Fundamentals of Web Development

Third Edition by Randy Connolly and Ricardo Hoar

Chapter 12

Server-Side Development 1: PHP

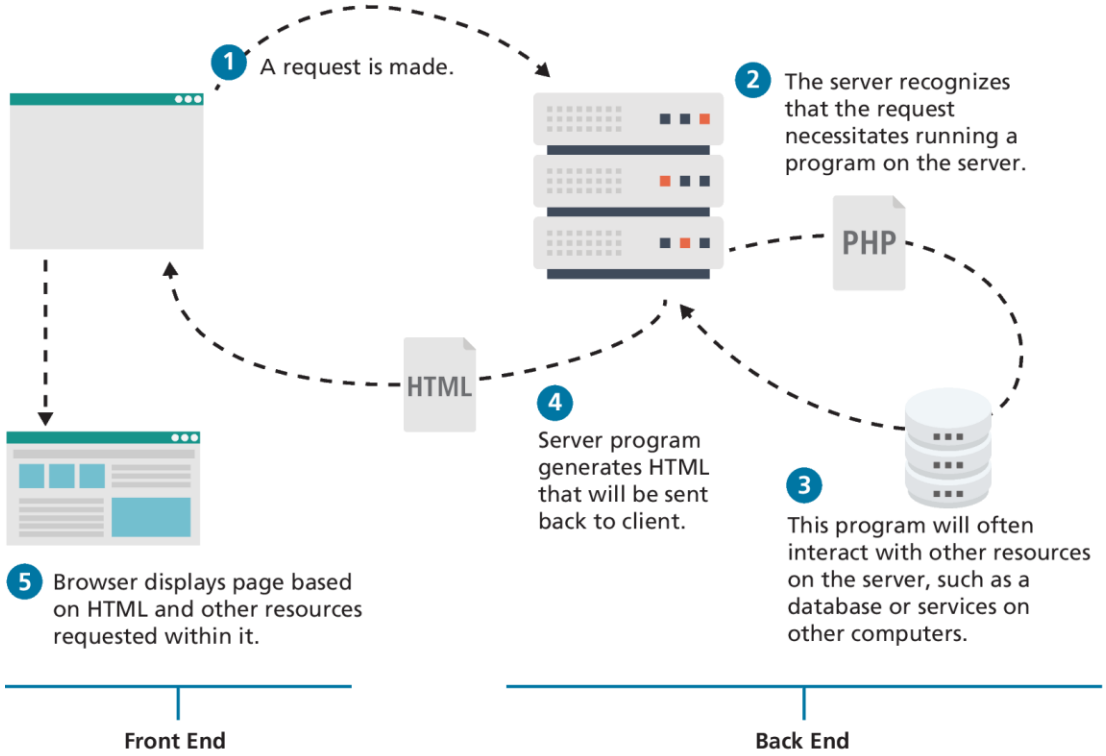
In this chapter you will learn . . .

- What is server-side development
- PHP language fundamentals
- PHP arrays, objects, and functions
- Using PHP superglobal arrays to access HTTP content

Front End versus Back End

Server-side technologies provide access to data sources, handled security, and allowed web sites to interact with external services such as payment systems.

Traditionally, most sites made use programs running on the server-side to programmatically generate the HTML sent to the browser.

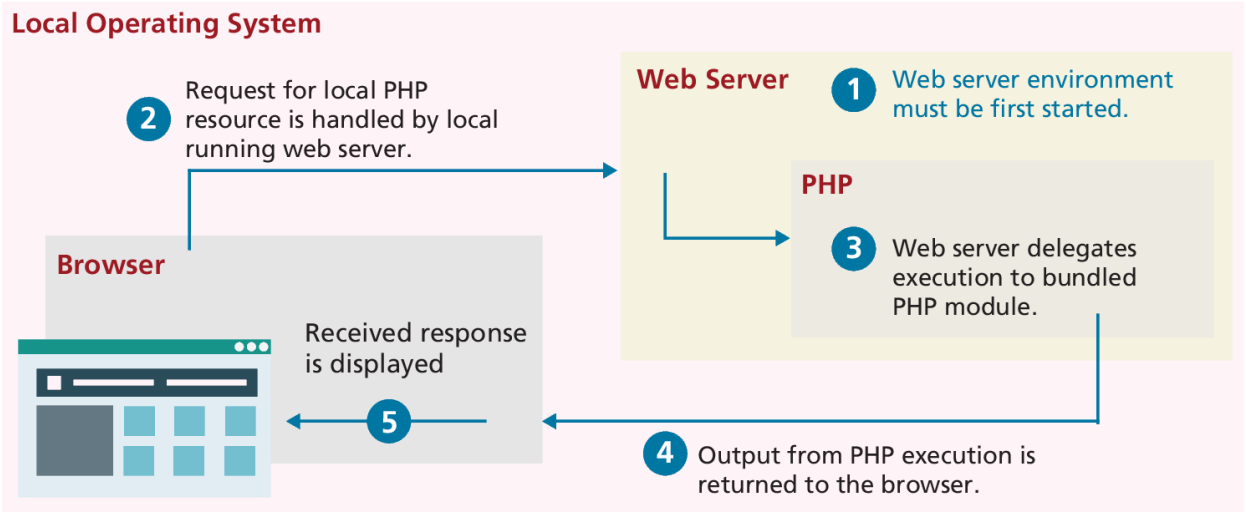


Common Server-Side Technologies

- **ASP (Active Server Pages).** This was Microsoft's first server-side technology (also called ASP Classic).
- **ASP.NET.** This replaced Microsoft's older ASP technology.
- **JSP (Java Server Pages).** JSP uses Java as its programming language and like ASP.NET it uses an explicit object-oriented approach and is used in large enterprise web systems and is integrated into the J2EE environment.
- **Node.js (**or just **Node).** Uses JavaScript on the server side
- **Perl.** excels in the manipulation of text.
- **PHP.** Like ASP, PHP is a dynamically typed language that can be embedded directly within the HTML
- **Python.** This terse, object-oriented programming language has many uses, including being used to create web applications.
- **Ruby on Rails.** This is a web development framework that uses the Ruby programming language.

Running PHP locally

Installing Apache, PHP, and MySQL for Local Development



PHP Language Fundamentals

PHP is a a dynamically typed language (with optional static typing), and provides classes and functions in a way consistent with other object-oriented languages such as C++, C#, and Java.

The syntax for loops, conditionals, and assignment is identical to JavaScript, only differing when you get to functions, classes, and in how you define variables.

PHP Tags and comments

PHP code can be embedded directly within an HTML file. However, instead of having an **.html** extension, a PHP file will usually have the extension **.php**.

Code must be contained within an opening **<?php** tag and a matching closing **?>** tag. Code within the <?php and the ?> tags is interpreted and executed, while any code outside the tags is echoed directly out to the client.

It is very common practice (especially when first learning PHP) for a PHP file to have HTML markup and PHP programming logic woven together.

