Onlinevarsity

Programming for the Web with PHP

Are you ☐☐☐☐☐☐☐☐☐☐ with Onlinevarsity.com? R G E T R E E S I D

Did you ☐☐☐☐☐☐☐ this ebook? D O O W L N A D

Do you have a ☐☐☐☐☐ copy of this ebook? L G A E L

Are you a victim of ☐☐☐☐☐? P C I A R Y

Answers: REGISTERED, DOWNLOAD, LEGAL, PIRACY

Download a **LEGAL** copy of this ebook which you can say is **mine** because **PIRACY** is a **crime**

Programming for the Web with PHP Learner's Guide

© 2016 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

First Edition - 2016

Aptech®
Unleash your potential

Aptech®
COMPUTER EDUCATION
Unleash your potential

Dear Learner,

We congratulate you on your decision to pursue an Aptech Worldwide course.

Aptech Ltd. designs its courses using a sound instructional design model – from conceptualization to execution, incorporating the following key aspects:

- Scanning the user system and needs assessment

Needs assessment is carried out to find the educational and training needs of the learner

Technology trends are regularly scanned and tracked by core teams at Aptech Ltd. TAG* analyzes these on a monthly basis to understand the emerging technology training needs for the Industry.

An annual Industry Recruitment Profile Survey# is conducted during August - October to understand the technologies that Industries would be adapting in the next 2 to 3 years. An analysis of these trends & recruitment needs is then carried out to understand the skill requirements for different roles & career opportunities.

The skill requirements are then mapped with the learner profile (user system) to derive the Learning objectives for the different roles.

- Needs analysis and design of curriculum

The Learning objectives are then analyzed and translated into learning tasks. Each learning task or activity is analyzed in terms of knowledge, skills and attitudes that are required to perform that task. Teachers and domain experts do this jointly. These are then grouped in clusters to form the subjects to be covered by the curriculum.

In addition, the society, the teachers, and the industry expect certain knowledge and skills that are related to abilities such as *learning-to-learn, thinking, adaptability, problem solving, positive attitude etc*. These competencies would cover both cognitive and affective domains.

A precedence diagram for the subjects is drawn where the prerequisites for each subject are graphically illustrated. The number of levels in this diagram is determined by the duration of the course in terms of number of semesters etc. Using the precedence diagram and the time duration for each subject, the curriculum is organized.

- Design & development of instructional materials

The content outlines are developed by including additional topics that are required for the completion of the domain and for the logical development of the competencies identified. Evaluation strategy and scheme is developed for the subject. The topics are arranged/organized in a meaningful sequence.

The detailed instructional material – Training aids, Learner material, reference material, project guidelines, etc.- are then developed. Rigorous quality checks are conducted at every stage.

- Strategies for delivery of instruction

  Careful consideration is given for the integral development of abilities like thinking, problem solving, learning-to-learn etc. by selecting appropriate instructional strategies (training methodology), instructional activities and instructional materials.

  The area of IT is fast changing and nebulous. Hence considerable flexibility is provided in the instructional process by specially including creative activities with group interaction between the students and the trainer. The positive aspects of web based learning –acquiring information, organizing information and acting on the basis of insufficient information are some of the aspects, which are incorporated, in the instructional process.

- Assessment of learning

  The learning is assessed through different modes – tests, assignments & projects. The assessment system is designed to evaluate the level of knowledge & skills as defined by the learning objectives.

