also needs to increment every time the user visits the site. We can increment the value of v and then resave the cookie using the updated value. Below the **if** statement you just added, enter the following bold code:

```
   setcookie("visits",1,time()+$expiration);
   setcookie("entryDateTime",date("F j, Y, g:i a"),time()+$expiration);
   setcookie("entryPage",$entryURL,time()+$expiration);
   setcookie("cameFrom",$cameFromURL,time()+$expiration);
}
else {
   $v = $_COOKIE["visits"];
   $v++;
   setcookie("visits",$v,time()+$expiration);
}
```

   v++ says to take the value of v and increment it by 1. Then we set the cookie equal to that new value.

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/form.php**
   - Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/form.php**
7. Try it out:
   - Click **Signup Now**.
   - Click the **back** button.
   - **Reload** the page.
   - Click **Signup Now**.
   - Click the **back** button.
   - **Reload** the page.
   - Click **Signup Now**.
   - Notice the **Number of Times Visited** just keeps going up and up!
8. Switch back to your code editor.

9. The **visits** cookie should increase every time the user visits the site—but not every time they load a page! We want the visits to increase once per session. (A session ends when the user quits the browser.) This will give a more accurate number for how many times the user has visited your site over time. To fix this we will add a cookie called **currentlyHere**. We'll use that to see whether the user is on the site. By not specifying an expiration, it will expire at the end of the session. Add the following bold code:

```
   setcookie("visits",1,time()+$expiration);
   setcookie("entryDateTime",date("F j, Y, g:i a"),time()+$expiration);
   setcookie("entryPage",$entryURL,time()+$expiration);
   setcookie("cameFrom",$cameFromURL,time()+$expiration);
}
else {
   $v = $_COOKIE["visits"];
   $v++;
   setcookie("visits",$v,time()+$expiration);
}
setcookie("currentlyHere","true");
```

10. Now let's check to see if that **currentlyHere** cookie exists. If it does not (meaning they haven't yet visited the page in this session), we will increase the visits. Change the **else** statement to an **elseif** as shown in bold:

```
   setcookie("visits",1,time()+$expiration);
   setcookie("entryDateTime",date("F j, Y, g:i a"),time()+$expiration);
   setcookie("entryPage",$entryURL,time()+$expiration);
   setcookie("cameFrom",$cameFromURL,time()+$expiration);
}
elseif ( !isset($_COOKIE["currentlyHere"]) ) {
   $v = $_COOKIE["visits"];
   $v++;
   setcookie("visits",$v,time()+$expiration);
}
setcookie("currentlyHere","true");
```

If the **currentlyHere** cookie is NOT set, then increase the visits counter by 1. This will only happen at the start of the session. For the rest of the session the **currentlyHere** cookie will already be set so the counter will not increase.

11. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/form.php**
- Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/form.php**

12. Try it out:

- Click **Signup Now**, then click the back button, and **Reload** a few times.
- Click **Signup Now**. Notice the **Number of Times Visited** doesn't go up. Good.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Sending an Email with the Cookie Info

We've already made a form validation and email script for you, so all we need to do is add the cookies to the **`mail()`** function. We need to change the form's action page from the thank you page to the page that will process our form.

1. Switch back to your code editor.
2. Around line **45** change the action as shown in bold:

   ```
   <form action="form-action.php" method="post" name="signup" id="signup">
   ```

   This links to the page **form-action.php** that we've made for you. When the form is successfully completed, it will then attempt to mail out the form. The page that does this is called **form-send-email.php**.

3. Save the page.
4. Open **form-send-email.php** from the **Tracking-Visitors-With-Cookies** folder. We have to do a little setting up on this page to make sure it is emailing to the correct address.
5. On line **2**, change the **`$to`** variable:

   - Mac:

   ```
   $to = "youremail@gmail.com";
   ```

   - Windows:

   ```
   $to = "newuser@localhost";
   ```

   Now we can actually add the cookies to the email. Let's think about what we want to do. Obviously we need to add each cookie to the **`$message`** variable. But what if the cookies weren't set (the user might have cookies disabled in their browser)? This would cause an error and the email might not get sent. So we must check whether the cookie is set before we reference it. Rather than repeating the **if isset** code over and over again we'll just loop through a list of cookies and see if each one has been set.

6. Around line **15** find the comment:

   ```
   //add cookies here
   ```

7. **Below** that line add:

   ```
   $cookies = array('visits','entryDateTime','entryPage','cameFrom');
   ```

   This is simply an array of all the cookies we want to check for. We can loop through this and save ourselves some coding time.

8. **Below** that add:

```
$cookies = array('visits','entryDateTime','entryPage','cameFrom');
foreach ( $cookies as $value ) {

}
```

This will loop through our array. **`$value`** will be the **name** of each cookie.

9. Now we want to check if each cookie has been set. Add the following bold code:

```
$cookies = array('visits','entryDateTime','entryPage','cameFrom');
foreach ( $cookies as $value ) {
   if ( isset($_COOKIE[$value]) ) {

   }
}
```

10. Lastly, we want to append the cookie value to the message. Add the following bold code:

```
$cookies = array('visits','entryDateTime','entryPage','cameFrom');
foreach ( $cookies as $value ) {
   if ( isset($_COOKIE[$value]) ) {
      $message .= "$value: $_COOKIE[$value]\r\n";
   }
}
```

Remember the **.=** will **append** our string to the **`$message`** variable.

11. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/Tracking-Visitors-With-Cookies/form.php**
    - Windows: **localhost/phpclass/Tracking-Visitors-With-Cookies/form.php**
12. Fill out and submit the form. Check your email, and you'll see the cookie values!
13. Switch back to your code editor.
14. Close any open files. We're done for now.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

The WWW is a stateless medium, meaning that one page does not "remember" or have any knowledge of another page. A variable you create on one page will not exist if you go to another page. The way around this problem is to use session variables. When you create a session variable it will be remembered across your application but only for a single user. This is useful in many situations, such as logging in a user and remembering them for that session (typically until they log out or quit the browser). Sessions can also be used to store shopping cart information, for example.

In this exercise we will create a simple login/logout application where we make a few pages password-protected.

Getting Started

1. Open **index.php** from the **session-start** folder in the **phpclass** folder.
2. In a browser go to:
   - Mac: **localhost:8888/phpclass/session-start/index.php**
   - Windows: **localhost/phpclass/session-start/index.php**
3. Click around and explore the site. We have a fake little news site, with a login form that doesn't yet do anything. What we'll do is password-protect all the pages except the index page and make it so a user can log in and log out.

Password-Protecting Some Pages

The first thing to do is make an include that will check to see if the session has started, and if the user is logged in. If the user is NOT logged in we will display the login page.

1. Create a new page called **passwordProtection.php** and save it into the **session-start** folder in the **phpclass** folder.
2. In order to access session variables you must use the function **`session_start()`**. It is very important to note a few things about **`session_start()`**:
   - It must be the very first thing at the top of the page before any other output has been sent to the browser. Otherwise you will get an error that headers have already been sent.
   - You can only run the function once per page.

3. In **passwordProtection.php** add the following bold code:

```
<?php

   session_start();

?>
```

Easy, this starts the session!

4. Next we'll check to see if the user is logged in. For our application we'll make a session variable called **loggedIN**. If **loggedIN** is set to **true** then the user is logged in. If it is **false**, or does not exist at all, then they are NOT logged in. Add the following bold code:

```
<?php

   session_start();

   if ( ! isset($_SESSION['loggedIN']) ) {

     header('Location: login.php');
     exit();

  }

?>
```

This says that if the session variable **loggedIN** is NOT set, switch to the login page and stop processing the script.

The **`header()`** function sends raw header information directly to the browser. If we include "Location:" in the `header()` it will relocate to whatever page we tell it to.

5. Save the page.

6. Now we'll add it to a page and try it out. Open **entertainment.php** from the **session-start** folder.

7. At the very top of the page make a new line and add the following bold code:

```
<?php require_once('passwordProtection.php'); ?>
```

This includes the page we just made.

8. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/session-start/entertainment.php**
- Windows: **localhost/phpclass/session-start/entertainment.php**

Instead of seeing the content you'll be redirected to the login page.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

9. Let's lock down all of our other pages (except for index.php). Switch back to your code editor.
10. In **entertainment.php** select:

    ```
    <?php require_once('passwordProtection.php'); ?>
    ```

11. Copy it.
12. Paste it at the **top** of the following pages (all found in the session-start folder). Make sure to save each of the files:
    - **international.php**
    - **local.php**
    - **national.php**
    - **scitech.php**
13. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/session-start/entertainment.php**
    - Windows: **localhost/phpclass/session-start/entertainment.php**
14. Click around to the different pages. You'll see that everything except the index.php page is password locked (click the logo to go to index.php).
15. Switch back to your code editor.
16. Switch to **index.php**. We would still like the index page to be aware of our session, but we don't want it to be password locked. To make it aware of the session, we just need to add **`session_start()`** at the top of the page.
17. At the top of the page, add the following bold code:

    ```
    <?php session_start(); ?>
    ```

18. Save the file.

    report-story.php should also be aware of the session, but not password protected.

19. Open **report-story.php** from the **session-start** folder and add the same **`session_start()`** line.

    This way all of our pages are either password protected, or at least aware of the session.

20. Save the file.

Logging a User In

Now that we've password protected our pages, we need a way for the user to actually log in!

1. Open **login.php** from the **session-start** folder.
2. In a browser go to:
   - Mac: **localhost:8888/phpclass/session-start/login.php**
   - Windows: **localhost/phpclass/session-start/login.php**

   We have a basic login page, but it doesn't yet do anything.
3. Switch back to your code editor.
4. Add the following at the **top** of the login.php page:

```
<?php

    session_start();

?>
```

   This starts the session for the page.
5. Also, we need to make sure the form action is pointed to itself (so that the page self-submits). Around line **25** find the **action** attribute in the form tag, and add the following bold code:

```
<form action="login.php" method="post" name="submitStory" id="submitStory">
```

   Next we want to process the login information if the user has submitted the form. We only want to process the login if both the username and password are passed along (they always will be if the user submits the form, but specifically requiring both is a little more secure).

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

6. At the top of the page underneath the `session_start()` add the following bold code:

```
<?php

   session_start();

   if ( isset($_POST['username']) && isset($_POST['password']) ) {

   }

?>
```

This says, if the username `$_POST` variable is set AND if the password `$_POST` variable is set, do something.

&& means "and," so both parameters in the if statement must be true.

If both values are submitted we must next check to see if they are the correct username and password. Normally we would check this information against a database, but for now we'll just keep things simple and have the username and password directly on the page.

7. **Inside** the if statement add another if statement:

```
<?php

   session_start();

   if ( isset($_POST['username']) && isset($_POST['password']) ) {

      if ($_POST['username'] == 'noble' && $_POST['password'] == 'noble') {

      }

   }

?>
```

This says: If the username is **noble** AND if the password is also **noble**, then we can log in the user.

8. Let's actually set the user to log in. In the if statement you just wrote add:

```
if ($_POST['username'] == 'noble' && $_POST['password'] == 'noble') {
   $_SESSION['loggedIN'] = true;
}
```

This sets the user to be "logged in."

9. Save the file.

10. We now have enough to actually try it out. In a browser go to:
    - Mac: **localhost:8888/phpclass/session-start/login.php**
    - Windows: **localhost/phpclass/session-start/login.php**
11. Enter **noble** for the username and password.