PHP uses the same commenting mechanisms as JavaScript, namely multi-line block comments using /* */ or end-of-line comments using //

PHP Tag Example

```
<?php
      $user = "Randy";
?>
<!DOCTYPE html>
<html>
      <body>
      <h1>Welcome <?php echo $user; ?></h1>
      <p>
The server time is
<?php
echo "<strong>";
echo date("H:i:s");
echo "</strong>";
?>
</p></body></html>
```

LISTING 12.1 Php Tags

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome Randy</h1>
<p>
The server time is <strong>02:59:09</strong>
</p>
</body>
</html>
```

LISTING 12.2 Output (HTML) from PHP script in Listing 12.1

Variables and Data Types

Variables in PHP are **dynamically typed**. The PHP engine makes a best guess as to the intended type based on what it is being assigned.

Variables are also **loosely typed** in that a variable can be assigned different data types over time.

- **Boolean** A logical true or false value
- **Integer** Whole numbers
- **Float** Decimal numbers
- **String** Letters
- **Array** A collection of data of any type (covered in the next chapter)
- **Object** Instances of classes

PHP constants

- A constant can be defined anywhere but is typically defined near the top of a PHP file via the define() function. The define() function generally takes two parameters: the name of the constant and its value. Notice that once it is defined, it can be referenced without using the $ symbol.

```
<?php
// uppercase for constants is a programming convention
define("DATABASE_LOCAL", "localhost");
define("DATABASE_NAME", "ArtStore");
define("DATABASE_USER", "Fred");
define("DATABASE_PASSWD", "F5^7%ad");
// ...
// notice that no $ prefaces constant names
$db = new mysqli( DATABASE_LOCAL,
DATABASE_NAME, DATABASE_USER,
DATABASE_PASSWD);
?>
```

LISTING 12.3 PHP constants

Writing to Output

To output something that will be seen by the browser, you can use the **echo**() function.

```
echo("hello");
echo "hello"; //alternate version (no parenthesis)
```

Strings can be appended together using the concatenate operator, which is the period (.) symbol. Consider the following code:

```
$username = "Ricardo";
echo "Hello " . $username;
```

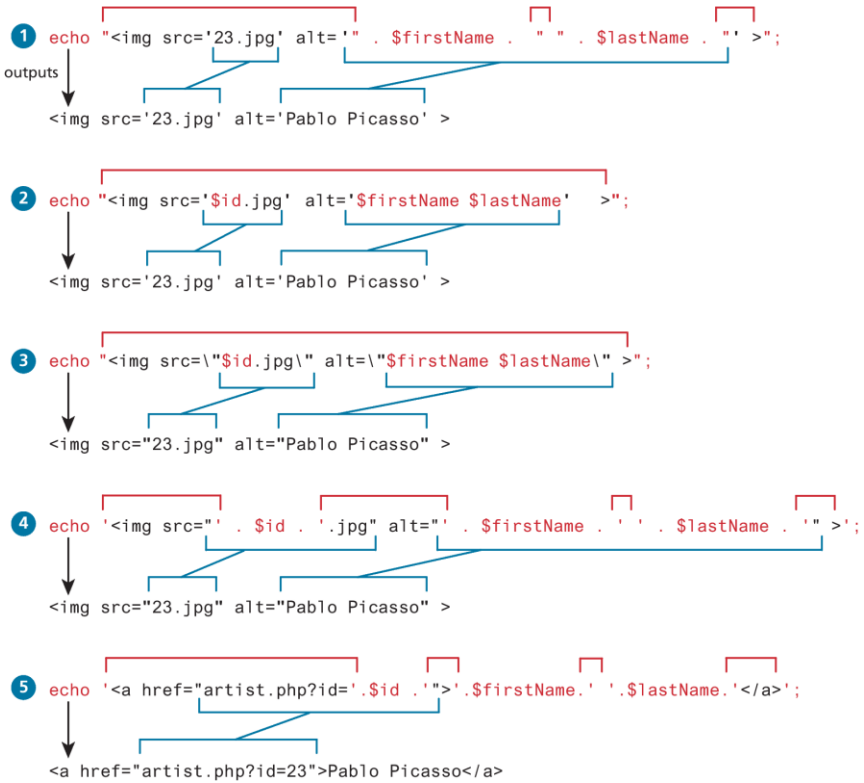
will output **Hello Ricardo** to the browser.

PHP quote and concatenation approaches

```
<?php
$firstName = "Pablo";
$lastName = "Picasso";

/*Example one:
These two lines are equivalent. Notice that you can reference
PHP variables within a string literal defined with double quotes.
The resulting output for both lines is:
<em>Pablo Picasso</em>*/

echo "<em>" . $firstName . " ". $lastName. "</em>";
echo "<em> $firstName $lastName </em>";

/* Example two: These two lines are also equivalent. Notice that
you can use either the single quote symbol or double quote
symbol for string literals. */

echo "<h1>";
echo '<h1>';

/* Example three: These two lines are also equivalent. In the
second example, the escape character (the backslash) is used to
embed a double quote within a string literal defined within
double quotes. */

echo '<img src="23.jpg" >';
echo "<img src=\"23.jpg\" >";
?>
```

LISTING 12.4 PHP quote usage and concatenation approaches

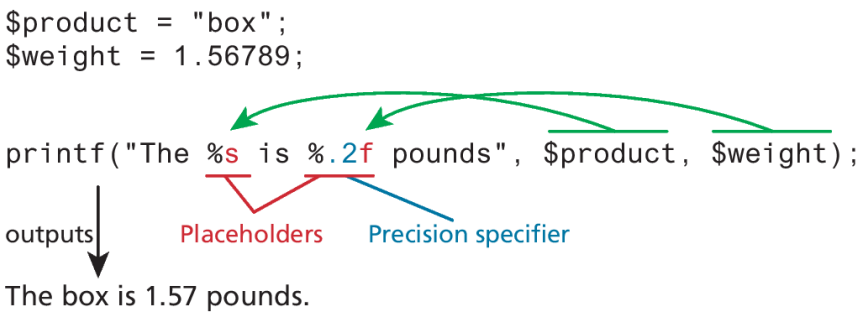
More concatenation examples



printf

The printf() function is derived from the C programming language is nearly ubiquitous in programming, appearing in many languages including Java, MATLAB, Perl, Ruby, and others.

The printf() function also allows a developer to apply special formatting, for instance, specific date/time formats or number of decimal places. It takes at least one parameter, which is a string, and that string optionally references parameters, which are then integrated into the first string by placeholder substitution.



Program Control

Just as with most other programming languages there are a number of conditional and iteration constructs in PHP.

- if . . . else
- switch . . . case
- while and do . . . while
- for

If…else

The syntax for conditionals in PHP is identical to that of JavaScript.

The condition to test is contained within `()` brackets with the body contained in `{}` blocks. Optional `else if` statements can follow, with an optional `else` ending the branch.

