

# PHP Web Development with Laravel



# **PHP Web Development with Laravel**

## **Learner's Guide**

**© 2022 Aptech Limited**

All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.

**APTECH LIMITED**

Contact E-mail: [ov-support@onlinevarsity.com](mailto:ov-support@onlinevarsity.com)

First Edition - 2022



Onlinevarsity



---

## Preface

---

Hypertext Preprocessor (PHP) is a server-side, open source Web scripting language. It is used for developing dynamic Web pages. PHP supports powerful features for form handling. PHP uses cookies to store information on the user's local computer and sessions to store user information on the Web server.

PHP 3.0 was the first version of PHP developed by Andi Gutmans and Zeev Surakshi in the year 1997. This version supported different databases, protocols, and Application Programming Interfaces (APIs). The developers used the extensibility feature of PHP 3.0 to add new features to it and enhance its functionality. In addition, PHP 3.0 also provided Object-Oriented Programming (OOP) support. In later years, several versions with improved and new features were released. This book covers PHP 8.0.13 and its various features. The book explains essentials of Web application development with PHP and Laravel.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

Design Team



**MANY  
COURSES  
ONE  
PLATFORM**



# Onlinevarsity App for Android devices

Download from Google Play Store

## **Table of Contents**

---

### **Sessions**

#### Architecting Web Applications Using PHP

- Session 1: Introduction to PHP**
- Session 2: PHP Basics and Syntax**
- Session 3: PHP Data Types and Strings**
- Session 4: Variables and Operators in PHP**
- Session 5: Conditional and Flow Control Statements in PHP and Arrays**
- Session 6: Form Handling**
- Session 7: Working with Functions in PHP**
- Session 8: Cookies and Sessions Management in PHP**
- Session 9: Database Management in PHP**
- Session 10: Advanced Features of PHP**
- Session 11: File Handling and Exception Handling in PHP**
- Session 12: Object Oriented Concepts in PHP**
- Session 13: Methods in PHP and Other OOP Concepts**
- Session 14: PHP Web Concepts**
- Session 15: PHP – AJAX**

#### Laravel Framework for Web Applications with PHP

- Session 1: Setting up the Environment**
- Session 2: Introduction to Laravel**
- Session 3: Writing a Simple Application**
- Session 4: Creating a Web Application**
- Session 5: The Eloquent ORM**
- Session 6: Implementing CRUD Operations**

# **Architecting Web Applications Using PHP**

# Session 1

## Introduction to

# PHP



### Learning Objectives

*In this session, students will learn to:*

- Describe the evolution of PHP
- List the important milestones in the development of PHP
- List the enhancements in PHP 8.0
- Outline JIT Compilation
- List the features and uses of PHP 8.0

This session begins with the history of Hypertext Preprocessor (PHP) covering the evolution map right from early days to the current release of PHP 8.0 in brief. It lists and elaborates important milestones in the development of PHP till the current release of PHP 8.0. The session also lists enhancements which have been made in PHP 8.0 from previous versions. Further, the session gives a brief outline of the Just-In-Time compilation feature design that has been introduced in PHP 8.0. Finally, the session ends with describing features and uses of PHP 8.0.

### **1.1 History of PHP**

History of PHP dates back to the year 1994 which is a successor of PHP/FI Product.

Table 1.1 lists the different versions released from 1994 to till date.

<b>PHP Version</b>	<b>Release Date</b>	<b>Notes</b>
<b>1.0</b>	October 1995	In 1994, Rasmus Lerdorf first created PHP and released it under the name Personal Home Page Construction Kit.
<b>2.0</b>	April 1996	Rasmus released a complete makeover of the earlier code as PHP/FI. In June 1996, the complete PHP/FI version 2.0 was released.
<b>3.0</b>	June 1997	The first version which resembles today's PHP was PHP 3.0. The inefficiency of PHP/FI 2.0 was realized by Andi Gutmans and Zeev Suraski of Tel Aviv, Israel. This new language was released as PHP and it removed the implication of limited personal use. PHP was the recursive acronym for Hypertext Preprocessor.
<b>4.0</b>	June 1999	A new engine was introduced here, which improved the codebase and the performance of complex applications. It was dubbed as 'Zend Engine' after the names of Zeev and Andi. Besides the much-improved performance, PHP 4.0 included several other key features.
<b>5.0</b>	July 2004	The core of PHP 5.0 was the Zend Engine 2.0 with a new object model. PHP 5.0 introduced powerful object-oriented programming support, thereby allowing users to write structured and enterprise-level code.
<b>6.0</b>		Was Skipped
<b>7.0</b>	2019	PHP 7.0 was the next major release after PHP 5.0. The core team made several optimizations in the interpreter, but did not introduce the JIT compilation in PHP 7.0 version. These optimizations were mainly done to keep the language backwardly compatible. Optimized RAM usage and improved syntax in PHP 7.0 boosted performance significantly.
<b>8.0</b>	2020	PHP 8.0 came up with JIT compilation and other features.

**Table 1.1: Different PHP Versions**

## **1.2 Introduction to PHP**

---

### **What is PHP?**

PHP is a widely-used open source general-purpose scripting language, suitable for Web development. It can be written and saved as a PHP script or embedded into HTML.

PHP can be used for command line scripting and also for developing client-side Graphical User Interface (GUI) applications that are platform independent.

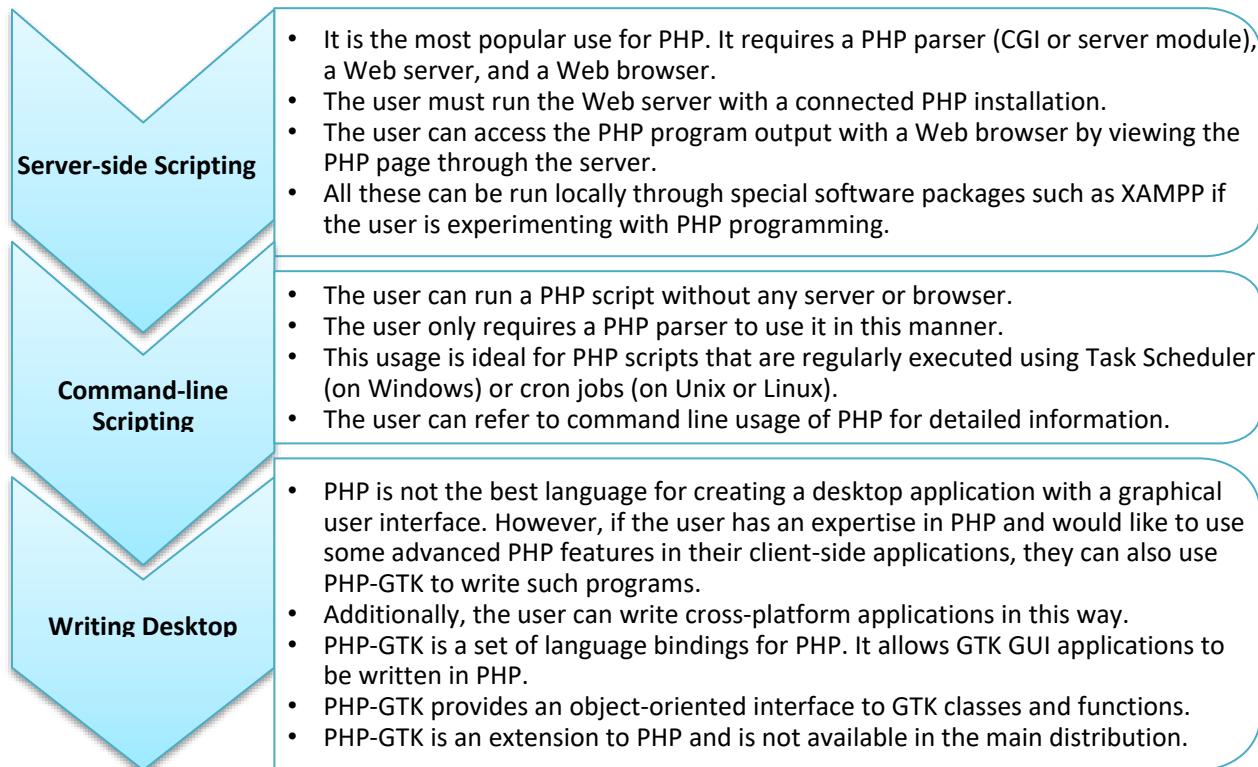
## **1.3 Features of PHP and Uses**

---

Some of the most widely known as well as significant features of PHP are as follows:

- PHP is relatively easy for a fresher and it also has many advanced features for a professional programmer.
- It runs efficiently on the server-side.
- PHP works on many operating systems such as Linux, Windows, and Mac OS X.
- PHP is free and one can download it from the official PHP resource: [www.php.net](http://www.php.net)
- PHP supports many databases such as Oracle, MySQL, MS SQL Server, Sybase, and so on.
- PHP can dynamically generate content of type HTML, PDF, Text, XML, CSV, and many others.
- Writing programs in PHP is easy and fast, which effectively means that it takes less time to build an application.
- Many popular PHP frameworks such as Zend, Laravel, and CodeIgniter are available for PHP.
- There are several Web hosting options available at a fair price for PHP.
- Code deployment is straightforward with PHP.

PHP scripts are used extensively in following three areas:



The users can use PHP on Linux, many Unix variants (including Solaris, HP-UX, and OpenBSD), Microsoft Windows, MacOS, RISC OS, and others. Today, PHP supports most Web servers, including Apache, IIS, and any Web server utilizing the Fast CGI PHP binary, such as Nginx and Lighttpd. PHP works either as a CGI processor or as a module.

Some famous companies and applications that are using PHP include Facebook, Wikipedia, Tumblr, WordPress, and Slack.

Therefore with PHP, the user is free to choose an operating system and a Web server. Furthermore, the user can also choose between procedural programming and Object-Oriented Programming (OOP), or a mixture of both.

The user can generate output in the form of HTML, images, and PDF files. The auto generation of text such as XHTML or any other XML file can be done by PHP, saving in the file system, instead of printing it out, thus forming a server-side cache for dynamic content.

PHP supports a wide range of databases. Therefore, it is very simple to write a database-enabled Web page using database-specific extensions. The user can also connect to any database which supports the Open Database Connection (ODC) standard using the ODBC extension.

PHP also supports connectivity to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows), and many others. The user can open raw network sockets and interact using any other protocol. PHP has support for the WDDX complex data exchange between virtually all Web programming languages. PHP has support for instantiation of Java objects and can use them transparently as PHP objects.

PHP has useful text processing features, including the Perl Compatible Regular Expressions (PCRE). Additionally, there are many tools and extensions for parsing and accessing XML documents. PHP not only standardizes all of the XML extensions, but it also extends the feature set by adding the support for SimpleXML, XMLReader, and XMLWriter. Several other extensions are grouped by category and alphabetically that may or may not be documented within the PHP manual itself, such as XDebug.

Code Snippet 1 shows a simple example of a PHP script inside an HTML file.

### **Code Snippet 1:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php
      echo "Hi, I'm a PHP script!";
    ?>
  </body>
</html>
```

PHP pages have HTML tags with embedded code. The PHP code is enclosed within the <? PHP and ?> tags, thereby allowing the developer to invoke PHP code blocks in an HTML file. <?php marks the start processing instruction while ?> denotes the end processing instruction.

In Code Snippet 1, `echo "Hi, I'm a PHP script!";` is the PHP code which when executed will display the output as "Hi, I'm a PHP script!"

It is recommended for users new to PHP to start directly from PHP 8.0 to avoid migration costs associated with previous versions. The new improvements in PHP 8.0 will allow cleaner code and better performance right from beginning. The user must note that if more code is written in older versions, then migration to PHP 8.0 will require more effort.

If users know in advance that there will be some dependencies incompatible with PHP 8.0 during release, then effort should be made to replace those dependencies right away.

### **How is PHP different from JavaScript?**

Key differences between the two are namely that PHP is designed for server-side scripting, whereas JavaScript is for client-side scripting. In PHP, the code is executed on the server and then, HTML is generated and sent to the client. The client receives the results of script execution, but the underlying code is not exposed. This is very much different from client-side JavaScript which is executed within the browser (client) itself. The developer can configure the Web server to process all HTML files with PHP without exposing the code.

What differentiates PHP from client-side JavaScript is that the execution of code is done on the server, which generates HTML to send to the client. The client receives the output of that running script, but would not know about the code inside it. The user can even configure the Web server to process all HTML files with PHP.

## **1.4 Enhancements in PHP 8.0**

PHP 8.0 was released worldwide on November 26, 2020. PHP as an engine constantly changes by either adding new bits or modifying older bits to optimize the code. These changes make PHP easier and uncomplicated to work. PHP 8.0 is regarded as a major update of the PHP language. It has many new features and optimizations that include the following:

- Named arguments
- Attributes
- Constructor property promotion

- Union types
- Match expression
- Nullsafe operator
- Enhanced error handling, and type system
- Consistency

## Named arguments

In this new feature, the user must specify only the required parameters and skip optional parameters. Arguments are self-documented and are order-independent. Table 1.2 shows how to pass a Named argument to a function in PHP 7.0 and PHP 8.0.

PHP 7.0	PHP 8.0
<code>htmlspecialchars(\$string, ENT_COMPAT   ENT_HTML401, 'UTF-8', false);</code>	<code>htmlspecialchars(\$string, double_encode: false);</code>

*Table 12.2: Named Arguments-Passing Arguments to a Function*

## Attributes

The user can now use structured metadata with PHP's native syntax, instead of PHP Doc annotations. Table 1.3 shows the syntax for attributes in PHP 7.0 and PHP 8.0 respectively.

PHP 7.0	PHP 8.0
<pre>class PostsController {     /**      *      * @Route("/api/posts/{id}",      * methods={"GET"})      */     public function get(\$id)     { /* ... */ }</pre>	<pre>class PostsController {     #[Route("/api/posts/{id}",     methods: ["GET"])]     public function get(\$id)     { /* ... */ }</pre>

*Table 1.3: Syntax for Attributes*

## Constructor Property Promotion

There is less boilerplate code to define and initialize properties.

Table 1.4 shows the example for constructor property in PHP 7.0 and PHP 8.0.

PHP 7.0	PHP 8.0
<pre>class Point {     public float \$x;     public float \$y;     public float \$z;      public     function __construct(         float \$x = 0.0,         float \$y = 0.0,         float \$z = 0.0     ) {         \$this-&gt;x = \$x;         \$this-&gt;y = \$y;         \$this-&gt;z = \$z;     } }</pre>	<pre>class Point {     public     function __construct(         public float \$x = 0.0,         public float \$y = 0.0,         public float \$z = 0.0,     ) {} }</pre>

Table 1.4: Creation of Constructors

## Union Types

The user can use native union type declarations that are validated at runtime, instead of the PHP Doc annotations.

Table 1.5 shows the example of native union type declarations in PHP 7.0 and PHP 8.0.

PHP 7.0	PHP 8.0
<pre>class Number {     /** @var int float */     private \$number;      /**      * @param float int \$number      */     public     function __construct(\$number)     {         \$this-&gt;number = \$number;     } }</pre>	<pre>class Number {     public     function __construct(         private int float         \$number     ) {} }  new Number('NaN'); // TypeError </pre>

<pre>         }     }  new Number('NaN'); // Ok </pre>	
--	--

*Table 1.5: Syntax for Union Type Declarations*

## Match Expression

Match expression in PHP 8.0 is similar to switch with the following features:

- Match is an expression, which effectively means that its outcome can be stored in a variable or it can be returned.
- Match branches only support single-line expressions and do not require a `break;` statement.
- Match does strict comparisons.

Table 1.6 shows the example of switch v/s match case in PHP 7.0 and PHP 8.0.

PHP 7.0	PHP 8.0
<pre> switch (8.0) {     case '8.0':         \$result = "Oh no!";         break;     case 8.0:         \$result = "This is what I expected";         break; } echo \$result; //&gt; Oh no! </pre>	<pre> echo match (8.0) {     '8.0' =&gt; "Oh no!",     8.0 =&gt; "This is what I expected", }; //&gt; This is what I expected </pre>

*Table 1.6: Comparing Switch Case and Match*

## Nullsafe Operator

The user can use a chain of calls with the new null safe operator instead of writing code for the null check conditions.

The entire chain execution is aborted when the valuation of one element in the chain fails and subsequently, the entire chain evaluates to null.

Table 1.7 shows the example of the use of nullsafe operator `?-` in PHP 7.0 and PHP 8.0.

PHP 7.0	PHP 8.0
<pre>\$country = null;  if (\$session !== null) {     \$user = \$session-&gt;user;      if (\$user !== null) {         \$address = \$user- &gt;getAddress();          if (\$address !== null) {             \$country = \$address- &gt;country;         }     } }</pre>	<pre>\$country = \$session?-&gt;user?-&gt;getAddress()?-&gt;country;</pre>

Table 1.7: Example of Usage of NullSafe Operator

## Enhanced Error Handling and Type System

There have been extensive alterations in different system types and error handling improvements, some of which are as follows:

- Stricter type checks for arithmetic/bitwise operators as compared to earlier versions
- Validating abstract trait method
- Magic methods now have correct signatures
- Engine warnings have been reclassified
- Incompatible method signatures give fatal errors
- Fatal errors are no longer silenced by the `@` operator
- Inheritance with private methods
- Mixed type
- Static return type
- Types for an internal functions Email thread
- Usage of Opaque objects instead of resources for OpenSSL, Gd, Curl, XML Writer, Sockets, and XML extensions

## **Consistency**

PHP 8.0 has consistent type errors for internal functions. Most of the internal functions throw an Error exception when there is a failure of validation of the parameters.

## **1.5 Just In Time (JIT) Compiler**

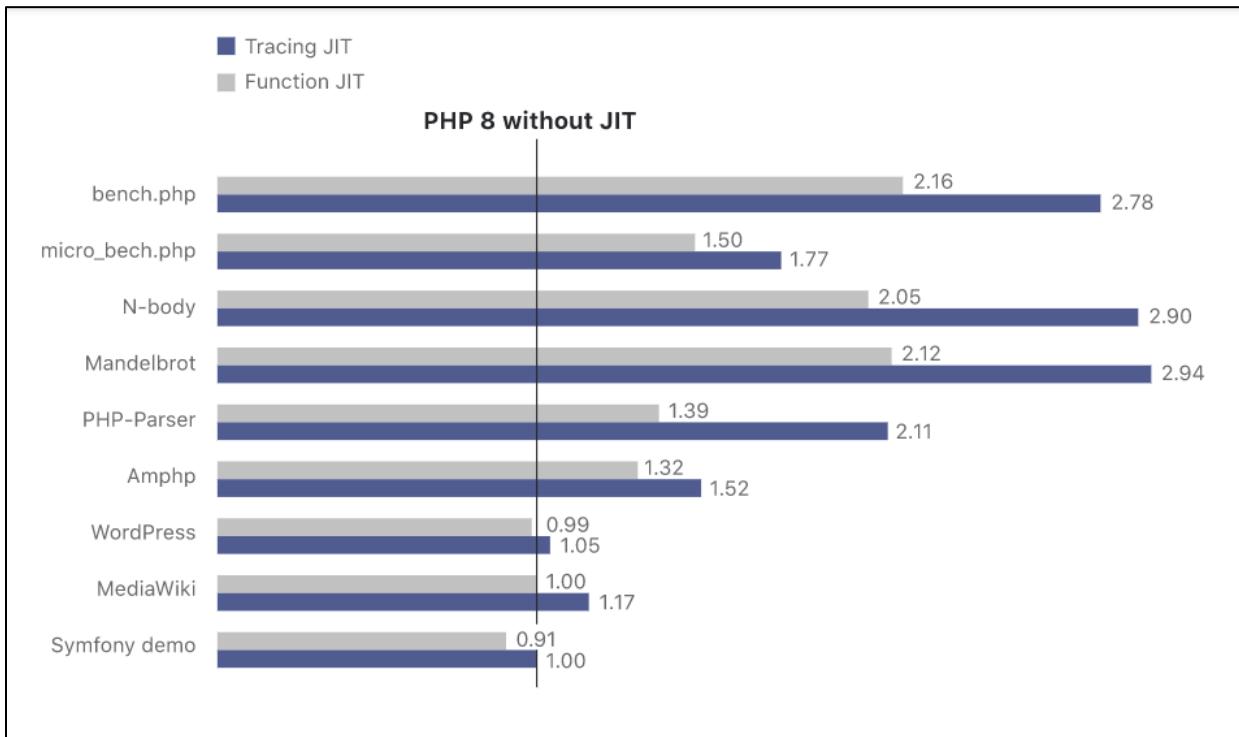
---

Being a deciphered or an interpreted language, PHP on compilation, does not run immediately at launch time. It runs in real-time. In earlier versions of PHP, during compiling, whenever a PHP code is run, the interpreter must first interpret, then compile, and finally execute the code. This is done repeatedly for each request, resulting in slow code execution and wastage of CPU resources. This problem has now been overcome with the introduction of the Just In Time (JIT) compiler, in PHP 8.0.

The JIT compiler allows users to compile a program into machine code just before it can be executed. As JIT bypasses the compilation stage, it provides flexibility in PHP code and brings about extensive improvements in code execution, memory usage, and performance. However, these improvements are applicable only to numerical or mathematical calculations, rather than the usual PHP Web applications. JIT is of immense use in programs that involve a long execution cycle suchas 3D rendering, data analysis, or Artificial Intelligence. PHP 8.0 introduces two JIT compilation engines, namely Tracing JIT and Function JIT.

The Tracing JIT is the most assuring of the two. On synthetic benchmarks, it has three times better performance. Additionally, there is twice the improvement on specific long-running applications.

Figure 1.1 shows a relative JIT contribution to the performance of PHP 8.0.



*Figure 1.1: Relative JIT Contribution to PHP 8.0 Performance*

*Image Credit: <https://www.droptica.com/blog/jit-compiler-php-8/>*

## 1.6 Summary

- PHP is a widely-used open source general-purpose scripting language, especially suited for Web development.
- PHP is easy to learn for freshers and includes several advanced features for seasoned developers.
- Users such as Website developers, WordPress plugin, and theme creators must upgrade their knowledge of PHP to stay ahead of their competitors.
- PHP 8.0 is considered as a major update of the PHP language, with many new features and optimizations.
- Just In Time (JIT) compiler is the most significant feature that has been added in PHP 8.0.
- PHP can dynamically generate content of type HTML, Flash, PDF, Text, XML, CSV, and many others.
- PHP has useful text processing features, including the Perl Compatible Regular Expressions (PCRE).

## 1.7 Test Your Knowledge



1. What does PHP stand for?
  - a) HyperText Markup Language
  - b) Hypertext Preprocessor
  - c) Programming Hypertext Language
  - d) Programming Web language
2. In which version of PHP, Just-In-Time (JIT) Compiler was introduced?
  - a) PHP 5.0
  - b) PHP 7.0
  - c) PHP 8.0
  - d) PHP 9.0
3. Which of the following describes PHP?
  - a) Client-side scripting
  - b) Server-side scripting
  - c) Both A and B
  - d) None of these
4. What is the function of the PHP interpreter?
  - a) To connect ISP and the server
  - b) To process HTML and PHP files
  - c) To translate User language to System Language
  - d) All of these
5. Which of the following is used to begin a PHP script?
  - a) <?php
  - b) <php
  - c) ?php
  - d) <php?

## **Answers to Test Your Knowledge**

---

1. HyperText Preprocessor
2. PHP 8.0
3. Server-side scripting
4. To Process HTML and PHP files
5. <?php

## Session 2

# PHP Basics and Syntax



### Learning Objectives

*In this session, students will learn to:*

- Describe system requirements for running PHP
- Explain installation process of PHP on different platforms
- Outline basics of PHP
- Compare various tags of PHP
- Explain how PHP works with HTML
- Illustrate how to write and run PHP scripts
- Explain how to add comments in PHP

Before a user deep-dives into PHP, it is essential to know about running PHP in their system. Users have to understand how they can install PHP on different platforms as per their requirements and kick start their learning in PHP.

This session begins with explaining installation of PHP on different platforms and goes on to cover basics of PHP, various tags of PHP, and how PHP works with HTML. Finally, the session describes how to write a simple script in PHP and execute it.

## **2.1 PHP Installation**

---

Installing PHP locally on a computer makes it possible for the user to securely create and test applications and PHP-based Websites.

### **2.1.1 System Requirements to Run PHP**

The user must have following three software installed on their local machines to run PHP programs:

1. A Web Server such as Apache
2. PHP Interpreter
3. A Database such as MySQL

Out of these three, the Database is optional, though it is recommended if one is creating practical, real-world oriented applications.

A Web server, a PHP Interpreter, and MySQL database can be installed separately by the user through downloads from respective official Websites. However, open source developers have made all-in-one setup packages such as WAMP, LAMP, MAMP, or XAMPP that take care of these installations easily. The specific package to choose depends on the platform one is comfortable with. Such a package will set up a PHP environment on the user's Windows, Linux, or Mac machines.

All three (LAMP, WAMP, and XAMPP) have common components: Apache HTTP Server, MySQL, and PHP.

Apache HTTP Server is the most critical part of the respective package. It runs the open source Web server on Windows or Linux. When Apache Web server is running on a local Windows/Linux machine, it is possible to test Web pages locally by a developer in a browser. This effectively means that the Web developer may not have to publish and make the page live on the Internet only for testing it.

MySQL and PHP are the other two components of the respective packages. These two are the most common technologies utilized for creating dynamic Websites.

PHP is the scripting language that can be utilized to access data from the high-speed database MySQL. Although PHP, MySQL, and Apache are open-source

components that can be installed individually, they are usually installed together for faster productivity.

## LAMP

LAMP stands for Linux, Apache, MySQL and PHP. LAMP is the most commonly used solution stack for various Web applications for Linux.

## WAMP

WAMP stands for Windows, Apache, MySQL, and PHP. WAMP is a LAMP variation for Windows systems and is often installed as a software bundle (Apache, MySQL, and PHP). Its main purpose is to facilitate Web development and internal testing. Additionally, it may also serve live Websites by acting as a local Web server.

MAMP is the equivalent installation of packages such as LAMP and WAMP on macOS.

WAMP can handle dynamic Websites, is easy to use with PHP, and is available in both 32-bit and 64-bit systems.

## XAMPP

XAMPP is an open-source cross-platform Web server solution stack package that consists of the Apache HTTP Server, MariaDB database, and a PHP interpreter. Moreover, it is free. MariaDB is a community-developed RDMS which has replaced MySQL.

It is easy to transit from a local test server to a live server as most Web Server deployments utilize similar components as XAMPP. XAMPP helps developers to create and test their programs on a local webserver. It allows easy and a faster deployment of a LAMP or a WAMP stack on an operating system by a developer.

### 2.1.2 Web Server

Before starting PHP, one must ensure existence of a Web server, either locally or online. This is because for executing PHP code, the user requires access to a Web server in which the PHP interpreter is running.

The user has three options to choose from:

### **Option 1**

In this option, users must either install Apache and PHP from their open source respective Websites or install an all-in-one package (such as WAMP, LAMP, MAMP, or XAMPP) according to the choice of their OS.

Then, post-installation, following actions can be performed:

- Launch Apache Server and PHP from the program list. In the Apache installation folder, the user will find the **www** directory
- Create and save PHP scripts or create project folders in the **www** directory
- Open any browser and execute the scripts by typing  
**http://localhost/filename.php**

where,

**filename.php** represents the PHP script file name. Regardless of where the script file is stored locally, the URL to execute it on the browser remains consistent in this format.

### **Option 2**

Install PHP and MySQL from their open source respective Websites, create appropriate scripts, and then, find a Web hosting plan with Apache, PHP, and MySQL support to execute your PHP scripts on the Web host.

### **Option 3**

The user can also make use of the interface at <http://www.runphponline.com> to interpret PHP code online without installing anything locally except an editor to create PHP scripts. However, this approach is not feasible for large applications or frequent use.

Nginx is a Web server alternative that is used sometimes instead of Apache. It is tricky to set up and configure Nginx on Windows and it requires more configuration.

#### **2.1.3 Installation of PHP on Windows 8.0 and Higher Versions with Apache**

The minimum requirement for PHP is at least Windows 2008/Vista, either 32-bit or 64-bit. Windows 2008 or Vista are not supported from PHP 7.2.0 onwards.

PHP requires the Visual C Runtime (CRT). Since many applications require it, the chances are that it may already be installed.

Most of the recent PHP versions work perfectly with the Microsoft Visual C++ Redistributable for Visual Studio 2019. The user must download x64 CRT for PHP x64 builds and x86 CRT for PHP x86 builds.

The CRT installer supports the /no restart and /quiet command-line switches, so the user script can run it.

If a user is using Internet Information Services (IIS) but wants to set up PHP, the easiest technique is to utilize Microsoft's Web Platform Installer (WebPI).

## Manual PHP Installation on Windows

### 1. Choose Web Server Apache

There exist several builds of Apache 2 for Windows. It is recommended that the user should use the Apache builds of Apache Lounge. Other options include BitNami, XAMPP, and WampServer. These three offer automatic installer tools. PHP can be used on Apache through mod\_fastcgi and mod\_php. The mod\_php requires a Thread Safe (TS) build of Apache built with the same version of Visual C and the same CPU (x86 or x64).

### 2. Choose Source Build of PHP

Windows based builds of latest versions of PHP can be downloaded from <http://windows.php.net/download/>.

In a programming context, a build is a version of a program.

There are four types of PHP builds:

#### Thread-Safe (TS) PHP build

This build is for single process Web servers, such as Apache with mod\_php.

#### Non-Thread-Safe (NTS) PHP build

This build is for IIS and other FastCGI Web servers (Apache with mod\_fastcgi) are recommended for command-line scripts.

#### x86

This build is for 32-bits systems.

#### x64

This build is for 64-bits systems.

## Apache 2.4 on Microsoft Windows

It is highly recommended to consult the official Apache Documentation to get a basic understanding of the Apache 2.4 Server and then, download Apache 2.4.

Post download, the user should first proceed with the Manual Installation Steps (<https://www.php.net/manual/en/install.windows.manual.php>) and then, proceed with the integration of Apache and PHP.

There are three ways to set up PHP and make it work with Apache 2.4 on the Windows platform - PHP can be run as a CGI, as a handler, or under FastCGI.

**Note:** One must remember that when adding path values in the Apache configuration files on the Windows environment, all backslashes such as c:\directory\file.ext must be converted to forward slashes:  
**c:/directory/file.ext**  
Additionally, in the case of directories, the trailing slash may also be necessary.

### Option 1: Installing as an Apache handler

To load the PHP module for Apache 2.4, following lines in the Apache httpd.conf configuration file must be inserted:

#### **Example #1 PHP and Apache 2.4 as handler**

**Step 1:** Important: # The name of the module was php7\_module prior to PHP 8.0.0.

```
LoadModule php_module "c:/php/php8apache2_4.dll"
<FilesMatch \.php$>
    SetHandler application/x-httdp-php
</FilesMatch>
```

#### **Step 2 # configure the path to php.ini**

```
PHPIniDir "C:/php"
```

**Important:** The actual installation path for PHP must be substituted instead of c:/php/ in the examples. Ensure that the file referenced in the LoadModule directive is at the specified location. **Use php8apache2\_4.dll for PHP 8.**

## **Option 2: Running PHP as CGI**

It is recommended that the user must consult Apache CGI documentation for a concise and complete understanding of executing CGI on Apache.

The PHP-CGI files must be placed in a directory designated as a CGI directory using the `ScriptAlias` directive, to execute PGI as CGI.

A `#!` line has to be placed in the PHP files, which must point to the location of the PHP binary.

**Example #2 PHP and Apache 2.x as CGI** (this code shows the location of PHP binary files)

```
#!C:/php/php.exe
<?php
    phpinfo();
?>
```

### **Warning**

When a server is deployed in CGI mode, it is exposed to many possible vulnerabilities. It is advised that the user must read the CGI security section in the manual to learn how to defend themselves from such attacks.

## **Option 3: Running PHP under FastCGI**

There are several advantages of running PHP under FastCGI as compared to running it as a CGI. It is also simple and easy to set it up this way.

Obtain `mod_fcgid` from <https://www.apachelounge.com>. Win32 binaries are available for download from that site. The user has to install the module to the accompanying instructions.

The user has to configure the Web server as shown in Example 3 and also take care to adjust any paths.

**Example #3 Configure Apache to run PHP as FastCGI**

```
LoadModule fcgid_module modules/mod_fcgid.so
# Where is your php.ini file?
```

```
FcgidInitialEnv PHPRC "c:/php"  
<FilesMatch \.php$> /* code to configure PHP and will run program  
using SetHandler fcgid-script FastCGI wrapper*/  
</FilesMatch>  
FcgidWrapper "c:/php/php-cgi.exe" .php
```

Once completed, the files with a .php extension will now be executed by the PHP FastCGI wrapper.

To install Apache with PHP 8.0 on Windows, one has to perform following steps. If the versions of Apache and PHP are different, then appropriate steps should be taken into consideration.

The Apache server should be downloaded from [www.apache.org/dist/httpd/binaries/win32](http://www.apache.org/dist/httpd/binaries/win32). It is advised to download the current version of the stable release with the **no\_src.msi** extension. Double-click the installer file to install. C:\Program Files is a common location. The installer will also prompt the user whether to run Apache as a service or from the command line or DOS prompt. Therefore, the user should not install it as a service, as that may cause problems with startup.

The PHP binary archive should be extracted using unzip utility. C:\PHP is a common location.

Some of the .dll files should be copied from the PHP directory to the system directory (usually C:\Windows). Refer the manual to know which files. The user will require php8ts.dll for every case. Copy the file corresponding to the Web server module - C:\PHP\Sapi\php8apache.dll. to Apache modules directory.

It is recommended to copy either **php.ini-dist** or **php.ini-recommended** to your Windows directory, and rename it to **php.ini**. This file has to be edited to get configuration directives, hence, open this file in a text editor. New users can set error reporting to **E\_ALL** within the PHP scripts they will create on their development machines. This will result in all errors being reported by PHP and will help new users debug or troubleshoot faster.

Next, the user has to configure and communicate to the Apache server from where to look for the PHP files and what will be the extension of the PHP files. Usually, .php is the standard, but the user can use .html, .phtml, or so on).

To do this, go to the appropriate HTTP configuration files directory and open **httpd.conf** with a text editor. For example, user may have installed Apache in C:\Program Files\Apache directory. In that case, the path for configuration files will be C:\Program Files\Apache\conf.

Then, search for the word **DocumentRoot** within the **httpd.conf** file. This word should appear twice. Change both paths to the directory from where to load the PHP files. The user has to add at least one PHP extension directive as shown in the first line of following code:

```
LoadModule php8_module modules/php8apache.dll AddType  
application/x-httdp-php .php .phtml
```

Additionally, user can also add following line: `AddModule mod_php8.c`

After this, user can stop and restart the WWW service as follows:

- Type **services.msc** in the Run command of Windows or launch Services from **Control Panel**.
- Scroll down the list to World Wide Web Publishing Service, right-click, and select **Stop**.
- Then, right-click and select **Start** to start it again.

If required, user may also restart the computer.

Further, user can launch a text editor and write a basic script to test the PHP installation:

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

Save this file in the Web server's document root as **test.php** and launch it in a browser in the format **http://localhost/test.php** or **http://127.0.0.1/test.php**. User should never launch the PHP file directly from local path, for example, **D:\Php codes\test.php** should not be given in the Address bar of the browser. This is because in that case, it will not be treated as a HTTP request which is required for the file to be processed.

## 2.1.4 PHP Installation on MacOS X with Apache

There are a few pre-compiled and pre-packaged flavors of PHP for the MacOS platform. These can assist a lot in a standard configuration setup. However, if required, for features such as a database driver or a different secure server, the user may have to build PHP and/or the Web server on their own. If the user is unfamiliar with compiling and building their software, it will be a good option to check whether there is any already built packaged version of PHP with required features.

Homebrew package management system provides the quickest installation of PHP on MacOS.

- Go to the Homebrew site (<https://brew.sh/>) and install homebrew.
- Use following command to start the installation: `brew install php`

The user can also refer to following alternative resources for easy to install packages and precompiled binaries for PHP on MacOS:

- MacPorts: <http://www.macports.org/>
- Fink: <http://www.finkproject.org/>

Mac users also have the option of having a pre-built source or a binary installation provided with the platform. In this case, user only requires to edit the Apache configuration file to update the version and switch on the Web server. Following steps describe installation process on MacOS:

**Step 1:** Open the Apache config file in a text editor as root.

```
sudo open -aTextEdit /etc/httpd/httpd.conf
```

**Step 2:** Edit the file. Uncomment following lines:

```
LoadModule php8_module  
AddModule mod_php8.c  
AddType application/x-httpd-php .php
```

**Step 3:** Restart the Web server.

```
sudo apachectl graceful
```

**Step 4:** Open a text editor and write the following basic script:

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

Save it as `test.php` in the Web server's document root.

**Step 5:** Launch the file in a browser in appropriate format (as described earlier).

### 2.1.5 PHP Installation on Linux and Unix OS with Apache

Similarly, PHP can be installed on UNIX variants or Linux. It requires following pre-requisites:

- The PHP source distribution for version 8  
<http://www.php.net/downloads.php>
- Latest Apache source distribution
- Database, if required (such as MySQL, Oracle, and so on)
- make utility - you can freely download it at  
<http://www.gnu.org/software/make>

#### Steps for installation of Apache and PHP 8.0.13 on Linux or Unix

Following are the steps for installing Apache and PHP 8 on Linux or Unix machines:

**Step 1:** Unzip and extract the tar files of the Apache source distribution.

```
cd apache_2.4.x
./configure --prefix=/usr/local/apache \
-enable-so
make
make install
```

**Step 2:** Configure Apache Server as follows:

```
./configure --with-apxs=/usr/sbin/apxs \
--with-mysql=/usr/bin/mysql
make install
```

**Step 3:** Unzip and extract the tar files of the PHP source distribution.

```
cp php.ini
/usr/local/lib/php.ini
```

**Step 4:** If a MySQL database is being used, configure PHP to work it with as follows:

**Step 5:** The php.ini file must be installed and edited to get configuration directives as follows:

Perform similar steps as the Windows installation for searching DocumentRoot and editing the configuration files.

**Step 6:** Edit httpd.conf to inform Apache server where to serve files from and what extension(s) to identify PHP files.

**Step 7:** The user can add atleast one PHP extension directive, followed by a second handler to parse all HTML files as PHP:

```
AddType application/x-httpd-php  
.php  
AddType application/x-httpd-php  
.html
```

**Step 8:** Restart server. Whenever user changes HTTP configuration or php.ini file, the server must be stopped and restarted again.

**Step 9:** Finally, create and test a simple PHP script similar to how it was described in earlier steps.

## 2.2 PHP Basics

PHP is used commonly for building highly dynamic and interactive Web pages for a robust user experience. It also communicates with the databases and provides with better flexibility and simplicity.

For this course, from this point onwards, it is assumed that XAMPP is installed with PHP 8.0.13 and MySQL on a Windows system and all codes will be run under this environment.

### 2.2.1 Working of PHP

When a user navigates to a .php page from their Web browser, the browser sends an HTTP request to the Web server. For example, when user types the URL of the file as `index.php` in the browser and hits **Enter**, the browser will send the request to Web server, and the server will start searching for this file on its file system. If the Web server locates the file, it will send this file to the **PHP interpreter**. Otherwise, the Web server will generate **Error 404** or **File Not Found**.

The Web server only sends files that have a .php extension to an interpreter. Other files that have extensions such as .html, .htm, and so on are not sent to a PHP interpreter, even if they contain PHP codes inside them.

Once the file is sent to the PHP interpreter, it scans through all the opening and closing PHP tags and then, proceeds with the processing of the PHP code within these tags.

PHP interpreter additionally checks if there is a database connection. If it discovers a database connection, it will send or retrieve data from the database after proper authentication.

PHP scripts are interpreted on the Web server and the outcome (HTML) is sent back to the client machine.

### 2.2.2 Writing PHP Scripts

Users can use a text editor to write PHP code. There are many good editors available today that provide strong language support and features such as autocomplete, syntax highlighting, and more. Notepad++, Sublime Text, and Atom are some of these editors.

The basic structure of a PHP script mainly consists of following:

- PHP Start and End Tags
- PHP code with HTML markup
- Comments in PHP (optional)

#### PHP Tags

A PHP code block starts with `<?php` tag and ends with `?>` tag.

Syntax for declaring PHP tags is as follows:

```
<?php  
    //your php code goes here  
?>
```

#### PHP with HTML

PHP has been designed to work with HTML. Therefore, the user can easily write and embed PHP code within HTML and vice versa.

Code Snippet 1 shows a simple script in PHP to display a message.

## Code Snippet 1:

```
<!DOCTYPE html>
<html lang="en">
<title>Hello program in PHP</title>
<body>
<?php echo "Hello, Welcome to PHP"; ?>
</body>
</html>
```

In Code Snippet 1, `echo` command of PHP is used, which enables users to write output data to the browser. Each PHP statement ends with a ; (semicolon). In case you write another statement without completing the first with a semicolon, PHP will report a syntax error.

### 2.2.3 Executing PHP Scripts

To run or execute an PHP program, the user must save the code to the Web server under the www or htdocs directory (depending on how installation is done) with a.php or .html extension (if PHP is embedded within HTML). Once it is done, the server has to be started.

Assume that XAMPP is installed with PHP 8.0.13. XAMPP will generate a htdocs folder, under which PHP scripts can be placed.

When the server is up and running, the user must open a Web browser, navigate to localhost, and type in the path of the file, for example, [http://localhost/code2\\_1.php](http://localhost/code2_1.php)

For example, the output of Code Snippet 1 is shown in Figure 2.1. The file is saved under the path C:\xampp\htdocs but in the browser, it must be launched as [http://localhost/code2\\_1.php](http://localhost/code2_1.php).

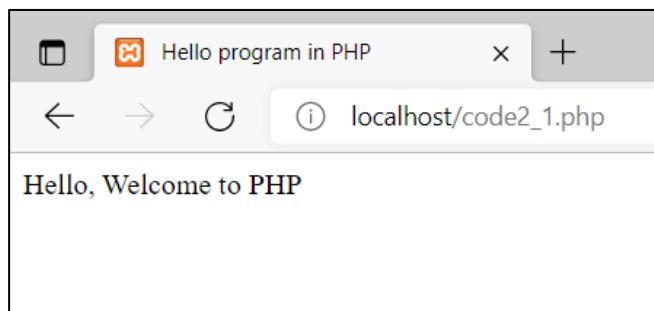


Figure 2.1: Output of Code Snippet 1

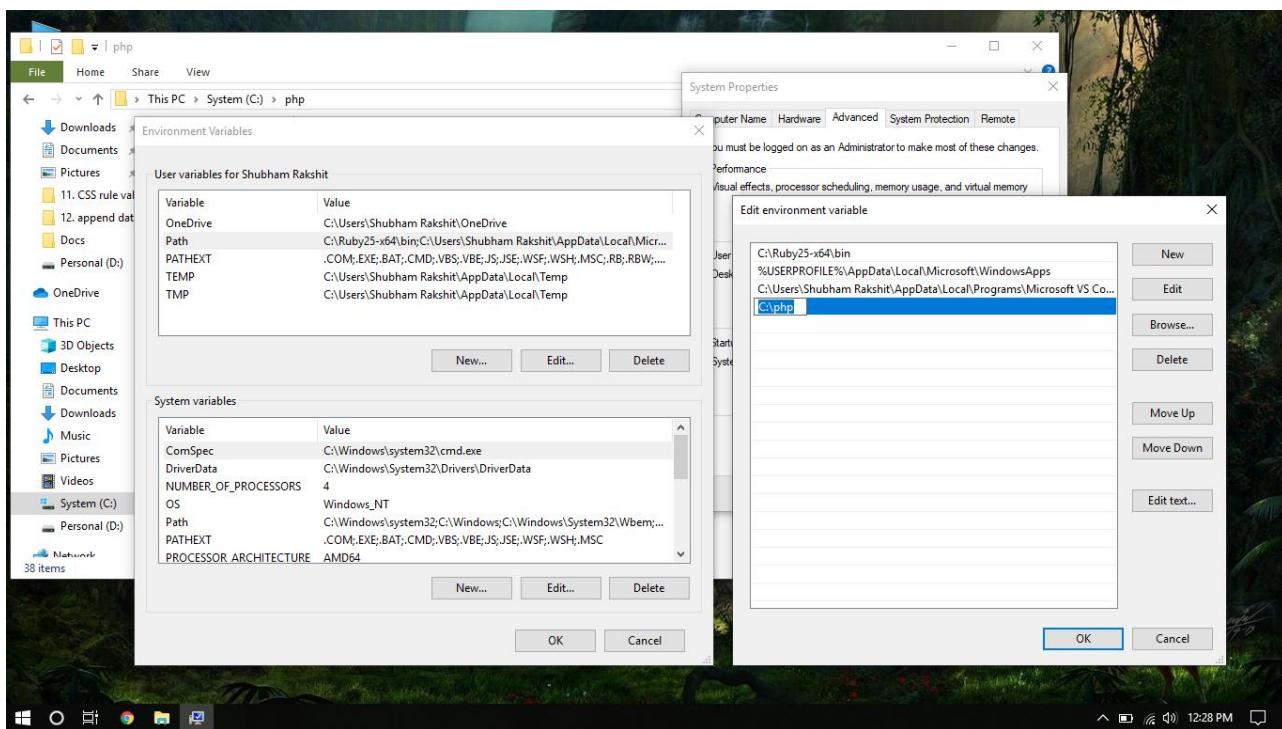
User can also run or execute a PHP script on the command prompt without the HTML tags.

### Steps to run PHP script in the command prompt:

Add the PHP installation path (such as C:\php) to the system environment variable paths for it to become accessible from the command prompt.

To add the path, go to Control Panel. Then, click **Advanced system settings** link and open **Environment Variables**. Click **System Variables** and then, select **Path** to append a new path to the existing Path variable. Click New, type the path, and click OK. Close all the remaining windows after clicking OK.

Figure 2.2 shows adding of path to Environment Variable Path.



*Figure 2.2: Adding Path to Environment Variable Path*

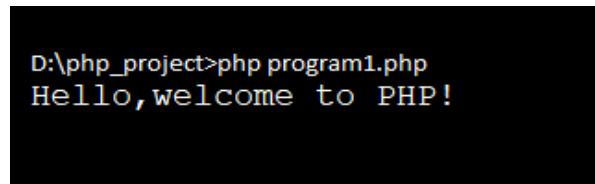
Once the Path is set, the php.exe command can be given at the command prompt from any folder, not necessarily the folder where PHP was installed.

Now, create the code given in Code Snippet 2 in the file test.php to be executed in the command prompt.

### **Code Snippet 2:**

```
<?php  
echo "Hello, Welcome to PHP!";  
?>
```

The user can execute this script at the command prompt as shown in Figure 2.3.



*Figure 2.3: Output of Code Snippet 2*

### **Comments in PHP**

Writing comments in a program is an important and good practice, as it makes code readable as well as easy to understand for developers. Consider that a developer Mark employed with a company has written considerable code for some product applications. Now, Mark has left his job and his code has been given to another developer Peter to continue further. However, Peter finds it difficult and cumbersome to understand what Mark has been doing in the code because there was no documentation and no commenting. Had Mark used comments suitably in his code, it would have been a seamless process for Peter to take over the code. Thus, comments play an important role in documenting code.

The PHP interpreter ignores execution of comment blocks, thereby making code readable with no overheads. Therefore, comments can be used anywhere in a program to add information about code blocks.

In PHP, // or # can be used to generate a single-line comment and /\* with \*/ to make a large comment block with multiple lines.

Code Snippet 3 shows an example of a PHP program that shows single and multiple line comments in PHP.

### **Code Snippet 3:**

```
<html>
<body>
<?php
// This is a single line commented text, and
# This is another single line commented text
/*
This is
a Multi-line comment
block area
*/
echo 'This script made use of comments in PHP';
?>
</body>
</html>
```

## 2.3 Summary

- Users require a Web Server, PHP interpreter, and optionally database installed on their local machines to run PHP.
- PHP installations can be done on various operating systems such as Windows, Linux, and macOS.
- There are all-in-one packages such as LAMP, WAMP, and XAMPP that facilitate easy installation of Apache Web server, PHP, and MySQL.
- The basic structure of a PHP script mainly consists of PHP start and end tags PHP code with HTML markup, and optionally comments
- The echo command displays the output data to the browser.
- PHP interpreter ignores the execution of the comment block.
- Adding comments in a program makes code readable as well as easy to understand for developers.
- PHP scripts can be executed on the command prompt without the HTML tags.

## 2.4 Test Your Knowledge



1. PHP files have a default file extension of \_\_\_\_\_.
  - a) .html
  - b) .xml
  - c) .php
  - d) .hphp
  
2. Which of the following is the correct syntax of PHP?
  - a) <?php >
  - b) <php >
  - c) ?php ?
  - d) <?php ?>
  
3. Which of the following is correct to add a PHP comment?
  - a) & ..... &
  - b) // .....
  - c) /\* ..... \*/
  - d) Both (b) and (c)
  
4. Which of the following is used for displaying the output in PHP?
  - a) Echo
  - b) Write
  - c) Print
  - d) Both (a) and (c)

## **Answers to Test Your Knowledge**

---

1. .php
2. <?php ?>
3. Both (b) and (c)
4. Both (a) and (c)

## 2.5 Try It Yourself

1. Write a PHP program to display the message “This is my first PHP Page”.
2. Write a program using HTML tags to display a message in multiple lines.

## Session 3

# PHP Data Types and Strings



### Learning Objectives

*In this session, students will learn to:*

- Identify different data types in PHP and their usage
- Explain different types of string functions, numbers, and math functions in PHP
- Elaborate on PHP constants and functions used to create constants

This session introduces different data types in PHP and explores them in detail. Various math functions are explained. The session concludes with a discussion of string types in PHP.

### 3.1 Data Types

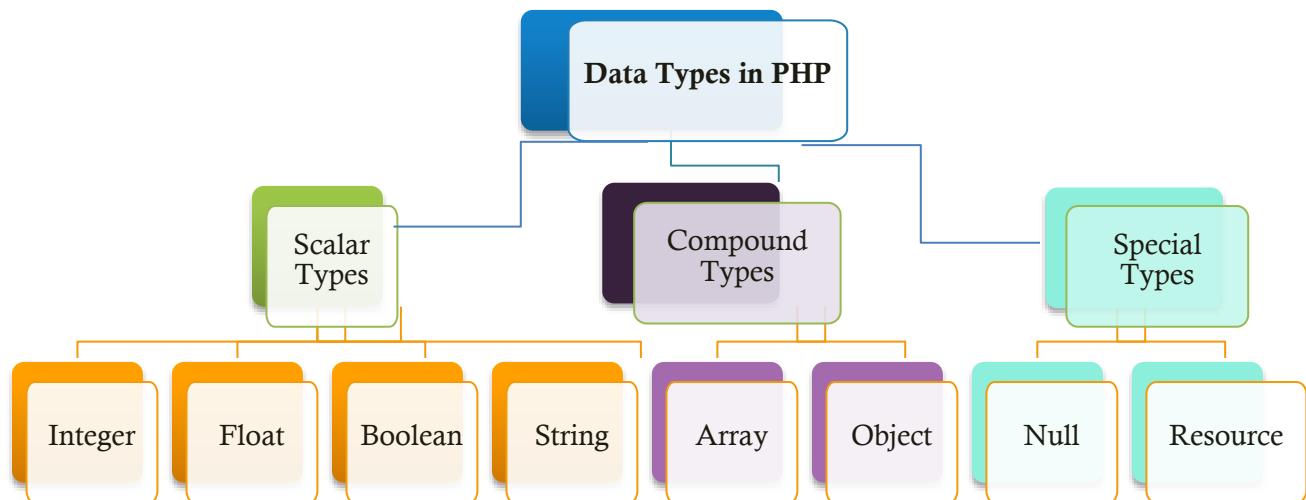
Generally, a data type refers to the classification of data based on its attributes. PHP, however, is a loosely typed programming language. As a result, it does not require users to explicitly define the data type. Instead, PHP analyzes attributes of data in order to determine the right data type.

PHP has a number of standard data types available along with attributes. Here are some standard PHP data types and their attributes, presented in Table 3.1.

Data Type	Attributes	Examples
Integer	Whole number	-3, 0, 69
Float	Number with a decimal point	1.5, -3.143, 9.8
String	Alphanumeric	'Hello World', 'Language'
Boolean	True or False	\$x = true; \$y = false;

*Table 3.1: Data Types*

Figure 3.1 depicts the data type classification in PHP.



*Figure 3.1: Data Type Classification In PHP*

## 3.2 Integers

The integer data type comprises a whole number (a non-decimal number). It is platform-dependent, and its value ranges between -2,147,483,648 and 2,147,483,647 on 32-bit machines and higher on 64-bit machines.

This data type is often used to represent numerical data comprising whole numbers in programs, such as product quantities, number of students in a class, population numbers, and so on.

Code Snippet 1 shows a simple example of using integer data type in a PHP script.

### **Code Snippet 1:**

```
<?php  
$patientId = 512;  
echo $patientId;  
?>
```

Here, though `a` is not explicitly declared to be an integer, it is assumed to be so by the compiler based on the data it contains. Since patient Identification number is a whole number and cannot be represented in fractions, the integer type is best suited for it. In applications, users should determine which data are whole numbers and use the integer type to represent that data.

The maximum allowed value for an integer in PHP on a system can be determined using the constant `PHP_INT_MAX`. This is a predefined constant defined by PHP Core.

Other predefined constants for integers in PHP include:

- `PHP_INT_MIN`: Supports the smallest integer
- `PHP_INT_SIZE` : Defines the size of an integer in bytes

Code Snippet 2 shows an example of how to get the maximum value of an integer data type using `PHP_INT_MAX`.

### **Code Snippet 2:**

```
<?php  
echo PHP_INT_MAX;  
?>
```

In this code, predefined constant `PHP_INT_MAX` is used to display the largest integer value allowed on a system.

### **Output:**

9223372036854775807

Essential criteria to store data as an integer data type are as follows:

- Should have at least one digit
- Should not have a decimal point
- Can be either positive or negative

- Can be represented with base 10-decimal, base 16 – hexadecimal, base 8 - octal, or base 2 - binary notation

`var_dump()` is a built-in function in PHP that dumps information about one or more variables onto the output. The information displayed will include data type and value of the variable(s). This function has no return type.

Code Snippet 3 shows how `var_dump()` returns the data type and value of a specified inputs.

### Code Snippet 3:

```
<?php
// PHP code to illustrate working of var_dump() function
var_dump(var_dump(3, 3.1, FALSE, array(10, 20, 30, 40)));
?>
```

### Output:

```
int(3) float(3.1) bool(false) array(4) {
    [0]=> int(10)    [1]=> int(20)    [2]=> int(30)    [3]=> int(40)
} NULL
```

Here since literals are used directly, `var_dump()` just displays the data type of each literal. If variables had been used, `var_dump()` would display both data type and value of the variable.

## 3.3 Float

---

A float data type comprises a number with a decimal point (fraction) or a number in exponential form. For example, 256.4, 10.358, 9.8, 7.64E+5, 5.56E-5, and so on. It is also called ‘double’ or ‘real number’. In PHP, float, double, and real are the same and are represented only as float.

This type is platform-dependent and maximum value of a float data type is up to 1.7976931348623E+308. It has a precision of 14 digits maximum.

Code Snippet 4 shows an example for the representation of float data.

## Code Snippet 4:

```
<?php  
var_dump(1097.5499563); // as of PHP [8.0.13]  
?>
```

In this code, `var_dump()` is used to display the data type of a given literal value. Since the literal is a decimal number, it belongs to `float` data type.

## Output:

```
float(1097.5499563)
```

PHP supports a few built-in functions to work with floating-point numbers. To determine whether a variable's data type is float or not, one can use following PHP functions:

- `is_float()`: If the variable is of type `float`, this method returns true (1); otherwise, it returns false.

Syntax: `is_float(variable)`

- `is_double()`: alias of `is_float()`

Syntax: `is_double(variable);`

Code Snippet 5 explains how to use `is_float()` function to determine whether the data type of a number is `float` or not.

## Code Snippet 5:

```
<?php  
var_dump(is_float(16.25));  
var_dump(is_float('xyz'));  
var_dump(is_float(789));  
?>
```

Here, in the code, `is_float()` function is used with various data to check whether each given data item is `float` or not. If it is `float`, the function returns true, otherwise, it returns false.

## Output:

```
bool(true)
```

```
bool(false)  
bool(false)
```

## 3.4 Boolean

---

The simplest variable type is Boolean, (also called bool in PHP) which acts as a switch. It specifies a truth value that can be either `true` or `false`. Booleans are frequently used in conditional statements; `true` if the condition is true, `false` otherwise.

For example, in an application, user may want to store a value to indicate whether student is new or existing. `isNew` can be defined as a bool variable and if `true`, it can indicate that student is new and if `false`, it will indicate that student is an existing one.

To represent a bool literal, use the PHP constants `true` or `false` (both are case-insensitive).

A PHP script to define Boolean is shown in Code Snippet 6.

### Code Snippet 6:

```
<?php  
$bool_var=TRUE;  
echo $bool_var . "\n";  
var_dump($bool_var);  
$bool_var1=false;  
echo $bool_var1;  
var_dump($bool_var1);  
?>
```

### Output:

```
1  
bool(true)  
bool(false)
```

## 3.5 Strings

---

A string is a collection of characters, such as "Hello Steve!". Typically, it has literals surrounded by single quotes ('') or double quotes (""). PHP has no support for Unicode, as it only supports a 256-character set.

Strings within double quotes are preprocessed as follows:

- Escape sequences which are special character beginning with backslash (\) are replaced with their equivalent representation.

Commonly used escape sequences are listed in Table 3.2.

Escape Sequence	Description
\n	Is replaced by the newline character
\r	Is replaced by the carriage-return character
\t	Is replaced by the tab character
\\$	Is replaced by the dollar sign itself (\$)
\"	Is replaced by a single double-quote ("")
\\"	Is replaced by a single backslash (\)

*Table 3.2: Escape Sequences in PHP*

- Variable names that begin with \$ are replaced with string representations of their values.

Code Snippet 7 demonstrates an example of using strings and escape sequences.

### **Code Snippet 7:**

```
<?php
    header('Content-type: text/plain');
    $name = "William";
    $str = '$name is displayed.\n';
    echo($str);
    echo "\n";
    $str = "$name is displayed\n". "Goodbye.";
    echo($str);
?>
```

### **Output:**

```
$name is displayed.\n
William is displayed
Goodbye.
```

Here, in the code, `header('Content-type: text/plain');` is given to indicate to the PHP interpreter that the output should be rendered in the browser

as plain text and not HTML. If this is not given, the \n escape sequence will not work in the browser. If the script is being executed at command prompt instead of browser, this statement is not necessary.

In the code, variable `name` is assigned a string literal, `William`. When this variable is used within a single quoted string, it will be treated as is and the value `William` will not be substituted in the output. Similarly, an escape sequence embedded in a single quoted string will not be processed. Hence, the first `echo` statement displays:

```
$name is displayed.\n
```

On the other hand, when the variable name is used within a double quoted string along with escape sequence \n, both the variable and the escape sequence will be processed by the PHP interpreter and `echo` statement will display `William is displayed` followed by a new line.

This is how PHP processes single quoted and double quoted strings differently.

## **3.6 PHP Array**

---

An array in PHP is a compound type. An array is a single variable that contains values of the same data type. In PHP, an array is a predefined map that connects values to keys.

By making use of arrays in code, users can reduce amount of code in programs because they will not have to define multiple variables to store lists of data items. Arrays are also easy to navigate as one can use loops to traverse through all the elements of an array. Arrays can be sorted and searching becomes easier.

An array, list (vector), stack, hash table (a map implementation), collection, dictionary, queue, and so on can all be represented with the array type.

### **Creation of an Array in PHP**

The `array()` language construct can be utilized to create an array. It can take any number of key and value pairs separated by comma as arguments.

#### **Syntax :**

```
array(
```

```

key  => value,
key2 => value2,
key3 => value3,
...
)

```

A PHP script to create an array is shown in Code Snippet 8. `$fruits` is an array in this code. The data type and values of elements are returned by the `var_dump()` function.

### **Code Snippet 8:**

```

<?php
$fruits = array("orange", "apple", "mango");
var_dump($fruits);
?>

```

### **Output:**

```

array(3) {
[0]=>
string(6) "orange"
[1]=>
string(5) "apple"
[2]=>
string(5) "mango"
}

```

Following are three types of arrays in PHP:

Multidimensional	Indexed	Associative
Arrays containing one or more arrays	Arrays with a numeric index	Arrays having named keys

### **ArrayLength - `count()` or `sizeof()` Function**

The `count()` or `sizeof()` function in PHP returns the number of elements or values in an array.

Code Snippet 9 shows an example of a PHP program to determine number of elements in an array using the function `count()` or `sizeof()`.

### **Code Snippet 9:**

```
<?php  
$Friends = array("A", "B", "C");  
echo count($Friends);  
?>
```

`count()` function is used in this code to count the array elements.

### **Output:**

3

## **3.7 PHP Object**

Objects are instances of user-defined classes that can store both values and functions. In simple terms, a class is a data structure that contains variables of different data types (termed as properties), constants, and functions (termed as methods). Objects must be declared explicitly.

A basic class definition has `class` keyword, followed by a class name, followed by definitions of properties and methods belonging to the class all of which are enclosed within a pair of curly braces. A class name can be any valid identifier, as long it is not a reserved word in PHP.

Code Snippet 10 illustrates a PHP application that creates a class, an object and calls the class methods.

### **Code Snippet 10:**

```
<?php  
class Color {  
    function Color() {  
        $color_name = "Green";  
        echo "color is: " . $color_name;  
    }  
    function ChangeColor() {  
        $color_name = "Red";  
        echo "<br>updated color is: " . $color_name;  
    }  
}  
$objColor = new Color();
```

```
$objColor -> Color();  
$objColor -> ChangeColor();  
?>
```

### **Output:**

color is: Green  
updated color is: Red

## **3.8 PHP Null Value**

---

The `null` type refers to a variable with no value. Only the value 'null' or 'NULL' is permitted for the `null` type (case-insensitive). When a variable is created without a value, it is given a `null` value by default.

Code Snippet 11 shows an example of a PHP script to declare a variable with `null` value.

### **Code Snippet 11:**

```
<?php  
$var = NULL;  
var_dump($var);  
?>
```

PHP has a built-in function, `empty()`, that returns true if value of a variable evaluates to false. This could mean an empty string, `NULL`, integer 0, or an array with zero elements. PHP also supports the built-in function, `is_null()` that return true if the variable has the value `NULL`.

Code Snippet 12 shows usage of null type.

### **Code Snippet 12**

```
<?php  
$x = "Hello Alexa!";  
$x = null;  
var_dump($x);  
?>
```

This code shows an example of a PHP script how to reset a variable with the `null` value to make it empty.

**Output:**

NULL

## 3.9 PHP Resource

---

In PHP, a resource is a compound type treated more like a special variable, than a specific data type. It acts as a repository for externally referenced functions and resources. A common example of a PHP resource is a database call.

The function `is_resource()` can be used to determine whether a variable is a resource or not. The function `get_resource_type()` returns the resource type. Resource variables can contain special handles pointing to files and database connections..

## 3.10 PHP String Functions

---

PHP supports built-in functions to work with strings.

**strlen()**: Returns the length of the string. `strlen()` is a built-in function that calculates the length of a string, including all special characters and whitespaces.

**Syntax :**

`strlen(string)`

where, `string` is an input string.

Code Snippet 13 shows an example of a PHP script to find the string length.

**Code Snippet 13:**

```
<?php  
echo strlen("Hello Steve!");  
?>
```

In this code, `strlen()` function is used to count the length of the string “Hello Steve!” including whitespaces and characters.

**Output:**

12

**str\_word\_count()** : The function `str_word_count()` returns the number of

words in a string.

#### **Syntax:**

```
str_word_count(string, returnval, chars)
```

where,

`string` is the input string, `returnval` specifies return value of the function and can be one of the following: 0 (returns number of words found), 1 (returns an array containing all words found within the string), or 2 (returns an associative array), and `chars` specifies a list of additional characters which will be considered as a word

Code Snippet 14 shows an example of a PHP script to count the number of words in the string "Hello Steve!".

#### **Code Snippet 14:**

```
<?php  
    echo str_word_count("Hello Steve!");  
?>
```

In this code, `str_word_count()` function is used to count the number of words in the string "Hello Steve!".

#### **Output:**

2

**strrev()**: This is a predefined function used to reverse a string. It is one of the most basic string operations which programmers and developers use.

#### **Syntax:**

```
string strrev ( string $string );
```

Code Snippet 15 shows an example of a PHP script to reverse the string "Hello Steve!".

#### **Code Snippet 15:**

```
<?php  
    echo strrev("Hello Steve!");  
?>
```

`strrev()` function is used to reverse the string Hello Steve! in this code.

**Output:**

```
!evetSolleH
```

**strpos()**: This function is used to find a specified text within a string. If there is a match, it returns the character position of the first match; if there is not, it returns FALSE.

**Syntax:**

```
strpos(string, find, start);
```

Code Snippet 16 shows an example of a PHP script to search for the text "Steve" in the string "Hello Steve!".

**Code Snippet 16:**

```
<?php  
echo strpos("Hello Steve!", "Seven");  
?>
```

`strpos()` is used here to find the character position of the string Seven.

**Output:**

```
6
```

**str\_replace()**: For replacing characters in a string this function is used.

**Syntax:**

```
str_replace($search, $replace, $string, $count)
```

where, `search` indicates value being searched for, `replace` is the replacement, `string` is the string in which to replace, and `count` indicates the number of replacements performed successfully.

Code Snippet 17 shows an example of a PHP script to replace the text "Steve" with "Diana" in the string "Hello Steve!".

**Code Snippet 17:**

```
<?php  
echo str_replace("Steve", "Diana", "Hello Steve!");  
?>
```

Here, `str_replace()` is used to replace the text Steve in the string Hello Steve with Diana.

### **Output:**

```
Hello Diana!
```

**`ucwords()`:** This function returns a string after converting first character of each word of given string into uppercase.

### **Syntax:**

```
string ucwords (string $str )
```

Code Snippet 18 shows an example of a PHP script to use `ucwords()` function.

### **Code Snippet 18:**

```
<?php  
$str="my name is cinderella.";  
$str=ucwords($str);  
echo $str;  
?>
```

Function `ucwords()` is used here to change the first letter of the words in a string to uppercase.

### **Output:**

```
My Name is Cindrella.
```

## **3.11 PHP Numeric Functions**

Infinity is defined as a numeric value greater than `PHP_FLOAT_MAX`. Following PHP functions are utilized to determine whether a numeric value is finite or infinite:

- `is_finite()`
- `is_infinite()`
- 

Code Snippet 19 shows an example of a PHP script to check whether a numeric value is finite or infinite.

### **Code Snippet 19:**

```
<?php  
// To check whether the given numeric value is finite or  
// infinite  
$x = 2.9e532;  
var_dump($x);  
?>
```

### **Output:**

float(INF)

### **PHP NaN**

Not a Number (NaN) is an acronym used for mathematical operations that are impossible to perform. Following function in PHP can be used to determine whether or not a value is a number:

- `is_nan()`

Code Snippet 20 shows an example to demonstrate an invalid calculation that returns a NaN value.

### **Code Snippet 20:**

```
<?php  
$x = acos(25);  
var_dump($x);  
?>
```

### **Output:**

float(NAN)

### **PHP Numerical Strings**

If a variable is a number or a numeric string, the function `numeric()` returns true; otherwise, it returns false.

Code Snippet 21 shows an example of a PHP script to check if the variable is numeric.

## Code Snippet 21:

```
<?php
// To check if the given variable is numeric
$a = 5985;
var_dump(is_numeric($a));
$b = "5985";
var_dump(is_numeric($b));
$c = "59.85" + 100;
var_dump(is_numeric($c));
$d = "Hello";
var_dump(is_numeric($d));
?>
```

In this code, `is_numeric()` is used to check if each given input is a numeric string or not.

### Output:

```
bool (true)
bool (true)
bool (true)
bool (false)
```

## 3. 12 PHP Type Conversion and Casting

One aspect of PHP is that it can convert data types automatically. As a result, if a variable is assigned an integer value, the variable's type will be an integer as well. The type of the variable will then change to a string if a string is assigned to it. There are chances that automatic conversion may damage the code.

In some scenarios, user may be required to perform explicit conversions or casting. PHP facilitates this through built-in functions.

### Converting Other Data Types to Float

- **String to Float:** In PHP, strings can easily be converted to floats, where a string is a series of characters and each character is the same as a byte. Most of the time, it will not be necessary since PHP performs implicit type conversion.

Following are some of the methods for converting a string into a float in PHP:

- Using `floatval()` function: A `float` is returned by this function. It is generated by typecasting the value of the variable passed to it as a parameter.

Syntax: `$floatvar = floatval($stringvar)`

- Using **Type-Casting**: In typecasting, the user explicitly specifies the data type into which the data should be cast.

Syntax: `$floatvar = (float)$stringvar`

Code Snippet 22 demonstrates use of `floatval` function.

### Code Snippet 22:

```
<?php  
$str="15.689943";  
$converted = floatval($str);  
var_dump($converted);  
?>
```

In this code, a floating-point value is stored within a string. Then, `floatval()` is used to convert the string to a `float` value.

### Output:

```
float(15.689943)
```

## PHP Casting Strings and Floats to Integers

It is sometimes necessary to convert a numerical value to a different data type. To convert a value to an integer, one can use following functions:

- `(int)`
- `(integer)`
- `intval()`

Code Snippet 23 shows an example of a PHP script to convert a float and string to an integer.

## Code Snippet 23

```
<?php
// Cast float to int
$a = 12345.768;
$int_cast = (int)$a;
echo $int_cast;
// Cast string to int
$b = "12345.768";
$int_cast = (int)$b;
echo $int_cast;
?>
```

Here, `int_cast()` is used to convert a float to integer.

### Output:

12345  
12345

## 3.13 PHP Math Functions

---

Math functions in PHP facilitate users to conduct mathematical operations on numbers.

### `pi()` Function

The approximate value of PI is returned by `pi()`. It is a crucial PHP mathematical function for solving math problems.

Code Snippet 24 shows an example where `pi()` is used in a formula to compute surface area of a circle.

### Code Snippet 24:

```
<?php
echo (pi()). "<br>";
$radius = 6;
$surfacearea = 4*pi()*$radius*$radius; // Surface Area = 4 x
pi x radius square
echo "Surface Area is: " . $surfacearea;
?>
```

**Output:**

```
3.1415926535898
Surface Area is: 452.38934211693
```

**min() and max()**

Lowest and highest values in a list of inputs can be discovered using `min()` and `max()` functions.

**min():** This function returns the lowest value.

**Syntax:** `min(array_values)`

or

```
min(value1,value2,value3...)
```

**max():** This function is used to find the highest value.

**Syntax:** `max(array_values)`

or

```
max(value1,value2,value3,value4...)
```

Code Snippet 25 shows an example of a PHP script to find the lowest and highest value in the array.

**Code Snippet 25:**

```
<?php
echo(min(0, 200, 30, 20, -3, -600)); // returns -600
echo(max(0, 600, 30, 20, -3, -600)); // returns 600
?>
```

Here, `min()` and `max()` are used to display lowest and highest value in an array.

**Output:**

```
-600
600
```

**abs()**

The `abs()` function returns absolute positive value of an integer. For a given whole number, it returns an `integer` value and for a given floating point value, it returns a `float` value.

**Syntax:** `abs(number)`

Code Snippet 26 show an example to find the absolute positive value.

### **Code Snippet 26:**

```
<?php  
echo (abs (-2.8)) ;  
?>
```

### **Output:**

2.8

### **sqrt()**

Function `sqrt()` returns the square root value of a given number.

Syntax:

```
float sqrt(value) ;
```

Code Snippet 27 shows an example to find the square root of a number.

### **Code Snippet 27:**

```
<?php  
echo (sqrt(16486)) ;  
?>
```

Function `sqrt()` is used to find easily square root value of 16486, which would have been difficult for an ordinary human being to compute. Thus, this function can prove useful in complex formulas too.

### **Output:**

128.39781929612

### **round()**

The `round()` function is used to round off the value of a floating-point number.

### **Syntax:**

```
float round($number, $precision, $mode)
```

Code Snippet 28 demonstrates use of `round()` function.

### Code Snippet 28:

```
<?php  
echo (round(0.59));  
echo (round(0.39));  
?>
```

### Output:

1  
0

### `rand()`

The `rand()` function is used to generate a random number.

Code Snippet 29 shows the use of `rand()` function.

### Code Snippet 29:

```
<?php  
echo rand() . "\n";  
echo rand() . "\n";  
echo rand(15, 35) . "\n";  
echo rand();  
?>
```

Here, the `rand()` function is called three times without any parameters and called once using two parameters. Each time it is called, it generates a different number. When it is called with two parameters, it generates a random number in the given range between them.

### Output:

204971092  
1063018108  
32  
1238196620

## **3.14 PHP Constants: constants() and define()**

---

Constants are similar to variables. However, once they are specified they cannot be altered.

### **Constants**

A constant is a term that identifies a single value. The value of the constant cannot be changed while the script is running. The constant name can only begin with a letter or an underscore (\$ sign should not be before the constant name).

Constants, unlike variables, are automatically global over the whole script.

### **Create a PHP Constant:**

Using `define()` function, constants can be created.

#### **Syntax:**

```
define(name, value, case-insensitive)
```

Table 3.2 shows the parameters used for `define()` function.

Parameter	Description	Default Value
name	Defines the name of the constant	NA
value	Specifies the value of the constant	NA
case-insensitive	Should the constant name be case-insensitive?	false

*Table 3.2: define() Parameters*

Code Snippet 30 shows an example to create a constant with a case-sensitive name.

#### **Code Snippet 30:**

```
<?php  
// case-sensitive constant name  
define("Hello", "What's new in PHP8?");  
echo Hello;  
?>
```

## **Output:**

What's new in PHP8?

### **3.14.1 PHP Constant Arrays**

In PHP 8.0.13, an array constant can be created using the `define()` function.

Code Snippet 31 shows an example to create an array constant.

#### **Code Snippet 31:**

```
<?php
define("cars", [
    "Alfa Romeo",
    "BMW",
    "Toyota",
    "Range-Rover"

]);
echo cars[0];
?>
```

## **Output:**

Alfa Romeo

### **3.14.2 Global Constants**

Constants exist on a global scale. As a result, they can be used throughout the script.

Code Snippet 32 shows an example of using a constant inside a function, although it is defined outside the function.

#### **Code Snippet 32:**

```
<?php
define("Hello", "What's new in PHP?");
function myTest() {
    echo Hello;
}
myTest();
?>
```

## **Output:**

What's new in PHP?

### **3.14.3 Magic Constants**

In PHP, values of some predefined constants change depending on the context in which they are used. These constants are known as Magic Constants. They are case-insensitive. There are nine magic constants in PHP, of which eight magic constants start and end with double underscores (\_\_):

1. \_\_LINE\_\_
2. \_\_FILE\_\_
3. \_\_DIR\_\_
4. \_\_FUNCTION\_\_
5. \_\_CLASS\_\_
6. \_\_TRAIT\_\_
7. \_\_METHOD\_\_
8. \_\_NAMESPACE\_\_
9. ClassName::class

Unlike regular constants, all of these nine constants are resolved at compile-time instead of run time.

## 3.15 Summary

- A data type is the classification of data based on its attributes.
- PHP is a loosely typed language which means that it does not require defining variables with types. Instead, PHP analyzes given attributes or values and determines appropriate data type.
- There are different data types, mainly Integer, Float, String, Boolean, and so on.
- PHP also supports compound types such as arrays and objects.
- PHP has many functions related to Strings, Numbers, NaN, Math, and so on.
- In PHP, a constant is an identifier (name) for a simple value and its value constant cannot be changed during the script execution.
- Magic constants are predefined constants in PHP and change their values with the context of their use.

## 3.16 Test Your Knowledge



1. What will be the output of following PHP code?

```
<?php
    header('Content-type: text/plain');
    $year = "2025";
    $str = '$year is arriving \t soon';
    echo($str);
    echo "\n";
    $str = "$year is arriving \t soon";
    echo($str);
?>
```

A	\$year is arriving \t soon 2025 is arriving soon
B	2025 is arriving \t soon 2025 is arriving soon
C	\$year is arriving \t soon \$year is arriving \t soon
D	None of these

- a) A  
b) B  
c) C  
d) D
2. Which of these is a compound data type in PHP?  
a) array  
b) struct  
c) dictionary  
d) integer

3. Which of the following is not a data type in PHP?
  - a) Resource
  - b) Object
  - c) Null
  - d) Void
4. How many values does Boolean data type hold?
  - a) 1
  - b) 2
  - c) 3
  - d) 4
5. Objects are specified as instances of user-defined classes that can hold \_\_\_\_\_.
  - a) Lists
  - b) Functions
  - c) Constants
  - d) Variables, Constants, and Functions

## **Answers to Test Your Knowledge**

---

1. A
2. array
3. Void
4. 1 (either true OR false)
5. Variables, Constants, and Functions

### 3.17 Try It Yourself

1. Write a program to calculate volume of a sphere given the radius value as 17.5.

The formula for volume of a sphere with radius r is :  $4/3 * \pi * r^3$ .

2. Write a program to convert a string such that first letters of each word are capitalized, then reverse it, count number of words in the string and replace the last word in the string with "Hooray!"

Assume that given string for this exercise is "We are going on a journey, yay!"

## Session 4

# Variables and Operators in PHP



### Learning Objectives

*In this session, students will learn to:*

- Describe variables in PHP
- Describe various data types used in PHP
- Define the scope of PHP variables
- Explain handling of regular expressions
- Identify and elaborate on different types of operators used in PHP

Any name or symbol that takes the place for a value is called a ‘Variable’. Any symbol that performs an operation is called an ‘Operator’. In PHP, both variables and operators are of prime importance. There are different types of variables and operators used in PHP programming. Along with this, numerous data types are also supported in PHP.

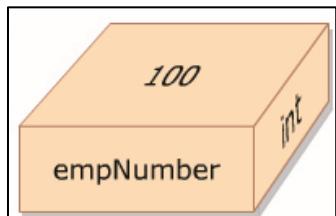
This session begins by explaining variables and various data types used in PHP. It defines the scope of these variables. The session also covers Portable Operating System Interface for uniX (POSIX) regular expressions. Further, the session briefly outlines object-oriented programming in PHP.

### **4.1 Variables**

---

Variables in a program are usually used for storing data or values that can be used anytime during the execution of a program. Variables are used for storing data such as numeric and character values, strings and memory addresses at

runtime. Figure 4.1 illustrates the concepts of variables. Here, `empNumber` is a variable name.



**Figure 4.1: Variable**

#### 4.1.1 PHP Variables

In PHP, the attributes related to the data being supplied determine the data type. There are various data types, such as floating-point numbers, strings, and so on, that have implicit support in PHP.

#### 4.1.2 Variable Naming

In PHP, there are some rules to be followed while creating variables. Though some are similar to those followed in other programming languages, some are unique to PHP. They are as follows:

- The names of all variable must start with the \$ sign. For example, `$my_var`. The syntax used to declare a variable is `$variablename=value;`.
- Names of variables are case sensitive, that is, `$my_var` is different from `$MY_VAR`. Therefore, `$ABC`, `$ABc`, `$Abc`, and `$abC` are four different variables.
- Names of variables must start with an alphabet, which can be in either lowercase or uppercase. This is then followed by other characters. For example, `$my_var1` is a valid variable name, whereas `$1my_var` is not.
- Names of variables cannot contain any spaces. Therefore, '`$first name`' is not a valid variable name. However, an underscore can be used instead of the space, such as `$first_name`. Characters such as \$ or – cannot be used to separate variable names.
- Since PHP is a loosely typed language, the declaration of data types for variables is not required. The PHP interpreter automatically analyzes the values and makes appropriately converts to the correct datatype.
- Once a variable has been declared, it can be reused throughout the code.
- You can apply the Assignment Operator (=) to assign any value to a variable.

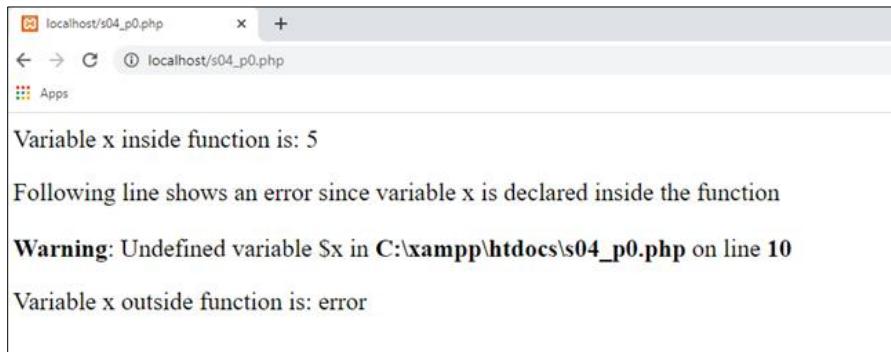
### 4.1.3 PHP Local Variables

Variables declared within a function are termed as local variables for that function. A variable declared within a function can only be accessed within that function. Code Snippet 1 shows how to declare a variable with local scope and demonstrates how it is used in PHP.

#### Code Snippet 1:

```
<?php
function Test() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
Test();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

Figure 4.2 shows the output for Code Snippet 1.



*Figure 4.2: Output for Code Snippet 1*

Here, since no value for `x` variable is declared outside the function, it will show error in the output when you try to print value of `x` outside the function. Local variables are recognized only by the function in which they have been declared. Therefore, local variables can exist with the same name across different functions. A variable that has been declared outside the function with the same name is completely different from the variable that has been declared inside the function.

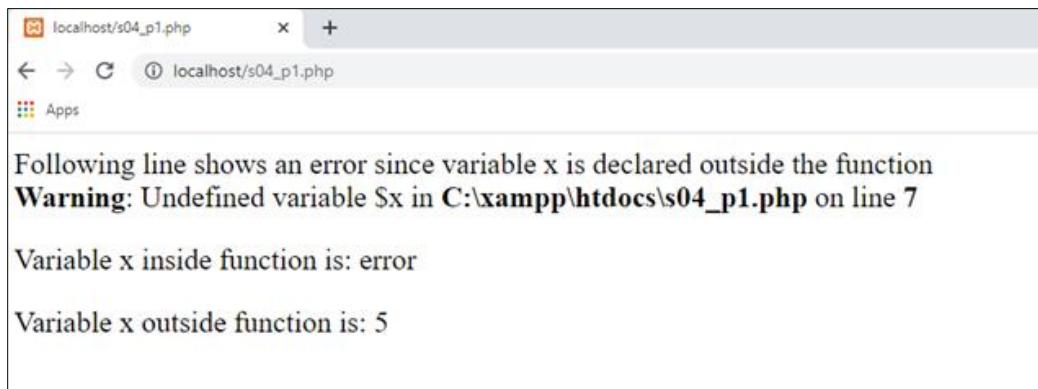
#### 4.1.4 PHP Global Variables

Global variables are declared outside the body of a function. These variables can be accessed from anywhere in the program. Code Snippet 2 shows an example of a program to demonstrate a global variable.

#### Code Snippet 2:

```
<?php
$x = 5; // global scope
function Test() {
    // using x inside this function will generate an error
    echo "Following line shows an error since variable x is
    declared outside the function.";
    echo "<p>Variable x inside function is: $x</p>";
}
Test();
echo "<p>Variable x outside function is: $x</p>";
?>
```

Figure 4.3 shows the output for Code Snippet 2.



**Figure 4.3: Output for Code Snippet 2**

Here, variable `x` inside function does not contain any value, since `x` has been declared as the global variable outside the function. The `echo` keyword is now used to print value of `x` variable outside the function. Therefore, variable `x` outside the function is 5.

#### 4.1.5 PHP Static Variables

In PHP, a variable is usually deleted, once it has been executed and its memory has been released. However, sometimes a variable is required even after its

execution. To handle this scenario, you can declare a variable as a static variable, which is present only in a local function. Static memory is the place where static variable is stored. You can create a static variable during the start of program execution and destroy it during the end of program execution. Code Snippet 3 shows how to declare a static variable. The `static` keyword in PHP enables developers to declare a variable as static.

### Code Snippet 3:

```
<?php
    function static_variable()
    {
        static $X = 10;      //static variable
        $Y = 20;             // non-static variable

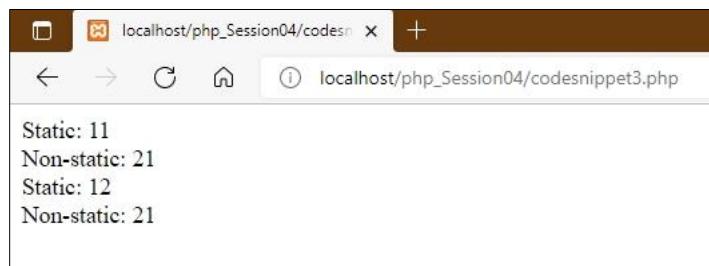
        $X++; //increment in static variable

        $Y++; //increment in non-static variable
        echo "Static: " . $X . "<br>";
        echo "Non-static: " . $Y . "<br>";
    }

    //first function call
    static_variable();

    //second function call
    static_variable();
?>
```

Figure 4.4 shows the output for Code Snippet 3.