    For now you're still being directed to the login form so it won't look like you've been logged in, but in fact you have. To prove it, try navigating to one of the pages and you'll be allowed in!

    Next we should make this a little more friendly for the user. One thing we can do is redirect them to the index page instead of the login screen.
12. Switch back to your code editor.
13. In **login.php**, underneath where you set the **`$_SESSION['loggedIN']`** to true, add the following bold code:

```
if ($_POST['password'] == 'noble' && $_POST['username'] == 'noble') {

    $_SESSION['loggedIN'] = true;

    header('Location: index.php');

    exit();

}
```

    The `header()` function redirects us to the index page, and the `exit()` function makes sure the current page stops processing.
14. Save the page.
15. Let's try this out. First we need to log out but because we haven't written a log out function yet we can just end the session. The easiest way to do that is to just quit your browser.
16. Switch to your browser.
17. Quit your browser.
18. Relaunch your browser and go to:
    - Mac: **localhost:8888/phpclass/session-start/entertainment.php**
    - Windows: **localhost/phpclass/session-start/entertainment.php**

    You should be redirected to the login screen. If you are still logged in, try quitting the browser and previewing again, or launch a different browser (sometimes browsers do not end the session when they quit, although they should).

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

19. Now log in again (with the **noble** username and password) and you will be redirected to **index.php**.

    Hmmm, wouldn't it be nice if the user was redirected to the page they clicked on instead of going to the index page every time? To do this we can find the current page's URL using the same function we used in the cookies exercise and store it in a session variable to tell the login script where to redirect.

20. Switch back to your code editor.

21. Open **landingURL.php** from the **session-start** folder.

    This page contains code to get the current page's URL. We just need to save it into a session variable.

22. Add the following bold code:

```
<?php

   function currentPageURL() {
      $isHTTPS = (isset($_SERVER["HTTPS"]) && $_SERVER["HTTPS"] == "on");
```

CODE OMITTED TO SAVE SPACE

```
      return $url;
   }

   $_SESSION['landingURL'] = currentPageURL();

?>
```

    This saves the output of **`currentPageURL()`** into the session variable **`$_SESSION['landingURL']`**. This will allow us to use the variable again on the next page.

23. Save the page.

24. Now we have to include this file across our site. Open **passwordProtection.php** from the **session-start** folder.

25. Save the file.

26. Add the following bold code:

```
<?php

   session_start();

   if ( ! isset($_SESSION['loggedIN']) ) {

      require_once('landingURL.php');

      header('Location: login.php');
      exit();

   }

?>
```

This way if they are not logged in it will get their current page URL.

27. We should also include it on the **index.php** page (because it is not password protected). Switch to **index.php**.

28. Add the following bold code:

```
<?php

   session_start();

   require_once('landingURL.php');

?>
```

Now this page will register the current page URL, too.

29. Save the file.

30. Switch to **login.php**.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

31. Instead of redirecting to the index.php page, we will redirect to the session variable we set earlier. Around line **11** find **`header('Location: index.php')`** and replace it with the following bold code:

```
$_SESSION['loggedIN'] = true;

if ( isset($_SESSION['landingURL']) ) {
   header('Location: ' . $_SESSION['landingURL']);
}
else {
   header('Location: index.php');
}

exit();
```

This way they are directed to the current page URL if it is defined, or directed to the index.php if it is not defined.

32. Save the page.
33. OK, let's try this out. Switch to your browser.
34. Quit your browser to end the current session.
35. Relaunch your browser and go to:
    - Mac: **localhost:8888/phpclass/session-start/national.php**
    - Windows: **localhost/phpclass/session-start/national.php**

    You should be taken to the login screen.
36. Log in using **noble** as both username and password and you should be taken to **national.php**.

Setting Up a Login Error Message

What if the user types in the password incorrectly? Right now they are just taken back to the login screen. We should display a message saying that the username or password they typed in is incorrect.

1. Switch to your code editor.
2. Switch to **login.php** if you aren't there already. Look around line **52** and you'll see that we've already set up a bit of PHP that will display a variable **`$errorDisplay`** if it exists. All we need to do is set this variable if the user is NOT logged in correctly.

3. Around line **19** find the **first** closing bracket and add an empty **else** statement. It should look as shown in bold below:

```
session_start();

if ( isset($_POST['username']) && isset($_POST['password']) ) {
   if ($_POST['password'] == 'noble' && $_POST['username'] == 'noble') {
      $_SESSION['loggedIN'] = true;
      if ( isset($_SESSION['landingURL']) ) {
         header('Location: ' . $_SESSION['landingURL']);
      }
      else {
         header('Location: index.php');
      }
      exit();
   }
   else {

   }
}
```

   Be sure the else statement is in this location! It will be triggered if the username or password is incorrect.

4. In the else statement add the following bold code:

```
else {
   $errorDisplay = '<p class="error">I\'m sorry, that username or password is
incorrect.</p>';
}
```

   This adds a bit of HTML code to the **`$errorDisplay`** variable. Notice also that we escape the single quote in **I'm** with a **`\'`**

5. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/session-start/login.php**
   - Windows: **localhost/phpclass/session-start/login.php**
6. Purposefully enter the wrong username or password. You should see your error message.
7. Enter in the correct username or password. You'll be logged in.

Displaying a Log In or Log Out Link

Now that we can log in we need a way to log out. The navbar should also indicate that the user is logged in. Currently the navbar has a link that says **Log In**. When the user is logged in this should switch to **Log Out**. Let's set this aspect up first.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

1. Open **nav.php** from the **inc** folder in the **session-start** folder.

   This page is an include that displays the navigation for the site. We will write a simple if/else statement that tests whether the user is logged in. If the user is NOT logged in, the nav will display a link that allows them to log in. If the user is already logged in, then nav will display a link that lets them log out.

   Because this page already has a lot of HTML code, we'll use an alternative syntax for the if/else statement that makes it easier to mix HTML and PHP.

2. Around line **8** find the `<a>` that links to **login.php**.

3. Add some returns so the entire `<a>` tag is on its own line with extra spaces around it, separate from the `<li>` as shown below:

```
<li id="login">

    <a href="login.php">Log In</a>

</li>
```

4. Next we want to wrap the if/else statement around this. First we'll put in an empty if/else and then add the logic to the if. Add the following bold code:

```
<li id="login">
   <?php if ():?>
      <a href="login.php">Log In</a>
   <?php else:?>
   <?php endif?>
</li>
```

   This alternative syntax makes it easy to mix HTML with PHP code. No more endless echo statements!

5. In between the else and endif add a link to the logout page:

```
<li id="login">
   <?php if ():?>
      <a href="login.php">Log In</a>
   <?php else:?>
      <a href="logout.php">Log Out</a>
   <?php endif?>
</li>
```

6. The logic for the if statement still needs to be written. It should be if the **$_SESSION['loggedIN']** variable is NOT set, or if it is set to false, then display the login page. In the parentheses of the if statement, add the bold code:

```
<?php if ( !isset($_SESSION['loggedIN']) || $_SESSION['loggedIN'] == false ):?>
   <a href="login.php">Log In</a>
 <?php else:?>
   <a href="logout.php">Log Out</a>
 <?php endif?>
```

7. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/session-start/national.php**
   - Windows: **localhost/phpclass/session-start/national.php**
8. If you are logged in, there should be a link to log out.
9. Quit the browser.
10. Relaunch your browser and go to:
    - Mac: **localhost:8888/phpclass/session-start/national.php**
    - Windows: **localhost/phpclass/session-start/national.php**

    You should see a link to log in.

Making a Log Out Function

Finally we can write a function to log out the user. We've pre-made a page that will display a message to the user saying they are logged out, but currently it does not actually log the user out. Let's write the logic that actually logs them out.

1. Switch back to your code editor.
2. Open **logout.php** from the **session-start** folder.
3. At the top of the page add:

```
<?php

   session_start();

   $_SESSION['loggedIN'] = false;

?>
```

Technically this is all you need to log out the user (remember we have to start the session any time we want to use session variables)! However, for safety's sake, it is best to set the session array to an empty array and then destroy the session.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

4. Add the following bold code:

```
<?php

   session_start();

   $_SESSION['loggedIN'] = false;

   $_SESSION = array();

   session_destroy();

?>
```

   This is a good security precaution to take whenever ending a session. In this case we don't have any valuable information stored in the session, but potentially the session could contain a user's contact or credit card info.

5. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/session-start/logout.php**
   - Windows: **localhost/phpclass/session-start/logout.php**
6. You'll be logged out. Click around and try logging in/out. Everything should work fine. Yippie!
7. Switch back to your code editor.
8. Close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Uploading files in PHP is a fairly easy process; however, it is also a dangerous one. You should think carefully before allowing your users to upload files, and you should be just as careful about who you let upload files to your server as you would about letting someone write files to your personal computer.

In this exercise we will show you some very basic security measures as well as how to upload files. It goes without saying that if you put this type of capability on your live site, it should be in a password-protected area or authenticated in some way. You should also be sure your server has some sort of virus protection.

In addition to basic computer security issues, there are other legal issues you should consider—for example, if you are allowing users to upload images to your site. Do the users own the content? Is it legal?

With these considerations in mind, let's upload some files.

Making a File Upload Form

1. In your code editor, open **upload.php** from the **phpclass** folder.

   Let's add a simple form. We'll set the action to post to itself. Because we are going to upload files, we must also give it the **enctype** of **"multipart/form-data"**.

2. In between the **body** tags, add the following bold code:

```
<form method="post" action="upload.php" enctype="multipart/form-data">

</form>
```

3. Next we need an input with the type of **file**. Add the following bold code:

```
<form method="post" action="upload.php" enctype="multipart/form-data">
   <p>
      <label for="myFile">Please choose a jpg, gif, or png.</label>
      <input type="file" name="myFile" id="myFile">
   </p>
</form>
```

   type="file" tells the browser to allow the user to choose a file for this input.

4. Finally, let's add a submit button. Add the following bold code:

```
<form method="post" action="upload.php" enctype="multipart/form-data">
   <p>
      <label for="myFile">Please choose a jpg, gif, or png.</label>
      <input type="file" name="myFile" id="myFile">
   </p>
   <p>
      <input type="submit" name="upload" id="upload" value="Upload">
   </p>
</form>
```

The $_FILES Array

We now have a very basic form with one input. Unlike other form elements such as text inputs, radio buttons, or checkboxes which all get stored in the **$_POST** array, any file form elements get put into the **$_FILES** array.

Let's check out this array to see what's going on.

1. Let's print out the array if the form has been submitted. At the top of the document, add the following bold code:

```
<?php

   if (isset($_POST['upload'])) {

      print_r($_FILES);

   }

?>
```

This will print out the **$_FILES** array if the form has been submitted.

2. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/upload.php**
   - Windows: **localhost/phpclass/upload.php**
3. Click the button to choose a file, then navigate to:
   - Mac: **Hard Drive > Applications > MAMP > htdocs > phpclass > files**
   - Windows: **C: > xampp > htdocs > phpclass > files**
4. Double–click **bird.jpg**.
5. Click **Upload**.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

6. You'll see the **$_FILES** array printed on screen (formatted nicely for you below), such as:

```
[myFile] => Array
    (
        [name] => bird.jpg
        [type] => image/jpeg
        [tmp_name] => /Applications/MAMP/tmp/php/phpcJLXml
        [error] => 0
        [size] => 31553
    )
```

First, everything is stored in an array called **myFile**. This name comes from the form input we set up earlier.

