COMP1230 LAB MANUAL

ADVANCED WEB PROGRAMMING

This booklet will help the reader understand the concepts, principles, and implementation of the PHP language. By the end of the booklet, the reader will be able to code comfortably in PHP.

Prepared by Ben Blanc and Maziar Masoudi

TABLE OF CONTENTS

CHAPTER 1

GETTING STARTED

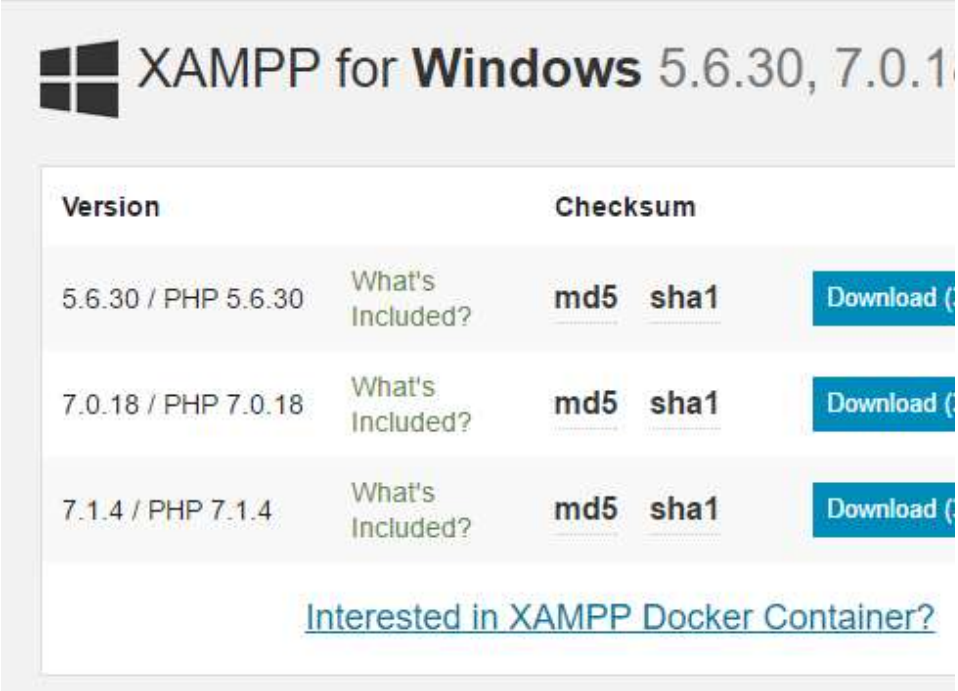
REQUIRED TOOLS

There are three required programs that you need to have installed on your personal machines to complete the lab manuals and practice coding: Xammp, FileZilla, and PHP Storm.

Xampp

Navigate to the following URL: https://www.apachefriends.org/download.html

And download the latest XAMPP with the latest PHP version.



It is available for both Windows and Mac machines. After downloading it, open and following the installation instructions.

FileZilla

Navigate to the following URL: https://filezilla-project.org/ and download the Client application

CHAPTER 1 AT A GLANCE

In this chapter you will learn where to download and how to install the required tools to run a PHP script, how to output information and receive user input, and how to debug in PHP.

It is available for both Windows and Mac machines. After downloading it, open and following the installation instructions.

PHPStorm

Following the instructions and video given to you in Week 1 on GBLearn. The instructions include

- If you don't have a JetBrains account, signing up for a Student Account with JetBrains at https://www.jetbrains.com/shop/eform/students
    - Use your George Brown email to signup
    - Confirming the confirmation email received from JetBrains.
- Downloading the PHPStorm product
- Downloading the JetBrains license
- Pasting the license onto your personal machine. (You may also validate the license by using your JetBrains account credentials)

RUNNING YOUR PHP APP

Now it is time to run your code.

In PHPStorm

Hover your mouse at the top right corner of the white page, and you will see web browser options. Select any web browser you like and run your PHP code.