**Figure 4.4: Output for Code Snippet 3**

Here, after each function call, `$X` is regularly incremented. However, `$Y` is not. Since `$Y` is not a static variable, its memory is released after each function call is executed. In case of static variable, it does not hold the previously incremented

value. Each new function call stores the new value assigned to the `Y` variable. As a result, the incremented value is shown inside the function. Whereas, in the case of `$X`, it retains the previous value. Each time the incremented value is printed, when it is being used in the second function call.

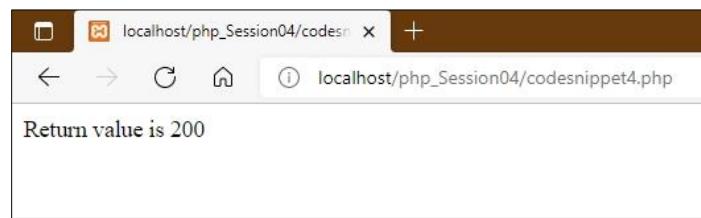
#### 4.1.6 PHP Function Parameters

There can be zero or multiple parameters for a function. Multiple parameters in a function are separated by a comma ( , ). If the number of arguments that are passed to the function are less than the number of parameters defined, then an error is raised by PHP. A developer should declare function parameters after the function name, but inside parentheses. They should be declared similar to how regular variables would be declared. In PHP 7.0, trailing commas were ignored by the PHP interpreter. However, from PHP 8.0 onwards, trailing commas can be placed differently. Code Snippet 4 show a program with function parameters.

#### Code Snippet 4:

```
<?php
// multiply a value by 20 and return it to the caller
function multiply($value)
{
    $value=$value*20;
    return $value;
}
$retval=multiply(10);
print "Return value is $retval\n";
?>
```

Figure 4.5 shows the output for Code Snippet 4.



*Figure 4.5: Output for Code Snippet 4*

Here, one parameter is passed to a function `multiply()` and then, the function is called and the result of `$value` variable is stored in the `retval` variable.

## 4.1.7 Variable Types

Based on the attributes, data can be classified into different categories, called as data types. Alphanumeric characters are classified as strings, whole numbers are classified integers, and numbers with decimal points are classified as floating points. There are eight data types that are used to construct variables.

### Integers

- Are whole numbers without a decimal point. For example, 4195.

### Doubles

- Are floating-point numbers. For example, 49.1 or 3.14159.

### Booleans

- Have only two possible values. They are either True or False.

### NULL

- Is a special type that has only one value, that is, NULL.

### Strings

- Are sequences of characters. For example, 'PHP supports string operations'.

### Arrays

- Are named and indexed collections of other values.

### Objects

- Are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.

### Resources

- Are special variables that hold references to resources that are external to PHP. For example, database connections.

Here, while the first five data types are simple types, arrays and objects are compound types. This means that they are able to package up other arbitrary values of arbitrary type.

### Integers

An integer is a number which does not have any decimal part. For example, the numbers 2, 256, -256, 10358, and -179567 are all integers.

An integer data type is defined as a non-decimal number. The range is different for 32-bit systems and 64-bit systems.

#### 32-bit Systems

- Between -2147483648 and +2147483647

#### 64-bit Systems

- Between -9223372036854775808 and +9223372036854775807

Since an integer's limit is exceeded, any value more or less than this, will be stored in the form of a float. The result will only be stored in the form of a float, even if just one of the operands is a float. For example, if  $4 * 2.5$  is 10, the result is still stored in the form of a float, since one of the operands (2.5) is a float.

Following are some of the rules that apply to integers:

- There must be at least one digit in an integer. For example, `$int_var = 69.`
- There must not be any decimal point in an integer. For example, `$int_var1 = 87654.`
- The integer can be either negative or positive. For example, `$another_int = -1245+134.`
- You can specify an integer in three different formats:
  - Octal (8-based and prefixed with 0)
  - Decimal (10-based)
  - Hexadecimal (16-based and prefixed with 0x)

Following are the three functions supported in PHP, to check if the type of a variable is an integer:

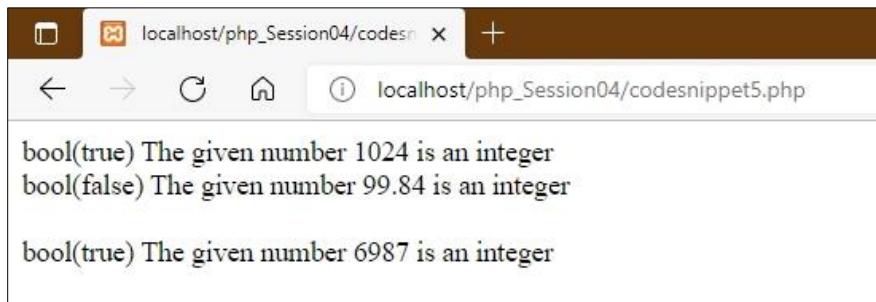
- `is_int()`
- `is_long()` - alias of `is_int()`
- `is_integer()` - alias of `is_int()`

Code Snippet 5 shows a program to check whether the variable is an integer or not.

### Code Snippet 5:

```
<?php
// Check if the type of a variable is integer
$x1 = 1024;
var_dump(is_int($x1));
echo "The given number $x1 is an integer </br>";
// Check again...
$x2= 99.84;
var_dump(is_int($x2));
echo "The given number $x2 is an integer </br>";
echo "<br>";
$y=6987;
var_dump(is_int($y));
echo "The given number $y is an integer </br>";
?>
```

Figure 4.6 shows the output for Code Snippet 5.



```
bool(true) The given number 1024 is an integer
bool(false) The given number 99.84 is an integer
bool(true) The given number 6987 is an integer
```

**Figure 4.6: Output for Code Snippet 5**

Here, the code checks to see if the number is an integer or not. A built-in function `var_dump` is used. This returns a type of data variable. Here, it returns `int`. another built-in function `is_int` is used to check if the number is type integer or not. If it is an integer, `true` will be returned. Since the first number 1024 is an integer, `true` is returned. The next number 99.84 is not an integer, therefore, `false` is returned.

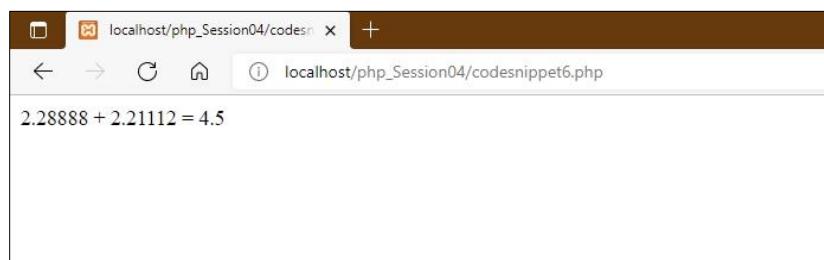
## Doubles

By default, doubles data type prints with the minimum number of decimal places. For example, 3.14159 or 49.1. Code Snippet 6 shows a program to print doubles.

### Code Snippet 6:

```
<?php
$many = 2.2888800;
$many_2 = 2.2111200;
$few = $many + $many_2;
print("$many + $many_2 = $few <br>");
?>
```

Figure 4.7 shows the output for Code Snippet 6.



```
2.28888 + 2.21112 = 4.5
```

**Figure 4.7: Output for Code Snippet 6**

Here, two floating-point numbers are added and the value is stored in variable `few`. The program shows that double numbers only display `few` decimal number places.

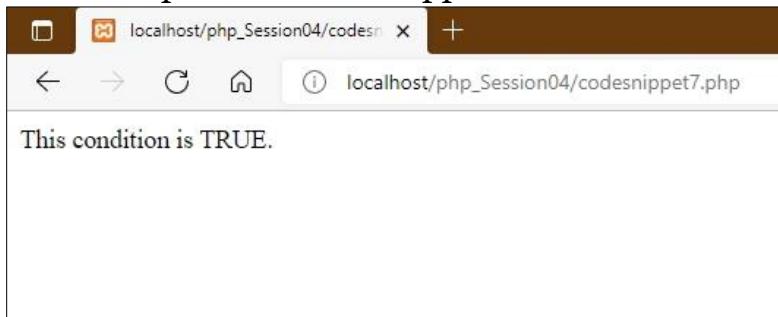
## Boolean

Booleans are the simplest data type and are similar to an electrical toggle switch. They can hold only one out of two values at a time, namely, TRUE (1) or FALSE (0). They are mostly used with conditional statements. If the condition is correct, then they return TRUE. If the condition is not correct, they return FALSE. Code Snippet 7 shows a program to highlight the usage of Boolean data type.

### Code Snippet 7:

```
<?php
    if (TRUE)
        echo "This condition is TRUE.";
    if (FALSE)
        echo "This condition is FALSE.";
?>
```

Figure 4.8 shows the output for Code Snippet 7.



*Figure 4.8: Output for Code Snippet 7*

Here, the Boolean value is shown as true. By default, the first value `true` is always printed, if there is no condition is present in the program. Hence, here, the `true` block is executed.

## NULL

The special data type Null has only one value `NULL`. It is generally written in capital letters, as it is case sensitive. If a variable is created without any value, it is automatically assigned the `NULL` value. A variable with `NULL` value will evaluate to FALSE in a Boolean context.

The syntax is:

```
$my_var = null;
```

Code Snippet 8 shows a program that shows the usage of data type NULL.

### Code Snippet 8:

```
<?php  
    $nl = NULL;  
    echo $nl; //it will not give any output  
?>
```

In Code Snippet 8, since the variable has been assigned with a null value, there is no output.

## Strings

A string is a series of characters, where the size of the character is one byte. In PHP, only the 256-character set is supported and not the native Unicode. However, in 32-bit systems, a string can have a size of 2 GB (maximum of 2147483647 bytes).

The user can specify a string literal in four different ways.

### Single quoted

Strings are enclosed in single quotes (').

### Double quoted

Strings are enclosed in double quotes (").

### Heredoc syntax

Strings with variables that contain numerous words can be easily defined to create text for Web pages. The syntax is \$strVar = <<<LABEL.

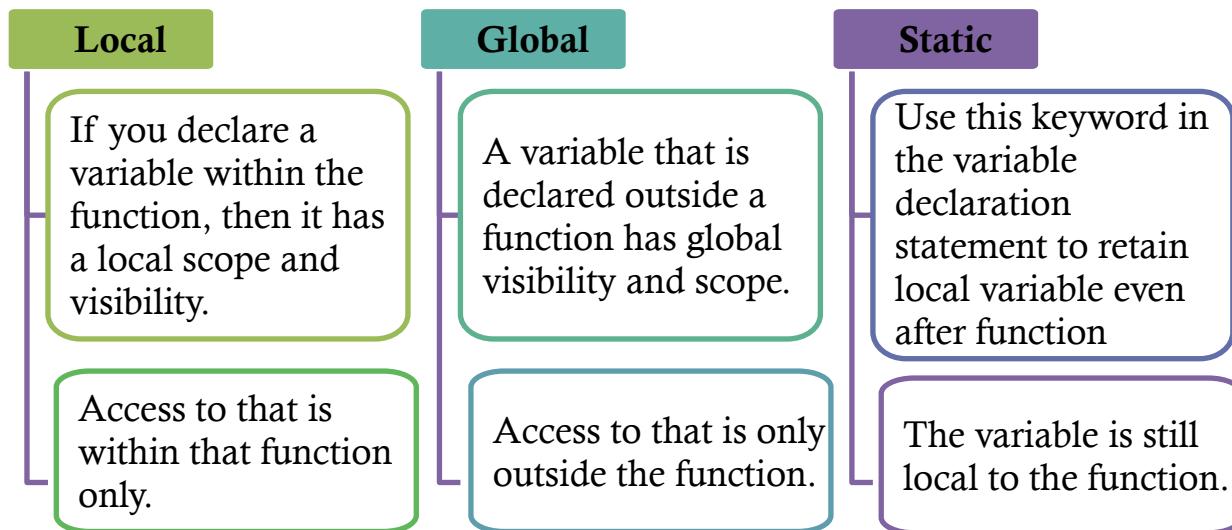
### Nowdoc syntax

Nowdoc statement begins with '<<<' sequence with their identifier enclosed with single quotes (').

## 4.1.8 PHP Variables Scope

Similar to any other programming language, each variable has a scope associated with it. This scope determines the variable's visibility. If a variable is global, all the scripts in an application can access it. On the other hand, it is possible to access a local variable within the script where it has been defined.

Variables can be declared anywhere within the script. However, the scope of a variable will depend where the variable has been declared. There are three different variable scopes.



### PHP – Global Keyword

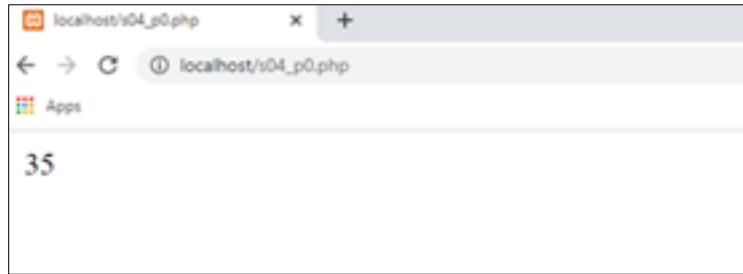
If you want to access the global variable inside a function, add the `global` keyword before the variable. However, you can directly access or use these variables outside the function too, without adding any keyword. In PHP, all global variables are stored in an array that is called `$GLOBALS[index]`. This holds the variable name. You can access this from within functions and also use it to directly upload the global variables.

Code Snippet 9 shows a program using the `global` keyword.

#### Code Snippet 9:

```
<?php
$x1 = 25;
$y1 = 10;
function myTest() {
    global $x1, $y1;
    $y1 = $x1 + $y1;
}
myTest(); // run the function
echo $y1; // display updated value for variable $y
?>
```

Figure 4.9 shows the output for Code Snippet 9.



**Figure 4.9: Output for Code Snippet 9**

Here, you are adding two global values using the function `myTest`. The function is called and the result is stored in variable `y1`. Finally, the result is printed. Here, since the value of the two given numbers becomes 35 after adding, the output is 35.

### **PHP – Static Keyword**

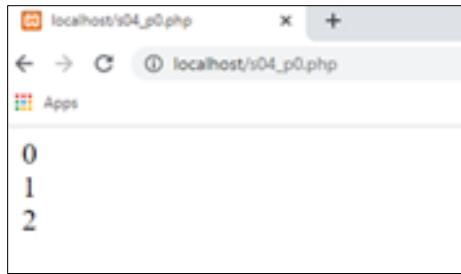
Static variables are created when the local variable is required and not to be deleted after function execution. To create a static variable, you can add the `static` keyword before the variable while first declaring it in a function. Each time the function is called, the variable will still retain the information from the last instance in which the function was called. Here, the variable is still local to the function.

Code Snippet 10 shows a program using the `static` keyword.

### **Code Snippet 10:**

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
```

Figure 4.10 shows the output for Code Snippet 10.



*Figure 4.10: Output for Code Snippet 10*

In Code Snippet 10, increment operator is used to print the value of `x`, where the value of `x` is initially zero. Function `myTest()` is called thrice with `echo`. Hence, the value of `x` gets incremented each time the function is called.

## 4.2 Regular Expression

---

Generally called regex, regular expressions are a sequence of characters that describe a specific search pattern in the form of text strings. They are commonly used in programming algorithms aimed at specific tasks. Regular expressions fetch required strings based on the pattern defined.

Regular expressions are used for performing all types of text search and replace operations. They are a compact way to describe string patterns that match a specific amount of text. Regular expressions are strings composed of delimiters, a pattern, and optional modifiers. The syntax used is `$exp = "/eliteschools/i";`. Here, the delimiter is `/`, `eliteschools` is the pattern that is being searched for and `i` is a modifier that makes the search case-insensitive.

Any character except an alphabet, number, backslash, or space can be a delimiter. The common delimiter that is most commonly used is the forward slash (`/`). However, if the pattern itself contains forward slashes, then one can choose any other character as the delimiters such as `#` or `~`.

There are two types of Regular Expressions.

POSIX Regular Expressions

PERL Style Regular Expressions

### 4.2.1 POSIX Regular Expressions

Portable Operating System Interface for uniX (POSIX) is a collection of standards that are used to define some of the functionality that should be supported by UNIX operating system. One such standard defines two types of regular expressions. POSIX regular expressions are to be used only with textual data. If data contains a NUL-byte (`\x00`), then, the regular expression will interpret that as the end of the string and matching will not happen beyond that point.

A POSIX regular expression's structure almost resembles an arithmetic expression. Numerous elements combine and form highly complex expressions. Any expression that is a match to a single character is the simplest regular expression.

Brackets `([])` can be used to find a range of characters. For example, if a match is to be found for any digit between 0 and 9, then, the expression is `[0-9]`. To get a match for a lowercase character between a and z, then use the expression `[a-z]`. To get a match for a character between uppercase A and uppercase Z, then use the expression `[A-Z]`. To find a match for a character between lowercase a and uppercase Z, then use the expression `[a-zA-Z]`.

### 4.2.2 PHP's Regexp POSIX Functions

At present, PHP has seven different functions to search for strings using POSIX-style regular expressions. They are as follows:

- `ereg()`: Using this function, search a string for a pattern.
- `ereg_replace()`: This function helps search for a pattern and if found, then replace it with the desired replacement.
- `eregi()`: This non-case sensitive function, searches for a string-specified string throughout a pattern-specified string.
- `eregi_replace()`: This function operates similar to `ereg_replace()`. However, this is not case sensitive.
- `split()`: This function helps in dividing a string into various elements. The boundaries for each element is based on how the string pattern occurs.
- `spliti()`: This non-case sensitive function is similar to `split()`.
- `sql_regcase()`: This function helps in converting an input parameter string's character into a bracketed expression that contains two characters.

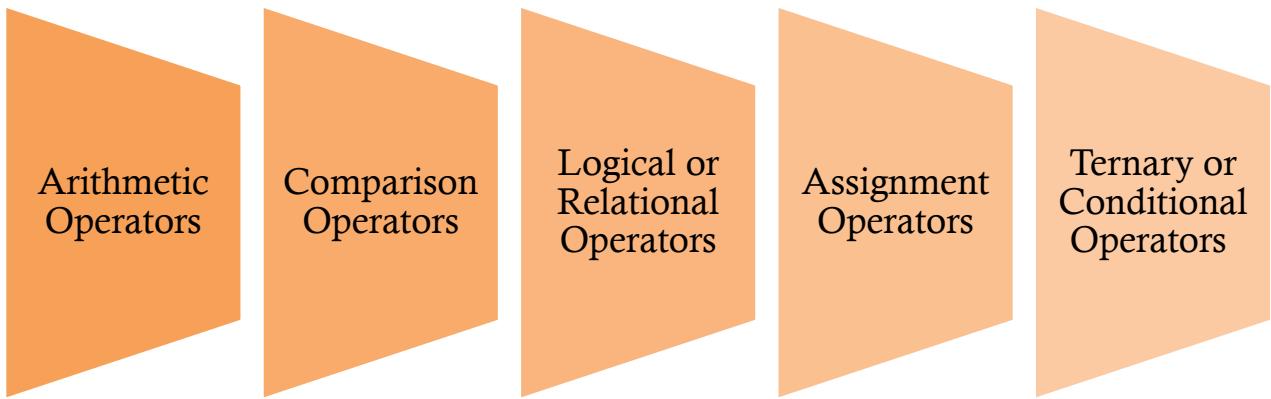
## 4.3 Operators in PHP

---

An operator is used to perform a variety of operations on variables and values. For example, take the expression  $6+5=11$ . Here, 6 and 5 are the operands and + is the operator.

### 4.3.1 Operator Types

There are different types of operators which perform different operations.



#### Arithmetic Operators

Arithmetic operators perform different arithmetic operations. For example, if variable A holds the value 30 and variable B holds the value 10, then the arithmetic operator \* is used to multiply both and given the result as 300.

Table 4.1 lists the arithmetic operators supported by PHP.

Operator	Name	Description
+	Addition	Returns the sum of the operands
-	Subtraction	Returns the difference between the two operands
*	Multiplication	Returns the product of two operands
/	Division	Returns the quotient after dividing the first operand by the second operand
%	Modulus	Returns the remainder after dividing the first operand by the second operand

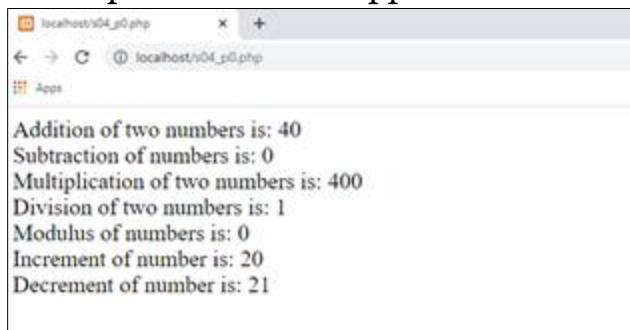
*Table 4.1: Arithmetic Operators in PHP*

Code Snippet 11 shows a sample program where all available arithmetic operators have been used.

### Code Snippet 11:

```
<?php  
$a = 20;  
$b = 20;  
$c = $a + $b;  
echo "Addition of two numbers is: $c <br/>";  
$c = $a - $b;  
echo "Subtraction of numbers is: $c <br/>";  
$c = $a * $b;  
echo "Multiplication of two numbers is: $c <br/>";  
$c = $a / $b;  
echo "Division of two numbers is: $c <br/>";  
$c = $a % $b;  
echo "Modulus of numbers is: $c <br/>";  
$c = $a++;  
echo "Increment of number is: $c <br/>";  
$c = $a--;  
echo "Decrement of number is: $c <br/>";  
?>
```

Figure 4.11 shows the output for Code Snippet 11.



**Figure 4.11: Output for Code Snippet 11**

Here, arithmetical operations are performed. Two given numbers are added and stored in variables A and B. The resulting value is stored in variable C. Similarly, subtraction, division, modulus, increment, and decrement of two given numbers are performed.

## Comparison Operators

Comparison operators help in comparing two operands. For example, if variable A holds the value 10 and variable B holds the value 40, then the comparison operator == is used to check if the value of the operands is equal and give the result as (A==B) as not true.

Table 4.1 lists the comparison operators supported by PHP.

Operator	Name	Description
==	Equal to	Returns true if both the operands are equal
====	Identical	Returns true if both the operands are equal and are of the same data type
!=	Not equal to	Returns true if the first operand is not equal to the second operand
<>	Not equal to	Returns true if the first operand is not equal to the second operand
!==	Not Identical	Returns true if the first operand is not equal to the second operand or they are not of the same data type
<	Less than	Returns true if the first operand is less than the second operand
<=	Less than or equal to	Returns true if the first operand is less than or equal to the second operand
>	Greater than	Returns true if the first operand is greater than the second operand
>=	Greater than or equal to	Returns true if the first operand is greater than or equal to the second operand

**Table 4.2: Comparison Operators in PHP**

Code Snippet 12 shows a sample program to compare two numbers using the comparison operator.

### Code Snippet 12:

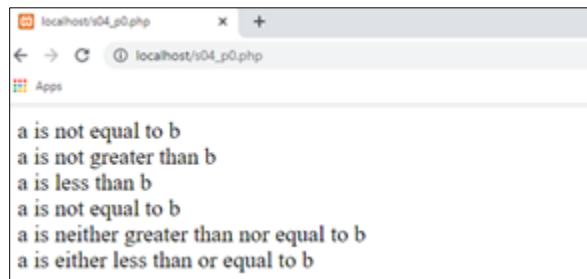
```
<?php
```

```

$a = 12;
$b = 20;
if( $a == $b ){
    echo " a is equal to b<br/>";
}
else{
    echo " a is not equal to b<br/>";
}
if( $a > $b ){
    echo " a is greater than b<br/>";
}
else{
    echo " a is not greater than b<br/>";
}
if( $a < $b ){
    echo " a is less than b<br/>";
}
else{
    echo " a is not less than b<br/>";
}
if( $a != $b ){
    echo " a is not equal to b<br/>";
}
else{
    echo " a is equal to b<br/>";
}
if( $a >= $b ){
    echo "a is either greater than or equal to b<br/>";
}
else{
    echo " a is neither greater than nor equal to b<br/>";
}
if( $a <= $b ){
    echo " a is either less than or equal to b<br/>";
}
else{
    echo " a is neither less than nor equal to b<br/>";
}
?>

```

Figure 4.12 shows the output for Code Snippet 12.



**Figure 4.12: Output for Code Snippet 12**

In Code Snippet 12, different comparison operators such as equal to, greater than, less than, not equal to, greater than or equal to, and less than or equal to are used. If the condition is true, it will display the first statement; otherwise, the else part is displayed.

## Logical Operators

Logical operators help to connect multiple expressions. PHP supports various logical operators. For example, if variable A holds the value 0 and variable B holds the value 10, then the comparison operator `and` is used to check if both operands are true and the condition is true or either a or b is false. Table 4.3 lists various logical operators.

Operator	Name	General Form	Description
AND <code>&amp;&amp;</code>	Logical AND operator	Expression1 AND Expression2  Expression1 <code>&amp;&amp;</code> Expression2	Returns true only if both the expressions are true
OR <code>  </code>	Logical OR operator	Expression1 OR Expression2  Expression1 <code>  </code> Expression2	Returns true if any one of the expressions is true
XOR	Logical XOR operator	Expression1 XOR Expression2	Returns true if either Expression 1 or Expression 2 is true, but not both
!	Logical NOT operator	<code>!Expression</code>	Returns true only if the condition is not true

*Table 4.3: Logical Operators in PHP*

Code Snippet 13 shows a sample program to compare two numbers using logical operator.

### Code Snippet 13:

```
<?php
$a = 0;
$b = 10;
```

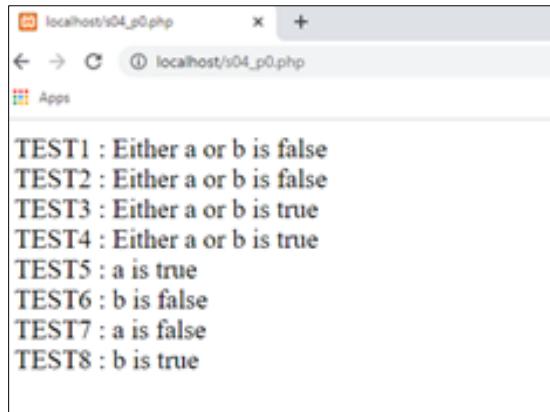
```

if( $a && $b ){
    echo "TEST1 : Both a and b are true<br/>";
} else{
    echo "TEST1 : Either a or b is false<br/>";
}
if( $a and $b ){
    echo "TEST2 : Both a and b are true<br/>";
} else{
    echo "TEST2 : Either a or b is false<br/>";
}
if( $a || $b ){
    echo "TEST3 : Either a or b is true<br/>";
} else{
    echo "TEST3 : Both a and b are false<br/>";
}
if( $a or $b ){
    echo "TEST4 : Either a or b is true<br/>";
} else{
    echo "TEST4 : Both a and b are false<br/>";
}

$a = 10;
$b = 0;
if( $a ){
    echo "TEST5 : a is true <br/>";
} else{
    echo "TEST5 : a is false<br/>";
}
if( $b ){
    echo "TEST6 : b is true <br/>";
} else{
    echo "TEST6 : b is false<br/>";
}
if( !$a ){
    echo "TEST7 : a is true <br/>";
} else{
    echo "TEST7 : a is false<br/>";
}
if( !$b ){
    echo "TEST8 : b is true <br/>";
} else{
    echo "TEST8 : b is false<br/>";
}
?>

```

Figure 4.13 shows the output for Code Snippet 13.



**Figure 4.13: Output for Code Snippet 13**

In Code Snippet 13, all types of logical operators are used. If variable `a` is assigned by any value other than 0, it becomes true; otherwise, it is false. The same applies to variable `b` too. Here, you are checking if both `a` and `b` values are true. The condition becomes false; therefore, the output is either `a` or `b` is false. For `or` logical operator, if any one variable is true it will enter the `true` Block of `if` statement. Therefore, the output for `or` logical statement for variable `a` and `b` is either `a` or `b` is true.

## Assignment Operators

Assignment operators are used in writing values to variables.

Besides the primary assignment operator represented by the symbol `=`, PHP also supports combined assignment operators.

For example, `=` is a simple assignment operator, which will help in assigning values from right-side operands to left-side operands, that is, `C = A + B` will assign the value of `A + B` into `C`.

`+=` and `-=` are examples of combined assignment operators that perform addition and subtraction in combination with assignment respectively.

Combined operators can also be used with array union and string operators.

Table 4.4 displays examples with the combination operators.

Shorthand Expression	Description
<code>\$a += \$b</code> $\$a = \$a + \$b$	Adds \$a and \$b and assigns the result to \$a
<code>\$a -= \$b</code> $\$a = \$a - \$b$	Subtracts \$b from \$a and assigns the result to \$a
<code>\$a *= \$b</code> $\$a = \$a * \$b$	Multiplies \$a and \$b and assigns the result to \$a
<code>\$a /= \$b</code> $\$a = \$a / \$b$	Divides \$a by \$b and assigns the quotient to \$a
<code>\$a %= \$b</code> $\$a = \$a \% \$b$	Divides \$a by \$b and assigns the remainder to \$a
<code>\$a .= \$b</code> $\$a = \$a . \$b$	Concatenates \$b with \$a and assigns the result to \$a

**Table 4.4: Shorthand Operators**

Code Snippet 14 shows a sample program using the assignment operators.

#### Code Snippet 14:

```
<?php
$a = 30;
$b = 20;

$c = $a + $b;      /* Assignment operator */
echo "Addition Operation Result: $c <br/>";

/* Value of c becomes 30 + 20 = 50 */
$c += $a;

/* Value of c becomes 30 + 20 + 30 = 80 */
echo "Add AND Assignment Operation Result: $c <br/>";
$c -= $a;           /* Value of c becomes 80 - 30 = 50

echo "Subtract AND Assignment Operation Result: $c <br/>";
$c *= $a;
/*Value of c was 50 * 30 = 1500 */
echo "Multiply AND Assignment Operation Result: $c <br/>;

$c /= $a;
/* Value of c becomes 1500/30 = 50*/

echo "Division AND Assignment Operation Result: $c <br/>";
$c %= $a;
echo "Modulus AND Assignment Operation Result: $c <br/>";
?>
```

Figure 4.14 shows the output for Code Snippet 14.

```
Addition Operation Result: 50
Add AND Assignment Operation Result: 80
Multiply AND Assignment Operation Result: 50
Division AND Assignment Operation Result: 1.66666666666667
Modulus AND Assignment Operation Result: 1
```

**Figure 4.14: Output for Code Snippet 14**

In Code Snippet 14, `a` has value 30 and `b` has value 20. First, values of both `a` and `b` are added. Then, the shortcut assignment operators are used one by one. First one is `+=`, so the result of `c` is added with value of variable `a` (30), which when expanded evaluates to `c=c+a`. Here, value of `c` is 50, then it is added by value 30. Therefore, the result stored in `c` is 80 and output of add assignment operation is 80.

Similarly, the Subtract operation can be expanded as `c=c-a`. The result becomes  $80-30=50$ , so output of `c` variable for shortcut subtract operation is 50. This works similarly for remaining arithmetic operations. Modulus operation yields remainder obtained by performing division operation of any two numbers.

## Conditional Operator

The conditional operator is used to perform various operations that are based on different conditions. PHP supports the conditional operator `?::`. It is used to first evaluate an expression for a TRUE or FALSE value and then, execute one of the two given statements depending upon the result of the evaluation. It is commonly used to evaluate the `if-then-else` operation. Code Snippet 15 shows a program to demonstrate the use of the conditional operator.

### Code Snippet 15:

```
<?php
$a = 10;
$b = 20;
/* If condition is true then assign a to result otherwise b */
$result = ($a > $b) ? $a :$b;
```

```

echo "TEST1 : Value of result is $result<br/>";
/* If condition is true then assign a to result otherwise b */
$result = ($a < $b ) ? $a :$b;
echo "TEST2 : Value of result is $result<br/>";
?>
</body>
</html>

```

Figure 4.15 shows the output for Code Snippet 15.



*Figure 4.15: Output for Code Snippet 15*

In Code Snippet 15, conditional operators that are also called comparisons operators are shown. If the condition is true, then only it will display first statement of if condition; otherwise, else part will be displayed. Here, value of `a` and `b` are compared; value of `a` is 10 and `b` is 20. In the first TEST, you are checking if `a` is greater than `b`. If yes, then it will display value of `a` variable; otherwise, the value of `b`. Here, the condition is false, therefore it displays value of `b` in TEST 1. In the second TEST, you are checking if `a` is smaller than `b`. If yes, then it will display value of `a` variable; otherwise, the value of `b`. However, the condition is true, therefore it displays value of `a` in TEST 2.

#### 4.3.2 Operator Categories

On the basis of the number of operands used, all the operators that have been discussed so far, can be categorized into different types as shown in Table 4.5.

Operator Category	Description
Unary prefix operators	Precede a single operand
Binary operators	Take two operands and execute different arithmetic and logical operations

Operator Category	Description
Conditional or ternary operators	Take three operands. Based on the how the first expression is evaluated, either the second or third expression will be evaluated
Assignment operators	Assign a value to a variable

*Table 4.5: Operator Categories*

#### 4.3.3 Precedence of PHP Operators

The precedence of operators decides the order of execution of operators in an expression. Some operators have a precedence than others. For example, the multiplication operator has higher precedence than the addition operator. If the expression is  $2+6*3$ , then the multiplication operator will be executed first, and then, the addition operator. The result is 20. This is because the multiplication operator has higher precedence than the addition operator.

To override precedence and force an operation, use parentheses. Using the same example  $2+6*3$ , now write is as  $(2+6)*3$ . In this instance, the addition operator takes precedence over the multiplication operator. The result is 24.

A few operators have an equal level of precedence. In such a case, the order of operations is decided by the order of associativity, that is left or right. For example, if using additive operators such as  $+-$ , then the order of associativity is from left to right.

## 4.4 Summary

- Variables can be used to store and access data during program execution.
- Three types of variables, local, global, and static are supported by PHP.
- There are different rules to be followed while naming a variable.
- Variables can be integer, Boolean, doubles, character strings, and floating-point numbers.
- A string literal can be specified in different ways such as single quoted, double quoted, heredoc, or nowdoc.
- You can categorize operators as unary prefix, binary, ternary, or assignment operators.
- PHP language provides support to different operator types such as arithmetic, comparison, logical, assignment, and conditional operators.
- A regular expression is a character sequence which forms a search pattern.

## 4.5 Test Your Knowledge



1. Which of the following is the correct method of declaring a variable in PHP?
  - a. \$variable name = value;
  - b. \$variable\_name = value;
  - c. \$variable\_name = value
  - d. \$variable name as value;
2. Which of the following is the correct use of the `strcmp()` function in PHP?
  - a. The `strcmp()` function is used for comparing the strings excluding case.
  - b. The `strcmp()` function is used for comparing the uppercase strings.
  - c. The `strcmp()` function is used for comparing the lowercase strings.
  - d. The `strcmp()` function is used for comparing the strings including case.
3. Which of the following is an invalid variable name?
  - a. \$newVar
  - b. \$new\_Var
  - c. \$new-var
  - d. All of these
4. String values in PHP must be enclosed within \_\_\_\_\_.
  - a. Double Quotes
  - b. Single Quotes
  - c. Double Quotes and Single Quotes
  - d. Double //

5. \_\_\_\_\_ can be used to test the type of any variable?
- a. showtype()
  - b. gettype()
  - c. setttype()
  - d. type()

## **Answers to Test Your Knowledge**

---

1. `$variable_name = value;`
2. The `strcmp()` function is used for comparing the strings including case
3. The `strcmp()` function is used for comparing the lowercase strings
4. Double Quotes and Single Quotes
5. `gettype()`

## 4.6 Try It Yourself

1. Write a program to determine addition of three given numbers.
2. The formula for calculating the simple interest is  $p * n * r / 100$ . Here 'p' stands for principal amount, 'n' stands for number of years and 'r' stands for rate of interest. You are given the input values for all three. Write a PHP program to compute the simple interest payable.

# Session 5

## Conditional and Flow Control Statements and Arrays



### Learning Objectives

*In this session, students will learn to:*

- List and explain different types of conditional statements and their usage
- Distinguish between `if`, `if...else`, and `if...elseif...else` statements
- Define and use `switch` statement
- List and explain different types of loop statements and their usage
- Distinguish between `while`, `do...while` and `for` loops
- Identify use of `foreach` loop statement
- Identify use of `break` and `continue` statement
- Outline different types of Arrays
- List the uses of different types of Sorting functions

Conditional and looping constructs are the lifeline of any programming language and PHP is no different. This session begins by explaining use of conditional statements in PHP. It introduces different types of conditional and loop statements, explaining how and where to use them. Then, it explains which type of constructs are to be used in different scenarios. Further, it introduces the concept of arrays in PHP and continues to explain different types of arrays. The session finally concludes after explaining various functions in PHP used for sorting arrays.

### **5.1 PHP Conditional Statements**

---

Conditional statements in PHP help the user to arrive at a decision based on certain conditions. When a user writes code in PHP, there will be scenarios where different actions must be performed for different conditions.

Conditional statements are used in such scenarios. These conditions are defined by a set of conditional statements which contain expressions that are evaluated to a Boolean value of true or false.

PHP language supports the following conditional statements:

**if**

Executes some code when one condition is true. In this case, the `if` statement only executes the code when the condition is **true**.

**if...else**

When the condition is true, the statement/statement block following `if..else` will be executed. In case condition is false, then statement after the `else` will be executed.

**if...elseif  
...else**

If there are more than two conditions, then `if...elseif...else` statement executes different codes.

**switch**

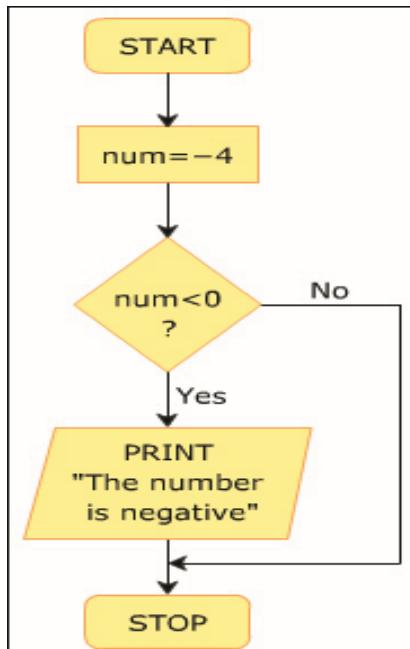
Selects one of many code blocks to be executed.

### 5.1.1 `if` Statement

Consider a real-life analogy - you plan to go out, but it is monsoon season so you first decide to check the sky outside. If it looks cloudy, you must carry an umbrella/raincoat. On the other hand, if the weather looks sunny and clear, you do not require to carry them. Thus, the act of carrying or not carrying rain gear depends on the condition of the sky. Such decision-making flow of action is accomplished in programming through the use of conditional statements.

An `if` statement allows you to execute one or more statements after evaluating a specified logical condition. It starts with the `if` keyword and is followed by the condition. If the condition evaluates to true, the block of statements following the `if` statement is executed. If the condition evaluates to false, the block of statements following the `if` statement is ignored and the statement after the block is executed.

Consider the flowchart representing an `if` statement in Figure 5.1. In the figure, the variable `num` is assigned by value -4. The code checks for the condition whether `num` is less than zero. If this condition is true, it prints 'the number is negative' , however, if the condition evaluates false, the code ends.



**Figure 5.1: Flowchart Representing if Statement**

Syntax for `if` statement is as follows:

### Syntax

```
if (condition) {
    code to be executed when the condition is true;
}
```

Code Snippet 1 demonstrates the logic in the flowchart through a small code block.

### Code Snippet 1:

```
<?php
$num = -764;
if ($num < 0) {
    echo "The number is negative";
}
?>
```

### Output:

The number is negative.

Code Snippet 2 shows a sample program that uses `if` statement to display '**'Happy Morning'** if the current time is less than 12 o'clock.

## **Code Snippet 2:**

```
<?php  
$timeofday = date("H");  
if ($timeofday < "12") {  
    echo "Happy Morning!";  
}  
?>
```

In Code Snippet 2, the program is executed when current time is less than 12 o'clock. Hence, the output is 'Happy Morning!'

### **Output**

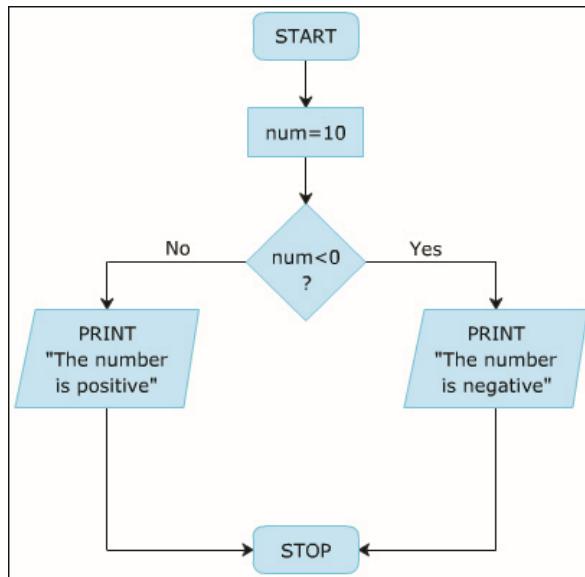
Happy Morning!

#### **5.1.2 if...else Statement**

The `if` statement executes a block of statements only if the specified condition is true. However, in some situations, it is required to define an action for a false condition. This is done using an `if...else` construct.

The `if...else` construct starts with `if` block followed by an `else` block. The `else` block starts with `else` keyword followed by a block of statements. If the condition specified in the `if` statement evaluates to false, the statements in the `else` block are executed.

Consider the flowchart representing `if...else` statement in Figure 5.2. Here, `num` variable is assigned to 10. Code checks the condition if `num` is less than zero. If condition is true, it will print 'The number is positive' and program ends. However, if condition evaluates to false, it will print the statement 'The number is negative' and program ends. Code Snippet 3 demonstrates the logic in the flowchart through a small code block.



*Figure 5.2: Flowchart for if...else Statement*

### Code Snippet 3:

```

<?php
$num=10;
if ($num<0) {
    echo "The number is negative";
}
else {
    echo "The number is positive";
}
?>
  
```

### Output

The number is positive

Syntax for if...else statement is as follows:

### Syntax

```

if (condition) {
    Code to be executed when the condition is true;
}
else {
    Code to be executed when the condition is false;
}
  
```

Code Snippet 4 demonstrates the use of if...else statement.

## Code Snippet 4:

```
<?php
$hourOfDay = date("H");
if ($hourOfDay < "18") {
    echo "Have a Nice Day ahead!";
}
else {
    echo "Good Night!";
?
>
```

In Code Snippet 4, the `else` statement will be executed and the output displayed is **Good Night!** as the current time hour is greater than 18.

However, if the time hour is less than 18, then `if` statement executes to display **Have a Nice Day ahead!**

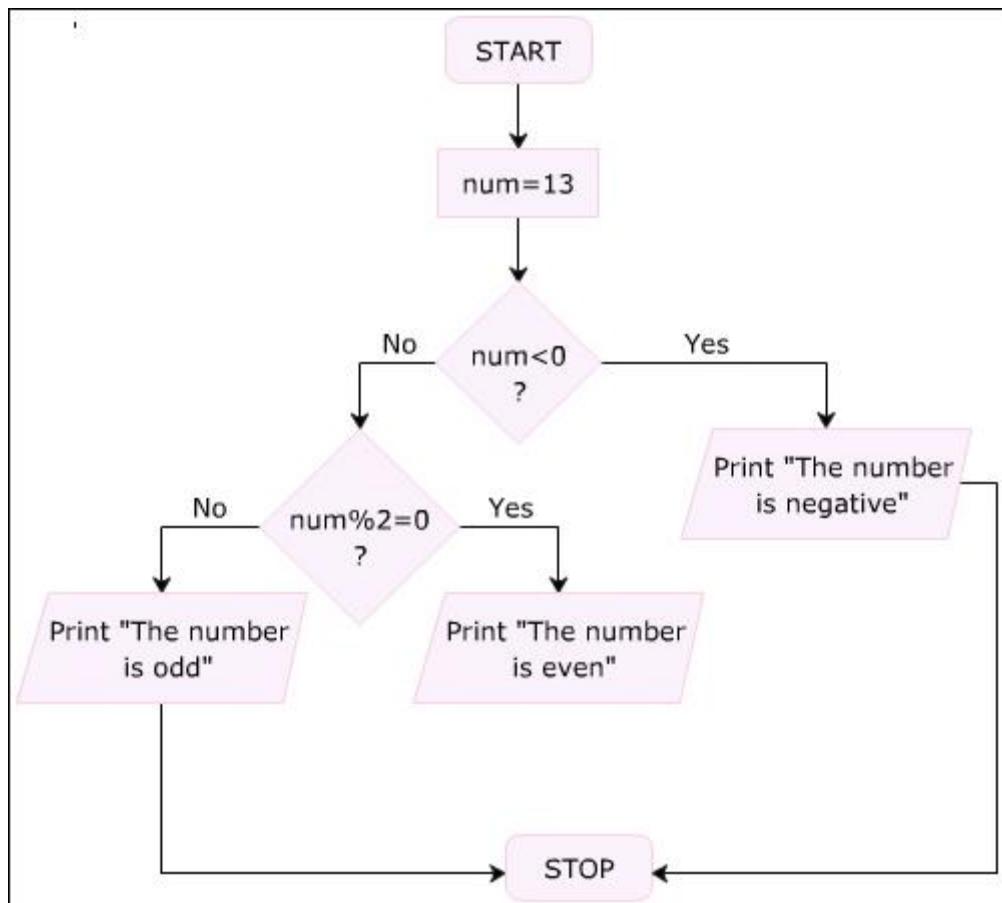
## Output

Good Night!

### 5.1.3 `if...elseif...else` Statement

Statement `if...elseif...else` is used when the user wants to handle multiple conditions. In other words, `if...elseif... else` statement executes different codes when there are more than two conditions.

Consider the flowchart representing `if...elseif...else` statement in Figure 5.3. Here, `num` variable is assigned to 13. Code checks whether `num` is less than 0, if condition is true then it prints ‘The number is negative’ and program ends. If condition evaluates false, `if elseif` statement is checked that is `num%2=0`. This statement checks if the number is divisible by 2. If it evaluates true then, it prints ‘The number is even’, if it evaluates false then, it will print ‘The number is odd’. Code Snippet 5 demonstrating the flowchart with a small code block.



*Figure 5.3: Flowchart for if...elseif...else Statement*

### Code Snippet 5:

```

<?php
$num = 1397;
if ($num < 0) {
    echo "The number is negative";
} elseif ($num%2==0) {
    echo "The number is even";
} else {
    echo "The number is odd";
}
?>

```

### Output:

The number is odd

Syntax for if...elseif...else statement is as follows:

## Syntax

```
if (condition) {  
    Executes when this condition of the code is true;  
} elseif (condition) {  
    Executes when the first condition of the code is false and  
    the second condition is true;  
} else {  
    Executes when all conditions of the code are false;  
}
```

Code Snippet 6 illustrates the use of if...elseif...else statement.

### Code Snippet 6:

```
<?php  
$t = date("H");  
echo "<p>The hour (of the server) is ". $t;  
echo " and message is:</p>";  
if ($t < "12") {  
    echo "Happy morning!";  
} elseif ($t < "20") {  
    echo " Nice day!";  
} else {  
    echo " Good night!";  
}  
?>
```

In the code, if...elseif...else statement is used to display '**Happy morning**' if current hour is less than 12 and '**Nice day**' if 20 is less than the current time. Else, output is '**Good night**'. Since the hour of the server is 15, output displayed is Good Day!

## Output

The hour (of the server) is 15 and message is:  
Happy Morning!

### 5.1.4 Switch Statement

In PHP, switch statement is used to take different actions based on different conditions. It is often used as an alternative for multiple if...elseif conditions.

Syntax for `switch` statement is as follows:

## Syntax

```
switch (n) {  
    case label1:  
        execute the code block when label1=n;  
        break;  
    case label2:  
        execute the code block when label2=n;  
        break;  
    case label3:  
        execute the code block when label3=n;  
        break;  
    ...  
    default:  
        execute the code block when n differs from all labels;  
}
```

In the first step, the expression `n` is evaluated once and various code blocks are executed based on the value of `n`. Expression's value is then equated to the label values for each case in the structure. If they match, the code block assigned to a particular case is executed. The `break` at the end of the code block prevents the code from automatically running into the next case. If no match is found, the `default` statement is used.

Code Snippet 7 shows how to use the `switch` statement.

### Code Snippet 7:

```
<?php  
$language = "PHP 8";  
switch ($language) {  
    case "C":  
        echo "Your favorite language is C ";  
        break;  
    case "PHP 8":  
        echo "Your favorite language is PHP 8";  
        break;  
    case "C++":  
        echo "Your favorite language is C++";  
        break;  
    default:  
        echo "Your favorite language is neither C, PHP 8, nor  
C++";  
}  
?>
```

In Code Snippet 7, the value stored in the variable language is PHP 8 and the control does not match the first case. Following that, the control matches the second case and the variable value. As a result, the output displayed is: Your favorite language is PHP 8.

In this code, the value for language is hard coded. In practical situations, however, the value would be accepted from the user.

## Output

Your favorite language is PHP 8!

## 5.2 PHP Loops

Similar to any other programming language, loops in PHP are used to repeat the same block of code until a specific condition is met.

PHP supports four types of loops: while, do...while, for, and foreach.

### 5.2.1 while Loop

In the while loop, the condition is checked first, and if the condition is met, then, the code block is executed until the condition becomes false.

#### Syntax

```
while (condition is true) {  
    code to be executed;  
}
```

Code Snippet 8 shows a sample program for displaying numbers from 1 to 10 using a while loop.

#### Code Snippet 8:

```
<?php  
$n = 1;  
while($n <= 10) {  
    echo "The number is: $n <br>";  
    $n++;  
}  
?>
```

In Code Snippet 8, the value of n assigned is initially 1, so the first number in the output is 1.

Next, `n` is incremented with the `++` operator, which is `n++`. The next numerical value becomes 2 and likewise, up to the number 10 will be displayed because here, `while` condition is `n <= 10`.

## Output

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

### 5.2.2 PHP do...while Loop

`do...while` loop runs the code block and executes before checking the condition. If the condition is true, the process is repeated until the condition becomes false.

## Syntax

```
do {
    code to be executed;
} while (condition is true);
```

Code Snippet 9 shows a sample program to display numbers from 1 to 5 using `do...while` loop.

### Code Snippet 9:

```
<?php
$n = 15;
do {
    echo "The number is: $n <br>";
    $n++;
} while ($n <= 20);
?>
```

In Code Snippet 9, initially the value of `n` assigned is 15 so the first number in output is 15. Next, `n` is incremented using `++` operator that is `n++`, hence the next number value becomes 16 and then, it checks the condition likewise till number 20 as the `while` condition is `n` is less than or equal to 20.

There is a major main difference between `while` and `do-while`. The `do-while` loop executes instruction first and then, checks condition, whereas, the `while` loop checks condition first and then, executes instruction.

## Output

```
The number is: 15  
The number is: 16  
The number is: 17  
The number is: 18  
The number is: 19  
The number is: 20
```

### 5.2.3 PHP for Loop

`for` loop in PHP is used to repeat a block of code a fixed number of times. In other words, when a user already knows how often to run the code, then, `for` loop can be used.

## Syntax

```
for (init counter; test counter; increment counter) {  
    execute the code block for each iteration;  
}
```

Following parameters are used in `for` loop:

`init counter` is used to initialize the counter value of loop.

`test counter` is used to evaluate iteration for each loop. The loop continues if it evaluates to TRUE. The loop ends if it evaluates to FALSE.

`increment counter` is used to increase the counter value of loop.

Code Snippet 10 shows a sample program to display numbers from 11 to 20 using `for` loop.

## Code Snippet 10:

```
<?php  
for ($n = 11; $n <= 20; $n++) {  
    echo "The number is: $n <br>";  
}  
?>
```

Code Snippet 10 initially, the value of n is assigned 11 so the first number in output is 11, next n is incremented using `++` operator that is `n++`, so the next number value becomes 12. Then, it checks the condition likewise till number 20.

## Output

```
The number is: 11  
The number is: 12  
The number is: 13  
The number is: 14  
The number is: 15  
The number is: 16  
The number is: 17  
The number is: 18  
The number is: 19  
The number is: 20
```

Here, `for` loop is used, in which assignment of variable value, checking of condition and increment of variable is done in single line only compared to a `while` and `do-while` loop.

## 5.3 foreach Loop Statement

The `foreach` loop is useful to iterate over array of elements. It only works with arrays and loops over all of the key/value pairs in an array.

### Syntax

```
foreach ($array as $value) {  
    code to be executed;  
}
```

For each of the loop iteration, the current array element's value is assigned to `$value` and the array pointer is moved by one until it reaches the last array element.

Code Snippet 11 shows a sample program that uses a `foreach` loop to display complete array's elements.

## Code Snippet 11:

```
<?php  
$vehicles = array("Car", "Bike", "Bus", "Bicycle");  
foreach ($vehicles as $value) {  
    echo "$value <br>";  
}  
?>
```

In Code Snippet 11, array values are displayed using `foreach` loop. It is usually used in the case of arrays only, so here variable `vehicles` holds an array of elements such as Car, Bike, Bus, and Bicycle. Variable `value` is used to display each array element.

### Output

Car  
Bike  
Bus  
Bicycle

Code Snippet 12 shows a sample program to display both keys and values of the array using `foreach` loop.

## Code Snippet 12:

```
<?php  
$flower = array("Rose"=>"10", "Lily"=>"20", "Lotus"=>"30");  
foreach($flower as $f => $value) {  
    echo "$f= $value<br>";  
} ?>
```

Code Snippet 12 displays flower names and their numbers using `foreach` loop. Here, it is displaying both keys and values that are using `=>` operator in the array and `foreach` loop.

### Output

Rose=10  
Lily=20  
Lotus=30

The **double arrow operator** `=>` is used as an access mechanism for arrays. It indicates that whatever is on the left of it will have a corresponding value of whatever is on the right in array context. This can be used to set values of any acceptable type into a corresponding index of an array.

## 5.4 break Statement

---

`break` statement is used to jump out of a loop construct. For instance, when `x` equals 4, the loop exits. `break` statement is also used in `switch case` statements. It is mainly used in a program to emerge out of a loop or exit from a code block. The program exits a particular block or loop when it encounters `break` and executes the code after the loop or block.

Code Snippet 13 shows a sample program to display the use of `break` statement and how to jump out of a loop.

### Code Snippet 13:

```
<?php
for ($n = 5; $n < 15; $n++) {
    if ($n == 9) {
        break;
    }
    echo "Number is: $n <br>";
}
?>
```

### Output

```
Number is: 5
Number is: 6
Number is: 7
Number is: 8
```

In Code Snippet 13, `for` loop is used to display numbers from 5 to 14, but a `break` is given within `if` block. Here, the loop iterates from 5 onwards but when the number reaches 9, program control will come out of the `for` loop and further numbers will not be displayed. Hence, only 5 to 8 are displayed in the output.

## 5.5 continue Statement

---

When a specified condition is met, the `continue` statement breaks an iteration in the loop and continues with the next iteration in the loop. If there are one or more statements in the loop following the `continue`, they will all be skipped as control will omit them and directly proceed to the next iteration.

Code Snippet 14 shows a sample program to display the use of `continue` statement and display values from 0 to 9 and skip number 4.

## Code Snippet 14:

```
<?php
for ($n = 0; $n < 10; $n++) {
    if ($n == 4) {
        continue;
    }
    echo "The number is: $n <br>";
}
?>
```

In Code Snippet 14, `for` loop is used to display numbers from 0 to 9, but a `continue` statement is given in the `if` block. Hence, when the value reaches 4, program control will not come out of `for` loop, but skips further statements and proceeds to the next iteration. As a result, 0-3 and 5-9 are the values displayed in the output.

## Output

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
```

## 5.6 Concepts of Arrays in PHP

---

Assume five names have to be saved and printed. This task is easily accomplished by using five distinct string variables. However, if the number increases from five to a hundred, it will be difficult for the user or developer to construct so many separate variables.

Consider another example. If you have a list of items, such as car names, storing the cars in single variables might look like this:

```
$cars1 = "Ford";
$cars2 = "Aston Martin";
$cars3 = "Toyota";
```

However, if the car list has 300 names, it becomes impractical to have 300 variables. It gets chaotic to repeat the same block of code in the program, to run a common block of code for all 300 names. In such scenarios, an array

comes into play and helps store every element within a single variable. An array is a special type of data structure that can store one or more similar types of values. Arrays help store a list of elements of similar types that can be accessed via their index or key, in a single variable. This saves the difficulty of creating a distinct variable for each data.

**array()** function is used to create an array in PHP.

An array can store multiple values with the same name and the values can be accessed by referencing an index number.

For example, if you want to hold 300 numbers, then instead of defining 300 variables, it is easy to define an array of length 300.

Arrays are classified into three types:

**Numerical array:** This array, as the name implies, has numerical index. The values are stored and accessed in a linear manner.

**Associative array:** This is an array with strings as an index. Rather than a strict linear index order, this array stores element values in association with specific key values.

**Multi-dimensional array:** An array containing one or more arrays and values are accessed using multiple indices.

### 5.6.1 Numerical Array

These arrays can hold numbers, strings, or any other object, but their indexes are represented by numbers. The array index is set to zero by default.

Code Snippet 15 shows a sample program that shows how to create and access numeric arrays.

#### Code Snippet 15:

```
<?php
/* Method for creating an array. */
$num = array(21, 22, 23, 24, 25, 26);
foreach( $num as $value )
{
echo "Value is $value <br />";
}
$num[0] = "one";
$num[1] = "two";
```

```
$num[2]=23;  
foreach( $num as $value )  
{  
echo "Value is $value <br />";  
}  
?>
```

In Code Snippet 15, an array is created and uses a `foreach` loop to display numbers ranging from 21 to 26.

## Output

```
Value is 21  
Value is 22  
Value is 23  
Value is 24  
Value is 25  
Value is 26  
Value is one  
Value is two  
Value is 23
```

### 5.6.2 Associative Array

In terms of functionality, associative arrays are very similar to numeric arrays, but they differ index-wise. Users can create a strong association between the key and values using an associative array with a string as an index. Refer to Code Snippet 16, where `John` is a key or index in the form of string for value 20, `Age['John']=20`.

In numerical arrays numbers are used as index. In Code Snippet 15, index of an array are in the form of numbers that is `num[0]="one"`, `num[2]="two"`, and so on.

To hold employee wages in an array, a numerically indexed array is not the right choice. Instead, the names of the employees should be the key in the associative array, with their respective salaries as the value. There are two ways to store values of arrays, in the first method, `array` keyword is used to create an array. In a single statement, all the values with associated keys can be stored.

```
$Age=array("John" => 20,"Roger" => 10,"Susan" => 60);  
      ↘ Key    ↙ Value
```

In the second method, `array` keyword is not used for creating arrays. Single line is used to store single value this results in storing values in multiple lines. For string index, only single quote is used to store and display the values.

That is, `$Age['John'] = "Adult";` ← value  
                    ↑  
                    String index

**Note:** The associative array should not be kept inside double quotes while printing, otherwise it will not return any value.

Code Snippet 16 shows a sample program to create an associative array.

### Code Snippet 16:

```
<?php  
/* One approach to create an associative array. */  
$Age = array(  
"John" => 20,  
"Roger" => 10,  
"Susan" => 60  
);  
echo "Age of John is ". $Age['John'] . "<br />";  
echo "Age of Roger is ". $Age['Roger'] . "<br />";  
echo "Age of Susan is ". $Age['Susan'] . "<br />";  
/* Another approach to create an associative array. */  
$Age['John'] = "Adult";  
$Age['Roger'] = "Child";  
$Age['Susan'] = "Senior Citizen";  
echo " John is ". $Age['John'] . "<br />";  
echo " Roger is ". $Age['Roger'] . "<br />";  
echo " Susan is ". $Age['Susan'] . "<br />";  
?>
```

Arrays are used in this code to store values. Array is assigned to a variable called `Age` and using the variable name, the values of array are displayed. Variable and key associated with a particular value is displayed. That is, in the first statement, `John` is the key and the value associated is 20. Hence, the output - Age of John is 20. Roger is the key associated with value 10 and so on.

## Output

```
Age of John is 20
Age of Roger is 10
Age of Susan is 60
John is Adult
Roger is Child
Susan is Senior Citizen
```

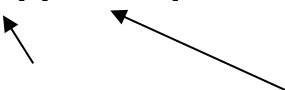
### 5.6.3 Multi-dimensional Arrays

A multi-dimensional array is one where each element in the main array can be an array. Each element in the sub-array can also be an array, and so forth.

Multiple indexes are used to access the values in the multi-dimensional array.

For example, to access the values of an array, name of a variable should be used that is, contacts in which array values are assigned. Hence, one must first write the name of the variable, array index, and array sub index as shown here:

```
$contacts[1] ['Name']="Peter";
```



Array Index                      Sub Index for accessing Peter

Code Snippet 17 shows a sample program to create a two-dimensional array to store contact information of three people that is name, email id, and number.

#### Code Snippet 17:

```
<?php
$contacts = array(
    array(
        "Name" => "David",
        "Email" => "David12@gmail.com",
        "Number" => 39365421
    ),
    array(
        "Name" => "Peter",
        "Email" => "Peter@gmail.com",
        "Number" => 299853412
    ),
)
```

```

=> array
(
"Name" => "John",
"Email" => "John13@yahoo.com",
"Number" => 39982145
)
);
/* Accessing multi-dimensional array values */
echo "Email ID of Peter : " ;
echo $contacts[1]['Email'] . "<br />";
echo "Contact number of David : ";
echo $contacts[0]['Number'] . "<br />";
echo "Contact number of John : ";
echo $contacts[2]['Number'] . "<br/>";
?>

```

In Code Snippet 17, multi-dimensional arrays are used to store names, emails, and contact numbers of three people. Here, a person's email ID and contacts are fetched from the array and displayed in the output. By specifying position and name of the array, one can get the data. Here, \$contacts[0] contains Name, Email, and Number of David, whereas, \$contacts[1] contains Name, Email, and Number of Peter. The array \$contacts[2] contains Name, Email, and Number of John.

## Output

```

Email ID of Peter: Peter@gmail.com
The contact number of David: 39365421
The contact number of John: 39982145

```

### 5.6.4 Indexed Arrays

An indexed array is an array which is defined using an index number. By default, index starts from zero. All elements in an array are represented by an index. In PHP, these types of index arrays are used to store strings, numeric, or any objects.

There are two methods to create index arrays:

#### Method 1

Automatically assign the index (index always starts at 0):

```
$Student=array("Peter", "John", "David");
```

## Method 2

Manually assign the Index:

```
$Student[0]= "Peter"  
$Student[1]= "John"  
$Student[2]= "David"
```

In the first method, to create an array and to store the values, keyword `array` is used. Here, variable name which is assigned with an array can be declared only in one line that is: `$Student=array("Peter", "John", "David")`. There is no requirement to write index numbers 0, 1, 2, and so on.

In the second method, `array` keyword is not used, instead variable name and index numbers are written, that is:

```
$Student[0]= "Peter"  
$Student[1]= "John"  
$Student[2]= "David"
```

In the second method, array values are declared in multiple lines.

Code Snippet 18 shows a sample program for creating an indexed array named `$Student`, assigning it with four elements. The output is a text that contains the array values.

### Code Snippet 18:

```
<?php  
$student=array("Peter", "John", "David", "Sean");  
echo "Names of the students are: ". $Student[0].  
", ".$Student[1]. ", ".$Student[2]. ", and ".$Student[3]. ".  
?>
```

In Code Snippet 18, the student names are displayed using array indexes. In this code, method 1 of assigning index automatically is used. Array is declared in a single line using keyword `array`. This method is used when the programmer wants an array to be declared in single line.

### Output

Names of the students are: Peter, John, David, and Sean.

### 5.6.5 Sorting Arrays

PHP includes a range of built-in functions for sorting array elements in various ways, such as alphabetically or numerically in ascending or

decreasing order. Let us look at some of the most popular functions for sorting arrays.

- `sort()` and `rsort()`: Indexed arrays are sorted using this array.
- `ksort()` and `krsort()`: Associative arrays are sorted by key using this array.
- `asort()` and `arsort()`: These are used to sort associative arrays by value.

Sorting enhances the ability to search significantly and also aids in rising or decreasing patterns based on a linear relationship between the data components. Table 5.1 lists some of the Sort functions for arrays in PHP.

Sort Functions	Description
<code>sort()</code>	Sort arrays in ascending order
<code>rsort()</code>	Sort arrays in descending order
<code>asort()</code>	Sort associative arrays in ascending order, according to the value
<code>ksort()</code>	Sort associative arrays in ascending order, according to the key
<code>arsort()</code>	Sort associative arrays in descending order, according to the value
<code>krsort()</code>	Sort associative arrays in descending order, according to the key

**Table 5.1: Array Sort Functions**

### Sort Array in Ascending Order - `sort()`

In Code Snippet 19, the elements of the `$Student` array are sorted in ascending alphabetical order.

#### Code Snippet 19:

```
<?php
$Student = array("Peter", "John", "David");
sort($Student);
$clength = count($Student);
for($x = 0; $x < $clength; $x++) {
    echo $Student[$x];
    echo "<br>";
}
?>
```

In Code Snippet 19, an array's built-in function `sort()` is used, which sorts student names in alphabetical order.

## Output

David  
John  
Peter

### Sort Array (Ascending Order), According to Value - `asort()`

Code Snippet 20 demonstrates an example for sorting an associative array in ascending order according to the value.

#### Code Snippet 20:

```
<?php
$age = array("Peter"=>"35", "John"=>"37", "David"=>"43");
asort($age);
foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

In the code, the `foreach` loop is used to display a person's name as key and age as value. This is where PHP's built-in function `asort()` is used, which sorts an associative array of values in ascending order.

```
Key=Peter, Value=35
Key=John, Value=37
Key=David, Value=43
```

### Sort Array (Ascending Order), According to Key - `ksort()`

Code Snippet 21 shows an example that sorts an associative array by key in ascending order.

#### Code Snippet 21:

```
<?php
$age = array("Peter"=>"35", "John"=>"37", "David"=>"43");
ksort($age);
foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

In the code, the `foreach` loop is used to display a person's name as key and age as value. This is where PHP's built-in function `ksort()` is called, which sorts an associative array of values. Here, key is sorted according to that

values of an array. Keys David, John, and Peter are sorted in alphabetical order.

## Output

```
Key=David, Value=43
Key=John, Value=37
Key=Peter, Value=35
```

### Sort Array (Descending Order), according to Value - **arsort()**

In Code Snippet 22, `arsort()` array is used to sort the associative array in descending order, according to the value.

#### Code Snippet 22:

```
<?php
$age = array("Peter"=>"35", "John"=>"37", "David"=>"43");
arsort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

In the code, the `foreach` loop is used to display a person's name as key and age as value. This is where PHP's built-in function `arsort()` is used, which sorts an associative array of values by key values of an array in descending order.

## Output

```
Key=David, Value=43
Key=John, Value=37
Key=Peter, Value=35
```

### Sort Array (Descending Order), According to Key - **krsort()**

Code Snippet 23 shows an example for sorting an associative array in descending order, according to the key.

#### Code Snippet 23:

```
<?php
$age = array("Peter"=>"35", "John"=>"52", "David"=>"43");
krsort($age);
foreach($age as $x => $x_value) {
```

```
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

In this code, `foreach` loop is used to display the name as key and the age of a person as value. Here, built-in function of PHP `arsort()` is used, which sorts an associative array of values according to the key of an array in reverse alphabetical order.

### Sort Array in Descending Order - `rsort()`

#### Output

```
Key=Peter, Value=35
Key=John, Value=52
Key=David, Value=43
```

Code Snippet 24 shows an example for sorting the elements of the `$Student` array in descending alphabetical order.

#### Code Snippet 24:

```
<?php
$Student = array("Peter", "John", "David");
rsort($Student);
$clength = count($Student);
for($x = 0; $x < $clength; $x++) {
    echo $Student[$x];
    echo "<br>";
}
?>
```

In this code, built-in function of the array is used that is `rsort()` which sorts the names of students in descending alphabetical order.

#### Output

```
Peter
John
David
```

## 5.7 Summary

- There are three types of conditional statements in PHP, namely `if`, `if...else`, and `if...elseif...else` statements.
- If a test condition can give multiple values, then the `switch` statement is used for testing multiple conditions and explaining different types of conditional statements and using them.
- Loops are used for executing the same block of code multiple times till a condition is satisfied.
- There are three types of loop statements, namely `while`, `do...while`, and `for` loops.
- `foreach` loop statement loops through a block of code for each element in an array.
- `break` statement is used to jump out of a particular code block.
- The `continue` statement skips the iteration in the loop and proceeds to the next iteration when a specified condition is satisfied.
- There are three different types of arrays, namely numeric, associative, and multi-dimensional array.
- The elements in an array can be sorted in alphabetical or numerical order and in descending or ascending manner.
- There are six functions for sorting in PHP namely, `sort()`, `rsort()`, `asort()`, `ksort()`, `arsort()`, and `krsort()`.

## 5.8 Test Your Knowledge



1. Which loop evaluates the conditional expression before processing the loop?
  - A. for loop
  - B. while loop
  - C. do-while loop
  - D. All of these
2. What will be the output of the following PHP code?

```
<?php
for ($num = 1; $num <= 10; $num += 2) {
    echo "$num ";
}
?>
```

- A. 1 3 5 7 9
- B. 1 2 3 4 5
- C. 9 7 5 3 1
- D. Error

3. What will be the output of the following PHP code?

```
<?php  
$num = 20;  
while ($num < 12) {  
    $num += 2;  
    echo $num, "\n";  
}  
?>
```

- A. Error
- B. No Output
- C. infinite loop
- D. Only one garbage value

4. Identify the output of the following PHP code:

```
<?php  
$arr = array (10, 20, 30);  
foreach ($arr as $val) {  
    echo "$val\n";  
}  
?>
```

- A. 10 20 30
- B. No Output
- C. 10
- D. undefined variable

5. Identify the output of the following PHP code:

```
<?php  
for ($a = 1; $a <= 10; $a++) {  
    for ($b = 1; $b <= 5; $b++) {  
        print "LFC";  
    }  
}  
?>
```

- A. "LFC" will be printed 10 times
- B. "LFC" will be printed 50 times
- C. "LFC" will be printed 5times
- D. "LFC" will be printed 1 time

6. Which in-built function will add a value to the end of an array?

- A. array\_unshift()
- B. into\_array()
- C. inend\_array()
- D. array\_push()

## **Answers to Test Your Knowledge**

---

1. while loop
2. 1 3 5 7 9
3. 1 2 3 4 5
4. undefined variable
5. "LFC" will be printed 1 time
6. array\_push()

## 5.9 Try It Yourself

1. Write a program in PHP using `for` loop and `continue` statement and skip alternate numbers between 1 to 25.
2. Write a program in PHP to arrange a series of given number in descending order using the sorting array concept.  
Assume that the numbers are: 46, 87, 23, 99, 24, 65
3. Write a program in PHP to arrange the names of colors in the rainbow in ascending alphabetical order.

## Session 6

# Form Handling



### Learning Objectives

*In this session, students will learn to:*

- Explain how to create a simple HTML Form
- Compare GET versus POST methods and identify their uses
- Explain \$\_REQUEST variable
- Explain different form validations
- Describe the importance of required fields in PHP forms
- Elaborate the process of validating in PHP forms

### Overview

This session describes how to create a simple HTML form, different GET and POST methods and their uses in HTML forms. Further, session details about form validation for other elements with PHP form security, required fields in a PHP form and error messages, validating names, and password.

#### 6.1 Simple HTML Form

A form is used to get data from the users and pass it to the Web server for processing. It is created using HTML tags. Different types of form controls for collecting data include Checkboxes, Radio buttons, Submit and Reset buttons, Clickable controls, and so on.

Data can be submitted to the server for processing using the following methods:



PHP POST  
method



PHP GET  
method

The GET method directs the Web browser to send the encoded user information appended at the end of the URL, to the processing agent. In this method, a question mark is added at the end of the URL. This question mark separates the URL and the form information. Each input variable in a form has a Name tag and a value that is assigned to it by the user.

The POST method directs the Web browser to send all the user information to the processing agent, through the message body of an HTTP request. This method has the capacity to transmit more information, because there is no physical limit on the amount of information passed through the body of the HTTP request.

PHP supports a concept called superglobal variables which refers to built-in variables that are always available in all scopes. `$_GET` and `$_POST` are the superglobal variables which are used to retrieve the data from submitted HTML form that have used GET or POST methods.

`$_GET` is internally represented as an array of variables passed to the current PHP script via the URL parameters and `$_POST` as an array of variables passed to current script via HTTP POST method.

Following is the syntax for GET and POST methods respectively:

```
<?php  
$_GET['variable_name'];  
?>  
  
<?php  
$_POST['variable_name'];  
?>
```

where, `$_GET` and `$_POST` are superglobals and are always accessible, irrespective of the scope. They can be accessed from any function, class, or file.

Code Snippet 1 shows a simple HTML form with two input fields and a submit button.

**Code Snippet 1:**

```

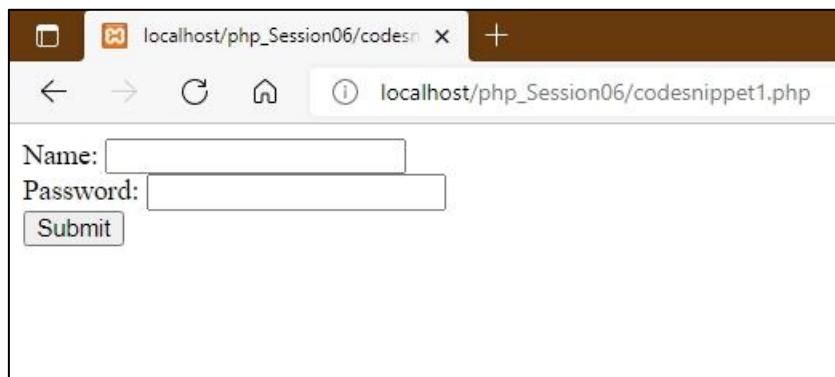
<html>
  <body>
    <form action="welcome.php" method="post">
      Name: <input type="text" name="name"><br>
      Password: <input type="password" name="password">
      <br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>

```

Code Snippet 1 shows code to render a form with two input fields for name and password respectively. When a user fills the form and clicks Submit button, the code redirects the user to another page, a PHP file named, which then displays welcome message with username and password.

The HTTP POST method is used to send the form data for processing to **welcome.php** present on the server.

Figure 6.1 shows the output of Code Snippet 1.



**Figure 6.1: Generating a Simple Form**

Code Snippet 2 shows statements to retrieve and display name and password of the user that were submitted through the form. The echo function is used to generate the output with the submitted data. These statements will be saved as **welcome.php**.

### Code Snippet 2:

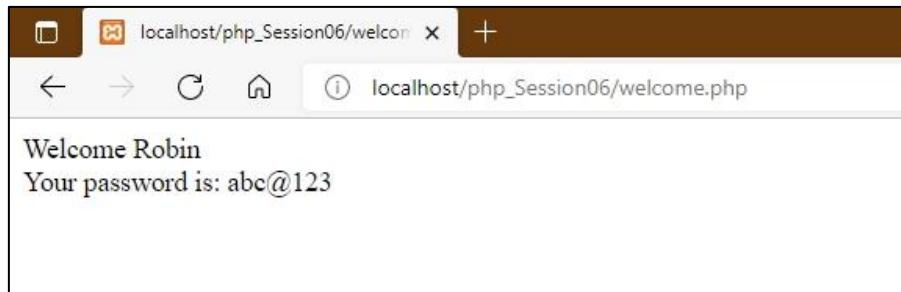
```

Welcome <?php echo $_POST["name"]; ?> <br>
Your password is: <?php echo $_POST["password"]; ?>

```

When user submits the form in Code Snippet 1 with the data, Here, `$_POST` superglobal variable is used to retrieve name and password submitted in the form.

Figure 6.2 shows the output of Code Snippet 2, based on data that is submitted when Code Snippet 1 is executed.



**Figure 6.2: Displaying Form Data**

Code Snippet 3 demonstrates use of `GET` to display the name entered by the user with a personalized greeting.

### **Code Snippet 3: (codesnippet3.php)**

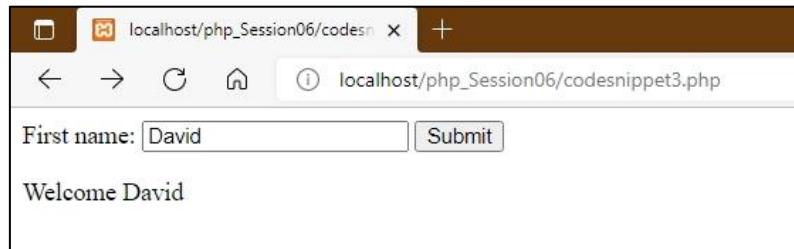
```
<html>
<body>
<form action="codesnippet3.php" method="get">
First name: <input type="text" name="firstname">
<input type="submit" value="Submit">
</form>
Welcome
<?php
// Turn off all error reporting
error_reporting(0);
echo $_GET["firstname"]
?>
</body>
</html>
```

In Code Snippet 3, the code for generating input form and generating output are written together in a single file named **codesnippet3.php**. When the code is executed, the user can enter data in the form page and click submit button. The user will then, be redirected to a refreshed version of the same page which retrieves data from the textbox using `GET` method and displays it with a personalized greeting.

When PHP encounters undefined variables such as `firstname` before the form is submitted, it will raise errors that are displayed in the browser. To turn this

OFF, you can use the `error_reporting(0);` statement as shown in Code Snippet 3.

Figures 6.3 shows the output of Code Snippet 3.



**Figure 6.3: Displaying Form Data with GET**

## 6.2 GET versus POST

---

Both `GET` and `POST` methods each create an array (Example: `array( key1 => value1, key2 => value2, key3 => value3, ... )`). The array contains key and value pairs, where, keys are names of the form controls and values are input data from the user. Each name/value pair is URL encoded by the browser, where spaces are replaced with the + sign and non-alphanumeric characters are replaced by hexadecimal values. The encoded information is then sent to the server.

### 6.2.1 Uses of GET

`GET` method displays all the values being sent in the URL itself. Hence, this method is recommended only for sending non-sensitive data. One should not use the `GET` method to send passwords or other sensitive information. This method also has a limit on the amount of information that can be sent. The limitation is about 2048 characters. However, it is possible to bookmark the page. That means, a browser can remember the page and it will be easy for you to access it next time. Unlike `POST` method, `GET` method can be bookmarked because the data required for the request is stored in the URL.

### 6.2.2 Uses of POST

`POST` method does not display values in the URL because data sent is in the package form and maintained in a separate communication processor. It has no limit on the amount of information sent because it is submitted via the HTTP's body.

Moreover, the `POST` method supports advanced functionality such as multi-part binary input while uploading files to a server. It also supports other data types such as string, numeric, and so on. However, as variables are not displayed in the URL, it is not possible to bookmark the page.

### **6.3 `$_REQUEST` Variable**

---

PHP `$_REQUEST` variable contains values of `$_GET`, `$_POST`, and `$_COOKIE`.

A cookie is a text file created on a user's computer to store user information for tracking that can be used by Web servers. `$_COOKIE` is an array of variables passed to the current script via HTTP Cookies.

PHP `$_REQUEST` variable is used to get the result from the form data sent with the `GET` and `POST` methods.

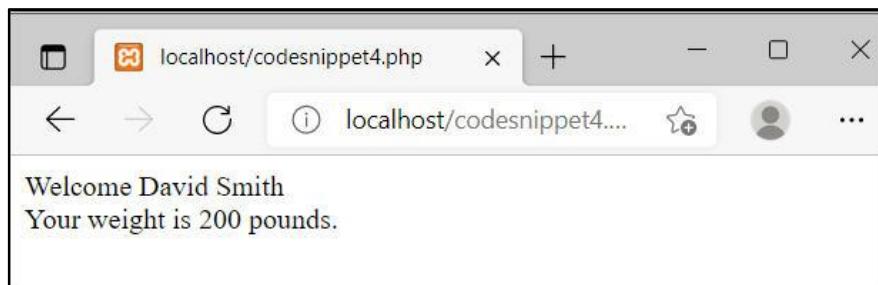
Code Snippet 4 shows the use of `$_REQUEST` variable.

#### **Code Snippet 4:**

```
<?php
    error_reporting(0);
    if( $_REQUEST["name"] || $_REQUEST["weight"] )
    {
        echo "Welcome ". $_REQUEST['name']. "<br />";
        echo "You are ". $_REQUEST['weight']. " kgs .";
        exit();
    }
?>
<html>
    <body>
        <form action=<?php $__PHP_SELF ?>" method="POST">
            Name: <input type="text" name="name" />
            Weight: <input type="text" name="weight" />
            <input type="submit" />
        </form>
    </body>
</html>
```

In this code too, the input form and display form are written in the same program. Name and weight of the user are provided as input. `$_PHP_SELF` variable is used to send the submitted form data to the same page upon button submission. `$_REQUEST` is used to return the values of user's name and weight.

Figure 6.4 shows the output of Code Snippet 4 after submission of input data through the form.



**Figure 6.4: Using `$_REQUEST`**

## 6.4 Form Fields Validation

---

Form validation is an important task done before submitting the data entered by the user in a form to the server/database. The scripts check the data entered in the fields against the validation rules already set by the developer.

In general, commonly used input fields in PHP form include:

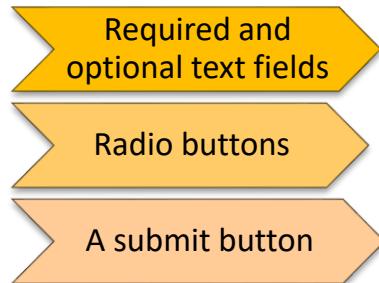


Figure 6.5 shows a sample form with validation, where mandatory fields are marked with red.

A screenshot of a web form titled "PHP Form Validation". The form includes several input fields: "UserID:" (mandatory), "Password:" (mandatory), "College:", "Department:", "Comment:" (with a rich text editor), and "Gender:" (radio buttons for Female, Male, Other). The mandatory fields ("UserID:" and "Password:") have red asterisks (\*) next to them, indicating they are required. A "Submit" button is at the bottom.

**Figure 6.5: Form Validation**

Table 6.1 shows some examples of validation rules that can be applied for a form.

Field	Validation Rules	Condition
Name	It must only contain letters and whitespace. The user must provide the input.	Mandatory
e-mail	It must contain a valid e-mail address (with '@' and '.'). The user must provide the input.	Mandatory
Website	If present, it must contain a valid URL.	Optional
Comment	Multi-line input field (textarea). If present user has to write comments in text area given.	Optional
Gender	The user must select one option.	Mandatory

**Table 6.1: Form Validation Rules**

#### 6.4.1 Text Fields

Text fields in a form can accept textual input. Name, address, occupation, and so on are common examples of text input. Text boxes and text areas are used to accept short and long text respectively.

Code Snippet 5 shows how to create a simple form that utilizes text fields for accepting Username, Password, Company, and Comment.

#### Code Snippet 5:

```
<form action="login.php" method="get">
UserID: <input type="text" name="userID"><br><br>
Password: <input type="password" name="password"><br><br>
Company: <input type="text" name="company"><br><br>
Mobile Number: <input type="text" name="Mobile
Number"><br><br>
Comments: <textarea name="comments" rows="6" cols="50">
</textarea>
</form>
```

In Code Snippet 5, a simple form is created where user can enter name, password, company, and comments. User enters his/her details in the textbox given for each field. The comment field contains more text area compared to other textboxes of given fields, as the comments can be the details of the product or the feedback.

Figure 6.6 shows the form with text fields.

The screenshot shows a web browser window with the URL `localhost/code6_5.php`. The page contains a form with the following fields:

- User ID: [Text Input Field]
- Password: [Text Input Field]
- Company: [Text Input Field]
- Mobile Number: [Text Input Field]
- Comments: [Text Area]

**Figure 6.6: Using Text Fields in a Form**

#### 6.4.2 Radio Buttons

Radio buttons are used in the forms when a user has multiple choices given and has to choose one given option.