The **myFile** array contains a sub-array that has the following elements:

- **name**: The name of the file.
- **type**: The type of file.
- **tmp_name**: The temporary name of the file. When PHP uploads the file, it stores it in a temporary directory. If you don't do anything with the file or reject the download, PHP will delete the temporary file. This is for security.
- **error**: An error, if there was any.
- **size**: The size of the file in bytes.

Some Basic Error Checking

1. In a browser go to:
   - Mac: **localhost:8888/phpclass/upload.php**
   - Windows: **localhost/phpclass/upload.php**
2. This time **don't** choose a file, just click **Upload**.

   You'll see that the **$_FILES** array is still there, but the name value is blank. Next we'll set our script up to check if the name exists and if not, display a message to the user.
3. Switch back to your code editor.

4. Delete the **print_r** statement, then add the following bold code in its place:

```
<?php

   if (isset($_POST['upload'])) {

      if ($_FILES['myFile']['name']) {

         //upload a file

      }
      else {

         $errorMsg = 'You must choose an image.';

      }

   }

?>
```

This means: If the file is chosen, upload it; if not set an error message.

5. Let's display that error message. Right above the **form** code add the following bold code:

```
<body>
<?php

   if (isset($errorMsg)) {

      echo $errorMsg;

   }

?>
   <form method="post" action="upload.php" enctype="multipart/form-data">
```

This will display **$errorMsg** if it is set.

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/upload.php**
   - Windows: **localhost/phpclass/upload.php**
7. Don't choose a file, just click **Upload**. You'll see the error message displayed.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Uploading the File

Let's actually upload a file. First we must choose a destination. For testing purposes, we will create a folder called **upload_test** and put it in our **phpclass** folder. In a live server environment, you can put this wherever is appropriate for your application, or wherever your host will allow. In addition, you may need to set permissions to allow uploads. Contact your host if you are having trouble.

1. First we need to make a folder to upload our files to. Minimize your code editor so you can see the **Desktop**.

   ### Mac

   - Navigate to: **Hard Drive > Applications > MAMP > htdocs > phpclass**
   - Make a new folder called: **upload_test**

   ### Windows

   - Navigate to: **C:/xampp/htdocs/phpclass/**
   - Make a new folder called: **upload_test**

2. Switch back to your code editor.

3. First we'll set a **`$destination`** variable. **Delete** the **upload a file** comment, then add the following bold code in its place (depending on your platform):

```
if ($_FILES['myFile']['name']) {

   Mac Users:
   $destination = '/Applications/MAMP/htdocs/phpclass/upload_test/';

   Windows Users:
   $destination = 'C:xampp/htdocs/phpclass/upload_test/';

}
else {

    $errorMsg = 'You must choose an image.';

}
```

   We'll use this variable (`$destination`) to tell PHP where to put the file.

4. PHP automatically uploads the file, but to actually use it and access it we need to move it to our new directory. We do this with the **`move_uploaded_file()`** function. This function takes two parameters: the name of the file to move (the temp name), and the location to put the file. For instance:

```
move_uploaded_file(filenameToMove, mydrive/directory/myNewFile.jpg)
```

5. Underneath the **$destination** variable, add the following bold code:

```
$destination = '/Applications/MAMP/htdocs/phpclass/upload_test/';
move_uploaded_file(
   $_FILES['myFile']['tmp_name'],
   $destination .$_FILES['myFile']['name']
);
```

   The first parameter here is the temporary name of the file we just chose. The second is the **$destination** path we set, concatenated with the **actual** name of the file we chose.

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/upload.php**
   - Windows: **localhost/phpclass/upload.php**
7. Click the button to choose a file, then navigate to:
   - Mac: **Hard Drive > Applications > MAMP > htdocs > phpclass > files**
   - Windows: **C: > xampp > htdocs > phpclass > files**
8. Double–click **bird.jpg**.
9. Click **Upload**.

   If all goes well, you won't see any errors.

10. Minimize any windows you may have open so you can see the **Desktop**.
    - Mac: Navigate to **Hard Drive > Applications > MAMP > htdocs > phpclass > upload_test**.
    - Windows: Navigate to **C:/xampp/htdocs/phpclass/upload_test**.
11. In the **upload_test** folder, you'll see the file you chose!

Some (Very) Basic Security

That's the basics of uploading a file, but let's make this a tad more secure. First, we should restrict the file type to only allow GIF, JPEG, and PNG. Second we'll set the name of the file to a generic one and set the file extension to an appropriate one as well.

The **$_FILES['myFile']['type']** variable will allow us to check the type of file the user has chosen. We'll use a switch statement to check against our three file types and set a file extension. If the file is not one of the proper file types we'll set the file extension to be blank.

1. Switch back to your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

2. Above the **$destination** variable, add the following bold code:

```
if ($_FILES['myFile']['name']) {

   switch($_FILES['myFile']['type']) {
      case    'image/jpeg':   $ext = 'jpg';    break;
      case    'image/gif':    $ext = 'gif';    break;
      case    'image/png':    $ext = 'png';    break;
      default:                $ext = '';       break;
   }

   $destination = '/Applications/MAMP/htdocs/phpclass/upload_test/';
```

This switches through the file type and sets a variable **$ext** which will become our filename extension.

3. Let's wrap our upload code in an if statement so it only runs if the **$ext** variable is NOT blank (that is, is the correct type). Wrap the bold code around the **$destination** variable and **move_uploaded_file()** function:

```
if ($ext) {

   $destination = '/Applications/MAMP/htdocs/phpclass/upload_test/';
   move_uploaded_file(
      $_FILES['myFile']['tmp_name'],
      $destination .$_FILES['myFile']['name']
   );

}
```

4. While we're at it, let's add an **else** statement that sets an error. Add the following bold code:

```
if ($ext) {

   $destination = '/Applications/MAMP/htdocs/phpclass/upload_test/';
   move_uploaded_file(
      $_FILES['myFile']['tmp_name'],
      $destination .$_FILES['myFile']['name']
   );

}
else {

   $errorMsg = 'That file is not the correct type.';

}
```

If the **$ext** variable is NOT blank, upload the file. Otherwise, set an error message.

5. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/upload.php**
   - Windows: **localhost/phpclass/upload.php**
6. Click the button to choose a file, then navigate to:
   - Mac: **Hard Drive > Applications > MAMP > htdocs > phpclass > files**
   - Windows: **C: > xampp > htdocs > phpclass > files**
7. Double–click **uploadme.txt**.
8. Click **Upload**.

   You should see an error message: **That file is not the correct type.**
9. Hit the back button in the browser and this time choose a JPG, GIF, or PNG from the folder. The file should upload (check for it in the **upload_test** folder).
10. Switch back to your code editor.
11. Finally, let's give the file we are copying a generic name with a proper extension. We'll timestamp the name and add an extension. Just above the **`$destination`** variable, add the following bold code:

```
if ($ext) {

   $newName = "myImage-";
   $newName .= date("Y_n_j-G_i_s.");
   $newName .= $ext;

   $destination = '/Applications/MAMP/htdocs/phpclass/upload_test/';
   move_uploaded_file(
      $_FILES['myFile']['tmp_name'],
      $destination .$_FILES['myFile']['name']
   );

}
```

   This sets a new variable, **`$newName`**, and sets it to be called **MyImage-**. It then concatenates a date and time, and sets the extension to the variable we set earlier.
12. We must now set this as the new name in the **`move_uploaded_file()`** function. Change the **`move_uploaded_file()`** function as shown in **bold**:

```
move_uploaded_file(
   $_FILES['myFile']['tmp_name'],
   $destination .$newName
);
```

   This changes the name from the original filename, to our new filename.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

13. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/upload.php**
    - Windows: **localhost/phpclass/upload.php**
14. Click the button to choose a file, then navigate to:
    - Mac: **Hard Drive > Applications > MAMP > htdocs > phpclass > files**
    - Windows: **C: > xampp > htdocs > phpclass > files**
15. Double–click **bird.jpg**.
16. Click **Upload**.
17. Switch to the **upload_test** folder and see your newly uploaded file. Success!
18. Switch to your code editor.
19. Close any open files. We're done for now.

Bonus (If You Have Time)

The security we have here is decent, but far from foolproof. Depending on the browser, typically MIME-type values are checked just by examining the 3-letter file extension. Obviously this is easy to change! A more robust method to determine MIME-type is to actually examine the file contents. This may sound complicated, but luckily PHP has a built-in function for doing exactly that, called **finfo**. Please note that this extension is only available on PHP 5.3 and higher.

XAMPP has this function turned off by default, so if you are on Windows you'll have to do some additional set. If you are using XAMPP, complete the following sidebar before proceeding.

Xampp Users Only: Activating finfo

1. Switch to the XAMPP Control Panel.
2. To the right of **Apache**, click the **Config** button and from the menu choose **PHP (php.ini)**.
3. This will open up **php.ini** in your default text editor.
4. Press **Ctrl–F** to open up the **Find** window.
5. Next to **Find what**, type **fileinfo** and click **Find Next**.
6. You should be taken to the following line of code:

   ```
   ;extension=php_fileinfo.dll
   ```

7. To turn this feature on, delete the **semi-colon (;)** at the beginning of the line:

   ```
   extension=php_fileinfo.dll
   ```

8. Save the file.
9. Close the file.
10. You should be back in the XAMPP Control Panel. We need to restart the Apache server for the change to take effect. If Apache is installed as a service, you will also need to restart your computer.
11. To the right of **Apache**, click the **Stop** button.
12. To the right of **Apache**, click the **Start** button.
13. If Apache does not start up (or it starts then immediately stops), you will need to restart your computer. After you restart your computer, relaunch XAMPP. The servers should start up without a problem.

1. In your code editor, open **upload.php** from the **phpclass** folder.
2. Just above the **switch** expression, add the following bold code:

   ```
   $file_info = new finfo;

   switch($_FILES['myFile']['type']) {
   ```

   This initializes the **finfo** method and saves into a new object called **$file_info**.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Let's use the new object. Add the following bold code:

```
$file_info = new finfo;
$mime_type = $file_info->buffer();

switch($_FILES['myFile']['type']) {
```

This sets a new variable called `$mime_type`. We also use the `$file_info` object and run the method **`buffer()`** on it. You can think of an object like a little application that you can run functions on. The **->** arrow says to run the function **`buffer()`**.

4. Next we need to specify the parameters for the **`buffer()`** function. We need to send it the actual file as the first parameter, and the second parameter will tell it what info we want to get from the file. In this case, we want to get the mime type. Add the following bold code:

```
$file_info = new finfo;
$mime_type = $file_info->buffer(file_get_contents($_FILES['myFile']
['tmp_name']), FILEINFO_MIME);

switch($_FILES['myFile']['type']) {
```

In order to get the actual file we need to use the **`file_get_contents()`** function. Inside that we send it the temporary file that PHP has created. The second parameter for **`buffer()`** is **FILEINFO_MIME** which tells PHP that we want the MIME type.