```
// if statement
if ( $hourOfDay > 6 && $hourOfDay < 12) {
      $greeting = "Good Morning";
}
else if ($hourOfDay == 12) { // optional else if
      $greeting = "Good Noon Time";
}
else { // optional else branch
      $greeting = "Good Afternoon or Evening";
}
```

LISTING 12.7 Conditional snippet of code using if . . . else

PHP and HTML in the same script

```
<?php if ($userStatus == "loggedin") { ?>
   <a href="account.php">Account</a>
   <a href="logout.php">Logout</a>
<?php } else { ?>
   <a href="login.php">Login</a>
   <a href="register.php">Register</a>
<?php } ?>
```

```
<?php
// equivalent
if ($userStatus == "loggedin") {
   echo '<a href="account.php">Account</a> ‘;
   echo '<a href="logout.php">Logout</a>';
}
else {
   echo '<a href="login.php">Login</a> ‘;
   echo '<a href="register.php">Register</a>';
}
?>
```

LISTING 12.8 Combining PHP and HTML in the same script

switch . . . case

The **switch** statement is similar to a series of **if ... else** statements.

```
switch ($artType) {
case "PT":
        $output = "Painting";
        break;
case "SC":
        $output = "Sculpture";
        break;
default:
        $output = "Other";
}
```

```
// equivalent
if ($artType == "PT")
        $output = "Painting";
else if ($artType == "SC")
        $output = "Sculpture";
else
        $output = "Other";
```

LISTING 12.9 Conditional statement using switch and the equivalent if-else

while and do . . . while

In the **while** loop, the condition is tested at the beginning of the loop; in the **do ... while** loop the condition is tested at the end of each iteration of the loop.

```
$count = 0;
while ($count < 10) {
    echo $count;
    $count++;
}
$count = 0;
```

```
do {
    echo $count;
    // this one increments the count by 2
each time
    $count = $count + 2;
} while ($count < 10);
```

LISTING 12.10 The while loops

For loops

The for loop in PHP has the same syntax as the for loop in JavaScript that we examined in Chapter 8. For loops contain the same loop initialization, condition, and postloop operations as in JavaScript.

```php
// this one increments the value by 5 each time
for ($count=0; $count < 100; $count+=5) {
   echo $count;
}
```

```php
// this one increments the count by 1 each time
for ($count=0; $count < 10; $count++) {
   echo $count;
}
```

LISTING 12.11 The for loops

Alternate Syntax for Control Structures

PHP has an alternative syntax for most of its control structures.

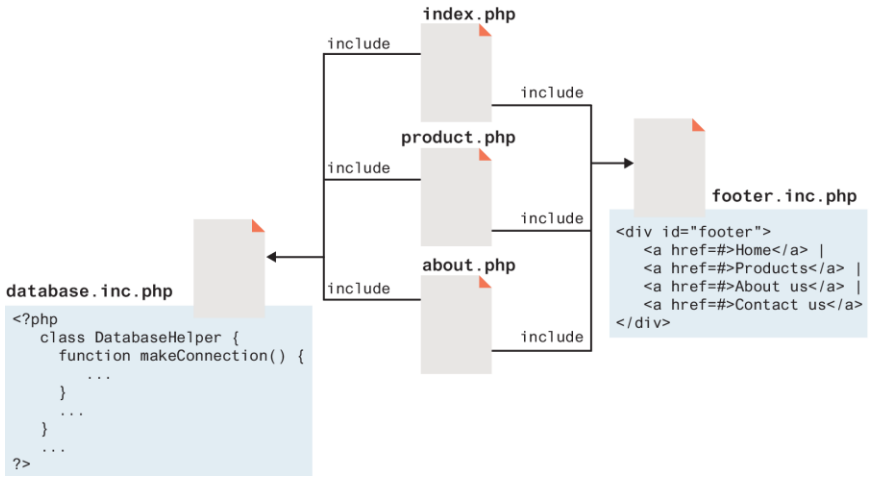
In this alternate syntax, the colon (:) replaces the opening curly bracket, while the closing brace is replaced with endif;, endwhile;, endfor;, endforeach;, or endswitch;.

```
<?php if ($userStatus == "loggedin") : ?>
                <a href="account.php">Account</a>
                <a href="logout.php">Logout</a>
<?php else : ?>
                <a href="login.php">Login</a>
                <a href="register.php">Register</a>
<?php endif; ?>
```

LISTING 12.12 Alternate syntax for control structures

Include Files

PHP has the ability to include or insert content from one file into another. Include files provide a mechanism for reusing both markup and PHP code




```
include "somefile.php";
include_once "somefile.php";
require "somefile.php";
require_once "somefile.php";
```

Functions

Just like with JavaScript, a **function** in PHP contains a small bit of code that accomplishes one thing.

- A **user-defined function** is one that you, the programmer, define.
- A **built-in function** is one of the functions that come with the PHP environment (or with one of its extensions).

Function Syntax

Functions can return values to the caller, or not return a value.

They can be set up to take or not take parameters.

Function definition requires the use of the `function` keyword followed by the function's name, round ( ) brackets for parameters, and then the body of the function inside curly { } brackets.

```
function getNiceTime(){
    return date("H:i:s");
}
```

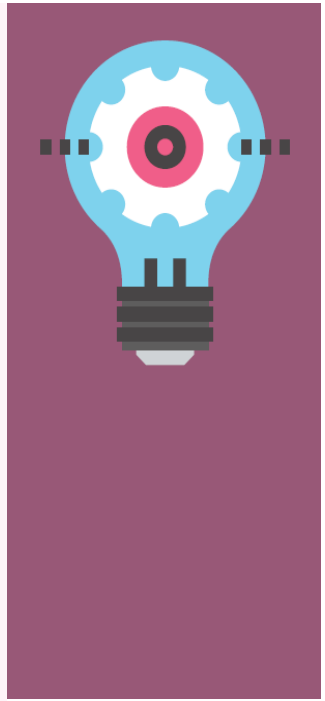
Return Type Declaration

PROTIP

In PHP 7.0, the ability to *explicitly* define a return type for a function was added. A **Return Type Declaration** explicitly defines a function's return type by adding a colon and the return type after the parameter list when defining a function.

```
function mustReturnString(string $name) : string {
        return "hello ". $name;
}
```

LISTING 12.15 Return type declaration in PHP 7.0



Invoking a Function

To invoke or call a function you must use its name with the () brackets. Since getNiceTime() returns a string, you can assign that return value to a variable, o echo that return value directly, as shown in the following example:

```
$output = getNiceTime();
echo getNiceTime();
```

If the function doesn't return a value, you can just call the function:

```
outputFooterMenu();
```

Parameters

Parameters are the mechanism by which values are passed into functions. To illustrate, let us write another version of **getNiceTime**() that takes an integer as a parameter to control whether to show seconds. You will call the parameter **showSeconds**

```
function getNiceTime($showSeconds) {
      if ($showSeconds==true)
            return date("H:i:s");
      else
            return date("H:i");
}
```

LISTING 12.16 A function with a parameter

Parameter Default Values

In PHP you can set **parameter default values** for any parameter in a function. However, once you start having default values, all subsequent parameters must also have defaults.