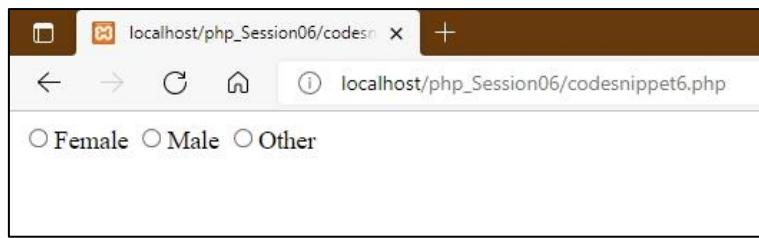
Code Snippet 6 creates gender fields in a form to be selected using radio buttons.

#### Code Snippet 6:

```
<form action="form1.php" method="get">
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
</form>
```

The code displays the radio buttons asking user to select the appropriate gender. User can select male, female, or other option.

Figure 6.7 shows the form that uses radio buttons.



**Figure 6.7: Using Radio Buttons in a Form**

In a practical scenario, the code in snippets 5 and 6 can be combined with more statements if necessary to create a full and proper form.

#### **6.4.3 Using `htmlspecialchars()` with Form Elements**

Form elements are the elements which are used inside the `form` tag. In a form, it is recommended that methods such as `GET` or `POST`, variables such as `$_SERVER["PHP_SELF"]`, and `htmlspecialchars()` function be used.

For example, general syntax for form elements can be as follows:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

where,

`htmlspecialchars()` is a function used to convert special characters into HTML entities. It replaces HTML characters such as < and > with &lt; and &gt;. This prevents attackers from manipulating code by inserting HTML or JavaScript code (Cross-site Scripting attacks) in forms.

On submitting, the form data is sent via method `POST`.

#### **6.4.4 Validating Form Data with PHP**

As a first step in validation, one should pass all variables through the PHP's `htmlspecialchars()` function to validate form data.

Consider a scenario where a malicious user tries to submit a hyperlink within a text field that is meant to receive user name or some such text. When the hyperlink is posted to the redirected page, which can lead to a harmful action.

When `htmlspecialchars()` function is used, if a user submits data such as the following in a text field, it will not be executed as a hyperlink:

```
<script> location.href('http://www.attack.com')</script>
```

This is because the `htmlspecialchars()` function caused the data to be saved internally as HTML escaped code:

```
&lt;script&gt;location.href('http://www.attack.com')&lt;/script&gt;
```

Thus, this code can now be displayed safely on a page or within an e-mail because it is treated as text instead of a hyperlink. On the other hand, if `htmlspecialchars()` function had not been used with the text field, the outcome would have been unsafe and potentially dangerous.

Two more actions can be done when the user submits the form:

- Remove (with the use of PHP `trim()` function) unnecessary characters (extra space, tab, and newline) from the user input data
- Remove (with the use of PHP `stripslashes()` function) backslashes (\) from the user input data

Next step is to create a function that will perform the data sanitizing process, which is more convenient than writing same code repeatedly.

Code Snippet 7 shows a program to check each `$_POST` variable with a `sanitize_data()` function.

### Code Snippet 7:

```
<!DOCTYPE html>
<head>
</head>
<body>
<?php
// Initialize variables during creation to empty values
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = sanitize_data($_POST["name"]);
    $password = sanitize_data($_POST["password"]);
    $company = sanitize_data($_POST["company"]);
    $comment = sanitize_data($_POST["comment"]);
    $gender = sanitize_data($_POST["gender"]);
    echo "Data is sanitized in appropriate format <br>";
}

function sanitize_data($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data, ENT_QUOTES);
```

```

    return $data;
}
?>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
Name: <input type="text" name="name">
<br><br>
Password: <input type="password" name="password">
<br><br>
Company: <input type="text" name="company">
<br><br>
Comment: <textarea name="comment" rows="5"
cols="40"></textarea>
<br><br>
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
<br><br>
<input type="submit" name="submit" value="Submit">
</form>
<?php
error_reporting(0);
echo $name;
echo "<br>";
echo $password;
echo "<br>";
echo $company;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>

```

Figure 6.8 shows the form with data before clicking Submit.

The screenshot shows a web browser window with the URL `localhost/code6_7.php`. The page displays a form with the following fields:

- Name:
- Password:
- Company:
- Comment:
- Gender:  Female  Male  Other
- Submit button

**Figure 6.8: Using a Function to Validate Data in a Form**

Here, the company data and comment data have some special characters such as back slashes and quotes.

Figure 6.9 shows the output upon clicking Submit. You will observe that the data displayed still shows single and double quotes in the output. This is because even after `htmlspecialchars` has performed the encoding, the browser's internal engine decodes the output before displaying. However, the data that is submitted to the server will be sent in encoded form as is. This can be verified in Figure 6.10 which is the Page source view of the .php file.

Data is sanitized in appropriate format

Name:

Password:

Company:

Comment:

Gender:  Female  Male  Other

David Smith  
d123  
AlkalinezLimitedInc  
'Chemicals' and "Pharmaceuticals" are our lifeline.  
male

**Figure 6.9: Output After Clicking Submit**

```

<!DOCTYPE html>
<head>
</head>
<body>
Data is sanitized in appropriate format <br><form method="post"
action="/code6_7.php">
Name: <input type="text" name="name">
<br><br>
Password: <input type="password" name="password">
<br><br>
Company: <input type="text" name="company">
<br><br>
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
<br><br>
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
<br><br>
<input type="submit" name="submit" value="Submit">
</form>
David Smith<br>d123<br>AlkalinezLimitedInc<br>'Chemicals' and "Pharmaceuticals" are our lifeline.<br>male</body>
</html>

```

**Figure 6.10: Page Source Showing Encoded Data**

This form currently does not have any mandatory fields hence, even if user submits empty data or blank fields, there will not be any issues.

To overcome this, further validation actions are required.

Forms can contain fields which are mandatory for the users to enter. These fields are collectively called as required fields. New variables \$nameErr, \$pwdErr, \$genderErr, and \$companyErr are added to the earlier code as shown in Code Snippet 8. These variables hold error messages for required fields in the form created in Code Snippet 7. if else statements are also added for each \$\_POST variable, which check whether the \$\_POST variable is empty (with the PHP empty() function). If the \$\_POST variable is empty, an error message is stored in the different error variables and if not empty, it sends the user input data through the sanitize\_data() function.

Code Snippet 8 shows how error variables store values.

### Code Snippet 8:

```
<?php
// define variables and set to empty values
$nameErr = $pwdErr = $genderErr = $companyErr = "";
$name = $password = $gender = $comment = $company = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = sanitize_data($_POST["name"]);
    }

    if (empty($_POST["password"])) {
        $pwdErr = "Password is required";
    } else {
        $password = sanitize_data($_POST["password"]);
    }

    if (empty($_POST["company"])) {
        $company = "";
    } else {
        $company = sanitize_data($_POST["company"]);
    }

    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = sanitize_data($_POST["comment"]);
    }
}
```

```

if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
} else {
    $gender = sanitize_data($_POST["gender"]);
}
?>

```

#### 6.4.5 Display Error Messages

Additional code is written after each required field in the form to generate the correct error message, if required. For example, an error message is displayed if the user tries to submit the form without filling required fields.

Code Snippet 9 shows a program to display an error message when the user submits the form without filling the required fields.

#### Code Snippet 9:

```

<html>
<head>
<style>
.error {color: #FF0000; }
</style>
</head>
<body>
<?php
error_reporting(0);
// Initialize variables during creation to empty values
$nameErr = $pwdErr = $genderErr = $companyErr = "";
$name = $pwd = $gender = $comment = $company = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = sanitize_data($_POST["name"]);
    }
    if (empty($_POST["password"])) {
        $pwdErr = "password is required";
    } else {
        $password = sanitize_data($_POST["password"]);
    }
    if (empty($_POST["company"])) {
        $company = "";
    } else {
        $company = sanitize_data($_POST["company"]);
    }
}

```

```

if (!empty($_POST["comment"])) {
    $comment = "";
} else {
    $comment = sanitize_data($_POST["comment"]);
}
if (!empty($_POST["gender"])) {
    $genderErr = "Gender is required";
} else {
    $gender = sanitize_data($_POST["gender"]);
}
}

function sanitize_data($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<p><span class="error">* indicates required field</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
    Name: <input type="text" name="name">
    <span class="error">* <?php echo $nameErr; ?></span>
    <br><br>
    Password: <input type="password" name="password">
    <span class="error">* <?php echo $pwdErr; ?></span>
    <br><br>
    Company: <input type="text" name="company">
    <span class="error"><?php echo $companyErr; ?></span>
    <br><br>
    Comment: <textarea name="comment" rows="5"
cols="40"></textarea>
    <br><br>
    Gender:
    <input type="radio" name="gender" value="female">Female
    <input type="radio" name="gender" value="male">Male
    <input type="radio" name="gender" value="other">Other
    <span class="error">* <?php echo $genderErr; ?></span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
</form>
<?php
echo $name;
echo "<br>";
echo $password;
echo "<br>";
echo $company;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;

```

```
?>  
</body>  
</html>
```

Figures 6.11 and 6.12 shows output of Code Snippet 9 before and after clicking Submit respectively. As you can observe, empty fields will not be accepted.

The screenshot shows a web browser window with the URL `localhost/code6_9.php`. The page contains a form with the following fields:

- Name:
- Password:
- Company:
- Comment:
- Gender:  Female  Male  Other \*
- Submit button

A red note at the top left says: \* indicates required field.

*Figure 6.11: Page Source Showing Encoded Data*

The screenshot shows a web browser window with the URL `localhost/code6_9.php`. The page contains a form with the following fields:

- Name:  \* Name is required
- Password:  \* password is required
- Company:
- Comment:
- Gender:  Female  Male  Other \* Gender is required
- Submit button

A red note at the top left says: \* indicates required field.

*Figure 6.12: Page Source Showing Encoded Data*

#### 6.4.6 Validating Against Special Characters

In a text field such as name, user can only enter letters and whitespaces. Numbers and special characters are not allowed in the name field.

Following statements check the name field for letters, dashes, apostrophes, and whitespaces:

```
$name = sanitize_data($_POST["name"]);
if (!preg_match("/^ [a-zA-Z- ]* $/", $name)) {
    $nameErr = "Only letters and white space allowed";
}
```

An error message is displayed if the value entered for the name field is not valid. The `preg_match()` function is used here to search a string for a pattern. It returns **true** if the pattern exists and **false** if the pattern does not exist.

The validation of password contains different criteria such as password entered by the user must be at least eight characters, with one uppercase character, one lower case character, one number, and one special character.

If all the criteria or conditions match, then user password is strong, otherwise a message displays saying the password is weak. Form will display a message asking user to enter a password with at least eight characters, one uppercase, one lower case character, one number and one special character.

Code Snippet 10 shows how to check if a password entered by the user is valid or not. An error message is displayed if the password entered is not valid.

#### Code Snippet 10:

```
$password = $_POST['password'];
if( strlen($password) < 8 ) {
    $pwdErr .= " Password too short. ";
}
if( strlen($password) > 20 ) {
    $pwdErr .= " Password too long!";
}
if( !preg_match("@[0-9]@", $password) ) {
    $pwdErr .= " Password must include at least one number.";
}
if( !preg_match("#[a-z]+#", $password) ) {
    $pwdErr .= " Password must include at least one letter.";
}
if( !preg_match("#[A-Z]+#", $password) ) {
```

```

$pwdErr .= " Password must include at least one uppercase
letter.";
}
if( !preg_match("#[^\\w]#", $password) ) {
$pwdErr .= " Password must include at least one symbol.";
}
if (empty($pwdErr)) {
if (preg_match("#.*^(?=.{8,20})(?=.*[a-z])(?=.*[A-Z])(?=.*[0-
9])(?=.*[^\\w]).*$#", $password)) {
echo "Your password is strong.";
} else {
echo "Your password is not safe.";
}
}

```

In this code, password entered by the user is validated. At first, `if` condition of the code checks the length of password that is if it is smaller than eight characters then, it will display error message saying the password is too short.

The next `if` condition checks if the password entered is bigger than 20 characters. If yes, then, it will display an error message saying the password is too long. Remaining `if` conditions in the code checks whether the password contains at least one number, one lowercase, one uppercase, and at least one special character. If all these conditions match in `preg_match` function, then the code displays "Your password is strong".

This code can be embedded in Code Snippet 9 to validate form password.

## 6.5 Summary

- Forms can be created using HTML tags and input elements to accept user data and send to a Web server for processing.
- GET and POST methods in PHP can be used in forms to send form data.
- \$\_GET and \$\_POST are superglobal variables and are always accessible, regardless of function, class, or file scope.
- The PHP \$\_REQUEST variable contains values of \$\_GET, \$\_POST, and \$\_COOKIE and is used to get result from form data sent with the GET and POST methods.
- The PHP form input fields have validation rules.
- PHP's htmlspecialchars() function can help to encode special characters into HTML entities and can be used to sanitize form data.
- An error message can be generated at the form level or field level through code if user tries to submit a form without filling mandatory fields.
- The preg\_match() function searches a string for a pattern, returning true if the pattern exists and false otherwise. It is used to validate the name entered.

## 6.6 Test Your Knowledge



1. The \_\_\_\_\_ specifies the URL that will process form data and send the feedback.
  - a) Method
  - b) Form control
  - c) Action attribute
  - d) Body of the HTTP message
2. Which of the following option(s) is/are the key difference between GET and POST methods?
  - a) GET is used with HTTP protocol and POST is used with HTTPS.
  - b) GET displays the submitted data as part of the URL. However, while using POST, this information is not shown.
  - c) GET is intended for changing server state and it carries more data than POST.
  - d) GET is more secure than POST and should be used for sensitive information.
3. Which of the following option(s) is/are used to enter text in PHP Forms?
  - a) input type= "text"
  - b) input type="radio"
  - c) input type="textarea"
  - d) inputtype="null"
4. Which of the following option(s) is/are correct if a user tries to submit a form without entering the text area or required field?
  - a) No error message will be displayed.
  - b) An error message will be displayed and the form will not get submitted.
  - c) An error message will display and the form is submitted.
  - d) The form is frozen.

5. Which of the following is the syntax for the GET method?
- a) \$varname = GET['variable']
  - b) varname = \$\_GET['variable']
  - c) \$varname = \$\_GET['variable']
  - d) \$varname = \$GET['variable']

## **Answers to Test Your Knowledge**

---

1. Action attribute
2. GET displays the submitted data as part of the URL. However, while using POST, this information is not shown
3. input type= "text"
4. An error message will be displayed and the form will not get submitted.
5. \$varname = \$\_GET['variable']

## **6.7 Try It Yourself**

1. Write a program in which GET and POST methods are used.
2. Write a program to create a form to accept student registration data (only for new students who have not yet enrolled) and display appropriate error messages if the user does not enter any required fields and tries to submit the form.
3. Write a program to validate 'E-mail', 'Name', and 'Password' of the form created in previous question.

## Session 7

# Working with Functions in PHP



### Learning Objectives

*In this session, students will learn to:*

- Describe PHP built-in functions
- Define PHP user-defined functions and explain how to create a user-defined function in PHP
- Describe PHP function arguments
- Identify the purpose of PHP default argument values
- Elaborate on return values in PHP functions
- Describe return type declarations
- Outline how to pass arguments by reference
- Identify the use of named arguments
- Define dynamic function calls - `date()` and `time()`

The session begins by describing built-in functions in PHP. The session also provides an overview of user-defined functions and explains how to create them. The session further covers PHP function arguments, default argument value, returning values, and return type declarations. It explains how to pass arguments by reference and describes named arguments, a new feature introduced in PHP 8. Further, the session outlines the dynamic function calls - `date()` and `time()`.

### **7.1 Functions in PHP**

A PHP function is a self-contained program segment that carries out a specific, well-defined task and can be invoked by users, multiple times as per their requirement.

Following are a few benefits of using functions in PHP:

Reusable code

PHP functions are reusable and can be invoked many times within the same program.

Minimal code

Using functions gives the flexibility to write code once and reuse the same whenever required. This reduces the amount of written code and also reduces time taken to write code.

Clarity of code

PHP functions segregate the programming logic. This allows the user to comprehend application flow as code is split into different functions.

PHP functions can be categorized as built-in functions and user-defined functions.

## 7.2 PHP Built-in Functions

Built-in functions are readymade functions provided by PHP. Some are core functions which are automatically available whereas, some require additional extensions in order to use them.

PHP has over 1000 built-in functions that help users to complete common tasks. This makes it easy for users to execute and rerun common tasks in a program.

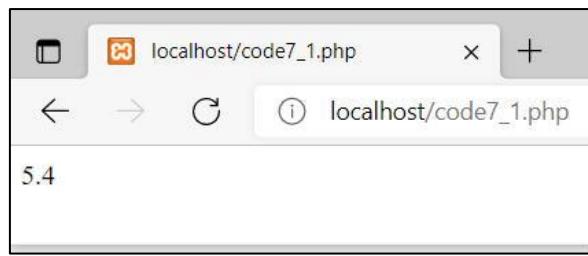
Code Snippet 1 shows how to use a built-in function, `abs()`.

### Code Snippet 1:

```
<?php  
echo abs(-5.4);  
?>
```

The built-in function `abs()` is used to calculate absolute (positive) value of any number passed to the function. Here, in the code, the value `-5.4` is passed to the function which then, returns `5.4` as the outcome. `echo` is used to display the result.

Figure 7.1 shows the output of Code Snippet 1.



**Figure 7.1: Using abs() Built-in Function**

Let us now understand about some built-in functions used for obtaining information about variables.

#### **gettype()**

A user can input a variable as an argument to this function and the output is a string value denoting the data type of the argument. Code Snippet 2 shows an example for `gettype()` function.

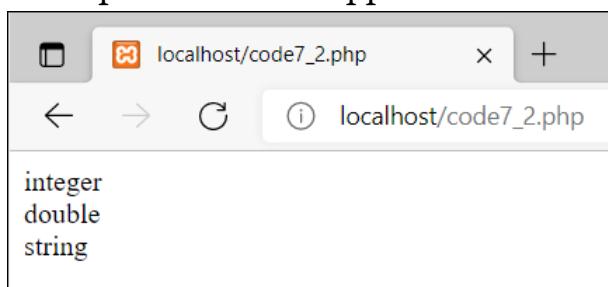
#### **Code Snippet 2:**

```
<?php
// PHP program to illustrate gettype() function
$var1 = 3; // integer value
$var2 = 5.6; // double value
$var3 = "Abc3462"; // string value
echo gettype($var1) . "<br>";
echo gettype($var2) . "<br>";
echo gettype($var3) . "<br>";
?>
```

Here, note that the definition for the `gettype()` function is not written by the user, instead it is built into PHP. As the function is already built in, users can just call it anytime within their PHP code, any number of times.

Code Snippet 2 retrieves and displays the type of data assigned implicitly to each variable namely, `var1`, `var2`, and `var3`. Based on the data assigned, `var1` is an integer, `var2` is a double, and `var3` is a string. Therefore, the output displayed is: Integer, Double, and String.

Figure 7.2 shows the output of Code Snippet 2.



**Figure 7.2: Using gettype() Built-in Function**

While the function `gettype()` returns the type of the variable, the function `var_dump()` gives complete details about the variable.

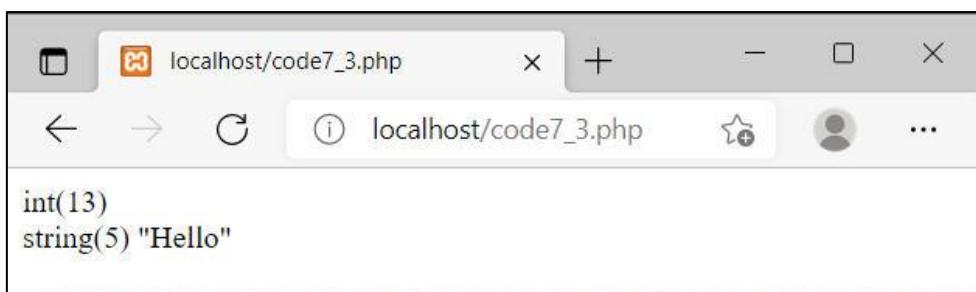
### **`var_dump()`**

This function also accepts a variable name as an argument and output is the details about the argument passed to it. Code Snippet 3 shows an example for `var_dump()` function.

### **Code Snippet 3:**

```
<?php  
$var1=13;  
$var2="Hello";  
var_dump($var1);  
echo "<br>";  
var_dump($var2);  
?>
```

In the code, the statement `var_dump($var1);` displays details about the variable `$var1`. First, the type and the variable are displayed `int(13)`. Next, `var_dump($var2);` displays details about the variable `$var2`. The type and the length are displayed along with the string value stored in `var2`. Figure 7.3 depicts the output.



**Figure 7.3: Using `var_dump()` Built-in Function**

## **PHP Array Functions**

In PHP, arrays are a part of built-in functions. An array can be generated using `array()`. This function allows users to work with arrays in multiple ways. Arrays help users to store, manage, and operate on sets of variables.

Following is the syntax for arrays:

### **Syntax**

```
array();
```

Or

```
array("value 1", "value 2", ...);
```

Code Snippet 4 shows how to use the `array` function.

#### Code Snippet 4:

```
<?php  
$names=array("David","Charlie","George","Peter");  
echo "Names are: $names[0], $names[1], $names[2] and  
$names[3]";  
?>
```

This code displays the names of a person using the `array` function. This function stores the values according to different memory positions. That is, the first value is stored in the zero place, the second value in the first place, and so on. The variable `names` is assigned to the function `array` and the values of the array are printed utilizing `echo`. For example `$names[0]=David`, `$names[1]=Charlie`, and so on. Figure 7.4 depicts the corresponding output.



**Figure 7.4: Generating an Array Using `array()` Function**

PHP supports both simple arrays and multidimensional arrays. They can be created either by the users or by another function.

Following are the three types of arrays that can be created:

Indexed  
arrays

Associative  
arrays

Multidimensional  
arrays

#### Keys and Values

The named variables in an array for which users can assign values are called keys. For example, `$age = array("Sam"=>"36", "John"=>"37", "Peter"=>"43")`; In the example, an array is assigned to the variable `age` and the array has number of other variables. Sam, John, and Peter are the

keys and 36, 37, and 43 are the values, respectively.

Table 7.1 shows some of the functions that can be used with arrays. They are part of PHP's core functions.

Function	Description
<code>array_change_key_case(array \$array, int \$case)</code>	This function switches all the keys in an array to lowercase or uppercase.
<code>array_chunk(array \$array, int \$length, bool \$preserve_keys)</code>	This function breaks an array into smaller chunks of arrays.
<code>array_column(array \$array, int string null \$column_key, int string null \$index_key)</code>	This function displays values from the column mentioned.
<code>array_combine(array \$keys, array \$values)</code>	This function generates an array utilizing elements from keys array and values array.
<code>array_count_values(array \$array)</code>	This function counts all the values of an array.
<code>array_diff(array \$array, array ...\$arrays)</code>	This function displays differences after comparing the arrays. It compares only values.
<code>array_diff_assoc(array \$keys, array \$values)</code>	This function also displays differences, but compares both the keys and values.
<code>array_diff_key(array \$array, array ...\$arrays)</code>	This function also displays differences, but compares only the keys.
<code>array_fill(int \$start_index, int \$count, mixed \$value))</code>	This function helps in inserting values to an array.
<code>array_filter(array \$array, ?callable \$callback, int \$mode))</code>	This function helps in filtering the array values utilizing a callback function.
<code>array_flip(array \$array)</code>	This function interchanges all the keys with their corresponding values in an array
<code>array_intersect(array \$array, array ...\$arrays)</code>	This function compares the values in arrays and displays the matches.
<code>array_map(?callable \$callback, array \$array, array ...\$arrays)</code>	This function transfers each value of an array to a user-made function, which in turn displays

Function	Description
	new values.
array_key_exists(string int \$key, array \$array)	This function checks the array for the mentioned key.
array_keys(array \$array)	This function displays all the keys of an array.
array_merge(array ...\$arrays)	This function combines one or more arrays into a single array.

*Table 7.1: Array Functions*

## PHP string Functions

The `string` functions are also a core part of PHP. They help the user to perform different operations on strings. To perform common string-related tasks, user can utilize various built-in `string` functions saved in the PHP installation package. These functions can be utilized without having to install anything new.

### `strrev()` function

This function accepts a string as the argument and displays the original string in reverse order. For example, `echo strrev("Hello, World!");`. This prints `!dlroW ,olleH`.

Users do not require to save values displayed by the functions into variables. Instead, they can use such values directly. In the example, `echo` is employed to display the value obtained by executing the `strrev()` function, for which `"Hello, World!"` is the argument.

### `strtolower()` function

PHP also offers a built-in function to alter the case of a `string`. `strtolower()` function helps users to change an argument string into all lowercase letters. For example, `echo strtolower("HELLO");` This prints `hello`.

### `str_repeat()` function

The function takes two arguments - first is a string and the second is a number. It outputs the argument string iterating it by the count mentioned in the second argument. For example, `echo str_repeat("hi", 10);`. This prints `hihihihihihihihihihihihi`.

The two arguments passed to the function `str_repeat()` are `hi` and `10`. So,

it displays the string `hi`, 10 times—`hihihihihihihihihihihihi`.

Table 7.2 shows some of the essential string related functions available in PHP.

<b>Function</b>	<b>Description</b>
<code>sprintf()</code>	This function writes a formatted string to a variable.
<code>sscanf()</code>	This function parses input from a string as per a format.
<code>strcasecmp()</code>	This function performs a comparison of the two given strings and it is case-insensitive.
<code>strchr()</code>	This function determines the first instance of a string within another string.
<code>strcmp()</code>	This function performs a comparison of the two given strings and it is case-sensitive.
<code>strcoll()</code>	This function performs a comparison of the two given strings and it is case-insensitive.
<code>strpos()</code>	This function displays the position of the first instance of a string within another string. It is case-insensitive.
<code>stristr()</code>	This function displays the first instance of a string within another string. It is case-insensitive.
<code>strlen()</code>	This function displays the length of a string.
<code>strpbrk()</code>	This function looks for a given set of characters in a string.
<code>strpos()</code>	This function displays the position of the first instance of a string within another string. It is case-sensitive.
<code>strrev()</code>	This function displays a given string in the reverse order.
<code>strripos()</code>	This function displays the position of the last instance of a string within another string. It is case-insensitive.
<code>strrpos()</code>	This function displays the position of the last instance of a string within another string. It is case-sensitive.
<code>strspn()</code>	This function displays the number of characters present in a string having only characters from a particular charlist.
<code>substr_count()</code>	This function displays the total number of times a substring appears in a string.
<code>wordwrap()</code>	This function wraps a string to a specified number of characters.
<code>vsprintf()</code>	This function writes a formatted string to a variable.

**Table 7.2: String Functions**

## PHP `stream` Functions

A `stream` is an easy way to access data from files, networks, or other sources without consuming unwanted memory. It allows users to access many types of data using a common set of functions and tools and builds an uniform interface. It can be read from or written to in a linear fashion and can find the location within the stream.

Additional code that guides the stream about how to manage particular protocols or encodings is called a wrapper. The `stream` functions are also a core part of the PHP. A user can begin to use these functions without having to install anything new.

Table 7.3 shows some of the key `stream` related functions.

Function	Description
<code>stream_bucket_append()</code>	This function joins the bucket to the brigade.
<code>stream_bucket_make_writeable()</code>	This function displays a bucket object from the brigade to further work on.
<code>stream_socket_server()</code>	This function generates an Internet or Unix domain server socket.
<code>stream_supports_lock()</code>	This function informs if the stream has locking or not.
<code>stream_wrapper_register()</code>	This function records a URL wrapper set up as a PHP class.
<code>stream_socket_shutdown()</code>	This function stops a full-duplex connection.
<code>stream_wrapper_restore()</code>	This function helps to restore a built-in wrapper that was previously not registered.
<code>stream_wrapper_unregister()</code>	This function helps to unregister a URL wrapper.
<code>stream_copy_to_stream()</code>	This function copies information from one stream to another stream.
<code>stream_is_local()</code>	This function inspects whether a stream is a local stream or not.

*Table 7.3: stream Functions*

Code Snippet 5 shows an example of how `stream` functions can be utilized.

The objective of the code is to check whether given streams can be opened or not and set the timeout period when the request to open the streams fails.

### Code Snippet 5:

```
<?php
$fp = fsockopen("www.education.com", 80);
if (!$fp) {
    echo "Unable to open\n";
}
else {
    fwrite($fp, "GET / HTTP/1.0\r\n\r\n");
    stream_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    $info = stream_get_meta_data($fp);
    fclose($fp);
    if ($info['timed_out']) {
        echo 'Connection timed out!';
    }
    else {
        echo $res;
    }
}
var_dump(stream_is_local("http://education.com"));
echo "<br>";
var_dump(stream_is_local("/etc"));
?>
```

This code sets the timeout value on `stream` expressed in terms of seconds and microseconds. When the function `stream_set_timeout` cannot open the intended file, it displays `Unable to open`. The function `var_dump` returns the type of variable stored, which is Boolean type. Then, the function `stream_is_local` checks whether a stream or a URL is a local one or not. If it is local, the function returns `true`, if not, it returns `false`.

Figure 7.5 depicts the corresponding output.



**Figure 7.5: Using stream Functions**

## 7.3 PHP User-defined Functions

---

While PHP has numerous built-in functions, it also gives users flexibility to create their own functions. Unlike built-in functions which are readily available, user-defined functions are created by users.

User-defined functions can be defined as those functions that are generated by the users depending on their requirements to complete a particular task. However, there may arise situations wherein none of the built-in functions can serve the purpose. In such cases, with user-defined functions, the user can create functions according to the task to be performed.

### Create a User-defined Function in PHP

Some of the rules that users must adhere to while creating a user-defined function are as follows:

-  The name of a function must include only alphabets, numbers, and underscores. Special characters cannot be a part of the function name.
-  A function name must begin with either an alphabet or an underscore. A number cannot be used at the beginning of the name.
-  Function names are case-insensitive.
-  After the function name, use the opening curly brace { to indicate the beginning of the function code and end it with the closing curly brace }.

Following is the syntax to create a user-defined function:

#### Syntax:

```
function functionName() {  
    <code block comprising one or more statements>  
}
```

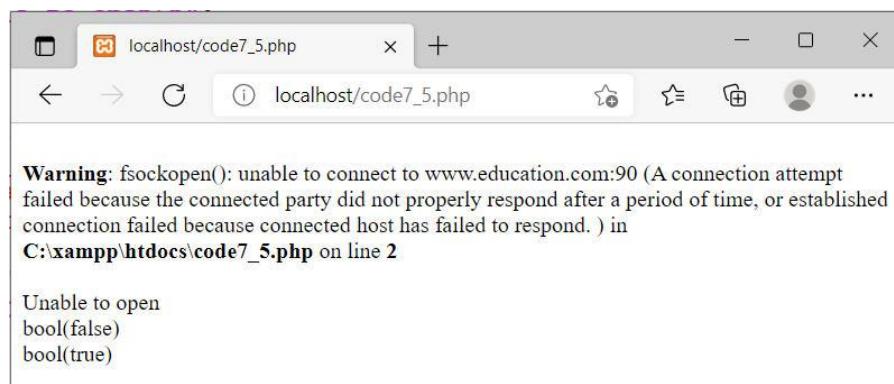
Code Snippet 6 shows how to display a message using a user-defined function.

## Code Snippet 6:

```
<?php
function even_number()
{
    for( $i=0; $i<=10; $i++ )
    {
        if( $i%2 == 0 ){
            echo "<br>", $i;
        }
    }
even_number();
?>
```

This code displays even numbers using `for` loop and `if` statements. `for` loop assigns variable `i` to 0 and then, increments it till 10 (`i<=10`). The `if` condition checks whether `i` is divisible by 2 or not. If yes, it displays `i`, otherwise it does not display any number. Note that `%` is the modulus operator that returns the remainder.

Figure 7.6 depicts the output of the code demonstrating a user-defined function.



*Figure 7.6: User-defined Function*

## 7.4 PHP Function Arguments and Parameters

For a function, users can write one or more parameters to which they can pass arguments during a function call. Arguments can be used to feed data to the functions. An argument is similar to a variable. To pass an argument to a function, users can specify it within the parentheses soon after the function name. Users can pass as many parameters as required by separating each one of them with a comma.

Code Snippet 7 shows how to display the names of people by passing one argument to a function.

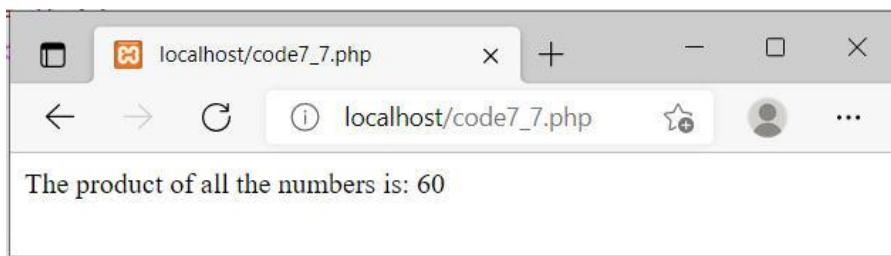
### Code Snippet 7:

```
<?php
function numbers($num1, $num2, $num3)
{
    $product = $num1 * $num2 * $num3;
    echo "The product of all the numbers is: $product";
}

// Calling the function
// Passing three arguments
numbers(4, 3, 5);
?>
```

In the code, the name of the function is `numbers()` and it has three parameters `num1`, `num2`, and `num3`. Now, when the `numbers()` function is called, all the three arguments are passed to the function parameters. Therefore, the program enters into a function body, calculates the product of all the three numbers 4, 3, and 5 and displays the output as 60.

Figure 7.7 depicts the output of the code.



*Figure 7.7: Function Arguments and Parameters*

## 7.5 PHP Default Argument Value

---

Function arguments play a vital role in executing the function code. There are instances when a user might not remember to pass the argument while calling the function, which expects arguments. This may lead to unexpected results. Therefore, to avoid such instances, a default value is assigned to the arguments. Such a value is called default argument value. This value will be utilized if no value is entered for the argument when the function is executed.

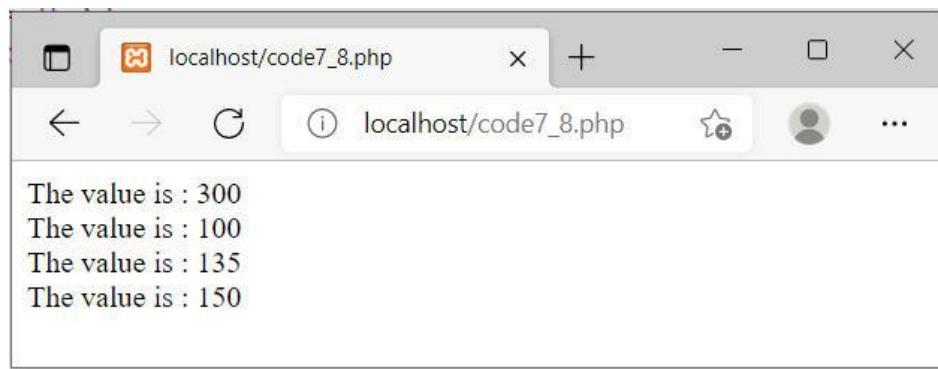
Code Snippet 8 shows how to use a default argument.

### Code Snippet 8:

```
<?php declare(strict_types=1); // strict requirement ?>
<body>
<?php
function value(int $Val_default = 100) {
    echo "The value is : $Val_default <br>";
}
value(300);
value();
value(135);
value(150);
?>
```

In the code, when the function `value()` is called, the numbers are passed to the function during function call and the respective numbers are displayed. However, when the function is called without an argument, it considers the default value `100` as the argument.

Figure 7.8 depicts the output of the code.



**Figure 7.8: Default Arguments**

## 7.6 PHP Functions - Returning Values

An important aspect of PHP functions is that they can also return results. For example, if there is a function defined to perform certain mathematical calculations, users might prefer to return the result of the calculations to the calling program. In such instances, when the function is called, the outcome of the function can be obtained in the calling statement. .

To do this, users can utilize the `return` statement within the function body as the last statement of the function. It helps in returning any variable or value from a function. Note that only one variable/value can be returned. The

`return` keyword cannot be followed by multiple values.

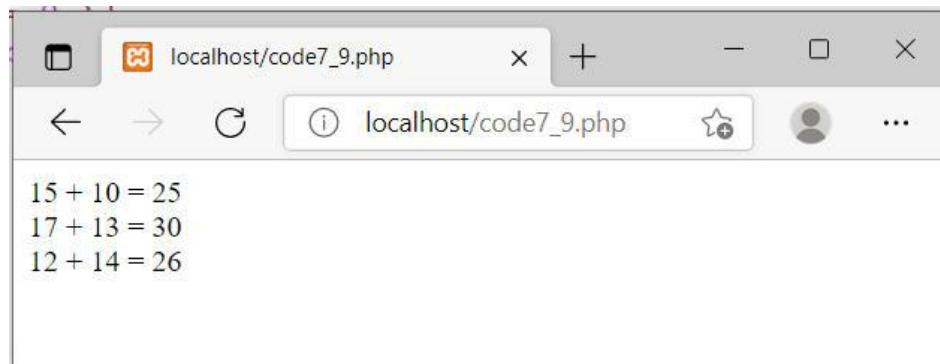
Code Snippet 9 shows how to return a value to a function using the `return` statement.

### Code Snippet 9:

```
<?php
function Add_Numbers(int $a, int $b) {
    $c = $a + $b;
    return $c;
}
echo "15 + 10 = " . Add_Numbers(15,10) . "<br>";
echo "17 + 13 = " . Add_Numbers(17,13) . "<br>";
echo "12 + 14 = " . Add_Numbers(12,14);
?>
```

The code shows how to add two given numbers and store the sum in variable `c`. The function named `Add_Numbers` is defined to accept two parameters namely, `a` and `b` and then, calculates the sum of variables `a` and `b`. The function returns the value of `c` to the calling program. In the calling program, the statements calling the function are embedded within echo statements.

Figure 7.9 depicts the output of the code.



*Figure 7.9: Returning Values*

## 7.7 PHP Return Type Declarations

PHP 8 offers a new feature called type declarations for `return` statements. Similar to type declaration for function arguments, if strict requirement is enabled, the code displays a 'Fatal Error' in case of a type mismatch.

In order to do a type declaration for a function return, while creating the function, users must insert a colon (`:`) and `type` before the opening curly (`{`)

bracket.

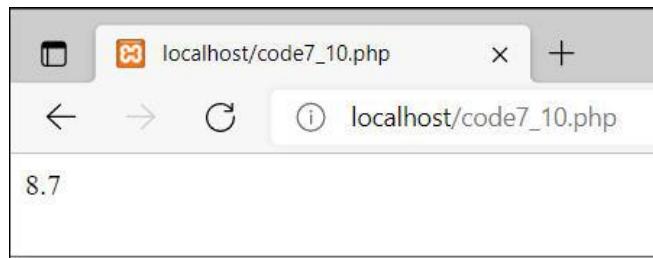
Code Snippet 10 shows an example of how to specify return type for a function.

#### **Code Snippet 10:**

```
<?php declare(strict_types=1); // strict requirement
function AddNumbers(float $n1, float $n2) : float {
    return $n1 + $n2;
}
echo AddNumbers(1.5, 7.2);
?>
```

In the code, a user-defined function `AddNumbers` is declared. It adds two floating point numbers saved in variables `n1` and `n2` respectively. It then returns the sum of the numbers utilizing the keyword `return`. The function `AddNumbers` is called and passed two floating point values. The return value is displayed using `echo`. All these actions seem similar to actions being performed in general in earlier programs. The difference here is that the return type is limited to a floating point value through the use of colon and `float` keyword.

Therefore, the output is sum of 1.5 and 7.2, which is 8.7, as shown in Figure 7.10.



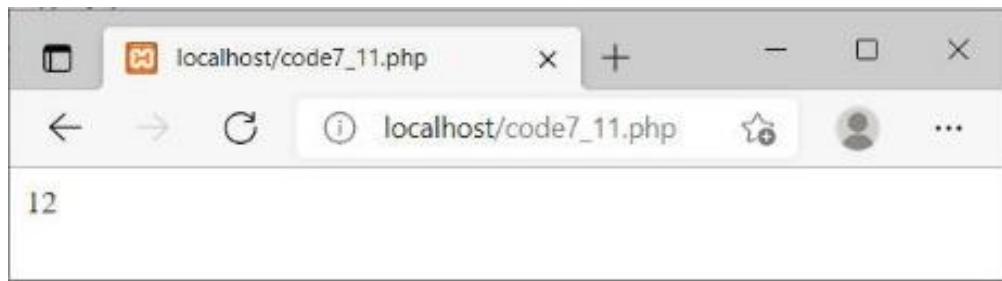
**Figure 7.10: Using Type Declarations in Return**

Code Snippet 11 shows how to set a `return` type that is different from the argument types by ensuring the `return` is the correct type.

#### **Code Snippet 11:**

```
<?php declare(strict_types=1); // strict requirement
function SumNumbers(float $x1, float $x2) : int {
    return (int)($x1 + $x2);
}
echo SumNumbers(5.2, 7.2);
?>
```

In the code, two floating point numbers are added, but the sum displayed is an integer. This is because, in the function declaration `SumNumbers()`, `int` is entered after the `:`. The two arguments and their data types passed are `float $x1` and `float $x2` respectively, but an integer value of sum is returned. `int` datatype specified after the keyword `return` displays an integer as the output while adding values of two variables `x1` and `x2`. Thus, the sum of `(5.2, 7.2)` is returned as `12` as shown in Figure 7.11.



**Figure 7.11: Setting a Return Type**

## 7.8 Passing Arguments by Reference

---

In PHP, arguments can be passed using one of two approaches:

- Pass by Value
- Pass by Reference

Generally, in PHP, by default, arguments are entered by value. This means that a replica of the value is utilized in the function and the actual variable that was entered into the function is not modified.

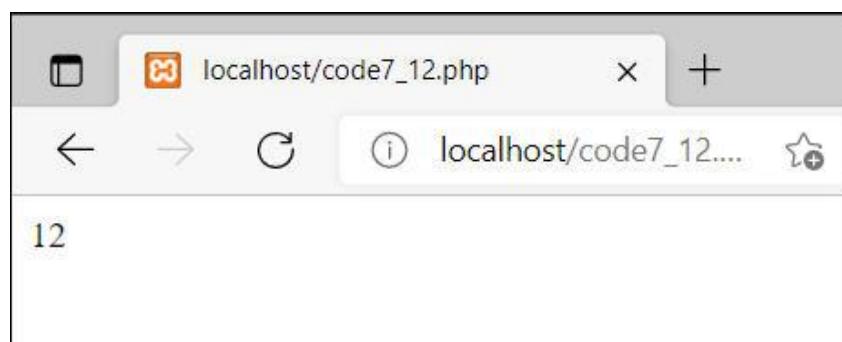
However, if a function argument is passed by reference, any modification done to the argument within the function will automatically modify the variable passed. In order to change a function argument into a reference, users can use the `&` operator.

Code Snippet 12 shows how to pass function arguments by reference.

### Code Snippet 12:

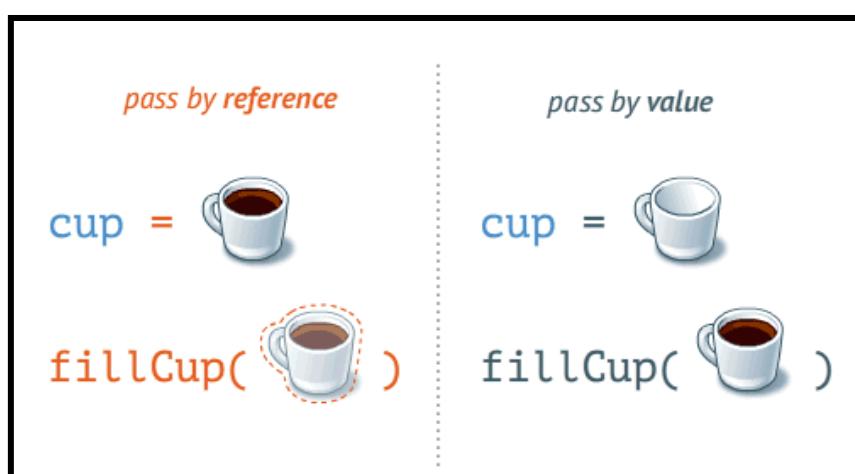
```
<?php
function add(&$value1) {
    $value1 += 10;
}
$num1 = 2;
add($num1);
echo $num1;
?>
```

In the code, a user-defined function named `add()` is created and an argument is passed named `value1`, which stores the number passed. Later, another variable `num1=2` is declared. Then, the function `add()` is called by passing the reference of `num1` variable. Therefore, it calculates the value as  $2+10=12$ . Here, it can be seen that the `value1` variable is replaced by `num1` when the function `add($num1)` is called. In this example, the original argument passed to the function is itself modified and hence, the modified value is reflected in the echo statement. Thus, in this approach, pass by reference, the original argument does not remain unchanged after the function has been called. Instead, it reflects the change made within the function body. Figure 7.12 depicts the output.



*Figure 7.12: Passing Arguments by Reference*

Figure 7.13 shows the concept of pass by reference and pass by value.



*Figure 7.13: Pass By Reference and Pass By Value*

In case of pass by reference, if any changes are made to the function definition the values also change. As seen in Figure 7.13, when a coffee mug is filled, that is, `cup=fillCup`, the changes reflect outside the function when `fillcup()` function is called. However, in case of pass by value, the value is

not affected by any changes made to `fillcup()` function.

## 7.9 Dynamic Function Calls

---

PHP allows users to allocate function names as strings to variables and later utilize these variables in the same way as the function name.

Code Snippet 13 shows how to call a function dynamically.

### Code Snippet 13:

```
<?php
function Test()
{
echo "Statement is displayed by a dynamic function call<br
/>";
}
$function_holder = "Test";
$function_holder();
?>
```

In the code, a function `Test()` is created, which is used to display a message. The function name is assigned as a string value to the variable `$function_holder` using the concept of dynamic function call. Later, variable `function_holder` is called similar to how a regular function would be called. This will result in function `Test()` being called indirectly.

This concept is used when users prefer not to call the function with the same name used during function declaration and definition. Therefore, the name is assigned to another variable and the function is called by that name during a function call.



*Figure 7.14: Using Dynamic Function Calls*

## 7.10 PHP Named Arguments

---

Named arguments is a new feature introduced in PHP 8.0. It permits users to pass arguments to a function considering only the parameter names and not the parameter positions. This allows users to pass the arguments without

being concerned about the position of the parameter. They can simply specify the correct names of the parameters and pass values to the function.

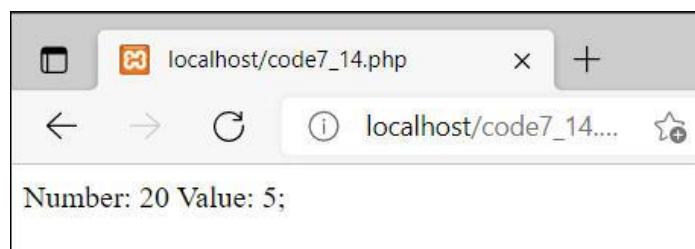
Code Snippet 14 shows how to display numbers using named arguments of a function.

#### Code Snippet 14:

```
<?php
function named_arguments ($number = 11, $value1 = 5) {
echo "Number: "; $number;
echo " ";
echo "Value: "; $value1;
}
named_arguments (value1: 5, number: 20); //Named arguments in
// different order
?>;
```

In the code, while defining the function, `number` parameter is written first and then, `value1` parameter.

However, while calling the function `named_arguments()`, `value1` variable is passed first and then, the `number` variable. That is, value 5 is passed first and then, 20 is passed. Note that the output is according to the function definition itself that is, the updated value of `number` variable (20) is displayed first and then, the `value1` variable value (5). Figure 7.15 depicts this output.



**Figure 7.15: Named Arguments**

From the example, it is evident that the order of the arguments during a function call does not matter. The arguments can be written in any order, but the name of the arguments during the function call must match with the parameters written during the function definition.

### **7.11 date() and time() Functions**

PHP also provides support for date and time.

## ➤ **date()** Function

The PHP `date()` function displays the current date and/or time of the server. This function helps users to format a given timestamp such that the date and time are more readable. A timestamp can be defined as a sequence of characters, representing the date and/or time during which a particular event took place.

Following is the syntax for the same:

```
date(format,timestamp)
```

The `format` parameter indicates how the date must be formatted. Following are some commonly utilized characters to represent the date:

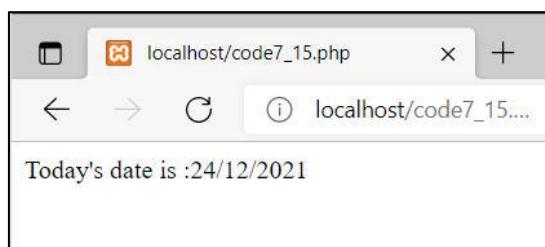
d	m	Y	l
This character indicates the day of the month in two digits, for example, 01 to 31.	This character indicates the month from 01 to 12.	This character indicates the year in four digits.	This character, which is the lowercase of the letter L, indicates the day of the week.

To perform additional formatting, users can utilize characters such as ‘/’, ‘.’, or ‘-’ between the characters. Code Snippet 15 shows how to format today's date in various ways.

### Code Snippet 15:

```
<?php  
echo "Today's date is :";  
$today1 = date("d/m/Y");  
echo $today1;  
?>
```

In the code, today's date is displayed using the built-in function `date()`. Arguments are passed to the `date` function to display the date as shown in the output. Figure 7.16 depicts the output.



**Figure 7.16: Using `date()`**

## ➤ **time()** Function

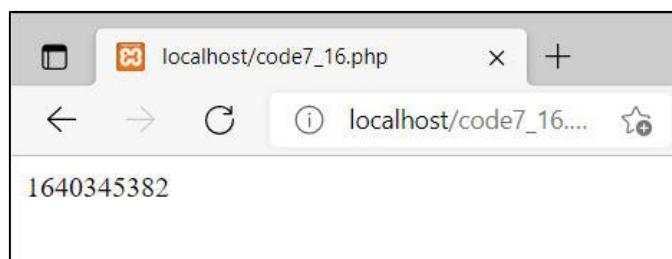
The PHP `time()` function displays the current time in terms of seconds. The `date()` function helps in converting the number of seconds to the current date.

Code Snippet 16 shows how to print the current time in seconds.

### Code Snippet 16:

```
<?php  
$currentTimeInSeconds = time();  
echo $currentTimeInSeconds;  
?>
```

In the code, the built-in function `time()` is used to display the current time in seconds. The function `time()` is assigned to a variable `currentTimeInSeconds` and then, the result is displayed using the `echo` keyword.



**Figure 7.17: Using `time()`**

Following are some commonly utilized characters to represent time:

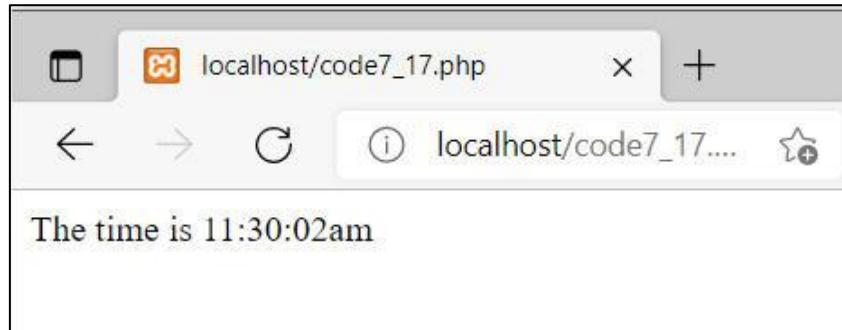
H	h	i	s	A
This character indicates a 24-hour format of an hour from 00 to 23.	This character indicates a 12-hour format of an hour from 01 to 12.	This character indicates the minutes from 00 to 59.	This character indicates the seconds from 00 to 59.	This character indicates Ante Meridiem (AM) and Post Meridiem (PM).

Code Snippet 17 shows how to output the current time in the specified format.

### **Code Snippet 17:**

```
<?php  
echo "The time is " . date("h:i:sa");  
?>
```

The code prints the current time using the commonly utilized characters in hour, minutes, and seconds format as shown in the output. Figure 7.18 shows the output.



*Figure 7.18: Using time() with Format*

## 7.12 Summary

- PHP has over 1000 built-in functions that help users to complete common tasks. This makes it easy for the users to execute and rerun common tasks in a program.
- Some of the advantages of using PHP functions are reusable code, minimal code, and clarity of code.
- PHP gives users the flexibility to create their own functions. User-defined functions can be created according to the task user wants to perform.
- An argument is similar to a variable and is used to pass input data into functions.
- Default argument value will be utilized if no value is entered for the argument when the function is executed.
- If a function argument is entered by reference, any modification done to the argument will automatically modify the variable that was entered.
- An important aspect of PHP functions is that they can also return results during a later stage of the program.
- PHP allows users to allocate function names as strings to variables and later utilize these variables in the same way as the function name.
- Named arguments is a new feature introduced in PHP 8, which permits users to pass arguments to a function considering only the parameter names and not the parameter positions.
- While the PHP `date()` function displays the current date and/or time of the server, the `time()` function displays the current time in terms of seconds.

## 7.13 Test Your Knowledge



1. Which of the following is not a built-in function in PHP?
  - a. print\_r()
  - b. fopen()
  - c. fclose()
  - d. filetype()
2. Which of the following cannot be used at the beginning of a function name?
  - a. alphabet
  - b. underscore
  - c. number
  - d. Both b and c
3. A function name is not case-sensitive (State True/False).
  - a. True
  - b. False
4. Which of the following is the correct output of the following code?

```
<?php declare(strict_types=1); // strict requirement ?>
<?php
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}
setHeight(350);
?>
```

- a. The height is: 350
- b. The height is: 50
- c. The height is: 1550
- d. Error

5. Which of the following is the correct output of the following?

```
<?php declare(strict_types=1); // strict requirement  
  
function addNumbers(int $a, int $b) {  
    return $a + $b;  
}  
echo addNumbers(5, "5 days");  
?>
```

- a. 5+5days
- b. 55days
- c. 25 days
- d. Error

## **Answers to Test Your Knowledge**

---

1. `print_r()`
2. Both b and c
3. True
4. The height is: 350
5. 5+5 days

## 7.14 Try it Yourself

1. Write a program to add two numbers using a user-defined function `Add()`.
2. Write a program in which only one argument is passed to a function to display the names of people with the same last name.
3. Write a program to explain the default argument value.
4. Write a program to return a value of a function using the `return` statement.
5. Write a program to explain the passing an argument by reference to a function.
6. Write a program to display today's date and time using the dynamic function `date()`.
7. Write a program to display today's date in three different formats.

# Session 8

## Cookies and Sessions Management in PHP



### Learning Objectives

*In this session, students will learn to:*

- Define PHP cookies and how they function
- Explain setting of cookies with PHP through several functions
- Identify how to work with cookies
- Elaborate on PHP Session management

The session begins by defining what a cookie is. The session then provides an overview on the anatomy of the cookie and explains how to work with cookies in PHP. Further it covers details on sessions in PHP and how to work with them. It also explains how to create, destroy, and modify PHP user sessions.

### **8.1 Introduction to Cookies**

---

A cookie in the world of Web and computer networks, is a tiny chunk of data carrying information that comes handy. Cookies enable Websites to store user information.

#### **8.1.1 What is a Cookie?**

Websites store user information in databases to maintain a track of their visits. However, there are users who do not register with the Website but frequently visit the Website. Cookies are used to store temporary information of such visitors.

Another instance is that of online shopping Website, where a user selects few products and adds them to a shopping cart. The user then navigates to the next page and selects additional products. In such a situation, user selection must be stored before the user navigates to the next page. Although the user has not yet completed the transaction, the Website stores the data in a variable called cookie.

Often, a cookie is a small piece of data that just identifies the user's information where the user is connected to a computer network. The values in cookies can be both created and retrieved through PHP. They are created at the Web server side and stored at the client Web browser.

Websites store two types of data namely, temporary and permanent. Temporary information is stored in cookies for a stipulated period. Permanent data is stored in cookies for a certain period and then, the required information is saved in the database. Websites use two types of cookies and they are as follows:

Persistent	Non-persistent
<ul style="list-style-type: none"><li>Exist in the Web browser for a period specified at the time of its creation</li></ul>	<ul style="list-style-type: none"><li>Deleted from the Web browser as soon as the user exits the browser</li></ul>

As cookies are text files saved on the client computer, these come handy for tracking purposes.

Steps involved in identifying returning users are as follows:



Websites use cookies to determine the following:

- Number of times the user has visited the Website
- Number of new visitors

- Number of regular users
- Frequency of a user visiting the Website
- Date on which the user had last visited the Website
- Customized Web page settings for a user

When a user visits the Website for the first time, the Web server creates a unique ID and sends the ID in the cookie to the Web browser. The browser stores the cookie and sends it back to the Website in subsequent requests. The Web server can read the information stored in the cookie only when it is related to the Website. The life of a cookie depends on the expiration time and date. The cookie is stored on the hard disk of the user's computer. This enables the Website to keep a track on the user visiting the Website. The information about the user is generally stored in the name-value pair.

PHP supports HTTP cookies. PHP cookies must always be used before the `<html>` tag.

## **8.2 Setting Cookies with PHP**

---

Cookies are incorporated in HTTP request and response headers. Setting a cookie means sending the cookie to the browser.

Web servers and Web browsers send cookies to each other in HTTP headers. The Web server sends the cookie to the browser in the `Set-Cookie` header field. This field is a part of the HTTP response. The Web browser stores the cookie and uses the same in subsequent requests to the same Web server.

Generally, HTTP headers are used to set cookies. They can also be set directly on a Web browser with the help of JavaScript.

Following example shows how a header sent via a PHP script can set a cookie:

```
HTTP/1.1 200 OK
Date: Wed, 08 Dec 2021 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=abc; expires=Wednesday, 08-Dec-21 22:03:38
GMT;
path=/; domain=aptech-education.com
Connection: close
Content-Type: text/html
```

The `Set-Cookie` part of the header here contains the name-value pair (both of which are URL encoded), a GMT date, a path, and a domain.

The Web browser, when configured to save cookies, will preserve the data until a given date (Wednesday, 08-Dec-2021 21:03:38 GMT). It will forget this information once the stipulated time lapses. The `expires` field contains the date until when the data has to be remembered by the browser. The cookie will be resent to the server, if the user points the Web browser at any Web page which is same as the cookie path and domain.

Following example shows how that Web browser's header may looks like:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X x.y;
rv:42.0) Gecko/20100101 Firefox/42.0 Host: hype.viz.co.uk:1126
Accept: image/jpeg, */
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=pqr
```

Once the Web browser header is set, a PHP script will have access to the cookie in the environmental variables. `$_COOKIE` or `$HTTP_COOKIE_VARS[]` is a superglobal variable holds all cookie names and values. Access to this cookie shown here can be provided using `$HTTP_COOKIE_VARS["name"]`.

### 8.2.1 PHP `setcookie()` Function

In PHP, a cookie can be set with the HTTP response using `setcookie()` function. Once the cookie is set, the `$_COOKIE` superglobal variable is used to access it. Following is the syntax for setting the cookie:

#### Syntax:

```
setcookie ( string $name [, string $value [, int $expire = 0 [, string $path[, string $domain [, bool $secure = false [, bool $httponly = false ]]]]]] )
```

Example of `setcookie()` function to define name and value is as follows:

```
setcookie("Sale_Item", "Denim Jeans");
```

Following example shows how to set expiry for cookies using `time()` function:

```
setcookie("Sale_Item", "Denim Jeans", time() + 2 * 48 * 60 * 60);
```

An example to set the path on the server where the cookie is accessible on and domain that the cookie is accessible to is as follows:

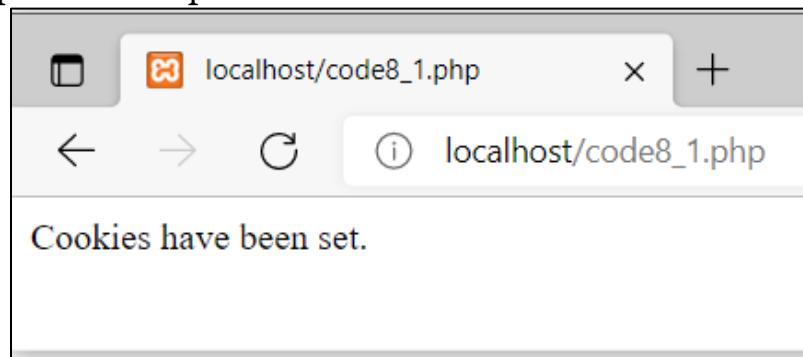
```
setcookie("Sale_Item", "Denim Jeans", time() + 2 * 48 * 60 * 60, "/mydir/", "tigerdomain.com", 1);
```

Code Snippet 1 shows how to create two cookies, `name` and `city`, which expire after half an hour.

### Code Snippet 1:

```
<?php  
setcookie("name", "George", time()+1800, "/", "", 0);  
setcookie("city", "New York", time()+1800, "/", "", 0);  
?  
<html>  
<body>  
<?php echo "Cookies have been set."?>  
</body>  
</html>
```

Figure 8.1 depicts the output of the code.



**Figure 8.1: Setting Cookies Through PHP Script**

### 8.2.2 Retrieving Cookie Values Using PHP `$_COOKIE`

Of the many ways that PHP allows access to cookies, the simplest way is to make use of either - `$_COOKIE` variable or `$HTTP_COOKIE_VARS` variable.

`$_COOKIE` is a superglobal variable in PHP and is used to retrieve value of a cookie.

### Syntax:

```
$value=$_COOKIE ["CookieName"] ;
```

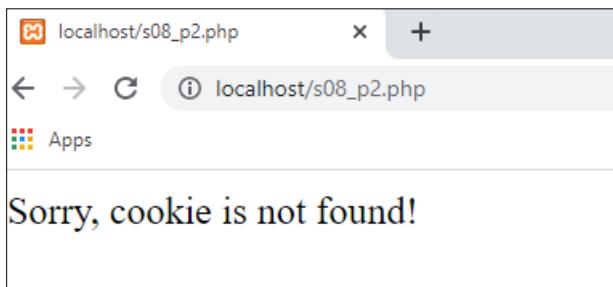
Code Snippet 2 is an example for retrieving a PHP cookie.

### Code Snippet 2:

```
<?php  
setcookie("user", "John");  
?>  
<html>  
<body>  
<?php  
if(!isset($_COOKIE["user"])) {  
    echo "Sorry, cookie is not found!";  
} else {  
    echo "<br/>Cookie Value: " . $_COOKIE["user"];  
}  
?>  
</body>  
</html>
```

In this code, at first when the code executes, cookie will not set, but after refreshing the browser, cookies will be set and output displays `Cookie value: John`

Figure 8.2 depicts the initial output of the code.



**Figure 8.2: Attempting to Retrieve Cookie Value Before Page Refresh**

Upon refreshing the page, the cookie will be set and the value displayed will be as shown in Figure 8.3.



**Figure 8.3: Retrieving Cookie Value After Page Refresh**

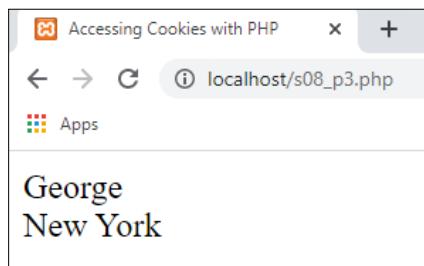
Code Snippet 3 shows how to retrieve all the cookies that were set in Code Snippet 1.

### **Code Snippet 3:**

```
<html>
<head>
<title>Retrieving Multiple Cookies with PHP</title>
</head>
<body>
<?php
echo $_COOKIE["name"] . "<br />";
echo $_COOKIE["city"] . "<br />";
?>
</body>
</html>
```

In this code too, cookie values are accessed using the superglobal cookie variable. In Code Snippet 1, cookie values for name and city were set to George and New York respectively. Those cookie values are now retrieved and displayed.

Figure 8.4 shows the output.



**Figure 8.4: Retrieving Multiple Cookie Values After Page Refresh**

## **8.3 Working with Cookies in PHP**

---

Besides creating cookies as shown in examples described so far, one can also modify and delete cookies using different functions.

### **8.3.1 Modifying Cookies**

The `setcookie()` function is not only used to create a cookie, but can be used to modify it as well.

#### **Syntax:**

```
setcookie (name, value, expire, path, domain, secure, httponly);
```

When you want to update an existing cookie with a new value, it is recommended to add '/' in the fourth argument which is the 'path' argument, to prevent creating another cookie with the same name.

#### **Code Snippet 4:**

```
<?php  
setcookie("name", "George Miller", time()+1800, "/", "", 0);  
setcookie("city", "New York", time()+1800, "/", "", 0);  
setcookie("name", "David Smith", time()+86400, "/", "", 0);  
?>  
<html>  
<body>  
<?php  
echo "Cookies have been set.";  
echo $_COOKIE["name"]. "<br />";  
?>  
</body>  
</html>
```

When Code Snippet 4 is executed, the output will show David Smith instead of George Miller as the cookie value for name because the existing cookie has been modified.

### **8.3.2 Deleting a Cookie**

The `setcookie()` function with a past expiration date is used to delete a cookie in PHP. After a certain period of time, the cookie mentioned inside `setcookie` function will automatically be deleted.

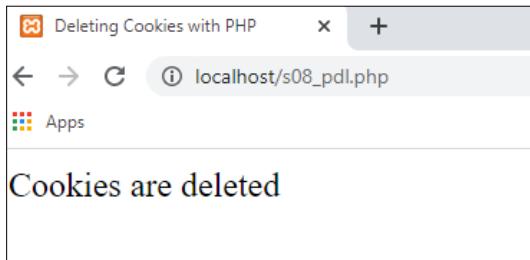
Code Snippet 5 shows how to delete a cookie.

### Code Snippet 5:

```
<?php
    setcookie("name", "", time()- 30, "/", "", 0);
    setcookie("city", "", time()- 30, "/", "", 0);
?>
<html>
    <body>
        <?php
            echo "Cookies are deleted" ;
?>
        </body>
    </html>
```

In this code, the expiration date and time are set to half an hour before current time. As an outcome of this code, cookies that were set in Code Snippet 3 are deleted after half an hour and expiry date.

Figure 8.5 displays the output of the code.



*Figure 8.5: Deleting Cookies*

#### 8.3.3 Checking Whether Cookies are Enabled

Sometimes, certain browsers may be configured to disable cookies. These days, due to privacy concerns, end users can withhold consent to having cookies stored on their machine. This means that if they deny the access to having cookies stored, your PHP script with cookies will not work.

Code Snippet 6 helps to inspect whether the cookies are enabled by the browser or disabled.

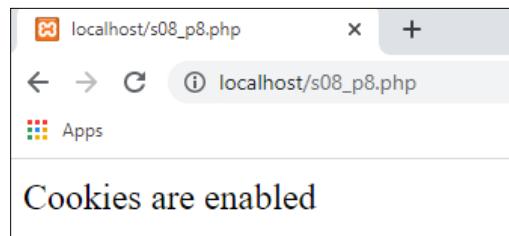
To begin with, create a cookie with the `setcookie()` function, then check the contents of `$_COOKIE` array variable.

### Code Snippet 6:

```
<?php  
    setcookie("product", "Microwave Oven");  
?>  
<html>  
<body>  
<?php  
    if (isset($_COOKIE['product'])) {  
        echo "Cookies are enabled";  
    }  
    else {  
        echo "Cookies are not enabled";  
    }  
?>  
</body>  
</html>
```

The function `isset()` is used to check whether a cookie named `product` is set or not in this code. The code checks if it is set or not. If it is set, cookies are enabled by the browser. If not, cookies are disabled by the browser.

Figure 8.6 shows a scenario of a browser that has cookies enabled.



**Figure 8.6: Checking Whether Browser Has Cookies Enabled**

## 8.4 Working with Sessions in PHP

---

Consider a scenario, where the user opens an application, does some edits and then, closes the application. This is similar to a Web session for a user. The computer identifies who the user is and the duration for which the user was logged in, while the user is working on an application.

However, on the Internet, as the HTTP address does not maintain this state, the Web server cannot recognize the user and usage details. In order to record this data, a PHP session comes handy; it temporarily stores and passes the information from one Web page to another, until the Website is closed.

#### **8.4.1 Cookies and Sessions**

Cookies enable to store data into a variable and access it across all the pages of the Website. Cookies are prone to security risks because user information is saved at the client-end. Risks involved are greater when users access Websites from public computers or shared computers.

Following are a few other disadvantages of using cookies:

- Deletion of cookies - Users can easily delete cookies from a client system. Cookies are created in temporary file folders. Users often delete temporary Internet files to improve performance of a system. Then, the Websites allot a new cookie to the user.
- Multiple cookies to the same user - Cookies enable Websites to identify users according to their computers. Websites allot a different cookie to the same person every time the user accesses the Website from different computers. The statistics of the Website records a new user entry for the same person using a different computer. In addition, a user has to set all the preferences again on different computers to visit the same Website.
- Size of the cookie - The amount of information stored in the cookie determines the size of the cookie. The size of the cookie determines the size of the Web page. Therefore, the size of the Web page increases when large amount of information is stored in a cookie. The increase in file size of the Web page results in poor performance.
- Cookies disabled - Websites store cookies on the hard disk of the client. This reduces the performance of computers with a low memory space. To improve the performance of such computers, users disable cookies. This makes the process of assigning cookies pointless.

The major difference between cookies and sessions is that pertinent cookies store information on the local computer, whereas a session enables PHP to store information on the Web server.

Table 8.1 explains the summarized difference between cookies and sessions.

Cookies	Sessions
Store user information on the client system (Web Browser)	Store user information on the Web server
Available even after user exits the Web browser	Destroyed when user exits the Web browser
Users can disable cookies	Users cannot disable sessions
Have size limits	Do not have size limits

**Table 8.1: Differences Between Cookies and Sessions**

#### 8.4.2 Using Sessions

The approach of utilizing PHP sessions is common in retail shopping sites where the information is frequently stored and the cart information is passed on from page to page. For example, username, products selected into the cart, and so on are stored and passed on from one Web page to another.

For every Web browser, a unique user id is created by a PHP session, to remember and recognize the user, and to avoid dispute/conflict between multiple Web browsers.

Session variables save user information to be used across multiple Web pages and they are kept alive until the Web browser is closed by the user. For one application, session variables are available for all pages. They save information about one single user.

Registered session variables and their values are stored in a file that is created by a session. This file is created in a temporary directory on the Web server. The data saved will be available to all pages on the site during that visit. The location of this temporary file can be obtained by setting a path called `session.save_path` in the `php.ini` file. One must set up this path prior to using any session variable.

Following occurrences take place when a session starts:

PHP creates a unique identifier for the session; This is a random string of 32 hexadecimal digits.

For example:

uuukae2pq37gljo7eae2f62  
499

The PHPSESSID cookie is automatically sent to the user's computer to save the unique session identification string.

An automatically created file on the Web server in the designated temporary directory bears the name of the unique identifier.

**Note:** The automatically created file is prefixed by sess\_, such as sess\_uuukae2pq37gljo7eae2f62499

PHP automatically retrieves the unique session identifier string from the PHPSESSID cookie when a PHP script requires a value from the session variable. It then checks in its temporary directory for the file with the same name so that verification can be done by collating both the values. The Web server usually terminates a session after a predetermined time (generally 30 minutes) if the user does not refresh a Web page in the application. A session is also terminated when a user loses the Web browser connection or leaves the Web page.

#### 8.4.3 PHP Session Lifecycle

There are three stages in the lifecycle of a session based on communication between Web browser and Web server, as follows:

- Starting the session
- Registering the session variable
- Ending the session

#### 8.4.4 Starting a PHP Session

A PHP session starts with making a call to the `session_start()` function. This function first checks whether a session is already started, if not then, it starts one. The call to `session_start()` function should be put at the beginning of the Web page.

The associative array `$_SESSION[]` saves the session variables which can be accessed during a session's lifetime.

Each time the Web page is visited during the session, the variable called counter gets incremented. This variable is registered at the beginning of the session.

**Note:** The function `isset()` checks whether the session variable is already set.

**Syntax:**

```
bool session_start (void)
```

The global variable `$_SESSION` is set along with session variables.

Code Snippet 7 creates a new Web page called `demo_session.php`. In the code, a new PHP session has begun and some session variables are set on this Web page.

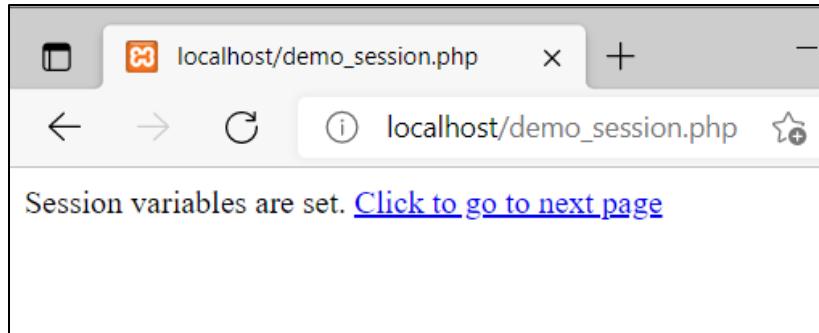
**Code Snippet 7:**

```
<?php
session_start(); // This begins the session
// set the session variables
$_SESSION["studentname"] = "Peter Hayward";
$_SESSION["studentid"] = "1623";
?>
<html>
    <body>
        <?php
            echo "Session variables are set.";
        ?>
        <a href="next.php">Click to go to next page</a>
    </body>
</html>
```

In this code, a session is started and then, the session variables `studentname` and `studentid` are declared. To start a session, `session_start()` function is invoked. If session variable is set then, the output displays Session variables are set along with a hyperlink to navigate to another page.

**Note:** The statement invoking `session_start()` function must always be the first statement of the Web page prior to any HTML tag.

Figure 8.7 shows the output of the code.



*Figure 8.7: Creating Session Variables*

#### 8.4.5 Getting PHP Session Variable Values

In Code Snippet 9, a session started and two session variables were set; it also offered a link to navigate to the second Web page `next.php`.

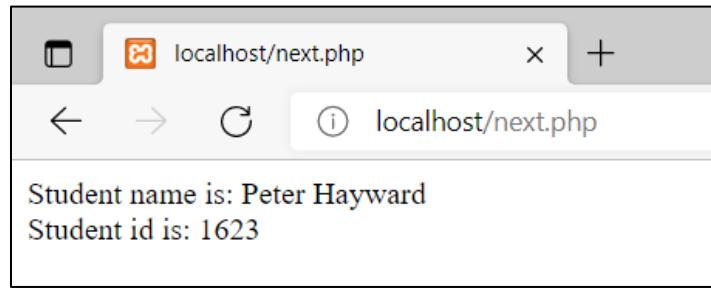
Code Snippet 8 shows how to fetch the variables that were set in the `demo_session.php`.

#### Code Snippet 8:

```
<?php
session_start(); // This begins the session
// Retrieve the session variable values
$studentname = $_SESSION["studentname"];
$studentid = $_SESSION["studentid"];
?>
<html>
    <body>
        <?php
            echo "Student name is: ".$studentname."<br/>";
            echo "Student id is: ".$studentid;
        ?>
    </body>
</html>
```

One may wonder why the function `session_start()` is used in Code Snippet 8 even though no new variables are set in the session variable. The reason is, `session_start()` initializes a new session (or fetches the ongoing session if already started). Global variable `$_SESSION` is used to set new values into the session or return the saved values.

Figure 8.8 shows the output.



**Figure 8.8: Retrieving Session Variable Values**

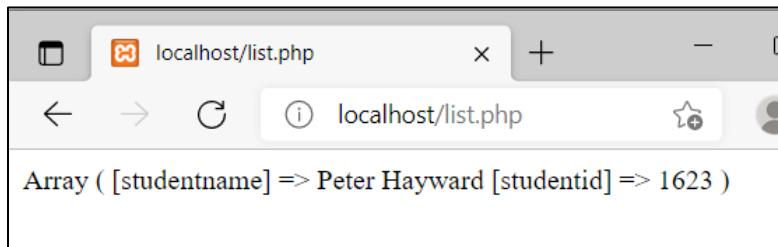
Code Snippet 9 prints all variable data related to the ongoing session. This code can be used if there are too many values saved in the session.

### **Code Snippet 9:**

```
<?php  
session_start(); // This begins the session  
  
?>  
<html>  
    <body>  
        <?php  
        print_r($_SESSION);  
        ?>  
    </body>  
</html>
```

As an outcome of the code, all existing session variables will be printed as a list. Hence, when the code is executed, the session variables that were set in Code Snippet 8 are listed. The code uses function `print_r()` and the session is called using `$_SESSION`.

Figure 8.9 shows the output.



**Figure 8.9: Listing All Session Variables and Values**

#### 8.4.6 Destroying a PHP Session

Built-in function `session_unset()` removes all saved values from the session variable and cleans the session variable. However, the session itself will remain active. Function `session_destroy()` destroys the session.

These functions are usually used on Web pages of retail shopping Websites in order to clean the session variable off user-specific data and to destroy the current session eventually. For example, logout or checkout pages may use these functions.

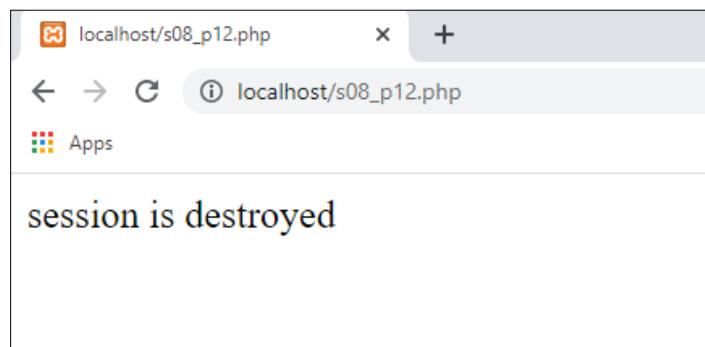
Code Snippet 10 shows how to destroy a PHP session.

#### Code Snippet 10:

```
<?php
session_start(); // This begins the session
?>
<html>
    <body>
        <?php
        // destroy the session
        session_destroy();
        echo "session is destroyed";
    ?>
    </body>
</html>
```

In this code, the session that was set in Code Snippet 9 is destroyed. When the function `session_destroy()` is called, all the sessions are cleared automatically.

Figure 8.10 shows the output.



*Figure 8.10: Destroying a Session*

#### 8.4.7 Modifying a PHP Session Variable

Updating/modifying a value saved or stored in the session variable can be performed easily. Begin the session by calling the `session_start()` function and then, simply overwrite the new value over the old one in the session variable. Code Snippet 11 shows how to update the session variable in PHP.

##### Code Snippet 11:

```
<?php
// start the session
session_start();
$studentname = $_SESSION["studentname"];
$studentid = $_SESSION["studentid"];

// update the session variable values
$_SESSION["studentid"] = "1024";
?>
<html>
    <body>
        <?php
        echo "Student name is: ".$studentname"<br/>";
        echo "Student id is: ".$studentid;
        ?>
    </body>
</html>
```

In Code Snippet 11, the value of student name is updated to David from Peter and `studentid` is updated to 1024 from 1623 which was set in Code Snippet 9. Figure 8.11 shows the output.



*Figure 8.11: Modifying a Session*

#### **8.4.8 Turning ON Auto Session**

PHP facilitates automatic start of a session through setting of a built-in boolean variable, `session.auto_start`. The value of this variable specifies whether the session module starts a session automatically on a page request.

When a user visits a Website, if `session.auto_start` is already set, then the session starts automatically and there is no necessity to call `start_session()` function in order to start a session.

By default, it is set to 0, which means automatic session start is disabled.

#### **8.4.9 Sessions Without Cookies**

If a user does not want the cookies to be stored on his/her local machine, then a Session ID (SID) can be sent to the browser through an alternate method. The constant `SID` which is a string containing session name and ID defined when the session started can be used in such cases. The built-in function `session_id()` can retrieve the session id for the current session. This can be embedded into URLs.

Code Snippet 12 initializes a session variable and sends it to another Web page through a SID value in the query string.

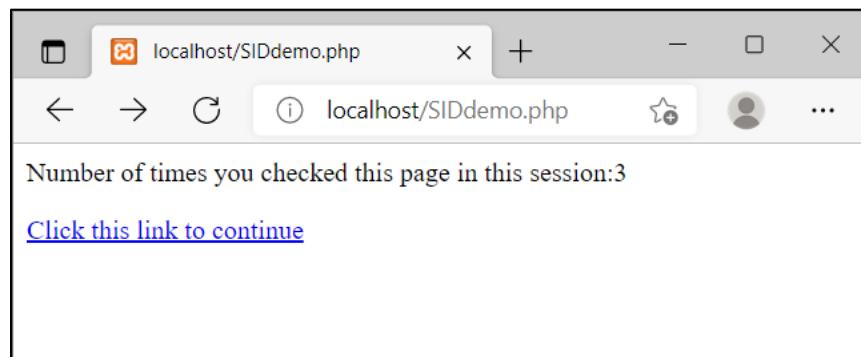
#### **Code Snippet 12:**

```
<?php
error_reporting(0);
$sess_id = session_id();
if(empty($sess_id))
    session_start();
if (!isset($_SESSION['counter'])) {
$_SESSION['counter'] = 1;
} else {
$_SESSION['counter']++;
}
$msg = "Number of times you checked this page in this
session:". $_SESSION['counter'];
echo $msg;
?>
<p>
```

```
<a href="nextpg.php?id=<?php echo  
htmlspecialchars(session_id());?>">Click this link to  
continue<a>  
</p>
```

In this code, cookies are not stored, instead session id is sent from one page to another. Using `isset()` function, the page of the browser is checked for counter value. If counter is not yet set, its value will be assigned as 1, otherwise it will be incremented. This means that each time the page is loaded or the page is refreshed, the counter value will increase. Current session id is retrieved using `session_id()`. The id is then added to a hyperlink as a query string. The `htmlspecialchars()` function is used in order to prevent XSS related attacks while rendering the SID in the URL through echo.

Figure 8.12 depicts the output when the page has been refreshed three times.



**Figure 8.12: Session without Cookies**

When the link on the page displayed in browser is clicked, the URL for the redirected page is of the form:

<http://localhost/nextpg.php?c7uuukae2pq37g1jo7eae2f624>

As you can observe, the session id has been embedded with the URL as a query string. In `nextpg.php`, this query string can be retrieved and displayed. Code Snippet 13 shows the code to do this.

### **Code Snippet 13:**

```
<?php  
error_reporting(0);  
$prevpage sess_id = $_GET['id'];  
if(!empty($prevpage sess_id))  
    echo $prevpage sess_id;  
?>
```

Thus, user session information has been sent from one page to another without using cookies.

## 8.5 Summary

- A cookie is a small file created at the Web server side and stored at the client Web browser.
- PHP cookies can be created, modified, retrieved, and even deleted.
- `setcookie()` function in PHP enables to create and modify cookies.
- Cookies have some disadvantages; hence, PHP sessions are sometimes used in place of them.
- PHP sessions temporarily store and pass the information from one Web page to another, until the Website is closed.
- The major difference between cookies and sessions is that cookies store information on local computer, whereas a session stores information on the Web server.
- User data is stored in session variables to be utilized across multiple Web pages and they last until the Web browser is closed by the user.
- There are three stages in the lifecycle of a session based on communication between Web browser and Web server, namely, Starting the session, registering the session variable, and ending the session.
- A PHP session starts by making a call to the `session_start()` function.
- `session_unset()` function removes all the saved values from the session variable and cleans the session variable.
- Session information can be sent from one page to another without cookies, by using SID.

## 8.6 Test Your Knowledge



1. Which of the following functions set a cookie in PHP?
  - a) `createcookie()`
  - b) `makecookie()`
  - c) `setcookie()`
  - d) None of these
2. The \_\_\_\_\_ option enables a session to automatically initialize if the session ID is not found in the browser request.
  - a) `session.start`
  - b) `session.auto_start`
  - c) `session.session_start`
  - d) `session.session_auto_start`
3. Identify the files that can be read and written from the Web server via PHP scripts and store user or session information.
  - a) Session
  - b) Cookies
  - c) Bytecode
  - d) All of them
4. Built-in function \_\_\_\_\_ removes all saved values from the session variable and cleans the session variable.
  - a) `session_unset()`
  - b) `session_clean()`
  - c) `session_remove()`
  - d) `session_clear()`
5. The \_\_\_\_\_ function checks whether the cookie is set.
  - a) `$_COOKIE[]`
  - b) `set()`
  - c) `setcookie()`
  - d) `isset()`

## **Answers to Test Your Knowledge**

---

1. setcookie( )
2. session.auto\_start
3. Session and Cookies
4. session\_unset()
5. isset()

## **8.7 Try it Yourself**

1. Write a program to create cookies that expires in two hours.
2. Write a program to check whether cookie is enabled by the browser, then, create cookies, and retrieve all the values of cookies.

# Session 9

## Database Management in PHP



### Learning Objectives

*In this session, students will learn to:*

- Describe PHP support for MySQL
- Identify prerequisites for databases in PHP
- Elaborate how to establish MySQL database connection
- Describe how to create and delete MySQL Database and tables using MySQL in PHP
- Explain data insertion and data retrieval to/from MySQL database
- Explain how to update data into MySQL database
- List and explain the three ways to backup MySQL database
- Explain selecting and filtering data from MySQL database using WHERE, ORDER BY, and LIMIT clauses

The session begins by giving a brief overview of PHP support for MySQL. The session explains about establishing a MySQL database connection. The session elaborates how to create and delete databases and tables in MySQL using PHP. Next, users will learn about data insertion and data retrieval to or from MySQL database. This session finally explains selecting and filtering data from MySQL database using various clauses.

### **9.1 PHP and MySQL**

---

PHP Web applications will involve data handling at some point or the other. A shopping cart application, an application to display and process a login form, and a ticket booking application – these are common examples of applications

that will use data. This data will be stored in databases to persist them. To store, retrieve, and maintain such data through scripts, PHP supports many database management systems including MySQL, MariaDB, Db2, MongoDB, Oracle, PostgreSQL, and SQLite.

For MySQL, versions 5.5 and newer are supported. For MariaDB, versions 5.5 and newer are supported.

### What is MySQL?

First developed by Oracle, MySQL is a Relational Database Management System (RDBMS) that is a free and open-source software. MySQL mostly uses the standard SQL and runs on the server, along with being used in Web application development. Being exceptionally fast, easy-to-use, and reliable, it helps organize the data in databases and is an ideal choice for small and large applications. PHP, along with MySQL, is cross-platform.

The 'My' in MySQL stands for the cofounder Monty Widenius's daughter's name and SQL for Structured Query Language.

In MySQL, data is stored in tables. A table can be defined as a data collection that has rows and columns. Databases can store information categorically. For example, the database of a company may have different categories of tables such as Employees, Customers, Products, Orders, and so on.

## 9.2 Prerequisites for Database in PHP

---

Key Prerequisite to use MySQL database in PHP 8 is that mysqli support should be included. When user is using XAMPP with PHP and MySQL installations, user can configure the `php.ini` as follows:

1. Click Config in XAMPP Control Panel and then, select `php.ini` to open it.
2. Locate the line `extension=mysqli` and uncomment it.
3. Locate for the line `mysqli.default_port` and set it to 3306.
4. Save the file.
5. Start Apache in XAMPP Control Panel.
6. Start MySQL in XAMPP Control Panel.
7. Click Admin option in XAMPP Control Panel next to MySQL. This will launch phpMyAdmin, the client through which user can connect to MySQL. A variant of MySQL called MariaDB will be used with XAMPP.
8. Verify through phpMyAdmin that MySQL is running.

Figure 9.1 depicts the phpMyAdmin page. The left panel lists databases and tables. In the center is the database and query editor.

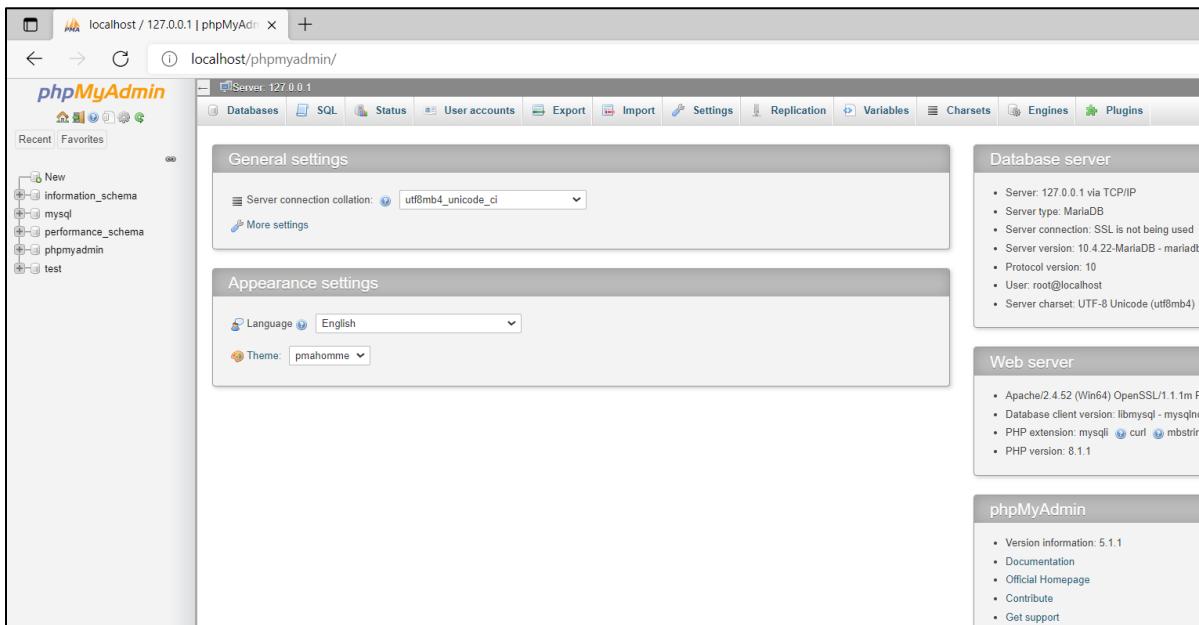
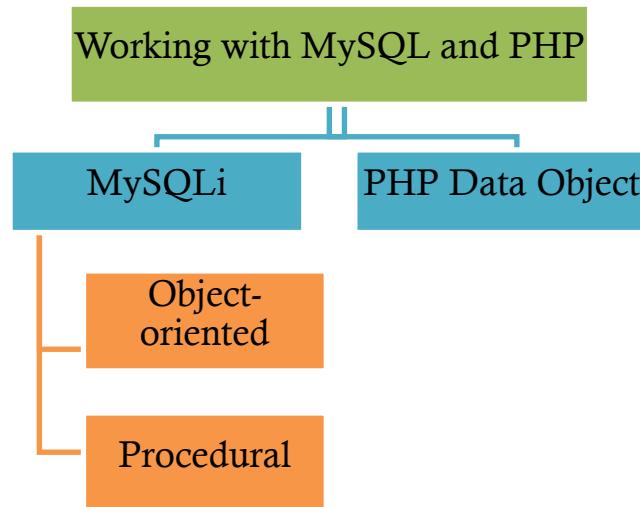


Figure 9.1: phpMyAdmin Page

### 9.3 MySQL Database Connection Through PHP Script

In order to set up a successful connection to MySQL to work with PHP, a user will require a username and password. PHP scripts with MySQL can be written using different approaches. Following are the main ways to work with MySQL and PHP:



- **MySQLi:** This is an updated or an improved extension that is used to access and work with the MySQL database. MySQLi provides two types of interfaces, **object-oriented** (based on the concept of objects) and **procedural** (based on structured programming or procedures).
- **PHP Data Object (PDO):** This is an extension that uses a general database abstraction layer to support MySQL and provide an interface for the user to work with numerous databases. It gives added flexibility in how data is returned. This approach aims to make a common API for all the databases access.

### **PHP `mysqli_connect()`**

Prior to PHP 8, an extension `mysqli_connect()` was used to work with PHP scripts. However, since version 8, this has been deprecated. Now, MySQL Improved (MySQLi) Extension, which is a database driver facilitating as a bridge between MySQL databases and PHP scripts is recommended for use.

The PHP `mysqli_connect()` function is used to connect with the MySQL database. It can return a resource, depending on whether the connection is null or established successfully. Following is the syntax used for the `mysqli_connect()` function:

#### **Syntax:**

```
resource mysqli_connect (username, server, password)
```

where, `username` is the account name of the user that will be used to connect to the database, `server` is the host name or IP address, and `password` is the MySQL password to authenticate the connection.

From PHP 5.0.0 onwards, the `mysqli` extension was brought into use. The user can install the new package on Linux with `sudo apt-get install php-mysql`. For PHP 5.3 onwards, the `mysqli` extension is already enabled on Windows. By default, it makes use of the MySQL Native Driver.

The parameters passed to `mysqli_connect()` function are collectively called as **connection string**. This includes `username`, `server` name, and `password`. Port number and database name can also be supplied in the connection string, if required.

Example of a connection string:

```
"localhost", "root", "root123", "sampledb", 3306, null, "utf8mb4"
```

Here, `localhost` is the server name, `root` and `root123` are username, and password respectively for the MySQL account, `sampledb` is the database name, `3306` is the port number, `null` is the socket number, and `utf8mb4` is the character set encoding.

Code Snippet 1 shows an example of working with PHP `mysqli_connect()`. Here, no specific database is mentioned in the connection string, hence, a connection is established to the database client.

### Code Snippet 1:

```
<?php
$host = "localhost:3306";
$username = "root";
$password = "root";
$phpconn = mysqli_connect($host, $username, $password);
if(! $phpconn) {
    die("Could not connect to database: Please verify the
        privileges" . mysqli_error());
}
echo "Connection to database is successful";
mysqli_close($phpconn);
?>
```

In Code Snippet 1, PHP's built-in function `mysqli_connect` is used to connect with the database. The variables `host`, `username`, and `passwd` are passed.

If the values are correct, connection to the database will be established and an output `Connection to database is successful` is displayed. Otherwise, the `die` function inside the `if` statement is called and an error message `Could not connect to database: Please verify the privileges` will be displayed.

`die()` is a built-in function in PHP. It is used to print message and exit from the current PHP script. It is equivalent to the `exit()` function in PHP. To successfully connect to the database, the user should have account credentials.

Figure 9.2 shows the output for Code Snippet 1.



*Figure 9.2: Output for Code Snippet 1*

## **9.4 Creating and Deleting a MySQL Database**

---

A user-defined MySQL database can be created or deleted using commands within PHP scripts.

### **Creating a MySQL Database Using PHP**

To create a MySQL database, the `CREATE DATABASE` statement is used. Code Snippet 2 shows an example of creating a database called `sampleDB` while working with MySQLi Object-oriented.

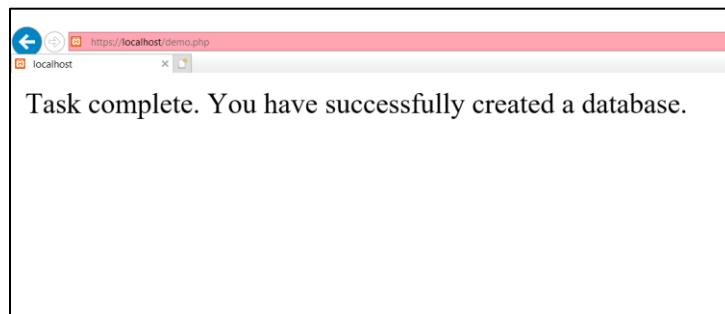
#### **Code Snippet 2:**

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
// Create a connection
$conn = new mysqli($servername, $username, $password);
// Verify whether connection was successful or not
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// Create database
$sql = "CREATE DATABASE sampleDB";
if ($conn->query($sql) === TRUE) {
    echo "Task complete. You have successfully created a
        database";
} else {
    echo "You have an error while creating database: " .
        $conn->error;
}
$conn->close();
?>
```

In Code Snippet 2, an object `conn` is being created. Using this object, PHP's built-in function `query` is called to check for Boolean condition.