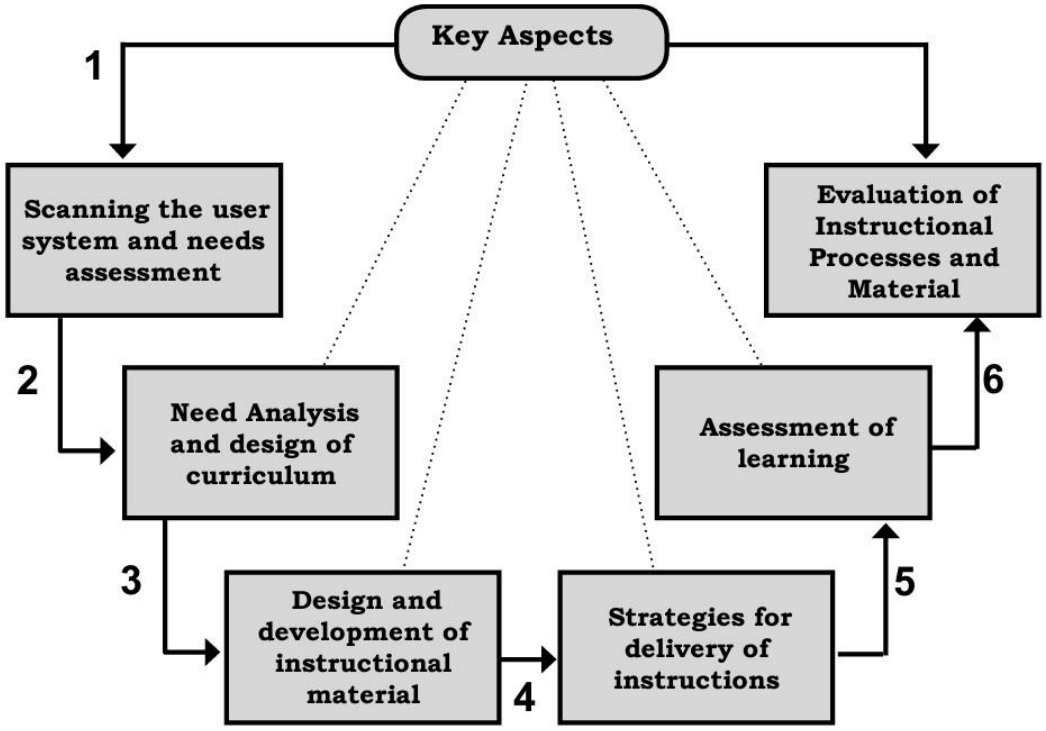
- Evaluation of instructional process and instructional materials

  The instructional process is backed by an elaborate monitoring system to evaluate - on-time delivery, understanding of a subject module, ability of the instructor to impart learning. As an integral part of this process, we request you to kindly send us your feedback in the reply pre-paid form appended at the end of each module.

*TAG – Technology & Academics Group comprises of members from Aptech Ltd., professors from reputed Academic Institutions, Senior Managers from Industry, Technical gurus from Software Majors & representatives from regulatory organizations/forums.

Technology heads of Aptech Ltd. meet on a monthly basis to share and evaluate the technology trends. The group interfaces with the representatives of the TAG thrice a year to review and validate the technology and academic directions and endeavors of Aptech Ltd.

Aptech New Products Design Model



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 supports cross-platform and supports various databases, such as MySQL, mSQL, Oracle, and PostgreSQL.

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 the year 2000, PHP 4.0 was released with security enhancements. PHP 4.0 provided support for multiple Web servers, Hypertext Transfer Protocol (HTTP) sessions, output buffering, and security for user inputs. PHP 5.0 was released in the year 2004. PHP 5.3.6 was released in March 2011. This book covers PHP 7.0.4.

In this book, various new features of PHP 7 are introduced such as spaceship operator, null coalescing operator, scalar type declarations, generator return expressions, and so on. Anonymous classes, behavior of weak type checking, behavior of strict type checking, and changes in error handling are also discussed in this book.

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

Onlinevarsity App

for Android devices

Download from Google Play Store

Table of Contents

Sessions

1. Introduction to PHP
2. Installing and Configuring PHP 7
3. New Features of PHP 7
4. Form Handling in PHP
5. Using Variables and Expressions in PHP
6. Using Variables and Expressions in PHP (Lab)
7. Scalar Type Declarations
8. PHP Operators
9. PHP Operators (Lab)
10. Conditional Statements in PHP
11. Conditional Statements in PHP (Lab)
12. Flow Control in PHP
13. Flow Control in PHP (Lab)
14. Functions in PHP
15. Functions in PHP (Lab)
16. Working with Arrays
17. Working with Arrays (Lab)
18. Handling Databases with PHP
19. Working with Cookies

Sessions

20. Working with Cookies (Lab)
21. Session Management in PHP
22. Handling E-mail with PHP
23. Handling E-mail with PHP (Lab)
24. OOP Concepts
25. OOP Concepts (Lab)
26. Generator Delegation and Throwable Interface

Onlinevarsity

BLOGS

WHERE THE EXPERTS SPEAK THE EXPERIENCE

Session 1

Introduction to PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the history of PHP.*
- *Identify the need for PHP.*
- *Explain PHP tools.*
- *Explain HTTP headers.*

1.1 Introduction

PHP stands for Hypertext Preprocessor. It is an open source scripting language embedded within Hypertext Markup Language (HTML) codes and is used for developing dynamic Web pages. PHP scripts are executed on the Web server.