5. Let's echo **`$mime_type`** to see what we have so far. Add the following bold code:

```
$file_info = new finfo;
$mime_type = $file_info->buffer(file_get_contents($_FILES['myFile']
['tmp_name']), FILEINFO_MIME);

echo $mime_type;
exit;
```

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/upload.php**
   - Windows: **localhost/phpclass/upload.php**
7. Click the button to choose a file, then navigate to:
   - Mac: **Hard Drive > Applications > MAMP > htdocs > phpclass > files**
   - Windows: **C: > xampp > htdocs > phpclass > files**
8. Double–click **bird.jpg**.

9. Click **Upload**. You'll see:

   image/jpeg; charset=binary

   If you are on Windows, and you see an error that begins with **Fatal error: Class 'finfo' not found in…** make sure to restart your computer after completing the sidebar earlier in the exercise for activating the finfo function in XAMPP.

10. Switch back to your code editor.

11. Delete the echo statement. We're done with it.

12. We want to get only the first part of this string. Our switch statement is expecting just the first part, not the **charset=binary**. One way to get just the first part is to turn the string into an array using the **`explode()`** function. We can turn it into a two-part array with the semi-colon as a delimiter. Add the following bold code:

```
$file_info = new finfo;
$mime_type = $file_info->buffer(file_get_contents($_FILES['myFile']
['tmp_name']), FILEINFO_MIME);

$myMimeArr = explode(';',$mime_type);
exit;
```

   This makes a new array called **`$myMimeArr`**.

13. Let's quickly print **`$myMimeArr`** to see what we have. Add the following bold code:

```
$file_info = new finfo;
$mime_type = $file_info->buffer(file_get_contents($_FILES['myFile']
['tmp_name']), FILEINFO_MIME);

$myMimeArr = explode(';',$mime_type);
print_r($myMimeArr);
exit;
```

14. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/upload.php**
    - Windows: **localhost/phpclass/upload.php**

15. Click the button to choose a file, then navigate to:

    - Mac: **Hard Drive > Applications > MAMP > htdocs > phpclass > files**
    - Windows: **C: > xampp > htdocs > phpclass > files**

16. Double–click **bird.jpg**.

17. Click **Upload**. You'll see:

    Array ([0] => image/jpeg [1] => charset=binary)

    Perfect. Now we can just use the first element in the array.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

18. Switch back to your code editor.
19. Delete the **`print_r`** function and the **exit** statement, too.
20. Edit the switch statement as shown in **bold**:

    ```
    switch($myMimeArr[0]) {
    ```

21. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/upload.php**
    - Windows: **localhost/phpclass/upload.php**
22. Click the button to choose a file, then navigate to:
    - Mac: **Hard Drive > Applications > MAMP > htdocs > phpclass > files**
    - Windows: **C: > xampp > htdocs > phpclass > files**
23. Double–click **uploadme.txt**.
24. Click **Upload**.

    You should see an error message: **That file is not the correct type**.
25. Hit the back button in the browser and this time choose a JPG, GIF, or PNG from the folder. The file should upload (check for it in the **upload_test** folder).
26. Switch back to your code editor, and close any files you may have open.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

One of the biggest reasons to use a language like PHP is to connect to a database, allowing you to get and store data for a huge variety of possible applications such as customer info, store and product info, blogs, forums, etc.

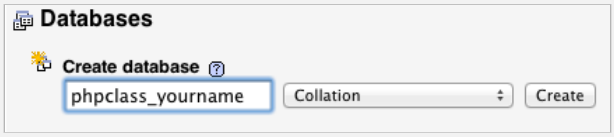
MySQL is one of the world's most popular databases because it is fast, free, stable, and feature-rich. It works great with PHP, and like PHP, it is available on almost every single host. Because of this, PHP and MySQL almost always go hand-in-hand.

In this exercise, you'll learn how to create a database in the phpMyAdmin control panel, as well as how to connect to a database and display some data.

Creating a New Database

Mac

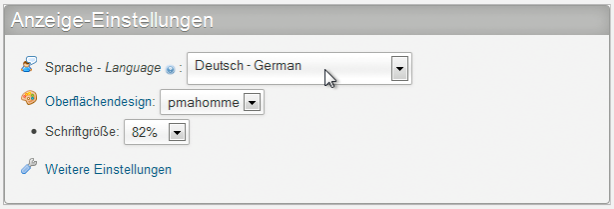
1. Open **MAMP Pro**. (Go to Hard Drive > Applications > MAMP and open MAMP Pro.app.)
2. Click the **WebStart** button.
3. This will open the MAMP start page in your default browser.
4. On the start page at the top, click on **Tools** and choose **phpMyAdmin**.
5. Click the **Databases** tab.
6. As shown below, under the **Create database** field, enter **phpclass_yourname**



7. Click the **Create** button.

Windows

1. If XAMPP is not already running, navigate to **C:/xampp**, then double–click **xampp-control.exe** and start the **Apache** and **MySQL** services.
2. In your browser, go to **http://localhost**
3. On the start page in the **Tools** section, click **phpMyAdmin**.
4. If everything is in German, you may need to switch the language. In the middle of the page, next to **Language** choose **English**.

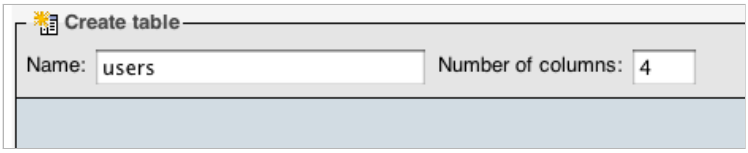


5. Click the **Databases** tab at the top.
6. Under **Create database**, for **Database name**, enter **phpclass_yourname**
7. Click the **Create** button. You'll see the database you just created appear in the list of databases below.

Adding Some Information to the Database

Now that we have a database, we need to make a table to hold some user information. In a staggering leap of creative genius we'll call it… **users**.

1. You should still be in the phpMyAdmin. Click on the **phpclass_yourname** link.
2. In the area that says **Create table**, for **Name** enter **users** and for **Number of columns** enter **4**.



INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. Click **Go**. You'll see an empty table with four blank fields. We're going to create fields for id, firstName, lastName, and email.

   Every new table should have an **id**. The **id** should also be a unique number—this way, it is easy to keep track of each row in the database. This unique identifier is called a **Primary Key**. We'll also set something called **Auto Increment** (A_I) which will automatically increment the **id** by 1 every time a new row is made. This ensures that each **id** number is unique and saves us the trouble of having to increment it manually.

4. Set the following:

Name	Type	Length/Values	Attributes	Index	A_I
id	INT		UNSIGNED	PRIMARY	checked
firstName	VARCHAR	255			
lastName	VARCHAR	255			
email	VARCHAR	255			

   To break this down:

   - **id** will be the name of the identity field.
   - **INT** is an integer with a range from –2147483648 to 2147483647. That's a lot of values!
   - We don't need to set a length for this field (it defaults to 10).
   - **UNSIGNED** sets it so that only positive numbers are allowed. This increases the usable range for our index to 0 to 4294967295. That's even more values! If you think your application will have more than 4 billion users you can use BIGINT instead of INT.
   - **PRIMARY** sets the id column to be the **Primary Key** for this table.
   - **A_I** stands for **Auto Increment** and increments the id by 1 every time a new row is added.
   - **VARCHAR** is a character data type. In our case we set the length to **255** which means we can store up to 255 characters of text.

5. In the bottom right, click **Save**. Your table will be created!

Adding Some Data

1. Click the **users** table.
2. At the top of the page, click the **Insert** tab.

3. In the **top** box enter the following:

Name	Value
firstName	Your First Name
lastName	Your Last Name
email	Your Email

Leave **id** blank because it will auto increment for us.

4. In the **bottom** box enter the following:

Name	Value
firstName	Noble
lastName	Desktop
email	noble@nobledesktop.com

5. Click **Go**. (You can click either Go button.) Two new rows will be created.

Connecting to the Database

There are three main ways PHP can interact with a database: **MySQL** Regular, **MySQL Improved (MySQLi)**, and **PDO**. The first, **MySQL** (regular), is the original language PHP used to interact with MySQL. However, it is no longer being developed and PHP recommends you do not use it anymore. It is important to be aware of it, as there are quite a few outdated examples online that still show it. **MySQLi** is the improved version of the MySQL PHP extension and it is what PHP recommends you use. It can be written in a procedural syntax like the original MySQL extension (in fact the biggest difference will be the "i" at the end of the functions), but can also be written in an object-oriented style. The last, **PDO**, is an extension that is not tied to MySQL specifically but rather is designed to work with any type of database (such as MS SQL or Oracle). If you plan on potentially moving your product to a different database, it is recommended you use PDO.

So which should you use? Do not use the original MySQL extension. Your choice is between MySQLi and PDO. For this book we will show you MySQLi as it is the one recommended by PHP. It is also highly unlikely that any application you develop will actually switch from MySQL to another database—in that rare case, you can learn PDO if you need it. PDO does have some other advanced features and some developers prefer it, but those advantages are beyond the scope of this book. MySQLi has the advantage of also being able to use every new feature of MySQL, whereas PDO can sometimes lag behind in support. Given that MySQLi is the official method and is easier to just learn one at a time, we will show MySQLi. Once you are very comfortable with it, you can learn PDO if you need it.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

1. In your code editor, open **mysql.php** from the **phpclass** folder. Here we have a simple page started for you that has an empty table. We're going to connect to our database and then display the **users** table.

   The first thing to do is establish a connection to the new database. First we create a mysqli object and save it into a variable. The mysqli() function takes a number of parameters, the four most common ones being: **server name**, **mysql username**, **mysql password**, and **database name**. Note that MAMP and XAMPP have different default passwords to connect to the MySQL database.

2. At the top of the document, add the following bold code (Make sure to add the correct line for your operating system.):

```
<?php

   Mac: $conn = new mysqli('localhost', 'root', 'root', 'phpclass_yourname');

   Windows: $conn = new mysqli('localhost', 'root', '', 'phpclass_yourname');

?>
```

   The MySQL server is running on the **localhost**, with the username of **root**, the password is either **root** (Mac) or **blank** (Windows), and we are connecting to the database we just made, **phpclass_yourname**.

3. We need to prepare our SQL statement. SQL is the language that is used to interact with the database. It allows us to view, update, insert, and delete records among other things.

   Add the following bold code:

```
<?php

   $conn = new mysqli('localhost', 'root', 'root', 'phpclass_yourname');
   $sql = 'SELECT * FROM users';

?>
```

   Let's break this down:

   - First we save the string into a variable **`$sql`** (it could be called anything).
   - The string is our SQL statement, and it says to select everything from the table **users**. SELECT does what it sounds like and selects, or "gets," rows from the database.
   - The * says to get **all** of the columns. If you only wanted to get certain columns, such as **id** and **email** you would list them like: `SELECT id, email FROM users`
   - You must always tell SQL which table you want to select records from (in this case it is **users**).

4. The SQL is written, but it still needs to be submitted to the server and the results need to be stored in a variable. Add the following bold code:

```
<?php

   $conn = new mysqli('localhost', 'root', 'root', 'phpclass_yourname');
   $sql = 'SELECT * FROM users';
   $result = $conn->query($sql) or die($conn->error);

?>
```

   - This stores the result in the **`$result`** variable.
   - It submits a query to the **`$conn`** object.
   - If there is an error, you can display it with **`$conn->error`**. In this case, we say **`or die()`** which means that if there is a problem, it will stop the page processing and display an error (on a production site, you would not want to display this error, but you probably want to still `die()` the page).

5. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/mysql.php**
   - Windows: **localhost/phpclass/mysql.php**

   You should see an empty table with some headers and hopefully no error messages.