In the code line, `$conn->query($sql)` call to MySQLi query function is carried out. `MySQLi::query()` returns Boolean `FALSE` on failure. Boolean `TRUE` will be returned on a successful query without result sets. A `mysqli_result` will be returned upon success with a result set.

If the condition is true, then database `sampleDB` will be created and an output "Task complete. You have successfully created a database." will be displayed. Otherwise, an error message "You have an error while creating database: will be displayed. The database name never allows spaces in the command shown in the code snippet. If the connection is not set, the `die` function is called and `connect_error` will display an output indicating that the connection has failed. Figure 9.3 shows the output for Code Snippet 2.



*Figure 9.3: Output for Code Snippet 2*

Ensure that while creating a new database, the first three arguments should be specified including the username, server name, and password. If the MySQL port is different from the default port of 3306, the user has to give the port number as a parameter, following the database name. For example, the user has to specify port number as 3307 in `$conn = new mysqli_connect("localhost", "username", "password", "database name", 3307)`.

Code Snippet 3 shows an example of using MySQLi procedural.

### **Code Snippet 3:**

```
<?php  
$servername = "localhost";
```

```

$username = "root";
$password = "root";
// Initially you should create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection to return the status of connection
if (!$conn) {
    die("Error in Connection: " . mysqli_connect_error());
}
// SQL command is given to create database
$sql = "CREATE DATABASE sample2DB";
if (mysqli_query($conn, $sql)) {
    echo "Task complete. You have successfully created a
database";
} else {
    echo "Error while creating database: Please check" .
mysqli_error($conn);
}
mysqli_close($conn);
?>

```

In Code Snippet 3, code looks similar to that in Code Snippet 2 is being executed. However, MySQLi procedural method is being used to specify the function parameters. The user can use any of these methods to create database. Here, a database with the name `sample2DB` is created and stored in the variable `sql`.

The `mysqli_query` function is used to execute SQL queries. It can be used to execute different query types such as `insert`, `select`, `update`, and `delete`. In the `if` statement, `mysqli_query` function is used to connect to the database and is passed the `sql` variable, which is a string containing the SQL command to create the database. If it is true, the output `Task complete. You have successfully created a database` is displayed. Otherwise, the output `Error while creating database: Please check` is displayed. The output for Code Snippet 3 is the same as for Code Snippet 2.

## **Deleting a MySQL Database Using PHP**

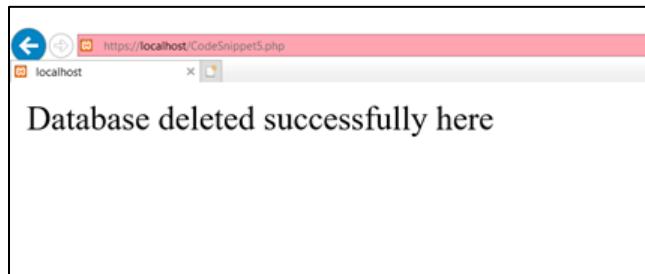
If there is no longer a requirement for a database, it can be deleted permanently. An SQL command can be passed to `mysqli_query` to delete a database.

Code Snippet 4 shows an example of deleting a database.

#### Code Snippet 4:

```
<?php
$dbhost = "localhost";
$username = "root";
$password = "root";
$conn = mysqli_connect($dbhost, $username, $password);
if(! $conn) {
    die("Could not connect: " . mysqli_error());
}
$sql = "DROP DATABASE sampleDB";
$retval = mysqli_query($conn, $sql);
if(! $retval) {
    die("Due to SQL error, deleting database sampleDB is not
        possible: " . mysqli_error());
}
echo " Database deleted successfully here";
mysqli_close($conn);
?>
```

In Code Snippet 4, an existing database is being deleted using the statement `DROP DATABASE sampleDB`. The statement will be executed through `mysqli_query()` function. The return value of the query is assigned to variable `retval`. If it is true, then database has been deleted successfully. Otherwise, it was not deleted. Figure 9.4 shows the output for Code Snippet 4.



*Figure 9.4: Output for Code Snippet 4*

## 9.5 Create/Add Tables in MySQL Database

A database alone is of no use. It must have one or more tables in it with data records. A database table consists of rows and columns along with its unique name.

### MySQL Table Creation Using MySQLi

Use the `CREATE TABLE` statement to create a table in MySQL.

The data type states the type of data the column can contain. Apart from the data type, there are several optional attributes that should be specified for each column. They are as follows:

NOT NULL	DEFAULT VALUE	UNSIGNED	AUTO INCREMENT	PRIMARY KEY
No null values are allowed as it is necessary for each row to have a value for the column.	The default value can be added at a time when no more values are passed.	Unsigned is used for the number types. It limits the data stored, to zero and positive numbers.	Each time a new record is added, the value of the field automatically increases by one.	The main use of the primary key is to figure out rows in the table. The column with the PRIMARY KEY setting is often an ID number and is often used with AUTO_INCREMENT. Each table should have a primary key column and its value must be unique for each record in the table.

Following is the MySQL code used to create a table:

```
CREATE TABLE Passengers (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
```

Here, the primary key is the `id` column.

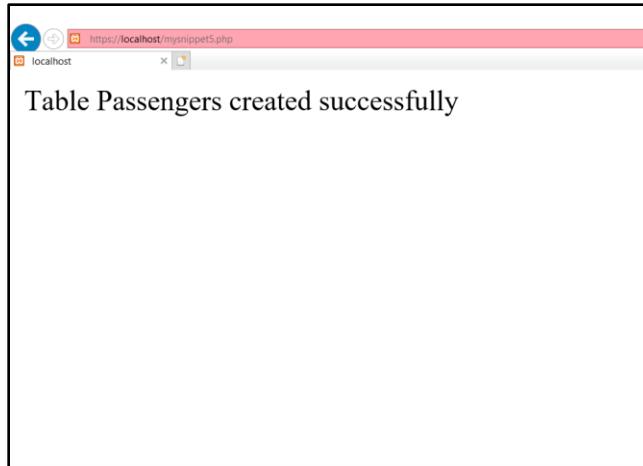
Code Snippet 5 shows how to create a table in PHP (MySQLi Object-oriented). It shows how to create a table named `Passengers` with five different columns named `id`, `firstname`, `lastname`, `email`, and `reg_date`.

## Code Snippet 5:

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Next, you should check the connection
if ($conn->connect_error) {
    die("Connection failed: Recheck the connection details" . 
        $conn->connect_error);
}
// Write an SQL command to create table and store in a variable
$sql = "CREATE TABLE Passengers (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
    CURRENT_TIMESTAMP
)";
if ($conn->query($sql) === TRUE) {
    echo "Table Passengers created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}
$conn->close();
?>
```

In Code Snippet 5, a table named `Passengers` is being created using the MySQL query `CREATE TABLE Passengers`. The required table details are added in it.

When the code in Code Snippet 5 is executed, an output Table for passengers is created is displayed. MySQLi (object-oriented) way is being used to write the code by creating an object of `mysqli` in the code snippet. Figure 9.5 shows the output for Code Snippet 5.



*Figure 9.5: Output for Code Snippet 5*

## **9.6 Data Insertion and Data Retrieval into/from MySQL Database**

---

Once the table and a database have been created in MySQL, users can insert or add data into, and retrieve data from a MySQL database, using MySQLi.

### **Database Queries**

A query can be defined as a request and also a question. Following is an example of a standard SQL query:

```
SELECT LastName FROM Employees
```

This query selects all the data in the `Lastname` column from a table named `Employees`.

#### **9.6.1 Inserting Data into MySQL Database**

To add new records into the MySQL table, the `INSERT INTO` statement can be used. Following is the syntax for the `INSERT INTO` statement:

#### **Syntax:**

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...)
```

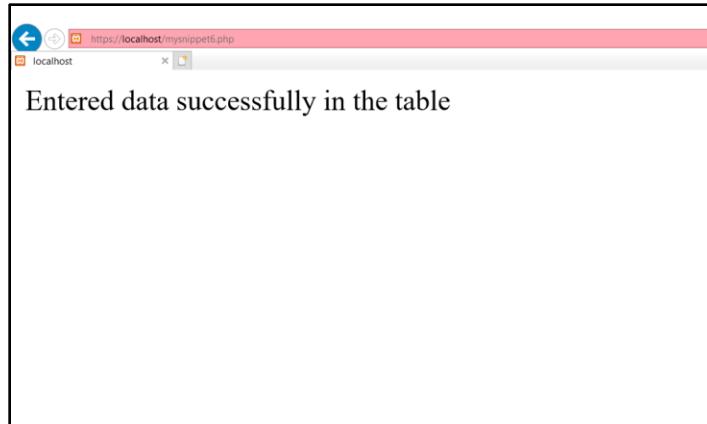
Following are some syntax rules:

- The SQL query should be quoted in PHP.
- The string values in the SQL query must be quoted.
- Numeric values should be quoted.
- The word NULL should not be quoted.

### Code Snippet 6:

```
<?php
$dbhost = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
$conn = mysqli_connect($dbhost, $username, $password);
if(! $conn) {
die("Issue in the connection. Please check the details again: "
    . mysqli_error());
}
$sql = "INSERT INTO Passengers (id,firstname,
        lastname,email,reg_date) ". "VALUES (101,'Derek',
        'Houston', 'derek@sample.com', NOW())";
mysqli_select_db($conn,"sample2DB");
$retval = mysqli_query($conn,$sql);
if(! $retval) {
die("Could not enter data: " . mysqli_error());
}
echo "Entered data successfully in the table\n";
mysqli_close($conn);
?>
```

In Code Snippet 6, data is being inserted in to the passengers table using `INSERT` statement. The database containing the table is selected first and then, the query is executed using `mysqli_query()` function. If the query is executed, an output `Entered data successfully in the table` is displayed. Otherwise, an output `Could not enter data:` will be displayed. Figure 9.6 shows the output for Code Snippet 6.



**Figure 9.6: Output for Code Snippet 6**

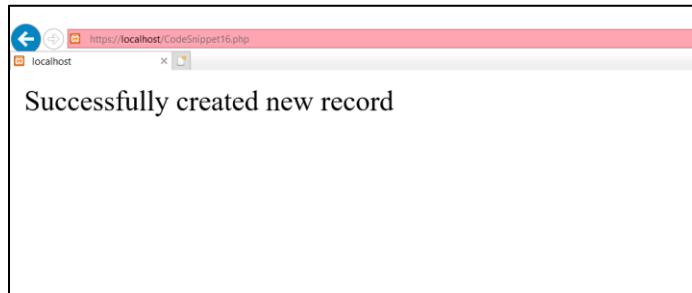
In a real-world application, input values will be taken using HTML forms, instead of being hard-coded. These values will be captured through a PHP script and will be added to MySQL tables. The rows inserted into the table can be checked by logging in to PhpMyAdmin and examining the database tables.

Code Snippet 7 shows how to add a new record to the table (MySQLi Object-oriented).

### **Code Snippet 7:**

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO Passengers (firstname, lastname,
    email)VALUES ('Hugh', 'Sheperd', 'hugh@sample.com')";
if ($conn->query($sql) === TRUE) {
    echo "Successfully created new record";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

In Code Snippet 7, data is being inserted into a table using MySQL query "INSERT INTO Passengers (firstname, lastname, email) VALUES ('Hugh', 'Sheperd', 'hugh@sample.com')". Here, values for the first name, last name, and email ID are being inserted. When the code in Code Snippet 7 is executed, an output Successfully created new record is displayed. Otherwise, an error message will be displayed. Figure 9.7 shows the output for Code Snippet 7.



*Figure 9.7: Output for Code Snippet 7*

## 9.6.2 Multiple Records Insertion

When user has multiple SQL statements to be executed, he/she can use the `mysqli_multi_query()` function. Code Snippet 8 shows an example of adding three new records to the `Passengers` table using MySQLi Object-oriented.

### Code Snippet 8:

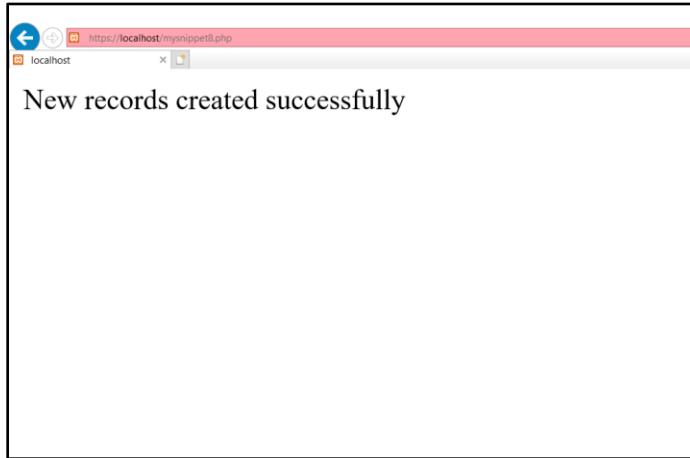
```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO Passengers (firstname, lastname, email)
        VALUES ('Luke', 'Thompson', 'Luke@sample.com');";
$sql .= "INSERT INTO Passengers (firstname, lastname, email)
        VALUES ('Ben', 'Smith', 'ben@sample.com');";
$sql .= "INSERT INTO Passengers (firstname, lastname, email)
        VALUES ('Ruby', 'Stokes', 'ruby@sample.com')";
if ($conn->multi_query($sql) === TRUE) {
```

```

echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>

```

In Code Snippet 8, multiple records are being inserted in to the table Passengers using `multi_query()` function. Each SQL statement must be separated by a semicolon. If the query is executed, an output New records created successfully is displayed. Otherwise, an error message will be displayed. Figure 9.8 shows the output for Code Snippet 8.



*Figure 9.8: Output for Code Snippet 8*

### 9.6.3 Getting Last Inserted ID

If an `INSERT` or `UPDATE` on a table is performed along with an `AUTO_INCREMENT` field, the ID of the last inserted or updated record can be obtained immediately.

Code Snippet 9 shows how to retrieve the ID of the last inserted record (MySQLi Object-oriented).

#### Code Snippet 9:

```

<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
// Create connection

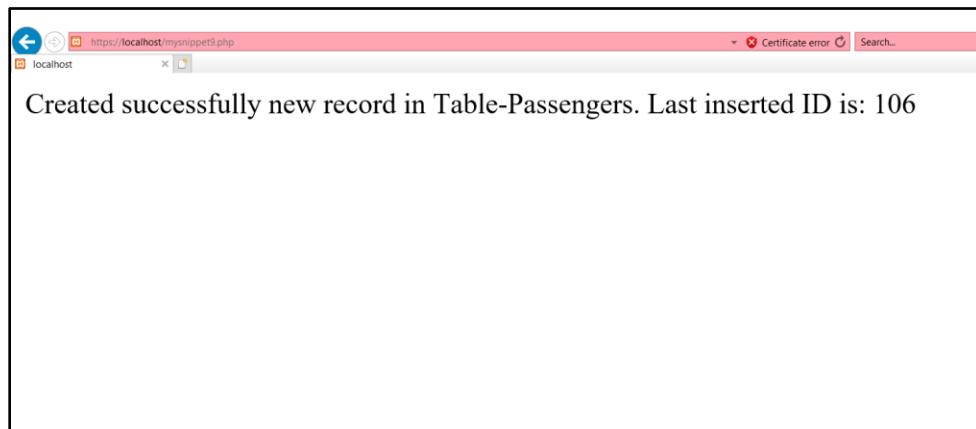
```

```

$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO Passengers (firstname, lastname, email)
        VALUES ('Mark', 'Mortan', 'Mark@sample.com')";
if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "Created successfully new record in Table-Passengers.
          Last inserted ID is: " . $last_id;
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>

```

In Code Snippet 9, code has been added to retrieve and display the ID of the last inserted record. An object of `mysqli` has been created and `insert_id` has been referred to. If the code in Code Snippet 9 is executed, an output `Created successfully new record in Table-Passengers. Last inserted ID is: 3` is displayed. Figure 9.9 shows the output for Code Snippet 9.



*Figure 9.9: Output for Code Snippet 9*

#### 9.6.4 Retrieving Data from MySQL Database

PHP enables users to retrieve data from a MySQL database. The `SELECT` statement can be used within a PHP script to select and fetch data from one or more tables.

Following is the syntax for using SELECT statement:

### Syntax:

```
SELECT column_name(s) FROM table_name
SELECT * FROM table_name
```

Code Snippet 10 shows an example for fetching records from the Passengers table. The function `mysqli_fetch_array()` is used in this code snippet. In this function, the row is returned as a numeric array or an associative array. In some cases, the row is returned as both numeric array and associative array. In case there are no more rows, MySQL returns it as false.

### Code Snippet 10:

```
<?php
$dbhost = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
$conn = mysqli_connect($dbhost, $username, $password);
if (! $conn) {
    die("Something went wrong!!Could not connect: " .
    mysqli_error());
}
$sql = "SELECT id,firstname, lastname,email FROM
    passengers";
mysqli_select_db($conn,"sample2DB");
$retval = mysqli_query($conn,$sql);
if(! $retval) {
    die("Error encountered while fetching the data: " .
    mysqli_error());
}
while($row = mysqli_fetch_array($retval)) {
    echo "Passenger ID. :{$row["id"]} <br> ".
        "First Name : {$row["firstname"]} <br> ".
        "Last Name: {$row["lastname"]} <br> ".
        "Email: {$row["email"]} <br> ".
        "-----<br>";
}
echo "Data successfully retrieved\n";
mysqli_close($conn);
?>
```

In Code Snippet 10, all passenger details are being fetched from the passengers table using an `SELECT` query. If the code in Code Snippet 10 is executed and connection is established with the database `sample2DB`, then passenger details

will be displayed in the output along with the message Data successfully retrieved.

Here, one argument namely, \$retval, is passed to a function `mysqli_fetch_array()`. The `mysqli_fetch_array()` function is used to display all the records from the passengers table. It can fetch the next row of a result set as an associative, a numeric array, or both. Note that \$retval is a `mysqli_result` object returned by `mysqli_query()`, `mysqli_store_result()`, `mysqli_use_result()` or `mysqli_stmt_get_result()`. Hence, this is passed as parameter to the `mysqli_fetch_array()` function, in order to iterate the records of the table.

Figure 9.10 shows part of the output for Code Snippet 10.

The figure displays two screenshots of a web browser window. The left screenshot shows the source code of a PHP file named mysnippet10.php. It contains three entries, each representing a passenger record with fields: Passenger ID, First Name, Last Name, and Email. The right screenshot shows the execution result of the code. It lists the same three passenger records in a structured format, separated by horizontal dashed lines. Below the records, a message 'Data successfully retrieved' is displayed. The browser address bar at the top of both screenshots shows the URL as https://localhost/mysnippet10.php.

Passenger ID.	First Name	Last Name	Email
:101	Derek	Houston	derek@sample.com
:102	Hugh	Sheperd	hugh@sample.com
:103	Luke	Thompson	

Passenger ID. :104  
First Name : Ben  
Last Name: Smith  
Email: ben@sample.com

Passenger ID. :105  
First Name : Ruby  
Last Name: Stokes  
Email: ruby@sample.com

Passenger ID. :106  
First Name : Mark  
Last Name: Mortan  
Email: Mark@sample.com

Data successfully retrieved

**Figure 9.10: Output for Code Snippet 10**

PHP provides another function `mysqli_fetch_assoc()`, which also returns the specific row or rows as an associative array. Code Snippet 11 shows example to display all the records from the Passengers table using the `mysqli_fetch_assoc()` function.

## Code Snippet 11:

```
<?php
    $dbhost = "localhost";
    $username = "root";
    $password = "root";
    $dbname = "sample2DB";
    $conn = mysqli_connect($dbhost, $username, $password);
    if(! $conn) {
        die("Could not connect: " . mysqli_error());
    }
    $sql = "SELECT id,firstname, lastname, email FROM passengers";
    mysqli_select_db($conn, "sample2DB");
    $retval = mysqli_query($conn, $sql);
    if(! $retval) {
        die("Could not get data: " . mysqli_error());
    }
    while($row = mysqli_fetch_assoc($retval)) {
        echo "Passenger ID : {$row['id']} <br> ".
            "First Name : {$row['firstname']} <br> ".
            "Last Name: {$row['lastname']} <br> ".
            "Email: {$row['email']} <br> ".
            "-----<br>";
    }
    echo "Data successfully retrieved\n";
    mysqli_close($conn);
?>
```

In Code Snippet 11, passenger details are being fetched from the table using an `SELECT` query. If the query is executed and connection is established with the database `sample2DB`, then passenger details will be displayed in the output along with the message `Data successfully retrieved`. The records from the `passengers` table are displayed using `mysqli_fetch_assoc()` function. Figure 9.11 shows the output for Code Snippet 11.

Passenger ID. :101  
First Name : Derek  
Last Name: Houston  
Email: derek@sample.com

Passenger ID. :102  
First Name : Hugh  
Last Name: Sheperd  
Email: hugh@sample.com

Passenger ID. :103  
First Name : Luke  
Last Name: Thompson

Passenger ID. :104  
First Name : Ben  
Last Name: Smith  
Email: ben@sample.com

Passenger ID. :105  
First Name : Ruby  
Last Name: Stokes  
Email: ruby@sample.com

Passenger ID. :106  
First Name : Mark  
Last Name: Mortan  
Email: Mark@sample.com

Data successfully retrieved

**Figure 9.11: Output for Code Snippet 11**

Constant `MYSQLI_NUM` can also be used as the second argument to the `mysqli_fetch_array()`. `MYSQLI_NUM` makes the function behave similar to the `mysqli_fetch_row()` function, fetching a numeric array. Code Snippet 12 shows an example where all records from the `passengers` table are displayed using the `MYSQLI_NUM` argument.

### **Code Snippet 12:**

```
<?php
$dbhost = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
$conn = mysqli_connect($dbhost, $username, $password);
if(! $conn) {
    die("Encountered an error, could not connect: " .
        mysqli_error());
}
$sql = "SELECT id,firstname, lastname,email FROM passengers";
mysqli_select_db($conn,"sample2DB");
$retval = mysqli_query($conn, $sql);
if(! $retval) {
    die("Error while fetching data: " . mysqli_error());
}
```

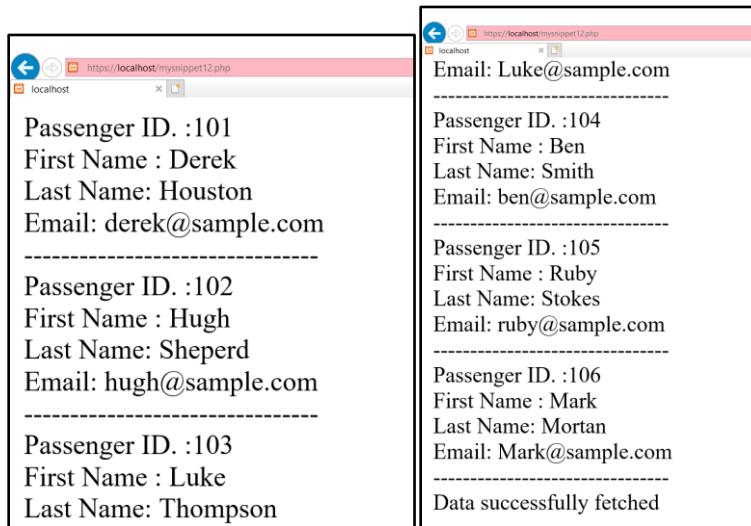
```

while($row = mysqli_fetch_array($retval, MYSQLI_NUM)) {
    echo "Passenger ID. :{$row[0]} <br> ".
        "First Name: {$row[1]} <br> ".
        "Last Name: {$row[2]} <br> ".
        "Email: {$row[3]} <br> ".
        "-----<br>";
}
echo "Data successfully fetched\n";
mysqli_close($conn);
?>

```

In Code Snippet 12, passenger details are being fetched from `passengers` table. If the query is executed and connection is established with the database `sampleDB`, then passenger details will be displayed in the output along with the message `Data successfully fetched`. All the records from the `Passengers` table are displayed using the `mysqli_fetch_assoc()` function. In Code Snippet 12, an additional argument `MYSQLI_NUM` is used with the function `mysqli_fetch_array()`. Hence, this time the array data are retrieved using numeric indexes rather than column names.

Figure 9.12 shows the output for Code Snippet 12.



**Figure 9.12: Output for Code Snippet 12**

Code Snippets 11 and 12 will produce the same result. This is because the data in both the code snippets data is being fetched from the same table and the same number of rows of the same database.

## Data Selection with MySQLi

Code Snippet 13 shows selecting the `firstname`, `lastname`, and `id` columns from the `Passengers` table and showcasing them on to the page. This uses MySQLi Object-oriented.

### Code Snippet 13:

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
// Let us create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection status
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM Passengers";
$result = $conn->query($sql);
//The given code iterates through all the records in table
if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]." ".$row["lastname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>
```

In Code Snippet 13, the data according to the number of rows present in the table, is being fetched. Here, the SQL query is set up to select the `firstname`, `lastname`, and `id` columns from the `Passengers` table. The code in the next line runs the query and places the data into the variable known as `$result`.

Then, the function `num_rows()` checks if there are more than zero rows returned. In case more than two rows are returned, all are put in an associative array by the function `fetch_assoc()` that can be looped through. The `while()` loop loops through the result set and outputs the data from the `firstname`, `lastname`,

and `id` columns. Figure 9.13 shows the output for Code Snippet 13. The `Name` column in the displayed table combines first name and last name.

ID	Name
101	Derek Houston
102	Hugh Sheperd
103	Luke Thompson
104	Ben Smith
105	Ruby Stokes
106	Mark Mortan

*Figure 9.13: Output for Code Snippet 13*

## **9.7 Updating Data into a MySQL Table**

---

To update data into MySQL tables, user can execute the `UPDATE` statement through PHP function `mysqli_query`.

Code Snippet 14 shows an example of the update operation.

### **Code Snippet 14:**

```
<html>
  <head>
    <title>Update a Record in mysqli Database</title>
  </head>
  <body>
    <?php
      if(isset($_POST["update"])) {
        $dbhost = "localhost";
        $username = "root";
        $password = "root";
        $conn = mysqli_connect($dbhost, $username,
                               $password);
        if(! $conn) {
          die("Could not connect: " . mysqli_error());
        }
        $pasngr_id = filter_input(INPUT_POST, 'id');
        $pemail = filter input(INPUT_POST, 'email');
```

```

$sql = "UPDATE passengers SET email ='" . $pemail
. "' WHERE id = $pasngr_id " ;
mysqli_select_db($conn,"sample2DB");
$retval = mysqli_query($conn,$sql);
if(! $retval) {
    die("Could not update data: " .
        mysqli_error());
}
echo "Updated data successfully\n";
mysqli_close($conn);
}

?>
<form method = "post" action = "<?php $_PHP_SELF
?>">
<table width = "400" border =" 0" cellspacing =
"1" cellpadding = "2">
<tr>
<td width = "100"> Passenger ID:</td>
<td>
<input name = "passenger_id" type = "text"
       id = "id"></td>
</tr>
<tr>
<td width = "100"> Email:</td>
<td><input name = "email" type = "text"
       id = "email"></td>
</tr>
<tr>
<td width = "100"> </td>
<td> </td>
</tr>
<tr>
<td width = "100"> </td>
<td>
<input name = "update" type = "submit" id =
"update" value = "Update">
       </td>
</tr>
</table>
</form>

</body>
</html>

```

Code Snippet 14 shows the code to update records into Passengers table based on given input through a form. To update records in a table, the specific record has to be located using some criteria with the WHERE conditional clause.

In this example, a passenger ID has to be provided as criteria to update the email, because passenger IDs are unique and only a specific record will be selected for the operation. Hence, Passenger email is updated using `UPDATE MySQL` query with passenger ID as the criteria in the `WHERE` clause. When the **Update** button is clicked, a message `Updated data successfully` is displayed in the output. Figure 9.14 shows the form generated through Code Snippet 14.

A screenshot of a web browser window titled "https://localhost/updated4.php". The page contains a form with two input fields: "Passenger ID:" and "Email:". Below the fields is a "Update" button. The browser's address bar shows the URL "https://localhost/updated4.php".

Passenger ID:

Email:

**Figure 9.14: Output for Code Snippet 14**

After entering data and clicking Update, when the message is displayed indicating data is updated successfully, go to the table in phpMyAdmin to verify the updated record.

## 9.8 Using PHP to Back Up MySQL Database

Regularly backing up a database is an ideal practice for users. In order to backup MySQL database, there are three options for users to choose from. They are as follows:

- Using SQL Command through PHP
- Using MySQL binary `mysqldump` through PHP
- Using `phpMyAdmin` user interface

### Using SQL Command through PHP

Using the SQL `SELECT` command to archive a backup is a great decision. In order to take an entire database dump, there will be a requirement to write a separate query for different tables. Each of these tables will be stored in a unique text file.

Code Snippet 15 shows an example of using `SELECT INTO OUTFILE` query for creating back up of a table.

### Code Snippet 15:

```
<?php
$dbhost = "localhost";
$username = "root";
$password = "root";
$conn = mysqli_connect($dbhost, $username, $password);
if(! $conn) {
    die("Error while connecting" . mysqli_error());
}
$table_name = "passengers";
$backup_file = "E:\\tmp\\passengers.sql";
$sql = "SELECT * INTO OUTFILE '$backup_file' FROM
        $table_name";
mysqli_select_db($conn,"sample2DB");
$retval = mysqli_query($conn,$sql);
if(! $retval) {
    die("Error while attempting data backup: Sorry this
        operation is not successful " . mysqli_error());
}
echo "Backed up data successfully\n";
mysqli_close($conn);
?>
```

In Code Snippet 15, the passengers table from database `sample2DB` is being backed up using a MySQL query. A local backup file name with extension `.sql` is specified in a string and then, substituted into the SQL query.

If the query is executed, an output `Backed up data successfully` is displayed. Otherwise, if an error occurs, the error message `Error while attempting data backup: Sorry this operation is not successful` is displayed. Figure 9.15 shows the output for Code Snippet 15.



*Figure 9.15: Output for Code Snippet 15*

In case data that has already been backed up must be restored, run the LOAD DATA INFILE query. Code Snippet 16 shows the same.

### Code Snippet 16:

```
<?php
$dbhost = "localhost";
$username = "root";
$password = "root";
$conn = mysqli_connect($dbhost, $username, $password);
if(! $conn) {
    die("Could not connect: " . mysqli_error());
}
$table_name = "passengers";
$backup_file = "E:\\tmp\\passengers.sql";
$sql = "LOAD DATA INFILE '$backup_file' INTO TABLE
        $table_name";
mysqli_select_db($conn, "sample2DB");
$retval = mysqli_query($conn, $sql);
if(! $retval) {
    die("Encountered an error, Could not load data: " .
        mysqli_error());
}
echo "Loaded data successfully\n";
mysqli_close($conn);
?>
```

In Code Snippet 16, code is written to restore the backed up data from database which is currently present in the local .sql file. The MySQL query LOAD DATA INFILE '\$backup\_file' INTO TABLE \$table\_name is used and the local file name is substituted.

Output Loaded data successfully is displayed upon successful execution of the query. Figure 9.16 shows the output for Code Snippet 16.

For the code to run successfully, the passengers table must be empty otherwise the backup data restoration will attempt to insert duplicate records into the table which is not permitted. Ensure table Passengers is empty before running Code Snippet 16.



**Figure 9.16: Output for Code Snippet 16**

## Using MySQL Binary mysqldump through PHP

MySQL provides a command-line utility `mysqldump`, to perform a database backup. Using this binary, the user can implement a complete database dump in a single command. Code Snippet 17 shows an example to perform a full database dump.

### Code Snippet 17:

```
<?php
$dbhost = "localhost:3036";
$username = "root";
$password = "root";
$dbname = "sample2DB";
$backup_file = $dbname. date("Y-m-d-H-i-s") . '.gz';
$command = "mysqldump --opt -h $dbhost -u $username -p
    $password ". "lct2DB | gzip > $backup_file";
system($command);
echo "Data backed up successfully";
?>
```

In Code Snippet 17, `mysqldump` to perform a database backup in a single command. Here, host, user, and password of database `sample2DB` are being passed and the data in the database is backed up to a file. `system()` is a PHP function to execute external commands. Since `mysqldump` is an external command-line utility typically run from the command line, to invoke it via PHP, user must call `system()`. The backup file constructed in the string `backup_file` will be substituted in the command string which is then passed to `system()`.

When the code in Code Snippet 17 is executed, an output `Data backed up successfully` is displayed. Figure 9.17 shows the output for Code Snippet 17.



*Figure 9.17: Output for Code Snippet 17*

### Using phpMyAdmin User Interface

If the phpMyAdmin user interface is available, then it is easy to take backup of the database. To do this, click the **export** link on the phpMyAdmin main page. From the options available, choose any of the database options that user wishes to backup, he/she should go through the different SQL options, and enter the backup file name.

## **9.9 Selecting and Filtering Data from a MySQL Table**

---

Adding filters to a query ensures that only the required data is displayed in the query result. Sorting helps arrange the rows in the query result in an order which gives meaning to the data that will be used. The `LIMIT` clause makes it easy to code multi page results or pagination with SQL and is very useful while working on large tables.

The three clauses used for these purposes are as follows:

- WHERE
- ORDER BY
- LIMIT

### **9.9.1 WHERE Clause**

The `WHERE` clause is used in MySQL query to filter records and hold a primary key in a table. When a particular field of data is to be fetched, then another field name with value of a table is provided. This ensures that the database can search and find which row element it has to fetch or select. Only records that meet specific conditions are extracted by the `WHERE` clause.

Following is the syntax used for WHERE clause:

### Syntax:

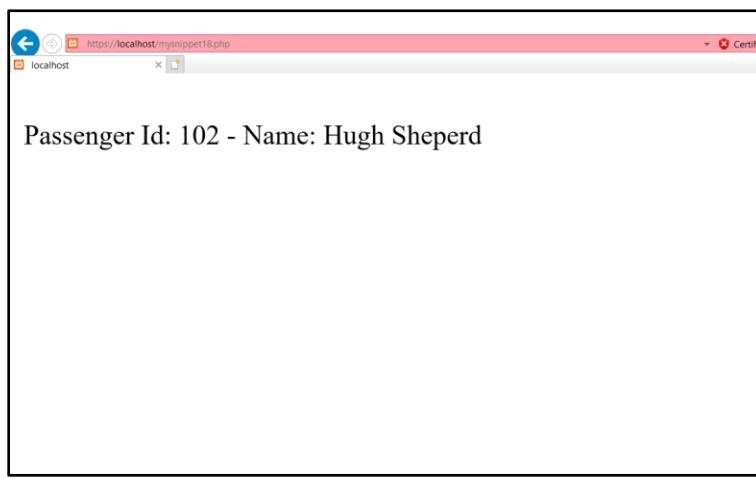
```
SELECT column_name(s) FROM table_name WHERE column_name operator  
value
```

Code Snippet 18 shows an example that selects the `firstname`, `lastname`, and `id` columns from the `Passengers` table and displays it on the page.

### Code Snippet 18:

```
<html>  
<body>  
<?php  
$servername = "localhost";  
$username = "root";  
$password = "root";  
$dbname = "sample2DB";  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection status  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}  
$sql = "SELECT id, firstname, lastname FROM Passengers WHERE  
lastname='Shepard'";  
$result = $conn->query($sql);  
//The given code checks for the number of rows retrieved  
if ($result->num_rows > 0) {  
    // output data of each row  
    while($row = $result->fetch_assoc()) {  
        echo "<br> Passenger Id: ". $row["id"] . " - Name: ".  
            $row["firstname"] . " " . $row["lastname"] . "<br>";  
    }  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>  
</body>  
</html>
```

In Code Snippet 18, only one row of a table is being fetched using the WHERE clause. The first step is setting up the SQL query that selects the `firstname`, `lastname`, and `id` columns from the `Passengers` table in which the last name is given as `Morse`. The next line of the code runs the query and places the data into the variable termed as `$result`. Then, the function `num_rows()` checks if there are more than zero rows returned. In case more than zero rows are returned, results are placed into the associative array by the function `fetch_assoc()` that can be looped through. Figure 9.18 shows the output for Code Snippet 18.



*Figure 9.18: Output for Code Snippet 18*

### 9.9.2 ORDER BY Clause

To shorten the extracted results either in the ascending or descending order, `ORDER BY` clause can be used. This clause can sort the record only in ascending order by default. However, if user wants to sort results into descending order, the `DESC` keyword can be used.

Following is the syntax to use the `ORDER BY` clause:

#### Syntax:

```
SELECT column_name(s) FROM table_name ORDER BY column_name(s)  
ASC | DESC
```

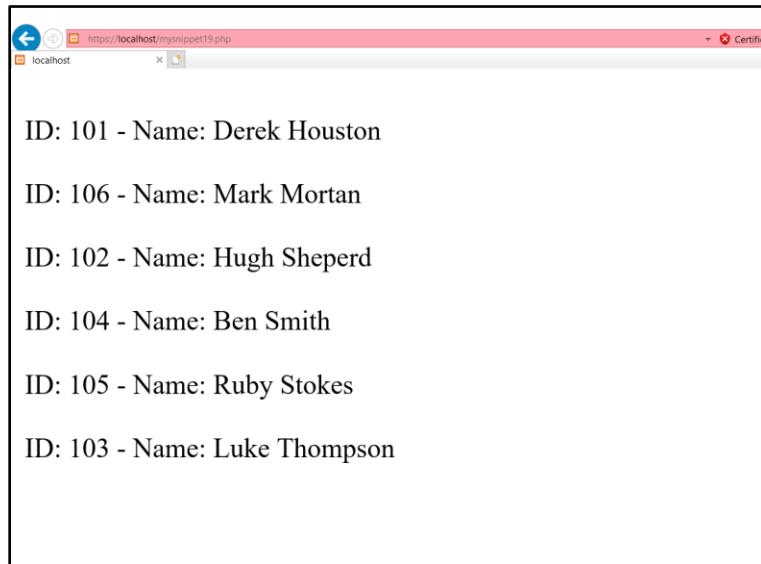
Code Snippet 19 shows an example to select the `firstname`, `lastname`, and `id` columns from the `Passengers` table using MySQLi.

## Code Snippet 19:

```
<html>
<body>
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "sample2DB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM Passengers ORDER BY
lastname";
$result = $conn->query($sql);
//The given if condition checks if any rows are fetched or not
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<br> ID: ". $row["id"]. " - Name: ". $row["firstname"] .
        " " . $row["lastname"] . "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
</body>
</html>
```

In Code Snippet 19, the `ORDER BY` clause of MYSQL is used to display the records in ascending order of rows that is 1, 2, and 3. The first step is setting up the SQL query, which selects the `firstname`, `lastname`, and `id` columns from the `Passengers` table. Through the `lastname` column, the record will be ordered. The next line of the code develops the data into the variable known as the `$result` and runs the query. Then, the function `num_rows()` checks if there are more than zero rows returned.

In case returned rows are more than two, all the acquired results are placed into an associative array by the function `fetch_assoc()`. The `while()` loop loops through the result set and outputs the data from the `firstname`, `lastname`, and `id` columns. Figure 9.19 shows the output for Code Snippet 19.



**Figure 9.19: Output for Code Snippet 19**

### 9.9.3 LIMIT Clause

When a large number of records are returned, it can impact the overall performance. The `LIMIT` clause is used to limit returned rows in a MySQL query to a specific number. The `LIMIT` clause is extremely useful while working with large tables, since it is easier to code multiple page results simultaneously.

Assume that the user wants to select only 30 from a table called `Orders` that has one hundred records. The SQL query will be written as follows:

```
$sql = "SELECT * FROM Orders LIMIT 30";
```

When this SQL query is run, it will return the first 30 records.

Further, assume that the user wants to select records 16–25 (inclusive). In this case, use `OFFSET`. Following SQL query specifies that it should return only 10 records, starting from record 16 (`OFFSET 15`).

```
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";
```

A shorter syntax can also be used to achieve the same result.

```
$sql = "SELECT * FROM Orders LIMIT 15, 10";
```

Notice that the numbers are reversed when a comma is used.

## 9.10 Summary

- MySQL is an open-source relational database system and is often used with Web applications.
- PHP `mysqli_connect()` function is used to connect with the MySQL database.
- Using PHP, a MySQL database can be created and deleted.
- The `CREATE TABLE` statement is used to create a table in MySQL and can be used in a PHP script.
- Data can be both inserted into and retrieved from a MySQL database through PHP scripts.
- The SQL `UPDATE` statement through PHP function `mysqli_query` is used to update data in to a MySQL table.
- Data can be selected from a MySQL database using the `SELECT` statement.
- The `WHERE` clause is used to filter data based on specific criteria.
- The `ORDER BY` clause is used to display extracted results either in ascending or descending order.
- The `LIMIT` clause is used to specify the numbers of returning records.
- MySQL database can be backed up in three ways, namely, using SQL Command through PHP, using MySQL binary `mysqldump` through PHP, and using phpMyAdmin user interface.

## 9.11 Test Your Knowledge



1. Which of the following database management systems lack a native PHP extension?
  - a. MySQL
  - b. PostgreSQL
  - c. Microsoft SQL Server
  - d. None of these
  
2. Which of the following is used to access a MySQL database in PHP?
  - a. mysqli-connect() function
  - b. mysqlconnect() function
  - c. mysqli\_connect() function
  - d. sql\_connect() function
  
3. Which of the following arrays refers to the array with strings as an index?
  - a. Binary array
  - b. Multi-dimensional array
  - c. Associative array
  - d. Indexed array
  
4. Which of the following mysqli functions frees the memory associated with a result?
  - a. mysqli.real\_query.php
  - b. mysqli.multi\_query.php
  - c. mysqli.prepare.php
  - d. mysqli-result.free.php
  
5. Which of the following client utilities helps create logical database backups?
  - a. mysqldump
  - b. mysqlcheck
  - c. mysqlirepair
  - d. mysqlanalyze

## **Answers to Test Your Knowledge**

---

1. None of these
2. mysqli\_connect() function
3. Associative array
4. mysqli-result.free.php
5. mysqlidump

## 9.12 Try it Yourself

1. Write a program to connect to a MySQL database named Test using PHP, where username and password are "root" and "root123" respectively.
2. Write a PHP script to create a table named Customers in this database. The structure of the table is:  
customer\_name(Varchar)  
customer\_address(Varchar)  
amount(float)  
purchase\_date(Date)
3. Write a PHP script to add 10 records into this table.
4. Write a PHP script to display first five records from this table in descending order of amount.

# Session 10

## Advanced Features of PHP



### Learning Objectives

*In this session, students will learn to:*

- Identify the process to set up a local server for sending email
- Describe how to send an email
- Outline the process of sending a plain text email
- Explain how to send an HTML email
- Elaborate the process of sending an attachment with an email
- Describe user authentication

This session begins with explaining how to send an email through a PHP application. The session provides an overview of how to send a plain text email as well as an HTML email. Then, the session outlines the process of sending an attachment with an email. Further, the session explains how to create a user authentication system in PHP.

### **10.1 Set up a Local Server for Email**

---

Simple Mail Transfer Protocol (SMTP) is a set of communication guidelines (protocol) that allow software applications to transmit an email over the Internet. An SMTP server enables sending messages to other computer users based on email addresses. The key objective of SMTP is used to set up communication rules between servers.

Users can create PHP scripts to send and receive emails using SMTP. To verify and test the functionality of these scripts, users will require an SMTP server. One of the easiest and affordable ways to use a local SMTP server is Papercut SMTP.

Papercut SMTP serves as a local SMTP server designed to receive emails locally. Users can check verified emails directly in the Papercut inboxes. Beginners can use this lightweight SMTP server as a means of testing how their mails are received. They can view every part of an email, that is, headers, body, attachments, and even

the raw coded data. Setting up and configuring this software is easy and absolutely free. The software must be downloaded from:

<https://github.com/ChangemakerStudios/Papercut-SMTP>

After downloading the zip file, extract to any folder, and then, open Papercut.exe. It is now ready and configured for use. Users do not have to be concerned about their test emails going to spam. Forward button is available for users to forward to specific email addresses to test further.

## 10.1 Sending an Email

---

In order to send an email, a user must ensure that the `php.ini` file includes the corresponding configurations for PHP. The file must also have precise details on how emails are managed by the system. Since user is using XAMPP, the file can be opened by right-clicking Config option in XAMPP Control Panel and then, selecting `php.ini`. This will automatically open the file in Notepad. Alternatively, if user wants to open the file through File Explorer, navigate to the folder where php is installed (for example, C:\xampp\php) locate the `php.ini` file, and then, open it using any text editor.

Once the file is open, look for `[mail function]` in the content.

The mail function section will have following directives:

- SMTP, which describes the user's email server address
- `sendmail_from`, which describes the user's email address
- `smtp_port=25`

An example of configuring SMTP in Windows is as follows:

```
[mail function] ; For Win32 only.  
SMTP = smtp.inovastudio.net ; For win32 only  
smtp_port=25  
sendmail_from = webmaster@inovastudio.com
```

However, since Papercut will be used here, all the user has to do is:

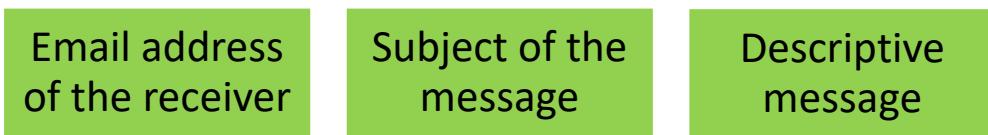
```
SMTP=localhost  
smtp_port=25
```

Save the file and launch Papercut SMTP server. Retain the default options in Papercut as is.

## 10.2 Sending a Plain Text Email

---

To send an email, PHP utilizes `mail()` function. Following are the three mandatory arguments that must be provided to this function:



Syntax of `mail()` function is as follows and this includes the optional argument too:

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

Table 10.1 explains each argument in this function:

Argument	Represents	Description	Mandatory/ Optional
<code>to</code>	Email address of the receiver	This argument includes the details of the email recipient. Ensure to use a comma separated list if there are two or more recipients.	Mandatory
<code>subject</code>	Subject of the message	This argument includes the subject of the email. Ensure to not include any newline characters here.	Mandatory
<code>message</code>	Descriptive message	This argument includes the descriptive message to be sent to the recipient. Ensure to separate each line with a Line Feed (LF) (\n) and it must not be more than 70 characters.	Mandatory
<code>headers</code>	<code>headers</code>	This is an optional argument that includes extra headers. Ensure to separate each header with a Carriage Return Line Feed (CRLF) (\r\n). Example: From, Cc, and Bcc.	Optional
<code>parameters</code>	<code>parameters</code>	This is also an optional argument that includes parameters, if any, to be passed to the send mail program.	Optional

*Table 10.1: mail() Function Arguments*

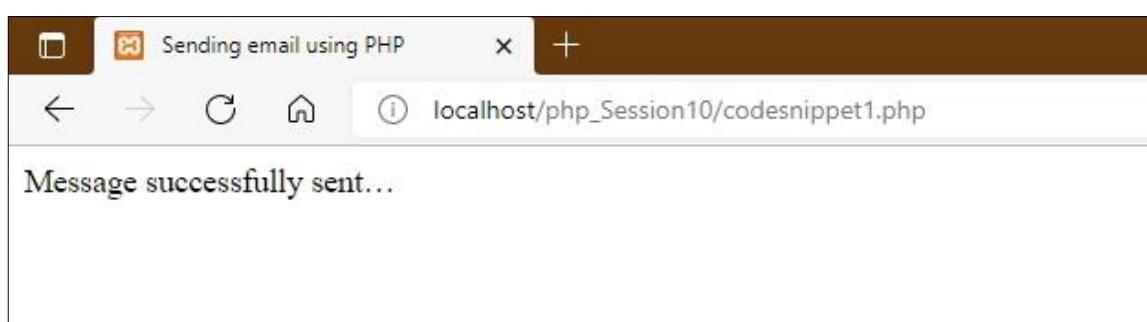
When `mail()` function is called, PHP tries to send out the email. If it is successfully sent, PHP returns `true`, otherwise, it returns `false`.

Code Snippet 1 shows an example of how to send an email through a PHP application.

### Code Snippet 1:

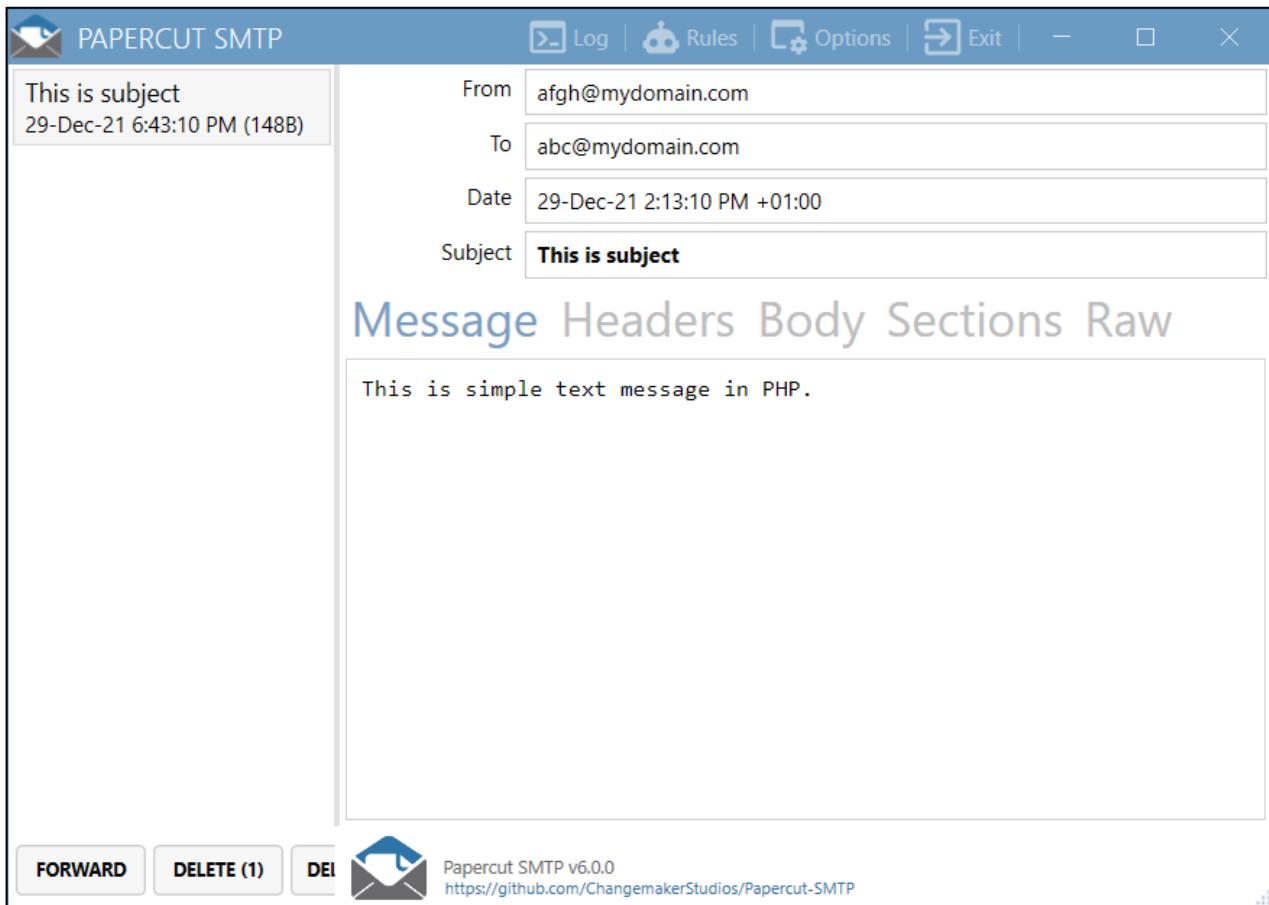
```
<html>
<head>
<title>Sending email using PHP</title>
</head>
<body>
<?php
$to = "abc@mydomain.com";
$subject = "This is subject";
$msg = "This is simple text message in PHP.";
$header = "From:afgh@mydomain.com \r\n";
$retval = mail ($to, $subject, $msg, $header);
if( $retval == true )
{
echo "Message successfully sent. . .";
}
else
{
echo "Message could not be sent . . .";
}
?>
</body>
</html>
```

The code transmits an email message to `abc@mydomain.com` from sender id `afgh@mydomain.com`. When the `mail()` function is called, PHP attempts to send the email. The function returns `true` if the email is sent successfully, otherwise, it returns `false`. Based on return value, an appropriate message is displayed to the user. Figure 10.1 shows the output in the browser for Code Snippet 1.



**Figure 10.1: Output for Code Snippet 1**

Figure 10.2 shows the Papercut server window showing the received mail.



*Figure 10.2: Papercut Server Output*

Thus, Papercut helps to verify working of a PHP script for sending and receiving mail.

### **10.3 Sending an HTML Email**

In PHP, when a user sends a text message, by default, it is interpreted as simple text. Even if the message contains HTML tags, it is treated as text. It is important to note that HTML tags are not formatted as per HTML syntax. However, if the user prefers to send an email in HTML format itself, there is an option for that as well. To send such an email, the user has to ensure to input the MIME version, content type, and character set.

Usually, built-in PHP functions `htmlspecialchars()` and `htmlentities()` are used to display HTML tags as simple text.

To show HTML tags as simple text on a Website, `< tag` is replaced with '`&lt;`' or '`&gt;`' and '`>`' is replaced with '`&gt;`' or '`&gt;`' whenever they are encountered in the markup.

For example, if you want to show, '<p>your html code is ready</p>' on the Web page, write it as &lt;p&gt; your html code is ready &lt;/p&gt;. Based on the email program of the receiver, the message sent appears in that particular format on the screen. The message sent in HTML format will be converted to plain text if the recipient's email program is set to convert messages that are received.

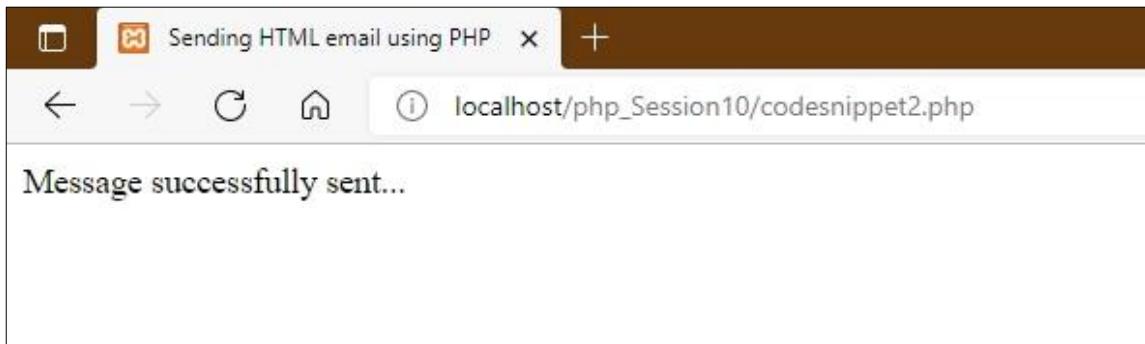
In general, three different types of message formats available are namely, HTML, Plain Text, and Rich Text Formats. However, in PHP, the most common formats are HTML or plain text.

Code Snippet 2 shows an example of sending an HTML email.

### **Code Snippet 2:**

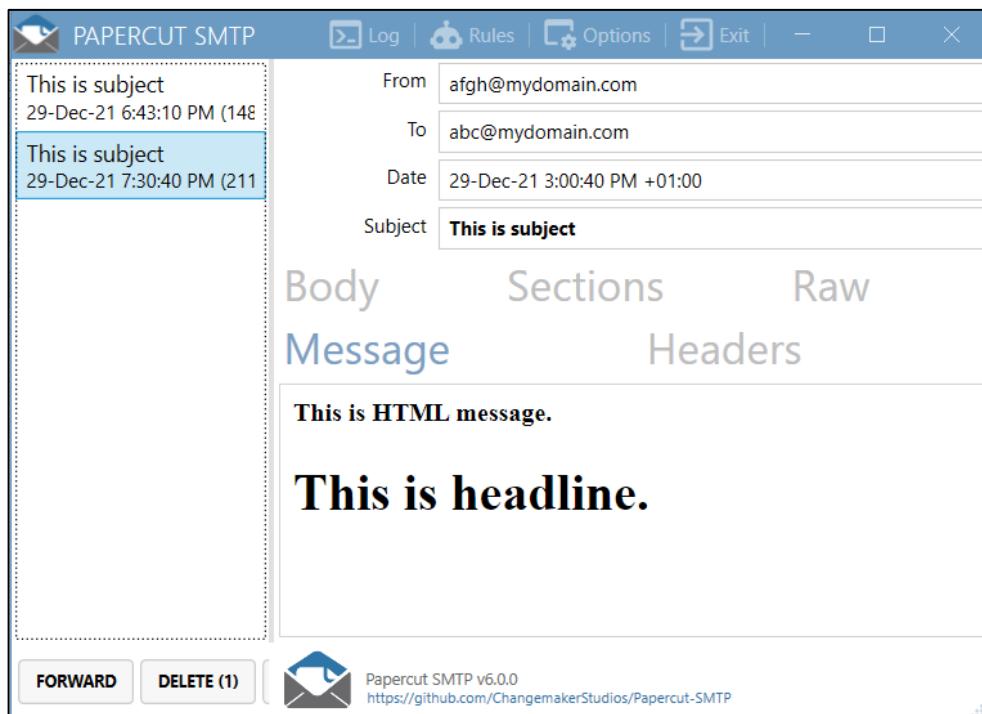
```
<html>
<head>
<title>Sending HTML email using PHP</title>
</head>
<body>
<?php
$to = "abc@mydomain.com";
$subject = "This is subject";
$msg = "<b>This is HTML message.</b>";
$msg .= "<h1>This is headline.</h1>";
$header = "From:xyz@mydomain.com\r\n";
$header .= "MIME-Version: 1.0\r\n";
$header .= "Content-type: text/html\r\n";
$retval = mail ($to,$subject,$msg,$header);
if( $retval == true )
{
echo "Message successfully sent...";
}
else
{
    echo "Message not sent...";
}
?>
</body>
</html>
```

This code transmits an HTML email message to abc@mydomain.com. Figure 10.3 shows the output in the browser for Code Snippet 2.



**Figure 10.3: Output for Code Snippet 2**

Figure 10.4 shows the Papercut server window showing the received mail in HTML format.



**Figure 10.4: Papercut Server Output for Code Snippet 2**

## 10.4 Sending Attachments with an Email

When a user prefers to send an email that has attachments, it is essential to set the Content-type header to multipart/mixed. Then, the user can indicate the text section and attachment section within boundaries.

To begin a boundary, enter two hyphens and after that enter a unique number. To generate the unique number, users can utilize the function `md5()`, which generates a 32-digit hexadecimal number. This unique number cannot be part of a message in the email.

```
$header .= "boundary=$num\r\n";
```

```
$header .= "--$num\r\n";
```

To end a boundary, which also marks the conclusion of the email, use two hyphens.

Code Snippet 3 shows an example of sending an email with attachment.

### Code Snippet 3:

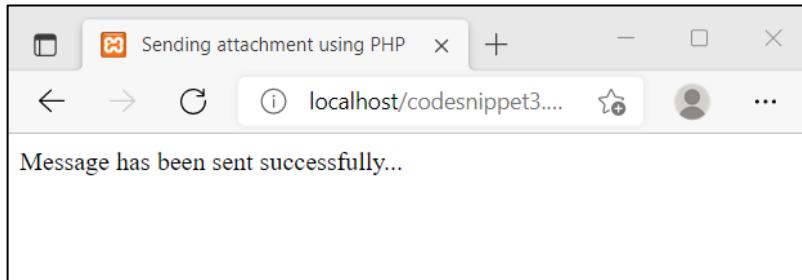
```
<html>
<head>
<title>Sending attachment using PHP</title>
</head>
<body>
<?php
$to = "Papercut@user.com";
$subject = "This is subject for the mail with attachment";
$msg = "<b>This is HTML message.</b>";
$msg .= "<h1>This is headline.</h1>";
# Open a file and read its contents into a variable.
$file = fopen( "c:\\misc\\test.c", "r" );
if( $file == false )
{
echo "Error in opening the file";
exit();
}
$size = filesize("c:\\misc\\test.c");
$content = fread( $file, $size );
$encoded_content = chunk_split( base64_encode($content));
# Generate a random 32-bit number by seeding time().
$num = md5( time() );
# Define the main headers.
$header = "From:Papercut@papercut.com\r\n";
$header .= "MIME-Version: 1.0\r\n";
$header .= "Content-type: text/html\r\n";
$header .= "Content-Disposition:attachment; ";
// header
$uid = md5(uniqid(time()));
// message & attachment
//$/file_name="c:\\misc\\test.c";
//$/filename= "test.c";
$nmessage = "--".$uid."\r\n";
$nmessage .= "Content-type:text/plain; charset=iso-8859-1\r\n";
$nmessage .= "Content-Transfer-Encoding: 7bit\r\n";
//$/nmessage .= $msg."\r\n";
$nmessage .= "--".$uid."\r\n";
$nmessage .= "Content-Type: application/octet-stream;
name=c:\\misc\\test.c\r\n";
$nmessage .= "Content-Transfer-Encoding: base64\r\n";
$nmessage .= "Content-Disposition: attachment;
filename=c:\\\\misc\\\\test.c\r\n";
```

```

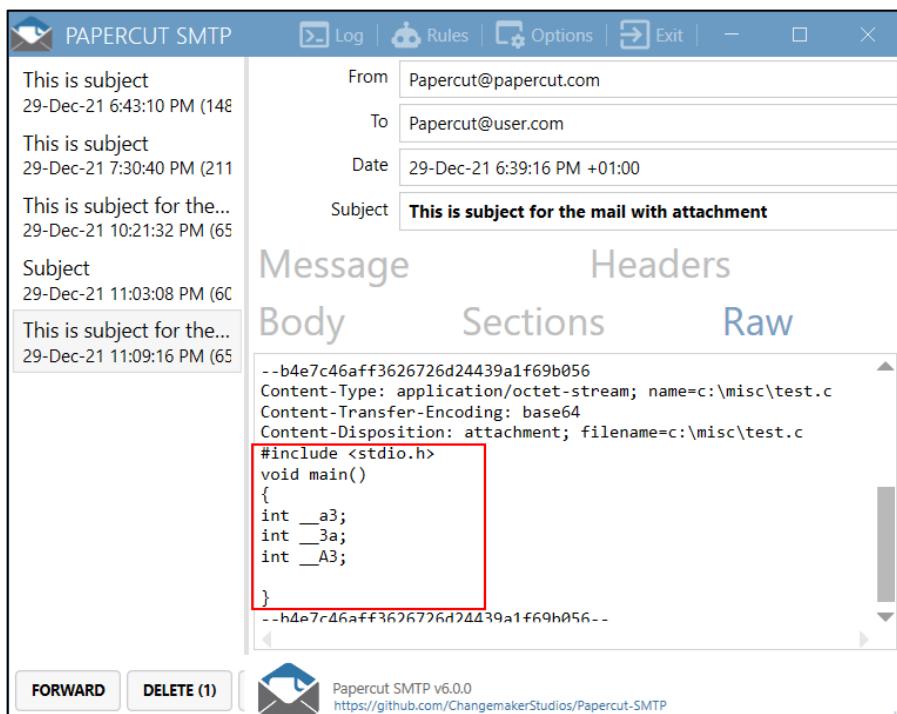
$nmensaje .= $content."\r\n";
$nmensaje .= "--".$uid."--";
$retval = mail ( $to, $subject, $nmensaje, $header);
if( $retval == true )
{
echo "Message has been sent successfully...";
}
else
{
echo "Message could not be sent...";
}
?>
</body>
</html>

```

This code sends the contents of a file **test.c** as an attachment with the email. Figures 10.5 and 10.6 show the output of Code Snippet 3 in the browser and in Papercut server.



*Figure 10.5: Output for Code Snippet 3 in Browser*



*Figure 10.6: Output for Code Snippet 3 in Papercut Server*

## 10.5 User Authentication

---

User authentication is a very crucial process in any Web application to ensure that unauthorized persons do not gain access to the application. In this process, the login data entered by the user is compared and matched with data present in the database of the server. Using this security mechanism, it is possible to restrict unauthorized users from accessing Web tools or applications.

### PHP User Authentication with MySQL

User authentication process helps to validate users. Validation is done using certain keys, tokens, or credentials. In case a user enters the approved credentials, the authentication process is considered successful, otherwise it is considered failed. If the process is successful, the user is permitted to access the system.

The process to create a user authentication system in PHP involves three main steps:

Create a MySQL database table.

Develop a login form for the user to pass the login details to PHP.

Create a query that performs a comparison of the user data with MySQL database.

Let us now understand each step in detail.

To generate a database and the corresponding table structure, execute the CREATE statement. Users have the flexibility to choose any database client, such as SQLYog and PHPMyAdmin. In this example, PHPMyAdmin through XAMPP is the database client chosen.

Consider a scenario of a student login form. Create a database and name it as `login`. Then, create a `student` table within `login`.

Tables 10.2 and 10.3 shows the `student` table structure and data.

ColumnName	Type	Description
StudentName	Varchar(100)	Stores student name
Password	Varchar(8)	Stores student password

*Table 10.2: Student Table Structure*

<b>StudentName</b>	<b>Password</b>
Peter	Peter234
Joesph	Joe76

**Table 10.2: Student Table Data**

Code Snippet 4 shows the code to display a Login Form to the users. They can input their authentication details in this form. The code comprises mostly HTML markup to render the form.

#### **Code Snippet 4:**

```

<html>
<body>
<form name="frmStudent" method="post" action="authenticate.php">
<div ></div>
<table border="0" cellpadding="10" cellspacing="1" width="500"
align="center" >
  <tr >
    <td align="center" colspan="2">Enter Login Details</td>
  </tr>
  <tr >
    <td>
      <input type="text" name="studentName" placeholder="Student
Name" ></td>
    </tr>
    <tr >
      <td>
        <input type="password" name="password" placeholder="Password"
></td>
      </tr>
      <tr >
        <td align="center" colspan="2"><input type="submit"
name="submit" value="Submit" ></td>
      </tr>
    </table>
</form>
</body>
</html>

```

The code displays a form with two input fields. The first one is for the name of the student and the next is for the password. When the user inputs the data for the two fields and submits the form, the data will be passed to PHP for authentication. This part of code is yet to be written. The authentication will be done against data stored in MySQL database.

After user submits login data in the form, user inputs are compared with the values present in the database.

Code Snippet 5 shows how to generate a query for this comparison operation in a PHP script.

### Code Snippet 5: (authenticate.php)

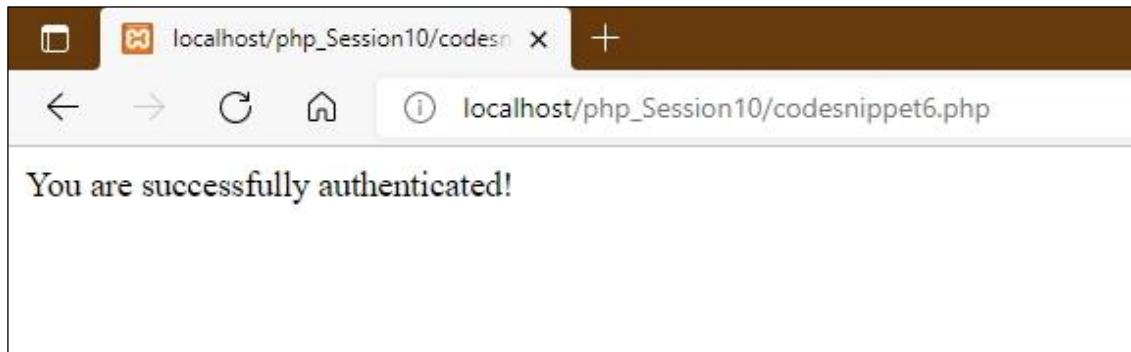
```
<?php
if(count($_POST)>0) {
    $conn = mysqli_connect("localhost","root","","","login");
    $str= "SELECT * FROM student WHERE studentname='".
$_POST["studentName"] . "' and password = '' .
$_POST["password"]."";
    $count = mysqli_num_rows(mysqli_query($conn, $str));
    echo $count;
    if($count==0) {
        $message = "Invalid Studentname or Password!";
    } else {
        $message = "You are successfully authenticated!";
    }
    echo $message;
}
?>
```

The code determines the length of `$_POST` global array to verify that a POST action has occurred. This array holds the user input data, which are obtained after the authentication form is submitted. The code then, connects to MySQL database by providing the required credentials such as username and password for MySQL. Next, the code generates a query to retrieve that record whose student name and password match with the data submitted in the form. This query is then executed against the database table, student. If the rows returned by the query is zero, it means no match was found for the given credentials. Thus, the user is unauthorized to access the Web application. However, if rows are non-zero, it means that a match was found and user is successfully authenticated. Appropriate message indicating this is displayed on the browser screen.

Figures 10.7 and 10.8 shows the output for Code Snippet 4.

The screenshot shows a web browser window with the URL `localhost/php_Session10/codesnippet4.php`. The page title is "Enter Login Details". It contains two text input fields: one for "Student Name" and one for "Password". Below the fields is a "Submit" button. The browser interface includes standard navigation buttons (back, forward, search, etc.) and a tab bar with other open windows.

**Figure 10.7: Output for Code Snippet 4 Showing Authentication Form**



*Figure 10.8: Output for Code Snippet 5*

Thus, using PHP code and MySQL database, user authentication can be performed.

## 10.6 Summary

- The SMTP server assists in sending emails to the users. Email application can be set using this server to send an email.
- In order to send an email, a user must ensure that the `php.ini` file includes the corresponding configurations for PHP and the details about how the user's system manages mails.
- To send an email, PHP utilizes `mail()` function. The three arguments that must be provided to this function are the receiver's email address, the subject of the message, and the descriptive message.
- In PHP, when a user sends a text message, it is interpreted as simple text. Even if the message contains HTML tags, it is treated as text.
- To send an HTML message, the user has to ensure to input the MIME version, content type, and character set.
- When a user prefers to send an email that has attachments, it is essential to set the `Content-type` header to `multipart/mixed`. Then, the user can indicate the text section and attachment section within boundaries.
- User authentication is a process that helps to verify and validate users. Validation is done using certain keys, tokens, or credentials.

## 10.7 Test Your Knowledge



1. Which one of the following functions is used to send an email using PHP script?
  - a. mail\_send()
  - b. send\_mail()
  - c. mailrr()
  - d. mail()
2. Which of the following describes the user's email server address?
  - a. sendmail\_from
  - b. send\_mail()
  - c. SMTP
  - d. None of these
3. The \_\_\_\_\_ symbol is used for separating recipient addresses.
  - a. /
  - b. &
  - c. ,
  - d. :
4. When a user wants to send an email that has attachments, it is essential to set the Content-type header to \_\_\_\_\_.
  - a. multipart/mixed
  - b. multipart/attachment
  - c. multipart/none
  - d. mixed/multipart
5. What is the process of validating a user's credentials before granting access to an application called as?
  - a. Parsing
  - b. Entry
  - c. Authentication
  - d. Validation

## **Answers to Test Your Knowledge**

---

1. mail()
2. sendmail\_from
3. , (Comma)
4. multipart/mixed
5. Authentication

## 10.8 Try it Yourself

1. Consider a scenario where an HTML email message must be sent to xyz@somedomain.com. Write a PHP script to achieve this.

# Session 11

## File Handling and Exception Handling in PHP



### Learning Objectives

*In this session, students will learn how to:*

- Describe file uploading and file handling in PHP
- Explain upload script creation and upload form creation
- Explain files in PHP through the `readfile()` function
- Define file open/read/close in PHP
- Identify and explain various file functions
- Elaborate the process of file upload using PHP
- Explain error and exception handling in PHP
- Illustrate `die()` function, `try...catch` statement, exception object, and `try...catch...finally` statement

The session starts with explaining file handling and file uploading in PHP language using `fwrite()` and `fopen()` functions. It offers an outline of upload form creation and upload script creation. The session also provides information about manipulating files in PHP. It further describes file open/read/close in PHP by covering functions such as `fgets()`, `fread()`, `fopen()`, `fgetc()`, and `feof()`. The session also introduces us to exception handling and error handling in PHP. Further, gives an overview of the `die()` function, `try...catch...finally` statement, `try...catch` statement, and exception object.

## **11.1 File Handling**

---

The PHP file system enables users to perform various file handling tasks. These include creating, writing, deleting, appending, and closing files. It also allows them to read a file character by character and line by line.

### **PHP Write File - `fwrite()`**

`fwrite()` function in PHP helps write some content of string(s) into a file. Following is the syntax for using the `fwrite()` function:

#### **Syntax:**

```
int fwrite (resource $handle, string $string [, int $length ])
```

Code Snippet 1 demonstrates how to use the `fwrite()` function.

#### **Code Snippet 1:**

```
<html>
<body>
<?php
$fp = fopen('datum.txt', 'w');
//opening the file in write mode
fwrite($fp, 'hi');
fwrite($fp, 'sample file');
fclose($fp);
echo "File has been written.";
?>
</body>
</html>
```

This code writes some text into the file `datum.txt`. The first parameter of `fwrite()` is the file name to write to and the other parameter contains the string to be written. In this case, `$fp` is the resource handle, while `hi` and `sample file` are the strings that get written to the `datum.txt` file.

Figure 11.1 shows the output for the code.



*Figure 11.1: Writing to File*

### **PHP Delete File - `unlink()`**

Users can delete a file using the PHP `unlink()` function.

Syntax for using the `unlink()` function is as follows:

#### **Syntax:**