```
function getNiceTime($showSeconds=true) {
      if ($showSeconds==true)
            return date("H:i:s");
      else
            return date("H:i");
}
```

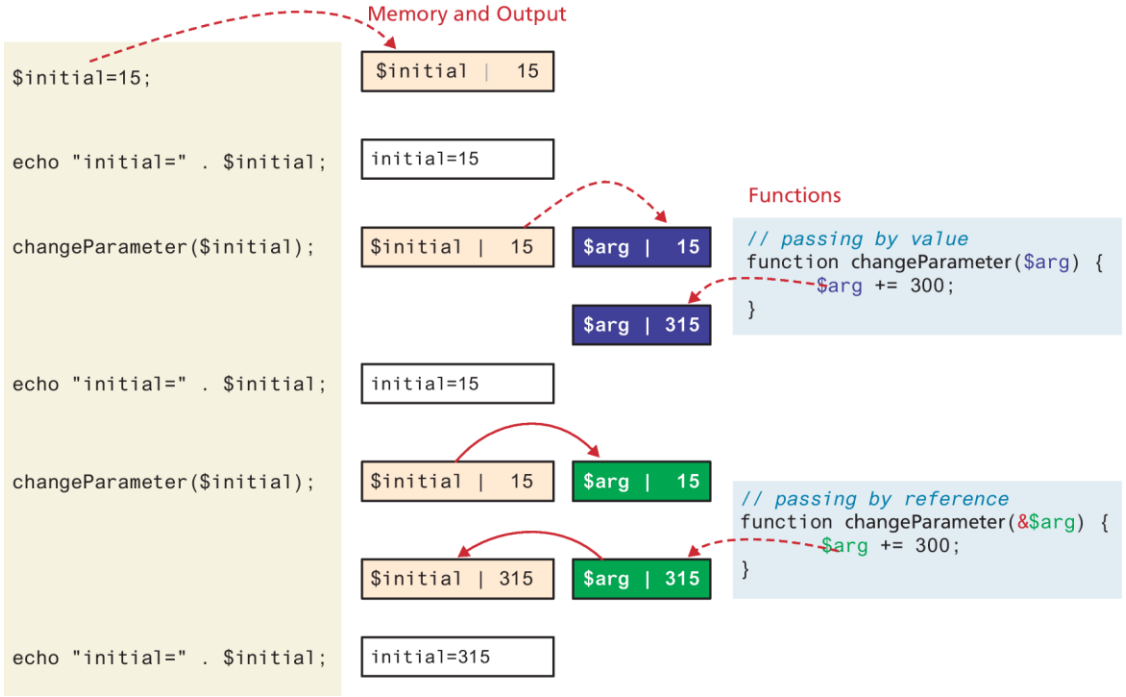
LISTING 12.17 A function with a parameter default

Passing Parameters by Reference

By default, arguments passed to functions are **passed by value** in PHP.

PHP also allows arguments to functions to be **passed by reference**

The mechanism in PHP is to add an ampersand (&) symbol next to the parameter name in the function declaration.



Parameter-type declaration

It is now possible to require that a particular parameter be of a particular type. To add a type to a parameter, add a type specification (*int*, *float*, *string*, *bool*, *callable*, or any class name you have defined) before the parameter name.

```
function getNiceTime(bool $showSeconds=1) {
        if ($showSeconds==true)
                return date("H:i:s");
        else
                return date("H:i");
}
```

LISTING 12.20 Using a parameter-type declaration

Variable Scope within Functions

All variables defined within a function have **function scope**, meaning that they are only accessible. within the function. It might be surprising though to learn that, *unlike JavaScript, any variables created outside of the function in the main script are unavailable within a function*.

PHP does allow variables with **global scope** to be accessed within a function using the global keyword, though generally speaking, its usage is discouraged.

Arrays

Unlike most other programming languages (including JavaScript), in PHP an array is actually an ordered map, which associates each value in the array with a key.

This allows you to use arrays in PHP in a manner similar to other languages' arrays, but you can also use them like other languages' collection classes.

- **Array keys** restricted to integers and strings
- **Array values**, unlike keys, are not restricted to integers and strings. They can be any object, type, or primitive supported in PHP.

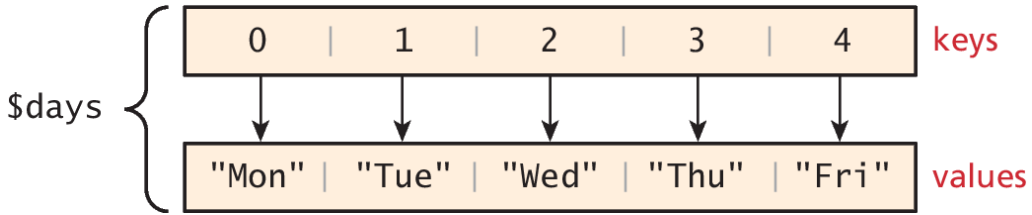
Defining an Array

The following declares an empty array named days:

```
$days = array();
```

To define the contents of an array as strings for the days of the week, you declare it using either of two following syntaxes:

```
$days = array("Mon","Tue","Wed","Thu","Fri");
$days = ["Mon","Tue","Wed","Thu","Fri"]; // alternate syntax
```



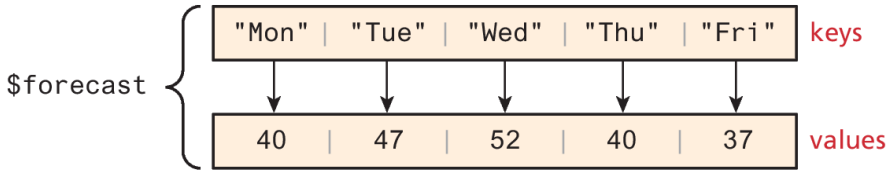
Accessing an Array

- Explicit control of the keys and values opens the door to keys that do not start at 0, are not sequential, and that are not even integers (but rather strings).

```
$forecast = array("Mon" => 40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" => 37);
```

key

value

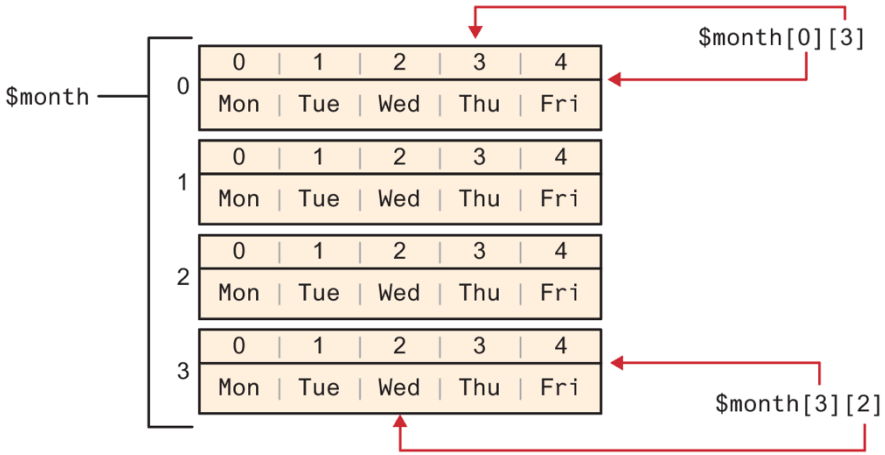