In this session, you will learn about the history of PHP. You will also learn about Hypertext Transfer Protocol (HTTP) headers using PHP. In addition, you will learn to pass variables using a URL.

1.2 History of PHP

PHP, created in 1994 by Rasmus Lerdorf originally stood for Personal Home Page. Rasmus Lerdorf used a set of Perl scripts to maintain his personal home page and track access. These were replaced with Personal Home Page Tools, which were a set of Common Gateway Interface (CGI) binaries written in C programming language. In order to add more functionality, these Personal Home Page Tools were incorporated with Form Interpreters (FI) to create an advanced version of PHP called PHP/FI. Form interpreters are a set of Perl scripts used by a form to process data from a Web form. PHP/FI enabled communication with databases and development of simple dynamic Web applications.

In the year 1997, PHP/FI advanced to PHP/FI 2.0 with thousands of users installing it in approximately 50,000 server domains. However, PHP/FI 2.0 lacked extensive features which led to the development of PHP 3.0.

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 syntax support.

Session 1

In the year 2000, PHP 4.0 was released with security enhancements. PHP 4.0 provided support for multiple Web servers, HTTP sessions, output buffering, and security for user inputs. PHP 5.0 was released in the year 2004. PHP 5.3.6 was released in March 2011. PHP 7 was launched in December 2015.

1.3 Need for PHP

PHP is a scripting language suitable for server-side scripting and provides dynamic content from a Web server to a Web client.

The advantages of using PHP are as follows:

- Is easy to learn, use, and implement
- Is freely available and can be customized
- Can be executed on any Web server on any platform

PHP is a set of scripting tools used for performing various server-side functions. In addition, it can also be used for command line scripting and for developing client-side Graphical User Interface (GUI) applications that are platform independent.

The different uses of PHP are as follows:

- **Application Control** - is specifically designed to control access logging for HTTP servers. Initially it was used as an application control language.
- **Database Access** - is designed to act as a middleware to provide a Web interface for accessing a database. PHP can access any Structured Query Language (SQL) or Open Database Connectivity (ODBC) database and read and write data from these databases.
- **File Access** - is used to work with files. It can read, write, and edit documents remotely. In other words, it can be used for file and directory maintenance. PHP can also be used for generating files in various formats, such as Portable Document File (PDF) and HTML. It is increasingly used to process Extensible Markup Language (XML) data for HTTP distribution. It generates an e-mail by retrieving data from documents and sending it through any standard mail protocol.
- **Graphics** - is used to create graphs and charts and generate image files, such as Graphics Interchange Format (GIF) and Portable Network Graphics (PNG).
- **Server-Side Scripting** - is designed to implement server scripts. The client requests will be processed on the server. A PHP parser, a Web server, and a Web browser are required to implement server-side scripting.

- **Command Line Scripting** - is used to execute scripts on the UNIX/Linux platforms. A Web server and a Web browser are not required in command line scripting; only the PHP parser is required.

- **Desktop Applications** - is used to create desktop applications with a GUI. Advanced features of PHP can be implemented in client applications to enable interactive GUI.

1.4 Introduction to PHP Tools

PHP tools are used for developing and designing a Web page. They are text editors that allow developers to design Web sites. PHP tools can be implemented after installation of PHP.

1.4.1 PHP Tools

PHP tools are programming text editors containing all the supporting features that enable fast development of Web sites. The tools that are used for developing dynamic Web pages are as follows:

- **PHPDebugger DBG** – Enables step by step execution and debugging of a PHP script without changing the PHP code. It supports multiple debugging processes simultaneously. A PHP Debugger simplifies PHP debugging because you can control the execution of PHP scripts without changing the code. For example, using PhpED's PHP Debugger from NuSphere, you can perform various PHP debugging operations including:

  - Inspecting PHP variable and class
  - Modifying the value of a PHP variable
  - Defining breakpoints to interrupt script execution
  - Invoking a PHP function to view results
  - Navigating through the code
  - Viewing the line generating error
  - Enabling secure debugging with Secure Shell (SSH)

- **ionCube Standalone PHP Encoder** - Protects the PHP code and ensures security and runtime performance.