```
bool unlink (string $filename [, resource $context ])
```

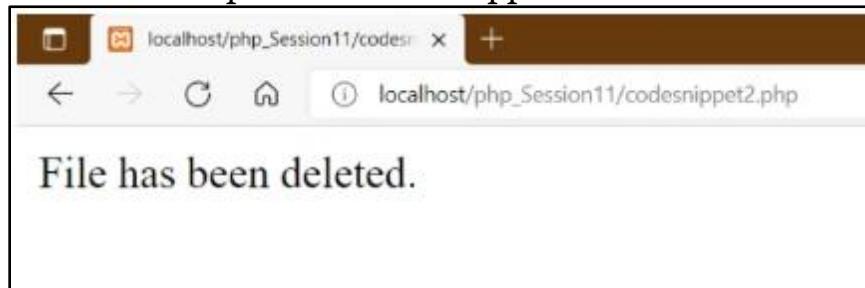
Code Snippet 2 shows how to use the `unlink()` function.

#### **Code Snippet 2:**

```
<html>
<body>
<?php
unlink('datum.txt');
    echo "File has been deleted.";
?>
</body>
</html>
```

In the given code, the `unlink()` function simply deletes the contents of `datum.txt` file.

Figure 11.2 shows the output for Code Snippet 2.



*Figure 11.2: Deleting a File*

## 11.2 File Uploading

---

Files can be uploaded to a server by using a PHP script along with an HTML form. In the beginning, files are uploaded into a temporary directory. Through a PHP script, these files can then be transferred to a target destination subsequently. The `phpinfo.php` page contains information that defines the temporary directory, that is, `upload_tmp_dir`, which is used for file uploads. It also includes `upload_max_size` describing maximum allowed file size that can be uploaded successfully. The `php.ini` PHP configuration file comprises these parameters.

A user can perform following process to upload a file:

- 
- Open the page consisting of an HTML form featuring a submit button, browse button, and text files.
  - Click or tap the browse button to choose a file for uploading from a local PC.
  - The complete path of the chosen file shows up in the text field.
  - Click submit button.
  - The chosen file gets transferred to the server's temporary directory.
  - The PHP script designated as the form handler in the action attribute inspects if the file has reached.
  - Then, the file is copied into the destined directory.
  - Finally, the PHP script acknowledges success to a user.

A noteworthy point is to have permission for file writing. This holds true for both, final and temporary locations. The process will inevitably fail if either of these is defined to be read-only. An uploaded file can be any document, image file, or text file.

## 11.3 Upload Form Creation

---

Using HTML form, PHP code allows users to upload files to the server. In the beginning, files are placed in a temporary directory. Then, a PHP script is used to move the files to a target destination.

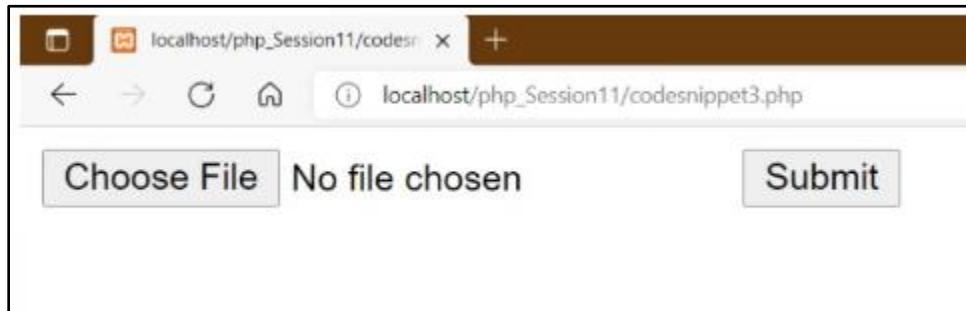
The HTML code given in Code Snippet 3 creates an uploader form. The form has the `enctype` attribute set to `multipart/form-data` and the `method` attribute assigned as a `post`.

### Code Snippet 3:

```
<?php
    if(isset($_FILES['image'])) {
        $errors= array();
        $file_size =$_FILES['image']['size'];
        $file_name = $_FILES['image']['name'];
        $file_type=$_FILES['image']['type'];
        $file_tmp =$_FILES['image']['tmp_name'];
        $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));
        if($file_ext != ".png" && $file_ext != ".jpeg" && $file_ext != ".jpg") {
            $errors[]="This file extension is not allowed, please choose a png or jpeg file.";
        }
        if($file_size > 2097152) {
            $errors[]='File size should be 2 MB';
        }
        if(empty($errors)==true) {
            move_uploaded_file($file_tmp,"image/".$file_name);
            echo "File Uploaded successfully!";
        }else{
            print_r($errors);
        }
    }
?>
<html>
    <body>
        <form action="" method="POST" enctype="multipart/form-data">
            <input type="file" name="image" />
            <input type="submit"/>
        </form>
    </body>
</html>
```

The code in Code Snippet 3 creates the form that prompts the user to upload a file. It also validates the file selection, however, the actual code to implement upload action is not yet included here.

Figure 11.3 shows the output for the code.



*Figure 11.3: Selecting File to Upload*

## 11.4 Upload Script Creation

The `$_FILES` global PHP variable is an associate double-dimension array that maintains all the required information associated with an uploaded file. PHP would generate five variables if the value allocated to the `name` attribute of the input while uploading the form was `file`. These variables are mentioned as follows:

<code>\$_FILES['file']['error']</code>	<code>\$_FILES['file']['type']</code>	<code>\$_FILES['file']['name']</code>	<code>\$_FILES['file']['size']</code>	<code>\$_FILES['file']['tmp_name']</code>
Error code related to file upload.	Multipurpose Internet Mail Extensions (MIME) type of the uploaded file.	Real name of the uploaded file.	Size of the uploaded file in bytes.	Uploaded file located in the temporary directory on a Web server.

Code Snippets 4a and 4b shows a complete example of uploading an image file. The output in the browser also displays information about the uploaded file.

## **Code Snippet 4a:** (code11\_4.html)

```
<html>
  <body>
    <form action="code11_4.php" method = "POST" enctype =
"multipart/form-data">
      <input type = "file" name = "image" />
      <input type = "submit"/>
    </form>
  </body>
</html>
```

## **Code Snippet 4ba:** (code11\_4.php)

```
<?php
$target_dir = "/uploads"; //Your system directory location
$target_file = $target_dir .
basename($_FILES['image']['name']);
if(isset($_FILES['image'])){
    $errors= array();
    $file_type = $_FILES['image']['type'];
    $file_name = $_FILES['image']['name'];
    $file_tmp = $_FILES['image']['tmp_name'];
    $file_size = $_FILES['image']['size'];
    $extension= explode('.',$file_name);
    $file_ext=strtolower(end($extension));
    $extensions= array("jpeg","png","jpg","gif");
    if(in_array($file_ext,$extensions)== false){

        $errors[]="This file extension is not allowed";
    }
    // Verify if image file is a fake image or an actual image
    $check = getimagesize($file_tmp);
    if($check !== false) {
        echo "The file is an image - " . $check["mime"] . ".";
    } else {
        $errors[]="The file is not an image.";
    }
    //Verify whether file size is valid or not
    if($file_size > 2097152) {
        $errors[]='File size must be exactly 2 MB';
    }
    //Verify whether file size exists already or not
    if (file_exists($target_file)) {
        $errors[]="Hey, the file exists already.";
    }
}
```

```

} //Verify whether file size is valid or not
if ($file_size > 500000) {
    $errors[]="File too large to be uploaded!.";

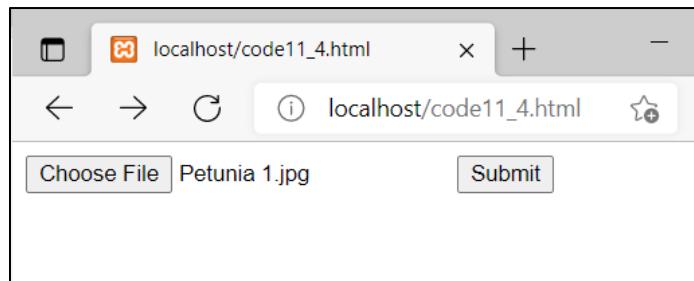
}

// Inspect whether the upload was successful or not
if(empty($errors)==true) {
    if(move_uploaded_file($file_tmp,"/uploads".$file_name )) {
        echo "The file ". $file_name. " has been uploaded
successfully!";
        echo "<ul>";
        echo " <li>Sent file: ". $_FILES['image']['name'];
        echo " <li>File size: " . $_FILES['image']['size'];
        echo " <li>File type: ". $_FILES['image']['type'];
        echo "</ul>";

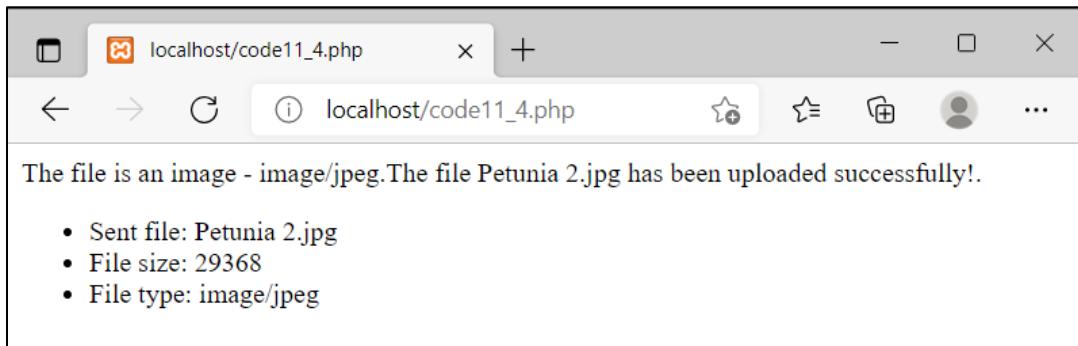
    } } else {
        print_r($errors);
    }
}

```

Figures 11.4 and 11.5 shows the outputs of Code Snippet 4a and Code Snippet 4b respectively for the code.



**Figure 11.4: Example of File Upload**



**Figure 11.5: File Upload Successful**

If the file was uploaded already, file extension was invalid, or file size was above limits, and so on, appropriate error message will be displayed on the browser based on the specific error.

## 11.5 Manipulating Files

---

PHP has multiple functions for reading, editing, creating, and uploading files. However, it is important to be cautious while manipulating files. A wrong move can prove to be detrimental to the computer system.

Following are common errors that occur during file manipulation:

Deleting a file's content

Filling a hard disk drive with garbage data

Editing the wrong file

Users can avoid these errors by being extra careful.

### 11.5.1 PHP `readfile()` Function

PHP's `readfile()` function reads a file's contents and writes them to the output buffer. Consider an example where there is a text file named 'words.txt' stored on the server.

Assume that it contains the following text:

```
C Programming  
Java Programming  
Jakarta Web Services  
PHP Hypertext Preprocessor  
Cloud Computing  
Artificial Intelligence
```

Code Snippet 5 demonstrates the PHP program that can read a file and display it on the output buffer. This is done by using the `readfile()` function.

#### Code Snippet 5:

```
<html>  
<body>  
<?php  
echo readfile("words.txt");  
?>
```

```
</body>  
</html>
```

In this code, the `readfile()` function displays the file contents on the output buffer and returns the number of bytes as well. Here, it is assumed that the file `words.txt` has been created in the same path as the PHP file containing the code. Figure 11.6 shows the output for the code.



**Figure 11.6: Using `readfile()` to Read File Contents**

## 11.6 File Open/Read/Close

This section details how to open, read, or close a file on the server using PHP script.

### 11.6.1 Open File - `fopen()`

PHP's `fopen()` function helps us to open files. It provides much more options as compared to the `readfile()` function.

Following is the syntax for `fopen()` function:

#### Syntax:

```
resource fopen (string $filename, string $mode [, bool $use_incl  
ude_path = false [, resource $context ]])
```

Code Snippet 6 shows how `fopen()` can help us read the contents of a file placed on a local system.

#### Code Snippet 6:

```
<?php  
$handle = fopen("c:\\\\folder\\\\text.txt", "r"); // 'r' opens the  
//text file in read-only mode  
?>
```

This code shows how `fopen()` opens a text file `text.txt` from the path '`c:\\\\folder\\\\text.txt`' in read-only mode.

## 11.6.2 Read File - `fread()`

The `fread()` function in PHP enables users to read the content of an open file. It accepts two parameters: file size and resource. The first argument is the name of the target file and the other one determines the maximum number of bytes to read.

### Syntax:

```
string fread(resource $handle, int $length)
```

## 11.6.3 Close File - `fclose()`

The `fclose()` function helps a user to close an opened file. This function's parameter consists of the variable which holds the filename that must be closed.

Syntax for `fclose()` function is as follows:

### Syntax:

```
bool fclose (resource $handle)
```

Consider a text file named 'textfl.txt' that contains the following content:

Hello Universe

Code Snippet 7 illustrates the usage of `fread()` and `fclose()` functions with this text file.

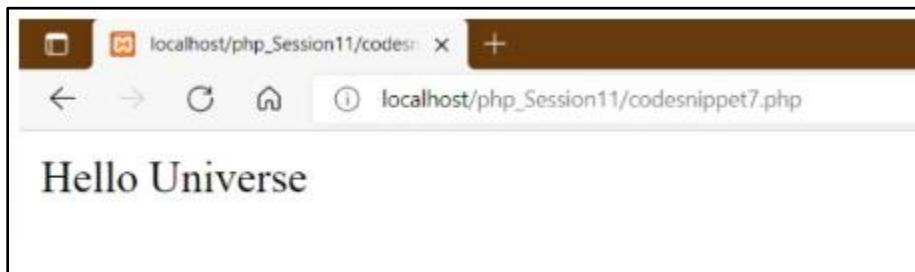
### Code Snippet 7:

```
<?php
$file = "c:\\textfl.txt";
$handle = fopen($file, "r");
//opening the file in read-only mode
$contents = fread($handle, filesize($file)); //reads file
echo $contents; //displays the data of file
fclose($handle); //closes the file
?>
```

This code uses the `fopen()` function to open the file `textfl.txt` in read-only mode. The `fread()` function reads the file. Then, it displays its contents to the output buffer using `echo`. Finally, function closes the file `textfl.txt`. The

`fclose()` function has `$handle` variable as the parameter that holds the name of the text file.

Figure 11.7 shows the output for Code Snippet 7.



*Figure 11.7: File Content Displayed*

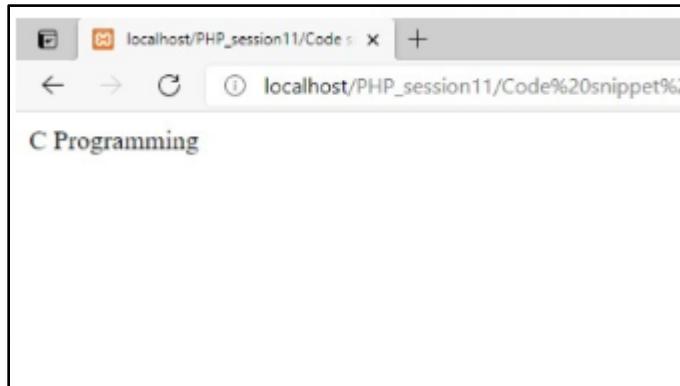
#### 11.6.4 Read Single Line - `fgets()`

PHP's `fgets()` function allows a user to read a single line from an open file. Code Snippet 8 depicts the usage of `fgets()` function. Let us consider the 'words.txt' text file used in Code Snippet 5.

### **Code Snippet 8:**

```
<html>
<body>
<?php
$ourfile = fopen("words.txt", "r") or die("failed to open your
file!");
echo fgets($ourfile);
fclose($ourfile);
?>
</body>
</html>
```

This code opens the file `words.txt` using the `fopen()` function. Then, the `fgets()` function along with `echo` statement is used to output the first line of the `words.txt` file as shown in Figure 11.8.



**Figure 11.8: First Line Displayed from Text File**

#### **11.6.5 Check End-Of-File - `feof()`**

PHP function `feof()` helps a user to loop through data of undisclosed length. It examines a file and verifies whether the user has reached the 'End-of-File' (EOF).

Code Snippet 9 demonstrates the usage of `feof()` function.

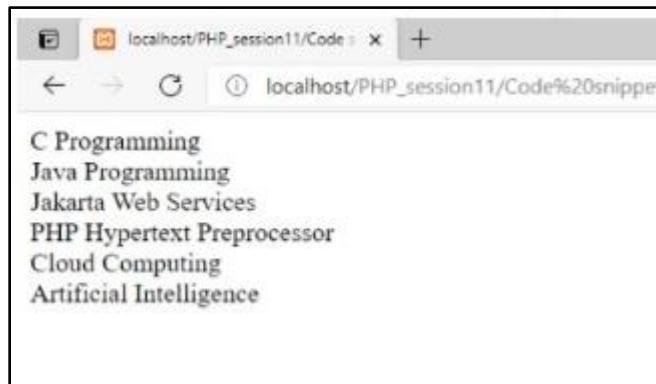
### **Code Snippet 9:**

```
<html>
<body>
<?php
$ourfile = fopen("words.txt", "r") or die("failed to open your
```

```
file!");
// Outputs one line until the end-of-file is reached
while(!feof($ourfile)) {
    echo fgets($ourfile) . "<br>";
}
fclose($ourfile);
?>
</body>
</html>
```

In this code, the `words.txt` file is held in the `$ourfile`. The `while` loop ensures that the code reads the file line by line until the EOF is reached by using `feof()` function in the condition. Moreover, the `echo` statement before the `fgets()` function displays the contents of the file as output. Lastly, `fclose()` closes the file.

Figure 11.9 shows the output for the code.



*Figure 11.9: Contents of the File Displayed*

### 11.6.6 Read Single Character - `fgetc()`

PHP's `fgetc()` function serves as a means to read a single character from the files.

Code Snippet 10 shows us how `fgetc()` function is used.

#### **Code Snippet 10:**

```
<html>
<body>
<?php
$ourfile = fopen("words.txt", "r") or die("failed to open your
```

```
file!");
// Outputs one character of the file until end-of-file is
reached
while(!feof($ourfile)) {
    echo fgetc($ourfile);
}
fclose($ourfile);
?>
</body>
</html>
```

This code reads the `words.txt` file character by character. The `$ourfile` variable has the `fopen()` function that opens the text file. The `while` loop has `feof()` function in the condition to make sure to run the loop statements until EOF is reached. The statement inside the loop, that is `echo fgetc($ourfile);`, displays contents of the file as shown in Figure 11.10. Finally, `fclose()` closes the file.



*Figure 11.10: Contents of the File Displayed*

## 11.7 PHP Files

A Web page that consists of PHP code is called as a PHP file. These PHP files can be read, modified, or written to. To perform these actions, PHP functions come into use.

### 11.7.1 Reading File - `fopen()` and `fread()`

Users can read a file's content using a function termed as `fread()` after it has been opened through the `fopen()` function. The `fread()` function requires two

arguments. These include the file pointer and the file length (specified in bytes). The `filesize()` function helps the users to find the file length. It returns the file size (expressed in bytes).

Following are the steps performed to read a file in PHP:

Use the `fopen()` function to open a file.

Use the `filesize()` function to get the length of the file.

Use the `fread()` function to read the contents of the file.

Use the `fclose()` function to close the file.

Code Snippet 11 explains the reading of a file in detail. Consider a text file called `sample.txt` that contains the following content:

Hello and welcome to the world of PHP programming. PHP enables you to create amazing Web-based applications.

### Code Snippet 11:

```
<html>
  <head>
    <title>Reading a file using PHP</title>
  </head>
  <body>
    <?php
      $fname = "sample.txt";
      $file = fopen($fname, "r"); //opens the file in read
                                //mode
```

```

if($file == false) {
    echo ("An error has occurred while opening the
          file");
    exit();
}
$filesize = filesize($filename);
$filetext = fread($file, $filesize);
fclose($file);

echo ("File size: $filesize bytes");
echo ("<pre>$filetext</pre>");
?>
</body>
</html>

```

This code assigns the content of the `sample.txt` text file to a variable `$filename`. `fopen()` function opens this file in the read mode. `filesize()` function fetches the size of the file. `fread()` function reads its contents. Finally, the code outputs the contents on a Web page using `echo` statements as shown in Figure 11.11.



**Figure 11.11: Contents Displayed on Web Page**

### 11.7.2 Writing to File - `fwrite()`

Users can use the `fwrite()` function to append text to an existing file or write a new file. This PHP function requires two arguments. One specifies the string of data to be written and the other specifies a file pointer.

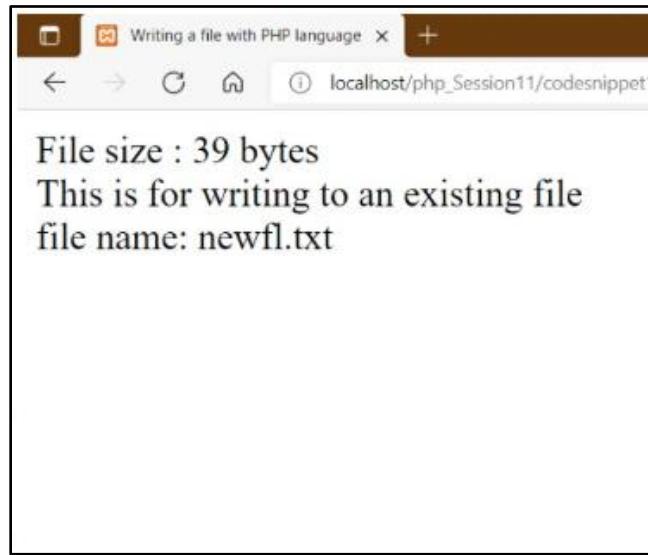
Users can also include a third optional integer argument for specifying the length of the data to be written. In case, third argument is included, the writing operation will be terminated after designated length has been reached.

Code Snippet 12 shows an example of how `fwrite()` function can be used.

### Code Snippet 12:

```
<?php
    $filename = "newfl.txt";
    $file = fopen($filename, "w"); //opening the file in write
    // mode
    if($file == false) {
        echo ("There was an error in opening your file");
        exit();
    }
    fwrite($file, "This is for writing to an existing file");
    fclose($file);
?>
<html>
    <head>
        <title>Writing a file with PHP language</title>
    </head>
    <body>
        <?php
            $filename = "newfl.txt";
            $file = fopen($filename, "r");
            if($file == false) {
                echo ("There was an error while opening your
file");
                exit();
            }
            $filesize = filesize($filename);
            $filetext = fread($file, $filesize);
            fclose($file);
            echo ("File size: $filesize bytes."<br>");
            echo ("$filetext" .<br>);
            echo("file name: $filename".<br>);
        ?>
    </body>
</html>
```

This code generates a new text file called `newf1.txt`. It opens the file from a pre-defined path in write mode. The program uses the `fwrite()` function and writes a string of text inside the file. Then, the `fclose()` function closes the new file. `echo` statements are used to display the output. It returns the size of the file along with the contents of the file. Figure 11.12 shows the output for Code Snippet 12.



*Figure 11.12: Size of the File with File Contents Displayed*

## **11.8 Error and Exception Handling in PHP**

---

Error handling can be defined as the process of detecting errors in a program and undertaking appropriate measures to get rid of them. Ignoring the evident errors may cause unwanted implications during programming.

### **11.8.1 `die()` Function**

Users must evaluate any error conditions while writing PHP code. The purpose of `die()` function is similar to that of an exit function. However, it can also help in error handling. `die()` function can be used to halt the program if any possible errors are foreseen.

Code Snippet 14 makes use of `die()` function in the code.

## Code Snippet 14:

```
<?php  
if(!file_exists("/downloads/cmp.txt")) {  
    die("File unavailable");  
} else {  
    $file = fopen("/downloads/cmp.txt", "r");  
    print "Opened your file successfully";  
}  
?>
```

If Code Snippet 14 is executed without having /downloads/cmp.txt file, an error message is displayed: File unavailable.

This enables writing an efficient code. Moreover, this function helps to display a much more user-friendly and meaningful message. Figure 11.13 and 11.14 show the output for Code Snippet 14 for two different cases.

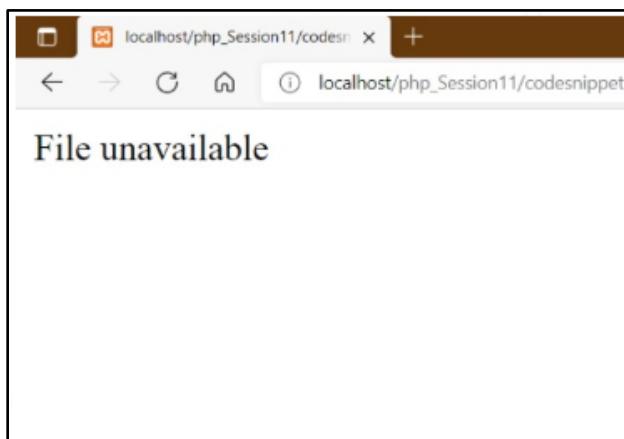


Figure 11.13: Using die() - File Unavailable

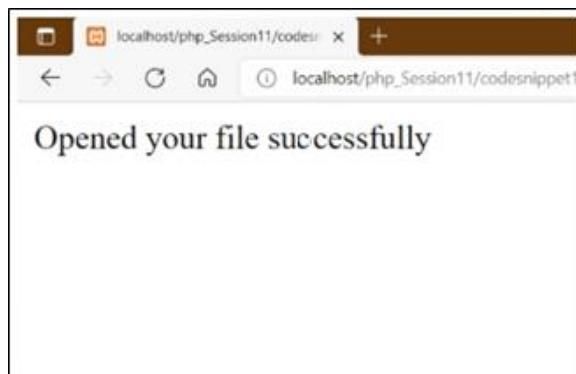


Figure 11.13: Using die() - File Available

## 11.8.2 Exception Handling

An exception is an event occurring during program execution that leads to disruption of the normal flow of the program's instructions. Typically, runtime errors cause exceptions.

PHP 8 has an exception handling model similar to several other programming languages. Exceptions offer better control than error handling. This makes them extremely significant in programming.

Exception Handling Mechanism in PHP (Three Main Keywords)		
<b>catch:</b> A catch block fetches an exception and generates an object carrying the information related to information.	<b>try:</b> Users must place any function employing an exception inside a try block. An exception gets thrown whenever an exception gets triggered in a code. On the contrary, the program will continue to execute normally if an exception is not triggered.	<b>throw:</b> This keyword is used to trigger an exception. Each throw must have a minimum of one catch.

When an exception is thrown, the program after the statement will stop running. Then, PHP will try to locate the first corresponding catch block. In case it is unable to catch an exception, users will see a PHP Fatal Error.

Key things to remember when an exception is thrown are as follows:

- In a catch block, exceptions can be thrown or re-thrown.
- Each try block must have a minimum of one matching catch block. Users can utilize multiple catch blocks for catching multiple classes of exceptions.
- An exception can be caught and thrown within PHP. The code that can successfully catch exceptions might be included in a try block.

## **try...catch Statement**

Users can implement the `try...catch` statement for catching exceptions and to catch exceptions and keep the process going.

### **Syntax:**

```
try {  
    code that may cause exceptions  
} catch(Exception $e) {  
    code that gets executed after catching an exception  
}
```

Code Snippet 15 shows how to use the `try...catch` statement.

### **Code Snippet 15:**

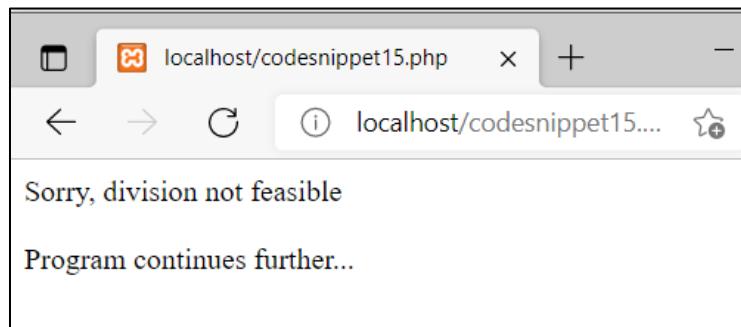
```
<html>  
<body>  
<?php  
try {  
    $val = 5/0;  
    echo $val;  
} catch(DivisionByZeroError $e) {  
    echo "Sorry, division not feasible";  
}  
echo "<br><br>Program continues further..."  
?>  
</body>  
</html>
```

In Code Snippet 15, an attempt is made to perform arithmetic division of two numbers, assign it to a variable, and then, display the result. However, the divisor is zero. This causes the `DivisionByZeroError` exception to occur. The program anticipates this error/exception and includes a catch block to 'handle' the exception. By handling the exception, user can display a customized message. The program can also continue with further statements because of the catch block. If there was no `try...catch` here in the snippet, the output would have just displayed an error and exited without processing any further statements.

Here, the zero value was hard-coded, so it was known that exception will occur. In real-world applications, user may not always know that an exception is

guaranteed to occur. Yet, the user must prepare for such an eventuality by enclosing code within appropriate `try...catch` blocks.

Figure 11.15 shows the output for Code Snippet 15.



**Figure 11.15: User Friendly Message Output**

### **try...catch...finally Statement**

Users can also use the `try...catch...finally` statement to catch exceptions.

The code written in a `finally` block will always get executed irrespective of whether an exception is caught or not. The `catch` block is optional in if the `finally` block already exists and catching the exception is not a priority.

#### **Syntax:**

```
try {  
    code that may cause exceptions  
} catch(Exception $e) {  
    code that gets executed after catching an exception  
} finally {  
    code that always gets executed regardless of whether an  
exception was caught or not  
}
```

Code Snippet 16 shows a program to display a message when an exception is thrown and then, signals that the process continues further.

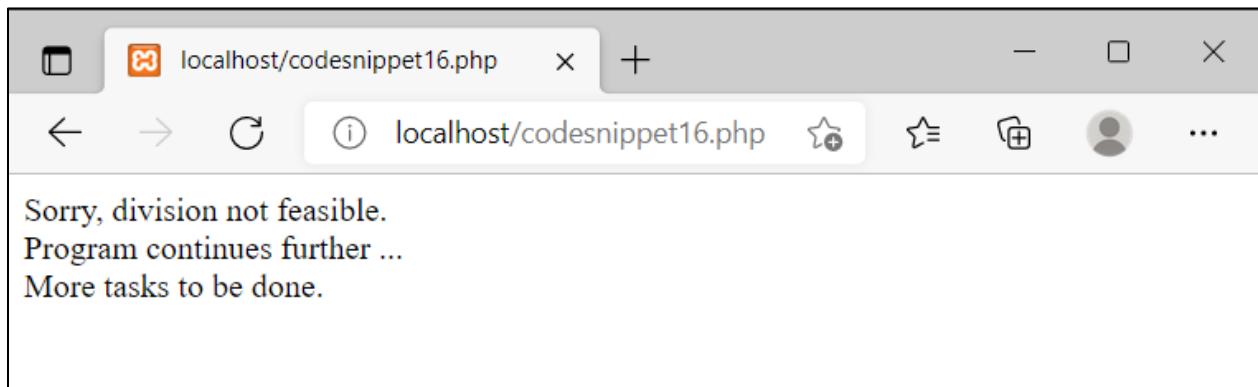
## Code Snippet 16:

```
<html>
<body>
<?php
function division($dividend, $divisor) {
    if($divisor == 0) {
        throw new Exception("Dividing by Zero");
    }
    return $dividend / $divisor;
}
try {
    echo division(5, 0);
} catch(Exception $e) {
    echo "Sorry, division not feasible. <br>";
} finally {
    echo "Program continues further ...<br>" ;
}

echo "More tasks to be done.";

?>
</body>
</html>
```

Figure 11.16 shows the output for Code Snippet 16.



*Figure 11.16: Output for Code Snippet 16*

## Throwing Exceptions

Various PHP classes ‘throw’ an exception. User-defined classes can throw exceptions as well.

## Throwing an Exception

The `throw` statement can be used to permit a user-defined method or function to throw an exception. The code following the statement will cease after an exception is thrown.

In Code Snippet 17, there is an attempt to throw an exception without trying to catch it.

### Code Snippet 17:

```
<html>
<body>
<?php
function division($dividend, $divisor) {
    if($divisor == 0) {
        throw new Exception("Dividing by Zero");
    }
    return $dividend / $divisor;
}
echo division(5, 0);
?>
</body>
</html>
```

In this code, a `throw` block is used to throw an exception. However, there is no `catch` block to catch the exception. Therefore, a fatal error is received as shown in Figure 11.17.



*Figure 11.17: Fatal Error*

## Exception Object

The `Exception` object comprises information about the unanticipated behavior or the error encountered by a function.

Table 11.1 elaborates the parameters used with the exception object in detail.

Parameter	Description
Previous	Optional. This is a string explaining why an exception was thrown
Code	Optional. This is an integer that can be employed to differentiate this exception from other exceptions of the identical type
message	Optional. It is advisable to pass an exception into the message parameter if that exception gets thrown in a catch block of some other exception.

**Table 11.1: Parameter Values of Exception Object**

## Methods

Certain methods can be used to retrieve information about an exception while catching it.

Table 11.2 precisely describes some of these methods.

Method	Description
getPrevious ()	This method returns a previous exception if the exception gets activated by another one.
getFile ()	This method returns the complete path of a file in which the exception gets thrown.
getMessage ()	This method returns a string explaining the reason for throwing the exception.
getCode ()	This method returns the exception code.
getLine ()	This method returns the line number of the line of code that where the exception was thrown.

**Table 11.2: Methods in Exception Handling**

Code Snippet 18 shows a program that displays the output information about a thrown exception.

### Code Snippet 18:

```
<html>
<body>
<?php
function division($dividend, $divisor) {
    if($divisor == 0) {
        throw new Exception("Dividing by Zero", 1);
    }
    return $dividend / $divisor;
}
try {
    echo division(5, 0);
} catch(Exception $ex) {
    $code = $ex->getCode();
    $message = $ex->getMessage();
    $file = $ex->getFile();
    $line = $ex->getLine();
    echo "Observed an exception thrown in $file on line $line:
[Code $code]
$message";
}
?>
</body>
</html>
```

This code uses the methods `getCode()`, `getFile()`, `getLine()`, and `getMessage()` to output the desired information about the thrown exception as shown in Figure 11.18.



**Figure 11.18: Exception Thrown**

## 11.10 Summary

- PHP file handling system enables users to perform tasks including writing, appending, creating, closing, and deleting files.
- PHP has various functions such as `unlink()`, `fwrite()`, `fopen()`, `fclose()`, `fread()`, `fgets()`, `fgetc()`, and so on for file handling.
- It is essential to stay cautious while manipulating files. Any mistakes can lead to errors such as editing a wrong file, deleting a file's content, and filling a hard disk with garbage data.
- Error handling in PHP is a process of identifying errors in a code and tackling those errors attentively.
- Exception handling model in PHP offers better control over errors. The main keywords for exception handling are `try`, `throw`, `catch`, and `finally`.

## 11.11 Test Your Knowledge



1. Which of the following is the right way to open the file "mytext.txt" as readable?
  - a) fopen ("mytext.txt", "r");
  - b) fopen ("mytext.txt", "r+");
  - c) fopen ("mytext.txt", "read");
  - d) fopen ("mytext.txt");
2. A PHP function used to read specific number of characters from a file is \_\_\_\_\_.
  - a) filegets()
  - b) fget()
  - c) fgets()
  - d) fread()
3. Which PHP function specifies the last access time of a file?
  - a) filetime()
  - b) fileatime()
  - c) filectime()
  - d) None of these
4. Which of the following statements invoke the exception class?
  - a) new throws Exception();
  - b) New Exception();
  - c) throws new Exception();
  - d) throw new Exception();
5. Users can handle fatal errors by using \_\_\_\_\_ block.
  - a) try-catch
  - b) try-handle
  - c) catch-handle
  - d) try-throw

## **Answers to Test Your Knowledge**

---

1. `fopen("mytext.txt", "r");`
2. `fgets()`
3. `fileatime()`
4. `throw new Exception();`
5. `try-catch`

## **11.12 Try It Yourself**

1. Write a program to write a file named "phptest.txt" on the D drive of your machine. (If D does not exist, use an alternative drive letter)
2. Write a program to delete the file phptest.txt.
3. Write HTML and PHP code that creates an uploader form to enable end users to upload files of type txt.
4. Enclose the code created so far in appropriate try catch blocks.

## Session 12

# Object- Oriented Concepts in PHP



### Learning Objectives

*In this session, students will learn to:*

- Explain Object-Oriented Programming Features in PHP
- Explain about PHP Constructors and Destructors
- Describe PHP Traits
- Elaborate on PHP Namespaces and Iterables

This session describes OOP features in PHP and explores traits and namespaces as well.

### 12.1 Classes and Objects in PHP

Unlike procedural-oriented programming which gives importance to procedure or functions, Object-Oriented Programming (OOP) gives importance to objects. An object is nothing but data (properties) and functions (methods) that are combined together. OOP is preferred over traditional, structured, or procedural programming languages because of advantages such as reusability. Classes and objects are used to implement OOP principles.

#### **Class**

A class can contain variables (termed as properties), constants, and functions (termed as methods) that are designed to perform one or more tasks. In the real

world, to build a house, a blueprint is created first. The blueprint contains clear and detailed plans for the house. By using it, any number of similar houses can be built. Similarly, a class defined once can be used to create any number of objects. Hence, a class acts as a blueprint or template for the creation of objects.

## Objects

Objects are individual copies or instances of a class. Any number of objects can be created for a single class. Each object that is created will have the same type of properties and behaviors as that of a class. However, each object may have different values in the variables.

For example, once a blueprint is created, any number of copies can be created. Although the blueprint is the same for each house, there can be different paint colors, interior designs, and so on. Here, all these details are the objects.

Table 12.1 lists some examples of a class and the objects within it.

Class	Objects
<b>Animals</b>	Elephant Tiger Leopard
<b>Motorbikes</b>	Harley-Davidson BMW Ducati

*Table 12.1: Examples of Classes and Objects*

Table 12.2 lists and describes some important class and objected related terms in PHP.

Terminology	Description
<b>Member Properties</b>	Variables declared inside a class are called member properties or member variables. These are also called data members. Member variables can be accessed only through member functions and not outside the class directly. Inside objects, these variables are called attributes.

Terminology	Description
<b>Member Methods</b>	Functions or methods declared inside the class are called member functions or methods. These methods can access data members of the class.
<b>Inheritance</b>	Through inheritance, a class can reuse the properties of an already defined class. The class which reuses the properties is called a <b>sub class</b> or <b>derived class</b> or <b>child class</b> . The class from which the properties are reused is called a <b>super class</b> or <b>base class</b> or <b>parent class</b> .  A derived class can reuse either all or only the required properties from the parent class.
<b>Constructor</b>	This is a special type of function, which is invoked automatically at the time of object creation.
<b>Destructors</b>	This is a special type of function, which is used to delete the memory occupied by an object. It is invoked automatically when an object is deleted or goes out of scope.

*Table 12.1: Important Terms Used in OOP in PHP*

### Defining a Class in PHP

A class is created by using the keyword `class`, followed by the class name and a pair of curly brackets (`{ }`). Properties and methods of the class are given inside the brackets. A class can have any number of properties and methods.

Although the PHP OOP terminology states 'methods', the `function` keyword is used to declare class methods.

Following is a basic example of a class:

```
<?php
class City {
    // Properties
    public $name;
    public $country;
    // Methods
    function set_name($name) {
        $this->name = $name;
    }
}
```

```

function get_name() {
    return $this->name;
}
function set_country($country) {
    $this->name = $country;
}
function get_country() {
    return $this->country;
}
}
?>

```

Here, a class named `City` is declared. There are two properties defined here namely, `$name` and `$country`. Two methods each are declared for each property to set and retrieve their values.

Code Snippet 1 shows the complete example of creating the class and its objects.

### **Code Snippet 1:**

```

<html>
<body>
<?php
class City {
    // Properties
    public $name;
    public $country;
    // Methods
    function set_name($name) {
        $this->name = $name;
    }
    function set_country($country)
{
    $this->country=$country;
}
    function get_country() {
        return $this->country;
    }

    function get_name() {
        return $this->name;
    }
}
// creating object of class

```

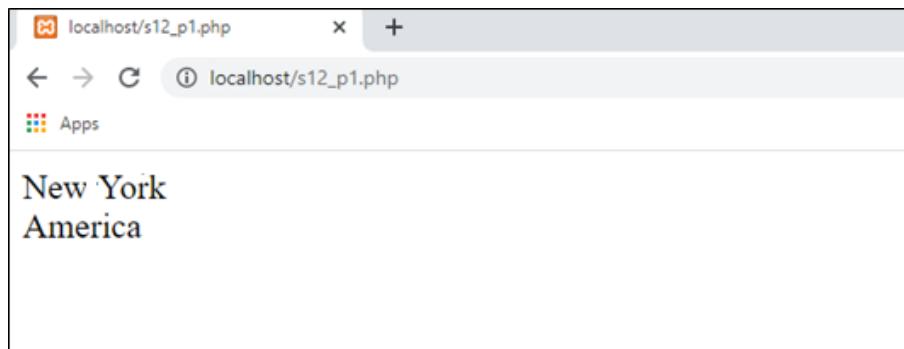
```

$NewYork = new City();
$America = new City();
$NewYork->set_name('New York');
$America->set_country('America');
echo $NewYork->get_name();
echo "<br>";
echo $America->get_country();
?>
</body>
</html>

```

In Code Snippet 1, a class named `City` is created. Four methods are written inside the class. Two of these methods are given to set the name and country properties of the class respectively, that is, `set_name()` and `set_country()`. The other two methods are designed to retrieve name and country values respectively, that is, `get_name()` and `get_country()`. Objects of a class are created using the `new` keyword. `New York` and `America` are instances or objects of class `city`. Class methods are invoked using arrow operator.

Figure 12.1 shows the output for Code Snippet 1.



*Figure 12.1: Output for Code Snippet 1*

## 12.2 Objects Creation in PHP

Once a class is defined, any number of objects can be created for the class type. To create objects, `new` operator should be used.

Following are some examples showing the usage of `new` operator to create the objects:

```

$physics = new Books;
$maths = new Books;
$chemistry = new Books;

```

Here, the user has created three objects for the class Books and these objects are not dependent on each other. Also, these objects will exist in separate memory locations. Note that here the parentheses after class name is omitted. This is allowed as they are optional.

## PHP – \$this Keyword

\$this is a reserved keyword (also referred to as pseudo variable) that is used to access the current object. Pseudo-types are keywords used in the PHP to specify the types or values an argument can have.

\$this keyword is only available inside a method and refers to the object of the current method. Code Snippet 2 shows use of \$this keyword inside a class method.

### Code Snippet 2:

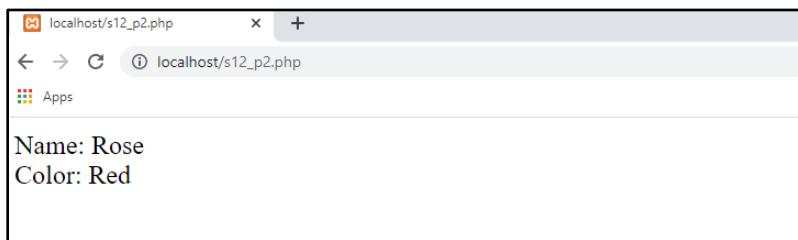
```
<html>
<body>
<?php
class Flower {
    // Properties
    public $name;
    public $color;
    // Methods
    function set_name($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
    function set_color($color) {
        $this->color = $color;
    }
    function get_color() {
        return $this->color;
    }
}
$Rose = new Flower();
$Red=new Flower();
$Rose->set_name('Rose');
$Red->set_color('Red');
echo "Name: " . $eRose->get_name();
echo "<br>";
```

```
echo "Color: " . $Red->get_color();
?>
</body>
</html>
```

In Code Snippet 2, `$this` keyword is used. It refers to the current object and is only available inside methods. After creating objects `$Rose` and `$Red` of class `Flower`, method `set_name()` is called using object `$Rose`. Then, it will be passed to `set_name()` method, where value is assigned to `$name` variable as `$Rose` using `$this` keyword.

Next line of the code is used outside the class where value of name is displayed by calling method `get_name()` using same object `$Rose`. This will then call the second method of a class `get_name()` which returns value of name using keyword `$this`.

Figure 12.2 shows the output for Code Snippet 2.



*Figure 12.2: Output for Code Snippet 2*

Following are the two ways of using `$this` keyword:

1. **Outside the class:** User can directly change the value of a property from outside the class.

Code Snippet 3 shows declaration of property outside the class.

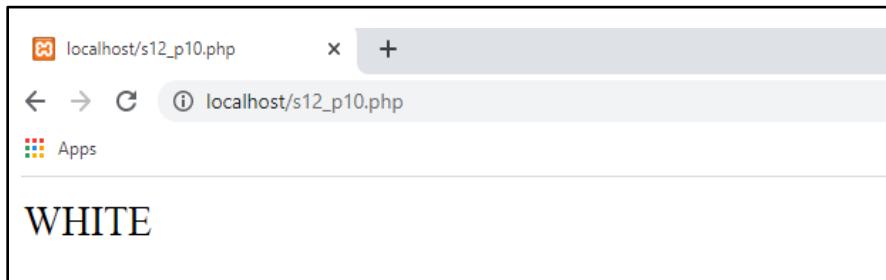
### **Code Snippet 3:**

```
<html>
<body>
<?php
class House {
    public $color;
}
$white = new House();
$white->color = "WHITE";
```

```
echo $white->color;  
?>  
</body>  
</html>
```

In Code Snippet 3, value of property `color` is being assigned outside the class. The `color` property value is assigned as `WHITE` using object `$white` of class `House`. No methods are declared here.

Figure 12.3 shows the output for Code Snippet 3.



*Figure 12.3: Output for Code Snippet 3*

2. **Inside the class:** Users can define a method in the class and then, call it to change the value of its own property.

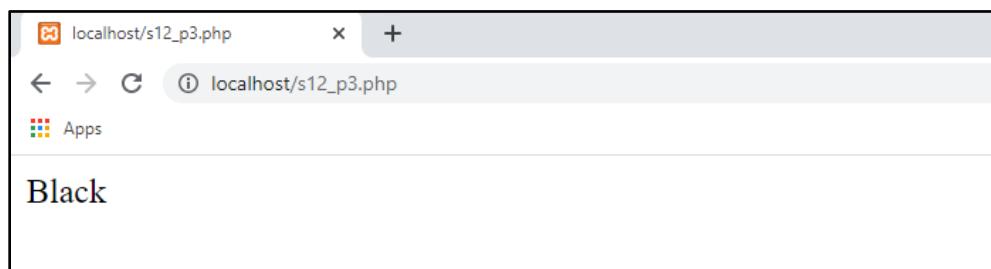
Code Snippet 4 shows declaration of a property inside the class.

#### **Code Snippet 4:**

```
<html>  
<body>  
<?php  
class House {  
    public $color;  
    function set_color($color) {  
        $this->color = $color;  
    }  
}  
$Black = new House();  
$Black->set_color("Black");  
echo $Black->color;  
?>  
</body>  
</html>
```

In Code Snippet 4, method `set_color()` is declared inside the class `House` and creating the object '`Black`' of class `House`. Using that object, user calls `set_color()` function and prints the value of `color`. Arrow (`->`) operator is used to call a method or value using objects.

Figure 12.4 shows the output for Code Snippet 4.



*Figure 12.4: Output for Code Snippet 4*

### **PHP – `instanceof` keyword**

PHP uses a keyword `instanceof` to check if an object belongs to a particular class. Code Snippet 5 shows the use of `instanceof` keyword in a script.

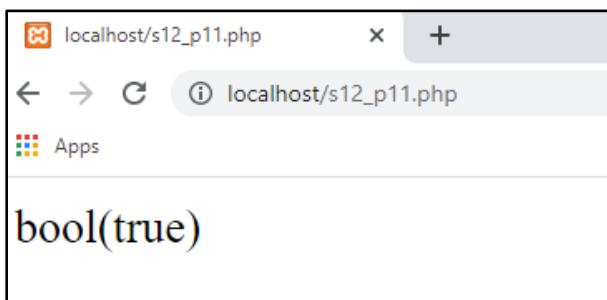
#### **Code Snippet 5:**

```
<html>
<body>
<?php
class City {
    // Properties
    public $name;
    public $country;
    // Methods
    function set_name($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
}
$London = new City();
var_dump($London instanceof City);
?>
</body>
</html>
```

In Code Snippet 5, keyword `instanceof` is used to know whether an object is of specific class type. This can be very useful in large applications that may have several classes and more than one developer working on the scripts.

Here in the code, `var_dump()` function is used to return type of the object. By using a combination of `var_dump()` and `instanceof`, user can check whether `$London` object belongs to `City` class or not. The statement returns bool value `true` here because object does belong to `City` class.

Figure 12.5 shows the output for Code Snippet 5.



*Figure 12.5: Output for Code Snippet 5*

## 12.3 PHP OOP – Constructor and Destructor

---

Once objects are declared, it is recommended that each and every member be initialized. Additionally, once the objects go out of scope, memory should be cleared. To perform these operations, special methods called constructors and destructors are used. Following are their purposes:

### Constructors

Constructors are special member methods invoked automatically when an object is created.

### Destructors

Destructors are special member methods invoked automatically when an object is destroyed or an object goes out of scope. Usually, it is invoked at the end of the script.

#### 12.3.1 Constructor Method

A constructor is usually declared as public and named as `__construct`. This method should start with two underscores (`__`).

Following is a basic code showing how to declare a constructor:

```
class Example {
```

```

public function __construct() {
    // constructor function
    echo "Hello World";
}
}

```

The `__construct()` method will be called automatically when an object is created for the class.

Code Snippet 6 shows the use of the `__construct()` method in a class.

### **Code Snippet 6:**

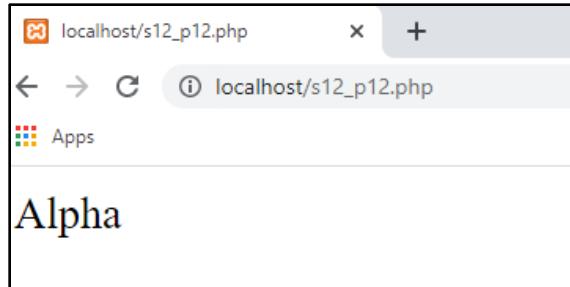
```

<html>
<body>
<?php
class Example {
    public $name;
    public $color;
    function __construct($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
}
$alpha = new Example("Alpha");
echo $alpha->get_name();
?>
</body>
</html>

```

In Code Snippet 6, using constructors saves time from calling the `set_name()` method thereby, reducing the amount of code. If user creates a `construct()` method, PHP will automatically call this method when an object is created from a class. Here, `$alpha` object automatically sets the value `Alpha` without calling the method `set_name()`. It is only required to call `get_name()` method while displaying the value.

Figure 12.6 shows the output for Code Snippet 6.



**Figure 12.6: Output for Code Snippet 6**

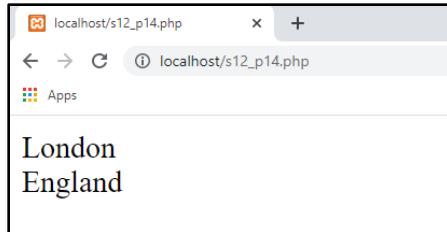
Code Snippet 7 shows constructors being used with two parameters.

### **Code Snippet 7:**

```
<html>
<body>
<?php
class City {
    public $name;
    public $country;
    function __construct($name, $country) {
        $this->name = $name;
        $this->country = $country;
    }
    function get_name() {
        return $this->name;
    }
    function get_country() {
        return $this->country;
    }
}
$london = new City("London", "England");
echo $london->get_name();
echo "<br>";
echo $london->get_country();
?>
</body>
</html>
```

In Code Snippet 7, constructor with two properties or parameters is used, that is, `name` and `country`. Therefore, values are displayed using object `London` of class `City`, by calling the methods `get_name()` and `get_country()`.

Figure 12.7 shows the output for Code Snippet 7.



*Figure 12.7: Output for Code Snippet 7*

### 12.3.2 Destructor Method

A destructor is called when an object is to be removed or has gone out of scope. As with constructor, a destructor too should start with two underscores (\_). If an object contains `__destruct()` method, it will be automatically called by PHP at the end of the script.

Code Snippet 8 shows the use of destructor in a class.

#### Code Snippet 8:

```
<html>
<body>
<?php
class Flower {
    public $name;
    public $color;
    function __construct($name) {
        $this->name = $name;
    }
    function __destruct() {
        echo "The flower is {$this->name} .";
    }
}
$Lotus = new Flower("Pink");
?>
</body>
</html>
```

In Code Snippet 8, the constructor is called automatically when the object is created. However, the destructor is called at the end of code when the object goes out of scope. Within the destructor method `__destruct()`, the value of property `name` is displayed using `this` keyword. It refers to the current object (`$Lotus`) of class `Flower` and prints the value as `Pink`.

Figure 12.8 shows the output for Code Snippet 8.



*Figure 12.8: Output for Code Snippet 8*

## 12.4 PHP OOP – Traits

---

A class in PHP supports only single inheritance. A child class can inherit only from one single parent. In large applications, instead of creating everything from scratch, it makes sense to reuse already defined classes or types and build upon their functionality further. One of the biggest advantages OOP offers is that of reusability. To inherit multiple behaviors, PHP uses a concept called traits.

A trait is similar to a class and can contain constants, properties, and methods. It can be used to group functionality in a single form or structure.

Following is the syntax used to declare a trait:

### Syntax:

```
<?php
trait TraitName {
    // some code...
}
?>
```

A user cannot instantiate a trait on its own. It has to be used within a class with the help of keyword `use`. Following is the syntax for `use` keyword:

### Syntax:

```
<?php
class ClassName {
    use TraitName;
}
?>
```

Code Snippet 9 shows an example of how to use `trait` keyword in a class.

## Code Snippet 9:

```
<html>
<body>
<?php
trait T1 {
    public function msg1() {
        echo "This is an example of a trait!";
    }
}
class Message {
    use T1;
}
$obj = new Message();
$obj->msg1();
?>
</body>
</html>
```

The output will display "This is an example of a trait!" in the browser.

In Code Snippet 9, a trait is `T1` declared with keyword `trait`. Its definition contains a method named `msg1()`. This trait can now be inherited by any class with the keyword `use`. This is done in class `Message`. Later, `msg1()` method is called using object `$obj` of class `Message` without having declared `msg1()` method inside the class `Message`. Thus, `Message` has inherited `msg1()` method from the trait `T1`. Hence, when `msg1()` method is called, it will print the output.

A class uses a trait so that the members defined in the trait can be inherited by the class. Code reusability is promoted through usage of traits.

Following are some of the features of traits:

With traits, users can declare methods that can be used in more than one class.

By using traits, code redundancy can be reduced.

You cannot instantiate a trait on its own. Objects cannot be created.

Traits are allowed to have methods and abstract methods in any visibility mode (public, private, or protected).

Traits are declared same as that of method, but with the `trait` keyword as prefix.

## PHP – Using Multiple Traits

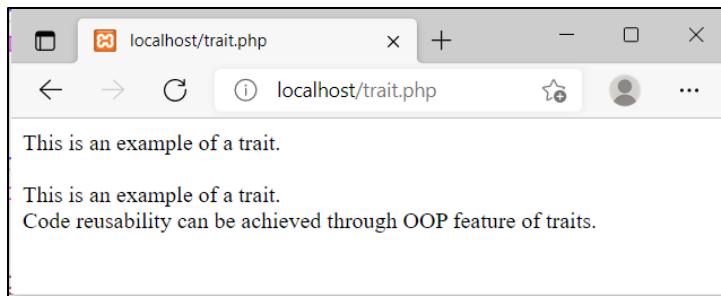
Code Snippet 10 shows usage of multiple traits in a class.

### Code Snippet 10:

```
<html>
<body>
<?php
trait T1 {
    public function msg1() {
        echo " This is an example of a trait! <br>";
    }
}
trait T2 {
    public function msg2() {
        echo "Code reusability can be achieved through OOP
feature of traits!";
    }
}
class Message1 {
    use T1;
}
class Message2 {
    use T1, T2;
}
$obj1 = new Message1();
$obj1->msg1();
echo "<br>";
$obj2 = new Message2();
$obj2->msg1();
$obj2->msg2();
?>
</body>
</html>
```

In Code Snippet 10, two traits are declared, namely `T1` and `T2` in two classes, namely `Message1` and `Message2`. The `msg1()` method is called by using object `obj1` of a class `Message1`. By using object `obj2` of a class `Message2`, both the methods `msg1()` and `msg2()` are being called. In Code Snippet 10, two traits `T1` and `T2` are used in a class `Message2`.

Figure 12.9 shows the output for Code Snippet 10.



*Figure 12.9: Output for Code Snippet 10*

## 12.5 PHP Namespaces

---

PHP allows having multiple classes with the same name by using Namespaces, which is a qualifier. It helps in better organization of code and allows to use same name for more than one class.

Following are some different explanations for it:

### Explanation 1

Assume that a program has a global variable named `$title`, inside a method. This program has another variable with the same name `$title`. If user assigns and changes values of `$title` inside of the method, the global variable is not affected. It will remain unchanged. This is called scope of a variable. Following the same method, two classes can be declared with same name to give it scope.

### Explanation 2

Assume that a user is creating an open-source PHP library to send mails and the user shares that with the developer community. The library of the user has a class named `Email`. If a developer who already has a class called `Email` downloads this library, name collision/conflict will occur. The user can now either rename those classes or use Namespacing.

### Explanation 3

Assume that there are two people with the name 'John'. To separate them, their surname can be used. In the same way, two classes having the same name can be separated by Namespacing.

By using namespaces two different problems are solved:

1. They group the classes that work together to perform a task. This allows the user to organize code in a better manner.

2. The same name can be used for more than one class, therefore, avoiding dilemma among names.

For example, consider the set of classes describing an HTML table, such as Table, Row, and Cell. Consider having another set of classes to describe furniture, such as Table, Chair, and Bed. By using Namespaces, these classes can be organized into two different groups while also preventing the two different classes with the same name Table, from being mixed up.

### **Requirement of Namespaces in PHP**

PHP programs cannot have two classes with the same name. For example, if a user has the requirement for two classes to handle blog users and app users, then those classes should have distinct names such as `Blog_User` and `App_User`. However, remembering these types of prefixes can be confusing.

Namespaces help the user to simplify and organize the process and give more control over the program.

### **Declaring a Namespace**

Namespaces are declared using the keyword `namespace` at the beginning of a file, that is, as the first statement in the PHP file. Following is the syntax to use while declaring a namespace:

#### **Syntax:**

```
<?php  
echo "Hello World!";  
namespace Html;  
...  
?>
```

Code Snippet 11 shows use of Namespace in a class.

#### **Code snippet 11:**

```
<?php  
namespace PHP;  
class Table1 {  
    public $title = "";  
    public $numRows = 0;  
    public function msg() {
```

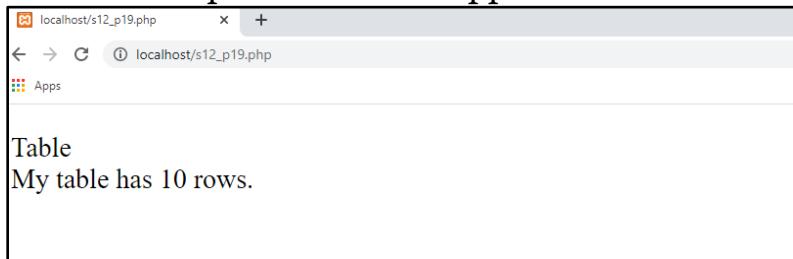
```

        echo "<p>Table {$this->title} has {$this->numRows}<br>
rows.</p>";
    }
}
$stable1 = new Table1();
$stable1->title = "<br>My table ";
$stable1->numRows = 10;
?>
<!DOCTYPE html>
<html>
<body>
<?php
$stable1->msg ();
?>
</body>
</html>

```

In Code Snippet 11, a namespace `PHP` is used and a method is declared inside the namespace. The property `title` is assigned with a dummy value. However, in the later part of the code, the code is assigning same variable `title` value as `My table` contains using object `table1` of class `Table1`.

Figure 12.10 shows the output for Code Snippet 10.



*Figure 12.10: Output for Code Snippet 11*

### Using namespaces

Any code following a namespace declaration will operate inside the namespace. Therefore, classes belonging to the namespace can be instantiated without any qualifiers. If the user wants to access classes from outside of a namespace, the class should have the namespace attached to it.

Code Snippet 12 shows the use of namespaces in two different PHP codes, first with `index.php` and then, with `Html.php`.

## Code Snippet 12: index.php

```
<?php
include "HTML.php";
$table = new Html1\Table1();
$table->title = "<br>My table";
$table->numRows = 7;
$row = new Html1\Row();
$row->numCells = 4;
?>
<html>
<body>
<?php $table->msg(); ?>
<?php $row->msg(); ?>
</body>
</html>
```

In this part of Code Snippet 12, another PHP code is included using keyword `include` as `include "HTML.php"`. The namespace `Html` and class name `Table1` object `$table` are created. Next, the code is assigning value to a `title` property by using an object. `title` property is declared in class `Table1`. Values are assigned to variables `numRows` and `numCells` as 7 and 3 using objects `$table` and `$row`.

`HTML.php`

```
<?php
namespace Html1;
class Table1 {
    public $title = "";
    public $numRows = 0;
    public function msg() {
        echo "<p>Table {$this->title} has {$this->numRows} rows.</p>";
    }
}
class Row {
    public $numCells = 0;
    public function msg() {
        echo "<p>The row has {$this->numCells} cells.</p>";
    }
}
?>
```

In this part of Code Snippet 12, namespace `Html1` and classes `Table1` and `Row` are declared. In these classes, you are declaring properties `title` and `numRows`. In the class `Row`, the code is declaring `$numCells`. Next, the code is displaying the value of `title` and `numRows` in a class `Table1` and value of `numCells` in class `Row`.

Figure 12.11 shows the output for Code Snippet 12.

```
<html>
<body>
<?php
function displayIterable(iterable $testIterable) {
    foreach($testIterable as $item) {
        echo $item;
    }
}
$arr1 = ["1", "2", "3"];
displayIterable($arr1);
```

*Figure 12.11: Output for Code Snippet 12*

Finally, the output is obtained as shown in Figure 12.11. `index.php` code has to be run to view the output.

It becomes easier to use `namespace` keyword when many classes from the same namespace are being used at the same time.

## 12.6 PHP Iterables

---

An `iterable` in PHP is a parameter type that can be looped with a `foreach` loop. `iterable` can be used as a data type for function arguments and also as the return values of a function.

Code Snippet 13 shows use of `iterable` in a `foreach` loop.

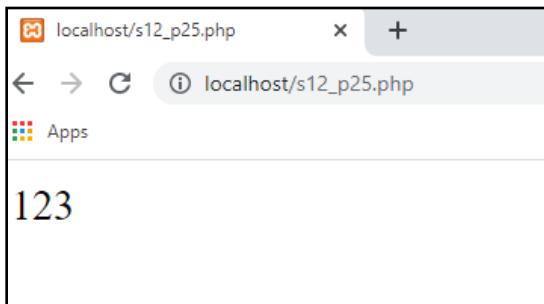
### Code Snippet 13:

```
<html>
<body>
<?php
function displayIterable(iterable $testIterable) {
    foreach($testIterable as $item) {
        echo $item;
    }
}
$arr1 = ["1", "2", "3"];
displayIterable($arr1);
```

```
?>
</body>
</html>
```

In Code Snippet 13, `iterable` keyword is used in function `displayIterable()` to declare `testIterable` is declared of `iterable` type. A `foreach` loop is used to traverse through each value of `testIterable`. Later, an array is defined and passed as argument to function `displayIterable()` call.

Figure 12.12 shows the output for Code Snippet 13.



*Figure 12.12: Output for Code Snippet 13*

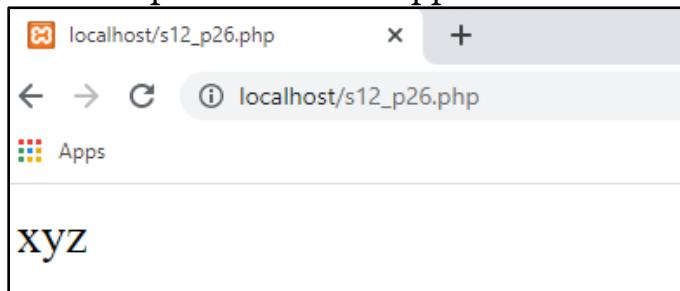
Code Snippet 14 shows use of `Iterable` keyword with `return` keyword inside the function.

#### **Code Snippet 14:**

```
<html>
<body>
<?php
function getIterable():iterable {
    return ["x", "y", "z"];
}
$testIterable = getIterable();
foreach($testIterable as $alpha) {
    echo $alpha;
}
?>
</body>
</html>
```

In Code Snippet 14, `return` keyword is used inside the function `getIterable()` with the keyword `:iterable` to return values 1, 2, and 3. The code is assigning the function `getIterable()` to variable `$testIterable` and using it in `foreach` loop. The value is displayed by `$num` variable.

Figure 12.13 shows the output for Code Snippet 14.



**Figure 12.13: Output for Code Snippet 14**

## Arrays

All arrays are iterables. Therefore, any array can be used as an argument of a method. However, it requires an iterable.

## Iterators

PHP contains an interface called `Iterator`, which can be used as an argument of a function that requires an `iterable`.

An iterator contains a list of items and provides functions to loop through. It also contains a pointer to one of the elements in the list. Each item in the list should have a key, through which the item can be located.

An iterator should have following methods:

current()	It returns the element that the pointer is currently pointing to. It can be of any data type.
key()	It returns the key associated with the current element in the list. The data type of the key can only be an integer, float, Boolean, or string.
next()	It moves the pointer to the next element in the list.
rewind()	It moves the pointer to the beginning (first element) of the list.
valid()	If the internal pointer is operated to point to an invalid element of the list (for example, if <code>next()</code> was called at the end of the list), this function should return false. In any other case, it returns true.

Code Snippet 15 shows implementation of Iterator in a class and pointers inside the function.

### Code Snippet 15:

```
<html>
<body>
<?php
// Create an Iterator
class TestIterator implements Iterator {
    private $alpha = [];
    private $pointer = 0;
    public function __construct($alpha) {
        $this->alpha = array_values($alpha);
    }
    public function current() {
        return $this->alpha[$this->pointer];
    }
    public function key() {
        return $this->pointer;
    }
    public function next() {
        $this->pointer++;
    }
    public function rewind() {
        $this->pointer = 0;
    }
    public function valid() {
        return $this->pointer < count($this->alpha);
    }
}
// A function that uses iterables
function printIterable(iterable $testIterable) {
    foreach($testIterable as $alpha) {
        echo $alpha;
    }
}

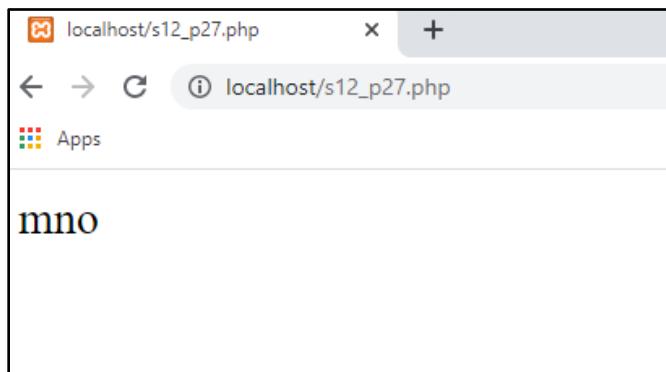
// Use the iterator as an iterable
$iterator = new TestIterator(["m", "n", "o"]);
printIterable($iterator);
?>
</body>
</html>
```

In Code Snippet 15, `iterable` is implemented to a class `TestIterator`. Two variables `$num` and `$pointer` are declared inside the class and assigning pointer

equal to 0. Then, the `constructor` function is declared inside which values of `num` are assigned using `this` keyword.

The functions `current()`, `key()`, `next()`, `rewind()`, and `valid()` are defined. Each function carries out a particular task. Currently, the pointer value is set to 0, which will point to number 1 of function argument. `key()` function will point to pointer, `next()` function will point to next element or number 2, `rewind()` function will again point to pointer as 0, and `valid()` will count numbers present by using `count()` function. Outside the class, the code is creating the object of class `Test iterable` and calling the function `print Iterable` by passing object as argument. `print Iterable()` function is using `iterable` type variable, which is used in `foreach` loop to print numbers.

Figure 12.14 shows the output for Code Snippet 15.



*Figure 12.14: Output for Code Snippet 15*

## 12.7 Summary

- Object-Oriented Programming (OOP) is a programming paradigm that is mainly based on the use of classes and objects.
- A class is a data type that can contain constants, variables (properties'), and functions (methods).
- An object is an instance of a class and is created using new operator.
- A class is created by using the keyword `class`, followed by the class name, and a pair of curly brackets (`{ }`).
- `$this` is a reserved keyword used to access the current object.
- PHP makes use of `instanceof` keyword to check if an object belongs to a particular class.
- Constructors and destructors are special member functions.
- A constructor is invoked automatically when an object is created.
- A destructor is invoked automatically when an object is destroyed or goes out of scope.
- In PHP, the traits concept is used to inherit multiple behaviors.
- Namespaces are qualifiers that help to organize code better and use the same name for multiple classes.

## 12.8 Test Your Knowledge



1. Which of the following is used to execute code automatically when a new instance of a class is created?
  - a. Destructor
  - b. Constructor
  - c. Friend
  - d. Initial
  
2. Which of the following keywords can be used to create objects from a class that has been defined?
  - a. New object
  - b. construct
  - c. new
  - d. Both a and c
  
3. Within a namespace, which of the following is used as a prefix to access the built-in PHP classes?
  - a. percent
  - b. ampersand
  - c. asterix
  - d. backlash
  
4. \_\_\_\_\_ can be written at a start of a class name and can be followed by numerous letters, periods, or underscores.
  - a. Letter
  - b. Name
  - c. Underscore
  - d. Asterix
  
5. Though multiple traits cannot be utilized in the same class, multiple interfaces can be used. (State True/False).
  - a. True
  - b. False

## **Answers to Test Your Knowledge**

---

1. Constructor
2. New
3. Percent
4. Letter
5. False

## **12.9 Try it Yourself**

1. Write a program to declare a class and create an instance of that class.
2. Write a program to create an object of the class using `this` keyword.

# Session 13

## Methods in PHP and Other OOP Concepts



### Learning Objectives

*In this session, students will learn to:*

- Explain calling of member methods
- Outline public, private, and protected members
- Explain how to use method overriding
- Elaborate the scope of methods within a class
- Explain inheritance, interfaces, and abstract classes
- Outline static and final keyword
- Explain static method
- Describe method overloading

In PHP OOP terminology, class member functions are called 'methods'. The function keyword is used to declare class methods. The session explains how to call member methods. The session further describes the concept of inheritance and explores public, private, and protected members in classes. Next, one will learn about method scope, interfaces, and abstract classes. The session further explains about `static` and `final` keywords. This session will also elaborate on method overloading.

### 13.1 Calling Member Methods

OOP is a paradigm based on the concept of objects that help in building complex and reusable applications using data and code. The key object-oriented concepts in PHP are class, object, inheritance, polymorphism, overloading, data abstraction, constructors, and destructors. An object is created in PHP using the `new` keyword.

Once the objects are created, the variables and member methods of the class can be accessed using the operator `->`. Member functions are accessed to get information on the object properties.

Following example shows how object properties are accessed through the member methods:

**Example:**

```
...
$lion->setTitle(" Alex in the Madagascar ");
$tiger->setTitle(" Joshua in the central zoo ");
$zebra->setTitle(" Marty in the central zoo ");
$lion->setQuantity(150);
$tiger->setQuantity(100);
$zebra->setQuantity(550);
```

Here, member methods are called to set the title and quantity of three animal-based books. In this example, using the `setTitle()` method of class `Animal`, the title variable can be set. Using the `setQuantity()` method of class `Animal`, the quantity can be set.

OOP has certain keywords called access modifiers that decide the accessibility for various class methods, variables, and other member methods. Some PHP keywords represent the access modifiers and are used to set access rights with their members and PHP classes. PHP access modifiers can be classified based on application with class, variables, and methods. Table 13.1 lists different access modifiers and their applicability for methods, variables, and classes.

Modifier	Methods	Variables	Classes
<code>public</code>	Applicable	Applicable	Not Applicable
<code>private</code>	Applicable	Applicable	Not Applicable
<code>protected</code>	Applicable	Applicable	Not Applicable
<code>abstract</code>	Applicable	Not Applicable	Applicable
<code>final</code>	Applicable	Not Applicable	Applicable

*Table 13.1: List of PHP Access Modifiers*

Code Snippet 1 shows how the member methods are called to get values. The code in the example shown previously has been used here.

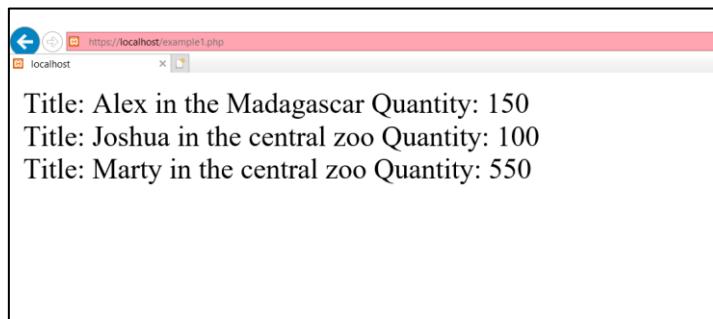
## Code Snippet 1:

```
<?php
class AnimalBooks {
    // Properties
    public $title;
    public $quantity;
    // Methods
    function setTitle($title) {
        $this->title = $title;    }
    function getTitle() {
        return $this->title;    }
    function setQuantity($quantity) {
        $this->quantity = $quantity;
    }
    function getQuantity() {
        return $this->quantity;
    }
}
$lion = new AnimalBooks();
$tiger = new AnimalBooks();
$zebra = new AnimalBooks();
$lion->setTitle("Alex in the Madagascar");
$tiger->setTitle("Joshua in the central zoo");
$zebra->setTitle("Marty in the central zoo");
$lion->setQuantity(150);
$tiger->setQuantity(100);
$zebra->setQuantity(550);
$lion->getTitle();
$tiger->getTitle();
$zebra->getTitle();
$lion->getQuantity();
$tiger->getQuantity();
$zebra->getQuantity();
echo " Title: " . $lion->getTitle();
echo " Quantity: " . $lion->getQuantity();
echo "<br>";
echo " Title: " . $tiger->getTitle();
echo " Quantity: " . $tiger->getQuantity();
echo "<br>";
echo " Title: " . $zebra->getTitle();
echo " Quantity: " . $zebra->getQuantity();
echo "<br>";
?>
```

In Code Snippet 1, access modifiers are used. Here, when defining the class members, `public` access modifier is used so that a user can access at class level. Member variables `title` and `quantity` of class `Animal` are declared

with public. To change the behavior of a class, access modifiers can be assigned to the class itself.

Figure 13.1 shows the output for Code Snippet 1.



*Figure 13.1: Output for Code Snippet 1*

## 13.2 Inheritance

---

Inheritance is an important principle in OOP, as it enables a class to use methods and properties of other classes. In programming, there is always a requirement to create a new class with functionalities of other or existing classes. In such situations, you can either copy or inherit all the methods and properties of the existing class into another class.

Inheritance is beneficial for reusability when creating multiple similar classes. Instead of creating classes from scratch, user can build upon existing classes and extend their functionality. The inherited class is called the Parent class (super or base class) and the inheriting class is called Child Class (Sub or derived class). The common properties and methods in the parent class can be inherited into the child class, based on access modifiers.

Following are the things to remember while using inheritance:

- The child class has access to only the non-private methods and properties on the parent class.
- The methods of the child class are not available to the parent class.
- The methods defined in the parent class can be overridden by the child class with its own implementation.

In PHP, the user can use `extends` keyword to specify name of the parent class while defining the child class. Following is the syntax used to inherit a class:

**Syntax:**

```
class <Child> extends class <Parent>
```

where, `Parent` is an existing class.

Following example shows `extends` keyword being used to inherit a class:

### Example:

```
...
    class Bank {
        // parent class code
    }
    class Transaction extends Bank {
        // child class code
    }
...
...
```

In this example, `extends` keyword is used to specify name of the parent class while defining the child class. Here, `Bank` class is the parent or base class and `Transaction` class is the child or derived class. The `Bank` class is inherited by the `Transaction` class. Thus, any object of `Transaction` class has all the properties and methods of `Bank` class well as any members defined within `Transaction`. This is the power and advantage of inheritance.

Code Snippet 2 shows an example with class `BankAccount` containing basic functions `debit()` and `credit()`. Child classes `Currentacct` and `Savings` are created extending the parent class `BankAccount`.

### Code Snippet 2:

```
<?php
    // parent class
    class BankAccount {
        // public property type
        public $type;
        // public function credit
        public function credit() {
            echo $this->type. " CREDIT transaction in
progress...<br/>";
        }
        // public function debit
        public function debit() {
            echo $this->type. " DEBIT transaction in
progress...<br/>";
        }
    }
    // child class
    class Savings extends BankAccount {
        // No code in child class
    }
    // child class
    class Currentacct extends BankAccount {
        // No code in child class
    }
}
```

```

// Instantiate the derived classes to create objects
$savings = new Savings();
$savings->type = "Salary Savings Account ";

$currentacct = new Currentacct();
$currentacct->type = "Trading Current Account ";

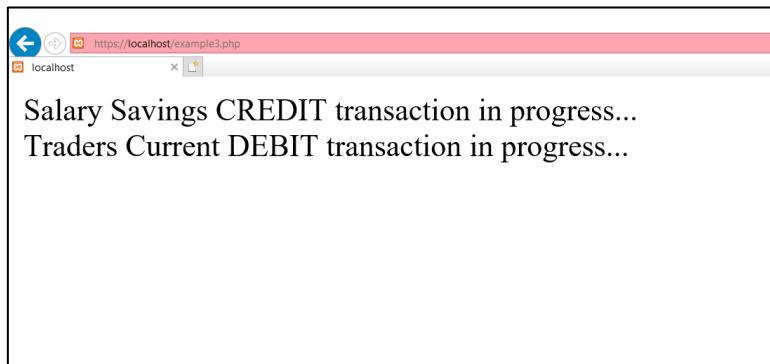
// call class methods
$savings->credit();
$currentacct->debit();

?>

```

In Code Snippet 2, the child classes `Savings` and `Currentacct` inherit from the `BankAccount` class. This allows them to access public properties and methods of parent class.

Figure 13.2 shows the output for Code Snippet 2.



**Figure 13.2: Output for Code Snippet 2**

### **Child Class with its Own Methods and Properties**

A child class can access the non-private members of the parent class. A child class can have its own member properties and methods.

Code Snippet 3 shows an example of a child class defining and accessing its own methods and properties.

### **Code Snippet 3:**

```

<?php
    // parent class
    class Television {
        // public property name
        public $name;

        // public function transmit_sound
        public function transmit_sound() {

```

```

        echo $this->name. " - does transmit sound
        successfully <br/>";
    }
    // public function transmit_video
    public function transmit_video() {
        echo $this->name. " - does transmit video
        successfully...<br/>";
    }
}
// child class
class Lcdtv extends Television {

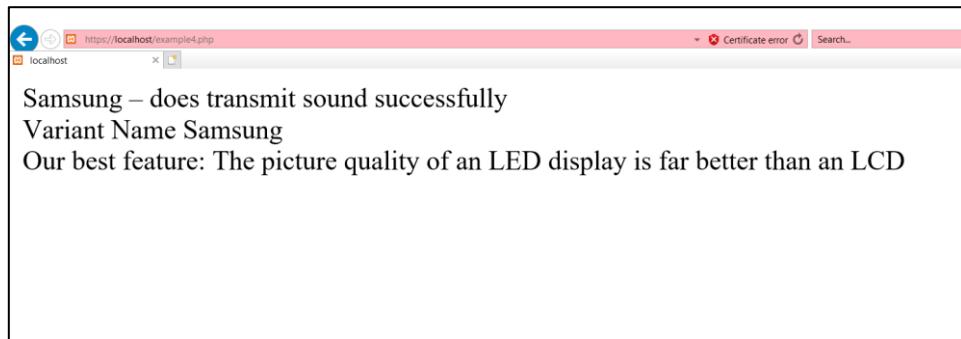
    public function control_feature() {
        echo "Manufacturer Name: " . $this->name.
        "<br/>";
        echo " Our best feature: Consuming much less
            power than plasma displays<br/>";
    }
}
// child class
class Ledtv extends Television {
    // specific category of Television
    // and methods
    public function control_feature() {
        echo "Manufacturer Name: " . $this->name.
        "<br/>";
        echo " Our best feature: The picture quality of
            an LED display is far better than an LCD<br/>";
    }
}
$ledtv = new Ledtv();
$ledtv->name = "Samsung";
// calling parent class method
$ledtv->transmit_sound();
// calling child class method
$ledtv->control_feature();
echo "<br/>";

$Lcdtv = new Lcdtv();
$Lcdtv->name = "Sony";
// calling parent class method
$Lcdtv->transmit_sound();
// calling child class method
$Lcdtv->control_feature();
?>

```

In Code Snippet 3, class `Television` is the parent or base class. There are two sub classes `Ledtv` and `Lcdtv` which inherit from `Television` class.

Figure 13.3 shows the output for Code Snippet 3.



**Figure 13.3: Output for Code Snippet 3**

The properties and methods defined in the parent class can be used by the child class without defining them repeatedly. Additionally, the child class can have its own set of methods and properties similar to any other class. This is possible even though it inherits the properties of the parent class.

After understanding what inheritance is, one can gain clarity on the working of access modifiers such as `public`, `private`, and `protected`.

### **13.3 Public Members**

---

The classes and their members are treated public, unless specified. For better understanding, it is a good practice to specify the classes and their members as public.

By defining the classes and their members as public, they can be accessed from any of the following:

- Outside the scope of a class
- Within the scope of a class
- Another class within the declared class

### **13.4 Private Members**

---

A class can be defined as private or protected to limit the accessibility of the class members. Private class members can be accessed from within the class. It means a private member can neither be accessed from outside the class in which it is declared nor the inherited class.

Code Snippet 4 demonstrates private members.

#### Code Snippet 4:

```
<?php
// Define the base class
class Footwear {
    private $price = "We have a fixed price of 3000";
    private function show()
    {
        echo "This is a private method of a Footwear(base
class)";
    }
}

// Define the derived classes
class Sneakers extends Footwear {
    function printPrice()
    {
        echo $this->price;
    }
}

// Create the object of the derived class
$obj= new Sneakers;

// The given line is trying to call a private method. Hence,
// trying to access private function show() throws error
$obj->show();

// The given line also throws error since $price is private
// member of parent class Footwear
$obj->printPrice();

?>
```

In this code, base class `Footwear` and child class `Sneakers` are created. In the base class, `$price` and `show()` are private to the base class. `$obj->show()` and `$obj->printPrice()` are ways of accessing private variables.

### **13.5 Protected Members**

A protected member is similar to a private member. Protected members can be accessed from the classes that are derived from a parent class. They can also be accessed in the subclasses in which they are declared. The class

member is made protected by adding the `protected` keyword and is not available outside the two kinds of classes.

Code Snippet 5 shows how a class member is declared protected by using the `protected` keyword.

### Code Snippet 5:

```
<?php
// Define the base class
class Footwear {
    protected $price1 = 5000;
    protected $price2 = 15000;
    function total() {
        echo $sum = $this->price1 + $this->price2;
        echo "<br>";

    }
}
// Define the derived classes
class Sneakers extends Footwear {
    function printBill() {
        $tax = 100;
        //Calculating the total + tax
        echo $sub = $this->price1 + $this->price2 + $tax;
        echo "<br>";
    }
}
$obj= new Sneakers;
$obj->total();
//This code accesses the function printBill which has
calculation with protected members of base class namely
$price1 and $price2
$obj->printBill();
?>
```

In this code, `$price1` and `$price2` are variables that are declared with `protected` access specifier in base class `Footwear`. Hence, they can be accessed only within the same class or within classes that are inheriting the base class. They cannot be accessed outside the derived class or the base class.

Private and protected access specifiers are often used to implement abstraction and encapsulation principles in OOP. For example, when defining built-in libraries, developers may want to give users some access but not full access. They may want to 'protect' some of the functionality from being modified.

In such cases, these specifiers are helpful.

## 13.6 Method Overriding

---

Method overriding is an important OOP concept in PHP, where both the parent and child classes have the same method name. The main objective is to change the behavior of the parent class method.

Method or function overriding is adopted when the class wants to use the parent class method differently. The method defined in the parent class is overridden with a different definition.

Following example shows overriding `getQuantity()` and `getTitle()` methods to return some values:

### Example:

```
...
function getQuantity() {
    echo $this->quantity. "<br/>";
    return $this->quantity;
}
function getTitle(){
    echo $this->title. "<br/>";
    return $this->title;
}
```

In this example, methods `getQuantity()` and `getTitle()` are overridden in the extended class, even though they are already declared in the parent class.

Code Snippet 6 shows how methods can be overridden.

### Code Snippet 6:

```
<?php
// parent class
class Car {
    // public property name
    public $title;
    // public method
    public function drive() {
        echo "Driving the Car<br/>";
    }
}
// child class
class BMW extends Car {
    public function drive() {
        echo "Driving the BMW<br/>";
    }
}
```

```

// child class
class HondaCity extends Car {
    public function drive() {
        echo "Driving the Honda City<br/>";
    }
}

$bmw = new BMW();
$bmw->title = "BMW 3 Series (G20/G21)";
// calling child class method
$bmw->drive();

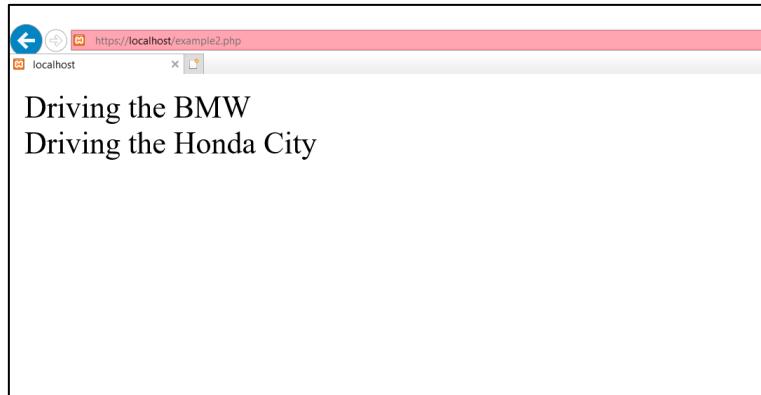
$hondacity = new HondaCity();
$hondacity->title = "Honda City V MT";
// calling child class method
$hondacity->drive();

?>

```

In Code Snippet 6, the parent class named `Car` has two child classes `BMW` and `Honda City` extending the parent class. In the parent class, the method `drive()` is overridden in the child classes and has different definitions to it.

Figure 13.4 shows the output for Code Snippet 6.



*Figure 13.4: Output for Code Snippet 6*

## 13.7 Scope of Functions within Class

---

Functions are by default global. Following is the syntax to define a PHP function:

### Syntax:

```

function <FunctionName>()
{
    code for the function
}

```

Parameters and return statements can be included. Functions have global scope and can be initiated from anywhere within the program. They can even be called from an instance of a class.

### **Inner Functions**

Inner functions, also known as nested functions, are global functions that behave the same way in which they are declared outside the containing function. For example, the variable defined within the function is local to that function. However, as per the global function rule, the function defined within another function is a local variable.

Following example shows how two functions can be defined one within the other:

#### **Example:**

```
. . .
function Function1()
{
    echo("Function1");
    function InnerFunction2()
    {
        echo("Inner Function 2");
    }
}
```

In this example, `InnerFunction2()` will not execute if the outer function `Function1()` is not executed first. This code works only once after calling `Function1`. Calling it again will generate an error.

To create a function more than once; one can use the idea of conditional declaration. Following shows an example of this:

#### **Example:**

```
. . .
function Function1()
{
    echo("Function1");
    if(!function_exists("InnerFunction2")){
        function InnerFunction2()
        {
            echo("Inner Function 2");
        }
    }
}
```

In this example, the `if` condition verifies whether the function name is declared global or not. If it does, then no error occurs. It is always a safe practice to enclose the inner function declarations within `if` statements.

The inner function can be called within the enclosed function only after it is defined. Following shows an example of this:

#### **Example:**

```
function Function1()
{
    echo ("My Function");
    InnerFunction2();
    if(!function_exists("InnerFunction2")){
        function InnerFunction2()
        {
            echo ("InnerFunction2 ");
        }
    }
}
```

In this example, `InnerFunction2()` is a function declared and also called inside `Function1()` function.

## **13.8 Interfaces**

Interfaces help in defining a blueprint for classes. They contain only public methods and no abstract methods. Interfaces also do not have variables. The classes that inherit the interface must define the methods declared inside the interface. It is possible for interfaces to have constants. Interface constants work exactly similar to class constants. Constants are identifiers whose value cannot change during the execution of the script. If the methods declared in the interface are not implemented, an error will occur. Interfaces are defined in the same way as a class. However, this is done using the `interface` keyword instead of the `class` keyword and without any of the methods having their contents defined.

Following is the syntax for defining a PHP interface:

#### **Syntax:**

```
<?php
    // interface declaration
    interface NameOfInterface {
    }
?>
```

Following is an example of creating an interface with a few methods declared in it:

**Example:**

```
 . . .
    // interface declaration
interface ClothingApp {
    // methods declaration
    public function login($phone, $password);
    public function register($phone, $password,
                           $username);
    public function logout();
}
```

In this example, an interface with name `ClothingApp` is created and has three methods declared in it, that contain different parameters. They are `login()`, `register()`, and `logout()`. These methods do not have any definitions, that is, they do not have any statements and are only declared.

Following example shows the code used to create a class that implements the `ClothingApp` interface:

**Example:**

```
 . . .
    // class declaration
class BestClothing implements ClothingApp {
    // methods definition
    public function login($mobile, $password) {
        echo "Login the Customer with phone number: ".
             $mobile;
    }
    public function register($mobile, $password,
                           $username) {
        echo "Customer registered: Phone
              Number=". $mobile. " and Username=". $username;
    }
    public function logout() {
        echo "Customer logged out!";
    }
}
```

In this example, the interface `ClothingApp` is implemented inside the class `BestClothing` and methods declared in the interface are defined with functionality inside the class.

## Implementing Multiple Interfaces

PHP allows a class to implement multiple interfaces. To implement multiple interfaces, the class will have to define methods declared in the interfaces that are implemented by the class. Following example shows creating another interface:

### Example

```
// interface declaration
interface Reward {
    // methods declaration
    public function earnReward($post);
}
```

Here, using the `interface` keyword, the user can declare interface `Reward` and a public method `earnReward` if defined inside the interface.

Following code shows an example of how to inherit from the additional interface in the class `BestClothing`:

### Example:

```
// class declaration
class BestClothing implements ClothingApp, Reward {
    // methods definition
    public function login($mobile, $password) {
        echo "Login with phone number: " . $mobile;
    }
    public function register($mobile, $password,
                           $username) {
        echo "Customer registered: Phone Number
              =" . $mobile . " and Username=" . $username;
    }
    public function logout() {
        echo " Customer logged out!";
    }
    public function earnReward ($post) {
        echo $post . " rewards earned!";
    }
}
```

In this example, the class `BestClothing` implements two different interfaces `ClothingApp` and `Reward`. Implementation of methods declared inside both interfaces should be done as given here.

**Note:** To execute the code successfully, the code for the interfaces and class must be placed in the same program. Also, note that here, the full

functionality for the logic is not yet implemented since, this is just given to illustrate multiple interface inheritance.

## 13.9 Abstract Classes

While implementing inheritance, it can be seen that one can create object of parent class as well as child class. However, what if the user wants to restrict direct use of the parent class? That is, in some cases, one might want to define a base class that declares the structure of a given entity without giving a complete implementation of every method. Such a base class serves as a generalized form that will be inherited by all of its subclasses. The methods of the base class serve as a contract or a standard that the subclass can implement in its own way.

An abstract class serves as a framework that provides certain behavior for other classes. The subclass provides the requirement-specific behavior of the existing framework. Abstract classes cannot be instantiated and they must be subclassed to use the class members. The subclass provides implementations for the abstract methods in its parent class.

PHP provides the `abstract` keyword to accomplish this task.

Following are some important points to know about abstract classes and methods:

- Abstract classes can have methods and properties similar to other classes. However, they cannot be instantiated.
- A child class has to be created to instantiate an abstract class.
- The class should be abstract to have an abstract method.
- Abstract method is a declaration with an empty body. The name of the method and argument is provided by the user.

Following is an example to show definition of an `abstract` class:

### Example:

```
...  
    // abstract class  
    abstract class Microwave {  
  
        // abstract function bake, with no implementation  
        abstract public function bake();  
    }
```

In this example, class `Microwave` is an abstract class with an abstract method.

The main purpose of the abstract class is to let developers follow certain guidelines. For example, to create a new class that extends the class `Microwave`, the definition of abstract method `bake()` should be provided. Hence, the child class must mandatorily define the `bake()` method.

### Non-Abstract Method in Abstract Class

An abstract method can have both abstract and non-abstract methods. A class with an abstract method must be declared abstract, which can be accessed by the child class without overriding.

Following example shows how to include the non-abstract method in the `Microwave` class defined earlier:

#### Example:

```
 . . .
    // abstract class
    abstract class Microwave {
        // protected variable
        protected $degree;
        // non-abstract public function start
        public function start() {
            echo $this->degree. " - Microwave start...<br/>";
        }
        // non-abstract public function stop
        public function stop() {
            echo $this->degree. " - Microwave stop...<br/>";
        }
        // non-abstract public function setDegree
        public function setDegree ($degree) {
            $this->degree= $degree;
        }
        // abstract function bake, with no implementation
        abstract public function bake ();
    }
```

In this example, three non-abstract methods namely `start()`, `stop()`, and `setDegree()` are added to abstract `Microwave` class.

### Inheriting Abstract Classes

Similar to other classes, abstract classes can be inherited too. The child class should define the abstract method defined as abstract class. On the contrary, the child class should be defined as abstract; it does not define the abstract method.

Following example shows how to create two child classes that inherit the class `Car` with definition for the abstract method `bake()`:

**Example:**

```
 . . .
    // child class
    class Onida extends Microwave {
        public function bake() {
            echo "This version baking temperature is - 180
                  to 300 degrees celsius";
        }
    }
```

In this example, `Onida` is the child class which extends the parent class `Microwave`. Now, the child class `Onida` is responsible for defining the code of the `bake()` method.

The abstract class can have as many as child classes. Following example shows this:

**Example:**

```
 . . .
    // child class
    class Samsung extends Microwave {
        public function bake() {
            echo " This version baking temperature is -
                  200 to 400 degrees celsius ";
        }
    }
```

In this example, class `Samsung` is the child class extending the parent class `Microwave`.

An abstract class can have objects indirectly – via child classes. Once a child class is defined, an object for it can be created. Following example shows this:

**Example:**

```
 . . .
    $samsung = new Samsung ();
    $samsung ->setDegree("300");
    $samsung ->bake();
```

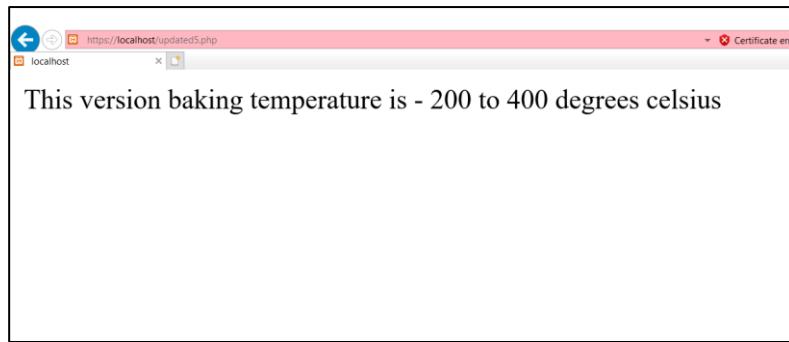
The example shows object creation in PHP. Here, the object `$samsung` is of class `Samsung`.

Code Snippet 7 shows the complete code.

### Code Snippet 7:

```
<?php
    // abstract class
    abstract class Microwave {
        // protected variable
        protected $degree;
        // non-abstract public function start
        public function start() {
            echo $this->degree. " - Microwave start...<br/>";
        }
        // non-abstract public function stop
        public function stop() {
            echo $this->degree. " - Microwave stop...<br/>";
        }
        // non-abstract public function setDegree
        public function setDegree($degree) {
            $this->degree = $degree;
        }
        // abstract function bake
        abstract public function bake();
    }
    // child class
    class Samsung extends Microwave {
        public function bake() {
            echo "This version baking temperature is - 200 to
                  400 degrees celsius ";
        }
    }
    $samsung = new Samsung();
    $samsung->setDegree("400");
    $samsung->bake();
?>
```

In Code Snippet 7, if the user tries to create an object of class `microwave`, an error will occur, as the class `microwave` is declared as `abstract`. However, a new class `Samsung` can extend that abstract class `Microwave` and implement the abstract method declared in parent class. Figure 13.5 shows the output for Code Snippet 7.



**Figure 13.5: Output for Code Snippet 7**

Following are some differences between an interface and an abstract class:

- While an interface cannot have concrete methods and have only abstract methods, an abstract class can have both.
- In an interface, the user can only declare the methods as public. However, in an abstract class, the user can declare methods as public, private, or protected.
- In a class, though multiple interfaces can be implemented, only one abstract class can be implemented.

## 13.10 final Keyword

---

The keyword `final` helps declare a method as final. It prevents a class from being inherited or overridden. This keyword when used with a method prevents a child class from overriding a method.

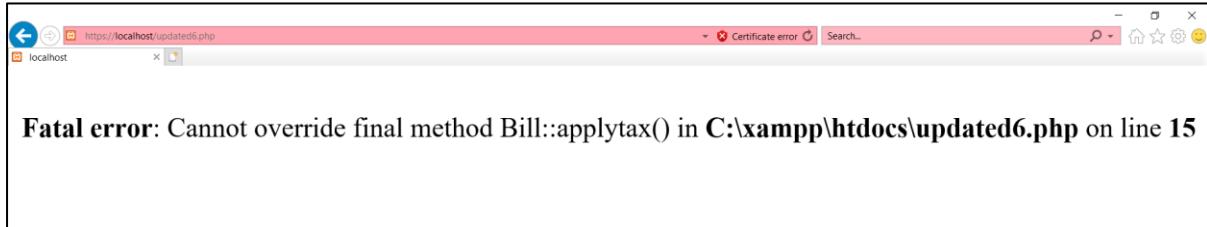
Code Snippet 8 shows an example where a `final` method cannot be overridden.

### Code Snippet 8:

```
<?php
    class Bill {
        public function check() {
            echo "Bill::check() called<br>";
        }
        final public function applytax() {
            echo "Bill::applytax() called<br>";
        }
    }
    class RentBill extends Bill {
        public function applytax() {
            echo "ChildClass::applytax() called<br>";
        }
    }
    $rentBill = new RentBill();
```

```
$rentBill->applytax();  
?>
```

In Code Snippet 8, an error occurs because overriding of final method is not permitted in `RentBill`. Figure 13.6 shows the output for Code Snippet 8.



*Figure 13.6: Output for Code Snippet 8*

## 13.11 Static Members

In PHP, static members are created using `static` keyword. Static members including methods can be accessed without instantiating the class. All such static members are bound to the class and not individual objects.

The `static` keyword can be used to declare properties and methods of a class as static. Static methods can be accessed without instantiating the class. However, a static member cannot be accessed using an instantiated class object. The scope resolution operator (`::`) is used to access static properties. However, the same cannot be done using the object operator (`->`).

A class can be referenced using a variable. However, the value of the variable should not be a keyword such as `self`, `parent`, `static`, and so on.

Following is the syntax to create a static method:

### Syntax:

```
<?php  
class ClassName {  
    public static function <staticMethod>() {  
        <code>  
    }  
}  
?>
```

To access a static method use the class name, a double colon (`::`), and the method name. Following is the syntax:

### Syntax:

```
ClassName::staticMethod();
```

Following example shows how to use the `static` keyword:

### Example:

```
....  
class Test {  
    public static $my_static = "test";  
    public function staticValue() {  
        return self::$my_static;  
    }  
}  
print Test::$my_static. "<br>";  
$test = new Test();  
print $test->staticValue() . "<br>";  
?>
```

In this example, to access `my_static` and `staticvalue` methods, there is no requirement to instantiate new class of `Test`. Instead, they can be accessed as shown here.

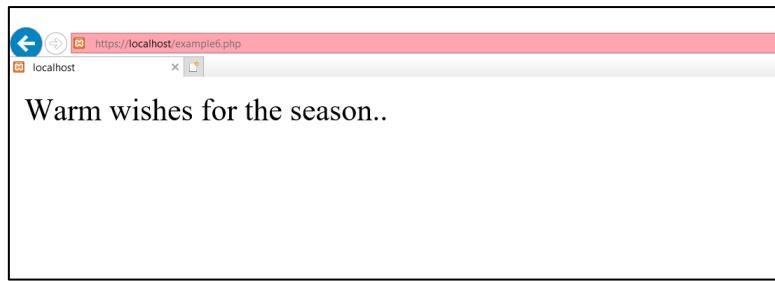
Code Snippet 9 shows an example where the static members are accessed.

### Code Snippet 9:

```
<!DOCTYPE html>  
<html>  
<body>  
<?php  
class greeting {  
    public static function welcome() {  
        echo "Warm wishes for the season..";  
    }  
}  
// Call static method  
greeting::welcome();  
?>  
</body>  
</html>
```

In Code Snippet 9, a static method `welcome()` is declared. Then, the static method is called by using the class name `greeting`, a double colon (`::`), and the method name `welcome()` (without creating an instance of the class first).

Figure 13.7 shows the output for Code Snippet 9.



**Figure 13.7: Output for Code Snippet 9**

## 13.12 Method and Property Overloading

---

The process of overloading the properties and methods with different functionality is known as method/property overloading respectively.

### Method Overloading

In PHP, method overloading allows you to create numerous methods with the same name but different functioning. Using method overloading, the user can create dynamic methods, which are not declared inside the scope of the class. Method overloading is meant to create methods dynamically. The overloaded methods are defined as public. The entities (methods and properties) can be accessed, once the object for a class is created.

Following example shows method overloading:

#### Example:

```
...
class Model {
    public function __call($name, $arguments) {
        echo "This is object method '$name' "
            . implode(', ', $arguments). "<br>";
    }
    public static function __callStatic($name, $arguments) {
        echo "This is static method '$name' "
            . implode(', ', $arguments). "<br>";
    }
}
// Create new object
$obj = new Model;
$obj->runModel("Object context");
Model::runModel("Static context");
```

In this example, `Model` class is defined in which magic functions have been created. These functions are `__call()` and `__callStatic()`. `__call()` is triggered while invoking overloaded methods in the object context. `__callStatic()` is triggered while invoking overloaded methods in static

context. One can access the functions through the new object, using respective arguments to call in object context and static context.

## Property Overloading

Property overloading in PHP allows creation of context-based dynamic properties. It does not require a separate line of code to create properties. An overloaded property is often neither associated with a class instance nor is it declared within the scope of the class.

Following are some functions used in property overloading:

- `__set()`: To initialize overloaded properties
- `__get()`: To read data from inaccessible properties
- `__isset()`: To check the overloaded properties
- `__unset()`: To be invoked when used for overloaded properties

Following example shows property overloading:

### Example:

```
...
// 
class Model {
    private $data = array();
    public $declared = 1;
    // Overloading used when accessed outside the class
    private $hidden = 2;
    // Function definition
    public function __set($name, $value) {
        echo "Set members '$name' to '$value'<br>";
        $this->data[$name] = $value;
    }
    // Function definition
    public function __get($name) {
        echo "Get members '$name': ";
        if (array_key_exists($name, $this->data)) {
            return $this->data[$name];
        }
        $trace = debug_backtrace();
        return null;
    }
    // Function definition
    public function __isset($name) {
        echo "Is '$name' trying to set?<br>";
        return isset($this->data[$name]);
    }
    // Definition of __unset function
    public function __unset($name) {
```

```

        echo "Trying to Unset '$name'<br>";
        unset($this->data[$name]);
    }
    // getHidden function definition
    public function getHidden() {
        return $this->hidden;
    }
}
// Create an object
$obj = new Model;
// Set value 1 to the object variable
$obj->a = 1;
echo $obj->a . "<br>";
// Use isset function to check 'a' is set or not
var_dump(isset($obj->a));
// Unset 'a'
unset($obj->a);
var_dump(isset($obj->a));
echo $obj->declared. "<br><br>";
echo "Private properties are allowed to access inside the
class ";
echo $obj->getHidden(). "<br><br>";
echo "Private properties are not allowed to access outside of
class<br>";
echo $obj->hidden. "<br>";

```

In this example, class `Model` is created in which an array has been created to locate overloading data. A variable `declared` has been created, along with one private variable `hidden`. The functions `_set`, `_get`, `_isset`, `_unset`, and `getHidden` functions are defined in `Model` class. An object `$obj` is created, instantiating the `Model` class. Then, the code sets value of 1 to the object variable. The `isset` function is used to check `a` is set or not. `var_dump()` function is used to dump information about one or more variables.

Private properties are allowed to access inside the class when `declared` is used. However, private properties are not allowed to access outside of class when `hidden` is used.

## Magic Methods

In PHP, method overloading also creates magic methods. Magic methods are triggered in object context and used to create dynamic entities. Following are some magic methods used in PHP:

- `__set()` is triggered when overloaded properties are initialized.

- `__get()` is triggered when overloaded properties are used with PHP print statements.
- `__isset()` is a magic method that is invoked when overloaded properties are checked with the `isset()` function.
- `__unset()` is a function that is invoked on using PHP `unset()` for overloaded properties.

## 13.13 Summary

- Variables and member methods of a class can be accessed through an object of the class by using the → operator.
- Access modifiers are keywords that decide the accessibility for various class methods, variables, and other member methods.
- Inheritance allows users to reduce code duplication by creating a new class with properties and methods inherited from other classes. The child class inherits the properties and methods of the parent class.
- An interface can be considered a blueprint of a class and allows the user to specify what all methods and properties a class should implement.
- Abstract class is a class that defines a structure for other classes to extend. An abstract class cannot be instantiated, has at least one abstract method, and can contain method definitions.
- Overloading is an important feature in PHP that allows users to dynamically create and use methods and properties.

## 13.14 Test Your Knowledge



1. Which of the following pseudo-variables is not available inside methods that are declared as static?
  - a. \$name
  - b. \$this
  - c. \$super
  - d. \$object
  
2. The properties and methods of an object are accessed using the \_\_\_\_\_ operator.
  - a. >>
  - b. →
  - c. =>
  - d. +
  
3. Identify the methods that a subclass inherits from the parent class, when the class is extended.
  - a. Private
  - b. Public
  - c. Protected
  - d. Both b and c
  
4. Child classes are defined using the keyword \_\_\_\_\_.
  - a. extents
  - b. extends
  - c. extend\_class
  - d. child\_class
  
5. Objects are created from the class using the \_\_\_\_\_ keyword, once the class is defined.
  - a. construct
  - b. new
  - c. new object
  - d. static

## **Answers to Test Your Knowledge**

---

1. \$this
2. ->
3. Both b and c
4. extends
5. new

## 13.15 Try it Yourself

1. Write a program to create an interface called Shape which has methods `color()` and `area()`. Implement Square, Rectangle, and Circle which extend this interface and display the results.
2. Write a program to explain the use of `static` keyword.
3. Write a program to explain the use of the `final` keyword.
4. Write a program to create an abstract class called Student with attributes name, roll, and class. It has abstract methods `setName()`, `setRoll()`, `setClass()`, and `showDetails()`. Display the results of at least three students.

# Session 14

## PHP Web Concepts



### Learning Objectives

*In this session, students will learn to:*

- Explain Browser and Platform Identification
- Elaborate how to display images randomly
- Identify the recent enhancements in PHP
- Define Browser Redirection
- Elaborate how to display the File Download dialog box
- Describe PHP and JSON

This session begins by explaining enhancements in PHP using HTML forms and gives an overview on browser redirection. Then, the session classifies PHP and JSON Functions.

### **14.1 Browser and Platform Identification**

---

Based on the input provided by a user, the browser type, or numbers that are generated randomly, PHP can provide dynamic content.

#### **Identifying Browser and Platform**

PHP uses environment variables available on the `phpinfo.php` page, to build the PHP environment. PHP provides a function known as `getenv()` to access the value of all the environment variables.

For example, `HTTP_USER_AGENT` is an environment variable. This is one of the environment variables set by PHP, which helps to identify the browser and the operating system of the user. This variable can be used to create influential and appropriate browser content.

Code Snippet 1 shows how to identify the browser and operating system of the user.

### Code Snippet 1:

```
<html>
<body>
<?php
$viewer = getenv( "HTTP_USER_AGENT" );
$browser = "An unidentified browser";
if( preg_match( "/MSIE/i", "$viewer" ) )
{
$browser = "Internet Explorer";
}
else if( preg_match( "/Netscape/i", "$viewer" ) )
{
$browser = "Netscape";
}
else if( preg_match( "/Mozilla/i", "$viewer" ) )
{
$browser = "Mozilla";
}
$platform = "An unidentified OS!";
if( preg_match( "/Windows/i", "$viewer" ) )
{
$platform = "Windows!";
}
else if ( preg_match( "/Linux/i", "$viewer" ) )
{
$platform = "Linux!";
}
echo("You are using $browser on $platform");
?>
</body>
</html>
```

Figure 14.1 depicts the output of the code.

Your browser: Google Chrome 51.0.2704.103 on windows reports:  
Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/51.0.2704.103 Safari/537.36

**Figure 14.1: Browser and Operating System of User**

Output for Code Snippet 1 displays the browser as Google Chrome and operating system as Windows. Output on other systems might vary based on the browser.

The `preg_match()` function helps to identify the browser which can be either Internet Explorer, Mozilla, or Netscape or any other browser such as Google Chrome. Further, it includes the code to check the operating system of the user, which can be either Linux or Windows.

## **14.2 Displaying Images Randomly**

PHP has two functions `rand()` and `srand()` to help users generate random numbers. To prevent the generation of an ordered pattern of numbers, PHP makes use of random number generator.

### **rand() function**

This function is used to generate a random number within a range.

### **srand() function**

This function is used to generate a random number that sets the seed number as its argument.

Consider images of four car brands, Alfa Romeo, Ferrari, Jaguar, and Porsche. The requirement is to display only one image randomly. An ordered pattern is not to be followed. Code Snippet 2 shows how to do this.

### **Code Snippet 2:**

```
<html>
<body>
<?php
```

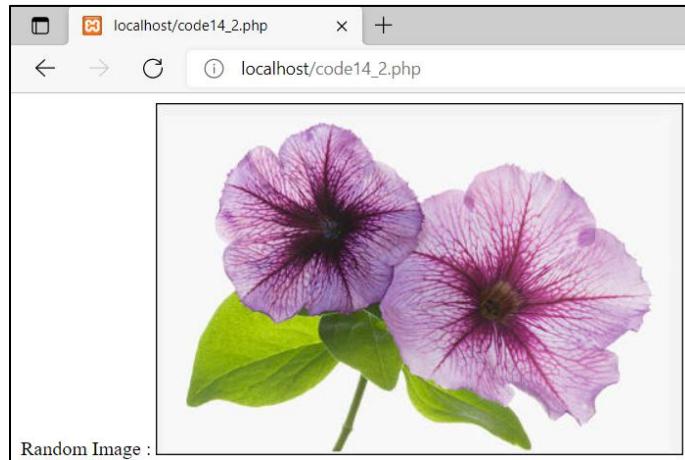
```

$num = rand( 1, 4 );
switch( $num ) {
case 1: $image_file = "/images/alfa.jpg";
break;
case 2: $image_file = "/images/ferrari.jpg";
break;
case 3: $image_file = "/images/jaguar.jpg";
break;
case 4: $image_file = "/images/p.jpg";
break;
}
echo "Random Image : <img src='". $image_file ."'>";?>
</body>
</html>

```

In Code Snippet 2, `rand()` function is used which is assigned to `num` variable. This function generates numbers from 1 to 4. If a user enters a number within 1 and 4, the program enters the `switch` statement and would pass through each case.

Figure 14.2 shows the output of Code Snippet 2.



**Figure 14.2: Random Image Displayed**

A random image is displayed as output from a set of images. If the value entered matches with case 1, it displays the image file given in case 1. If the value entered matches with case 2, it displays image file given in case 2 and similarly, the images in cases 3 and 4 are displayed if the value entered matches case 3 and 4 respectively. Otherwise, image present in another source file will be displayed. The `echo` statement outside the `switch` statement displays the image in the output.

## 14.3 Recent Enhancements in PHP 8

PHP has become more methodical, definitive, and dependable with new enhancements introduced in version 8.

### Constructor Property Promotion

In PHP, at present, all necessary properties must be frequently repeated at least a minimum of four times. As per PHP 8 Request for Comments (RFC), the constructor and the parameter definition will be merged, making it much easier to declare parameters.

An Request for Comments (RFC) is a document describing specifications for a technology and become a standards document.

Only parameters prefixed with `private`, `public`, and `protected` keywords, which are called promoted parameters can be defined and merged with the constructor as per the RFC.

As an example, consider following class sample:

```
class Square {  
    public float $p;  
    public float $q;  
    public float $r;  
    public function __construct()  
        float $p = 0.0,  
        float $q = 0.0,  
        float $r = 0.0,  
    ) {  
        $this->p = $p;  
        $this->q = $q;  
        $this->r = $r;  
    }  
}
```

The properties are repeated four times in the example as follows:

- 1) During the declaration of the property
- 2) During the constructor parameters
- 3) During the assignment of property
- 4) The property type is repeated two times.

In PHP 8, the code becomes much simpler as shown here:

```
class Square {  
    public function __construct(  
        public float $p = 0.0,  
        public float $q = 0.0,  
        public float $r = 0.0,  
    ) {}  
}
```

Both the examples are same, but with the enhanced feature of constructor property promotion, the class becomes more concise and simpler.

## Arrays Starting With a Negative Index

With PHP 8, arrays beginning with a negative index change their behavior.

If an array begins with a negative index (`start_index < 0`), the remaining indices will start from 0.

Example for declaring negative index is as follows:

```
$a = array_fill(-6, 5, true);  
var_dump($a);
```

The second index will change as `start_index + 1`, irrespective of `start_index` value.

## Consistent Type Errors for Internal Functions

When an illegal parameter type is passed, the behavior of internal and user-defined functions are different. The internal functions throw a warning and return `null` whereas user-defined functions throw a `TypeError`.

With PHP 8, whenever there is a parameter type mismatch, the internal parameter parsing APIs will always generate a `ThrowError`.

Syntax used for retrieving the name of a class is `Foo\Bar::class`.

With PHP 8, the same syntax is extended to the objects as well. Therefore, it becomes much easier to retrieve the name of the class for a given object.

## **Nullsafe Operator**

PHP 8 introduces nullsafe operator `$->`. During an evaluation, if an operator in a chain evaluates to `null`, then the execution of the chain stops and results in `null`. Only when the first operator is not evaluated as `null`, the second operator can be evaluated, otherwise the result is displayed as `null`.

## **Saner String to Number Comparisons**

In older versions of PHP, during the comparison of numbers and strings, strings were converted to numbers and then, a comparison was made between integers or floats.

PHP 8 made a small change here, while comparing a string to a numeric string, the existing behavior, that is, number comparison is still carried out. However, a string comparison is performed by converting the number to a string.

## **Saner Numeric Strings**

In PHP, there are three main categories for the strings containing numbers, they are as follows:

1. Numeric strings: They are strings that contain a number and are preceded by whitespaces.
2. Leading-numeric string: They are strings whose first characters are numeric strings and following characters are non-numeric.
3. Non-numeric string: Strings that are neither numeric strings nor leading-numeric strings fall under this category.

PHP 8 combines various numeric string modes into a single concept. Numeric characters will have numbers with both leading and trailing whitespaces. Any other type of string is considered non-numeric and will throw `TypeError`.

## **Trailing Comma in Parameter List**

Commas that are appended to the list of items during various circumstances are known as Trailing commas. These trailing commas were introduced in list syntax, in PHP 7.2 and PHP 7.3 introduced them in function calls.

PHP 8 version introduced the trailing commas in parameter lists with methods, functions, and closures.

## 14.4 Browser Redirection

---

Browser Redirection is one of the functionalities that can be done smoothly in PHP. PHP uses `header()` function to accomplish this.

The script for redirection should be written at the beginning of the code, before running or loading any other PHP code. The `header()` function of PHP helps to redirect the browser, by providing the raw HTTP headers to it. This helps to redirect to a different location. One can specify a destination using the location and pass the header as the argument to the `header()` function. To end the script for redirection, the `exit()` function should be used to stop computing the rest of the code.

Code Snippet 3 is a source code that redirects the browser request to a different Web page.

### Code Snippet 3:

```
<?php
error_reporting(0);
if($_POST["location"] )
{
$location = $_POST["location"];
header( "Location:$location" );
exit();
}
?>
<html>
<body>
<p>Select a Website to visit :</p>
<form action=<?php $_PHP_SELF ?>" method="POST">
<select name="location">
<option value="http://w3c.org">
World Wide Web Consortium
</option>
<option value="http://www.google.com">
Google Search Page
<option value="http://www.sample.com">
Php tutorial page
</option>
</select>
```

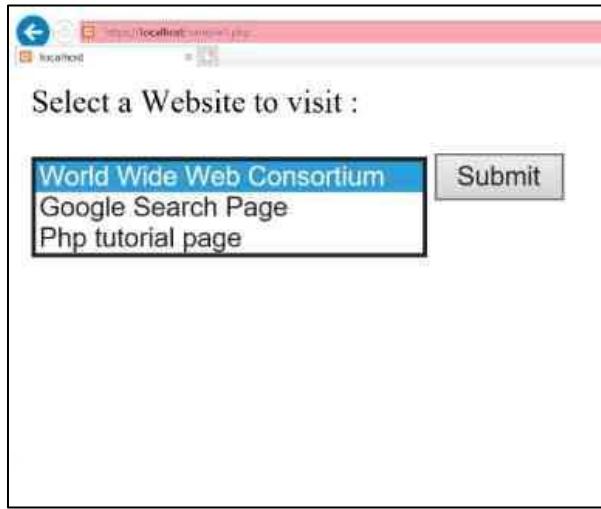
```
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Code Snippet 3 contains code that redirects the browser request. When the code is executed in the browser, user can choose the site he/she wants to visit. When user selects a particular option and clicks submit button, he/she will be redirected to that specific Website.

Figure 14.3 and 14.4 show the output for Code Snippet 3.



*Figure 14.3: Initial Output of Code Snippet 3*



*Figure 14.4: Selecting a Website from List*

Upon selecting a link, the user will be redirected to that site.

## **14.5 Displaying ‘File Download’ Dialog Box**

---

Another functionality of the HTTP header is that it helps to create links.

When the user clicks these links, he/she would receive a pop-up window or a dialog box that help in ‘File Download’. Here, **Content-Type** must be sent as `text/html\n\n` and the HTTP header will not be the actual header.

Following is the syntax for making a `FileName` file downloadable from a link:

```
#!/usr/bin/perl
# HTTP Header
print "Content-Type:application/octet-stream;
name=\"FileName\"\r\n";
print "Content-Disposition: attachment;
filename=\"FileName\"\r\n\r\n";
# Actual File Content
open( FILE, "<FileName" );
while(read(FILE, $buffer, 100) ){
print("$buffer");
}
```

The content-type given in the syntax is `application/octet-stream`.

The original filename will be included along with it.

Code Snippets 4 and 5 depict an example of how to perform ‘File download’.

### **Code Snippet 4:**

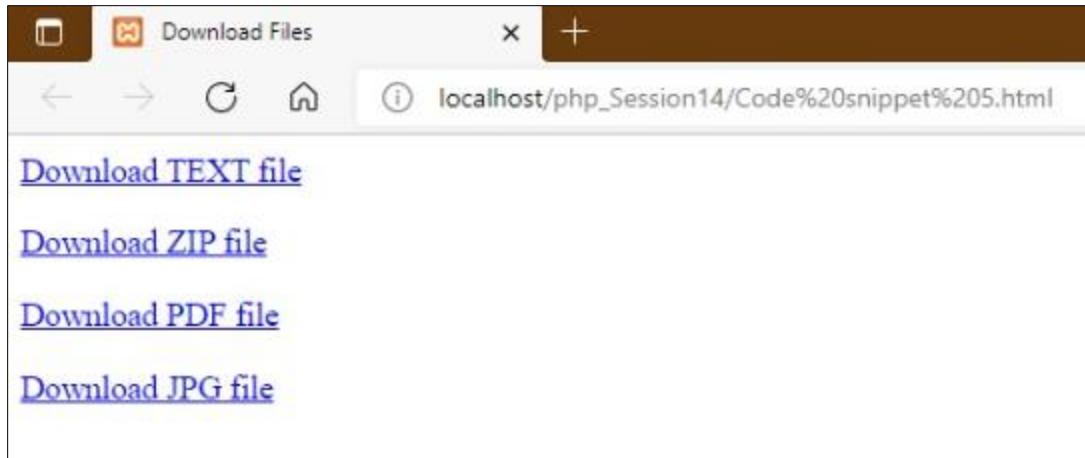
```
<html>
<head>
<title>Download Files</title>
</head>
<body>
<p><a href="download.php?path=xyz.txt">Download TEXT
file</a></p>
<p><a href="download.php?path=table.zip">Download ZIP
file</a></p>
<p><a href="download.php?path=student.pdf">Download PDF
file</a></p>
<p><a href="download.php?path=module.jpg">Download JPG
file</a></p>
</body>
</html>
```

### Code Snippet 5: (download.php)

```
<?php
if(isset($_GET['path']))
{
//Read the filename
$filename = $_GET['path'];
//Check if the file exists or not
if(file_exists($filename)){
//Define header information
header('Content-Description: File Transfer');
header('Content-Type: application/octet-stream');
header("Cache-Control: no-cache, must-revalidate");
header("Expires: 0");
header('Content-Disposition: attachment;
filename="'.basename($filename).'"');
header('Content-Length: ' . filesize($filename));
header('Pragma: public');
//Clear system output buffer
flush();
//Read the file size
readfile($filename);
//Terminate from the script
die();
}
else{
echo "There is no File to download.";
}
}
else
echo "Name of the File is not defined here."
?>
```

Code Snippet 4 has HTML code which is written at the beginning and contains four files in **htdocs** folder of XAMPP server. When the user clicks the file name link, the file is downloaded. Both the PHP and HTML files must be saved in the **htdocs** folder of XAMPP server. The files to be downloaded must also be present in same path. If the files are not found in the same folder, output displays as ‘There is no File to download.’.

Figure 14.5 shows the output for the code.



*Figure 14.5: List of Files to Download*

## 14.6 PHP and JSON

Introduced by Douglas Crockford, JSON stands for JavaScript Object Notation. It is a syntax to store and exchange or interchange data. JSON helps to decode or parse the syntax. It can be implemented with the help of the JSON extension in PHP.

### 14.6.1 PHP JSON Functions

PHP supports working with JSON data by including built-in functions.

Table 14.1 describes some of the PHP JSON functions.

Function	Description
<code>json_decode()</code>	Decodes a JSON string
<code>json_encode()</code>	Encode a value to JSON format
<code>json_last_error()</code>	Returns the last error occurred
<code>json_last_error_msg()</code>	Returns the error string of the last <code>json_encode()</code> or <code>json_decode()</code> call

*Table 14.1: PHP JSON Functions*

➤ `json_encode()`

Following is the syntax for `json_encode()`:

#### Syntax:

```
string json_encode ( mixed $value [, int $options = 0 [, int $depth = 512 ]] )
```

There are two different parameters to the `json_encode()` function, namely Value and Option.

Value

It works only with the UTF-8 encoded data.  
It is the value being encoded.

Option

It is the bitmask that consists of optional values.  
There are different bitmasks that are used in PHP such as:  
`JSON_NUMERIC_CHECK`, `JSON_HEX_QUOT`,  
`JSON_UNESCAPED_SLASHES`, `JSON_HEX_TAG`,  
`JSON_HEX_AMP`, and so on.

Code Snippets 6 and 7 show how encoding takes place in JSON.

#### **Code Snippet 6:**

```
<?php  
$arr = array('car' => 1, 'bike' => 2, 'bicycle' => 3, 'bus' =>  
4, 'truck' => 5);  
echo json_encode($arr);  
?>
```

Code Snippet 6 has a sample code that converts PHP array into JSON encoded array elements. Array type used in the code is called Indexed arrays.

#### **Output:**

```
{"car":1,"bike":2,"bicycle":3,"bus":4,"truck":5}
```

#### **Code Snippet 7:**

```
<?php  
$arr2 = array('brandname' => 'BMW', 'version' => 'X5', 'year'  
=> '2020');  
echo json_encode($arr2);  
?>
```

Code Snippet 7 has the code that converts PHP array into JSON encoded array elements. Array type used in the code is called Associative arrays.

#### **Output:**

```
{"brandname": "BMW", "version": "X5", "year": "2020"}
```

## ➤ `json_decode()`

Following is the syntax for `json_decode()`:

### Syntax:

```
mixed json_decode ( string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0 ]]] )
```

There are four different parameters in `json_decode()` function, namely **json\_string**, **assoc**, **depth**, and **options**.

json_string	assoc	depth	options
<ul style="list-style-type: none"><li>It is an UTF-8 encoded data and an encoded string.</li></ul>	<ul style="list-style-type: none"><li>Returned objects will be converted into associative arrays when <code>assoc</code> parameter is used. When set to TRUE, it works like a boolean type parameter.</li></ul>	<ul style="list-style-type: none"><li>It specifies the recursion depth and is an integer type parameter.</li></ul>	<ul style="list-style-type: none"><li>It supports <code>JSON_BIGINT_AS_STRING</code> and is an integer type bitmask of JSON decode.</li></ul>

Code Snippets 8 and 9 show how decoding is done in JSON.

### Code Snippet 8:

```
<?php
$jsondata = ' {"Red":6,"Green":7,"Blue":8,"White":9,
              "Black":10}';
var_dump(json_decode($jsondata, true));//true means returned
// object gets converted into associative array
?>
```

Code Snippet 8 first defines JSON content in a variable named `$jsondata`. Then, `var_dump()` function is used to identify the data type of a given variable, which here is array type and its value. The array values are elements of Integer type. `json_decode()` function is used in the code to convert JSON string to a PHP variable. The Associative array is used in the code and JSON syntax is used for declaring the array.

## **Output:**

```
array(5) {
    ["Red"] => int(6)
    ["Green"] => int(7)
    ["Blue"] => int(8)
    ["White"] => int(9)
    ["Black"] => int(10)
}
```

## **Code Snippet 9:**

```
<?php
$json2 = '{"Color" : "Red", "Object" : "Apple", "Group" :
"Fruits"}';
var_dump(json_decode($json2, true));
?>
```

Code Snippet 9 first defines JSON content in a variable named `$json2`. The `var_dump()` function is then used to identify the data type of a given variable, which is array type and its value, which are elements of String type. The elements in the code are namely, ‘firstName’, ‘lastName’, and ‘email’.

`json_decode()` function is used in the code to convert JSON String to PHP variable. The Associative array is used in the code and JSON syntax is used for declaring this array.

## **Output:**

```
array(3) {
    ["Color"]=> string(3) "Red"
    ["Object"]=> string(5) "Apple"
    ["Group"]=> string(6) "Fruits"
}
```

### **14.6.2 PHP – Accessing Decoded Values**

PHP also includes several predefined JSON Constants. Table 14.2 describes some of the PHP predefined JSON Constants and their type.

Constant	Type	Description
JSON_ERROR_NONE	Integer	No error has occurred
JSON_ERROR_DEPTH	Integer	Maximum stack depth has been exceeded
JSON_ERROR_STATE_MISMATCH	Integer	Invalid/Malformed JSON
JSON_ERROR_CTRL_CHAR	Integer	Control character error
JSON_ERROR_SYNTAX	Integer	Syntax error
JSON_ERROR_UTF8	Integer	Malformed UTF-8 characters. Valid since PHP 5.3
JSON_ERROR_RECURSION	Integer	Invalid recursive reference values. Valid since PHP 5.5
JSON_ERROR_INF_OR_NAN	Integer	Invalid NAN or INF values. Valid since PHP 5.5
JSON_ERROR_UNSUPPORTED_TYPE	Integer	Invalid type. Valid since PHP 5.5
JSON_ERROR_INVALID_PROPERTY_NAME	Integer	Invalid property name. Valid since PHP 7.0
JSON_ERROR_UTF16	Integer	Malformed UTF-16 characters. Valid since PHP 7.0

**Table 14.2: Predefined JSON Constants Supported by PHP**

### 14.6.3 PHP – Accessing Decoded Values

In PHP, the decoded values can be accessed either from an object or from an associative array. Code Snippet 10 shows how to access the decoded values.

#### Code Snippet 10:

```
<?php
$jsonobj = '{"John":29,"Nick":28,"Sean":31}';
$obj = json_decode($jsonobj);
echo $obj->John . " ";
echo $obj->Nick . " ";
echo $obj->Sean. " ";
?>
```

In Code Snippet 10, an object variable named `jsonobj` is created with JSON content. Then, `json_decode()` function is used to parse this JSON content and return the parsed content into a variable named `obj`. The `echo` keyword is used to access values of the object with keys such as John, Nick, and Sean with a blank space in between each key.

### **Output:**

29 28 31

#### **14.6.4 PHP - Looping Through JSON Values**

The code in Code Snippet 10 is good for a few keys and values, but what if there are hundreds of such keys and values in the data? It would be cumbersome and tedious for the user to write statements for each. PHP `foreach()` loop can be used in such cases to loop through the values of a JSON array or a list.

Code Snippets 11 and 12 show how to loop the values of a PHP object containing JSON data. Although this is still simple JSON data, the code can be reused for larger arrays too, without having to write separate `echo` statements for each element of the array.

### **Code Snippet 11:**

```
<?php
$jsonobj = '{"John":29,"Nick":28,"Sean":31}';
$obj = json_decode($jsonobj);
foreach($obj as $key => $value) {
    echo $key . " => " . $value . "<br>";
}
?>
```

Code Snippet 11 shows array elements with keys and values. Here, `foreach()` loop is used to display each array elements of JSON. The output is converted into PHP array elements, using `json_decode()` function.

### **Output:**

John => 29  
Nick => 28  
Sean => 31

## Code Snippet 12:

```
<?php  
$jsonobj = '{"David":25,"Joseph":47,"Joe":43}';  
$arr = json_decode($jsonobj, true);  
foreach($arr as $key => $value) {  
    echo $key . " => " . $value . "<br>";  
}  
?>
```

Code Snippet 12 shows array elements with keys and values. Here, `foreach()` loop is used to display each array element of JSON data. The output is converted into PHP array elements using `json_decode()` function.

### Output:

David=>25  
Joseph=>47  
Joe=>43

## 14.7 Deploying PHP on a Remote Web Server Using the NetBeans IDE

For deploying PHP applications on a remote Web server, it is mandatory for the user to do the following:

- Register an account on a hosting provisioner
- Deploy a database such as MySQL, on the remote server itself

To run a PHP application on a remote Web server, an FTP connection profile is required to be set, along with the Run Configuration.

To either create a new project or a NetBeans project on a remote server, for a PHP application, the FTP connection settings found in the **Run Configuration** panel have to be specified. By default, a remote run configuration is used.

NetBeans is a popular IDE and provides facility to link to a Web server.

After selecting and setting the required remote connection for the project, upload the source files as per requirements. The available upload options are as follows:

- **On Run** – The source files generally get uploaded to the server when the project is run (or executed).

- **On Save** – Modifications such as create, rename, edit, or delete are immediately passed on to the remote server. A progress bar will be displayed if the operation takes more than a second.
- **Manually** – For a manual upload, the function in IDE has to be used. The FTP server of the projects offers both download and upload options for the user.

Once the required upload option is selected, a dialog box with a tree view of the source files is displayed. Here, the user can select individual files to be uploaded.

To run the PHP application, select the **Remote Web Site** option from the **Run As** drop-down list on the **Properties** panel. Verify the **Run Configuration** settings. Select **right arrow** toolbar, if the project is set as **Main**. Otherwise, place the cursor on the project node and select **Run** from the menu.

User can also choose frameworks such as Laravel to deploy their PHP applications to remote servers.

## 14.8 Summary

- PHP provides a function `getenv()` to access the value of all the environment variables, which in turn helps to retrieve `HTTP_USER_AGENT` value containing information about the local browser.
- PHP `rand()` and `srand()` functions are used to generate random numbers.
- PHP codes can now be methodical, definitive, and dependable with the new enhancements in PHP 8.
- In PHP, browser redirection can be achieved using the `header()` function.
- JSON is a popular data exchange format for the Web.
- `json_encode()` is used to encode JSON and `json_decode()` is used to decode JSON in PHP.
- In PHP, JSON decoded values can be accessed either from an object or from an associative array.
- PHP `foreach()` loop is used to loop through the values of a JSON array or a list.
- To deploy the PHP application on a remote server, user has to register an account on a hosting provisioner and deploy a database such as MySQL on the remote server.

## 14.9 Test Your Knowledge



1. Which of the following is not a type in JSON?
  - a. Date
  - b. Object
  - c. Array
  - d. String
2. What will be the value of the variable \$input in the following PHP code?

```
<?php
$input = "Juliet<td>Lawrence</td>you are really<i>pretty</i>!";
$input = strip_tags($input,"<i></i>");
echo $input;
?>
```

- a. Juliet Lawrence you are really pretty!
  - b. Juliet <td>Lawrence</td> you are really<i>pretty</i>!
  - c. Juliet <td>Lawrence</td> you are really pretty!
  - d. Juliet Lawrence you are really<i>pretty</i>!
3. Which of the following functions is used to redirect the browser?
  - a. exit()
  - b. header()
  - c. break()
  - d. None of these

4. Which of the following functions is used to parse or convert the JSON content to PHP variable?
  - a. json\_decode()
  - b. json\_encode()
  - c. json\_last\_error
  - d. None of these
  
5. Which of the following PHP functions is used to access the value of all environment variables?
  - a. getenv()
  - b. getvar()
  - c. getnow()
  - d. gethour()

## **Answers to Test Your Knowledge**

---

1. Date
2. Juliet Lawrence you are really*pretty*!
3. header()
4. json\_decode()
5. getenv()

## 14.10 Try it Yourself

1. Write a program to demonstrate how you can identify a client browser and an operating system.
2. Write a program to demonstrate how you can display a different image each time, out of six images in a folder named *images*.
3. Write a program to demonstrate how you can redirect a browser request to another Web page.
4. Write a program to encode JSON and decode a JSON string array comprising color names in the rainbow.

## Session 15

# PHP – AJAX



### Learning Objectives

*In this session, students will learn to:*

- Explain AJAX, work with AJAX, and Internet standards of AJAX
- Elaborate working with AJAX and PHP
- Illustrate working with AJAX and MySQL
- Explain how to use XML with AJAX
- Explain a program to perform AJAX live search technique

The session begins by defining what AJAX is. The session then, provides an overview of working with AJAX and Internet standards of AJAX and explains how to work with AJAX and PHP. Further, it covers details on how to work with AJAX and MySQL. It also lists the advantages of PHP.

### **15.1 Introduction to AJAX**

---

Asynchronous JavaScript and XML (AJAX) is not a new technology in itself, but a new approach to using JavaScript, HTML, CSS, and XML for creating faster and better Web applications.

Traditionally, Web applications transfer information to and from the server in a Synchronous manner. This means that when the user fills a form and clicks submit, the user will be directed to a new page with updated information from the server.

With AJAX, Web applications make fast incremental updates to the user interface without reloading the entire page. In simple terms, using AJAX, one can make fast and dynamic Web Pages.

Applications or Web pages developed using traditional non-AJAX techniques reload the whole Web page even for changing few parts of the content. This can lead to server overhead and reduced performance.

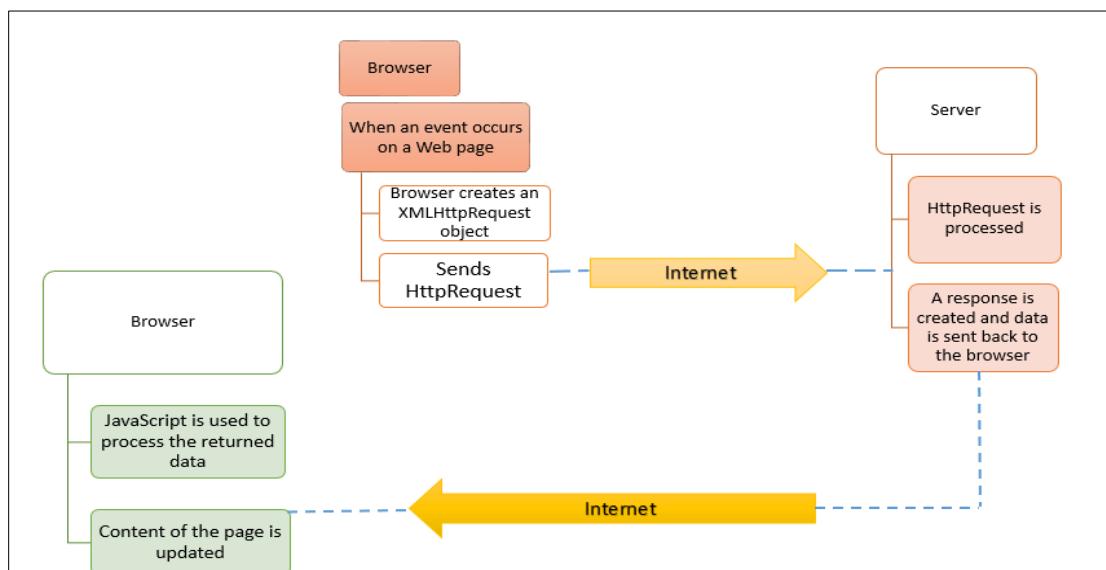
With AJAX, the server is communicated behind the curtain. This means that the user would not notice the communication of the Web page to the server. This makes the application more interactive to the user's actions.

Google Search, Reddit handling of votes, Twitter updation of trending posts, and Facebook tabs are some examples of Web pages using AJAX.

### 15.1.1 Working of AJAX

Whenever an event occurs in a browser, an `HttpRequest` object is created and transferred to the server over the Internet. The server then, processes this request and sends an `HttpResponse` back to the browser. Browser uses JavaScript to execute this data and update the page contents.

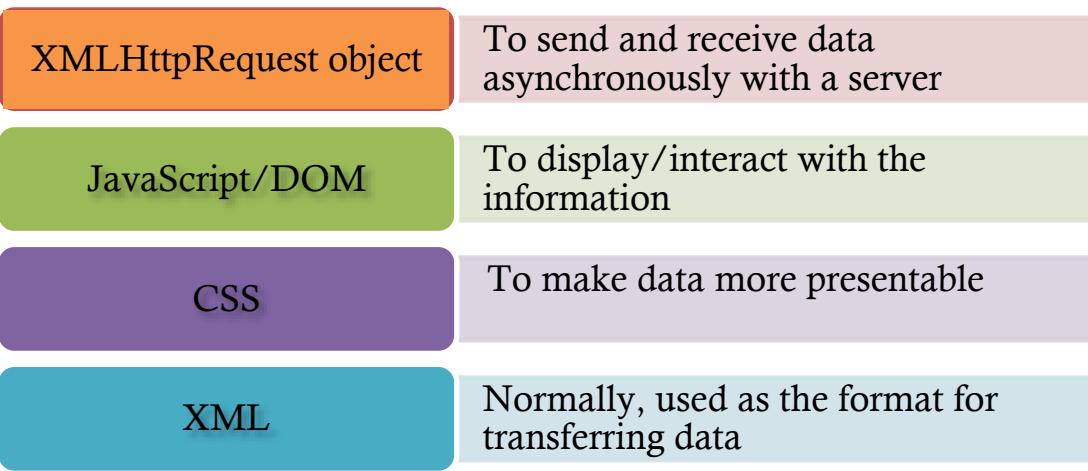
Figure 15.1 explains the workflow of AJAX.



**Figure 15.1: Workflow of AJAX**

### 15.1.2 Internet Standards of AJAX

AJAX is mainly used in Web development. It is based on Internet standards. It is a combination of XMLHttpRequest object, JavaScript/DOM, CSS, and XML.



AJAX is independent of Web server software.

## 15.2 AJAX and PHP

---

AJAX integrated with PHP and MySQL establishes smooth and robust communication with the database and the server. It helps in improving performance of the application.

### 15.2.1 Example for AJAX PHP

It becomes easy to access information from the database using AJAX and PHP. Consider an example to understand this. To use AJAX and PHP in the example with a database, one must build some MySQL queries. In this example, the user creates a table called `customer` and sends a query via AJAX. The results are then displayed in the browser Web page using '`customerdetail.html`'.

As a prerequisite to execute the example properly, a MySQL table should be created as follows:

```
CREATE TABLE customer (
    name varchar(50) NOT NULL,
    gender varchar(1) NOT NULL,
```

```
bill int(10) NOT NULL,  
year int(10) NOT NULL,  
PRIMARY KEY (name)  
)
```

The user must also insert customer data into the table as shown here:

```
INSERT INTO 'customer' VALUES ('Paul', 'm', 10000, 1998);  
INSERT INTO 'customer' VALUES ('Tyron', 'm', 5000, 1992);  
INSERT INTO 'customer' VALUES ('Hans', 'm', 25000, 1994);  
INSERT INTO 'customer' VALUES ('Erin', 'f', 12000, 1997);  
INSERT INTO 'customer' VALUES ('Vinn', 'f', 2000, 2000);
```

Code Snippet 1 is a client-side code which displays a Web Page for a dynamic user interaction.

### Code Snippet 1:

```
<html>  
<body>  
<script language="javascript" type="text/javascript">  
//Checking Browser Compatibility  
function runAjax(){  
    var ajaxHttpRequest; // Key variable that is necessary for AJAX  
    try{  
        // Opera 8.0+, Firefox, Safari  
        ajaxHttpRequest = new XMLHttpRequest();  
    }  
    catch (e){  
        // Internet Explorer Browsers  
        try{  
            ajaxHttpRequest = new ActiveXObject("Msxml2.XMLHTTP");  
        }  
        catch (e) {  
            try{  
                ajaxHttpRequest = new ActiveXObject("Microsoft.XMLHTTP");  
            }  
        }  
        catch (e){  
            // Something went wrong  
            alert("Your browser is not working!");  
            return false;  
        }  
    }  
}  
//Setting up ajax to update the page on receiving the query results
```

```

ajaxHttpRequest.onreadystatechange = function() {
    if(ajaxHttpRequest.readyState == 4) {
        var displayResponse = document.getElementById('result');
        displayResponse.innerHTML = ajaxHttpRequest.responseText;
    }
}
//Sending the input data to server-side
var bill = document.getElementById('bill').value;
var year = document.getElementById('year').value;
var gender = document.getElementById('gender').value;
var queryString = "?bill=" + bill ;
queryString += "&year=" + year + "&gender=" + gender;
ajaxHttpRequest.open("GET", "customerdetail.php" +
queryString, true);
ajaxHttpRequest.send(null);
}
</script>
<form name='customerForm'>
Total bill of the customer:  &nbsp;&nbsp;<input type='number' id='bill' /> <br />
Year the customer joined: <input type='number' id='year' />
<br />
Gender: <select id='gender'>
    <option value="m">m</option>
    <option value="f">f</option>
</select>
<input type='button' onclick='runAjax()' value='Submit' />
</form>
<div id='result'>Your result will be shown here</div>
</body>
</html>

```

**Note:** The query variables are passed according to HTTP standards.

In Code Snippet 1, the user enters a bill amount, a year a customer joined, and a gender. Upon clicking 'Submit', the Web page displays data of all customers whose total bill is more than or equal to the input bill, year in which they joined is less than or equal to the input year, and gender equal to given gender (m/f).

The query is sent by PHP and AJAX updates the Web page with the results without reloading the entire Web page. This action is done at the server side.

## Server-Side PHP

Once client-side scripting is done, server-side script should be written. This server-side code will retrieve bill, year, and gender from the database based on given inputs and return it to the client. Code Snippet 2 is saved in the file named as `customerdetail.php`. Assume that a database named `test01` has been created.

### Code Snippet 2:

```
<?php
    //Passing the credentials to access the database
    $db_host = "localhost";
    $db_user = "root";
    $db_pass = "";
    $db_name = "test01";
    //Establishing the connection
    $con = mysqli_connect($db_host, $db_user, $db_pass,
        $db_name);
    $bill = $_GET['bill'];
    $year = $_GET['year'];
    $gender = $_GET['gender'];
    //Constructing the query with all conditions
    $query = "SELECT * FROM customer WHERE gender = '$gender'";
    if(is_numeric($bill))
        $query .= " AND bill >= $bill";
    if(is_numeric($year))
        $query .= " AND year <= $year";
    //Running the query
    $qry_result = mysqli_query($con, $query) or
        die(mysql_error());
    //Table structure
    $display_tb = "<table>";
    $display_tb .= "<tr>";
    $display_tb .= "<th>Name</th>";
    $display_tb .= "<th>Total Bill</th>";
    $display_tb .= "<th>Year Joined</th>";
    $display_tb .= "<th>Gender</th>";
    $display_tb .= "</tr>";
    //Inserting data for each person
    while($row = mysqli_fetch_array($qry_result)){
        $display_tb .= "<tr>";
        $display_tb .= "<td>$row[name]</td>";
        $display_tb .= "<td>$row[bill]</td>";
        $display_tb .= "<td>$row[year]</td>";
```

```

$display_tb .= "<td>$row[gender]</td>";
$display_tb .= "</tr>";
}
echo "Displaying the results for " . $query . "<br />";
$display_tb .= "</table>";
echo $display_tb;
?>

```

In Code Snippet 2, a connection is made with MySQL database to fetch the results of the query. Data supplied in the form is retrieved using `$_GET`. Each of these data parameters are then, substituted in the SELECT query so that the relevant matching data can be retrieved.

Figures 15.2 and 15.3 shows the output of Code Snippet 1 and 2 respectively.

**Figure 15.2: Web Page For User Inputs**

Once the user enters the required data in the input box and clicks ‘Submit’, the results are displayed as shown in Figure 15.3. Here, the entire page will not be reloaded. Only the results section will be refreshed with the retrieved data.

Name	Total Bill	Year Joined	Gender
Hans	25000	1994	m
Paul	10000	1998	m

**Figure 15.3: Output of Server Side Script Using AJAX**

## 15.3 AJAX and MySQL

---

AJAX can also be used for communicating with the database interactively. One can create a simple live database search functionality by utilizing AJAX and PHP techniques, where the search results will be displayed as the characters are typed in the search input box.

Consider an example where user wants to display train information based on a given train name or train number. This input is to be accepted via a drop-down list. The data will be retrieved from a database using AJAX. The values for the drop-down are hardcoded in this example, but in practical scenarios, they may be populated from the database too.

The MySQL database table used in this example contains following data:

Train_Name	Train_No	Source	Destination
Inter	612158	Germany	France
Orient	829150	Paris	Budapest
Xpress	555432	France	Switzerland
BNF	125434	Vienna	Graz

In the example, whenever a user clicks any of the two drop-downs and chooses either the train number or the train name from the list, a function named ‘displayTrain()’ is invoked. The `onchange()` event triggers this function. For this, the HTML code shown in Code Snippet 3 is used.

### Code Snippet 3: (displayTrain.html)

```
<html>
<head>
<script>
//Function to use Ajax to transfer the data
function displayTrain(train_data) {
    var ajaxHttpRequest = new XMLHttpRequest();
    if (train_data == "") {
        document.getElementById("result").innerHTML = "";
        return;
    }
    ajaxHttpRequest.onreadystatechange = function() {
        if(ajaxHttpRequest.readyState == 4) {
```

```

        var displayResponse = document.getElementById('result');
        displayResponse.innerHTML = ajaxHttpRequest.responseText;
    }
}

//The variable is passed to PHP to retrieve the result
ajaxHttpRequest.open("GET", "displayTrain.php?q=" +
    train_data, true);
ajaxHttpRequest.send(null);
}

</script>
</head>
<body>
    <form>
        <select name="train_name" onchange =
            "displayTrain(this.value)">
            <option value="">Select the train name:</option>
            <option value="Inter">Inter</option>
            <option value="Orient">Orient</option>
            <option value="Xpress">Xpress</option>
            <option value="BNF">BNF</option>
        </select>
        &nbsp;&nbsp;&nbsp;
        or
        &nbsp;&nbsp;&nbsp;
        <select name="train_number" onchange =
            "displayTrain (this.value)">
            <option value="">Select the train number:</option>
            <option value="125434">125434</option>
            <option value="555432">555432</option>
            <option value="612158">612158</option>
            <option value="829150">829150</option>
        </select>
    </form>
    <br>
    <div id="result"><b>Train's information...</b></div>
</body>

</html>

```

In Code Snippet 3, the user first checks for any empty input through the condition `train_data= ""`. If the condition is true, the function returns an empty string. Whereas, if the condition is false and there is any selected data from either drop-down, an `XMLHttpRequest` object is created. Once the server response is ready, it sends the request to the server-side file. The data is passed as a parameter 'q' in the URL.

Corresponding PHP code will be stored in a file named ‘displayTrain.php’. This code executes a query from the database, and the result is returned to the HTML table. Code Snippet 4 shows this code, which will be saved as displayTrain.php. It returns the data of the input train or number entered by the user.

#### **Code Snippet 4: (displayTrain.php)**

```
<html>
<head>
<style>
table {
    width: 100%;
    border-collapse: collapse;
}

table, td, th {
    border: 1px solid black;
    padding: 5px;
}

th {text-align: left;}
</style>
</head>
<body>

<?php
$q = $_GET['q'];
$con = mysqli_connect('localhost','root','','test01');
if((int)$q> 0){
$sql="SELECT * FROM train WHERE Train_No= '". $q."'";
}else {
$sql="SELECT * FROM train WHERE Train_Name= '". $q."'";
}
$result = mysqli_query($con,$sql);

echo "<table>
<tr>
<th>Train Number</th>
<th>Train Name</th>
<th>Source</th>
<th>Destination</th>
</tr>" ;
while($row = mysqli_fetch_array($result)) {
```

```

echo "<tr>";
echo "<td>" . $row['Train_No'] . "</td>";
echo "<td>" . $row['Train_name'] . "</td>";
echo "<td>" . $row['Source'] . "</td>";
echo "<td>" . $row['Destination'] . "</td>";
echo "</tr>";
}
echo "</table>";
mysqli_close($con);
?>
</body>
</html>

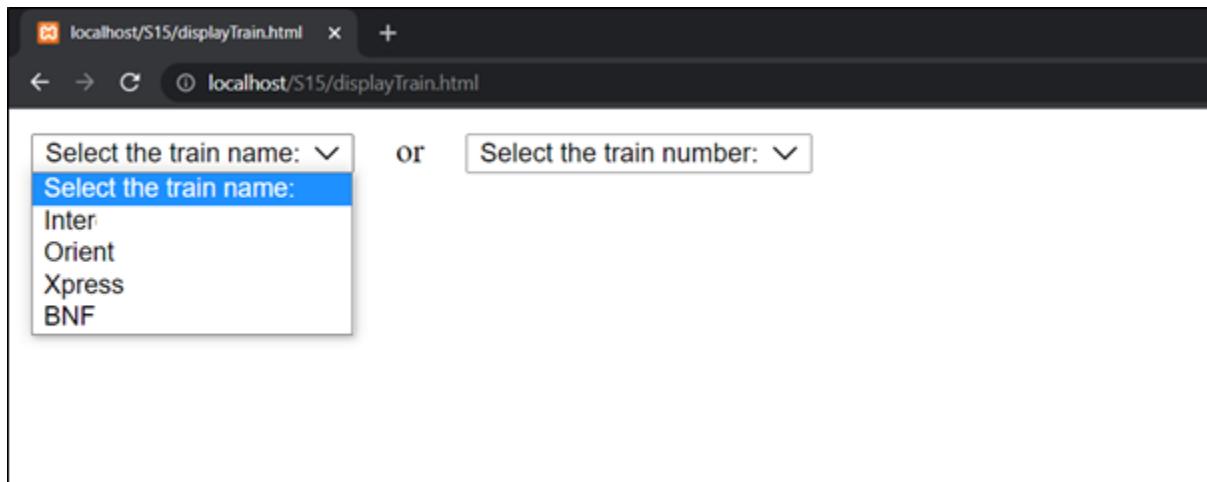
```

Once JavaScript sends the query to the PHP file, following actions will happen:

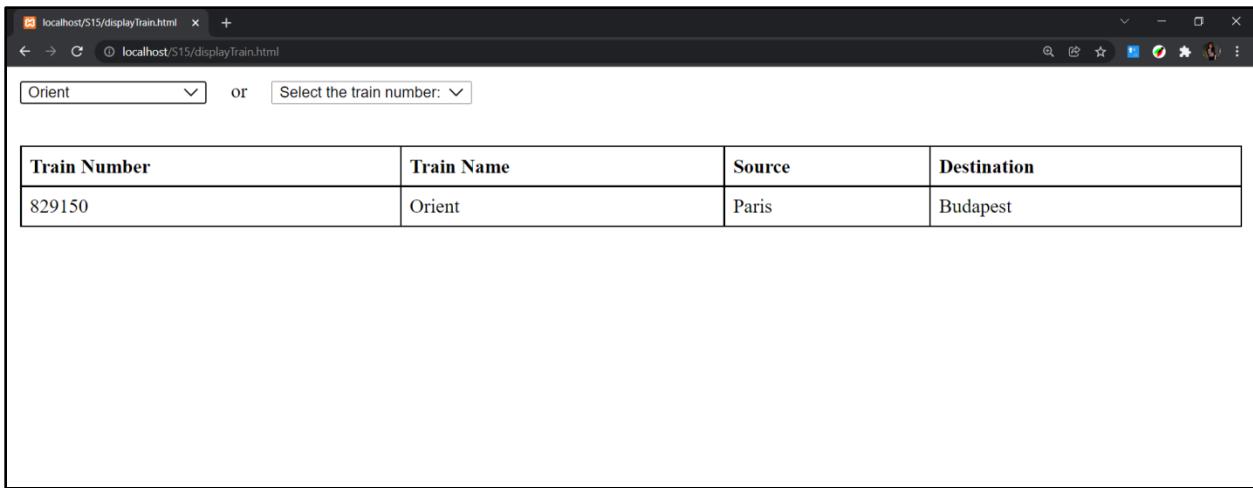
- A new connection is established by PHP to the MySQL database server.
- The corresponding train with all the details is found from database.
- An HTML table is created, filled with found data, and is sent back to the ‘result’ placeholder.

Code for these actions is implemented in Code Snippet 4.

Figures 15.4 and 15.5 show the output for Code Snippet 3 and 4 respectively.



*Figure 15.4: Web Page for Selecting Trains*



**Figure 15.5: Details of the Train Selected is Displayed**

## 15.4 AJAX and XML

---

XML is a markup language similar to HTML. XML document contains plain text along with tags enclosed within < and >. XML is commonly used as a data format. XML and AJAX is used for sending a request and receiving a response to deliver data on the Web pages.

There are two major differences between HTML and XML:

- HTML defines a certain set of tags to be used compulsorily whereas XML does not.
- XML is strict about document structure, how it is stored, and transferred, but HTML does not follow this strictly.

XML gives its users a lot more freedom than HTML. HTML has a defined set of tags for each function. For example, `<a></a>` tag surrounds a link, the `<br>` breaks a line, and so on.

Conversely, XML documents can use any tags according to user's choice such as putting `<rating></rating>` tags around a movie rating, `<height></height>` tags around a person's height, and so on. Thus, XML gives users an option to define user's own tags.

HTML allows the user to play with the tags and ignores some opening and closing tags, unlike XML.

Following example shows an invalid XML code:

```
<ul>
    <li>It's raining
    <li>It's sunny
    <li>It's cold
</ul>
```

This code is invalid in XML because, list tags, `<li>` are opened, but not closed. There should be three closing `</li>` tags, to match with three opening `<li>` tags. Every opened tag must have a closing tag in an XML document.

Following example shows a valid XML code:

```
<ul>
    <li>Baked Pie</li>
    <li>Spiced Pies</li>
    <li>Ala Kiev with Minced Pie</li>
</ul>
```

## 15.5 AJAX Live Search

---

A live search box is a search input box that displays the search results as and when the user starts typing words. It uses a reliable database to retrieve information faster. As an example, let us create a live search box that will search the employees table and show the results asynchronously.

For this example, a table `emp` is created which stores the details of the employees such as `Emp_No` and Employee name.

First, the user creates the table structure as follows:

```
CREATE TABLE emp (
    Emp_No INT(15) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL
);
```

Once the table is created, data must be inserted so that the results are fetched accordingly.

After the data is inserted, a Web page must be created for user interaction. Thus, as and when the user types to search for an employee, the employee details will

be displayed. It is quite similar to autocomplete text boxes or typeahead.

Code Snippet 5 shows a file `employeeSearch.php` written in PHP.

### Code Snippet 5: (`employeeSearch.php`)

```
<html>
  <head>
    <h1> PHP Live MySQL Database Search </h1>
    <style>
      /* Formatting search box */
      .search-box{
        width: 500px;
        position: relative;
        display: inline-block;
        font-size: 14px;
      }
      .search-box input[type="text"]{
        height: 32px;
        padding: 5px 10px;
        border: 1px solid #CCCCCC;
        font-size: 14px;
      }
    </style>
    <script>
      function displayEmployee() {
        var ajaxHttpRequest = new XMLHttpRequest();
        var name = document.getElementById('name').value;
        if(name == "") {
          document.getElementById("result").innerHTML = "";
          return;
        }

        ajaxHttpRequest.onreadystatechange = function(){
          if(ajaxHttpRequest.readyState == 4) {
            var displayResponse = document.getElementById('result');
            displayResponse.innerHTML = ajaxHttpRequest.responseText;
          }
        }

        var query = "?name=" + name ;
        ajaxHttpRequest.open("GET", "getEmployee.php" +
        query, true);
        ajaxHttpRequest.send(null);
      }

    </script>
  </head>
```

```

<body> <div class="search-box">
<input id="name" type="text" autocomplete="off"
placeholder="Search employee" onkeyup="displayEmployee()" />
<br></br>
<div id="result">Results will be displayed here...</div>
</div>
</body>
</html>

```

The keyup or input change event will occur whenever the search input value is changed. This will generate an AJAX request which is sent to the `getEmployee.php` file. It fetches records that match the data in the employee table. The records are then inserted inside a `<div>` tag which are ready to be displayed to the user.

Once a query is sent by AJAX, PHP searches the database and sends the results to the browser. Code Snippet 6 shows the PHP code for `getEmployee.php`.

### **Code Snippet 6: (`getEmployee.php`)**

```

<?php
$db_host = "localhost";
$db_user = "root";
$db_pass = "";
$db_name = "test01";

//Establishing the connection
$con = mysqli_connect($db_host, $db_user, $db_pass, $db_name)
or die(mysql_error());
$name = $_GET['name'];
$age = mysqli_real_escape_string($con, $name);
//Constructing the query with condition
$sql = "SELECT * FROM emp WHERE Name LIKE '$name%'";
//Retrieving the result from the database
$result = mysqli_query($con, $sql);
//Displaying the employee details
if(mysqli_num_rows($result) > 0){

    while($row = mysqli_fetch_array($result)){
        echo "<p>" . $row["Emp_No"] . "&nbsp;" .
"&nbsp;" . $row["Name"] . "</p>" ;
    }

} else {
    echo "<p>No matches found</p>";
}

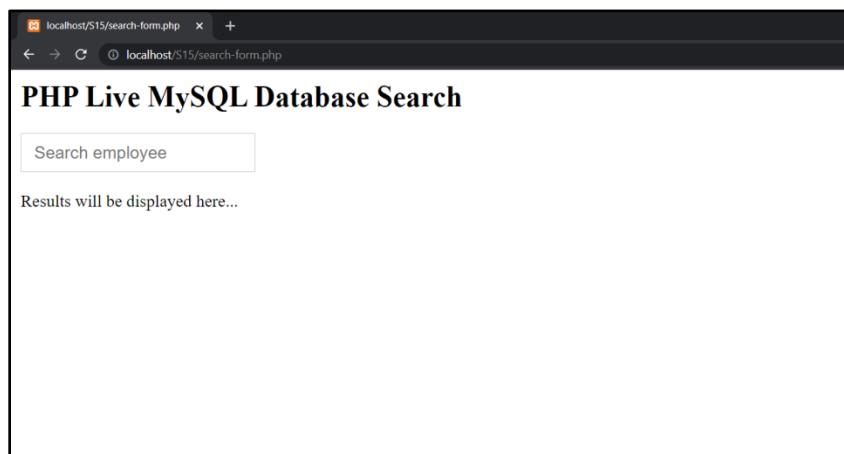
```

```
        }
    mysqli_close($con);
?>
```

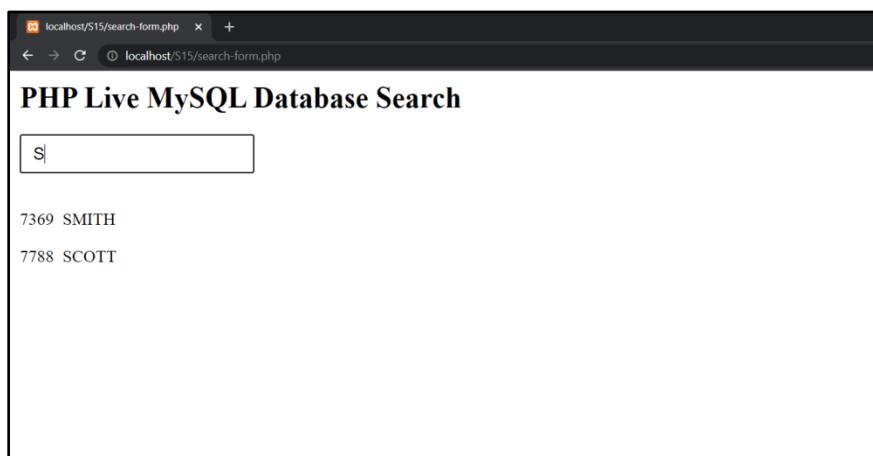
The SQL SELECT and LIKE operators are used to match records and fetch the Emp\_No and Name as results. If the entered value is not matching with any record then ‘No matches found’ result is displayed.

One can make use of PHP `mysqli_real_escape_string()` method to truncate special characters present in the input. This will avoid any additional spaces and special characters and make it a valid SQL string.

Figures 15.6 and 15.7 show the output for Code Snippets 5 and 6 respectively.



*Figure 15.6: PHP Live Search Box*



*Figure 15.7: Output of Server-side Code*

## 15.6 Summary

- AJAX is used with PHP for asynchronous applications.
- Without AJAX, Web pages reload the entire page instead of updating the page, which results in server overhead and slower performances.
- AJAX combined with PHP and MySQL establishes a smooth and strong communication with the database and the server, improving the performance of applications.
- XML is widely used as a data format to send a request and receive a response.
- AJAX on Web pages are executed using an XMLHttpRequest object.
- AJAX improves user experience, supports multiple browsers, and has a good response time.

## 15.7 Test Your Knowledge



1. AJAX is based on \_\_\_\_\_.
  - a. JavaScript and XML
  - b. JavaScript and Java
  - c. VB Script and XML
  - d. JavaScript and HTTP requests
  
2. When does a browser create an XMLHttpRequest?
  - a. When HttpRequest is Processed
  - b. When an event occurs on a Web Page
  - c. When data is sent back to the browser
  - d. When HttpRequest is processed
  
3. Which property returns the response data as a string?
  - a. getText
  - b. responseText
  - c. responseAllText
  - d. getResponseText
  
4. What is the role of XMLHttpRequest object in AJAX?
  - a. It is used to send and receive data synchronously with a server
  - b. It is used to process the request
  - c. It is used to display the results
  - d. It is used to send and receive data asynchronously with a server
  
5. What are the advantages of AJAX?
  - a. Better user interaction
  - b. Faster data retrieval
  - c. Bandwidth utilization
  - d. All of these

## **Answers to Test Your Knowledge**

---

1. JavaScript and XML
2. When an event occurs on Web page
3. `responseText`
4. It is used to send and receive data asynchronously with a server
5. All of these

## 15.8 Try It Yourself

1. Write a PHP program to display a list of students. On selecting a student, the particular student's details must be fetched from Student table defined in a MySQL database and displayed in a table format. Use AJAX to implement this.
2. Write a PHP program to insert a new employee into an employee table in a MySQL database. The employee table must be displayed and updated each time a new employee is inserted. Use AJAX to implement this.



# Laravel Framework for Web Applications with PHP





# Session 01: Setting up the Environment



## Objectives

- Identify components to set up the development environment
- Learn to set up PHP and Composer
- Understand the basics of VirtualBox, Vagrant, Homestead, and Laravel
- Learn how to verify installations and versions of various software



### 1.1 Setting up the Environment

Before developing Web applications with Laravel, it is important to setup an environment that optimizes the development process.

#### 1.1.1 Identifying Components



1. **PHP:** PHP should be installed on the Windows host operating system to use composer.
2. **Composer:** It is a dependency management tool for PHP. In the host operating system, Composer will be used to automate installation of Homestead along with vagrant.
3. **VirtualBox:** It is a free and open source hypervisor that will allow running guest VMs, such as the Homestead environment.
4. **Vagrant:** It is an automation tool for building and managing virtual machine environments. Homestead is a vagrant box.
5. **Git Bash:** It is a shell that comes along with Git for Windows. It helps use git commands and provides Unix-type commands that can be run on Windows. Git Bash will be used to generate SSH keys.
6. **Homestead:** Homestead is a virtual environment that provides resources to efficiently code Web applications in Laravel using PHP. It comes with PHP and composer pre-installed in it

## Knowledge Check 1:

Q1. Which of the following provides Unix-type commands that can be run on Windows?

- a. Homestead
- b. Vagrant
- c. Git Bash
- d. Composer

## 1.1.2 Setting up PHP



2. Add the path of the extracted php files to the environment variables. To add the path, follow the steps given at following URL:  
<https://secure.php.net/manual/en/faq.installation.php#faq.installation.addtopath>

To use composer and homestead on Windows, PHP binaries are required. Download them from PHP's official Website <https://windows.php.net/download/>.

1. Depending on the architecture, download the latest x86/x64 Non Thread Safe version of PHP from the official Website. Figure 1.1 shows the different PHP versions.

The screenshot shows the PHP website with the title "PHP: Hypertext Preprocessor". Under the "PHP For Windows" section, it says "This site is dedicated to supporting PHP on Microsoft Windows. It also supports ports of PHP extensions or features as well as providing special builds for the various Windows architectures." Below this, there is a link to "Wiki". The main content area lists three PHP 7.3 versions for Windows:

- VC15 x64 Non Thread Safe (2019-Feb-06 02:14:41)**
  - [Zip](#) [24.19MB]
  - sha256: 37803f927a4f5bfcff996d5fab7092fb0fd01c9390de501e5bcf89ff0f74595c
  - [Debug Pack](#) [22.82MB]
  - sha256: 69ae45a349212f7865954504c54d4e12faef3c31b0bf0e0a8daf28f5d27ebcc
- VC15 x64 Thread Safe (2019-Feb-06 02:14:58)**
  - [Zip](#) [24.31MB]
  - sha256: e4391dce3d431ccaf64372212a0dafcd8cf3d4aa79b684ea225cf32ffdb238
  - [Debug Pack](#) [22.9MB]
  - sha256: 527d5f874716b8e68ff824b3ca8d44d8e4cae36b0dde40777b40b83bdb22ff61
- VC15 x86 Non Thread Safe (2019-Feb-06 02:14:42)**
  - [Zip](#) [22.56MB]

Figure 1.1: PHP Versions

## 1.1.3 Setting up Composer

Installing composer on a Windows operating system is very easy. It can be easily downloaded and installed from the official Website  
<https://getcomposer.org/download/>.

Go to the Website and click **Composer-Setup.exe** as shown in Figure 1.2 that will download the setup for installing composer on Windows.

## Download Composer      Latest: v1.8.5

### Windows Installer

The installer will download composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

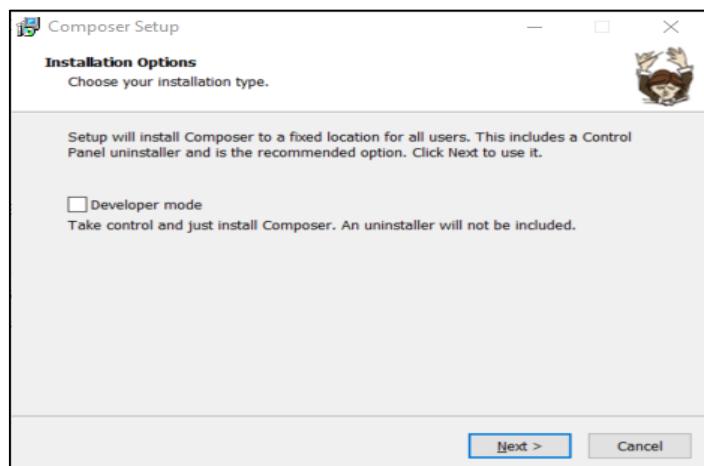
Download and run [Composer-Setup.exe](#) - it will install the latest composer version whenever it is executed.