In Xammp

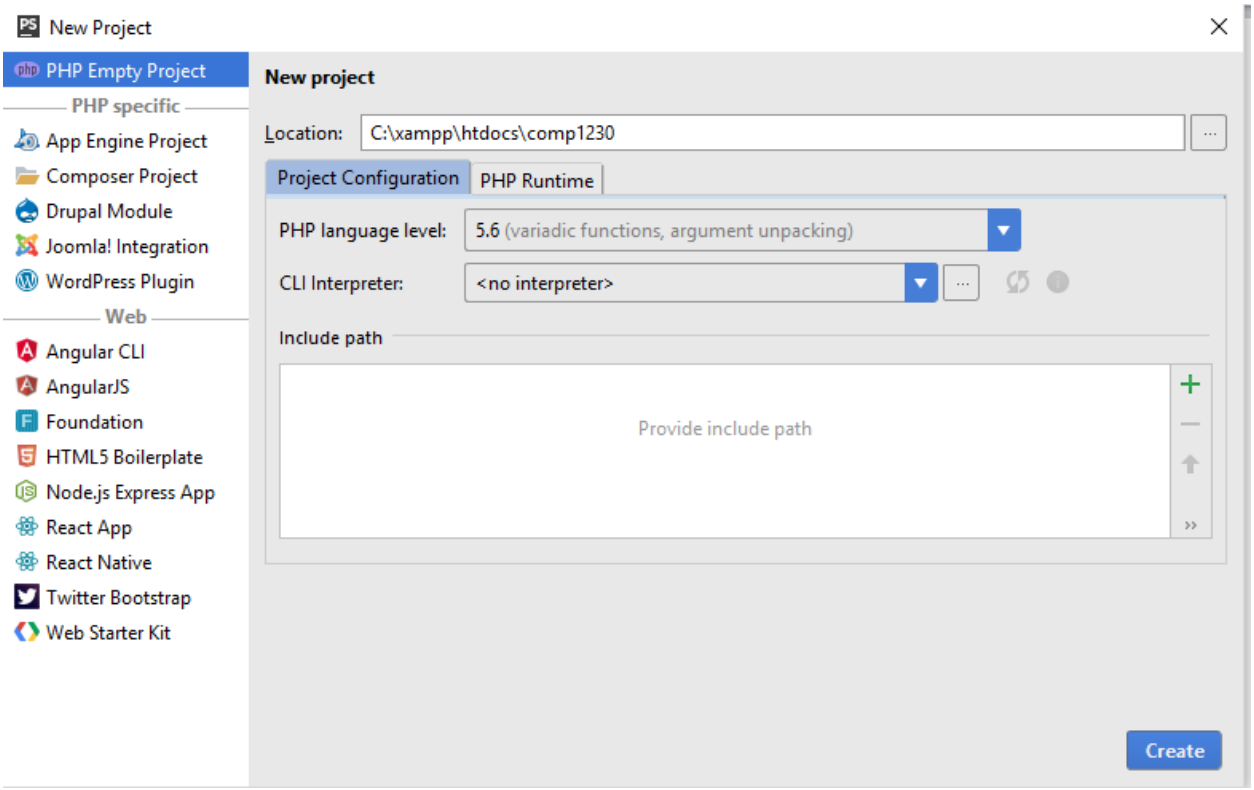
To run your code in XAMMP, you will need to move the files into your htdocs folder of your XAMMP installation. Then you can navigate to http://localhost:(port)/directory/file_name in your web browser.

In Filezilla

To run your code on Filezilla, you need to log-in, navigate to the public_html folder, drag your files into that folder, then navigate to http://USERNAME.gblearn.com/directory/file_name in your web browser's public_html folder.

CREATING FIRST PHP APP

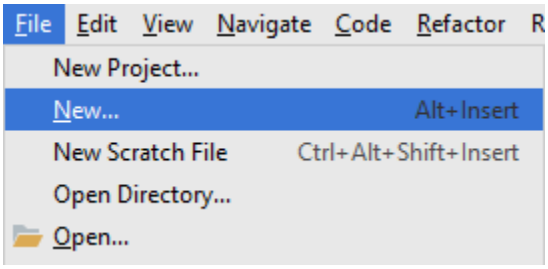
Open PHP storm and click on New Project



Choose the location of the project and select the PHP language level.