```
echo $forecast["Tue"];  // outputs 47
echo $forecast["Thu"];  // outputs 40
```

Multidimensional Arrays

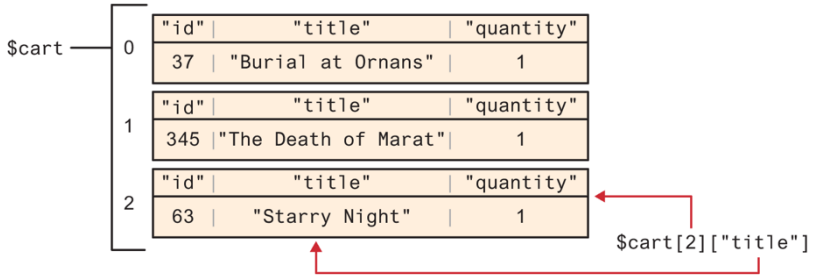
```
$month = array(
        array("Mon","Tue","Wed","Thu","Fri"),
        array("Mon","Tue","Wed","Thu","Fri"),
        array("Mon","Tue","Wed","Thu","Fri"),
        array("Mon","Tue","Wed","Thu","Fri")
);
```

LISTING 12.22 Multidimensional arrays



Multidimensional Arrays (ii)

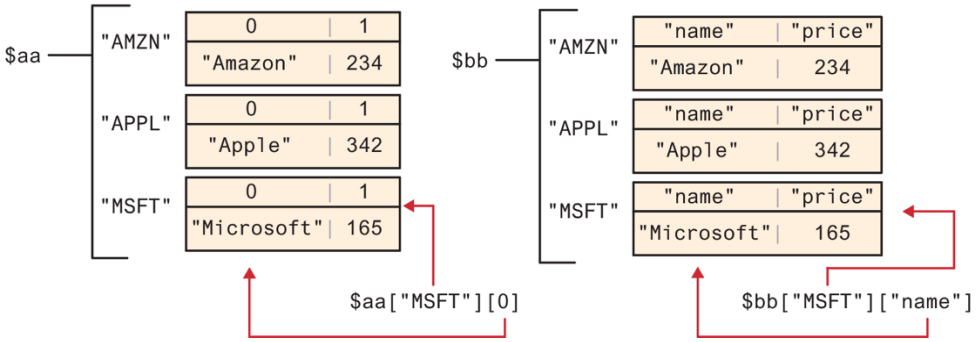
```
$cart = [];
$cart[] = array("id" => 37,
                "title" => "Burial at Ornans",
                "quantity" => 1);
$cart[] = array("id" => 345,
                 "title" => "The Death of Marat",
                "quantity" => 1);
$cart[] = array("id" => 63,
                "title" => "Starry Night",
                "quantity" => 1);
```



LISTING 12.22 Multidimensional arrays

Multidimensional Arrays (iii)

```
$stocks = [
["AMZN", "Amazon"],
["APPL", "Apple"],
["MSFT", "Microsoft"]
];

$aa = [
"AMZN" => ["Amazon", 234],
"APPL" => ["Apple", 342],
"MSFT" => ["Microsoft", 165]
];

$bb = [
"AMZN" => ["name" =>"Amazon", "price" => 234],
"APPL" => ["name" => "Apple", "price" => 342],
"MSFT" => ["name" => "Microsoft", "price" => 165]
];
```



LISTING 12.22 Multidimensional arrays

Iterating through an Array

```php
// while loop
$i=0;
while ($i < count($days)) {
      echo $days[$i] . "<br>";
$i++;
}

// for loop
for ($i=0; $i<count($days); $i++) {
echo $days[$i] . "<br>";
}
```

LISTING 12.23 Iterating through an array using while, do while, and for loops

```php
// iterating through the values
foreach ($forecast as $value) {
      echo $value . "<br>";
}

// iterating through the values AND
the keys
foreach ($forecast as $key => $value){
      echo "day[" . $key . "]=" . $value;
}
```

LISTING 12.24 Iterating through an associative array using a foreach loop

Adding and Deleting Elements

In PHP, arrays are dynamic, that is, they can grow or shrink in size.

An element can be added to an array simply by using a key/index that hasn't been used, as shown below:

```
$days[5] = "Sat";
```

A new element can be added to the end of any array using empty square brackets after the array name, as follows:

```
$days[] = "Sun";
```

You can also explicitly delete array elements using the unset() function

Checking if a Value Exists

To check if a value exists for a key, you can therefore use the **isset**() function, which returns true if a value has been set, and false otherwise

```
$oddKeys = array(1 => "hello", 3 => "world", 5 => "!");
if (isset($oddKeys[0])) {
        // The code below will never be reached since
        //$oddKeys[0] is not set!
        echo "there is something set for key 0";
}

if (isset($oddKeys[1])) {
        // This code will run since a key/value pair was defined
        //for key 1
        echo "there is something set for key 1, namely ". $oddKeys[1];
}
```

LISTING 12.26 Illustrating nonsequential keys and usage of isset( )

Classes and Objects

Classes outline properties and methods like a blueprint. Each variable created from a class is called an object or **instance**, and each object maintains its own set of variables, an behaves (largely) independently from the class once created.

Book class

Defines properties such as:
title, author, and number of pages.

Objects (or instances of the Book class)

Each instance has its own title, author, and number of pages property values.

Defining and instantiating Classes

The PHP syntax for **defining** a class uses the class keyword followed by the class name and { } braces.

```
class Artist {
        public $firstName;
        public $lastName;
        public $birthDate;
        public $birthCity;
        public $deathDate;
}
```

LISTING 12.27 A simple Artist class

Properties

instantiate objects using the new keyword

$picasso = **new** Artist();

$dali = **new** Artist();

You can access and modify the properties of each object separately using its variable name and an arrow **(->)**,

```
$picasso = new Artist();
$dali = new Artist();
$picasso->firstName = "Pablo";
$picasso->lastName = "Picasso";
$picasso->birthCity = "Malaga";
$picasso->birthDate = "October 25 1881";
$picasso->deathDate = "April 8 1973";
```

LISTING 12.28 Instantiating and using objects

Constructors

You should therefore define **constructors**, which lets you specify parameters during instantiation to initialize the properties within a class right away.

In PHP, constructors are defined as functions with the name **__construct().**

(Note: there are *two* underscores_ before the word construct.)