**Figure 1.2: Composer-Setup.exe**

After the download, start the setup to initiate an installation wizard. Next, perform the following steps:



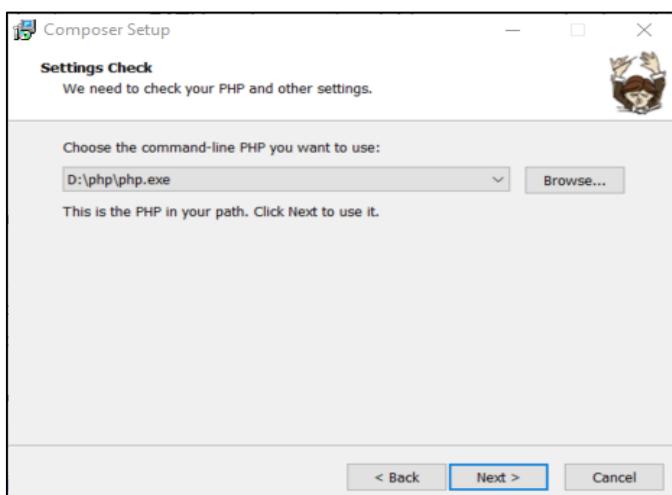
On this **Installation Options** page as shown in Figure 1.3, click **Next** to proceed further.



**Figure 1.3: Installation Options Page**



On the **Settings Check** page as shown in Figure 1.4, select the required php version exe path.



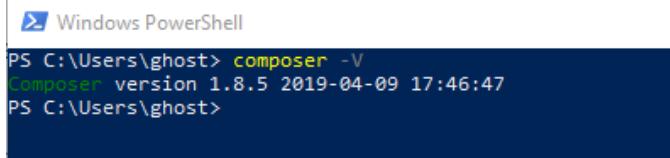
**Figure 1.4: Settings Check Page**



During this step if it prompts for the path to the PHP interpreter, then the PHP path in the environment variables have not been configured correctly.

3

Click **Next** on all successive pages till the wizard tells you that the Composer is successfully installed.



```
Windows PowerShell
PS C:\Users\ghost> composer -v
Composer version 1.8.5 2019-04-09 17:46:47
PS C:\Users\ghost>
```

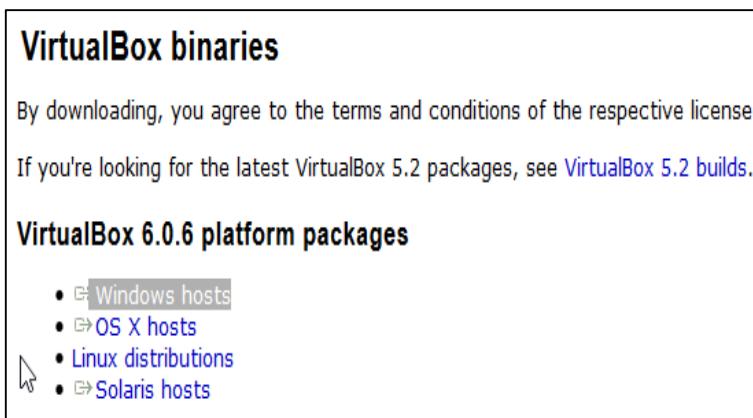
Figure 1.5: Verifying Composer

To verify installation, open cmd or PowerShell and type the command as shown in Figure 1.5.



## 1.1.4 Installing VirtualBox

To Download and install VirtualBox from <https://www.virtualbox.org/wiki/Downloads> perform the following steps:



**VirtualBox binaries**  
By downloading, you agree to the terms and conditions of the respective license.  
If you're looking for the latest VirtualBox 5.2 packages, see [VirtualBox 5.2 builds](#).

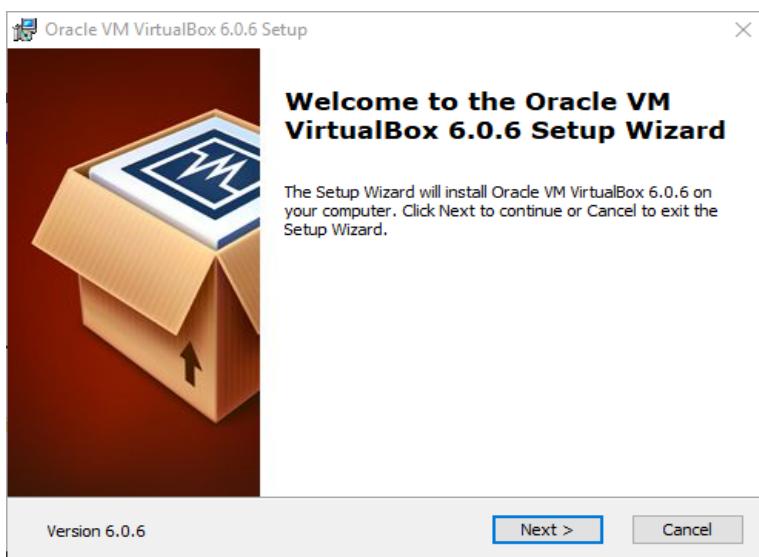
**VirtualBox 6.0.6 platform packages**

- Windows hosts
- OS X hosts
- Linux distributions
- Solaris hosts

Figure 1.6: Windows hosts

1

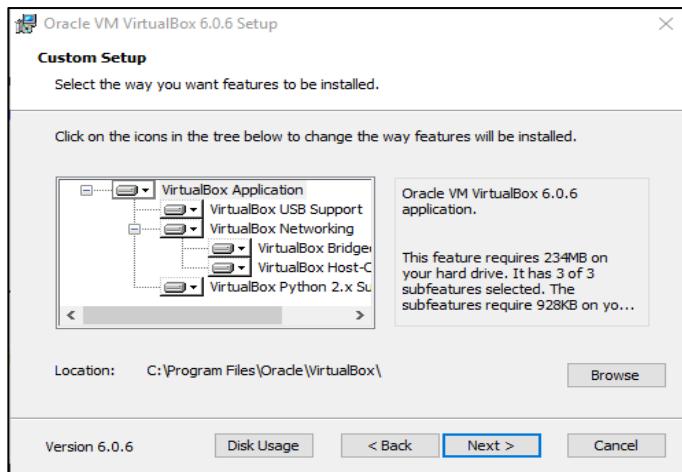
To download the setup, on the Website, click **Windows hosts** as shown in Figure 1.6.



2

Run the setup after downloading is finished an installation wizard appears. On the Welcome to the Oracle VM VirtualBox 6.0.6 Setup Wizard page as shown in Figure 1.7, click **Next**.

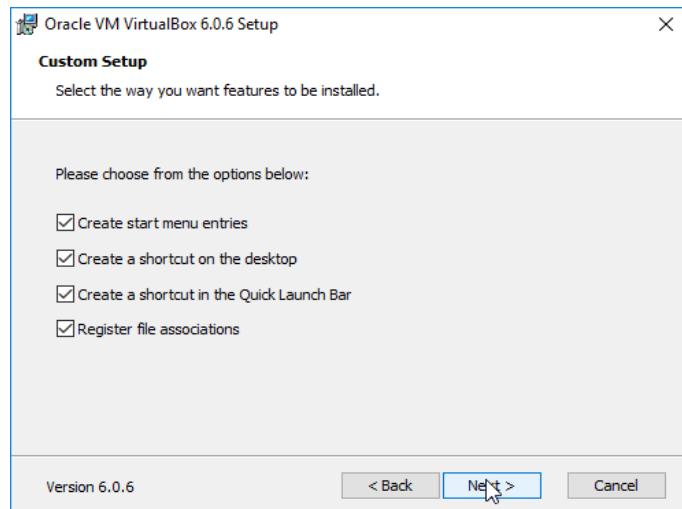
Figure 1.7: Welcome to the Oracle VM Wizard Page



3

On the **Custom Setup** page as shown in Figure 1.8, click **Next** to proceed further.

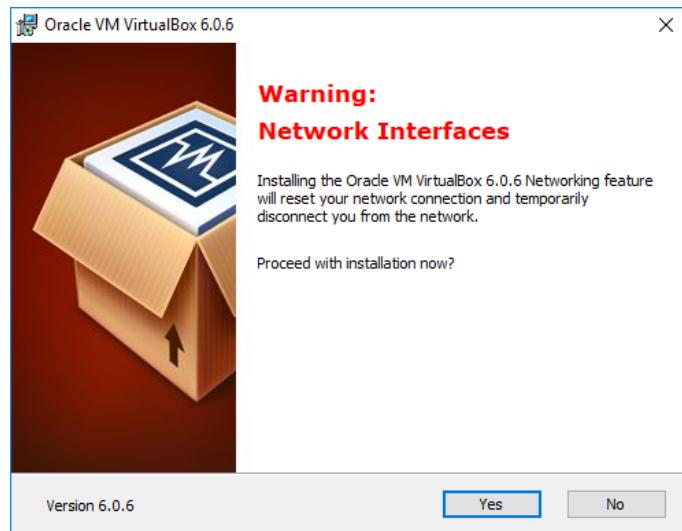
Figure 1.8: Custom Setup Page



4

To select the way for installing features, select the required options and click **Next** as shown in Figure 1.9.

Figure 1.9: Options to Install Features



5

On the Warning page as shown in Figure 1.10, click **Yes** to proceed further.

Figure 1.10: Warning Page

6

On the Installation Complete page as shown in figure 1.11, click **Finish**. This completes the VirtualBox setup.



Figure 1.11: Installation Complete Page

## 1.1.5 Installing Vagrant

Similar to composer and VirtualBox, Vagrant is easy to install. For installation, perform the following steps:

1. Visit the official Website, <https://www.vagrantup.com/downloads.html> as shown in Figure 1.12.

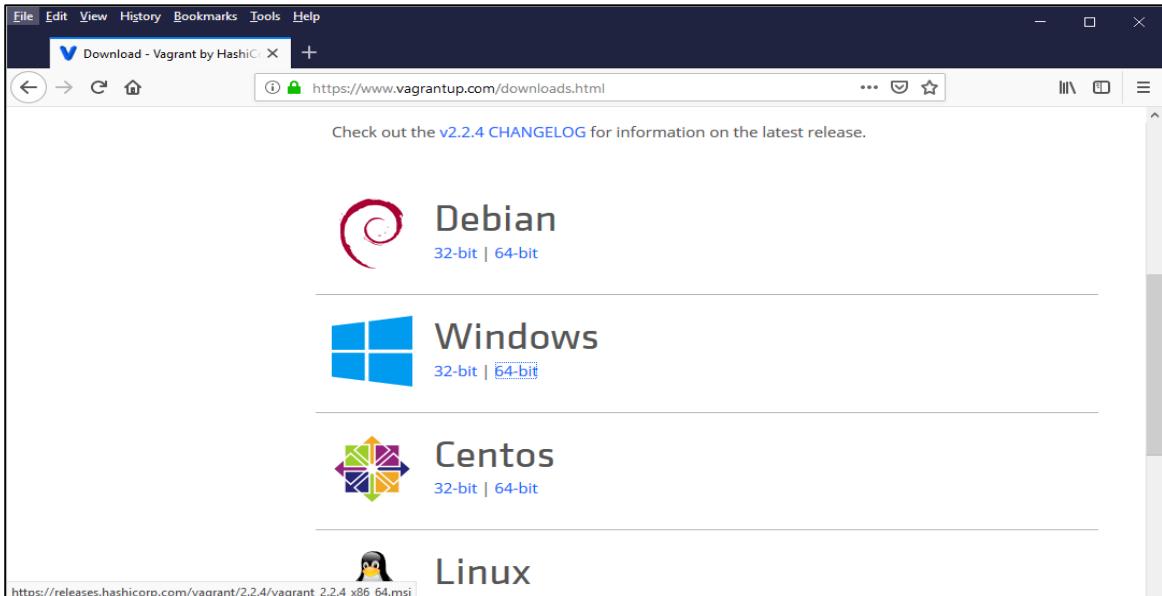


Figure 1.12: Official Website to Download Vagrant

2. Depending on the architecture, select either 32-bit or 64-bit version and download the file.
3. Open the file to display the wizard.
4. On the Welcome to the Vagrant Setup Wizard page as shown in Figure 1.13, click **Next**.

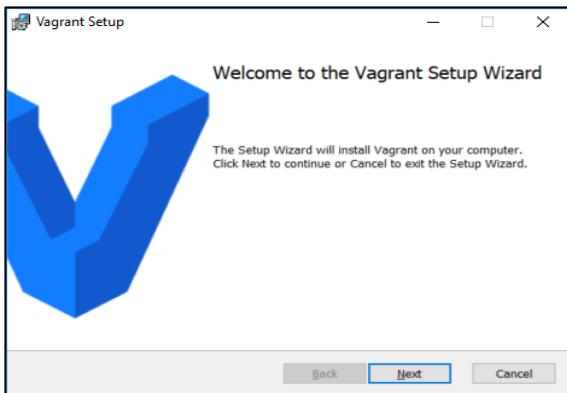


Figure 1.13: Vagrant Setup Wizard

5. On the **End-User License Agreement** page as shown in Figure 1.14, click **Next**.

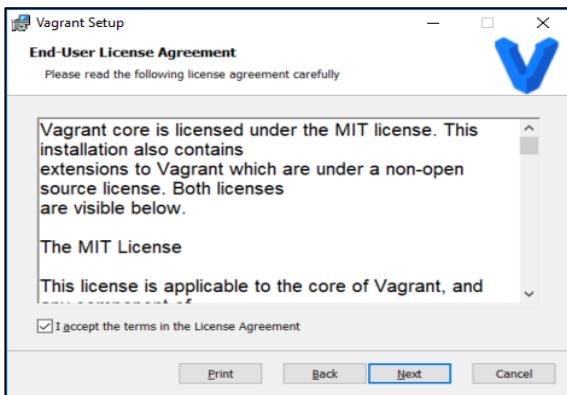


Figure 1.14: End-User License Agreement Page

6. On the **Destination Folder** page that displays as shown in Figure 1.15, choose the installation directory and click **Next**.

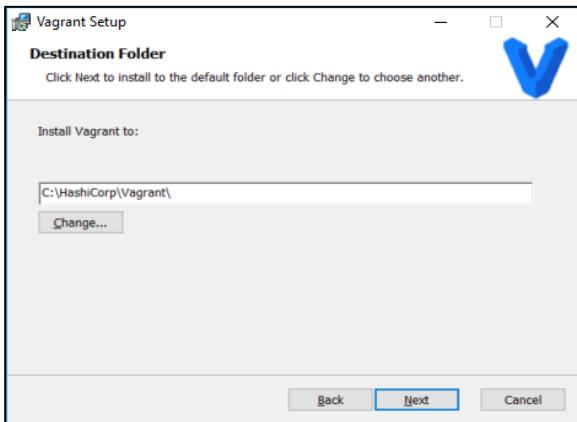


Figure 1.15: Destination Folder Page

7. On the **Ready to install Vagrant** page that displays as shown in Figure 1.16, click **Install**.

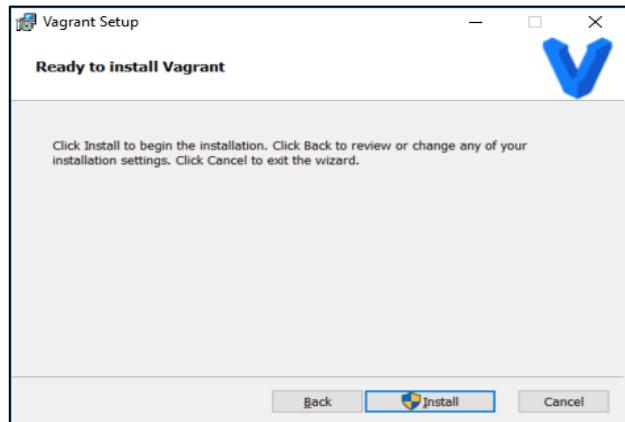


Figure 1.16: Ready to install Vagrant Page

The installation process begins as shown in Figure 1.17.

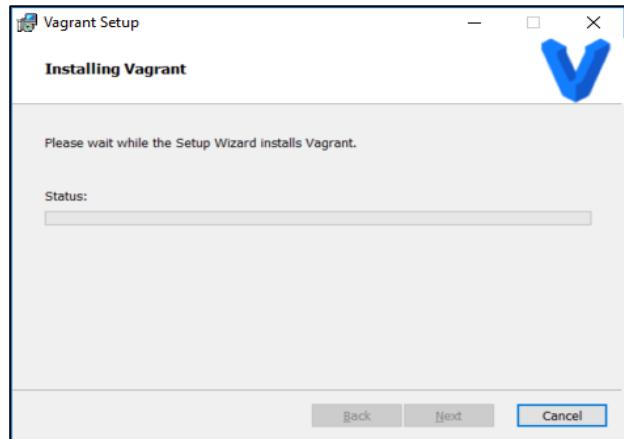
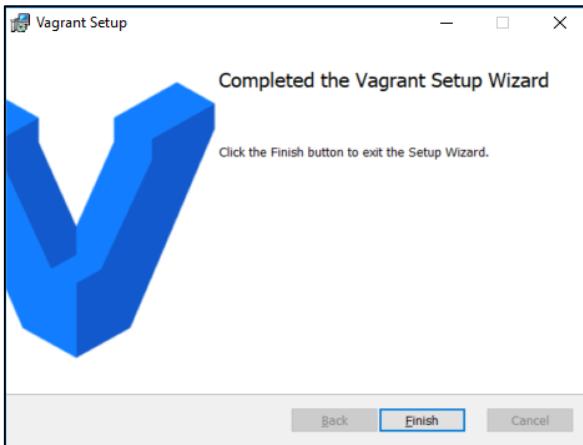


Figure 1.17: Vagrant Installation Process

8. Once the installation is complete, on the Completed Vagrant Setup page, as shown in Figure 1.18, click **Finish**.



**Figure 1.18: Completed Vagrant Setup Page**

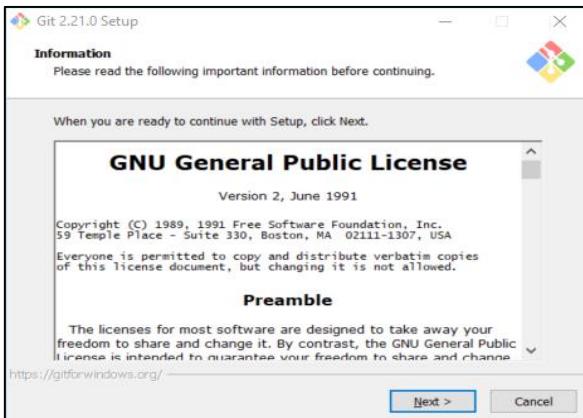
Vagrant is installed successfully now.

## 1.1.6 Setting up Gitbash

To setup Gitbash, perform the following steps:

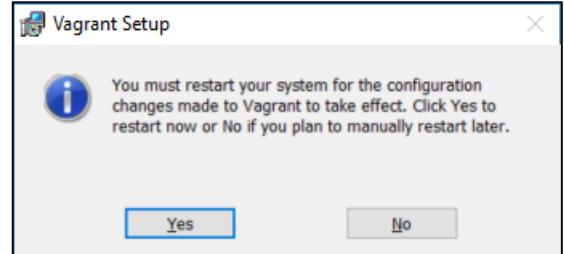
To download the Git executable file, visit the Website, <https://git.scm.com/download/win>. After downloading is finished follow the following steps:

1. Execute the .exe file.
2. **Information** page displays GNU License information as shown in Figure 1.20 and click **Next** to visit next screen.



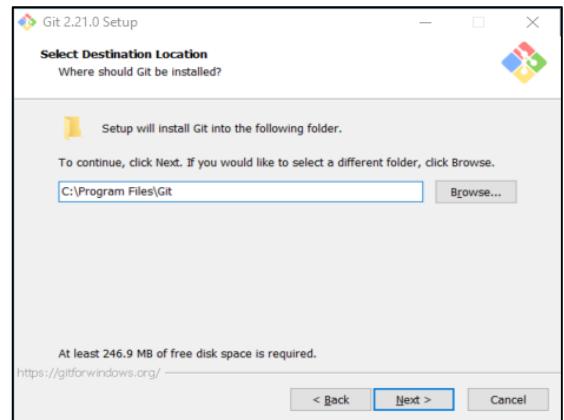
**Figure 1.20: Information Page**

9. For the configuration changes to take effect, a dialog box displays as shown in Figure 1.19 that prompts to restart the system. Click **Yes**.



**Figure 1.19: Prompt to Restart the System**

3. On the **Select Destination Location** page select required destination as shown in Figure 1.21 and click **Next** to visit next screen.



**Figure 1.21: Select Destination Location Page**

4. On the **Select Components** page select required components as shown in Figure 1.22 and click **Next** to visit next screen.

# Laravel Framework for Web Applications with PHP

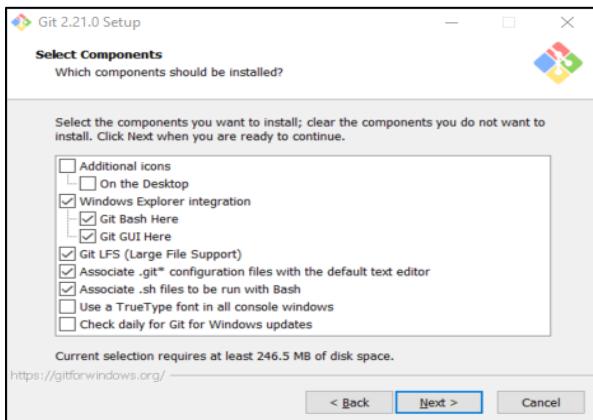


Figure 1.22: Select Components Page

5. On the **Select Start Menu Folder** page, select required folder name as shown in Figure 1.23 and click **Next** to visit next screen.

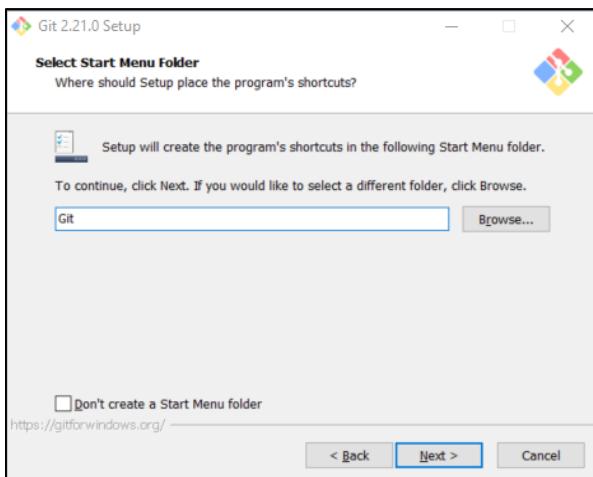


Figure 1.23: Select Start Menu Folder Page

6. To choose the default editor, select editor you want to use as shown in Figure 1.24 and click **Next** to visit next screen.

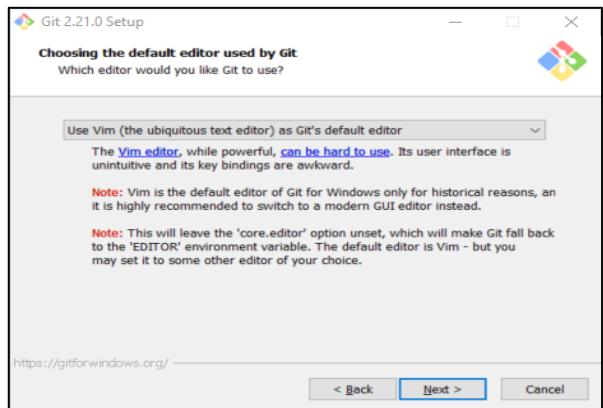


Figure 1.24: Editor Page

7. Select Git from the command line, on the Path Environment page as shown in Figure 1.25 and click **Next** to visit next screen.

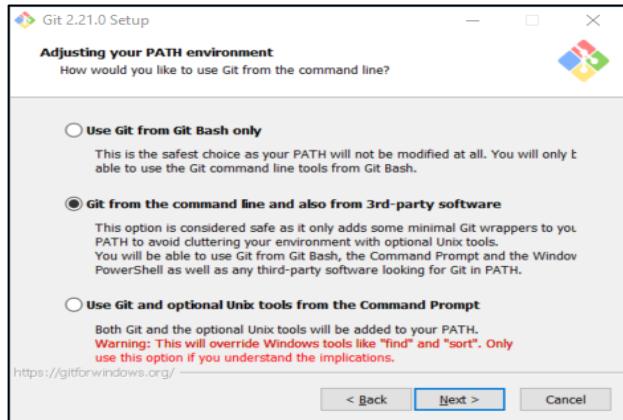


Figure 1.25: Path Environment Page

8. Select the OpenSSL library as the HTTPS transport backend, on the HTTPS transport backend page as shown in Figure 1.26 and click **Next** to visit next screen.

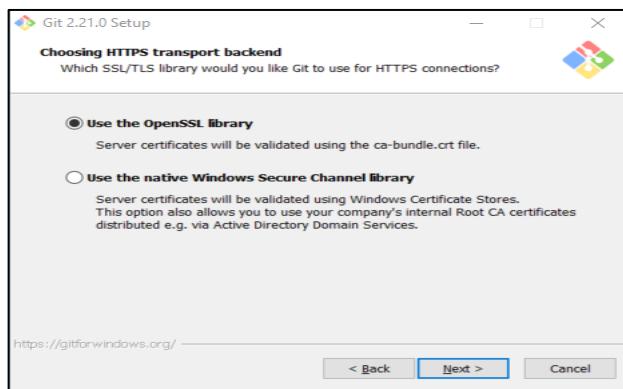


Figure 1.26: HTTPS Transport Backend Page

# Laravel Framework for Web Applications with PHP

9. Check text files in Windows-style or not, on the line ending Conversions page as shown in Figure 1.27 and click **Next** to visit next screen.

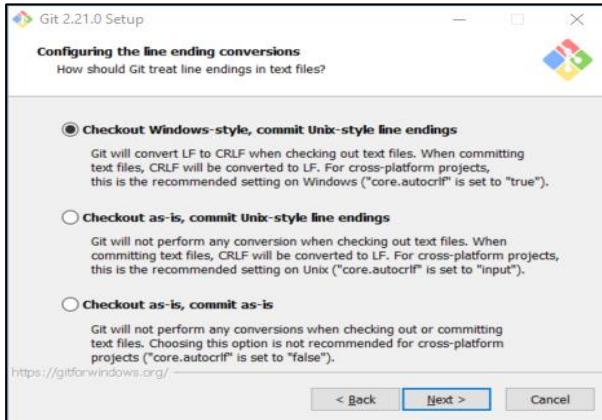


Figure 1.27: Line Ending Conversions Page

11. To configure extra options select required options, on the **Configuring extra options** page as shown in Figure 1.29. Click **Install** to start the installation.

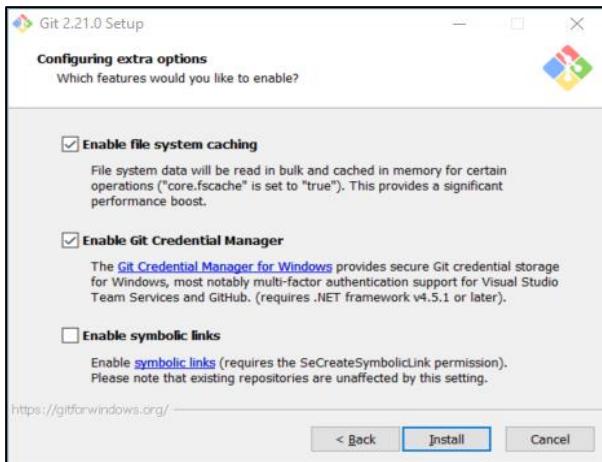


Figure 1.29: Configuring Extra Options Page

13. To verify installation, right-click any empty space on the desktop and select **Git Bash Here** as shown in Figure 1.31.

10. To use MinTTY as the terminal emulator, select **Use MinTTY** as shown in Figure 1.28 and click **Next** to proceed further.

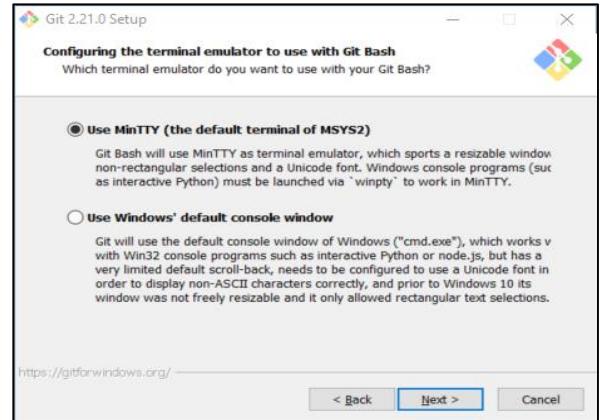


Figure 1.28: Terminal Emulator

12. To exit setup click **Finish**, on the **Completing the Git Setup Wizard** page as shown in Figure 1.30.

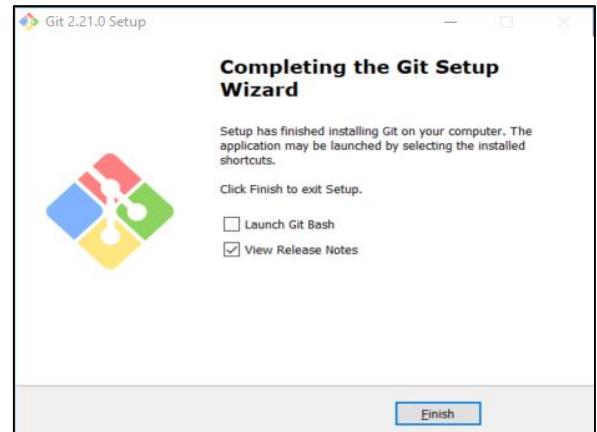


Figure 1.30: Completing the Git Setup Wizard Page

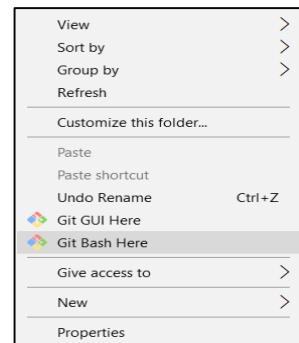


Figure 1.31: Verifying Git Bash Installation

## 1.1.7 Setting up Homestead

Homestead is a vagrant box. To install Homestead, perform the following steps:

1. Use the command vagrant box add laravel/homestead on any other command line tools such as cmd, PowerShell, and Git Bash as shown in Code Snippet 1.

### Code Snippet 1:

```
vagrant box add laravel/homestead
```

2. Select virtualbox as hypervisor when prompted as shown in Figure 1.32.

This box can work with multiple providers! The providers that it can work with are listed below. Please review the list and choose the provider you will be working with.

- 1) hyperv
- 2) parallels
- 3) virtualbox
- 4) vmware\_desktop

Figure 1.32: Virtualbox as Hypervisor

3. Use the composer command as shown in Code Snippet 2.

### Code Snippet 2:

```
mkdir homestead  
cd homestead  
composer require laravel/homestead --dev  
.\\vendor\\bin\\homestead make
```

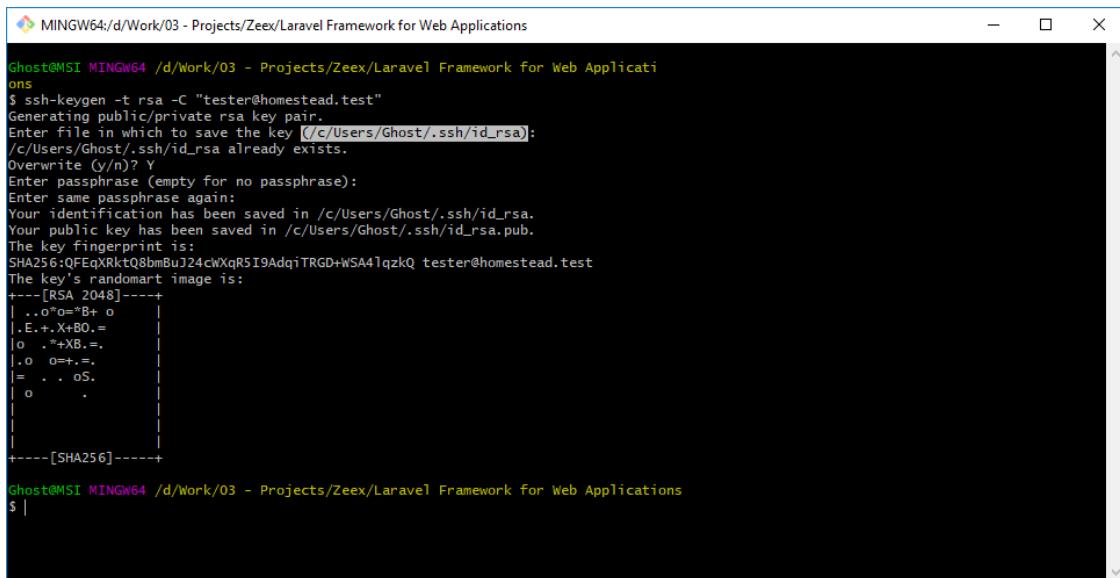
4. To configure the ssh keys, start Git Bash and input command as shown in Code Snippet 3.

### Code Snippet 3:

```
ssh-keygen -t rsa -C "tester@homestead.test"
```

# Laravel Framework for Web Applications with PHP

This displays a command line wizard as shown in Figure 1.33 to generate ssh keys.



```
MINGW64:/d/Work/03 - Projects/Zeex/Laravel Framework for Web Applications
$ ssh-keygen -t rsa -C "tester@homestead.test"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Ghost/.ssh/id_rsa):
/c/Users/Ghost/.ssh/id_rsa already exists.
Overwrite (y/n)? Y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Ghost/.ssh/id_rsa.
Your public key has been saved in /c/Users/Ghost/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:QEqXRktQ8bmBuJ24cWxqR5I9AdqiTRGD+W5A4TqzkQ tester@homestead.test
The key's randomart image is:
+---[RSA 2048]---+
| ..o*o*Bt o
| .E.+XeB0.=
| o .^+XB.=.
| .o o=+=.=.
|= . . OS.
| o .
|
+---[SHA256]---+
Ghost@MSI MINGW64 /d/Work/03 - Projects/Zeex/Laravel Framework for Web Applications
$ |
```

Figure 1.33: Generating ssh Keys

5. Copy the address as shown in Figure 1.33 to the ssh keys. The address generated is in Unix style.
6. To convert the Unix style address into Windows style map /c/Users/Ghost/.ssh/id\_rsa to C:\Users\Ghost\.ssh\id\_rsa.
7. In the Homestead directory that is created, modify the **Homestead.yaml** file and change the values depending on the desired locations as shown in Example 1.

## Example 1:

```
ip: 192.168.10.10
memory: 1024
cpus: 1
provider: virtualbox
authorize: C:\Users\Ghost\.ssh\id_rsa.pub
keys:
  - C:\Users\Ghost\.ssh\id_rsa
folders:
  -
    map: 'D:\Work\Laravel Framework for Web Applications\homestead'
    to: /home/vagrant/code
```

```
sites:
  -
    map: homestead.test
    to: /home/vagrant/code/public
databases:
  -
    homestead
name: homestead
hostname: homestead
```



RAM can be set to any value.

8. Note down the IP address and host in the .yaml file.
9. Set up the Homestead environment. Follow the steps to do so:
  - a. Open a powershell/cmd/git bash window in the homestead directory and run the command as shown in Code Snippet 4.

#### Code Snippet 4:

```
vagrant up
```

The homestead environment is successfully configured as shown in Figure 1.34.

```
Windows PowerShell
Homestead vagrant up
Bringing machine 'homestead' up with 'virtualbox' provider...
==> homestead: Importing base box 'laravel/homestead'...
==> homestead: Matching MAC address for NAT networking...
==> homestead: Checking if box 'laravel/homestead' version '7.1.0' is up to date...
==> homestead: Setting the name of the VM: homestead
==> homestead: Clearing any previously set network interfaces...
==> homestead: Preparing network interfaces based on configuration...
==> homestead: Adapter 1: nat
    homestead: Adapter 2: hostonly
    homestead: Forwarding ports...
    homestead: 80 (guest) => 8000 (host) (adapter 1)
    homestead: 443 (guest) => 43000 (host) (adapter 1)
    homestead: 3306 (guest) => 33060 (host) (adapter 1)
    homestead: 4040 (guest) => 4040 (host) (adapter 1)
    homestead: 5432 (guest) => 54320 (host) (adapter 1)
    homestead: 8080 (guest) => 2222 (host) (adapter 1)
    homestead: 5701 (guest) => 2217 (host) (adapter 1)
    homestead: 22 (guest) => 2222 (host) (adapter 1)
    homestead: Running 'pre-boot' VM customizations...
==> homestead: Booting VM...
==> homestead: Waiting for machine to boot. This may take a few minutes...
    homestead: SSH address: 127.0.0.1:2222
    homestead: SSH username: vagrant
    homestead: SSH auth method: private key
    homestead: 
    homestead: Vagrant insecure key detected. Vagrant will automatically replace
    homestead: this with a newly generated keypair for better security.
    homestead: Inserting generated public key within guest...
    homestead: Machine booted and ready
    homestead: Cleaning up any extra interfaces in VM...
    homestead: The guest additions on this VM do not match the installed version of
    homestead: VirtualBox! In most cases this is fine, but in rare cases it can
    homestead: prevent things such as shared folders from working properly. If you see
    homestead: shared folder errors please make sure the guest additions within the
    homestead: virtual machine match the version of VirtualBox you have installed on
    homestead: your host and reload your VM.
    homestead: Guest Additions Version: 5.2.8_kernelUbuntu r120774
    homestead: VirtualBox Version: 6.0
    homestead: setting host-only adapter
    homestead: Configuring and enabling network interfaces...
    homestead: Mounting shared folders...
    homestead: /Vagrant => D:/Work/03 - Projects/Zex/Laravel Framework for Web Applications/homestead
    homestead: /home/vagrant/code/homestead => D:/Work/03 - Projects/Zex/Laravel Framework for Web Applications/homestead
    homestead: Run provisioning: file
    homestead: D:/Work/03 - Projects/Zex/Laravel Framework for Web Applications/homestead/aliases => /tmp/bash_aliases
    homestead: Running provisioner: shell...
```

Figure 1.34: Homestead Environment

## 1.1.8 Setting Up Laravel

The Homestead virtual environment includes all tools required to create a Laravel application.

Laravel is not an application, it is a framework. Laravel command line utility as shown in Code Snippet 5, it can be used to create an application based on the Laravel framework.

### Code Snippet 5:

```
laravel new <project-name>
```

## 1.1.9 Verifying Installation

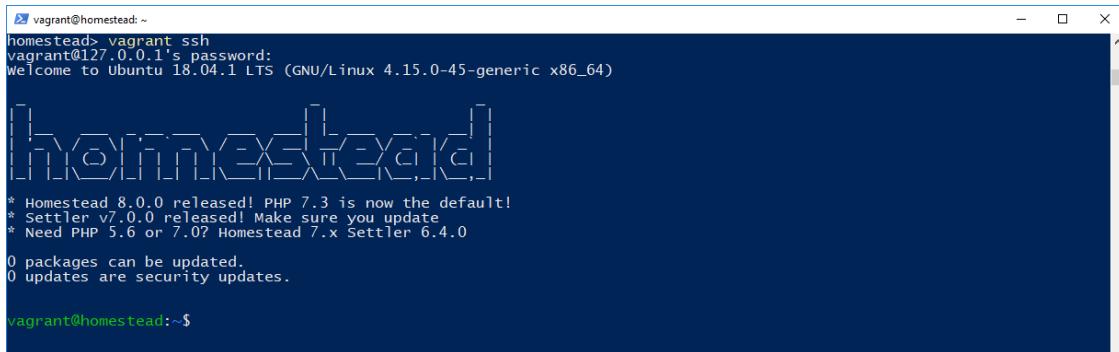
Once all components are set, it is time to verify the installation and check version number for newly deployed Homestead environment.

To do so, type the following command in PowerShell as shown in Code Snippet 6.

### Code Snippet 6:

```
vagrant ssh
```

This prompts for a password as shown in Figure 1.35. The default password is vagrant.



```
vagrant@homestead: ~
homestead: vagrant ssh
vagrant@127.0.0.1's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-45-generic x86_64)

 * Homestead 8.0.0 released! PHP 7.3 is now the default!
 * Settler v7.0.0 released! Make sure you update
 * Need PHP 5.6 or 7.0? Homestead 7.x Settler 6.4.0

0 packages can be updated,
0 updates are security updates.

vagrant@homestead:~$
```

Figure 1.35: Verifying Homestead Environment

## Verifying Versions

Code Snippet 7, shows how to verify the PHP version in the ssh session.

### Code Snippet 7:

```
vagrant@homestead:~$ php --version
```

Following is the output for Code Snippet 7.

```
PHP 7.3.1-1+ubuntu18.04.1+deb.sury.org+1 (cli) (built: Jan 13 2019 10:19:33) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.1, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.3.1-1+ubuntu18.04.1+deb.sury.org+1, Copyright (c) 1999-201
8, by Zend Technologies
    with blackfire v1.24.3~linux-x64-non_zts73, https://blackfire.io, by Blackfire
```

Code Snippet 8 shows how to verify the composer version in the ssh session.

### Code Snippet 8:

```
vagrant@homestead:~$ composer -V
```

Following is the output for Code Snippet 8.

```
Composer version 1.8.4 2019-02-11 10:52:10
```

Code Snippet 9 shows how to verify the SQLite version in the ssh session.

### Code Snippet 9:

```
vagrant@homestead:~$ sqlite3 -version
```

After executing code in Code Snippet 9, it shows following output:

```
3.22.0 2018-01-22 18:45:57 0c55d179733b46d8doba4d88e01a25e10677046ee3da1d5b1
581e86726f2alt1
```

Code Snippet 10 shows how to verify the Homestead version in the ssh session.

## Code Snippet 10:

```
vagrant@homestead:~$ ./code/vendor/bin/homestead -v
```

After executing code from Code Snippet 10, it shows Laravel Homestead version as follows:

```
Laravel Homestead 8.2.0
```

## Knowledge Check 2:

Q2. Following code is used to verify: `vagrant@homestead:~$ php -version`

- a. SQLite
- b. PHP
- c. Homestead
- d. Vagrant

Onlinevarsity



WHERE THE EXPERTS SPEAK THE EXPERIENCE

## SUMMARY

-  Components in an environment to develop Web applications with Laravel include PHP, Composer, Virtual Box, Vagrant, Git Bash, and Homestead.
-  Composer is a dependency management tool for PHP.
-  VirtualBox is a free and open source hypervisor that will allow running guest VMs, such as the Homestead environment.
-  Vagrant is an automation tool for building and managing virtual machine environments. Homestead is a vagrant box.
-  Git Bash is a shell that comes along with Git for Windows. It helps to use git commands and provides Unix-type commands that can be run on Windows.
-  Homestead is a virtual environment that provides resources to efficiently code Web applications in Laravel using PHP.
-  After installing the components, verify the installation and check version number by sshing into the newly deployed Homestead environment.



## Check Your Progress

Q1. Which one of the following components is an open source hypervisor that allows running guest VMs, such as the Homestead environment?

- |              |               |
|--------------|---------------|
| A. Vagrant   | B. VirtualBox |
| C. Homestead | D. Composer   |

Q2. \_\_\_\_\_ is used to generate SSH keys.

- |               |              |
|---------------|--------------|
| A. Git Bash   | B. PHP       |
| C. VirtualBox | D. Homestead |

Q3. The \_\_\_\_\_ virtual environment includes all tools required to create a Laravel application.

- |              |               |
|--------------|---------------|
| A. PHP       | B. VirtualBox |
| C. Homestead | D. Git Bash   |

Q4. Which one of the following is a vagrant box?

- |              |               |
|--------------|---------------|
| A. Git Bash  | B. PHP        |
| C. Homestead | D. VirtualBox |

Q5. Which one of the following is the next step after extracting PHP files?

- |   |  |
|---|--|
| A. Add the path of the extracted php files to the environment variables | B. Add the path of the extracted php files to the Composer |
| C. Install Git Bash   | D. Run the Vagrant command                                 |



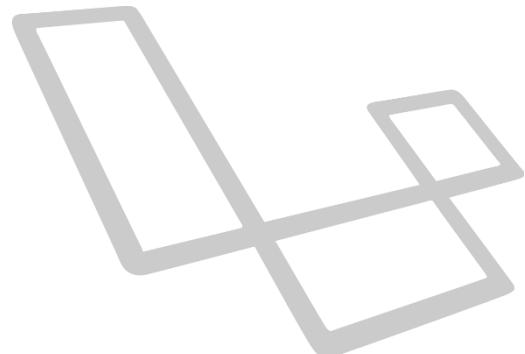
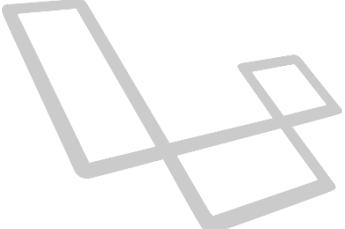
# Quiz answers

Question No	Answers
1	B
2	A
3	C
4	C
5	A



# Knowledge Check Answers

Question No	Answers
1	C
2	B



# Session 02:

# Introduction to Laravel



## Objectives

- Explain the details of Laravel framework
- Understand the necessity of frameworks
- Describe the features of Laravel
- Learn to view the Laravel Directory structure



## 2.1 Understanding the Laravel Framework

Laravel is an open source PHP framework for users working on Web application development. Considered as an alternative to CakePHP and CodeIgniter, Laravel uses the Model-View-Controller (MVC) design pattern to facilitate development of Web applications in a quick and easy manner.

### DID YOU KNOW?



Official documentation for Laravel 4.2 defines it as follows:

Laravel is a Web application framework with expressive, elegant syntax. Laravel eases the use of common tasks used in the majority of Web projects, such as authentication, routing, sessions, and caching.



For more information on Laravel, visit <https://laravel.com>.

For information about how Laravel works, visit  
<https://laravel.com/docs/5.7>



## 2.2 Necessity for Frameworks

Hypertext Processor, commonly known as PHP, is an HTML based scripting language that is widely used for Web development.

**DID YOU  
KNOW?**



PHP is also one of the first languages learnt by an aspiring Web developer.

PHP was earlier called Personal Home Page and has undergone many version updates to reach its current form. Due to its ease of working, it is one of the most widely supported programming languages, especially by the open source community. The immense support has resulted in the widespread availability of PHP libraries that are available on Internet as open source code.

Although there are a variety of libraries available, it is important to meet the standards and guidelines defined for writing Web application codes, which is a time-consuming process. It also takes effort to find out accurate information on the Internet, install dependencies, configure each library, and include them in different files. This is where frameworks are useful.

A framework is a basic templating structure that is used to develop, test, and deploy applications rapidly and efficiently. Using framework helps a developer focus on the actual problem and application logic, and provides tools to perform tasks easily. It is also used for following purposes:

- To provide abstraction
- To handles repetitive and menial tasks
- To arranges things in a logical structure

The Laravel framework reuses and assembles existing components, dependencies, and libraries to automate time-consuming tasks. This helps a Web developer to build Web applications in a structured manner.



## 2.3 Features of Laravel

Laravel includes a consistent API and includes several components to perform common tasks such as database interactions with ease.

### 2.3.1 Convention over Configuration

Laravel has many conventions. If these conventions are properly followed, it can predict the dependencies and automate the configuration related tasks. The Convention over Configuration feature of Laravel reduces the time and effort required to set up and configure services. Such environments are suitable for Rapid Application Development.

One simple example of Convention over Configuration can be seen in the process of using models to retrieve records stored in a table. If a Model is created with the name **Employee**, then it will implicitly try to retrieve records from a table name **employees**, which is the plural snake case name of Employee. If the table name is not the plural snake case name of the Model, then the table name would have to be explicitly defined in the code of the Model.

**DID YOU KNOW?**



Rapid Application Development is a development model. It includes rapid prototyping. Developers use this model to make multiple iterations and updates to software without a development schedule from scratch.

### 2.3.2 MVC Architecture

Since, Laravel is based on the Model–View–Controller (MVC) architecture, the programming logic is divided into the following:

- **Models** - Models define the way data is visualized and stored. Models are objects that represent resources utilized in an application. Usually, they are used to handle records in a database. Hence, models can be used to represent different entities, such as an employee or a user or an item. Models are objects that are instances of the Eloquent's base Model class. This object can be efficiently used to hold records from a database table and perform operations on it. The Eloquent ORM programming techniques allow models to interact more with records in a database.

- **Views** - Views define the way Web pages appear. They are responsible for generating Web pages returned as responses with the help of a controller. They can be written as standard PHP files or can be dynamically built using the Blade Templating language, which gives a modular approach of creating them. Blade allows complex views to be created from simpler layouts.
- **Controllers** - Controllers define the requisite responses for different requests. In simpler terms, controllers take a request and then, depending on the request generate appropriate responses. Similar to a server, controllers are where the actual processing of requests takes place, such as handling form submissions and interacting with the databases.

All of these have powerful classes that define a lot of methods. These methods do most of the work and provide abstraction from the underlying code. This allows a developer to focus more on what logic should be implemented than how it should be implemented. Figure 2.1 illustrates the MVC architecture.

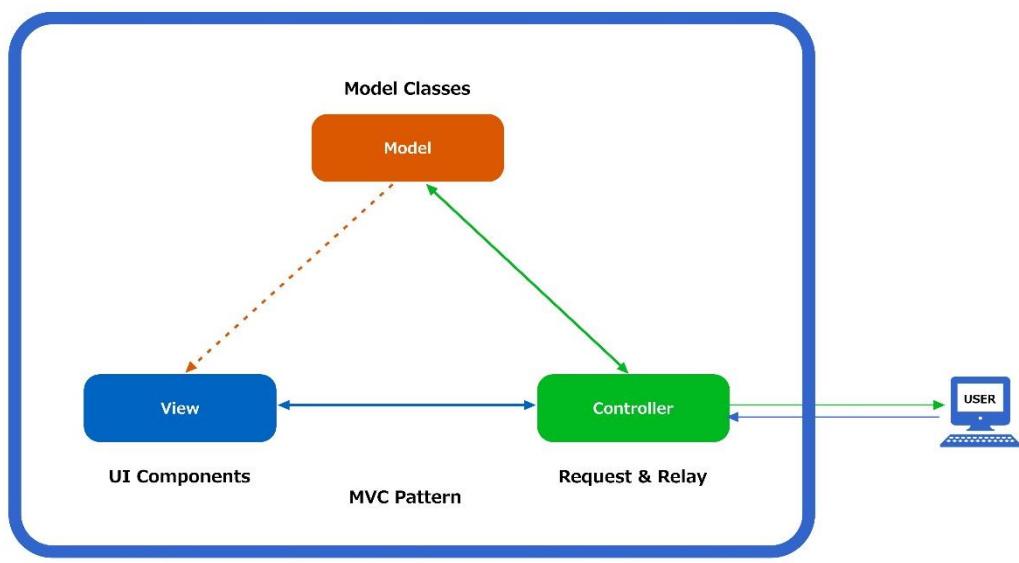


Figure 2.1: MVC Architecture

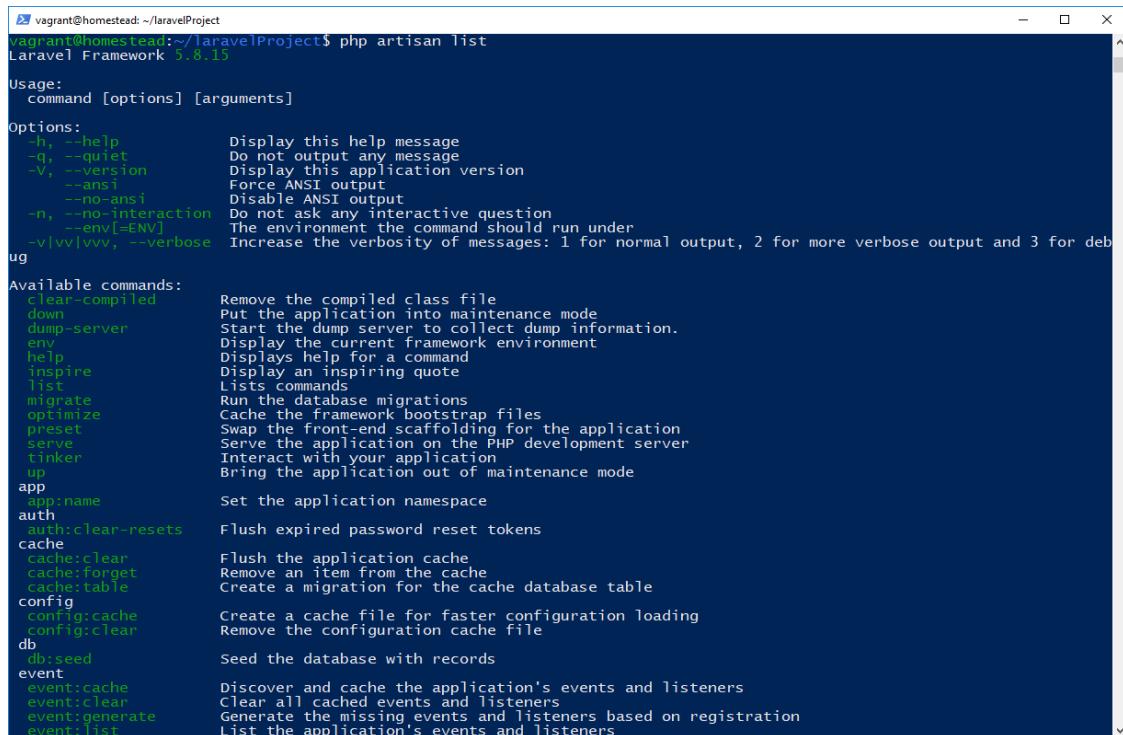
### 2.3.3 Artisan Command for Automation

The artisan tool is included in the Laravel suite. It is an automation tool that helps to set up and configure the various components for operations related to Web application development. It can be used to create models, views, and controllers pre-filled with a template. It can also be used to create dummy server, perform database operations, and many more. Code Snippet 1 shows how to get a list of commands available for artisan.

## Code Snippet 1:

```
php artisan list
```

Figure 2.2 shows the output of the artisan command.



The screenshot shows a terminal window with the following text:

```
vagrant@homestead:~/laravelProject$ php artisan list
Laravel Framework 5.8.15

Usage:
  command [options] [arguments]

Options:
  -h, --help          Display this help message
  -q, --quiet         Do not output any message
  -v, --version       Display this application version
  --ansi             Force ANSI output
  --no-ansi           Disable ANSI output
  -n, --no-interaction
  --env[=ENV]          The environment the command should run under
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for deb
ug

Available commands:
  clear-compiled      Remove the compiled class file
  down                Put the application into maintenance mode
  dump-server         Start the dump server to collect dump information
  env                 Display the current framework environment
  help                Displays help for a command
  inspire             Display an inspiring quote
  list                Lists commands
  migrate             Run the database migrations
  optimize            Cache the framework bootstrap files
  preset              Swap the front-end scaffolding for the application
  serve               Serve the application on the PHP development server
  tinker              Interact with your application
  up                  Bring the application out of maintenance mode

  app
    app:name          Set the application namespace
  auth
    auth:clear-resets Flush expired password reset tokens
  cache
    cache:clear        Flush the application cache
    cache:forget       Remove an item from the cache
    cache:table        Create a migration for the cache database table
  config
    config:cache       Create a cache file for faster configuration loading
    config:clear       Remove the configuration cache file
  db
    db:seed            Seed the database with records
  event
    event:cache        Discover and cache the application's events and listeners
    event:clear         Clear all cached events and listeners
    event:generate     Generate the missing events and listeners based on registration
    event:list          List the application's events and listeners
```

Figure 2.2: List of artisan command

### 2.3.4 Helper Functions

There are many commonly used functions in Web application development that may require a developer to import different libraries. Laravel includes a variety of global PHP functions called Helpers that help in reducing time and efforts of a developer by taking care of commonly used operations.

Laravel itself utilizes these helper functions in its classes, functions, and even in configurations. Laravel classifies these helpers in the following categories:

**Arrays and Objects:** These helpers can be used to perform operations on Arrays and objects. Some operations include sorting values(`Arr::sort()`), getting the value of the first (`head()`) or last element (`last()`), and setting values at a position (`Arr::set()`).

**Paths:** These helpers store the paths of commonly used locations such as the path of the resource directory (`resource_path()`), path of the app directory (`app_path()`), and path of the configuration directory (`config_path()`).

- **Strings:** These helpers can be used to perform operations on strings such as finding if it contains another substring(Str::contains()), and transforming the string into camel case(Str::camel()).
- **URLs:** These helpers can be used to sanitize or generate URLs, such as sanitizing a url into a secure url.
- **Miscellaneous:** The operations that do not fit any categories are present in miscellaneous, such as finding values of environment variables (env()), finding current date (today()), and redirecting the user to another url (redirect()).



To know the complete list of helper functions, visit  
<https://laravel.com/docs/5.8/helpers>.

## 2.3.5 Extensive Library

Laravel is a framework that includes a lot of popular and commonly used libraries. These libraries would have to be manually installed and configured when creating a Web Application without Laravel. Authentication is one such example, which can be easily used to authenticate users while taking care of security issues such as handling attacks that includes SQL Injection, Authentication bypass, and Cross-site Request forgery, encryption, and logging.

It also allows you to include external libraries using the composer package manager as shown in Code Snippet 2.

### Code Snippet 2:

```
Composer require <library-name>
```

The list of available libraries can be found on <https://packagist.org/>.

## 2.4 Exploring the Laravel Directory Structure

To explore the directory structure, first create a Laravel application. This can be done by executing the command as shown in Code Snippet 3 in the Homestead directory through ssh.

## Code Snippet 3:

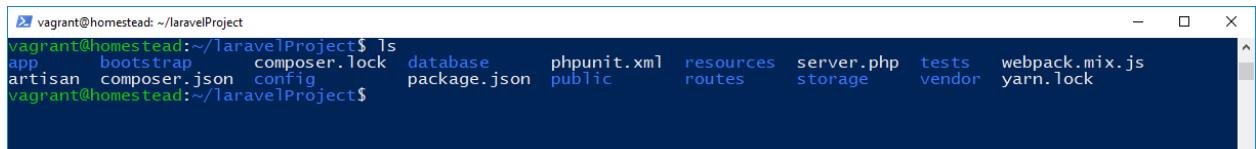
```
laravel new laravelProject
```

This creates a new directory **laravelProject** which will contain all the files and directories required for the Laravel Framework. Code Snippet 4 shows how to view the directory structure.

## Code Snippet 4:

```
cd laravelProject  
ls
```

Figure 2.3 shows the output of the command.



The screenshot shows a terminal window on a Windows operating system. The command `ls` is run in the directory `~/laravelProject`. The output lists several files and directories: `app`, `bootstrap`, `composer.json`, `database`, `phpunit.xml`, `resources`, `server.php`, `tests`, `artisan`, `composer.lock`, `config`, `package.json`, `public`, `storage`, `vendor`, `webpack.mix.js`, and `yarn.lock`. The terminal window has a dark blue background with white text and a standard window title bar.

Figure 2.3: Directory Structure

The directory structure in Laravel is divided into the following logical parts:

- **The app directory:** Contains the core code that runs the Laravel application. Most classes are stored in this directory and its subdirectories.
- **The bootstrap directory:** Includes the code that collates all files spread throughout the directories and run the application.
- **The config directory:** Includes files related to configuration of different components.
- **The database directory:** Contains database migrations, seeds, model factories, and the SQLite database that will be used in subsequent sessions.
- **The public directory:** Includes files that must be exposed to the Internet. Thus, the `index.php` file calls the `app.php` file in the `bootstrap` directory to start the application whenever it receives a request. All HTML, CSS, and Image files are also stored in this directory.
- **The resources directory:** Includes the Views that will be used to display the final Web page as well as the un-compiled assets related to javascript and CSS.
- **The routes directory:** Provides the route definition for the application that will match the request with its response.

- **The storage directory:** Includes data generated by the Laravel framework for storage and cache.
- **The tests directory:** Contains the test specifications for automated testing.
- **The vendor directory:** Contains information on third party dependencies imported by the composer dependency manager.

## Knowledge Check 1:

Q1. Which of the following is not a helper function?

- a. Paths
- b. String
- c. URL's
- d. Vendor Directory

## Knowledge Check 2:

Q2. Which of the following is not a part of Laravel directory structure?

- a. config
- b. route
- c. storage
- d. setting

# Onlinevarsity



**ANYTIME | ANYWHERE**



## SUMMARY

- ▶ Laravel is an open source PHP framework for users working on Web application development.
- ▶ A framework is a basic templating structure that is used to develop, test, and deploy applications rapidly and efficiently.
- ▶ The Laravel framework reuses and assembles existing components, dependencies, and libraries to automate time-consuming tasks.
- ▶ Laravel includes a consistent API.
- ▶ Laravel has many conventions, if followed, can predict the dependencies and automate the configuration related tasks.
- ▶ Laravel is based on the Model–View–Controller (MVC) architecture.
- ▶ Laravel utilizes helper functions in its classes, functions, and even configurations.



## Check Your Progress

Q1. Which of the following defines the way data is visualized and stored?

- |           |               |
|-----------|---------------|
| A. Views  | B. Controller |
| C. Models | D. MVC        |

Q2. The app.php file in the \_\_\_\_\_ directory is called to start the application whenever it receives a request.

- |             |              |
|-------------|--------------|
| A. routes   | B. bootstrap |
| C. resource | D. test      |

Q3. The \_\_\_\_\_ directory contains information on third party dependencies imported by the dependency manager.

- |           |            |
|-----------|------------|
| A. public | B. storage |
| C. app    | D. vendor  |

Q4. Composer is a \_\_\_\_\_.

- |                        |                         |
|------------------------|-------------------------|
| A. PHP dependency tool | B. Programming language |
| C. Library             | D. Package manager      |

Q5. Which one of the following options does Laravel use as a framework?

- |        |        |
|--------|--------|
| A. ORM | B. PHP |
| C. MVC | D. NPM |



# Quiz answers

Question No	Answers
1	C
2	B
3	D
4	A
5	C



# Knowledge Check Answers

Question No	Answers
1	d
2	d



# Session 03: Writing a Simple Application



## Objectives

- Learn to create simple Web application using Laravel
- Explain how to use routes to serve different Web pages
- Understand how views can enhance presentation of an application
- Use the Blade Templating Engine
- Learn to Add Master Template and Nested Views



### 3.1 Introduction

This session describes the steps to create and host Web pages using Laravel and Homestead. This will help in understanding the fundamentals of Laravel by writing the code to serve Web pages that include static content, dynamic content, and input processing.



Views are part of the Model-View-Controller (MVC) design pattern that helps in improving the presentation of the Web page.



### 3.2 Planning the Application

A new Laravel application called simpleApp will be created to:

- Configured and run on the domain name simpleApp.local.
- Serve a static home page (<http://simpleApp.local/>)
- Serve a dynamic test page (<http://simpleApp.local/test>) - that will fetch input from the MVC URI structure, and reflect the value.



## 3.3 Configuring the Environment

By default, all resources required to run a Web application are installed in the Homestead virtual environment. However, minor configuration updates may be required, such as mapping the domain `simpleApp.local` to a subdirectory in the environment. This allows Homestead to serve the correct Web application in case there are multiple applications being served in a single instance.



The `simpleApp.local` application is hosted on the private network, not on the Internet. Therefore, the Windows host file must be modified to enable browsers to locate its IP address.

### 3.3.1 Configuring Homestead

To configure the homestead environment, follow the steps:

**01**

Modify the `Homestead.yaml` file to reflect the changes as shown in Figure 3.1.

```

D:\Work\homestead\Homestead.yaml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Homestead.yaml x
1 ip: 192.168.10.10
2 memory: 1024
3 cpus: 1
4 provider: virtualbox
5 authorize: C:\Users\Ghost\.ssh\id_rsa.pub
6 keys:
7   - C:\Users\Ghost\.ssh\id_rsa
8 folders:
9   -
10    map: 'D:\Work\homestead\code'
11    to: /home/vagrant/code
12 sites:
13   -
14     map: simpleApp.local
15     to: /home/vagrant/code/simpleApp/public
16 databases:
17   - homestead
18 name: homestead
19 hostname: homestead
20

```

Figure 3.1: Homestead.yaml



## Note

Configuration file is in the Homestead directory.



To map the simpleApp.local domain to the /home/vagrant/code/simpleApp/public path in the Homestead environment, follow Code Snippet 1.

## Code Snippet 1:

```
sites:  
-  
  map: simpleApp.local  
  to: /home/vagrant/code/simpleApp/public
```



Save the updates to the file.



Open a terminal in the same folder and type the command as shown in Code Snippet 2.

## Code Snippet 2:

```
vagrant reload --provision
```

Configurations are applied to the file.

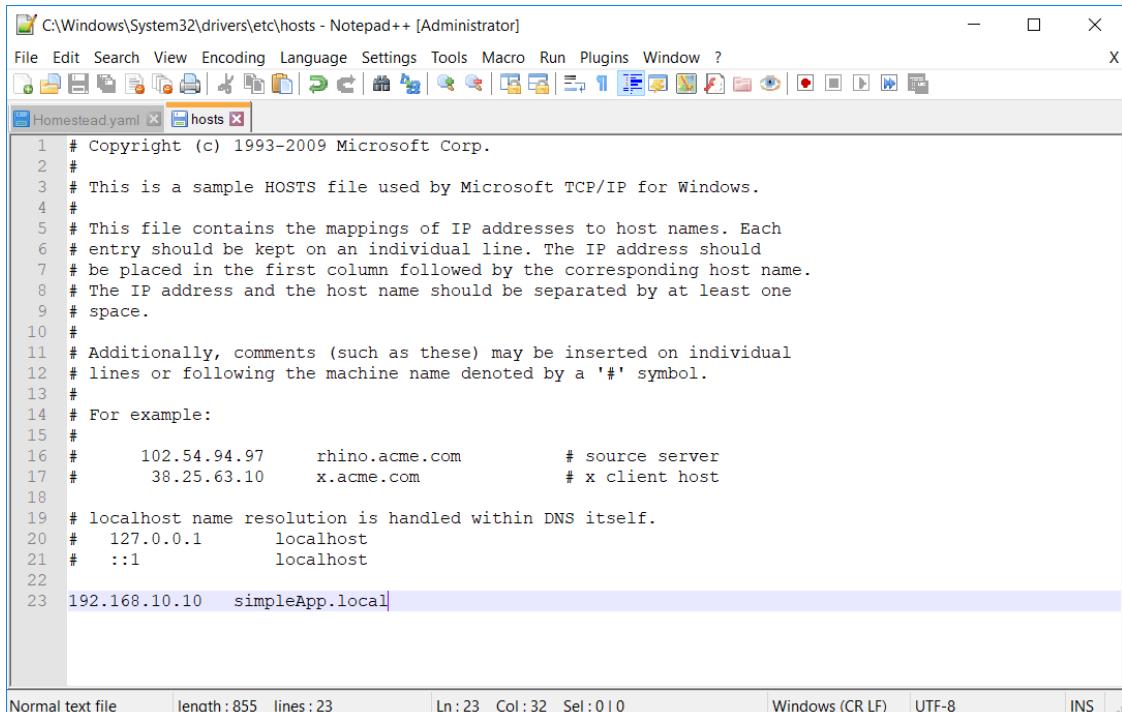
### 3.3.2 Configuring the Hosts File

The hosts file in Windows is the first file that is referenced when the operating system tries to resolve a domain name. Here, domain names can be mapped to the respective IP addresses.

To save changes to the hosts file, open a text editor with administrator privileges. To do so, follow the steps:

1. Right-click the editor in the **Start** menu.

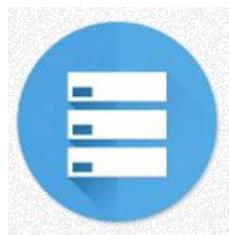
2. Click Run as Administrator. Figure 3.2 displays the sample hosts file that appears.



The screenshot shows a Notepad++ window with the title bar "C:\Windows\System32\drivers\etc\hosts - Notepad++ [Administrator]". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar has various icons for file operations. Below the menu is a tab bar with "Homestead.yaml" and "hosts". The main text area contains the standard Windows hosts file content, with line 23, "192.168.10.10 simpleApp.local", highlighted in blue. The status bar at the bottom shows "Normal text file", "length : 855 lines : 23", "Ln : 23 Col : 32 Sel : 0 | 0", "Windows (CR LF)", "UTF-8", and "INS".

```
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #      102.54.94.97    rhino.acme.com        # source server
17 #              38.25.63.10    x.acme.com          # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 #    127.0.0.1          localhost
21 #    ::1                localhost
22 #
23 192.168.10.10    simpleApp.local
```

Figure 3.2: Sample hosts File



## 3.4 Creating the Laravel Application

Follow the steps to create a Laravel application.

1

To SSH into the Homestead environment, type `vagrant ssh`.



All vagrant related code should be executed in the Homestead directory.

The interface to the Linux command line on the Homestead box appears.

2

To create a new Laravel project, use the following syntax:



syntax  
laravel new [name]

A new folder called name is created. Although the name of the folder is not significant, the Laravel application should be inside the folder with path /home/vagrant/code/simpleApp/.

To create the application at the required path, follow Code Snippet 3.

### Code Snippet 3:

```
cd code  
laravel new simpleApp
```

Figure 3.3 displays the new Laravel project.

The screenshot shows a terminal window on a Linux system (Ubuntu 18.04.1 LTS). The user has run the command `vagrant ssh` to connect to a Homestead virtual machine. In the terminal, the user runs `laravel new simpleApp`. The output shows the creation of a new Laravel project named `simpleApp`. It includes a welcome message about Homestead 8.0.0 and Settler 7.0.0, information about Canonical Livepatch, and a note that 0 packages can be updated. The user then changes directory to `simpleApp` and runs `php artisan serve`.

```
vagrant@homestead: ~/code  
homestead: vagrant ssh  
Vagrant@127.0.0.1's password:  
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-45-generic x86_64)  
  
 * Homestead 8.0.0 released! PHP 7.3 is now the default!  
 * Settler v7.0.0 released! Make sure you update  
 * Need PHP 5.6 or 7.0? Homestead 7.x Settler 6.4.0  
  
 * Canonical Livepatch is available for installation.  
 - Reduce system reboots and improve kernel security. Activate at:  
   https://ubuntu.com/livepatch  
  
0 packages can be updated.  
0 updates are security updates.  
  
Last login: Mon Mar 11 12:12:40 2019 from 10.0.2.2  
vagrant@homestead: $ cd code  
vagrant@homestead:~/code$ laravel new simpleApp
```

Figure 3.3: New Laravel Project

The public subfolder inside a Laravel application must include the necessary files that are exposed to the Internet in order to run the application. Therefore, when the application is created in the mentioned path, the domain `simpleApp.local` will be mapped to the public folder. The files outside this folder cannot be accessed using the Internet.

3

To name the application, use the command as shown in Code Snippet 4.

## Code Snippet 4:

```
cd simpleApp  
php artisan app:name simpleApp
```

Figure 3.4 displays the application renamed as simpleApp, which will be used for the PHP namespace.

```
vagrant@homestead:~/code/simpleApp$ cd simpleApp/  
vagrant@homestead:~/code/simpleApp$ php artisan app:name simpleApp  
Application namespace set!  
Compiled views cleared! Compiled views cleared!  
Application cache cleared!  
Route cache cleared!  
Configuration cache cleared!  
Compiled services and packages files removed!  
Caches cleared successfully!  
vagrant@homestead:~/code/simpleApp$
```

Figure 3.4: New App Name

DID YOU  
KNOW?



The artisan tool is shipped with the Laravel framework. It enables the user to add elements to the project that follow the Laravel convention.

4

To view the running Web application, open the browser and navigate to <http://simpleApp.local>. Figure 3.5 shows the default welcome page.

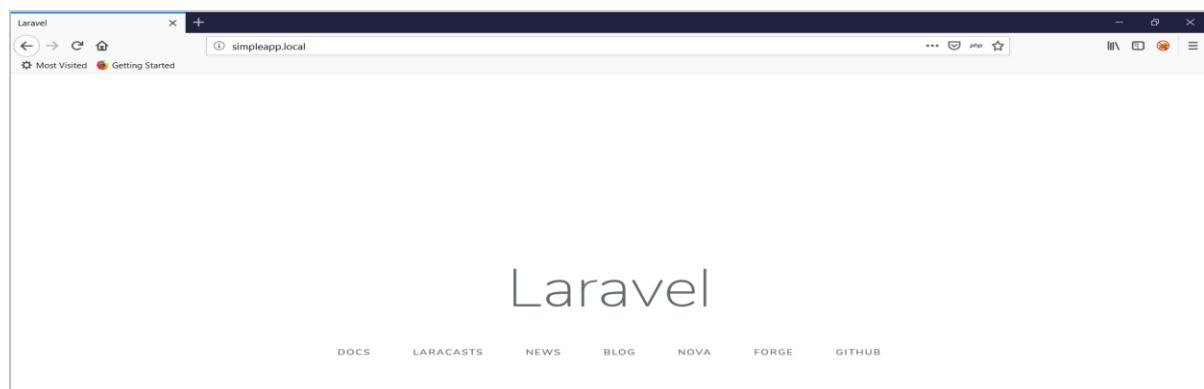


Figure 3.5: Default Welcome Page

Figure 3.5 shows that it is important to follow conventions properly to run a full-fledged application in a short time.



## 3.5 Writing a Route

Laravel leverages the MVC design pattern to provide an interactive interface to the user.

In this architectural pattern, a URI is more a logical construct type rather than an actual location of Web pages. Using Laravel, logical schemes can be mapped to actual resources easily using routes.

Routing is an essential feature in Laravel that enables all application requests to direct towards its respective controller. Web based routes are located at simpleApp\routes\Web.php .

Each Laravel application includes the following pre-configured route as shown in Code Snippet 5.

### Code Snippet 5:

```
Route::get('/',function () {  
    return view('welcome');  
});
```



The code folder inside the Homestead directory and the Homestead environment are the same.

By default, Laravel has a Route class, which has more than one method to handle each type of HTTP request, such as get and put.

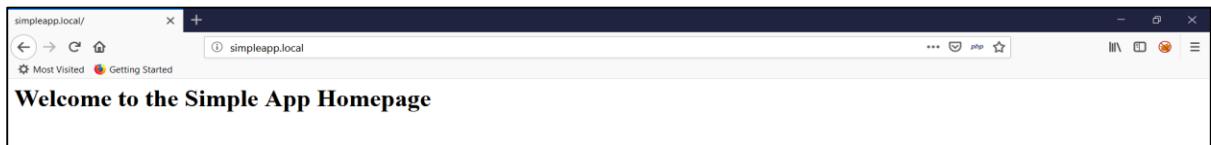
The get method uses the URI, two arguments, and the closure function that is executed in response to the URI.

Modify the following in the Web.php file as shown in Code Snippet 6, and revisit the <http://simpleApp.local> domain.

## Code Snippet 6:

```
Route::get('/', function () {
    return "<h1>Welcome to the Simple App Homepage</h1>";
});
```

Figure 3.6 displays the output of the command as a custom homepage.



**Figure 3.6: Custom Home Page**

The closure function returns the code/tag in the get method, which is processed as a PHP file. The output is sent to the Web client as a response. As a result, the application Home page appears. It is possible to write a variety of complex codes inside the closure function so that a well-designed Web page is returned as a response. However, Laravel recommends the use of Views when writing codes.



**TIP!** Follow proper Laravel conventions to create and run applications using minimal steps. In the example, a Webserver runs on the domain simpleApp.local, along with a static homepage being served on it without writing a PHP file.

## Knowledge Check 1:

Q1. Which one of the following options is an essential feature in Laravel that enables all application requests to direct towards its respective controller?

- a. Views
- b. Controller
- c. Routing
- d. Model



## 3.6 Creating a Dynamic Test Page

The MVC design pattern provides the editing feature that eliminates the necessity to create a new files and multiple pages can be added to the same application. This approach makes MVC design better than the traditional approach.

To use the editing feature, follow these steps.

1. Add another route in the Web.php file inside the routes directory as shown in Code Snippet 7.

## Code Snippet 7:

```
Route::get('test', function () {
    $content = "To access a specific test, visit
http://simpleApp.local/test#id, replace #id with the test id. For
example http://simpleApp.local/test1"
    return $content;
})
```

2. Save the file.
3. Browse to `http://simpleApp.local/test` in the browser. There are now two Web pages that have been created by editing a few configuration files.
4. Add the route to serve multiple pages on the server as shown in Code Snippet 8.

## Code Snippet 8:

```
Route::get('test{id}', function ($id) {
    return "You are trying to access Test #$id";
});
```

5. Save the file.
6. Open `/test198`, `/test202`, and any other combination for `http://simpleApp.local/test101`. Remember that the server serves Web pages for each request.

Usually, developers prefer using the `test/{id}` structure instead of `test{id}`; which is recommended for the production environment.



**Warning**

*Do not reflect user input back the way it is sent. This leads to XSS attacks.*

## 3.7 Using Views

Modularity is an important aspect of the Laravel framework, which means the codes can be independent or interchanged, based on the requirement. Views provide the modularity required for Web application development. Views are located at resources\views; they allow Web application codes for different pages to have their own file.

Views contain both, HTML and PHP, codes for Web pages, and help in separating the controller/application logic from the presentation logic. Thus, it is possible to use the Views that are stored in the simpleApp\resources\views\ directory to present the PHP and HTML code, so that the Web application uses the route file (routes\Web.php) to define the routes.

### 3.7.1 Creating a View for the Home Page

In order to understand more about Views, here is an example that improves the ‘Simple Web Application’ by creating different Views for all the pages - Home, About, and Test. In our example, all the routes are commented out or deleted in the routes/Web.php file. Everything will be started afresh to avoid any conflict in configuration.

To create a View for the Home page, follow the steps.

To create a View for the Home page, follow the steps.

1. Open notepad++.
2. Create a new file using Code Snippet 9.

#### Code Snippet 9:

```
<html>
  <body>
    <h1>Hello. Welcome to your own Simple Web Application</h1>
  </body>
</html>
```

3. Save the file as home.php in the following location - resource\views\.
4. Replace the get method for URI \ in the Web routes Web.php file as shown in Code Snippet 10.

## Code Snippet 10:

```
Route::get('/', function () {  
    return view('home');  
});
```



The View function provides the .php extension. It automatically uses home.php from the resources\views\ folder. This is an example of using convention over configuration.

## Knowledge Check 2:

Q2. Which one of the following options helps in separating the controller/application logic from the presentation logic?

- a. HTML
- b. Home
- c. Views
- d. Routes

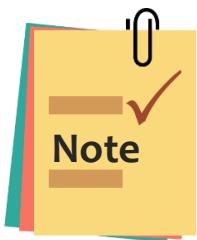


## 3.8 Using the Blade Templating Engine

Blade is another powerful feature of Laravel. It is a lightweight template language that provides multiple short codes. Therefore, the Blade engine makes writing PHP applications easy.

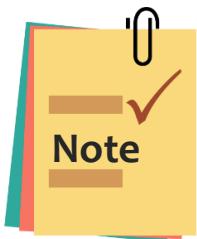
The Blade template engine reduces the number of keystrokes and increases the readability of templates.

For example, developers perform a variety of processing tasks. They save the complete output in a single variable. The Blade template engine has a shortcode {{ \$output }}, which expands into the php code <?php echo htmlentities(\$output); ?>. This eliminates the use of manually entering certain parts of the code, and therefore, saves time and effort.



Blade syntax is PHP equivalent for example, `{ !! $output !! }`

Blade also supports most PHP constructs to create loops and conditions, such as @for, @foreach, @while, @if, and @elseif; thereby removing the dependency of opening and closing <?php tags at each instance in the template.



For more information about Blade, you can refer official document for Blade Template at <https://laravel.com/docs/5.8/blade>.

### Knowledge Check 3:

Q3. Which one of the following options is not a PHP construct?

- a. @while
- b. @output
- c. @for
- d. @if

## 3.8.1 Adding Master Template and Nested Views

Since, Blade is a templating engine, it allows templates to be nested and extended similar to most other templating software's used in industry. Many templating engines have an in-built master layout, but there is no master layout in Blade.

However, it is possible to add this functionality by creating a `master.blade.php` file in the `views` directory. Use of `extend` blade directive in other templates makes it possible, so that it can be used as the master layout.

## SUMMARY

- ▶ By default, all resources required to run a Web application are installed in the Homestead virtual environment.
- ▶ Laravel leverages the MVC design pattern to provide an interactive interface to the user.
- ▶ The hosts file in Windows is the first file that is referenced when the operating system tries to resolve a domain name.
- ▶ Routes serve different Web pages and Views enhance their presentation.
- ▶ Modularity is an important aspect of the Laravel framework, which means the codes can be independent or interchanged, based on the requirement.
- ▶ Blade is a lightweight template language that provides multiple short codes.
- ▶ Master Template and Nested Views can be configured in the master.blade.php file.
- ▶ Laravel follows the ‘Convention is better than Configuration’ method.



## Check Your Progress

Q1. \_\_\_\_\_ tool enables the user to add elements to the project that follow the Laravel convention.

- |              |            |
|--------------|------------|
| A. Homestead | B. Laravel |
| C. Artisan   | D. MVC     |

Q2. The closure function returns the code/tag in the \_\_\_\_\_ method.

- |         |           |
|---------|-----------|
| A. post | B. get    |
| C. put  | D. return |

Q3. It is possible to create a master.blade.php file in the \_\_\_\_\_ directory.

- |           |           |
|-----------|-----------|
| A. Master | B. routes |
| C. app    | D. views  |

Q4. Which of the following files is required to modify and configure the Homestead environment?

- |                        |                     |
|------------------------|---------------------|
| A. homestead.yaml file | B. master.blade.php |
| C. home.php            | D. Web.php          |

Q5. \_\_\_\_\_ is more of a logical construct type.

- |                                |                |
|--------------------------------|----------------|
| A. Uniform Resource Locator    | B. Routes      |
| C. Uniform Resource Identifier | D. Controllers |



# Quiz answers

Question No	Answers
1	C
2	B
3	D
4	A
5	C



# Knowledge Check Answers

Question No	Answers
1	c
2	a
3	b

# Onlinevarsity

## GO DIGITAL





# Session 04: Creating a Web Application



## Objectives

- Learn to create a new Laravel application
- Understand integration of Laravel application with database
- Learn to create table schema
- Explain process migration
- Understand Web page creation using Master Blade Layout, Forms, and Routes



## 4.1 Introduction

This session focuses on the steps to create a Web application that will be connected to a working database. A query builder class will be used to create, read, update, and delete records in a database.



## 4.2 Planning the Application

An application will be developed for a human resource project. The project will help to manage employees and the department they work for.



Create, Read, Update, and Delete operations are referred to as CRUD operations.

Following is detailed description of the project:

- Name of the application - hrm.

- SQLite 3 will be used as the Database Management System (DBMS) for this application.
- It will include:
  - Home page with links to other pages such as Add, Display, Update, and Delete. These pages will implement CRUD operations respectively.
  - The application will be accessible through the hrm.local domain.
- There will be two database tables employees and departments.
  - employees table – This table will include columns for ID, Name, Department, and Salary.
  - departments table – This table will have Department ID, Department Name, Manager Name, and Manager ID.



## 4.3 Building a Laravel Application

Follow the steps to build a Laravel application.

1. Edit the Homestead.yaml file to map hrm.local to the Homestead/code/hrm/public folder. Figure 4.1 shows the Homestead.yaml file.

```
ip: 192.168.10.10
memory: 2048
cpus: 1
provider: virtualbox
authorize: C:\Users\Ghost\.ssh\id_rsa.pub
keys:
  - C:\Users\Ghost\.ssh\id_rsa
folders:
  -
    map: 'D:\Work\Homestead\code'
    to: /home/vagrant/code
sites:
  -
    map: hrm.local
    to: /home/vagrant/code/hrm/public
    databases:
      - homestead
    name: homestead
    hostname: homestead
```

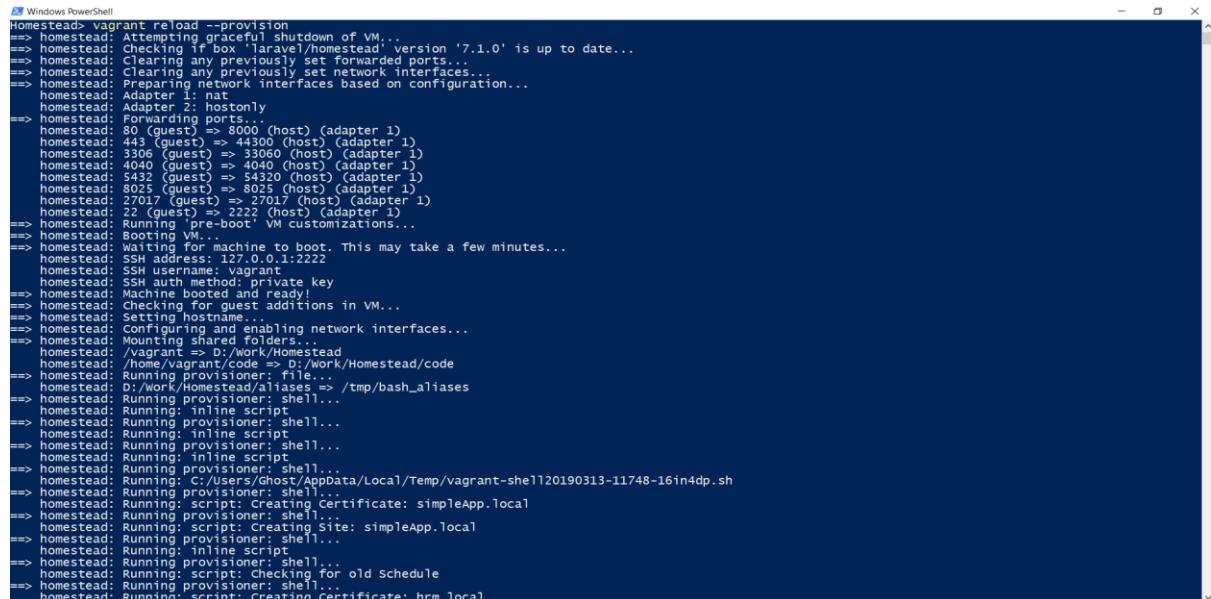
Figure 4.1: Homestead.yaml File

2. Save the changes.
3. To reload provision in homestead, run vagrant reload --provision command as shown in Code Snippet 1 in a terminal inside the Homestead directory.