- **Codelock** - Enables to encrypt both PHP and HTML code and protect Web pages.

- **PHing** - Is a PHP build system, which enables to design a Web application in a structured manner.

- **NuSphere PHPEd** - Is an Integrated Development Environment (IDE) for PHP and a complete platform for developing PHP based Web applications. It enables to create, debug, profile, deploy, and integrate PHP code.

- **xored:WebStudio** - Is an IDE for PHP. Built on Eclipse platform, this tool comprises a set of Eclipse editing, debugging, and deployment tools.

- **PHPmole** - Is a combination of Dreamweaver and Microsoft Visual Studio and runs on a GNOME platform to work with PHP. A GNOME platform is a desktop environment for Linux. This tool has an object based design and a user friendly interface.

- **Simplewire PHP SMS Software Development Kit (SDK)** - Provides a wireless text-messaging platform. It enables to embed messaging services into an application, which can be sent to mobile devices.

- **Quanta Plus Web Development Environment** - Is a Web development environment to edit XML, HTML, PHP, and other text based Web documents.

- **K PHP Develop** - Is an integrated Web development tool with different modules, such as server, server setup, and client and plug-ins to access database servers, such as MySQL and Sybase.

- **gedit** - Is a GNOME based text editor for writing PHP scripts. It is installed as the default text editor in Linux operating systems.

1.5 HTTP Headers

HTTP is a network transmission protocol that transfers hypertext files. It provides instructions for communication between the client and the server. HTTP runs on the Transmission Control Protocol/Internet Protocol (TCP/IP) suite, which is the foundation protocol suite for the Internet.

1.5.1 Structure of an HTTP Message

An HTTP header is an Internet protocol containing instructions to transfer information between a Web client and a Web server. The instruction can either be a request sent by the client to the server for some resource or a response from the server to a client request. There are two types of headers namely, request headers and response headers.

The format for request and response headers is similar and contains the following structure or components:

- A request or a response line

- HTTP header lines

- A blank line
- A message body, which is optional

The format of an HTTP message is as follows:

```
<initial line, different for request vs. response>
    Header1: value1
    Header2: value2
    Header3: value3
    Blank line
    <optional message body, like file contents or query data>
```

1.5.2 Initial Request or Response Line

The first part of an HTTP header, which is a request or a response line, differs in format.

A request line contains the following information separated by spaces:

- An HTTP method name
- The address or path of the requested resource. It is also called the Uniform Resource Identifier (URI)
- The HTTP version being used

Code Snippet 1 displays an initial line with a request message.

Code Snippet 1:

```
GET /sample.html HTTP/1.1
```

The code uses the HTTP method, `GET`, to request for a `sample.html` file for an HTTP client of version 1.1. The HTTP version is always specified in '`HTTP/x.x`' upper case format.

A response line consists of the following three components separated by spaces.

- The HTTP version
- A response code indicating the result of the request
- An English phrase describing the response code

Concepts

Code Snippet 2 displays an initial line with a response message.

Code Snippet 2:

```
HTTP/1.0 500 Internal Server Error
```

where,

`HTTP/1.0` - specifies the HTTP version

`500` - specifies the response code

`Internal Server Error` - specifies the description of the response code

1.5.3 Header Lines

The header lines provide information about the request or response or the data sent in the message body.

The syntax for a header line is as follows:

Syntax:

```
Header-Name: value
```

HTTP headers are classified into the following categories:

- **General** - Is used to control the processing of a message and provide extra information to the receiver. They are not specific to any request or response message.
- **Entity** - Provides information about the entity, if present in the body of a request.
- **Request or response** - Provides the server with details about the client's request and enables the client to have control on the processing of requests. On receiving the request, the server returns the response header attached with the response being sent. The headers are specific for request or response messages.

Code Snippet 3 displays HTTP header lines.

Code Snippet 3:

```
GET /sample.html HTTP/1.1
User-agent: Mozilla/4.0
Last-Modified: Mon, 11 Apr 2011 23:07:07 GMT
Accept-Language: en
[ blank line above ]
```

where,

`GET` - is the initial request line that specifies GET as the method, the requested file name and the version of HTTP used

`User-agent` - is a header line that specifies the name of the browser and the version

`Last-Modified` - is a header line that specifies the date and time when the resource was last modified

`Accept-Language` - is a header line that specifies the language preference as English

1.5.4 Message Body

The third and an optional component of an HTTP header is the message body that appears after the header lines. In a response message, the requested resource is returned to the client in the message body. In a request message, the message body will contain user data and uploaded files that are sent to the server.

