Chapter 2

How to code a PHP application

Objectives

Applied

1. Given the specifications for a PHP application that requires only the skills and language elements presented in this chapter, code, test, and debug the application. That includes these skills:
   - Creating variables with valid names and assigning values to them
   - Using literals and concatenating strings
   - Using the built-in $_GET and $_POST arrays
   - Using echo statements to display data on a page
   - Coding string and numeric expressions
   - Using compound assignment operators
   - Using the built-in intdiv(), number_format(), date(), isset(), empty(), and is_numeric() functions

Objectives (continued)

Applied (continued)

- Using the htmlspecialchars() function to escape special characters on a page
- Using the filter_input() function to filter input from the $_GET and $_POST arrays
- Coding conditional expressions
- Coding if, while, and for statements
- Using built-in functions like include() and require() to pass control to another page

2. Access and use the online PHP documentation.

Objectives (continued)

Knowledge

1. Explain how PHP is embedded within an HTML document.
2. Distinguish between PHP statements and comments.
3. Describe these PHP data types: integer, double, Boolean, and string.
4. List the rules for creating a PHP variable name.
5. Describe the code for declaring a variable and assigning a value to it.
6. Describe the use of the built-in $_GET and $_POST arrays.
7. Describe the use of the echo statement.
8. Describe the rules for evaluating an arithmetic expression, including order of precedence and the use of parentheses.

Objectives (continued)

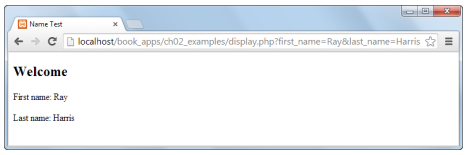
Knowledge (continued)

9. Describe the use of these built-in functions: intdiv(), number_format(), date(), isset(), is_numeric(), htmlspecialchars(), filter_input(), include(), and require().
10. Describe how an XSS attack works and how to protect against this type of attack.
11. Describe the rules for evaluating a conditional expression, including order of precedence and the use of parentheses.
12. Describe the flow of control of an if, while, or for statement.

A PHP file that includes HTML and embedded PHP

```
<?php
    // get the data from the request
    $first_name = $_GET['first_name'];
    $last_name = $_GET['last_name'];
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Name Test</title>
        <link rel="stylesheet" type="text/css"
              href="main.css"/>
    </head>
    <body>
        <h2>Welcome</h2>
        <p>First name: <?php echo $first_name; ?></p>
        <p>Last name: <?php echo $last_name; ?></p>
    </body>
</html>
```

The PHP file displayed in a browser



PHP code: comments and statements

```
<?php
    /*********************************************
     * This program calculates the discount for a
     * price that's entered by the user
     *********************************************/

    // get the data from the form
    $list_price = $_GET['list_price'];

    // calculate the discount
    $discount_percent = .20;  // 20% discount
    $discount_amount =
        $subtotal * $discount_percent;
    $discount_price =
        $subtotal - $discount_amount;
?>
```

Another way to code single-line comments

```
# calculate the discount
$discount_percent = .20;      # 20% discount
```


Syntax rules

- PHP statements end with a semicolon.
- PHP ignores extra whitespace in statements.

The six PHP data types

- integer
- double
- boolean
- string
- array
- object

Integer values (whole numbers)

```
15              // an integer
-21             // a negative integer
```

Double values (numbers with decimal positions)

```
21.5            // a floating-point value
-124.82         // a negative floating-point value
```

The two Boolean values

```
true            // equivalent to true, yes, or on
false           // equivalent to false, no, off, or 0
```

String values

```
'Ray Harris'    // a string with single quotes
"Ray Harris"    // a string with double quotes
''              // an empty string
null            // a NULL value
```


Double values that use scientific notation

```
3.7e9           // equivalent to 3700000000
4.5e-9          // equivalent to 0.0000000037
-3.7e9          // equivalent to -3700000000
```

Using the assignment operator (=) as you declare a variable and assign it a value

```
$count = 10;              // an integer literal
$list_price = 9.50;       // a double literal
$first_name = 'Bob';      // a string literal
$first_name = "Bob";      // a string literal
$is_valid = false;        // a Boolean literal

$product_count = $count;  // $product_count is 10
$price = $list_price;     // $price is 9.50
$name = $first_name;      // $name is "Bob"
$is_new = $is_valid;      // $is_new is FALSE
```


Rules for creating variable names

- Variable names are case-sensitive.
- Variable names can contain letters, numbers, and underscores.
- Variable names can't contain special characters.
- Variable names can't begin with a digit or two underscores.
- Variable names can't use names that are reserved by PHP such as the variable named $this that's reserved for use with objects.