## Code Snippet 1:

```
vagrant reload --provision
```

Figure 4.2 shows the output of Code Snippet 1.



```
Windows PowerShell
Homestead: vagrant reload --provision
--> homestead: Attempting graceful shutdown of VM...
--> homestead: Checking if box 'laravel/homestead' version '7.1.0' is up to date...
--> homestead: Clearing any previously set forwarded ports...
--> homestead: Clearing any previously set network interfaces...
--> homestead: Preparing network interfaces based on configuration...
--> homestead: Adapter 1: Intel PRO/100 MT Desktop
--> homestead: Adapter 2: bostonly
--> homestead: Forwarding ports...
homestead: 80 (guest) => 8000 (host) (adapter 1)
homestead: 443 (guest) => 44300 (host) (adapter 1)
homestead: 3006 (guest) => 30050 (host) (adapter 1)
homestead: 4000 (guest) => 4000 (host) (adapter 1)
homestead: 5432 (guest) => 54320 (host) (adapter 1)
homestead: 8025 (guest) => 8025 (host) (adapter 1)
homestead: 27017 (guest) => 27017 (host) (adapter 1)
homestead: 22 (guest) => 2222 (host) (adapter 1)
--> homestead: Running pre-boot VM customizations...
--> homestead: Booting VM...
--> homestead: Waiting for machine to boot. This may take a few minutes...
homestead: SSH address: 127.0.0.1:2222
homestead: SSH username: vagrant
homestead: SSH auth method: private key
--> homestead: Machine booted and ready!
--> homestead: Checking for guest additions in VM...
--> homestead: Setting hostname...
--> homestead: Configuring and enabling network interfaces...
--> homestead: Mounting shared folders...
homestead: /vagrant => D:/work/Homestead
homestead: /home/vagrant => D:/work/Homestead/code
--> homestead: Running provisioner: shell...
homestead: D:/work/Homestead/aliases => /tmp/bash_aliases
--> homestead: Running provisioner: shell...
homestead: Running: inline script
--> homestead: Running provisioner: shell...
homestead: Running: inline script
--> homestead: Running provisioner: shell...
homestead: Running: inline script
--> homestead: Running provisioner: shell...
homestead: Running: inline script
homestead: Running: c:/Users/Ghost/AppData/Local/Temp/vagrant-shell20190313-11748-16in4dp.sh
--> homestead: Running provisioner: shell...
--> homestead: Running provisioner: shell...
--> homestead: Running provisioner: shell...
homestead: Running provisioner: shell...
homestead: Running: certificate: simpleApp.local
--> homestead: Running provisioner: shell...
homestead: Running: script: Creating Site: simpleApp.local
--> homestead: Running provisioner: shell...
homestead: Running: inline script
--> homestead: Running provisioner: shell...
homestead: Running: certificate: hrm.local
--> homestead: Running provisioner: shell...
homestead: Running: script: Creating Certificate: hrm.local
```

Figure 4.2: Reloading Provision in Homestead

4. To add the mapping of the IP address, add an entry for hrm.local in the hosts file corresponding to the IP 192.168.10.10 as shown in Figure 4.3.

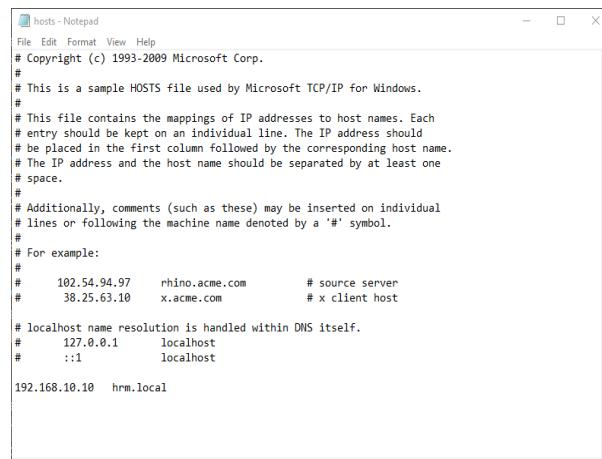


Figure 4.3: IP Address Mapping

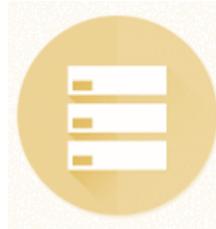
5. To ssh into the vagrant machine, use the command `vagrant ssh`.
6. To create a new Laravel application in the `hrm` folder, navigate to the code directory as shown in Code Snippet 2.

## Code Snippet 2:

```
cd code  
laravel new hrm  
cd hrm  
php artisan app:name HRM
```

The `php artisan app:name HRM` names the application as HRM.

7. To view the default Laravel page, browse to `http://hrm.local`.



## 4.4 Preparing the Database

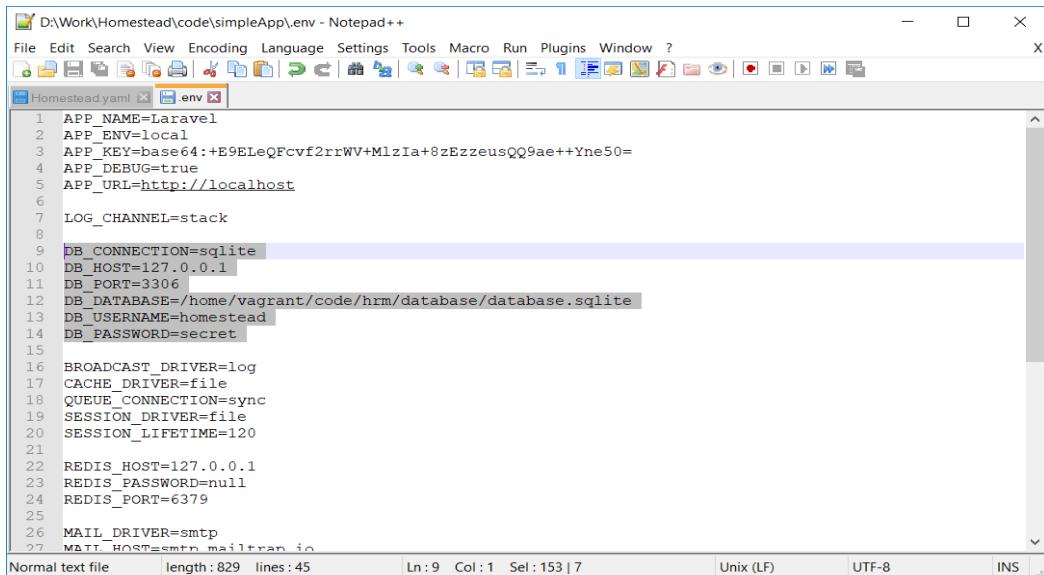
SQLite is a fast and reliable database engine, which will be used as the DBMS software for the `hrm` Web application. Ensure that a blank file is created before configuring the newly created Laravel application as SQLite dumps the entire database in a file. Create a blank file, as shown in Code Snippet 3.

## Code Snippet 3:

```
touch database/database.sqlite
```

### 4.4.1 Configuring the Application

After creating the dummy file, open the `.env` file in the `hrm` directory in notepad++ and edit as highlighted in Figure 4.4.



The screenshot shows a Notepad++ window with the file 'D:\Work\Homestead\code\simpleApp\.env'. The content of the file is as follows:

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:+E9ElcQFcvf2rrWV+M1zIa+8zEzzeusQQ9ae++Yne50=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysqli
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=/home/vagrant/code/hrm/database/database.sqlite
DB_USERNAME=homestead
DB_PASSWORD=secret

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
```

**Figure 4.4: .env File**

The application is now configured to use SQLite as its DBMS software.

## 4.4.2 Creating the Table Schema

Schema is the structure of the table that will be created for the application. The Schema builder class in Laravel helps in creating tables. In addition DB query builder class helps in implementing CRUD operations. They also perform other operations without any SQL code written specifically for the purpose.

To create the employees table using the Schema builder class, follow Code Snippet 4.

### Code Snippet 4:

```
Schema::create('employees', function (Blueprint $table) {
    $table->bigIncrements('id');
    $table->string('name');
    $table->string('designation');
    $table->integer('salary');
});
```

The Schema class instance takes in two arguments - table name and a closure function containing the structure of the table.

The \$table instance defines the structure of the table. In the example, the employees table will have six columns such as ID, Name, Designation, Salary, Created at, and Updated at. The last two columns are created automatically by Laravel. The Created at column will contain the timestamp of the time and date at which the database was created. The Updated at column will always contain the timestamp of the time and date when the table was last modified.

DID YOU  
KNOW?



Laravel by default constraints the autoIncremental column as the primary key.

This code will create the table when it is executed from an executable location. A new route can also be written to create the table. However, by **convention**, routes are the preferred way to create records, not tables. Tables must be created using the Migration method.

### 4.4.3 Creating Migrations

Laravel includes the Migrations feature, which also acts as a version control system for a database that includes many tables. Migrations construct, reconstruct, and deconstruct tables and their schemas. They are primarily used when shifting one DBMS system to another or for creating a backup of systems. Migration can be implemented by editing the .env file and running the migrations command.

Before creating a new migration, delete the migrations that have been shipped with Laravel as shown in Code Snippet 5.

#### Code Snippet 5:

```
rm database/migrations/*
```

To create a new migration, execute the code as shown in Code Snippet 6.

#### Code Snippet 6:

```
php artisan make:migration create_employees_table --  
create=employees
```

Code Snippet 6 creates a new migration [timestamp]\_create\_employees\_table at the database/migrations/ location. The migration includes a pre-written code for creating the employees table using the --create=employees parameter. Figure 4.5 displays the migration file.

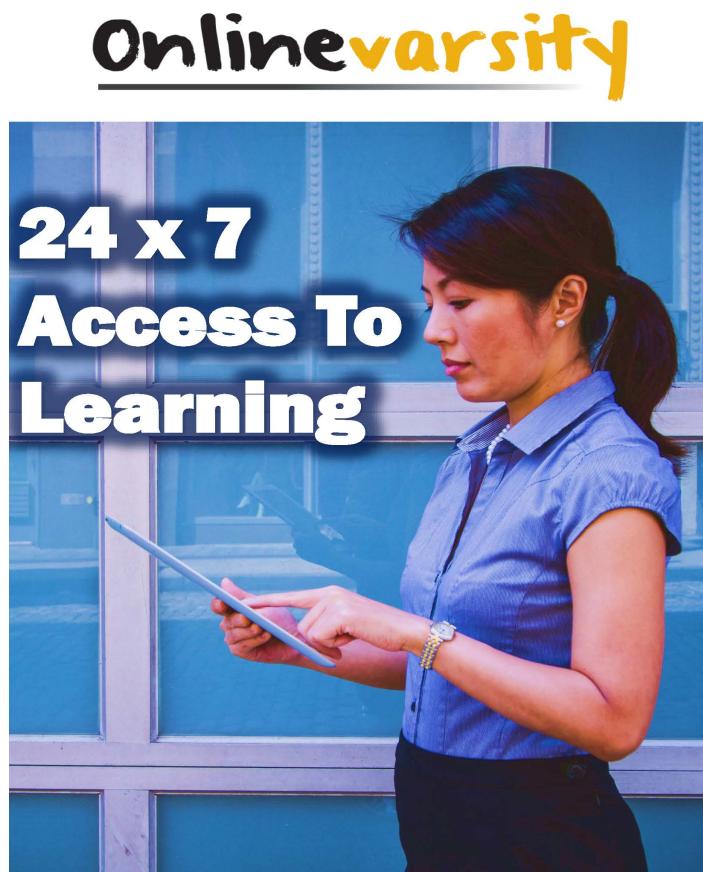
```
1 <?php
2
3 use Illuminate\Support\Facades\Schema;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Database\Migrations\Migration;
6
7 class CreateEmployeesTable extends Migration
8 {
9     /**
10      * Run the migrations.
11     *
12     * @return void
13     */
14    public function up()
15    {
16        Schema::create('employees', function (Blueprint $table) {
17            $table->bigIncrements('id');
18            $table->timestamps();
19        });
20    }
21
22    /**
23     * Reverse the migrations.
24     *
25     * @return void
26     */
27    public function down()
28    {
29        Schema::dropIfExists('employees');
30    }
31
32 }
```

**Figure 4.5: Migration File**

The migration file includes the following two methods.

- `up()` - This method is used to create a table.
- `down()` - This method is used to drop the table.

The Schema builder code used in section 4.4.2 can be pasted in the up section to create the table. The final code appears as shown in Code Snippet 7.



## Code Snippet 7:

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateEmployeesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('employees', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('name');
            $table->string('designation');
            $table->integer('salary');
        });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('employees');
    }
}
```

To create the table, execute code from Code Snippet 7 and run command `php artisan migrate` as shown in Code Snippet 8.

## Code Snippet 8:

```
php artisan migrate
```

Figure 4.6 shows successful migration.

```
vagrant@homestead:~/code/hrm$ php artisan migrate
Migrating: 2019_03_12_214546_create_employees_table
Migrated: 2019_03_12_214546_create_employees_table
vagrant@homestead:~/code/hrm$
```

**Figure 4.6: Successful Migration**

To verify that the table is created, enter the command as shown in Figure 4.7.

```
vagrant@homestead:~/code/hrm$ php artisan migrate
Migrating: 2019_03_12_214546_create_employees_table
Migrated: 2019_03_12_214546_create_employees_table
vagrant@homestead:~/code/hrm$ sqlite3 database/database.sqlite
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .tables
employees  migrations
sqlite>
```

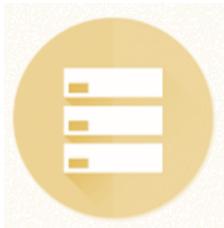
**Figure 4.7: Verifying Migration**

Figure 4.7 displays that the employees table is created and ready for use. Next, repeat the procedure for table creation to create the departments table.

## Knowledge Check 1:

Q1. Which one of the following options is the correct code to create a table?

- a. `php artisan route`
- b. `php migrate artisan`
- c. `php table migrate`
- d. `php artisan migrate`



## 4.5 Creating Web Pages

In the MVC architecture, Views take care of the presentation of the Web Application. The Blade templating engine included in Laravel allows you to write Views more efficiently. It does so by providing a modular approach for writing Web pages through templates/layouts.

Most Web Pages share a lot of components such as the Web Application logo, name, menu bar, navigation bar, header, footer, and many more. These basic contents of a Web application can be written in a single template called master template. This template can define sections that will be reused throughout the application and can embed contents from other templates. This page specific content is saved inside other templates.

Using Blade directives, individual views can include the master template and inject page specific content.

## 4.5.1 Writing the Master Blade Layout

Follow the steps to write the Master Blade layout.

1. To create a file, use the code as shown in Code Snippet 9.

### Code Snippet 9:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>Human Resource Management</title>
        <link rel="stylesheet" href="add thing here">
    </head>
    <body>
        <div class="container">
            <div class="page-header">
                @yield('header')
            </div>
            @yield('content')
        </div>
    </body>
</html>
```

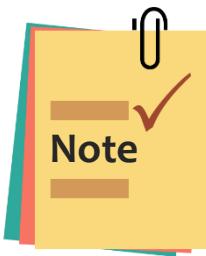
2. Save the file at the location - resources/views/master.blade.php.

The Bootstrap CSS framework has a better UI/UX, which uses the asset() helper function. It helps to avoid writing of absolute paths and creates a working relative path, irrespective of the location.

Currently, there are no bootstrap files in the Laravel app.

3. Download the files from <http://getbootstrap.com>.
4. Place the minified CSS file at public/css/.

Although the Master Blade template uses the yield directive, it is replaced by the content in the sections directive of child templates.



The yield directive is used inside master templates.

## Knowledge Check 2:

Q2. The base structure of Web application pages is saved in the \_\_\_\_\_.

- a. CSS folder
- b. Master template
- c. Home directory
- d. Child template

### 4.5.2 Adding Forms

The Home page, in a Web application, is a simple static page with links to other pages. However, forms are necessary to link other pages as well as receive input from users.

Forms can be created using the form builder class, which is pre-installed in Laravel versions up to 4.x. Higher versions of Laravel are shipped with Advanced Blade directives.

The Form builder class is easier to use than Blade. This has led to the creation of a repository called `LaravelCollective`, by the open source community in order to help developers use the Form builder class.

Follow the steps to add the repository.

1. To install the package, enter the command as shown in Code Snippet 10.

#### Code Snippet 10:

```
composer require laravelcollective/html
```

2. To add a new provider, add `HTMLServiceProvider` to the list of providers in the `config/app.php` file as shown in Code Snippet 11.

## Code Snippet 11:

```
'providers' => [
    ...
    Collective\Html\HtmlServiceProvider::class,
    ...
],
```

3. To use classes anywhere, create an alias for Form in the config/app.php file as shown in Code Snippet 12.

## Code Snippet 12:

```
'aliases' => [
    ...
    'Form' => Collective\Html\FormFacade::class,
    ...
],
```

4. Create a blank form using the Form class. The Form::open() method writes a form. Form::close() denotes the end of the form.

## Knowledge Check 3:

Q3. The \_\_\_\_\_ method writes a form.

- a. close()
- b. get()
- c. open()
- d. fetch()

## 4.5.3 Creating Views with Forms

Assume that the Web application requires the following pages to be served:

- Home page
- Add Employee Data (/add)
- Displaying Employee Data (/display)
- Modifying Employee Data (/edit)
- Deleting Employee Data (delete)

The Home page view has been created and the views for add, display, edit, and delete are required to be created.

Populate code in **add.php** as shown in Code Snippet 13.

## Code Snippet 13:

```
<!-- Stored in resources/views/add.blade.php -->

@extends('layouts.master')
@section('header')
    <h1>Use the following form to add a record for new
employee</h1>
@endsection
@section('content')
    {{ Form::open(['url'=>'add', 'method'=>'post']) }}
    <pre>
        {{ Form::label('name', 'Name:') }} {{ Form::text('username') }}
        {{ Form::label('designation', 'Designation:') }} {{ Form::text('designation') }}
        {{ Form::label('salary', 'Salary:') }} {{ Form::number('name', 'value') }}
        {{ Form::submit('Submit') }}
    </pre>
    {{ Form::close() }}
@endsection
```

Populate code in **display.php** as shown in Code Snippet 14.

## Code Snippet 14:

```
<!-- Stored in resources/views/display.blade.php -->

@extends('layouts.master')
@section('header')
    <h1>Use the following form to display a record for an employee
with their id</h1>
@endsection
@section('content')
    {{ Form::open(['url'=>'display', 'method'=>'post']) }}
    <pre>
        {{ Form::label('id', 'ID:') }} {{ Form::text('id') }}
        {{ Form::submit('Search') }}
    </pre>
    {{ Form::close() }}
@endsection
```

Populate code in **edit.php** as shown in Code Snippet 15.

## Code Snippet 15:

```
<!-- Stored in resources/views/edit.blade.php -->

@extends('layouts.master')
@section('header')
    <h1>Use the following form to fetch and edit an employee record
with their id</h1>
@endsection
@section('content')
    {{ Form::open(['url'=>'edit', 'method'=>'post']) }} 
    <pre>
        {{ Form::label('id','ID:') }} {{ Form::text('id') }} 
        {{ Form::submit('Submit') }} 
    </pre>
    {{ Form::close() }} 
@endsection
```

Populate code in **delete.php** as shown in Code Snippet 16.

## Code Snippet 16:

```
<!-- Stored in resources/views/delete.blade.php -->

@extends('layouts.master')
@section('header')
    <h1>Use the following form to delete an employee record</h1>
@endsection
@section('content')
    {{ Form::open(['url'=>'delete', 'method'=>'delete']) }} 
    <pre>
        {{ Form::label('id','ID:') }} {{ Form::text('id') }} 
        {{ Form::submit('Search') }} 
    </pre>
    {{ Form::close() }} 
@endsection
```

## 4.5.4 Adding Routes

To verify that the forms are created properly, add a few temporary routes. Edit the `routes\Web.php` file and write the routes as shown in Code Snippet 17.

### Code Snippet 17:

```
<?php
/*
|--------------------------------------------------------------------------
|
| Web Routes
|
|
| Here is where you can register Web routes for your application.
These
| routes are loaded by the RouteServiceProvider within a group
which
| contains the "Web" middleware group. Now create something
great!
|
*/
Route::get('/', function () {
    return view('home');
});

Route::get('add', function () {
    return view('add');
});

Route::get('display', function () {
    return view('display');
});

Route::get('edit', function () {
    return view('edit');
});

Route::get('delete', function () {
    return view('delete');
});
```

## SUMMARY

- ▶ A query builder class is used to Create, Read, Update, and Delete records (CRUD operations).
- ▶ SQLite is a fast and reliable database engine that is used as the DBMS software for Web applications.
- ▶ Schema is a builder class used to create tables.
- ▶ The Migrations feature acts as a version control system for a database that includes many tables.
- ▶ The Bootstrap CSS framework uses the asset() helper function to wrap the other folders and resources together.
- ▶ LaravelCollective repository is created by the open source community to help developers who use Form builder class.

**Onlinevarsity**



WHERE THE EXPERTS SPEAK THE EXPERIENCE



## Check Your Progress

Q1. To verify that forms have been created properly, \_\_\_\_\_.

- |                               |                                 |
|-------------------------------|---------------------------------|
| A. Add the Blade template     | B. Edit the public\css file     |
| C. Modify the update.php file | D. Edit the routes\Web.php file |

Q2. Bootstrap CSS framework uses \_\_\_\_\_ function to wrap the resources.

- |                   |           |
|-------------------|-----------|
| A. closure        | B. public |
| C. asset() helper | D. return |

Q3. Migration is possible by editing the \_\_\_\_\_ file and running the migration command.

- |                   |         |
|-------------------|---------|
| A. homestead.yaml | B. .env |
| C. .php           | D. .css |

Q4. In the CRUD operations scheme, 'R' stands for \_\_\_\_.

- |           |           |
|-----------|-----------|
| A. Review | B. Repeat |
| C. Recall | D. Read   |

Q5. The Master Blade template uses the \_\_\_\_\_ directive.

- |          |             |
|----------|-------------|
| A. yield | B. sections |
| C. asset | D. routes   |



# Quiz answers

Question No	Answers
1	D
2	C
3	B
4	D
5	A



# Knowledge Check Answers

Question No	Answers
1	d
2	b
3	c



## Session 05: The Eloquent ORM



## Objectives

- Describe the functionalities offered by Eloquent Object Relational Mapping (ORM)
- Learn to perform database operations using Eloquent ORM
- Understand database testing using Routes and Eloquent ORM



### 5.1 Introduction

This session explains Eloquent Object Relationship Model (ORM) and the usage of Models in facilitating database operations without writing SQL queries.



### 5.2 What is the Eloquent ORM?

The Eloquent Object Relationship Model acts as the model layer in the MVC design pattern. In Laravel, the Eloquent ORM is useful for creating models that help a user to work with database without writing a single line of SQL code. In addition, while migrating to other databases, it only requires modification in the .env file.



### 5.3 Eloquent Conventions and the Mighty Model

Similar to Laravel, Eloquent too has basic conventions that must be followed in order to implement it using minimal configurations.

Studly-case is the format of text with capital letters. By convention, Eloquent model name is studly-cased version of the table name that it corresponds to. Laravel capitalizes the first letter of the word.

For example, the Eloquent model name for the employees table is Employee. With the name Employee, Eloquent by default performs database operations on a table, which has the snake-case plural version of its name. In a snake-case, the text is formatted with small letters. The snake case for Employee is employee, and its plural is employees.



This table is created in Session 4.

To create an Eloquent model, execute the command as shown in Code Snippet 1.

## Code Snippet 1:

```
php artisan make:model Employee
```

The app/Employee.php file corresponding to the Employee model is created in the app directory and it includes the code as shown in Code Snippet 2.

## Code Snippet 2:

```
<?php  
  
namespace HRM;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Employee extends Model {  
    //  
}
```

If Eloquent conventions are followed properly, the model can perform database operations. For example, to retrieve all records in a database (which is yet to be created), use the web.php routes as shown in Code Snippet 3.

**Code Snippet 3:**

```
<?php

use HRM\Employee;

Route::get('/', function () {
    return Employee::all();
});
```



## 5.4 Inserting, Retrieving, Saving, and Deleting Data

The common actions that can be performed on a database are insert, retrieve, save, and delete. These actions are synonymous to **Create**, **Read**, **Update**, and **Delete**; also known as the **CRUD** operations that can be used with Eloquent ORM.

### 5.4.1 Creating a Record

The Employee model supports the CRUD operations. The create method uses the associative array as an argument in the Employee model. Values in the respective column are inserted using the keys of the associative array.

To create a record in a model, perform the following steps:

1. To create an associative array, execute the command as shown in Code Snippet 4.

**Code Snippet 4:**

```
$data = ['name' => 'Albert', 'salary' => 90000, 'designation' =>
'Senior Engineer'];
```

2. To reference the Employee model, use the use statement and then, create an instance of the Employee model as shown in Code Snippet 5.

## Code Snippet 5:

```
use HRM\Employee;  
$employee = new Employee;
```

In Code Snippet 5, HRM is the application name, which must be modified if the application name is different.

3. To insert the data array, use the create method of the Employee instance as shown in Code Snippet 6.

## Code Snippet 6:

```
$employee->create($data);
```

4. For test purpose, delete the content in the Web routes Web.php file.
5. Create a new route for the / path.
6. Enter the code from Code Snippet 6 inside the closure function. Code Snippet 7 shows the final route file.

## Code Snippet 7:

```
<?php  
use HRM\Employee;  
  
Route::get('/', function () {  
    $data = ['name' => 'Albert', 'salary' => 90000, 'designation'  
=> 'Senior Engineer'];  
    $employee = new Employee;  
    $employee->create($data);  
});
```

7. To view the result, browse to the Web application at <http://hrm.local/>. A MassAssignmentException occurs.



A Mass Assignment is a case where the attributes of a Model are blindly updated. For example, if the \$data array comes when a user submits a form then, the user can update any value in the same database. This method is detrimental because a hacker can gain access and update the fields.

To prevent any such malicious attempt, it is necessary to include the \$fillable keyword array. Doing so informs the application about the fields that can be updated by a user.

8. To modify the model, open the Employee model in app/Employee.php and enter the code as shown in Code Snippet 8.

### Code Snippet 8:

```
<?php  
  
namespace HRM;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Employee extends Model  
{  
    protected $fillable = ['name', 'salary', 'designation'];  
}
```

The \$fillable array signifies that only the fields, name, salary, and designation can be updated manually.

9. To view the result, browse to the Web application. A blank page appears because there is no return statement inside the closure function.

### Knowledge Check 1:

Q1. The \_\_\_ array signifies that specific fields can be updated.

- a. \$update
- b. \$retrieve
- c. \$fillable
- d. \$enter

## 5.4.2 Retrieving Records

Records can be retrieved by referencing the method via the class name.

To know if the new record is successfully added, perform the following steps:

1. To reference to the method, replace the routes file content with the code as shown in Code Snippet 9.

### Code Snippet 9:

```
<?php
use HRM\local;
Route::get('/', function() {
    return Employee::all();
});
```

2. Save the file.
3. To view the page, browse to the application as shown in Figure 5.1.

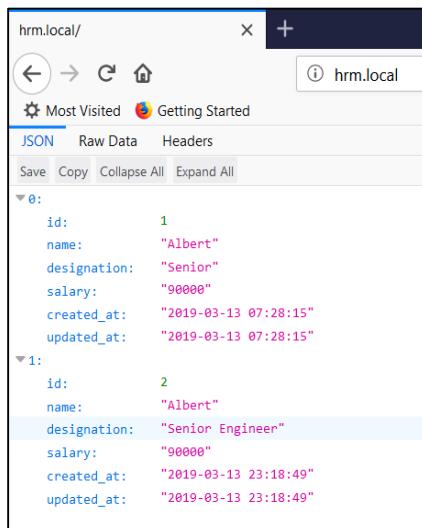


Figure 5.1: Viewing Records

### all() Method

Another method that Laravel uses is the all() method to retrieve records. When the Employee::all() method is used, it returns a collection object and Laravel returns as a json object. The output in each instance will be different.

The all() method returns all records in the database table. To view a specific record, use the find() method; the records are sorted according to the filtered criteria. Replace the route file with the code as shown in Code Snippet 10 and browse to the Web application.

## Code Snippet 10:

```
<?php  
use HRM\local;  
Route::get('/', function() {  
    return Employee::find(2);  
});
```

The record with primary key value two is displayed. Remember that the primary key is also the column id because the bigIncrements method was used to define the id column during database migration.

Eloquent is bundled with a feature-rich query builder that has methods for common SQLite directives, such as, LIMIT, ORDER, and WHERE. The main reason to use the query builder over writing SQL code manually is that it writes different SQL code depending on the database being used. Consider migrating database from sqlite to MySql, then only the .env file must be changed and rest of the modification is taken care of automatically. This is another good example of using conventions over configurations.

## WHERE Method

The WHERE clause as shown in Code Snippet 11 displays similar results as the find()method. WHERE also provides the option to select the column name.

## Code Snippet 11:

```
Employee::where('id', '=', 1)->first();
```

As mentioned earlier in this section, methods of a Model class return the Collection object type. This collection object has multiple methods which return another collection object. This feature of the Eloquent Model is called chaining. Clauses are chained so that filters are created to filter records rather than writing WHERE clauses.

## get Method

The get method is shown in Code Snippet 12 that helps in extracting specific columns.

## Code Snippet 12:

```
Employee::get('id');
```

Follow Code Snippet 13 to find multiple columns.

## Code Snippet 13:

```
Employee::get(['id', 'name']);
```

## Knowledge Check 2:

Q2. Which of the following is used to define the column ID?

- a. get(id)
- b. bigIncrements
- c. all()
- d. find()

## 5.4.3 Modifying and Saving Records

To access the values after retrieving a record, enter the code as shown in Code Snippet 14.

## Code Snippet 14:

```
$employee = Employee::find(1);  
$id = $employee->id;  
$name = $employee->name;
```

The values can also be changed using basic assignments, and the changes can be reflected in the database using the save command. Rewrite the routes file as shown in Code Snippet 15.

## Code Snippet 15:

```
<?php

use HRM\Employee;

Route::get('modify', function () {
    $employee = Employee::find(2);
    $employee->name = 'Barry';
    $employee->salary = 70000;
    $employee->designation = "Assistant Engineer";
    $employee->save();
    return "Successfully modified";
});

Route::get('/', function() {
    return Employee::all();
});
```

Code Snippet 15 creates two routes, for ‘/’, and ‘modify’. The ‘/’ route displays all the entries within the table and as soon as you visited the second route, the record with the id of two is updated. Going back to ‘/’ will help see the reflect changes. Figures 5.2, to 5.4 show the Output.

The screenshot shows a browser window with the URL 'hrm.local/'. The page title is 'hrm.local'. Below the title bar, there are navigation icons (back, forward, search, etc.) and a link to 'hrm.local'. Underneath the title bar, there is a toolbar with 'Most Visited' and 'Getting Started' buttons. The main content area displays a JSON response. At the top of the JSON structure, there is a '0:' entry. Below it, there are two entries labeled '1:' and '2:'. Each entry contains the following fields: 'id', 'name', 'designation', 'salary', 'created\_at', and 'updated\_at'. The values for entry '1:' are: id: 1, name: "Albert", designation: "Senior", salary: "90000", created\_at: "2019-03-13 07:28:15", updated\_at: "2019-03-13 07:28:15". The values for entry '2:' are: id: 2, name: "Albert", designation: "Senior Engineer", salary: "90000", created\_at: "2019-03-13 23:18:49", updated\_at: "2019-03-13 23:18:49".

Figures 5.2: All Entries for ‘/’ Route

The screenshot shows a browser window with the URL 'hrm.local/modify'. The page title is 'hrm.local/modify'. Below the title bar, there are navigation icons (back, forward, search, etc.) and a link to 'hrm.local/modify'. Underneath the title bar, there is a toolbar with 'Most Visited' and 'Getting Started' buttons. The main content area displays a JSON response with a single entry: 'Successfully modified'.

Figures 5.3: Updating Record

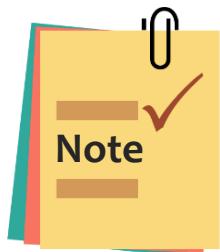
```
hrm.local/      x  +
← → ⌂ ⌂ hrm.local
Most Visited Getting Started
JSON Raw Data Headers
Save Copy Collapse All Expand All
{
  "0": {
    "id": 1,
    "name": "Albert",
    "designation": "Senior",
    "salary": "00000",
    "created_at": "2019-03-13 07:28:15",
    "updated_at": "2019-03-13 07:28:15"
  },
  "1": {
    "id": 2,
    "name": "Barry",
    "designation": "Assistant Engineer",
    "salary": "70000",
    "created_at": "2019-03-13 23:18:49",
    "updated_at": "2019-03-14 00:04:22"
  }
}
```

Figures 5.4: Updated Record

#### 5.4.4 Deleting Records

By default, Eloquent deletes records from the database in two ways: hard delete and soft delete.

Hard delete leads to permanent removal of data and cannot be retrieved. Soft delete does not remove the data from the database; instead a `deleted_at` timestamp is added.



Any record with a valid `deleted_at` will not be shown through most queries from Eloquent ORM, but can still be seen by using the `sqlite` command line interface.

Records can be deleted in any of the following ways:

- Retrieving the record and using the `delete` method.
- Using the `destroy()` method from the Model itself.

##### Delete Method

To delete the record with `id=1`, use the `delete` method as shown in Code Snippet 16.

##### Code Snippet 16:

```
//using the delete method
$employee = Employee::find(1);
$employee->delete();
```

## Destroy Method

To destroy the record with id=2, use the destroy method as shown in Code Snippet 17.

### Code Snippet 17:

```
Employee::destroy(2);
```

After deleting the record, the Web.php route file will appear as shown in Code Snippet 18.

### Code Snippet 18:

```
<?php

use HRM\Employee;

Route::get('modify', function () {
    $employee = Employee::find(2);
    $employee->name = 'Barry';
    $employee->salary = 70000;
    $employee->designation = "Assistant Engineer";
    $employee->save();
    return "Successfully modified";
});

Route::get('/', function() {
    return Employee::all();
});

Route::get('delete', function() {
    $employee = Employee::find(1);
    $employee->delete();
    return 'The record has been deleted';
});

Route::get('destroy', function(){
    Employee::destroy(2);
    return 'The record has been destroyed';
});
```

Now, perform the following steps.

1. To view all records, browse to hrm.local.
2. To delete the record with id=1, browse to hrm.local/delete.
3. To view the deleted data, browse to hrm.local.
4. To test the destroy method, browse to hrm.local/destroy.
5. To view the deletion, browse to hrm.local.

Figures 5.5, to 5.9 shows the output.

The screenshot shows a browser window with the URL 'hrm.local/'. The page title is 'hrm.local'. The content area displays a JSON response with two records:

```
0:  
id: 1  
name: "Albert"  
designation: "Senior"  
salary: "90000"  
created_at: "2019-03-13 07:28:15"  
updated_at: "2019-03-13 07:28:15"  
1:  
id: 2  
name: "Barry"  
designation: "Assistant Engineer"  
salary: "70000"  
created_at: "2019-03-13 23:18:49"  
updated_at: "2019-03-14 00:04:22"
```

Figure 5.5: All Records

The screenshot shows a browser window with the URL 'hrm.local/'. The page title is 'hrm.local'. The content area displays a JSON response with one record:

```
0:  
id: 2  
name: "Barry"  
designation: "Assistant Engineer"  
salary: "70000"  
created_at: "2019-03-13 23:18:49"  
updated_at: "2019-03-14 00:04:22"
```

Figure 5.7: Remaining Records

The screenshot shows a browser window with the URL 'hrm.local/destroy'. The page title is 'hrm.local/destroy'. The content area displays a message: 'The record has been destroyed'.

Figure 5.8: Destroying Records

The screenshot shows a browser window with the URL 'hrm.local/delete'. The page title is 'hrm.local/delete'. The content area displays a message: 'The record has been deleted'.

Figure 5.6: Record Deletion

The screenshot shows a browser window with the URL 'hrm.local/'. The page title is 'hrm.local'. The content area displays a JSON response with an empty array:

```
[]
```

Figure 5.9: No Records

To soft delete a record, the model and migration is required to be modified. Consider Code Snippet 19. When models are soft deleted, they aren't removed from the database. Instead, a deleted\_at flag is set on the model and inserted into the database. Now, to check whether a model is deleted or not, we can check deleted\_at value. If it is non-null, it can be concluded that the model has been soft deleted.

## Code Snippet 19:

```
use Illuminate\Database\Eloquent\SoftDeletes;  
...  
class Employee extends Model  
{  
    ...  
    use SoftDeletes;  
    protected $dates = ['deleted_at'];  
    ...  
}
```

To enable this feature, we must use `Illuminate\Database\Eloquent\SoftDeletes` trait on the model. We also have to add an extra column named `deleted_at` to the `$dates` property of the model. The final model file includes Code Snippet 20.

## Code Snippet 20:

```
<?php  
  
namespace HRM;  
  
use Illuminate\Database\Eloquent\SoftDeletes;  
use Illuminate\Database\Eloquent\Model;  
  
class Employee extends Model  
{  
    use SoftDeletes;  
    protected $fillable = ['name', 'salary', 'designation'];  
    protected $dates = ['deleted_at'];  
}
```

Add code from Code Snippet 21 in the up function.

## Code Snippet 21:

```
$table->softDeletes();
```

The final migration file includes Code Snippet 22.

## Code Snippet 22:

```
<!-- Stored in  
database/migrations/2019_03_12_214546_create_employees_table.php -->  
<?php  
  
use Illuminate\Support\Facades\Schema;  
use Illuminate\Database\Schema\Blueprint;  
use Illuminate\Database\Migrations\Migration;  
  
class CreateEmployeesTable extends Migration  
{  
    /**  
     * Run the migrations.  
     *  
     * @return void  
     */  
    public function up()  
    {  
        Schema::create('employees', function (Blueprint $table) {  
            $table->bigIncrements('id');  
            $table->string('name');  
            $table->string('designation');  
            $table->integer('salary');  
            $table->timestamps();  
            $table->softDeletes();  
        });  
    }  
  
    /**  
     * Reverse the migrations.  
     *  
     * @return void  
     */  
    public function down()  
    {  
        Schema::dropIfExists('employees');  
    }  
};
```

To retrieve the deleted values through ORM, use the `withTrashed` method as shown in Code Snippet 23. Further queries can be chained to it.

## Code Snippet 23:

```
$employee = Employee::withTrashed()->get();
```

To view the deleted record, consider Code Snippet 24.

## Code Snippet 24:

```
$employee = Employee::onlyTrashed()->get();
```

To undelete a record, fetch a deleted record and use the restore method as shown in Code Snippet 25.

## Code Snippet 25:

```
$employee = Employee::onlyTrashed()->get()->find(1);  
$employee->restore();
```

To permanently delete a record when soft deletes are enabled, use the forceDelete() method as shown in Code Snippet 26.

## Code Snippet 26:

```
$employee->forceDelete();
```

## Knowledge Check 3:

Q3. Which of the following is used to permanently delete a record?

- a. `restore`
- b. `forceDelete()`
- c. `withTrashed`
- d. `deleted_at`



## 5.5 Query Scopes

At times, chaining can become complex and requires considerable time and effort. For example, to find a specific record, multiple WHERE and get methods must be chained. If the query must be used again, then the code must be typed again. The re-type action may lead to typos and error in code.

The solution to such a situation is to use Query Scopes. These are functions, defined in the Model and replace the long chain with a simple one. For example, to find an employee with designation ‘Senior Engineer’ and salary above 80,000, consider the following chained query as shown in Code Snippet 27.

### Code Snippet 27:

```
$employee->where('salary', '>', 80000)->where('designation', '=', 'Senior Engineer');
```

To use the same query multiple times, define the function in the model as shown in Code Snippet 28.

### Code Snippet 28:

```
public function highPaidHighRank($query)
{
    return $query->where('salary', '>', 80000)-
>where('designation', '=', 'Senior Engineer');
}
```

Eloquent automatically provides the chained query as an argument to the \$query variable. The highPaidHighRank query scope then makes further chains from that point; thereby allowing the query scope to be used as a chained method in itself.

The final Employee model appears as shown in Code Snippet 29.

## Code Snippet 29:

```
<?php

namespace HRM;

use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Database\Eloquent\Model;

class Employee extends Model
{
    use SoftDeletes;
    protected $fillable = ['name', 'salary', 'designation'];
    protected $dates = ['deleted_at'];
    public function highPaidHighRank($query)
    {
        return $query->where('salary', '>', 80000)-
>where('designation', '=', 'Senior Engineer');
    }
}
```



## 5.6 Model Events

Eloquent fires numerous events at different points, such as when a model is being saved or deleted. Following is a list of methods that Eloquent models can fire:

- creating
- created
- saving
- saved
- deleting
- deleted
- updating
- updated
- restoring
- restored

The naming convention of the methods is self-explanatory. For example, when the save method is called, the saving event is triggered. Likewise, when the save method saved the record, the saved event is triggered.

## 5.6.1 Registering Event Listeners

An event listener is an internal function, in computer programs, that waits for an event to occur. Examples of an event are: a user clicking the mouse, network activity, or pressing a key on the keyboard.

The `EventServiceProvider` is also bundled with Laravel and is located in `app/providers/EventServiceProvider`. It includes all the event listeners in the application so that other functionalities can be used via artisan.

All events (keys) and their listeners (values) are included in the `listen` property. Multiple events can be added to the array as the application requires.

To enter the code for deleting event, edit the `EventServiceProvider` class as shown in Code Snippet 30.

### Code Snippet 30:

```
<?php

namespace simpleApp\Providers;

use Illuminate\Support\Facades\Event;
use Illuminate\Auth\Events\Registered;
use Illuminate\Auth\Listeners\SendEmailVerificationNotification;
use Illuminate\Foundation\Support\Providers\EventServiceProvider as ServiceProvider;

class EventServiceProvider extends ServiceProvider
{
    /**
     * The event listener mappings for the application.
     *
     * @var array
     */
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,
        ],
    ];
}
```

```
/**  
 * Register any events for your application.  
 *  
 * @return void  
 */  
public function boot()  
{  
    parent::boot();  
  
    Employee::deleting(function($employee) {  
        // the employee variable will get the employee record  
        // that is being deleted  
        // Write more code to do some validation before deleting  
    })  
}  
}
```

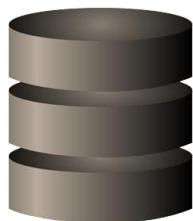


## 5.7 Relationships

Sometimes, multiple tables share the same fields. In such a case, Eloquent helps define the relationship between tables using the common fields.



To know more about Relationships, visit <https://laravel.com/docs/5.8/eloquent-relationships>.



## 5.8 Collections

The collection object in Eloquent ORM allows the chaining of methods, which facilitates interaction with the database without using a SQL query.

Methods, such as all, find, get, where are part of collections.



To know more about these and other methods, visit  
<https://laravel.com/docs/master/eloquent#collections>.





# SUMMARY

- ▶ Eloquent is a tool to write Models for databases without writing any SQL code.
- ▶ Models is part of the MVC design pattern.
- ▶ Eloquent has basic conventions that is required to be followed to make operations simpler.
- ▶ It follows the CRUD structure of operation.
- ▶ Query scopes are functions that are defined inside a Model; they can replace a long-chained method with a simple method.
- ▶ There are numerous events in Model, such as creating, created, deleted, and saving.
- ▶ The collection object in Eloquent ORM allows the chaining of methods, which facilitates interaction with the database without using a SQL query.



## Check Your Progress

Q1. The all () method returns \_\_\_\_\_.

A. json object	B. columns
C. Web.php file	D. collection object

Q2. How does the Eloquent ORM help?

A. It helps write SQL code.	B. It helps write basic conventions.
C. It helps write Models.	D. It helps create controller.

Q3. Identify the correct code snippet that references the Employee model.

A. use HRM\Employee;	B. use HRM\Employee;
\$employee = new Employee;	\$employee = \$new Employee;
C. use HM\Employee;	D. use HRM\Employee;
\$employee = new Employee;	employee = \$new Employee;

Q4. Where are query scopes defined?

A. In View	B. Controller
C. SQL	D. Model

Q5. To test the destroy method, browse to \_\_\_\_\_.

A. hrm.local/destroy	B. hrm.local
C. Web.php	D. hrm.local/delete



# Quiz answers

Question No	Answers
1	D
2	C
3	A
4	D
5	A



# Knowledge Check Answers

Question No	Answers
1	c
2	b
3	b

# ENHANCED LEARNING EXPERIENCE

Onlinevarsity





# Session 06: Implementing CRUD Operations



## Objectives

- Explain different aspects of MVC design patterns
- Understand user inputs and its validation
- Implement Session, Request, and Response
- Understand concept of Paginations
- Explain the importance of Authentication



## 6.1 Introduction

This session describes the use of Controllers in the MVC design pattern. Controllers are classes that are predefined in Laravel. They move requests and responses from the routes file to their own separate files and functions. By doing this, Controllers help in organizing Web applications. Using Controllers, it is also easy to implement Create, Read, Update, and Delete (CRUD) operations.



## 6.2 From Routes to Controllers?

Sometimes, there may be a requirement to include more features or functions in the Web application. In such a case, many closure functions are included in route files to handle HTTP requests. Thus, the application logic required to execute these increased number of functions, it lead to additional lines of code in the route file as well.

Controllers help in organizing these codes by grouping requests that handle the codes in a single route and class. Moreover, with proper conventions, multiple routes with multiple functions can be mapped to a single Controller class. The app/Http/Controllers directory stores Controllers.

Controllers can be initiated using:

- Single Action Controllers
- Resource Controllers

Create an Employee Controller, as shown in Code Snippet 1.

## Code Snippet 1:

```
php artisan make:controller EmployeeController
```

Code Snippet 1 creates a basic controller called EmployeeController.php in hrm/app/Http/Controllers. The controller will include the code as shown in Code Snippet 2.

## Code Snippet 2:

```
<?php  
  
namespace HRM\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
class EmployeeController extends Controller  
{  
    //  
}
```

Code Snippet 3 shows the code in vagrant ssh console to create a resource controller with the same name.

Delete the basic controller, if created.

## Code Snippet 3:

```
rm app/Http/Controllers/EmployeeController
```

## Knowledge Check 1:

Q1. Which one of the following directories stores Controllers?

- a. app/Http/Controllers/EmployeeController
- b. app/Http/Controllers
- c. rm/app/Http/Controllers/EmployeeController
- d. HRM\Http\Controllers



## 6.3 Working with the Resource Controller

A Resource Controller is preconfigured using a single line of artisan command. It creates a Controller framework that is commonly used for CRUD operations.

To create a Resource Controller, append the --resource flag after the original command as shown in Code Snippet 4.

### Code Snippet 4:

```
php artisan make:controller EmployeeController
```

Code Snippet 5 shows the source code of the Controller that displays the CRUD functions.

### Code Snippet 5:

```
<?php  
  
namespace HRM\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
class EmployeeController extends Controller  
{  
    /**  
     * Display a listing of the resource.  
     *  
     * @return \Illuminate\Http\Response  
     */
```

```
public function index()
{
    //
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    //
}

/**
 * Display the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    //
}
```

```
/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    //
}

/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    //
}
```

It includes the following functions:

- create()
- destroy()
- edit()
- index()
- show()
- store()
- update()

The names of the functions denote the actions that they perform and relate to CRUD operations. Moreover, the comments in the code also indicates the action. The functions can be configured to automate declaration tasks, which help in reducing the development time.

Code Snippet 6 displays the destroy function skeleton. It also includes the ‘Remove the specified resource from storage’ comment, which describes the type of parameter that can be accepted as well as the type of object that the function must return.

## Code Snippet 6:

```
/**  
 * Remove the specified resource from storage.  
 *  
 * @param int $id  
 * @return \Illuminate\Http\Response  
 */  
public function destroy($id)  
{  
    //  
}
```

Another important feature in Laravel is resource routing. This feature automatically assigns routes to all functions that have been declared in the Resource Controller with a single line of code.

Code Snippet 7 shows how to register the resourcefull route for the `http://hrm.local/employee` path.

## Code Snippet 7:

```
Route::resource('employee', 'EmployeeController');
```



1. Resourceful routes work well in subdirectories.
2. The show and destroy methods cannot be used in the Web root directory path (/) else it may create problems.
3. For this reason, the employee subdirectory will be used to implement CRUD operations on the Employee database.

Code Snippet 8 shows the redirect function to redirect to the employee directory from the root path (/).

## Code Snippet 8:

```
Route::get('/', function () {  
    return redirect('employee');  
});
```

Code Snippet 9 shows the final route routes/Web.php file where two routers are added and previously defined routers are removed.

## Code Snippet 9:

```
<?php

/*
|--------------------------------------------------------------------------
|
| Web Routes
|
|
| Here is where you can register Web routes for your
| application. These
| routes are loaded by the RouteServiceProvider within a group
| which
| contains the "Web" middleware group. Now create something
| great!
|
*/
Route::get('/', function () {
    return redirect('employee');
});

Route::resource('employee', 'EmployeeController');
```

Table 6.1 lists the routes that correspond to the defined path (employees) and are associated with the respective functions in the EmployeeController Controller.

Route	Method	Function	URI	Purpose
.create	GET	create	/create	To display the form to create a record
.destroy	DELETE	destroy	/{id}	To destroy a particular record
.edit	GET	edit	{id}/edit	To display a form to edit records
.index	GET	index	/	To display the index page
.show	GET	show	/{id}	To show a particular record
.store	POST	store	/	To add a record with details from form
.update	PUT/PATCH	update	/{id}	To update a particular record

Table 6.1: Routes and their Corresponding Path

The list also appears when the artisan command is used as shown in Code Snippet 10.

## Code Snippet 10:

```
php artisan route:list
```

Code Snippet 11 shows the output.

## Code Snippet 11:

Domain	Method	URI	Name
Action	Middleware		
	GET HEAD	/	
Closure	Web		
	GET HEAD	api/user	
Closure	api,auth:api		
	GET HEAD	employee	employee.index
HRM\Http\Controllers\EmployeeController@index	Web		
	POST	employee	employee.store
HRM\Http\Controllers\EmployeeController@store	Web		
	GET HEAD	employee/create	employee.create
HRM\Http\Controllers\EmployeeController@create	Web		
	GET HEAD	employee/{employee}	employee.show
HRM\Http\Controllers\EmployeeController@show	Web		
	PUT PATCH	employee/{employee}	employee.update
HRM\Http\Controllers\EmployeeController@update	Web		
	DELETE	employee/{employee}	employee.destroy
HRM\Http\Controllers\EmployeeController@destroy	Web		
	GET HEAD	employee/{employee}/edit	employee.edit
HRM\Http\Controllers\EmployeeController@edit	Web		

It is time to implement a few CRUD operations using the MVC pattern.



Modify the Views to allow different routes and functionalities that have been defined by the Resource Routing. Do not modify the Employee model.

To use the Employee model, add the namespace of the model by typing use HRM\Employee.

Code Snippet 12 shows the first few lines of the EmployeeController code.

### Code Snippet 12:

```
<?php  
  
namespace HRM\Http\Controllers;  
  
use Illuminate\Http\Request;  
use HRM\Employee;
```

#### 6.3.1 index() Function

To use the index() function, perform the following steps:

1. Modify the master blade template.
2. Include a few bootstrap 4 classes.

Code Snippet 13 shows an example of the same.

The image shows the Online Varsity logo. At the top, it says "Onlinevarsity". Below that is a dark bar with the text "MULTIPLE EXPERTS | MULTIPLE TOPICS". The central part of the logo features the word "BLOGS" in large, bold letters. Each letter of "BLOGS" is filled with a collage of various people's faces, suggesting a diverse community of experts. At the bottom, there is another dark bar with the text "WHERE THE EXPERTS SPEAK THE EXPERIENCE".

**Code Snippet 13:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.mi
n.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/umd/
popper.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap
.min.js"></script>

    <title>@yield('title')</title>
</head>
<body>
    <div class="container">
        <ul class="nav">
            <li class="nav-item">
                <a class="nav-link" href="{{
route('employee.index')}}">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link"
href="{{ route('employee.create')}}">Create</a>
            </li>
        </ul>
        @yield('message')
        <h1>@yield('heading')</h1>

        <div>
            @yield('content')
        </div>
    </div>
</body>
</html>
```

Code Snippet 13 uses **Bootstrap 4** for styling purposes. For example, a navigation bar is added to help a user return to the Home page from multiple routes and views the Web application may use.

The template also includes four extensible sections that are defined by the following constructs:

- @yield('content'): This is a placeholder for the actual code to perform CRUD operations.
- @yield('heading'): This is a placeholder for different headings of different views.
- @yield('title'): This is a placeholder for different titles of different child views.
- @yield('message'): This is a placeholder for displaying messages to show the result of a CRUD operation.

The route function is also used. It is a new helper function that returns a URL for the given route.

The index() function corresponds with the first route `http://hrm.local/employee`. The function is used to list resources; the kind of information that is included in the Contents page.

Code Snippet 14 shows an example where the index()function is modified such that view corresponding to the index() function prints minor details of all employees present in the database.

## Code Snippet 14:

```
public function index()
{
    $employees = Employee::all();
    return view('index')->with('employees', $employees);
}
```

In Code Snippet 12, the `$employee` variable will be passed to all the records that are in the employees table using the `Employee::all()` method. The variable will then be passed to the view index created with variable `$employee`.

To print all the records, modify the index view as shown in Code Snippet 15.

## Code Snippet 15:

```
<!-- Stored in resources/views/index.blade.php -->
@extends('master')

@section('title')
HRM - Home
@endsection

@section('heading')
Welcome to the Human Resource Management System
@endsection

@section('content')
You can find the list of all employees below. To create a new
record click the link on the navigation bar at top.



| ID                 | Name                 | Actions                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {{\$employee->id}} | {{\$employee->name}} | <a class="btn btn-success" href="{{route('employee.show', ['id'=&gt;\$employee-&gt;id])}}">Show</a>   <a class="btn btn-warning" href="{{route('employee.edit', ['id'=&gt;\$employee-&gt;id])}}">Update</a>   {{ Form::open(['method' => 'DELETE', 'route' => ['employee.destroy', \$employee->id], 'style'=>'display:inline']) }}  {{ Form::submit('Delete', ['class' => 'btn btn-danger']) }}  {{ Form::close() }} |

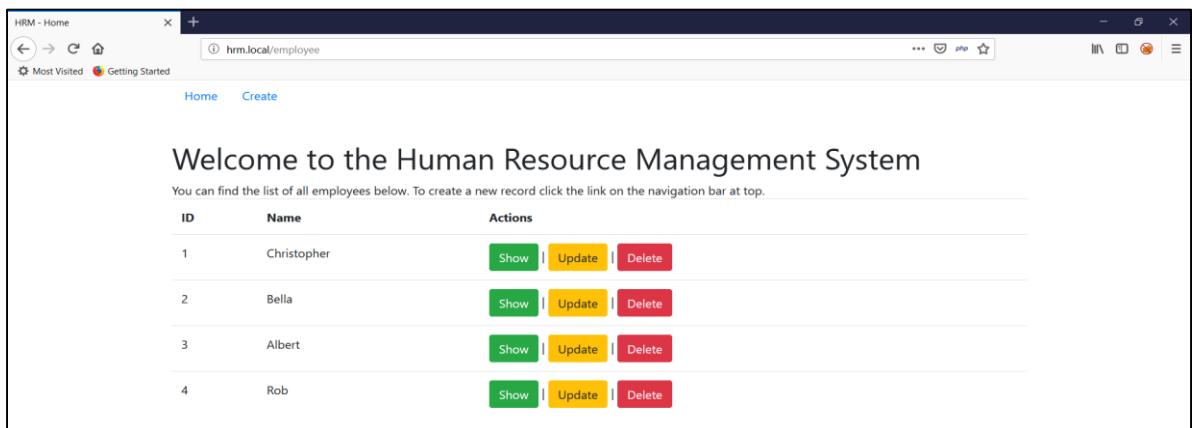

```

## Code Snippet 15:

```
@endforeach  
  
</table>  
@endsection  
  
@section('message')  
@if ($message = Session::get('Message'))  
    <div class="alert alert-success">{{ $message }}</div>  
@endif  
@endsection
```

An equivalent php for..each loop replaces the foreach construct and displays the ID and name of all the records in the database.

A page similar to the following Home page appears in the testing environment as shown in Figure 6.1.



**Figure 6.1: Home Page**

The index view includes buttons and links to different CRUD operations. The buttons and links can be replaced as per requirement.



The destroy function link is mapped with a HTTP DELETE verb. The route() helper function can only create URLs that send an HTTP GET verb. Therefore, it is necessary to use forms that can send a DELETE HTTP request to the server.

## Knowledge Check 2:

Q2. The \_\_\_\_\_ function is a new helper function that returns a URL for the given route.

- a. *yield*
- b. *bootstrap*
- c. *helper*
- d. *route*

### 6.3.2 create() and store() Functions

The create() and store() function are bundled together to form the Create function in the CRUD operations. By default, the create() function of the Resource Controller displays a form to enter values for a new employee record and the store() function includes the code to create a new record in the database.

To perform actions, the create() function can be populated with code as shown in Code Snippet 16.

#### Code Snippet 16:

```
public function create()
{
    return view('add');
}
```



In Code Snippet 14, the create() function returns the View with the name add. This can be changed as per requirement. Laravel puts no conventions on the name of views used. This does not work for the home view as the view will be overwritten if authentication is implemented.

The add View can be modified as shown in Code Snippet 17.

## Code Snippet 17:

```
<!-- Stored in resources/views/add.blade.php -->

@extends('master')
@section('title')
HRM - Create a record
@endsection

@section('heading')
    Create a new Employee record
@endsection

@section('content')
{{Form::open(['route'=>'employee.store', 'method'=>'post'])}}
<pre>
    {{ Form::label('name', 'Name:') }} {{ Form::text('name') }}
}
    {{ Form::label('designation', 'Designation:') }} {{ Form::text('designation') }}
    {{ Form::label('salary', 'Salary:') }} {{ Form::number('salary', 'value') }}
    {{ Form::submit('Submit') }}
</pre>
{{ Form::close() }}
@endsection

@section('message')
@if ($message = Session::get('Message'))
    <div class="alert alert-success">{{ $message }}</div>
@endif
@endsection
```

The View contains a form, which will send a post request to the employee.store route. The ‘message’ section prints the status message, which will be sent from the store function.

Populate the store function of the controller as shown in the Code Snippet 18.

## Code Snippet 18:

```
public function store(Request $request)
{
    Employee::create($request->all());
    return redirect()->route('employee.create')-
>with('Message', 'Employee added successfully');
}
```

Figure 6.2 shows the form where the entries can be added. The form should get displayed by clicking the **create** link on the NAV bar.

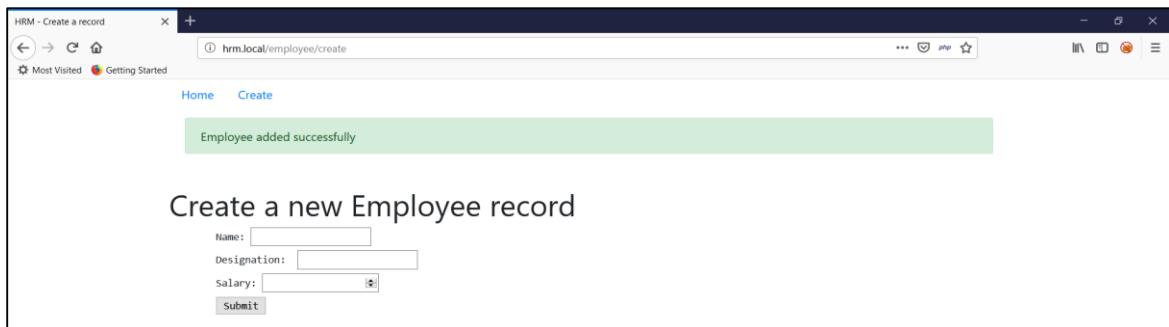


Figure 6.2: Create Employee Record

### 6.3.3 show() Function

The `show()` method displays the specified resource. It also facilitates the Read operation in the CRUD operations. Code Snippet 19 shows an example of implementing the `show()` method.

#### Code Snippet 19:

```
public function show($id)
{
    $employee = Employee::find($id);
    return view('display')->with('employee', $employee);
}
```

Code Snippet 20 shows the corresponding display view.

#### Code Snippet 20:

```
<!-- Stored in resources/views/display.blade.php -->
@extends('master')

@section('title')
HRM - Employee Details
@endsection

@section('heading')
Details for employee with id: {{$employee->id}}
@endsection
```

```
@section('content')
<table class="table">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Designation</th>
        <th>Salary</th>
        <th>Created at</th>
        <th>Last Updated at</th>
    </tr>
    <tr>
        <td>{{ $employee->id }}</td>
        <td>{{ $employee->name }}</td>
        <td>{{ $employee->designation }}</td>
        <td>{{ $employee->salary }}</td>
        <td>{{ $employee->created_at }}</td>
        <td>{{ $employee->updated_at }}</td>
    </tr>
</table>
@endsection
```

When the show button is clicked, the Web page appears as shown in Figure 6.3 and display the details of record added.

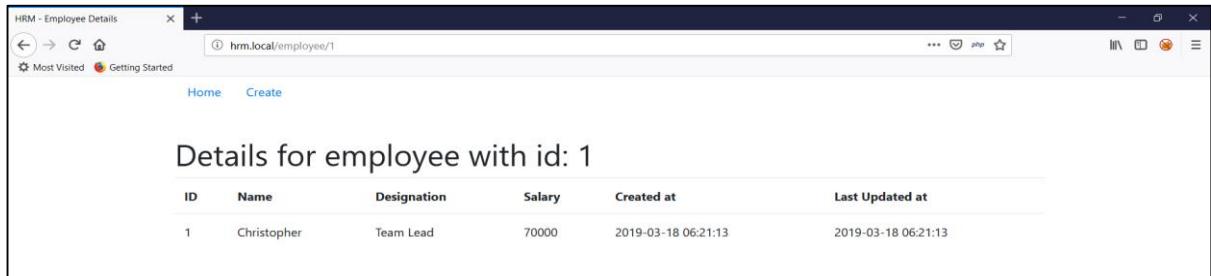


Figure 6.3: Details of Record

The page can be modified by adding a delete or edit button.

## 6.3.4 edit() and update() Operations

The Update operation is similar to the Create operation and is divided into:

- `edit()` function: Deals with providing a form to edit a record.
- `update()` function: Includes the code to save the changes.

Code Snippet 21 shows an example of using the `edit()` function.

## Code Snippet 21:

```
public function edit($id)
{
    $employee = Employee::find($id);
    return view('edit')->with('employee', $employee);
}
```

Code Snippet 22 shows the edit view.

## Code Snippet 22:

```
<!-- Stored in resources/views/edit.blade.php -->
@extends('master')

@section('title')
HRM - Edit a record
@endsection

@section('heading')
Updating records for employee with id: {{$employee->id}}
@endsection

@section('content')
{{Form::model($employee, ['route' => ['employee.update',
$employee->id], 'method'=>'PATCH'])}}
<pre>
    {{ Form::label('name', 'Name:') }} {{ Form::text('name') }}
}
    {{ Form::label('designation', 'Designation:') }} {{ Form::text('designation') }}
    {{ Form::label('salary', 'Salary:') }} {{ Form::text('salary') }}
    {{ Form::submit('Submit') }}
</pre>
{{ Form::close() }}
@if ($message = Session::get('success'))


{{$message}}


@endif
@endsection

@section('message')
@if ($message = Session::get('Message'))


{{ $message }}


@endif
@endsection
```

In the View shown in Code Snippet 22, the form builder class uses the model binding method, using the Form::model method. With this binding, names of all text boxes corresponding to a column name in the table are populated automatically with the data corresponding to the record in question.

Code Snippet 23 shows an example for the update() function.

### Code Snippet 23:

```
public function update(Request $request, $id)
{
    $employee = Employee::find($id);
    $employee->name = $request->input('name');
    $employee->designation = $request->input('designation');
    $employee->salary = $request -> input('salary');
    $employee->save();
    return
        redirect(action('EmployeeController@edit', $employee->id))->with('Message', 'Record Modified successfully');
}
```

Figure 6.4 shows the Edit page.

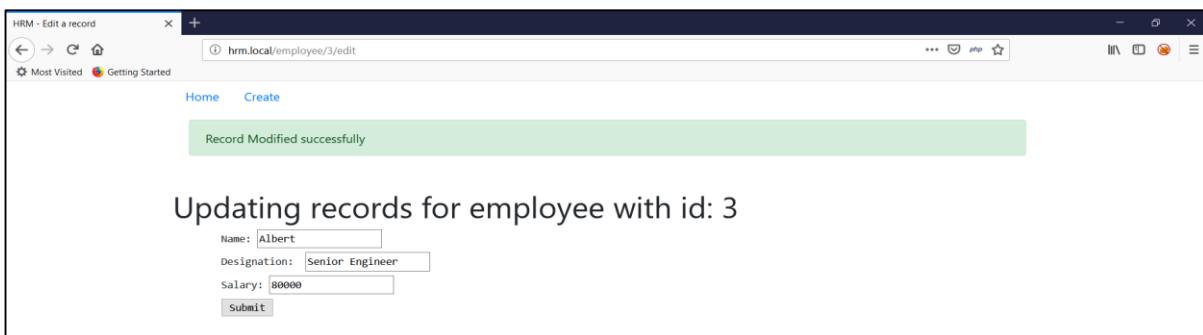


Figure 6.4: Edit Page

### 6.3.5 destroy() Function

The destroy() function deletes records. Code Snippet 24 shows an example of deleting a record.

## Code Snippet 24:

```
public function destroy($id)
{
    $employee = Employee::find($id);
    $employee->delete();
    return redirect('employee')->with('Message', 'Record
successfully deleted');
}
```

The code redirects the user back to the home page, so no additional views must be created.

Figure 6.5 shows the delete page.

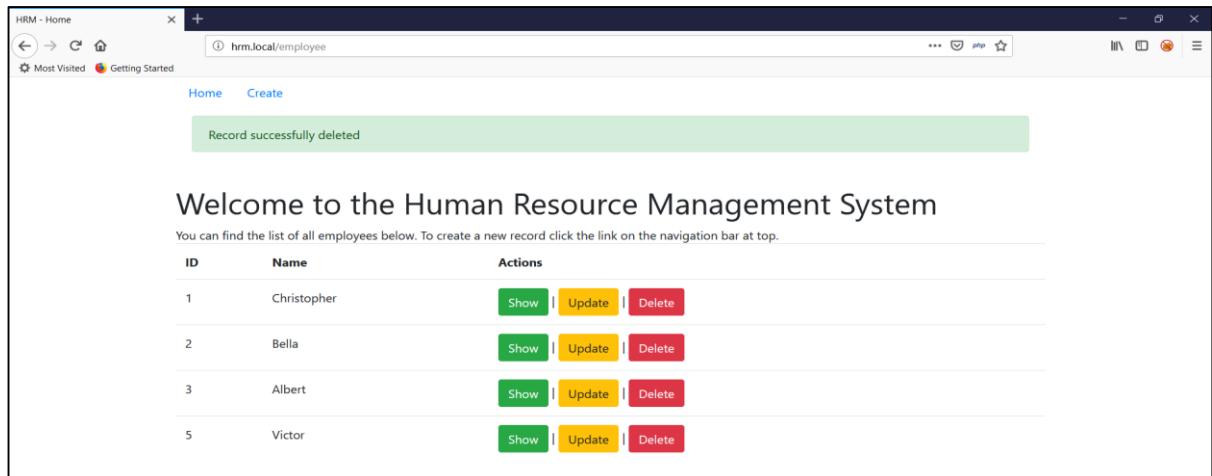


Figure 6.5: Delete Page

A Web application that can implement the basic CRUD operations is now created.



## 6.4 Understanding Requests and Response

There are parameters in the Controller class with the type Request. Code Snippet 25 shows an example.

## Code Snippet 25:

```
public function update(Request $request, $id)
{
    $employee = Employee::find($id);
    $employee->name = $request->input('name');
    $employee->designation = $request->input('designation');
    $employee->salary = $request -> input('salary');
    $employee->save();
    return
    redirect(action('EmployeeController@edit', $employee->id))->with('Message', 'Record Modified successfully');
}
```

In Code Snippet 25, the `$request` parameter is an object of the `Request` class. The `$request` object has all data relating to request, such as HTTP method and input parameters. In Code Snippet 25, the `$request` object is used to retrieve values that have been passed through forms.

The `$request->input()` method is used to retrieve values that are passed through a POST or GET request, using their parameter name. Similarly, the `$request->path()` retrieves the URL that is making the request. The `$request->method()` retrieves the method used.

The Response object returns a response object to the requesting client. All the return statements used in the routes and Controllers return a Response object.

Laravel automatically converts strings, views, redirects, and many more to a valid request object. This saves a lot of time for the developers as they don't have to deal with creating different responses depending on different computations.



## 6.5 Basic Validations in Controllers

There are many ways in which Laravel validates user input sent to servers. Incoming data is validated in the `store()` and `update()` functions of the `EmployeeController` class.

The request object `Illuminate\HTTP\Request` provides a `validate` method to validate the input. If the validation rules pass, the code executes normally, else, an exception is thrown. In addition, the corresponding error message is stored in an array named `errors`. The errors can then be handled accordingly.

Code Snippet 26 shows an example of modifying the `store()` method to add some validations.

## Code Snippet 26:

```
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required | string | max:100',
        'designation' => 'required',
        'salary' => 'required | numeric |
between:0,10000000.00',
    ]);
    Employee::create($request->all());
    return redirect()->route('employee.create')-
>with('Message', 'Employee added successfully');
}
```

The validate method takes in the incoming request through the \$request variable as its first argument, and an array containing validators as its second argument.

The associative array contains key value pairs. The keys should contain the same name as the column that they are validating. The value should contain validating keywords separated by the pipe | symbol.

Following describes the validators.

- between:x,y: Validates if the value is between x, and y.
- max:x: Validates if the string is less than x characters in length.
- numeric: Validates if the value entered is a numeric value.
- required: Validates if any value was passed for that column.
- string: Validates if the input passed is a string value.

If a new record is created with invalid values, the Laravel Web application redirects to the requesting page without modifying any values. Laravel compiles all invalidations as errors and stores them in the \$errors array along with a descriptive message.

Code Snippet 27 shows an example of modifications done to the create view to display the errors.

**Code Snippet 27:**

```
<!-- Stored in resources/views/create.blade.php -->
@extends('master')
@section('title')
HRM - Create a record
@endsection

@section('heading')
Create a new Employee record
@endsection

@section('content')
{{Form::open(['route'=>'employee.store', 'method'=>'post'])}}
<pre>
{{ Form::label('name', 'Name:') }} {{ Form::text('name') }}
{{ Form::label('designation', 'Designation:') }} {{ Form::text('designation') }}
{{ Form::label('salary', 'Salary:') }} {{ Form::number('salary', 'value') }}
{{ Form::submit('Submit') }}
</pre>
{{ Form::close() }}
@endsection

@section('message')
@if ($message = Session::get('Message'))
<div class="alert alert-success">{{ $message }}</div>
@endif

@if ($errors->any())
<div class="alert alert-danger">There were some errors in
your input:
<ul>
@foreach ($errors->all() as $error)
<li>{{$error}}</li>
@endforeach
</ul>
</div>
@endif
@endsection
```

Code Snippet 28 shows the code added in the 'message' section.

## Code Snippet 28:

```
@if ($errors->any())
    <div class="alert alert-danger">There were some errors in
your input:
    <ul>
        @foreach ($errors->all() as $error)
            <li>{{$error}}</li>
        @endforeach
    </ul>
</div>
@endif
```

Figure 6.6 displays the validation errors.

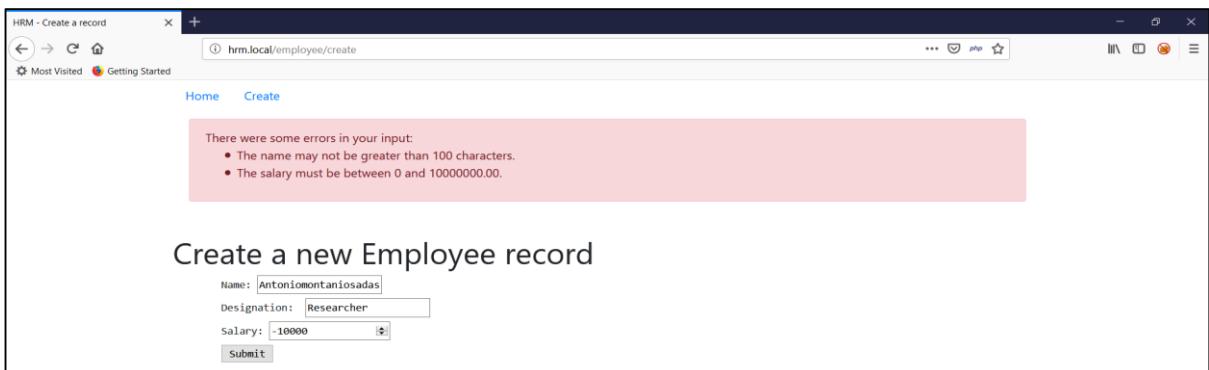


Figure 6.6: Validation Errors

Validations for the update() function can also be added.

## Knowledge Check 3:

Q3. Key value pairs must include validating keywords separated by the \_\_\_ symbol.

- a. @
- b. #
- c. |
- d. \*



To know more about different types of validators, visit <https://laravel.com/docs/5.8/validation#available-validation-rules>.



## 6.6 Adding Pagination

Pagination refers to the concept of sending records in blocks. Consider a database includes 10000 records. It is difficult to show them on a single page. In addition, sending large response increases the server load and bandwidth. These blocks of records can be organized in pages to save bandwidth and enhance accessibility.

Laravel's pagination is integrated with the query builder and the Eloquent ORM. It provides a convenient way to use pagination of database results. The HTML generated by the pagination is compatible with the Bootstrap CSS framework.

The paginate method of a Model is the simplest way to paginate. Code Snippet 29 shows how to paginate a Model.

### Code Snippet 29:

```
$employee = Employee::paginate(5);
```

Code Snippet 30 shows an example of modifying the index() method of the Controller.

### Code Snippet 30:

```
public function index()
{
    $employees = Employee::paginate(5);
    return view('index')->with('employees', $employees);
}
```

As a result, five records are sent in an instance. However, it is important to provide the ability to navigate through the pages. Code Snippet 31 shows the navigation feature in the index blade template.

### Code Snippet 31:

```
{{ $employees->links() }}
```

Code Snippet 32 shows the final index blade template.

## Code Snippet 32:

```
<!-- Stored in resources/views/index.blade.php -->
@extends('master')

@section('title')
HRM - Home
@endsection

@section('heading')
Welcome to the Human Resource Management System
@endsection

@section('content')
You can find the list of all employees below. To create a new
record click the link on the navigation bar at top.



| ID                 | Name                 | Actions                                                                                                                                                                                                                                                                                                              |
|--------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {{\$employee->id}} | {{\$employee->name}} | <a \$employee-&gt;id],="" &gt;show&lt;="" &gt;update&lt;="" 'delete',="" 'route'="&gt;" 'style'="&gt;'display:inline'])" <="" <a="" ['employee.destroy',="" a&gt;="" class="btn btn-warning" form::open(['method'="&gt;" href="{{route('employee.edit',['id'=&gt;\$employee-&gt;id)}}}" td="" {{=""  ="" }}=""> </a> |


```

```
{ { Form::submit('Delete', ['class' => 'btn btn-danger']) } }
        {{ Form::close() }}
    </td>
</tr>
@endforeach
</table>
{{ $employees->links() }}
@endsection

@section('message')
@if ($message = Session::get('success'))
    <div class="alert alert-success">{{ $message }}</div>
@endif
@endsection
```

The data on screen is now organized in pages and can be accessed and navigate through effortlessly.



## 6.7 Understanding Sessions

HTTP driven applications are stateless. This means that each request is independent of any other request. To maintain a state and give a relation to multiple requests, Web applications utilize the concept of sessions. These sessions provide a way to store information about the user across multiple requests.

Different servers may have different implementations for sessions. To help with compatibility, Laravel ships with variety of session backends that can be accessed through an API.

The session configuration is stored at config/session.php. Laravel has multiple drivers to define how the session will be stored. Following describes these drivers:

- array: sessions are not persistent; they are stored in temporary PHP arrays.
- cookie: sessions are stored in encrypted cookies.
- database: sessions are stored in the configured database.
- file: sessions are stored in storage/framework/sessions.
- memcached / redis: sessions are stored in cloud-based services.

By default, Laravel uses the file driver.

Sessions seldom require any modifications to work effectively. Although they are pre-configured, they can be configured to be stored in a database instead of a file. To configure sessions, modify the config/sessions.php file. Code Snippet 33 shows how to locate the file.

### Code Snippet 33:

```
'driver' => env('SESSION_DRIVER', 'file'),
```

Code Snippet 34 is an example of modifying the file.

### Code Snippet 34:

```
'driver' => env('SESSION_DRIVER', 'database'),
```

Now to use the session, a table is required in the database. Code Snippet 35 shows how to create a table.

### Code Snippet 35:

```
php artisan session:table  
php artisan migrate
```

Code Snippet 35 creates a new session table in the SQLite database.

Session variables can be modified using the Session class or the Request class. Session variables are defined in a key:value pair and can be defined as shown in Code Snippet 36.

### Code Snippet 36:

```
$request->session()->put('key', 'value');  
  
$value = $request->session()->get('key1');
```



## 6.8 Authentication in Laravel

Authentication is an essential part of any Web application. Each day, there are many users accessing sites. It is important for the site to identify the user credentials in order to authenticate the user. Therefore, authentication is a security based feature in a database related application.

In Laravel, authentication can be initiated as shown in Code Snippet 37.

### Code Snippet 37:

```
php artisan make:auth
```



It is recommended to use the command shown in Code Snippet 37 while creating a new application because there are chances of the Views being overwritten by Views generated by Laravel authentication mechanism.

Code Snippet 37 generates multiple Views and adds three authentication routes to the route file routes/web.php. The routes file appears as shown in Code Snippet 38.

### Code Snippet 38:

```
<?php
/*
| -----
| -----|
| Web Routes
| -----|
| |
| Here is where you can register Web routes for your application.
These
| routes are loaded by the RouteServiceProvider within a group
which
| contains the "Web" middleware group. Now create something
great!
|
*/
```

```
Route::get('/', function () {
    return redirect('employee');
});

Route::resource('employee', 'EmployeeController');
Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

Auth::routes();

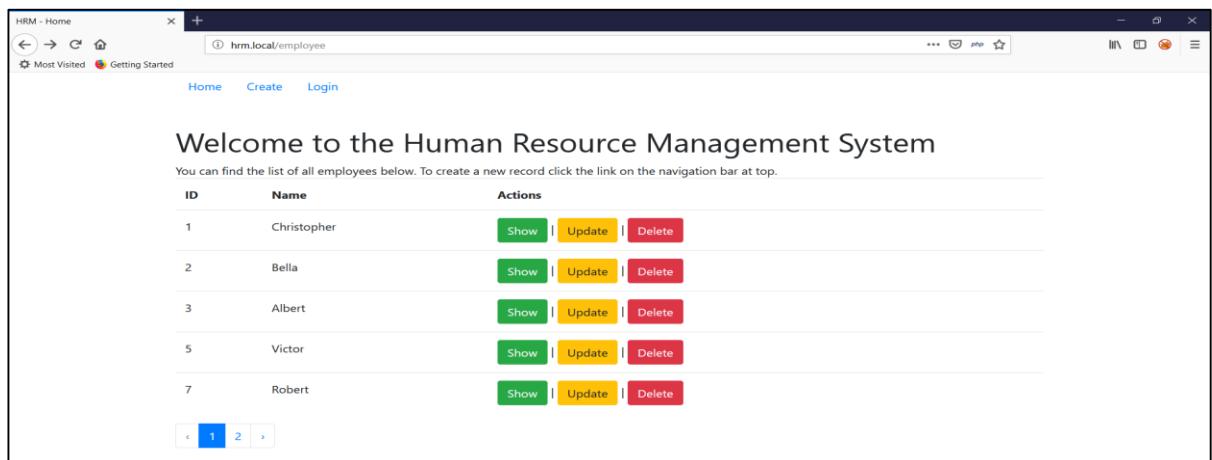
Route::get('/home', 'HomeController@index')->name('home');
```

To add the login link to the Web pages, another link can be added in the navbar section of the master layout. Code Snippet 39 shows the modifications.

### Code Snippet 39:

```
<ul class="nav">
    <li class="nav-item">
        <a class="nav-link" href="{{ route('employee.index') }}>Home</a>
    </li>
    <li class="nav-item">
        <a class="nav-link"
            href="{{ route('employee.create') }}>Create</a>
    </li>
    <li class="nav-item">
        <a class="nav-link"
            href="{{ url('/login') }}>Login</a>
    </li>
</ul>
```

Figure 6.7 shows the login option on the Web page.



**Figure 6.7: Login Option**

Clicking the Login option displays a prebuilt login page, and an option to register.

To make the routes accessible only by the registered users, Laravel provides an auth middleware. It can be directly in the Web routes file, `routes/web.php`, where the middleware method can be used on the resource route. This protects all routes defined by the resource route. Code Snippet 40 shows an example of the same.

## Code Snippet 40:

```
Route::resource('employee', 'EmployeeController')-  
>middleware('auth');
```

Now, whenever a user visits the page without logging in, the user will be redirected automatically to the login page.

To individually protect routes, edit the `EmployeeController` class, and add the code shown in Code Snippet 41.

## Code Snippet 41:

```
$this->middleware('auth');
```

The final code for the `create()` method appears as shown in Code Snippet 42.

## Code Snippet 42:

```
public function create()
{
    $this->middleware('auth');
    return view('create');
}
```

The code redirects all unauthenticated users to the login page.



To know more about all the authentication features, visit  
<https://laravel.com/docs/5.8/authentication>.

# Onlinevarsity





# SUMMARY

- ▶ With proper conventions, multiple routes with multiple functions can be mapped in a single Controller class.
- ▶ Controllers can be initiated using Single Action Controllers and Resource Controllers.
- ▶ A Resource Controller creates a Controller framework that is commonly used for CRUD operations.
- ▶ The resource routing feature automatically assigns routes to all functions that have been declared in the Resource Controller with a single line of code.
- ▶ The `create()` and `store()` functions are bundled together to form the Create function in the CRUD operations.
- ▶ The `show()` method facilitates the Read operation in the CRUD operations.
- ▶ Pagination refers to the concept of sending records in blocks.
- ▶ Laravel's paginator is integrated with the query builder and the Eloquent ORM.



## Check Your Progress

Q1. \_\_\_\_\_ validates that the value was passed for the respective column.

A. numeric	B. required
C. between:x,y	D. string

Q2. Which of the following functions includes the code to save the changes?

A. edit	B. save
C. create	D. update

Q3. Which of the following features does Laravel provide to protect routes?

A. auth middleware	B. routeware
C. request function	D. response object

Q4. \_\_\_ sessions are stored in cloud-based services.

A. cookie	B. database
C. memcached	D. array

Q5. Sending records in blocks is a feature of \_\_\_\_\_.

A. Validation	B. Session
C. Resource Controllers	D. Pagination



# Quiz answers

Question No	Answers
1	B
2	D
3	A
4	C
5	D



# Knowledge Check Answers

Question No	Answers
1	b
2	d
3	c