Session 1

Summary

- PHP is an open source scripting language embedded within HTML codes and used for developing dynamic Web pages.
- PHP is used for executing scripts from the command line and for developing client-side GUI applications that are platform independent.
- PHP is used for generating files in PDF and HTML formats.
- PHP can generate an e-mail by retrieving data from documents and sending it through any standard mail protocol.
- The popular text editor used for writing PHP scripts on Linux platform is gedit.
- A HTTP message or protocol is divided into three parts, the request or response line, the HTTP header, and the body of the protocol.

Session 1

Check Your Progress

1. PHP works in _____ language and uses _____ tags within it.

   a. HTML, FORM

   b. HTML, PHP

   c. Scripting, HTML

   d. C, directive

2. Which of the following command is used in the HTTP header to specify a request?

   a. echo

   b. GET

   c. display

   d. PRINT

3. Which of the following PHP tool enables to control the step-by-step execution of a script?

   a. PHPEd

   b. PHPDebugger DBG

   c. Codelock

   d. Phing

4. xored PHP tool is built using _____.

   a. GNOME

   b. Eclipse

   c. MySQL

   d. Scala

Check Your Progress

5. Which of the following PHP tool provides a wireless text-messaging platform?

   a. Phing

   b. xored

   c. PHPEd

   d. Simple wire PHP SMS Software Development kit

Session 1

Introduction to PHP

Answers

1. c
2. b
3. b
4. b
5. d

Onlinevarsity

JOIN GROUPS
FOR COLLABORATIVE
LEARNING

Session 2

Installing and Configuring PHP 7

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the pre-requisites for installing PHP 7.*
- *Describe the steps to configure PHP 7.*
- *Identify the steps to install PHP 7.*
- *Describe the process to create simple PHP scripts.*
- *Explain how to use HTTP headers in PHP.*

2.1 Introduction

This session explains how to install and configure PHP 7. This includes various steps to be followed for a successful installation. The session also describes how to create and execute simple PHP scripts and use HTTP headers in PHP.

2.2 Pre-requisites for Installing PHP 7

In order to install PHP 7, you first need to have a Web server and database installed on your system.

Web servers supporting PHP are as follows:

- Apache
- Microsoft Internet Information Service (IIS)
- Nginx

Some of the databases that could be used with PHP include:

- DB2
- Microsoft SQL Server
- MySQL

- Oracle
- PostgreSQL

To run PHP 7, you need to have a Web browser installed in addition to a Web server.

Note: Ensure that you remove any existing PHP 5.x installation before attempting to install PHP 7.x.

2.3 Installing the Packages

You can install PHP on Linux, Windows, or Mac OS systems. On the http://php.net/downloads.php Web site, there are various downloads pertaining to the stable releases. For example, PHP 7.0.4 can be downloaded from http://php.net/get/php-7.0.4.tar.gz/from/a/mirror page.

To extract the PHP packages on Linux, perform these steps:

1. Run the following commands at the command prompt:

```
# tar xjvf php-7.0.4.tar.gz

# tar -xf php-7.0.4.tar

# cd php-7.0.4/
```

   It is required to install some development packages before compiling PHP 7.0.4.

2. To do this, run the following command at the command prompt.

```
# dnf install aspell-devel bzip2-devel freetype-devel gmp-devel libXpm-
devel libcurl-devel libjpeg-turbo-devel libmcrypt-devel libpng-devel
libxml2-devel libxslt-devel mariadb-devel recode-devel uw-imap-devel gcc
openssl-devel -y
```

2.4 Configuring PHP 7.0.4

To configure PHP 7.0.4, perform the following steps:

1. Type the following at the command prompt.

```
#cd php-7.0.4

# ./configure --prefix=/usr/local/php7 --with-zlib-dir --with-freetype-dir
--enable-mbstring --with-libxml-dir=/usr --enable-soap --enable-calendar
--with-curl --with-mcrypt --with-zlib --with-gd --disable-rpath --enable-
inline-optimization --with-bz2 --with-zlib --enable-sockets --enable-
sysvsem --enable-sysvshm --enable-pcntl --enable-mbregex --enable-exif
--enable-bcmath --with-mhash --enable-zip --with-pcre-regex --with-mysqli
--with-pdo-mysql --with-mysqli --with-jpeg-dir=/usr --with-png-dir=/usr
--enable-gd-native-ttf --with-openssl --with-fpm-user=apache --with-fpm-
group=apache --with-libdir=/usr/lib --enable-ftp --with-kerberos --with-
gettext --with-xmlrpc --with-xsl --enable-opcache --enable-fpm
```

Figure 2.1 shows PHP 7.0.4 being configured.

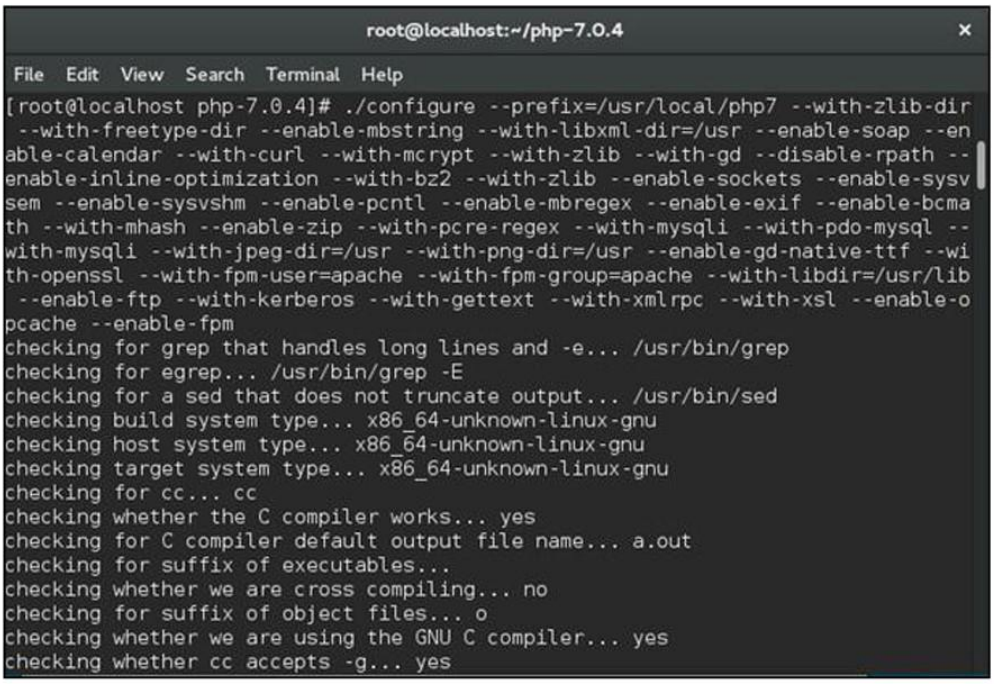


Figure 2.1: Output of the Configure Command

Concepts

2.5 Installing PHP 7.0.4

Once you have extracted the packages, installed the prerequisites, and configured the PHP files, you are ready to install it. The procedure for the same is as follows:

1. Run the following command from the command prompt:

   ```
   #make
   ```

   The `make` command is used to determine the files that are large to recompile and issue commands to recompile those large files.

   Figure 2.2 shows recompiling the files with the `make` command.