For the interpreter, click on the dots, click on the plus sign on the top left corner of the new window (Other local), and navigate to the location of your php.exe of your XAMMP installation. After completing that step, you can select a PHP interpreter from the drop down

Once you have finished, you will see a blank project. Click on File -> New. Then select PHP file and fill out a name



You will see a blank page with comments. Add the following code to the page:

```
echo "hello world";
```

Output

Two main ways you can output information from a user is by the PRINT and ECHO method.

PRINT

Below are examples demonstrating the output and requirements of the PRINT method

```
print "This";
print "Will";
print "Be";
print "One";
print "Word";
```

When you run the code, all the output will be joined together with no line breaks. The output can be seen below.

ThisWillBeOneWord

We can add spaces and HTML code to each PRINT statement

```
print "This ";
print "Will ";
print " <strong> Be </strong> ";
print " <p> Many </p>";
print "Words";
print " <p> And </p>";
print " <i> Lines </i>";
```

The output for the code above is the following:

This Will **Be**

Many

Words

And

Lines

ECHO

The ECHO method works in the exact same manner as print but you can pass multiple arguments to it. See the examples below

```
echo "This";
echo "Will";
echo "Be";
echo "One";
echo "Word";
```

OR

```
echo "This", "Will", "Be", "One", "Word";
```

Would output

ThisWillBeOneWord

In the same manner,

```
echo "This ";
echo "Will ";
echo " <strong> Be </strong> ";
echo " <p> Many </p>";
echo "Words";
echo " <p> And </p>";
echo " <i> Lines </i>";
```

OR

```
echo "This ","Will "," <strong> Be </strong> ",
" <p> Many </p>","Words"," <p> And </p>"
," <i> Lines </i>";
```

Would output

This Will **Be**

Many

Words

And

Lines

DIFFERENCE BETWEEN PRINT & ECHO

The notable differences between the two output statements are

- PRINT returns true on every statement and is thus marginally slower than ECHO
- You can pass multiple arguments to ECHO

Input

Input in PHP is done in two ways

- Form input
- Query strings

In either way, the global variable $_GET or $_POST is used to access information.

$_GET

This variable is an array variable that takes an index that represents the name of the parameter passed to in. You can pass a parameter name-value pair to the $_GET variable by a query string in the following way.

- Create a new PHP file named **get.php**
- Input the following PHP code into your file

```php
$n1 = $_GET['n1'];

$n2 = $_GET['n2'];

print 'The value of '.
'n1 is ';

echo $n1;

echo "<br/>";

print 'The value of '.
    'n2 is ';

echo $n2;
```

Run your code through your localhost (XAMMP) and add the following text to your URL
get.php?n1=hello&n2=world

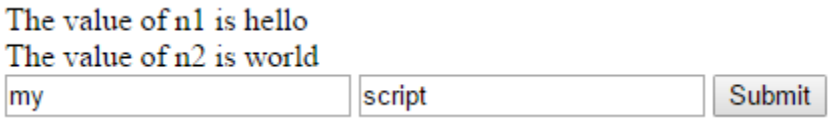
The same feat can be accomplished if you create the following form and submit it.

- Create a new PHP file named **get_form.php**
- Copy the contents of **get.php**
- Paste it into the body of **get_form.php**
- Add the following code to the end of the file.

```php
?>

<form method="get">

    <input type="text" name="n1" />

    <input type="text" name="n2" />

    <input type="submit" name="submit" />


</form>
```

- Run your code through your localhost and add the following text to your URL **get_form.php?n1=hello&n2=world**
- The output should be

- Use the form to change the values of n1 and n2



- Upon pressing Submit, the URL values will change, and the screen values will reflect this change.