6. OK, we're getting pretty close. All that has to happen now is to loop through the results. Return to the code, and around line **31** find the empty **`<tr>`**. We will loop through the query results and output a new **`<tr>`** each time. Add the following bold code around the empty **`<tr>`**:

```
<?php while ($row = $result->fetch_assoc()):?>
   <tr>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
   </tr>
<?php endwhile;?>
```

   This uses the alternative syntax of **while** statements, making it easier to mix into the HTML. It fetches the **`$result`** one row at a time. The **`$row`** is an associative array that contains the row's information.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

7. Now we can output each field's data. Add the following bold code:

```
<?php while ($row = $result->fetch_assoc()):?>
   <tr>
      <td><?php echo $row['id']; ?></td>
      <td><?php echo $row['firstName']; ?></td>
      <td><?php echo $row['lastName']; ?></td>
      <td><?php echo $row['email']; ?></td>
   </tr>
<?php endwhile;?>
```

The name of each field is the corresponding name of the array element.

8. Save the page and then in a browser go to:

-Mac: **localhost:8888/phpclass/mysql.php**
-Windows: **localhost/phpclass/mysql.php**

You'll see a list of the two rows of data that you input earlier!

Display the Number of Rows Returned

It would be good to display how many rows were returned. This can be useful for searches, or for logins to make sure there were results found.

1. In between the `<body>` and the `<table>` tag, add the following bold code:

```
<p>
   <?php echo $result->num_rows; ?> rows found.
</p>
```

The magic here is in the **`$result->num_rows;`**. This, as the name suggests, gets the number of rows returned from our query.

2. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/mysql.php**
- Windows: **localhost/phpclass/mysql.php**

You should see the number of rows found.

3. Switch back to your code editor.

4. Leave **mysql.php** open. We'll continue working with it in the next exercise.

It may seem like a lot of work, but you've just learned an immensely powerful tool. In the next few exercises, we'll take it further and learn how to insert, update, and delete rows, as well as use filters to only select certain information.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Because we will be connecting to the same database for the rest of the book, it would make sense to save the connection script into another page. That way, we can just include it at the top of any page that needs to connect to the database—no need to retype it every time.

If you did not do the previous exercise, do it before starting this one.

1. You should still be in **mysql.php**. If you closed the file, reopen **mysql.php** from the **phpclass** folder.
2. At the top of the page, select the connection script that reads:

```
$conn = new mysqli('localhost', 'root', 'root', 'phpclass_yourname');
```

3. Cut it.
4. Create a new file.
5. Paste in the connection code.
6. Wrap it in php tags as shown in bold:

```
<?php

   $conn = new mysqli('localhost', 'root', 'root', 'phpclass_yourname');

?>
```

7. Save the file as **dbConnect.php** into the **inc** folder in the **phpclass** folder.
8. Let's add a bit of error-checking to make the script more robust. We'll check to see if there is a connection error. If there is one, display the error and time out. Add the bold code:

```
$conn = new mysqli('localhost', 'root', 'root', 'phpclass_yourname');

if ($conn->connect_errno) {
   echo "Connection Failed: " . $conn->connect_error;
   exit;
}
```

`$conn->connect_errno` returns the error number of our connection if there is one. If there is an error we should `exit()` out of the page because the rest of our page will likely fail to work without a database connection. We also echo out the specific error.

9. Save the page.
10. Last we need to include this script in our page. Switch back to **mysql.php**.

11. At the top of the page above the other code, add the following bold code:

```
<?php

   require_once('inc/dbConnect.php');
   $sql = 'SELECT * FROM users';
   $result = $conn->query($sql) or die($conn->error);
```

12. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/mysql.php**
    - Windows: **localhost/phpclass/mysql.php**
13. It should work just like it did before. You'll use this same script to connect to the database in later exercises.

Sorting Results

Sometimes when writing a query, you'll want to output the results in a certain order. Luckily this is a fairly easy process.

1. Switch back to your code editor.
2. First let's output the records and order them by **lastName**. Edit the **$sql** string as shown in bold:

```
require_once('inc/dbConnect.php');
$sql = 'SELECT * FROM users ORDER BY lastName';
$result = $conn->query($sql) or die($conn->error);
```

3. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/mysql.php**
    - Windows: **localhost/phpclass/mysql.php**

    The records will now display in alphabetical order by last name.
4. Switch back to your code editor.
5. What if we wanted them in reverse order? Edit the **$sql** string as shown in bold:

```
$sql = 'SELECT * FROM users ORDER BY lastName DESC';
```

6. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/mysql.php**
    - Windows: **localhost/phpclass/mysql.php**

    The records will be in reverse alphabetical order by last name.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

7. You can also combine ORDER BY statements so that it will order by multiple items. For example, say you wanted to order by last name, and then by first name. This way if you had multiple people with the same last name, they would then be ordered by their first names. Edit the **`$sql`** string as shown in **bold**:

   ```
   $sql = 'SELECT * FROM users ORDER BY lastName, firstName';
   ```

8. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/mysql.php**
   - Windows: **localhost/phpclass/mysql.php**

   The records will be in alphabetical order by last name, and then if you have any records with the same last name, you'll see them ordered by first name as well.

9. Close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

In this exercise we are going to show how to select a certain row of data in a database. We'll start by selecting all the records in the database that are equal to a certain email. The same concept could be used to select a particular ID. In addition we'll show how to prevent SQL Injection attacks by using prepared statements.

Getting Started

1. Open **search-results-simple.php** from the **sql-prepared** folder in the **phpclass** folder.

   Take a moment to look over the code. It is the same type of SELECT statement we made in the last exercise. It selects all the rows in the **users** table and then outputs the information.

2. Before we can run the page, we have to include the connection script we made earlier in order to connect to the database (if you did not do the previous exercise do it now). At the top of the page inside the PHP tags, add the following bold code:

```
<?php

   require_once('../inc/dbConnect.php');

   $sql = "SELECT *
           FROM users
           ";
```

3. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/sql-prepared/search-results-simple.php**
   - Windows: **localhost/phpclass/sql-prepared/search-results-simple.php**

   You'll see that the entire table is output.

4. Switch back to your code editor.

5. What if we wanted to only select rows that have a certain email? Change the SQL statement so it reads:

```
$sql = "SELECT *
        FROM users
        WHERE email = 'noble@nobledesktop.com'
        ";
```

   This selects everything from the users table **where** the **email** is **exactly** equal to **noble@nobledesktop.com**.

   Notice that the email address is inside of quotes. This tells SQL that it is searching for a string. If we were selecting the id we wouldn't need to wrap it in quotes (for example: WHERE id = 24).

6. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/sql-prepared/search-results-simple.php**
   - Windows: **localhost/phpclass/sql-prepared/search-results-simple.php**

   You'll see that only one row is output.
7. Switch back to your code editor.
8. What if we wanted to let a user enter this email from a form? We would use the `$_POST` array and select on that. Change the SQL as shown in bold:

```
$sql = "SELECT *
        FROM users
        WHERE email = '$_POST[email]'
        ";
```

   This will now select on whatever the user enters into a form.
9. Save the page.
10. Open **form.php** from the **sql-prepared** folder. This is a form we already made that links to **search-results-simple.php**. Let's preview it in a browser.
11. In a browser go to:
    - Mac: **localhost:8888/phpclass/sql-prepared/form.php**
    - Windows: **localhost/phpclass/sql-prepared/form.php**
12. In the Email field enter: **noble@nobledesktop.com**

    You must enter it exactly.
13. Click **Submit** and you'll see that it returns one row. If you don't type it exactly, it won't find anything (this isn't a search; we'll do that later) because we are looking for a row that has an email **exactly** equal to what we enter.

    Go ahead and try it with your own email that you entered into the database in the last exercise, too.

SQL Injection

We have everything working fine, but there is a problem. Because we are allowing user input directly into the SQL statement, we have opened ourselves up to a huge security risk. Let's check it out.

1. In a browser go to:
   - Mac: **localhost:8888/phpclass/sql-prepared/form.php**
   - Windows: **localhost/phpclass/sql-prepared/form.php**

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

2. In the Email field enter the following **exactly**:

   `' or '1' = '1`

3. Click **Submit**. All the rows in the table will be displayed! What happened? We modified the SQL statement to read:

   ```
   SELECT *
   FROM users
   WHERE email = '' or '1' = '1'
   ```

   This says select everything from the users table where the email is blank **or where 1 equals 1**. Because 1 is always equal to 1, it will return every row in the database! What if this was a login form? With a little bit of knowledge a hacker wouldn't need to guess a password, they would just have to enter code similar to this and potentially gain access to your system!

   So how do we prevent this? There are a few different methods. One of the easiest is to wrap each variable in the function **`real_escape_string()`**. This will escape any quotes and prevent most types of SQL injection attacks. While this method works fine, it is still potentially vulnerable to some attacks (can you really escape *every* possible combination of characters a hacker might use?).

   A better method is to use **prepared statements**. A prepared statement separates the SQL query from the variables so that the server processes them separately. First the general query is sent to the database server, for example:

   ```
   SELECT * FROM users WHERE email = ?
   ```

   The **?** is a placeholder. We then tell the database server that it should expect a string for this placeholder. Then the variable is sent separately. Because the server is sent the SQL statement beforehand it is impossible for a hacker to modify the SQL statement in any way. Any variable that is passed will just be processed as a string.

   Prepared statements have the added benefit of being potentially faster as well. If you are reusing the same query over and over again with only the variables changing, the database server doesn't need to compile the SQL over and over, it just changes the variables it is searching for.

Prepared Statements

First let's set the form action to point to the prepared statement page we are going to make.

1. Switch back to your code editor.

2. In **form.php**, around line **16**, change the action as shown in bold:

   `<form name="form1" id="form1" method="post" action="`**`search-results-prepared.php`**`">`

3. Save the page.
4. Open **search-results-prepared.php** from the **sql-prepared** folder.
5. At the top of the page add the include to connect to the database:

```
<?php

   require_once('../inc/dbConnect.php');

?>
```

6. Next add the SQL statement, but with a **?** placeholder for the variable. Instead of selecting all, we should select only the columns we want. This is a bit more work, but is faster, more secure, and is generally best practice. Add the bold code:

```
require_once('../inc/dbConnect.php');

$sql = "SELECT id, firstName, lastName, email
       FROM users
       WHERE email = ?
       ";
```

7. Now we have to **initialize** the statement:

```
$sql = "SELECT id, firstName, lastName, email
        FROM users
        WHERE email = ?
        ";

$stmt = $conn->stmt_init();
```

This initializes a statement object called **$stmt**. The **$conn** object is the connection that was set up in the include.

8. Then we need to **prepare** the statement:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

9. The next step is to bind the parameters (variables) to the statement. We have to specify the type (string, integer, binary, or double) and the name of the variable. Add the bold code:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('s',$_POST['email']);
```

The **s** stands for string. If we had two string parameters to bind it would look like: `bind_param('ss',$firstVar,$secondVar)`

The different types that bind_param() accepts are:

- **s**: String (any text)
- **i**: Integer (any whole number)
- **d**: Double (floating point number)
- **b**: Binary (Binary data like an image, PDF, or other file)

10. Besides parameters, it's also best to bind the results. This way we can give our output variables nice names like **`$id`**, **`$email`**, etc. Add the bold code:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('s',$_POST['email']);
$stmt->bind_result($id, $firstName, $lastName, $email);
```

These variables must be in the same order as the SELECT statement.