```
class Artist {
// variables from previous listing still go here
...
    function __construct($firstName, $lastName,
                         $city, $birth,$death=null) {
        $this->firstName = $firstName;
        $this->lastName = $lastName;
        $this->birthCity = $city;
        $this->birthDate = $birth;
        $this->deathDate = $death;
    }
}
```

LISTING 12.29 A constructor added to the class definition

Methods

Methods define the tasks each instance of a class can perform and are useful since they associate behavior with objects.

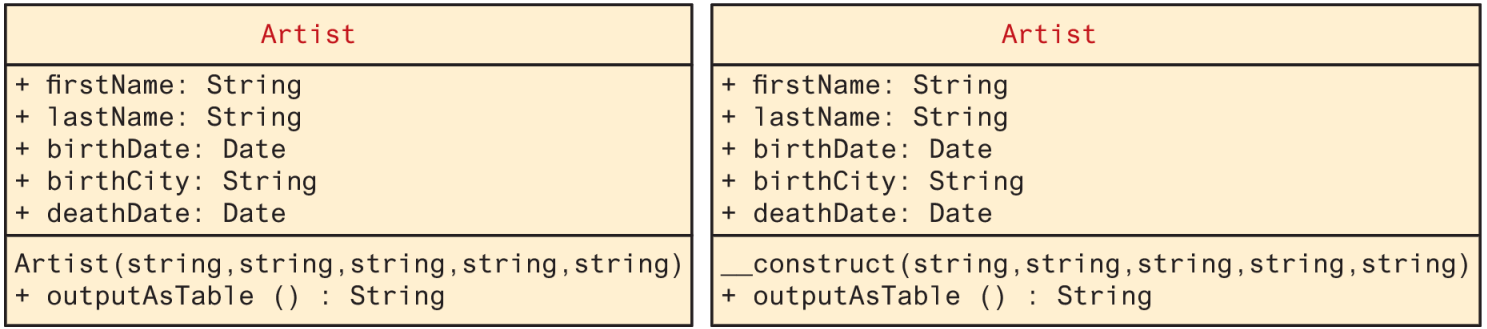
Call the method as follows:

```
$picasso = new Artist(. . .)

echo $picasso->outputAsTable();
```

```
class Artist {
    public function outputAsTable() {
        $table = "<table>";
        $table .= "<tr><th colspan='2'>";
        $table .= $this->firstName . " " . $this->lastName;
        $table .= "</th></tr>";
        $table .= "<tr><td>Birth:</td>";
        $table .= "<td>" . $this->birthDate;
        $table .= "(" . $this->birthCity . ")</td></tr>";
        $table .= "<tr><td>Death:</td>";
        $table .= "<td>" . $this->deathDate . "</td></tr>";
        $table .= "</table>";
        return $table;
    }
}
```

LISTING 12.30 Method definition

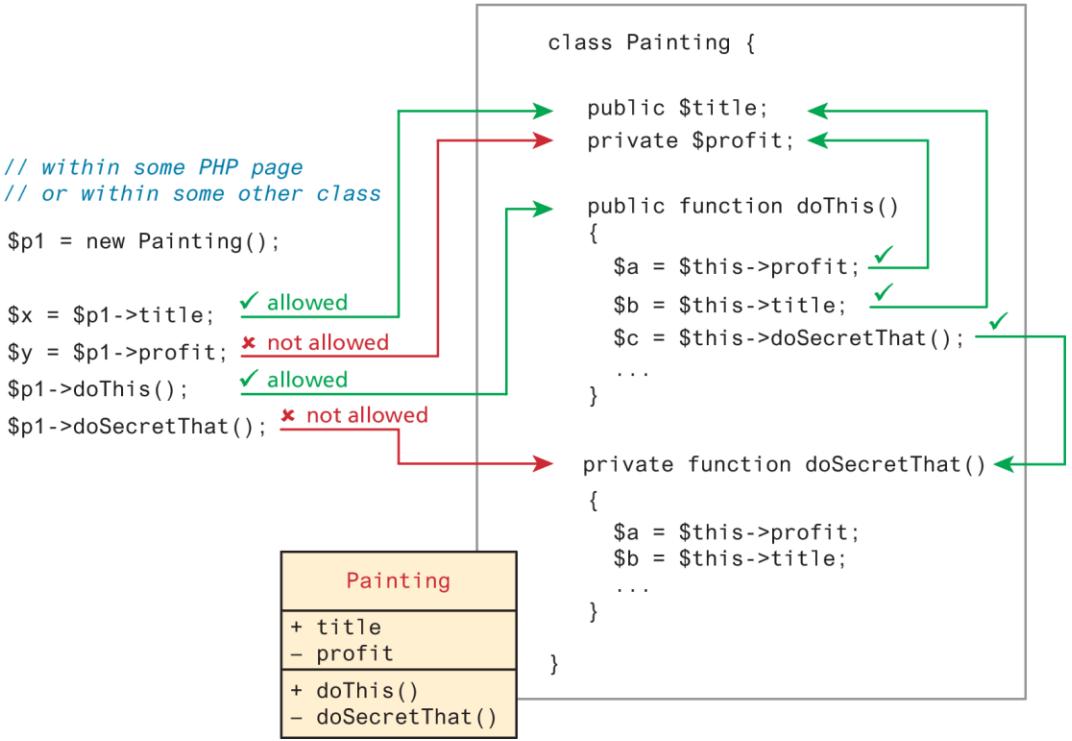
Sample ways to diagram a class using UML



Visibility

The **visibility** of a property or method determines the accessibility of a class member

A property or method and can be set to **public**, **private**, or **protected**



Static Members

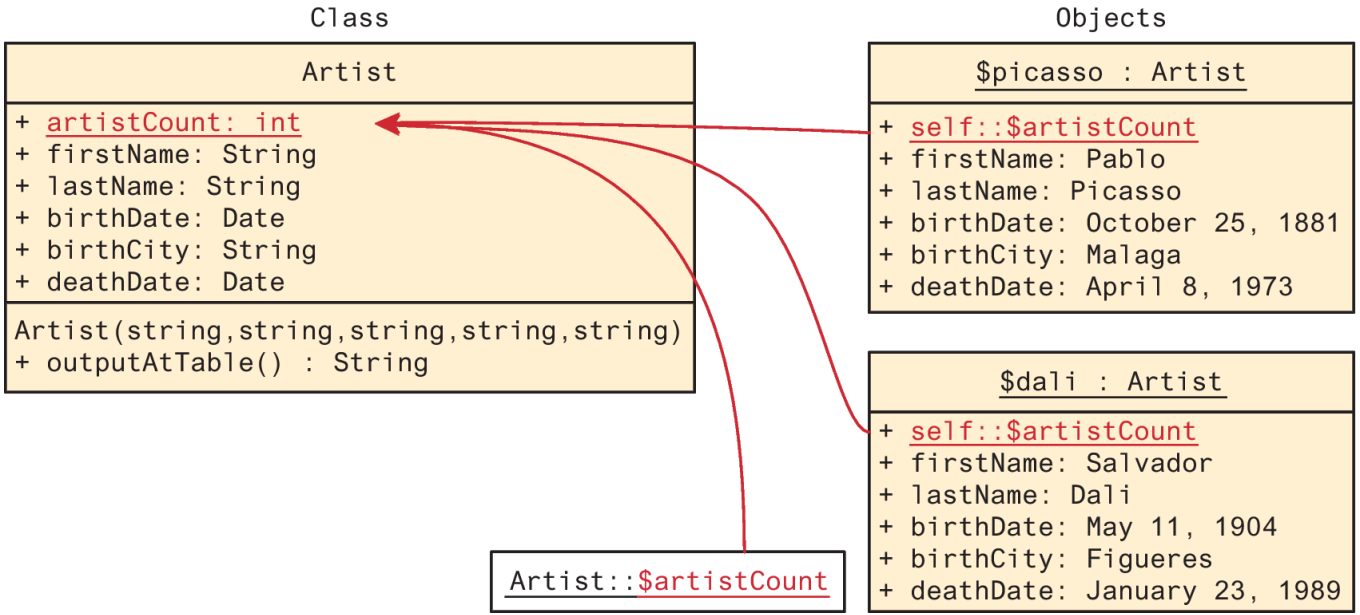
Static member is a property or method that all instances of a class share.

To illustrate how a static member is shared between instances of a class, we will add the static property **artistCount** to our **Artist** class, and use it to keep a count of how many Artist objects are currently instantiated