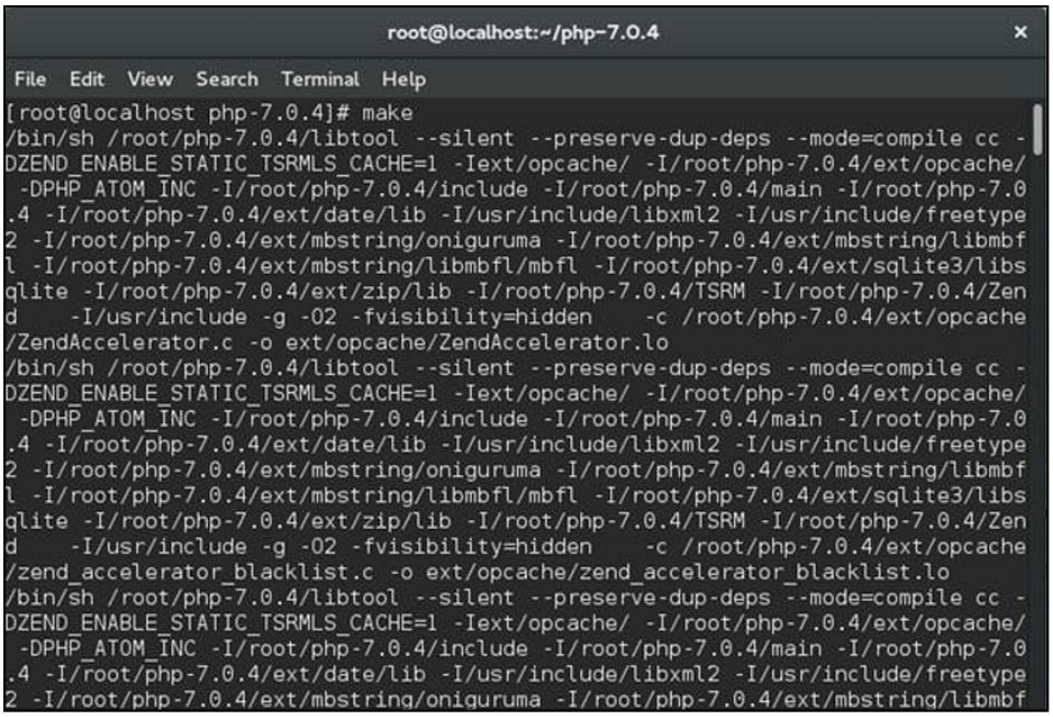


Figure 2.2: Output of the make Command

2. Run the following command to install PHP 7.0.4:

```
# make install
```

The output is shown in figure 2.3.

```
root@localhost:~/php-7.0.4
File  Edit  View  Search  Terminal  Help
[root@localhost php-7.0.4]# make install
exit 0; -t /root/php-7.0.4/ext/json/php_json_scanner_defs.h --no-generation-date
 -bci -o /root/php-7.0.4/ext/json/json_scanner.c /root/php-7.0.4/ext/json/json_s
canner.re
Installing shared extensions:     /usr/local/php7/lib/php/extensions/no-debug-no
n-zts-20151012/
Installing PHP CLI binary:        /usr/local/php7/bin/
Installing PHP CLI man page:      /usr/local/php7/php/man/man1/
Installing PHP FPM binary:        /usr/local/php7/sbin/
Installing PHP FPM config:        /usr/local/php7/etc/
Installing PHP FPM man page:      /usr/local/php7/php/man/man8/
Installing PHP FPM status page:   /usr/local/php7/php/php/fpm/
Installing phpdbg binary:         /usr/local/php7/bin/
Installing phpdbg man page:       /usr/local/php7/php/man/man1/
Installing PHP CGI binary:        /usr/local/php7/bin/
Installing PHP CGI man page:      /usr/local/php7/php/man/man1/
Installing build environment:     /usr/local/php7/lib/php/build/
Installing header files:           /usr/local/php7/include/php/
Installing helper programs:       /usr/local/php7/bin/
  program: phpize
  program: php-config
Installing man pages:             /usr/local/php7/php/man/man1/
  page: phpize.1
  page: php-config.1
```

Figure 2.3: Output of the Make Install Command

2.6 Setting Up Apache to Use PHP

Once you have successfully installed PHP, you need to set up your Web server to work with PHP. If you are using Apache as the Web server, open the httpd.conf file and add the following directives:

```
AddHandler application/x-httpd-php .php

LoadModule php7_module C:\php7.dll

AddType application/x-httpd-php .php

PHPIniDir C:\php
```

Concepts

You may change the path of the PHP installation folder depending upon your actual path. Then, save and restart the Apache Web server. To test if the action was successful, launch the following link in a browser.

```
http://localhost
```

2.7 Writing a Simple PHP Script

Writing a PHP script is similar to writing an HTML script. A PHP file can include simple text, HTML tags, and PHP script.

The rules to be followed while creating a PHP script are as follows:

- Embed PHP scripts in the BODY tag of an HTML file
- Start and end every block of PHP code with <?php and ?> tags
- End a PHP statement with a semicolon, ';'
- Save all PHP files with a **.php** extension

Note: The .php extension enables the Web server to process the file as a PHP file.

Code Snippet 1 displays a simple PHP script.