11. Next we have to **execute** the statement. Add the bold code:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('s',$_POST['email']);
$stmt->bind_result($id, $firstName, $lastName, $email);
$stmt->execute();
```

This sends the SQL to the database server to run.

12. It is best to add some error-checking code at this point to make sure things have run correctly. In a production server you may not want to display errors to the end user for security reasons, but for developing an application, having accurate error-reporting is invaluable. Add the bold code:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('s',$_POST['email']);
$stmt->bind_result($id, $firstName, $lastName, $email);
$stmt->execute();
if ($stmt->error){
        echo 'There was an error: '    . $stmt->error;
    }
```

If there is an error, display it. You can also use **`$stmt->errno`** to display the exact error code. Both are useful although a plain English error is a little easier to understand than "error 2013."

13. Last we should store the result. Using **`store_result()`** is optional. The benefit of using it is that we can then access the number of rows like: **`$stmt->num_rows`**. Add the bold code:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('s',$_POST['email']);
$stmt->bind_result($id, $firstName, $lastName, $email);
$stmt->execute();
if ($stmt->error){
        echo 'There was an error: '    . $stmt->error;
    }
$stmt->store_result();
```

14. OK, that's a prepared statement! To loop through and display the results use **`$stmt->fetch()`** to fetch the results. Around line **44**, find the empty `<tr>` tags and wrap the following bold code:

```
<?php while ( $stmt->fetch() ):?>
   <tr>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
   </tr>
<?php endwhile;?>
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

15. Then, to display the results just use the variable we bound earlier. Add the following bold code:

```
<?php while ( $stmt->fetch() ):?>
   <tr>
      <td><?php echo $id; ?></td>
      <td><?php echo $firstName; ?></td>
      <td><?php echo $lastName; ?></td>
      <td><?php echo $email; ?></td>
   </tr>
<?php endwhile;?>
```

16. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/sql-prepared/form.php**
    - Windows: **localhost/phpclass/sql-prepared/form.php**
17. In the Email field enter: **noble@nobledesktop.com**
18. Click **Submit** and it should return one row.

    Now let's double-check to see if this will repel our simple SQL injection attack.
19. Hit the back button.
20. In the Email field enter the following **exactly**:

    `' or '1' = '1`
21. Click **Submit**. No rows will be returned!

    It may seem like a bit of extra work to use prepared statements, but the added security and speed are well worth it.
22. Switch back to your code editor.
23. Lastly, we need to add the number of rows. Make sure you are in **search-results-prepared.php**.
24. Between the **`<body>`** and the **`<table>`** tags, add the following bold code:

```
<body>
<?php echo $stmt->num_rows ?>
<table>
   <tr>
      <td><strong>ID</strong></td>
      <td><strong>First Name</strong></td>
      <td><strong>Last Name</strong></td>
      <td><strong>Email</strong></td>
```

25. Save the file and close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Interacting with a database wouldn't be much use if we couldn't add data to it. Here we'll learn how to insert records into a database using prepared statements.

1. Open **insert-easy.php** from the **phpclass** folder.

   We're going to write a simple **INSERT** statement to add some data to the **users** table. We'll use prepared statements. Although it's not necessary for this simple example because we won't be accepting user input, it's still good practice to get used to writing them.

2. First we need to add the connection script: At the top of the page, add the following bold code:

```
<?php

   require_once('inc/dbConnect.php');

?>
```

3. Next, we'll add the INSERT statement. Add the bold code:

```
require_once('inc/dbConnect.php');

$sql = "INSERT INTO users (firstName, lastName, email)
       VALUES (?, ?, ?)
       ";
```

   This is the basic syntax of a SQL INSERT statement. It says "insert firstName, lastName, and email into the users table, with the following values." Because we are using prepared statements we use **?** placeholders.

4. We have to make variables to hold our firstName, lastName, and email parameters. This is necessary because the bind_param() function we will use later only accepts variables—not strings. Add the bold code:

```
$sql = "INSERT INTO users (firstName, lastName, email)
        VALUES (?, ?, ?)
        ";

$firstName = 'George';
$lastName = 'Washington';
$email = 'george@foundingfathers.gov';
```

5. Next we need to initialize the statement and prepare the SQL. Add the bold code:

```
$firstName = 'George';
$lastName = 'Washington';
$email = 'george@foundingfathers.gov';

$stmt = $conn->stmt_init();
$stmt->prepare($sql);
```

6. Then we bind the parameters. Add the bold code:

```
$firstName = 'George';
$lastName = 'Washington';
$email = 'george@foundingfathers.gov';

$stmt = $conn->stmt_init();
$stmt->prepare($sql);

$stmt->bind_param('sss', $firstName, $lastName, $email);
```

   Remember, the **'sss'** in the first portion of `bind_param()` specifies that each variable is a **s**tring. There are three **s**'s because there are three variables.

7. The statement is all prepped and ready to go. We'll execute it and add some basic error-checking. Add the bold code:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);

$stmt->bind_param('sss', $firstName, $lastName, $email);
$stmt->execute();
if ($stmt->error) {
    echo $stmt->errno . ": " . $stmt->error;
}
```

   This executes the query, then checks if there is an error. If there is an error, it outputs the error number (`$stmt->errno`) and concatenates it with a colon followed by the plain-English error (`$stmt->error`). (If there was an error the output would read something like: 2031: No data supplied for parameters in prepared statement.)

8. Save the page and then in a browser go to:

   - Mac: **localhost:8888/phpclass/insert-easy.php**
   - Windows: **localhost/phpclass/insert-easy.php**

9. If everything works correctly you'll see a blank page with no errors. Not terribly exciting but we've worked some SQL magic in the background and added some data to the database.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Viewing the Table in phpMyAdmin

1. Getting to phpMyAdmin is slightly different whether you're on a Mac or a PC. Follow the appropriate instructions for your platform:

Mac

- Switch to **MAMP PRO**.
- Click the **WebStart** button.
- In the page that opens, click on **Tools** and choose **phpMyAdmin**.

Windows

- Open a browser and go to **http://localhost**
- On the start page, in the **Tools** section click **phpMyAdmin**.

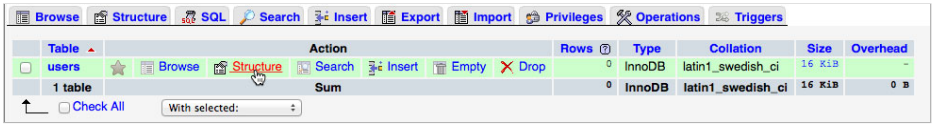
2. On the left, click **phpclass_yourname** to go to your database.
3. You'll see the users table. As shown below, click the **Browse** icon to view the records in the database.

You should see all the rows in the database listed, including the George Washington row you just inserted.

Inserting Information from a Form

We'll now show a slightly more realistic situation and insert user information that is submitted from a form. We've already made the form and set up basic validation on the form data. We'll take this already sanitized input and put it into the database.

1. At the top, click **phpclass_yourname** to go to your database.
2. You'll see the users table. As shown below, click the **Structure** icon to alter the structure of the table.



3. You'll see the structure of the table. We want to add 3 fields at the end of the table. As shown below, enter **3** into the **Add** field, then hit **Go**.

4. Set the following:

Name	Type	Length/Values	null
publications	VARCHAR	255	checked
comments	TEXT		checked
subscribe	TINYINT	1	checked

To break this down:

- **publications** will be a VARCHAR field with a max length of 255 characters.
- **comments** will be a TEXT field. TEXT fields are for storing longer blocks of text such as what may be entered into a comments field. The maximum length for a TEXT column is 65,535 bytes, or around 64kb. That's a decent length of text. For applications that require more storage (such as a content management system) you can use MEDIUMTEXT (stores around 16MB) or LONGTEXT (stores around 4GB).
- **subscribe** will be a TINYINT field with a max length of 1. TINYINT is good for storing very small numbers. In this case subscribe will only be true (1) or false (0) so we want to use the smallest size possible.
- **NULL**: Because all of the fields are optional we want the option of allowing NULL values. A NULL value can be thought of to mean that the user didn't answer the question. In the case of our form, for example, we ask which publications they read. If they don't answer the question at all the field should have NULL value. If they did answer the question but didn't choose any of the values then that could mean that they don't read any of the publications we list. A subtle distinction, but an important one!

5. In the bottom right, click **Save**. Your columns will be added.

Setting Up the PHP

1. Switch back to your code editor.
2. Open **form.php** from the **form-insert** folder.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

3. In a browser go to:
   - Mac: **localhost:8888/phpclass/form-insert/form.php**
   - Windows: **localhost/phpclass/form-insert/form.php**
4. Fill out the form and click Sign me Up!.

   This page already has the validation and input sanitization programmed for you. All we need to do is write the code that adds a record to the database.
5. Open **form-action.php** from the **form-insert** folder.

   This is the page that contains the validation code. We'll add the insert code to this page.
6. Scroll down to around line **66** and find the comment:

   ```
   //insert into database
   ```
7. Below that line, add the following bold code:

   ```
   //insert into database
   require_once('form-insertUser.php');
   ```
8. Save the page.
9. Close it, we're done with it for now.
10. Create a new file.
11. Save it as **form-insertUser.php** into the **form-insert** folder.
12. The first thing to do in this new file is link to the database connection script we made earlier. Add the following bold code:

    ```
    <?php

       require_once('../inc/dbConnect.php');

    ?>
    ```
13. Next we'll write the SQL INSERT statement. We'll add the columns that we just made to the statement. Add the following bold code:

    ```
    require_once('../inc/dbConnect.php');

    $sql = "INSERT INTO users (firstName, lastName, email, publications, comments,
    subscribe)
           VALUES (?, ?, ?, ?, ?, ?)
           ";
    ```

    This will insert our values into the users table. Remember the **?** are all placeholders for the prepared statement we are writing. We save the SQL string into a variable called **$sql**.

14. Next we have to initialize the connection and prepare the **$stmt** object. Add the following bold code:

```
require_once('../inc/dbConnect.php');

$sql = "INSERT INTO users (firstName, lastName, email, publications, comments,
subscribe)
     VALUES (?, ?, ?, ?, ?, ?)
     ";

$stmt = $conn->stmt_init();
$stmt->prepare($sql);
```

15. Now we bind the parameters using the `bind_param()` function. First specify what type of data we are inserting (in this case it will only be strings or integers), and what the name of the variables are. Add the following bold code:

```
require_once('../inc/dbConnect.php');

$sql = "INSERT INTO users
(firstName,lastName,email,publications,comments,subscribe)
     VALUES (?, ?, ?, ?, ?, ?)
     ";

$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('sssssi', $firstName, $lastName, $email, $publications,
$comments, $subscribe);
```

The **'sssssi'** specifies the data of each variable. An **s** represents **s**tring, and an **i** represents **i**nteger. The variables names come from the validation script we wrote. Notice that they are not `$_POST` variable but instead have their own names. The form-action.php script gives them these names and sanitizes the user input.

16. Finally, we can execute the statement and check for errors. Add the following bold code:

```
$stmt->prepare($sql);
$stmt->bind_param('sssssi', $firstName, $lastName, $email, $publications,
$comments, $subscribe);
$stmt->execute();
   if ($stmt->error) {
      echo $stmt->error;
      exit();
   }
```

This executes the `$stmt` object and checks for an error. If there is an error it will output the error information and abort the script.

17. Save the page.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

18. In a browser go to:
    - Mac: **localhost:8888/phpclass/form-insert/form.php**
    - Windows: **localhost/phpclass/form-insert/form.php**
19. Fill out the entire form and click Sign Me Up!.

    You'll see the thank you page, but notice that at the top there is a notice such as:

    ```
    Notice: Array to string conversion in /Applications/MAMP/htdocs/phpclass/form-insert/form-insertUser.php on line 12
    ```

    Notice also that the page did not stop executing even though we said to `exit()` if there was an error. Technically this was not an error, but a **notice**. A **notice** tells you that you're doing something not totally correct, but it is correct enough to let the page still execute.

    In our case, we are inserting an array where a string should be. The **What do you read?** checkbox selector in the form is an array, but in the script we are inserting it as a string.
20. Let's go to phpMyAdmin and see what's being inserted into the database.

    Mac
    - Switch to **MAMP PRO**.
    - Click the **WebStart** button.
    - In the page that opens, click on **Tools** and choose **phpMyAdmin**.

    Windows
    - Open a browser and go to **http://localhost**
    - On the start page, in the **Tools** section click **phpMyAdmin**.
21. On the left click **phpclass_yourname** to go to your database.
22. You'll see the users table. Click the **Browse** icon to view the records in the database.
23. You'll see all the records in the database. Scroll down to the last entry and look in the **publications** column. It will say **Array**. It should be listing the publications they read instead. What we'll do is use a function to convert the array into a list.
24. Switch back to your code editor.
25. You should still be in **form-insertUser.php**.

26. We want to loop through the array of form values and check if each value is an array. If it is an array, convert it to a list string. Luckily the form-action.php script already has an array of all the form values called **`$expected`**. We'll loop through this array. At the top of the page add the following bold code:

```
<?php

   foreach ($expected as $value) {

   }

   require_once('../inc/dbConnect.php');
```

This loops through the **`$expected`** array.

27. First let's see what this loop is outputting. Add the following bold code:

```
foreach ($expected as $value) {
   echo $value;
   echo '<br>';
}
```

This outputs `$value` and adds a line break after each entry.

28. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/form-insert/form.php**
- Windows: **localhost/phpclass/form-insert/form.php**

29. Fill out the entire form and submit it.

You'll see a list of the names of each form field:

```
firstName
lastName
email
publications
comments
subscribe
```

Notice that each one is just the name of the field.

30. Switch back your code editor.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

31. Now we can check to see if **`$value`** is an array. **Delete** the two **echo** lines we just added and replace them with the following bold code:

```
<?php

   foreach ($expected as $value) {
      if ( is_array(${$value}) ) {

      }
   }

   require_once('../inc/dbConnect.php');
```

`${$value}` takes the string that `$value` outputs and evaluates it like a variable. So **publications** becomes **`$publications`**.

32. Finally we can use the function implode() to take the array and convert it to a list. Add the following bold code:

```
foreach ($expected as $value) {
   if ( is_array(${$value}) ) {
       ${$value} = implode(", ", ${$value});
   }
}
```

This says to redefine the variable to a list. **`implode()`** takes two parameters, the first is the **delimiter**. In our case we want a comma-delimited list followed by a space. The second is the array variable we want to implode.

33. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/form-insert/form.php**
- Windows: **localhost/phpclass/form-insert/form.php**

34. Fill out the entire form and under What do you read? check on both options. You shouldn't see any errors or notices—only the thank you page.

35. Go to phpMyAdmin:

Mac

- Switch to **MAMP PRO**.
- Click the **WebStart** button.
- In the page that opens, click on **Tools** and choose **phpMyAdmin**.

Windows

- Open a browser and go to **http://localhost**
- On the start page, in the **Tools** section click **phpMyAdmin**.

36. On the left, click **phpclass_yourname** to go to your database.
37. You'll see the **users** table. Click the **Browse** icon to view the records in the database.
38. Now in the **publications** column you shouldn't see **Array**, but will instead see a nice comma-delimited list of publications such as:

    ```
    Daily Racing Form, Elle
    ```

39. Switch back to your code editor.
40. Close any open files. We're done for now.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

This exercise will show you the SQL syntax for updating a record in a database as well as how to use a form to update user information including checkboxes and hidden fields.

Update Syntax

Updating a record in the database is simple. If we wanted to update a user in the **users** table, the syntax might look like:

```
UPDATE users
SET
   firstName = 'Albert',
   lastName = 'Einstein',
   email = 'albert@someemail.com'
WHERE
   id = 231
```

This says to update the users table and set the firstName, lastName, and email to their corresponding values, where the id equals 231. It is extremely important to always add the **where** line, otherwise you would update **every** row in the database! You must always specify which particular row you'd like to update. That's why having a unique id field in every table is so important, otherwise you might update the wrong data.

A Simple Example

Let's see this in action. First, we need to look up an id in the users table that we are going to modify. It doesn't really matter which id you choose.

1. Go to phpMyAdmin:

Mac

- Switch to **MAMP PRO**.
- Click the **WebStart** button.
- In the page that opens, click on **Tools** and choose **phpMyAdmin**.

Windows

- Open a browser and go to **http://localhost**
- On the start page, in the **Tools** section click **phpMyAdmin**.

2. On the left click **phpclass_yourname** to go to your database.

3. Next to the **users** table click the **Browse** icon to view the records in the database.

4. You'll see all the records in the database. Find a record you'd like to modify and make a note of the **id**. It doesn't matter which one you choose. If in doubt just choose the first one, **id = 1**.

5. Open **update-easy.php** from the **phpclass** folder.

6. Set up the variables we are going to add. At the top of the page add the bold code (for **$id** set it to whatever **id** you chose earlier):

```
<?php

   $firstName = 'Update';
   $lastName = 'Me';
   $email = 'updated@update.com';
   $id = 1;

?>
```

7. Add the connection script by adding the following bold code:

```
$firstName = 'Update';
$lastName = 'Me';
$email = 'updated@update.com';
$id = 1;

require_once('inc/dbConnect.php');
```

8. Next, we'll add the UPDATE statement. Add the bold code:

```
require_once('inc/dbConnect.php');

$sql = "UPDATE users
       SET
          firstName = ?,
          lastName = ?,
          email = ?

       WHERE
          id = ?
       ";
```

Remember, because we are using prepared statements we use **?** placeholders.

9. The rest will be identical to the other prepared statements we've written, first connecting and initializing the statement, then preparing the SQL. Below the $sql, add the bold code:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
```


INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

10. Then bind the parameters:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('sssi', $firstName, $lastName, $email, $id);
```

11. Lastly, execute the statement and add some basic error checking:

```
$stmt = $conn->stmt_init();
$stmt->prepare($sql);
$stmt->bind_param('sssi', $firstName, $lastName, $email, $id);
$stmt->execute();
if ($stmt->error) {
   echo $stmt->error;
   exit();
}
```

12. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/update-easy.php**
    - Windows: **localhost/phpclass/update-easy.php**

    If all goes well you'll see a blank page.

13. Switch back to **phpMyAdmin** and **Browse** the **users** table.

14. Look for the **id** you updated and you'll see that the changes have been made.

Update Form

For the rest of this exercise, we'll explore a more useful real world example and build an update form that could be used for a user admin.

1. Open **userList.php** from the **update** folder.

2. In a browser go to:

    - Mac: **localhost:8888/phpclass/update/userList.php**
    - Windows: **localhost/phpclass/update/userList.php**

3. You'll see a list of all the records in the **users** table. The first column contains links that say "Edit" that will bring the user to an edit page where they can update the user info. Also notice that the last column, **Subscribe**, contains 1s and 0s. We'll change this to a more useful Yes or No.

4. Switch back to your code editor.

5. Around line **49** find the code that displays the **$subscribe** variable:

```
<td><?php echo $subscribe; ?></td>
```

While we could use an if statement to display **Yes** if $subscribe is equal to 1, and **No** if it is not, this is a perfect time to introduce you to PHP's ternary operator. The ternary operator serves as shorthand notation for if statements.

6. Change the code as shown in bold. (Don't forget to delete the semi-colon and add parentheses around $subscribe!)

```
<td><?php echo ($subscribe) ? 'Yes' : 'No' ?></td>
```

The ternary operator uses three expressions separated by a question mark and a colon. The question mark follows the test expression (what's in parentheses) and can be thought of as asking, "Is it true?". The colon then separates the two possible values; the first is chosen if the test expression is true, the second if it is false.

In our example the ternary operator evaluates $subscribe and returns the first response (Yes) if it evaluates to true; the second (No) if it is false. If $subscribe is equal to 0, or is NULL it will evaluate as not true.

Next, we'll make a link that brings the user to an update form. To do that, we need to pass the user id to a form page. We'll do that by putting the id in the URL. The anchor tag we're going to build will look like this:

```
<a href="userForm.php?id=3">Edit</a>
```

The special thing about this URL is that it has a question mark after the page name, followed by **id=3**. Anything after the question mark will be treated as a URL variable. To access the URL variable on the following page, we'll use PHP's $_GET array.

7. Around line **43**, find the **Edit** link. Currently it links to **userForm.php**, but does not have a URL variable:

```
<td><a href="userForm.php">Edit</a></td>
```

8. Let's add the start of the variable. Add the bold code:

```
<td><a href="userForm.php?id=">Edit</a></td>
```

9. Next, add the PHP that will echo each user's id. Directly after the **id=** add the bold code:

```
<td><a href="userForm.php?id=<?php echo $id; ?>">Edit</a></td>
```

10. Save the page and then in a browser go to:

- Mac: **localhost:8888/phpclass/update/userList.php**
- Windows: **localhost/phpclass/update/userList.php**

11. Try clicking a few of the links and notice that you are taken to **userForm.php** and that the id of the user you click is in the URL.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Display Data in the Update Form

1. Switch back to your code editor.
2. Open **userForm.php** from the **update** folder.
3. Look at the top of the page and note that we've already written the SQL and PHP code for you to display a user record. This is a similar SELECT statement to the ones you've made in previous exercises, although one thing to note is that on line **9** we reference the URL variable like so:

   ```
   $stmt->bind_param('i',$_GET['id']);
   ```

   This is the one parameter in the select statement, the user **id**. Here we are telling the `bind_param()` function to expect one integer, the 'id' in the `$_GET` array (which is passed on from the preceding page).

   Because the select statement is already written, we can focus on just display data.
4. First let's output the first name. Around line **36** notice that the **value=""** is empty. We'll fill it with the user info. Add the bold code:

   ```
   <input name="firstName" type="text" id="firstName" size="40"
   value="<?php echo $firstName; ?>">
   ```
5. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/update/userList.php**
   - Windows: **localhost/phpclass/update/userList.php**
6. Click one of the **Edit** links. You'll be taken to the form, and the first name should already be filled out.
7. Let's do the same for the last name. Switch back to your text code editor.
8. Around line **40** add the bold code:

   ```
   <input name="lastName" type="text" id="lastName" size="40"
   value="<?php echo $lastName; ?>">
   ```
9. And the same for the email. Around line **44** add the bold code:

   ```
   <input name="email" type="text" id="email" size="40" value="<?php echo
   $email; ?>">
   ```
10. Do the same for the comments. Around line **54** add the bold code:

    ```
    <textarea name="comments" id="comments"
    cols="38" rows="5"><?php echo $comments; ?></textarea>
    ```

    Note that this should actually all be on one line! It's just a bit too long to display in one line in this book, but because white space in a `<textarea>` matters, there shouldn't be any space between the `<textarea>` tags.

11. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/update/userList.php**
    - Windows: **localhost/phpclass/update/userList.php**
12. Click one of the **Edit** links. You'll be taken to the form, and the first name, last name, email, and comments should be filled out.

Display Checkboxes

Displaying whether checkboxes have been checked or not is pretty easy. We just have to test if the user had selected a particular option and then add a **checked** to the `<input>` tag. Let's do it for the subscribe tag first.

1. Switch back to your code editor.
2. In **userForm.php**, around line **59**, find the **subscribe** input. Add an if statement that checks to see if $subscribe is true and if it is, output **checked**. Add the bold code:

```
<input name="subscribe" type="checkbox" id="subscribe"
value="1" <?php if ($subscribe) {echo 'checked';}?>>
```

3. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/update/userList.php**
    - Windows: **localhost/phpclass/update/userList.php**
4. Click one of the **Edit** links and make a note of whether the **subscribe** input is checked or not.
5. The same principle will work for the **publications** checkboxes, except that first we must search through the `$subscribe` string to see if they selected the checkbox earlier. Switch back to your code editor.
6. In **userForm.php**, around line **49**, find the checkbox input for **Daily Racing Form**. Add the bold code:

```
<input name="publications[]" type="checkbox" id="publications_drf"
value="Daily Racing Form" <?php if () { echo 'checked';} ?>> Daily Racing
Form</label>
```

This is a blank **if** statement. In the next step we will add the function that searches through a string to see if the user selected the option.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

7. In between the empty parentheses of the **if ()** add the bold code:

```
<?php if ( stristr($publications,'Daily Racing Form') ) { echo 'checked';} ?>
```

stristr() is a case-insensitive search function. In this case we are searching through the $publications variable for the string 'Daily Racing Form'. If it finds it, it outputs **checked**.

8. Add the same code for the **Elle** input, except replace Daily Racing Form with **Elle**. Inside the **Elle** input, add the bold code:

```
<input name="publications[]" type="checkbox" id="publications_elle"
value="Elle" <?php if ( stristr($publications,'Elle') ) { echo 'checked';} ?>>
Elle</label>
```

This searches through the $publications variable for the string `'Elle'` and outputs **checked** if it finds it.

9. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/update/userList.php**
   - Windows: **localhost/phpclass/update/userList.php**
10. Click one of the **Edit** links making note of which publications the user selected. When you are taken to the form the **publications** checkboxes should correspond.

Adding a Hidden Field

The update form is nearly done, but the user id still needs to be passed along with the rest of the user information. We do that by adding it to a hidden field in the form. This way the user cannot accidentally change the id, thereby making sure that the proper user info is updated.

1. Switch back to your code editor.
2. In **userForm.php**, around line **32**, find the opening `<form>` tag.
3. Make a new line below it and add the following bold code:

```
<form action="form-action.php" method="post" name="signup" id="signup">
<input type="hidden" name="id" value="">
```

This is a hidden input field with the name **id**.

4. To add the PHP that echoes the user's **id**, add the following bold code:

```
<input type="hidden" name="id" value="<?php echo $id ?>">
```

5. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/update/userList.php**
   - Windows: **localhost/phpclass/update/userList.php**

Building the Update SQL

1. Switch back to your code editor.
2. Open **form-action.php** from the **update** folder.

   The update form submits to the page **form-action.php**. This is similar to the action page we've used in previous exercises. It loops through the form and validates the input. It also renames the $_POST values to the values we give in the $expected array.
3. Scroll down to line **66** and find the comment:

```
//update user in database
```

4. Underneath that comment add the bold code:

```
//update user in database
require_once('updateUser.php');
```

5. Save the page.
6. Open **updateUser.php** from the **update** folder.

   To save some time, we've already written most of the PHP code for you. All the connection code is already written. We just need to write the SQL and then bind the parameters.
7. Around line **13** find the empty **$sql** variable and add the following bold code:

```
$sql = "UPDATE users
        SET
          firstName = ?,
          lastName = ?,
          email = ?,
          publications = ?,
          comments = ?,
          subscribe = ?

       WHERE
          id = ?
       ";
```

   This says to **update** the users table and set the following values where the id is equal to a certain value.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

8. Around line **28** find the comment:

   ```
   //bind params here
   ```

9. We will bind the parameters there. Replace the comment by adding the following bold code:

   ```
   $stmt->bind_param('ssssssi', $firstName, $lastName, $email, $publications,
   $comments, $subscribe, $id);
   ```

   Note that all of the variables are marked as a **s**tring, except for the id which is an **i**nteger.

10. Save the page and then in a browser go to:

    - Mac: **localhost:8888/phpclass/update/userList.php**
    - Windows: **localhost/phpclass/update/userList.php**

11. Click **Edit** next to one of the users.

12. Make some changes in the form and click **Update User**. You'll be taken back to the list of users and should see the updates!

13. Switch back to your code editor.

14. Close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

Guess what? This exercise shows you how to delete records from a database.

Delete Syntax

Deleting a record uses similar syntax to updating a user. To delete user 231 from the users table you'd write:

```
DELETE FROM users
WHERE id = 231
```

Remember again that you must always include the **WHERE** line. If you do not you'd delete every record in the table!

We'll build a few pages that allow the user to delete records from the user table.

1. Open **userList.php** from the **delete** folder.
2. In a browser go to:
   - Mac: **localhost:8888/phpclass/delete/userList.php**
   - Windows: **localhost/phpclass/delete/userList.php**

   You'll see a list of all the records in the **users** table. The last column contains links that say **Delete** that will bring the user to a confirmation page to ask if they are sure they want to delete the user.
3. Switch back to your code editor.
4. Around line **49**, find the **Delete** link. Currently it links to **deleteConfirm.php**, but does not have a URL variable:

   ```
   <td><a href="deleteConfirm.php">Delete</a></td>
   ```
5. Add the start of the variable as shown in bold:

   ```
   <td><a href="deleteConfirm.php?id=">Delete</a></td>
   ```
6. Next we'll add the PHP that will echo each user's id. Directly after the **id=** add the bold code:

   ```
   <td><a href="deleteConfirm.php?id=<?php echo $id; ?>">Delete</a></td>
   ```
7. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/delete/userList.php**
   - Windows: **localhost/phpclass/delete/userList.php**

8. Try clicking a few of the links. Notice that you are taken to **deleteConfirm.php** and that the id of the user you click is in the URL.

   deleteConfirm.php is simply a page that contains two links. One will take the user to the page that actually deletes the user, and the other returns the user back to the user list.

9. Switch back to your code editor.

10. Open **deleteConfirm.php** from the **delete** folder.

11. Around line **14** add the user id to the **deleteUser.php** link as shown below in bold:

```
<p><a href="deleteUser.php?id=<?php echo $_GET['id']; ?>">Delete User</a></p>
```

   Because the user id was passed along from the previous page, it is available in the `$_GET` array.

12. Save the page.

13. Open **deleteUser.php** from the **delete** folder.

   Most of the PHP database code has already been written for you. All that is left to do is to write the SQL, bind the parameters, and direct the user back to the user list after the record has been deleted.

14. Around line **5** find the empty **`$sql`** variable and add the bold code:

```
$sql = "DELETE FROM users
        WHERE
        id = ?
        ";
```

   This says to delete from the users table where the id is equal to the bound parameter.

15. Around line **12** find the **//bind params here** comment and replace it with the bold code:

```
$stmt->bind_param('i', $_GET['id']);
```

   This binds the **id** variable and tells PHP to expect an integer.

16. Around line **19** find the **//go back to user list** comment and underneath it add the bold code:

```
//go back to user list
require_once('userList.php');
```

   This will include the user list page.

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

17. Save the page and then in a browser go to:
    - Mac: **localhost:8888/phpclass/delete/userList.php**
    - Windows: **localhost/phpclass/delete/userList.php**
18. Click one of the **Delete** links and be sure to make note of which one you clicked.
19. On the confirmation page click **Delete User** and you'll see that record get deleted.
20. Switch back to your code editor.
21. Close any open files. We're done for now.

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Exercise Overview

There are an enormous number of ways to search for information in a database—far more than can be covered in this book. We'll show how to perform a basic wildcard search on a column.

Simple Search

To search for a user by email, the syntax would be:

```
SELECT *
FROM users
WHERE email LIKE '%mySearchTerm%'
```

This will match any record in the database that has an email that contains text in the variable **mySearchTerm**. The **%** signs on either side of the search are the wildcard filters and mean to match any text before or after what the user enters.

1. Open **search.php** from the **search** folder.
2. In a browser go to:
   - Mac: **localhost:8888/phpclass/search/search.php**
   - Windows: **localhost/phpclass/search/search.php**

   It's just a simple form with one input for an email. It submits to a page called **searchResults.php** which is the page that will perform the search and display the results.
3. Switch back to your code editor.
4. Open **searchResults.php** from the **search** folder.

   Most of the page has already been written. It contains a SELECT statement and outputs the results to a table. We'll add the wildcard filters and format the user input so it is compatible with the parameterized query.
5. Around line **5**, modify the **$sql** variable by adding the bold text as shown below:

```
$sql = 'SELECT id, firstName, lastName, email, publications, comments,
subscribe
FROM users
WHERE email LIKE ?
';
```

This will filter the results to only display records that are LIKE the bound parameter.

Modifying the Search Term

Note in the syntax example at the beginning of the exercise that the search term is surrounded by percent signs. PHP won't allow those percent signs to be put directly in the SQL—rather they need to be added to the search term itself.

1. Concatenate percent signs to the `$_POST['email']` that is submitted from the search form. At the top of the page above the **$sql** variable, add the following bold code:

   ```
   $searchTerm = '%' . $_POST['email'] . '%';
   ```

   This takes the email that is submitted from the form (`$_POST['email']`) and adds percent signs on either side of it.

2. Finally we need to bind this parameter to the query. Around line **14** find the **//bind params here** comment and replace it with the bold code:

   ```
   $stmt->bind_param('s', $searchTerm);
   ```

   This binds the **`$searchTerm`** variable and tells PHP to expect a string.

3. Save the page and then in a browser go to:
   - Mac: **localhost:8888/phpclass/search/search.php**
   - Windows: **localhost/phpclass/search/search.php**
4. Enter a full or partial email and test it out. Excellent!
5. Switch back to your code editor.
6. Close any open files. That's all folks!

INTRO TO PHP 5 & MYSQL • COPYRIGHT NOBLE DESKTOP
LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM

Check Out
OUR OTHER WORKBOOKS!

Web Development Level 1 and 2

JavaScript & jQuery

GreenSock Animation

Mobile & Responsive Web Design

WordPress.org

PHP & MySQL

Photoshop for Web & UI

Photoshop Animated GIFs

Sketch

HTML Email

Responsive HTML Email

Ruby on Rails

PowerPoint

Adobe InDesign

Adobe Illustrator

Adobe Photoshop

Photoshop Advanced

Adobe Lightroom

Adobe After Effects

Adobe CC: Intro to InDesign, Photoshop, & Illustrator

NOBLEDESKTOP.COM/BOOKS

LICENSED BY: DAVID H RAMIREZ • RAMIREZD@SHARPSEC.COM