```
class Artist {
        public static $artistCount = 0;
        public $firstName;
        public $lastName;
        public $birthDate;
        public $birthCity;
        public $deathDate;
        function __construct($firstName, $lastName,
                                        $city,$birth, $death=null) {
                $this->firstName = $firstName;
                $this->lastName = $lastName;
                $this->birthCity = $city;
                $this->birthDate = $birth;
                $this->deathDate = $death;
                self::$artistCount++;
        }
}
```

LISTING 12.31 Class definition modified with static members

A static property in UML



Inheritance

Inheritance enables you to create new PHP classes that reuse, extend, and modify the behavior that is defined in another PHP class.

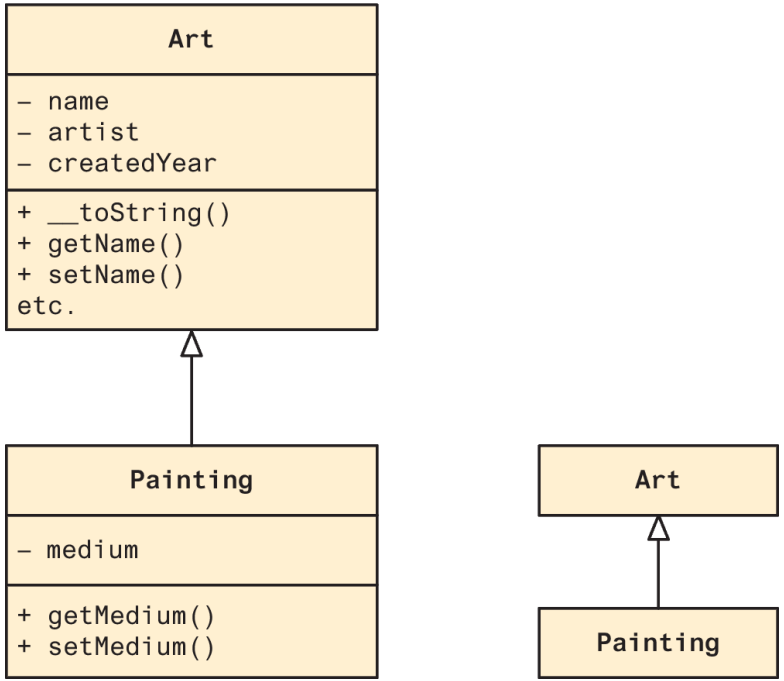
- A class that is inheriting from another class is said to be a **subclass** or a derived class.
- The class that is being inherited from is typically called a **superclass** or a base class.
- Just as in Java, a PHP class is defined as a subclass by using the extends keyword.

```
class Painting extends Art { . . . }
```

UML showing inheritance

Both references below work because it is *as if* the base class public members are defined within the subclass.

```
$p = new Painting();

echo $p->getName(); // defined in base class

echo $p->getMedium(); // defined in subclass
```



$_GET and $_POST Superglobal Arrays

PHP uses special predefined associative arrays called **superglobal arrays** that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information

$GLOBALS Array for storing data that needs superglobal scope

$_COOKIES Array of cookie data passed to page via HTTP request

$_ENV Array of server environment data

$_FILES Array of file items uploaded to the server

$_GET Array of query string data passed to the server via the URL

$_POST Array of query string data passed to the server via the HTTP header

$_REQUEST Array containing the contents of $_GET, $_POST, and $_COOKIES

$_SESSION Array that contains session data

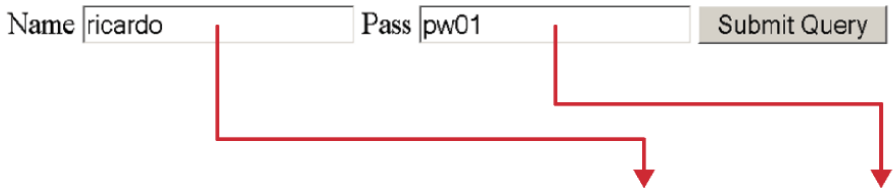
$_SERVER Array containing information about the request and the server

Illustration of flow into $_GET array

HTML (client)

```
<form action="processLogin.php" method="GET">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser (client)



HTTP request

```
GET processLogin.php?uname=ricardo&pass=pw01
```

PHP (server)

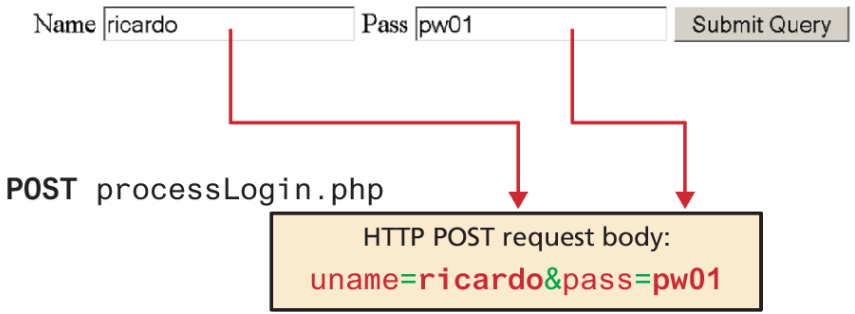
```
// within processLogin.php
echo $_GET["uname"]; // outputs ricardo
echo $_GET["passv    // outputs pw01
```

Illustration of flow into $_POST array

HTML (client)

```
<form action="processLogin.php" method="POST">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser (client)



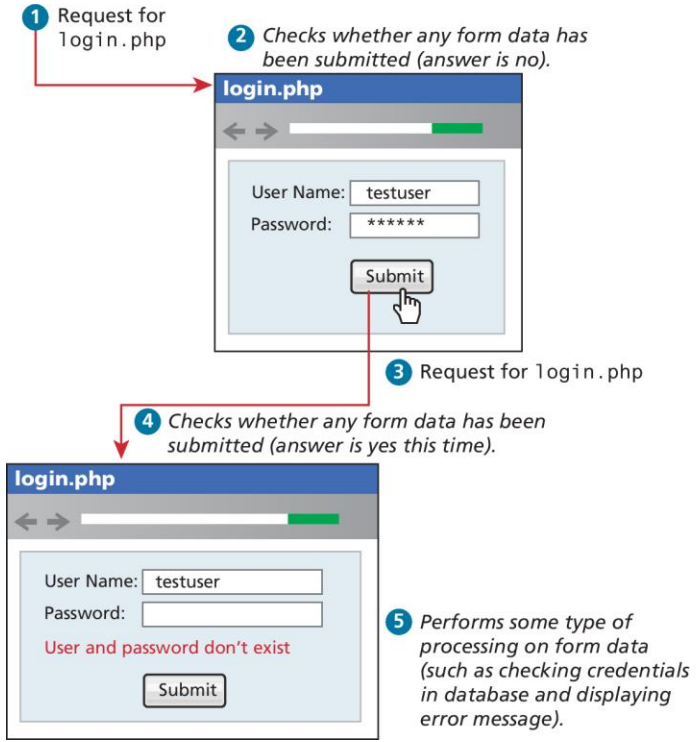
HTTP request

PHP (server)