How to declare a constant

```
define('MAX_QTY', 100);     // an integer constant
define('PI', 3.14159265);   // a double constant
define('MALE', 'm');        // a string constant
```


An HTML form that does an HTTP GET request

```
<form action="display.php" method="get">
    <label>First name: </label>
    <input type="text" name="first_name"/><br>
    <label>Last name: </label>
    <input type="text" name="last_name"/><br>
    <label> </label>
    <input type="submit" value="Submit"/>
</form>
```

The URL for the HTTP GET request

```
//localhost/.../display.php?first_name=Ray&last_name=Harris
```

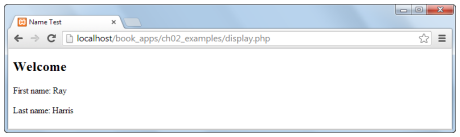
Getting the data and storing it in variables

```
$first_name = $_GET['first_name'];
$last_name = $_GET['last_name'];
```


An <a> tag that performs an HTTP GET request

```
<a href="display.php?first_name=Joel&last_name=Murach">
   Display Name
</a>
```


A PHP page for an HTTP POST request



An HTML form that specifies the POST method

```
<form action="display.php" method="post">
```

Code that gets the data from the $_POST array

```
$first_name = $_POST['first_name'];
$last_name = $_POST['last_name'];
```

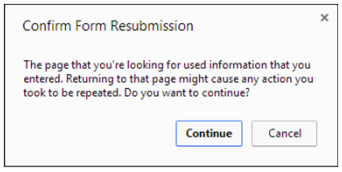
When to use the HTTP GET method

- When the request is for a page that gets data from a database server.
- When the request can be executed multiple times without causing any problems.

When to use the HTTP POST method

- When the request is for a page that writes data to a database server.
- When executing the request multiple times may cause problems.
- When you don't want to include the parameters in the URL for security reasons.
- When you don't want users to be able to include parameters when they bookmark a page.
- When you need to transfer more than 4 KB of data.

The Chrome dialog box that's displayed if the user tries to refresh a post



How to assign string expressions

Use single quotes to improve PHP efficiency

```
$first_name = 'Bob';
$last_name = 'Roberts';
```

Assign NULL values and empty strings

```
$address2 = '';             // an empty string
$address2 = null;           // a NULL value
```

Use double quotes for variable substitution

```
$name = "Name: $first_name";          // Name: Bob
$name = "$first_name $last_name";     // Bob Roberts
```

Mix single and double quotes for special purposes

```
$last_name = "O'Brien";               // O'Brien
$line = 'She said, "Hi."';            // She said, "Hi."
```


How to use the concatenation operator (.)

How to use the concatenation operator for simple joins

```
$first_name = 'Bob';
$last_name = 'Roberts';
$name = 'Name: ' . $first_name;          // Name: Bob
$name = $first_name . ' ' . $last_name;  // Bob Roberts
```

How to join a number to a string

```
$price = 19.99;
$price_string = 'Price: ' . $price;    // Price: 19.99
```


The syntax for the echo statement

```
echo string_expression;
```

How to use an echo statement

```
<p>Name: <?php echo $name; ?></p>
```

How to use an echo statement with the htmlspecialchars() function

```
<p>Name: <?php echo htmlspecialchars($name); ?></p>
```

Common arithmetic operators

Operator	Example	Result
+	5 + 7	12
-	5 - 12	-7
*	6 * 7	42
/	13 / 4	3.25
%	13 % 4	1
++	$counter++	adds 1 to counter
--	$counter--	subtracts 1 from counter

Some simple numeric expressions

```
$x = 14;
$y = 8;
$result = $x + $y;          // 22
$result = $x - $y;          // 6
$result = $x * $y;          // 112
$result = $x / $y;          // 1.75
$result = $x % $y;          // 6
$x++;                       // 15
$x--;                       // 14
```