$_POST

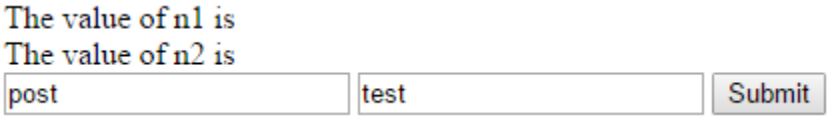
This variable is very similar to the $_GET variable with the following differences

- It can only be used in a form
- Name-value pairs are hidden from the URL (more secure)
- Create a new PHP file named **post_form.php**
- Copy the contents of **get_form.php**
- Paste it into the body of **post_form.php**
- Make the following changes to the code

```
$n1 = @$_POST['n1'];

$n2 = @$_POST['n2'];

print 'The value of '.
    'n1 is ';

echo $n1;

echo "<br/>";

print 'The value of '.
    'n2 is ';

echo $n2;



?>

<form method="post">

    <input type="text" name="n1" />

    <input type="text" name="n2" />

    <input type="submit" name="submit" />


</form>
```

Note that we have added the @ character to suppress a notice error the first time we will run this script.

- Use the form to submit the values for n1 and n2



- Run your code through your localhost

DIFFERENCES BETWEEN $_GET & $_POST

- The name-value pairs
  - NAME refers to either
    - the input name (of a form)
    - the query string name (of a URL)
  - VALUE refers to
    - the data the user inputted (in a form)
    - value of that comes after the equals [=] sign (of a URL)
- Name of input or query matches index of the $_GET or $_POST variable
  - Index refers to the value in quotes found within the square brackets
- $_GET and/or $_POST variable is case sensitive
- $_POST compare to $_GET is more secure since it doesn't display name and value pair in the URL.
- $_GET doesn't require a form. (?key=val)

DEBUG PHP APPS

There are a few ways you can turn on debugging on your PHP applications.

You can navigate to your php.ini and the directive display_errors to **On**

You can also use the PHP function ini_set at the top of your PHP application.

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

After the display_errors directive has been set to On, you would then set which level of error messages you want to display:

Version	Description
5.4.0	**E_STRICT** became part of **E_ALL**.
5.3.0	**E_DEPRECATED** and **E_USER_DEPRECATED** introduced.
5.2.0	**E_RECOVERABLE_ERROR** introduced.
5.0.0	**E_STRICT** introduced (not part of **E_ALL**).

Sample code would be the following:

```
// Report all PHP errors (see changelog)
error_reporting(E_ALL);
```

Together, it would be

```
error_reporting(E_ALL);
ini_set("display_errors", 1);
```

Types of Errors

There are 5 types of error you will encounter in PHP

FATAL

This error forces the script to stop executing because of a critical error. An example is trying to use a function that doesn't exist or if a file passed by the require() method cannot be found.

PARSE

In most programming languages, this is known as a Syntax error. It occurs when there is a violation of the language syntax. It stops the execution of the script. An example is missing a semi-colon.

WARNING

This error happens when you violate the parameters of a function in PHP. It does not stop the execution of the script. An example is passing the method strlen() without data.

NOTICE

This error is the most frequent error that happens. It is the lowest level error and does not stop the execution of the script. An example is trying to access a variable not declared. To suppress this warning in known situations, add the @ character (like we did in the $_POST example).

LOGICAL

A logical error is an error that does not produce any error that causes your application to stop the execution, but does not produce the desired result.

An example of a logical error is calculating an average of two numbers by the following code:

8+12 / 2

The desired result is 10, but 14 will the actual result because of the order of operations.

The correct code should be

(8 + 12) / 2

Var_Dump()

The `var_dump()` function is used to display structured information (type and value) about one or more variables. Use it to get information and diagnose an issue with your variables.

```
$var="boo-hoo";

var_dump($var);
```

Results in the following output

```
string(7) "boo-hoo"
```

Comments

Comments are a way you can communicate what your code is doing and why. There are three ways to add comments to a PHP file.

SINGLE LINE

Single line comments are added in two ways in PHP

```
//This is a single line comment

#This is also a single line comment
```

MULTILINE

Multiline comments are added by the following code

```
/*
 * This is
 * A multiline
 * Comment
 */
```

PHP DOCUMENTATION

PHP has a great documentation that you can view at this link: http://php.net/docs.php