```
//File processLogin.php
echo $_POST["uname"]; //outputs "ricardo";
echo $_POST["pass"];  //outputs "pw01";
```

Determining If Any Data Sent

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if ( isset($_POST["uname"]) && isset($_POST["pass"]) )
    {
    // handle the posted data.
    echo "handling user login now ...";
    echo "... here we could redirect or authenticate ";
    echo " and hide login form or something else";
    }
}
```



Accessing Form Array Data

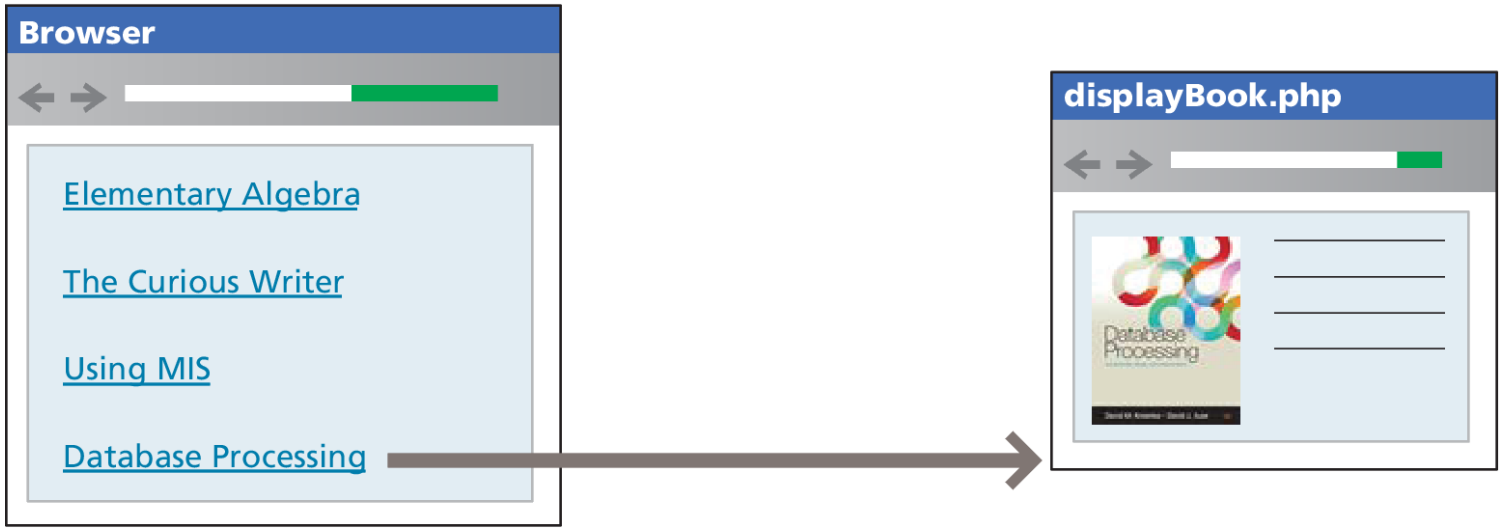
Sometimes in HTML forms, you might have multiple values associated with a single name. Unfortunately, if the user selects more than one day and submits the form, the $_GET['day'] value in the superglobal array *will only contain the last value from the list* that was selected.

To overcome this limitation, you must change the HTML in the form. In particular, you will have to change the name attribute for each checkbox from day to day[]

```
echo "You submitted " . count($_GET['day']) . "values";
            foreach ($_GET['day'] as $d) {
            echo $d . " <br>";
}
```

LISTING 12.34 PHP code to display an array of checkbox variables

Using Query Strings in Hyperlinks




```
<a href="displayBook.php?isbn=0132145375">Database Processing</a>
```

Query string

Sanitizing Query Strings

One of the most important things to remember about web development is that you should actively distrust all user input.

The process of checking user input for incorrect or missing information is sometimes referred to as the process of **sanitizing user inputs**.

```
// This uses a database API ... we will learn about it in Chapter 14
$pid = mysqli_real_escape_string($link, $_GET['id']);
if ( is_int($pid) ) {
        // Continue processing as normal
}
else {
        // Error detected. Possibly a malicious user
}
```

LISTING 12.35 Simple sanitization of query string values

Working with the HTTP Header

So far in this chapter, PHP has been used to modify the response sent back to the browser. In PHP, echo statements adds content *after* the HTTP response header.

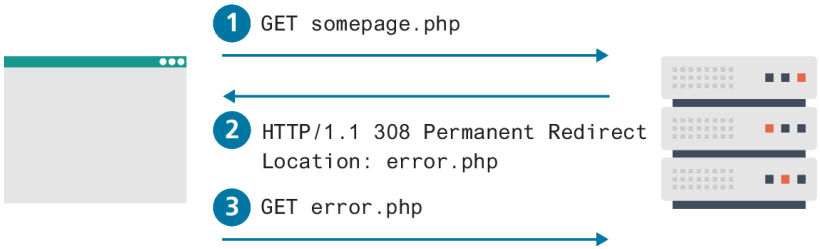
It is possible in PHP to modify the response header using the header() function, but why would we?

- Redirecting Using Location Header
- Setting the Content-Type Header

Redirecting Using Location Header

One of the most common uses of this function in PHP is to redirect. For instance, a PHP page might redirect to an error page when an expected querystring parameter is missing

```
<?php
        if (! isset($_GET['id']) {
                header("Location: error.php");
        }

...?>
```



Setting the Content-Type Header

The Content-Type HTTP header is used to tell the browser what type of content (using a MIME type) it is receiving in the response.

Normally, the PHP environment automatically sets this header to text/html. However, you might want to change this header value. 2 common examples are:

- Returning JSON Data
- Outputting Custom Images

Key Terms

array

array keys

array values

associative arrays

branch

built-in function

classes

Common Gateway Interface (CGI)

concatenation

constant

constructors

data types

dynamically typed

function

function scope

global scope

inheritance

instance

instantiate

local repository

loosely typed

magic methods

methods

naming conventions

one-way hash

parameter default values

parameters

passed by reference

passed by value

properties

remote repository

return-type declarations

sanitizing user inputs

server-side includes (SSI)

subclass

superclass

superglobal arrays

user-defined function

visibility

Copyright

This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.