A statement that uses the intdiv() function (PHP 7 and later)

```
$result = intdiv($x, $y);   // 1
```


Statements that calculate a discount

```
$list_price = 19.95;
$discount_percent = 20;
$discount_amount = $list_price * $discount_percent * .01;
$discount_price = $list_price - $discount_amount;
```


The order of precedence

Order	Operators	Direction
1	++	Left to right
2	--	Left to right
3	* / %	Left to right
4	+ -	Left to right

Order of precedence and the use of parentheses

```
3 + 4 * 5           // 23
(3 + 4) * 5         // 35
```


The compound assignment operators

- .=
- +=
- -=
- *=
- /=
- %=

Two ways to append string data to a variable

The standard assignment operator

```
$name = 'Ray ';
$name = $name . 'Harris';      // 'Ray Harris'
```

A compound assignment operator

```
$name = 'Ray ';
$name .= 'Harris';             // 'Ray Harris'
```

Three ways to increment a counter variable

The standard assignment operator

```
$count = 1;
$count = $count + 1;
```

The compound assignment operator

```
$count = 1;
$count += 1;
```

The increment operator

```
$count = 1;
$count++;
```


More examples

How to append numeric data to a string variable

```
$message = 'Months: ';
$months = 120;
$message .= $months;          // 'Months: 120'
```

How to work with numeric data

```
$subtotal = 24.50;
$subtotal += 75.50;           // 100
$subtotal *= .9;              // 90 (100 * .9)
```


A function for formatting numbers

```
number_format($number[, $decimals])
```

Statements that format numbers

```
$nf = number_format(12345);           // 12,345
$nf = number_format(12345, 2);        // 12,345.00
$nf = number_format(12345.674, 2);    // 12,345.67
$nf = number_format(12345.675, 2);    // 12,345.68
```


A function for getting the current date

```
date($format)
```

Commonly used characters for date formatting

Character	Description
`Y`	A four-digit year such as 2017.
`y`	A two-digit year such as 17.
`m`	Numeric representation of the month with leading zeros (01-12).
`d`	Numeric representation of the day of the month with leading zeros (01-31).

Statements that format a date

```
$date = date('Y-m-d');    // 2017-09-12
$date = date('m/d/y');    // 09/12/17
$date = date('m.d.Y');    // 09.12.2017
$date = date('Y');        // 2017
```


Three functions for checking variable values

```
isset($var)
empty($var)
is_numeric($var)
```

Function calls that check variable values

```
isset($name)          // TRUE if $name has been set
                      // and is not NULL
empty($name)          // TRUE if $name is empty
is_numeric($price)    // TRUE if $price is a number
```


Two functions for converting user-entered data for display

Name	Description
`htmlspecialchars($string)`	Converts certain HTML special characters (&, ', ", <, and >) to their corresponding HTML entities and returns the resulting string.
`htmlentities($string)`	Converts all HTML characters that have corresponding HTML entities and returns the resulting string.

The filter_input() function

Name	Description
`filter_input($type, $variable_name [, $filter])`	Gets a value from a superglobal variable and optionally filters it. Returns the requested value on success, a FALSE value if the filter fails, or a NULL value if the requested value is not set.

The first three arguments

Name	Description
`type`	Specifies the superglobal variable to access. Common values include INPUT_GET, INPUT_POST, and INPUT_COOKIE.
`variable_name`	The name of the value to retrieve.
`filter`	Optional. The constant for the filter to apply.

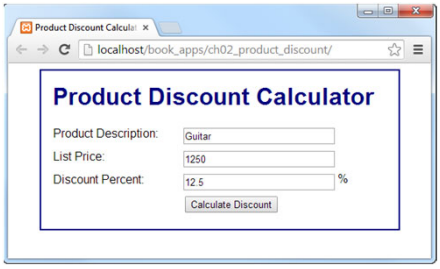
Common constants for filters

Name	Description
`FILTER_VALIDATE_INT`	Validates an integer value.
`FILTER_VALIDATE_FLOAT`	Validates a floating-point (double) value.
`FILTER_VALIDATE_EMAIL`	Validates an email address.
`FILTER_VALIDATE_URL`	Validates a URL.
`FILTER_VALIDATE_BOOLEAN`	Returns a TRUE value for "1", "true", "on", or "yes". Otherwise, it returns a FALSE value.

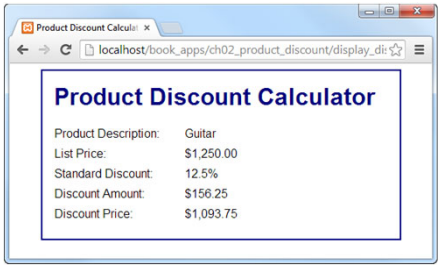
Statements that retrieve values from the superglobal variables

```
$product_description =
    filter_input(INPUT_GET, 'product_description');
// NULL if 'product_description' has not been set
// in the $_GET array

$investment =
    filter_input(INPUT_POST, 'investment',
                 FILTER_VALIDATE_FLOAT);
// NULL if 'investment' has not been set
// in the $_POST array
// FALSE if 'investment' is not a valid float value

$years = filter_input(INPUT_POST, 'years',
                      FILTER_VALIDATE_INT);
// NULL if 'years' has not been set in the $_POST array
// FALSE if 'years' is not a valid integer value
```


The first page (index.html)



The second page (product_discount.php)



The code for the form on the first page

```
<form action="display_discount.php" method="post">

    <div id="data">
        <label>Product Description:</label>
        <input type="text" name="product_description"><br>
        <label>List Price:</label>
        <input type="text" name="list_price"><br>
        <label>Discount Percent:</label>
        <input type="text"
               name="discount_percent"><span>%</span><br>
    </div>

    <div id="buttons">
        <label> </label>
        <input type="submit" value="Calculate Discount"><br>
    </div>

</form>
```

The PHP file (display_discount.php)

```
<?php
    // get the data from the form
    $product_description = filter_input(INPUT_POST,
            'product_description');
    $list_price = filter_input(INPUT_POST, 'list_price');
    $discount_percent = filter_input(INPUT_POST,
            'discount_percent');

    // calculate the discount and discounted price
    $discount = $list_price * $discount_percent * .01;
    $discount_price = $list_price - $discount;

    // apply formatting
    $list_price_f = "$".number_format($list_price, 2);
    $discount_percent_f = $discount_percent."%";
    $discount_f = "$".number_format($discount, 2);
    $discount_price_f = "$".number_format($discount_price, 2);
?>
```


The PHP file (display_discount.php) (continued)

```
<!DOCTYPE html>
<html>

<head>
    <title>Product Discount Calculator</title>
    <link rel="stylesheet" type="text/css" href="main.css">
</head>

<body>
    <main>
        <h1>Product Discount Calculator</h1>

        <label>Product Description:</label>
        <span><?php echo htmlspecialchars(
                $product_description); ?></span>
        <br>
```


The PHP file (display_discount.php) (continued)

```
        <label>List Price:</label>
        <span><?php echo htmlspecialchars(
                $list_price_f); ?></span>
        <br>

        <label>Standard Discount:</label>
        <span><?php echo htmlspecialchars(
                $discount_percent_f); ?></span>
        <br>

        <label>Discount Amount:</label>
        <span><?php echo $discount_f; ?></span>
        <br>

        <label>Discount Price:</label>
        <span><?php echo $discount_price_f; ?></span>
        <br>
    </main>
</body>
</html>
```


The relational operators

Operator	Example
==	$last_name == "Harris" $test_score == 10
!=	$first_name != "Ray" $months != 0
<	$age < 18
<=	$investment <= 0
>	$test_score > 100
>=	$rate / 100 >= 0.1
===	$investment === FALSE $years === NULL
!==	$investment !== FALSE $years !== NULL

The logical operators in order of precedence

Operator	Example
!	!is_numeric($age)
&&	$age > 17 && $score < 70
||	!is_numeric($rate) || $rate < 0

An if statement with no other clauses

```
if ( $price <= 0 ) {
    $message = 'Price must be greater than zero.';
}
```

An if statement with an else clause

```
if ( empty($first_name) ) {
    $message = 'You must enter your first name.';
} else {
    $message = 'Hello ' . $first_name . '!';
}
```

An if statement with else if and else clauses

```
if ( empty($investment) ) {
    $message = 'Investment is a required field.';
} else if ( !is_numeric($investment) )  {
    $message = 'Investment must be a valid number.';
} else if ( $investment <= 0 )  {
    $message = 'Investment must be greater than zero.';
} else {
    $message = 'Investment is valid!';
}
```


A compound conditional expression

```
if ( empty($investment) || !is_numeric($investment) ||
    $investment <= 0 ) {
    $message = 'Investment must be a valid number greater than
    zero.';
}
```

A nested if statement

```
if ( empty($months) || !is_numeric($months)
                    || $months <= 0 ) {
    $message = 'Please enter a number of months > zero.';
} else {
    $years = $months / 12;
    if ( $years > 1 ) {
        $message = 'A long-term investment.';
    } else {
        $message = 'A short-term investment.';
    }
}
```


A while loop that stores the numbers 1 through 5

```
$counter = 1;
while ($counter <= 5) {
    $message = $message . $counter . '|';
    $counter++;
}
// $message = 1|2|3|4|5|
```


A for loop that stores the numbers 1 through 5

```
for ($counter = 1; $counter <= 5; $counter++) {
    $message = $message . $counter . '|';
}
// $message = 1|2|3|4|5|
```


A while loop that calculates the future value of a one-time investment

```
$investment = 1000;
$interest_rate = .1;
$years = 25;
$future_value = $investment;

$i = 1;
while ($i <= $years) {
  $future_value += $future_value * $interest_rate;
  $i++;
}
```


A for loop that calculates the future value of a one-time investment

```
$investment = 1000;
$interest_rate = .1;
$years = 25;
$future_value = $investment;

for ($i = 1; $i <= $years; $i++) {
  $future_value += $future_value * $interest_rate;
}
```

Built-in functions that pass control

```
include($path)
include_once($path)
require($path)
require_once($path)
exit([$status])
die([$status])
```


The include function

```
include 'index.php';    // parentheses are optional
include('index.php');   // index.php in the current
                        // directory
```

The require function

```
require('index.php');   // index.php in the current
                        // directory
```

The exit function

```
exit;                   // parentheses are optional
exit();
exit('Unable to connect to DB.');
                        // passes a message to the browser
```

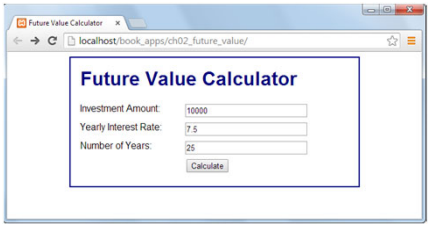

How to pass control to another PHP file in the current directory

```
if ($is_valid) {
    include('process_data.php');
    exit();
}
```

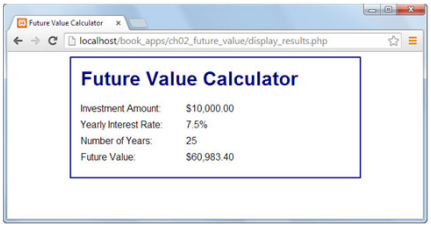
How to navigate up and down directories

```
include('view/header.php');  // down one directory
include('./error.php');      // in the current directory
include('../error.php');     // up one directory
include('../../error.php');  // up two directories
```

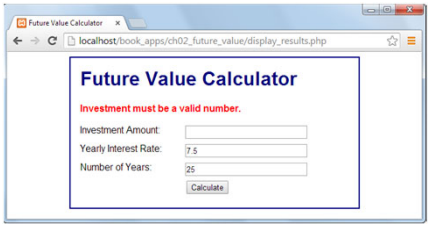

The first page



The second page



The first page with an error message



The index.php file

```
<?php
    //set default value of variables for initial page load
    if (!isset($investment)) { $investment = ''; }
    if (!isset($interest_rate)) { $interest_rate = ''; }
    if (!isset($years)) { $years = ''; }
?>
<!DOCTYPE html>
<html>
<head>
    <title>Future Value Calculator</title>
    <link rel="stylesheet" type="text/css" href="main.css">
</head>

<body>
    <main>
    <h1>Future Value Calculator</h1>
    <?php if (!empty($error_message)) { ?>
        <p class="error"><?php echo htmlspecialchars(
                          $error_message); ?></p>
    <?php } ?>
    <form action="display_results.php" method="post">
```


The index.php file (continued)

```
      <div id="data">
            <label>Investment Amount:</label>
            <input type="text" name="investment"
                   value="<?php echo htmlspecialchars(
                                $investment); ?>">
            <br>
            <label>Yearly Interest Rate:</label>
            <input type="text" name="interest_rate"
                   value="<?php echo htmlspecialchars(
                                $interest_rate); ?>">
            <br>
            <label>Number of Years:</label>
            <input type="text" name="years"
                   value="<?php echo htmlspecialchars($years); ?>">
            <br>
        </div>
        <div id="buttons">
            <label> </label>
            <input type="submit" value="Calculate"><br>
        </div>
    </form>
    </main>
</body></html>
```


The display_results.php file

```
<?php
    // get the data from the form
    $investment = filter_input(INPUT_POST, 'investment',
        FILTER_VALIDATE_FLOAT);
    $interest_rate = filter_input(INPUT_POST, 'interest_rate',
        FILTER_VALIDATE_FLOAT);
    $years = filter_input(INPUT_POST, 'years',
        FILTER_VALIDATE_INT);

    // validate investment
    if ($investment === FALSE ) {
        $error_message = 'Investment must be a valid number.';
    } else if ( $investment <= 0 ) {
        $error_message = 'Investment must be greater than zero.';
```


The display_results.php file (continued)

```
    // validate interest rate
    } else if ( $interest_rate === FALSE )  {
        $error_message = 'Interest rate must be a valid number.';
    } else if ( $interest_rate <= 0 ) {
        $error_message = 'Interest rate must be greater than zero.';

    // validate years
    } else if ( $years === FALSE ) {
        $error_message = 'Years must be a valid whole number.';
    } else if ( $years <= 0 ) {
        $error_message = 'Years must be greater than zero.';
    } else if ( $years > 30 ) {
        $error_message = 'Years must be less than 31.';

    // set error message to empty string if no invalid entries
    } else {
        $error_message = '';
    }
```


The display_results.php file (continued)

```
    // if an error message exists, go to the index page
    if ($error_message != '') {
        include('index.php');
        exit();
    }

    // calculate the future value
    $future_value = $investment;
    for ($i = 1; $i <= $years; $i++) {
        $future_value += $future_value * $interest_rate * .01;
    }

    // apply currency and percent formatting
    $investment_f = '$'.number_format($investment, 2);
    $yearly_rate_f = $interest_rate.'%';
    $future_value_f = '$'.number_format($future_value, 2);
?>
```

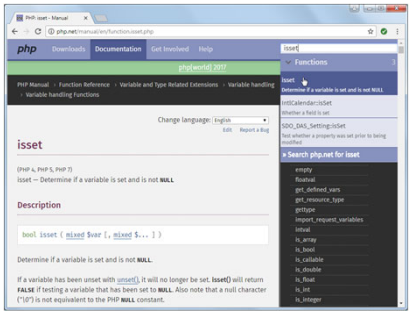

The display_results.php file (continued)

```
<!DOCTYPE html>
<html>
<head>
    <title>Future Value Calculator</title>
    <link rel="stylesheet" type="text/css" href="main.css">
</head>
<body>
    <main>
        <h1>Future Value Calculator</h1>

        <label>Investment Amount:</label>
        <span><?php echo $investment_f; ?></span><br>

        <label>Yearly Interest Rate:</label>
        <span><?php echo $yearly_rate_f; ?></span><br>

        <label>Number of Years:</label>
        <span><?php echo $years; ?></span><br>

        <label>Future Value:</label>
        <span><?php echo $future_value_f; ?></span><br>
    </main>
</body></html>
```

The URL for the PHP documentation

http://php.net/docs.php

Documentation for the isset() function



How to access the PHP manual

- On the first page of the website, click on the name of the language that you want to use. That will access the first page of the PHP manual, which includes a table of contents.

How to use the PHP manual

- Scroll down the contents until you find the link you're looking for, click on it, and continue this process until the right information is displayed.
- As you display information, you can use the links in the right pane of the window to display other topics at the same level.

How to find the documentation for a function when you know its name

- Type the function name in the Search text box and select from the autocomplete list.