Code Snippet 1:

```
<html>
<body>
<title>PHP Syntax Example</title>
<?php
echo "Hello World";
?>
</body>
</html>
```

The `echo` command in PHP is used to send data to the browser. This command is used to print data on a browser. The instructions for the `echo` command are included within the `php` tags `<?php` and `?>` and ends with a semicolon. The file is saved with a **.php** extension and will display 'Hello World' on the browser when executed.

2.7.1 Comments in a PHP Script

Comments in a code assist a programmer to interpret the meaning of a code. They are not displayed in the output and are meant only for the programmers. PHP supports both single-line and multi-line comments.

Code Snippet 2 illustrates the use of comments in a PHP script.

Code Snippet 2:

```
<?php
// This is a single-line comment
/* and this is a
multi-line
comment */
?>
```

To display current date using the PHP script, open the **gedit** text editor and enter the code as shown in Code Snippet 3.

Code Snippet 3:

```
<HTML>
<BODY>
The Date is:
<?php echo gmdate("M d Y");
?>
</BODY>
</HTML>
```

Figure 2.4 displays the text editor after adding the code.

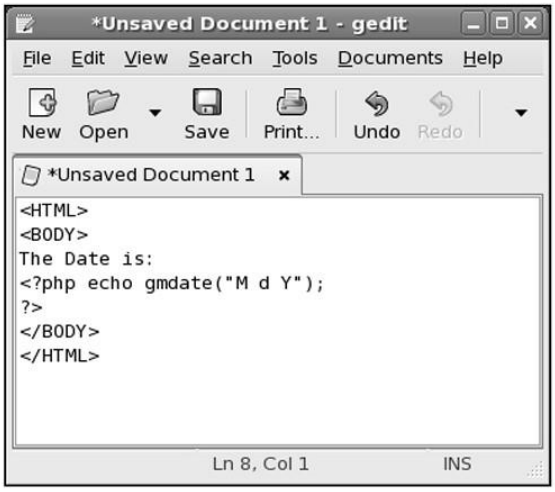


Figure 2.4: PHP Script to Display the Date

In figure 2.4, the PHP script is included within the `BODY` tag of the HTML code. A code line must end with a semicolon to separate one instruction from another.

The PHP script engine processes the text enclosed within `<?php` and `?>` tags. After processing the PHP script, the contents are returned to the browser as a normal Web page.

1. Save the file as `date.php` in the `/usr/local/apache2/htdocs` directory.

2. Open the Mozilla Firefox Web browser.

3. Enter `http://localhost/date.php` in the Address bar and press **Enter**. Figure 2.5 displays the output.

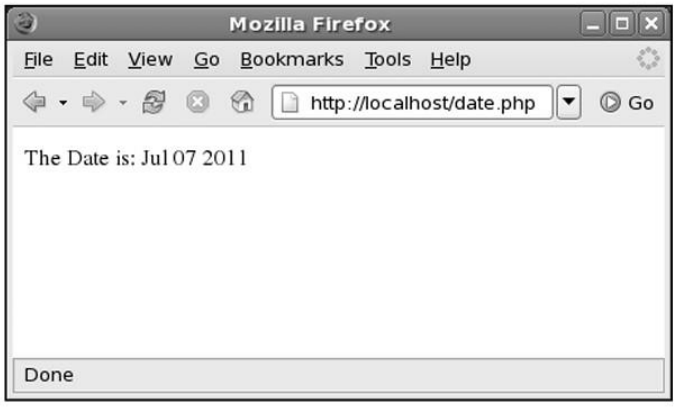


Figure 2.5: Displaying Current Date

Session 2

In the PHP script, an `echo` command is used. The `echo` command displays the output in a Web browser. The `gmdate()` is a PHP function that enables to display the current date and time in the browser. The letters gm stand for Greenwich Mean Time. The letter, `M`, used in the `gmdate()` function displays only first three letters of the month in the current date. The letter `d` displays the current date and `Y` displays all the four digits of the current year.

To display a simple text in the browser using the PHP script:

1. Open a new file in the gedit text editor.

2. Enter the code, as shown in Code Snippet 4.

Code Snippet 4:

```
<HTML>
<BODY>
<?php echo "Hello Everybody";
?>
</BODY>
</HTML>
```

Figure 2.6 displays the text editor after adding the code.

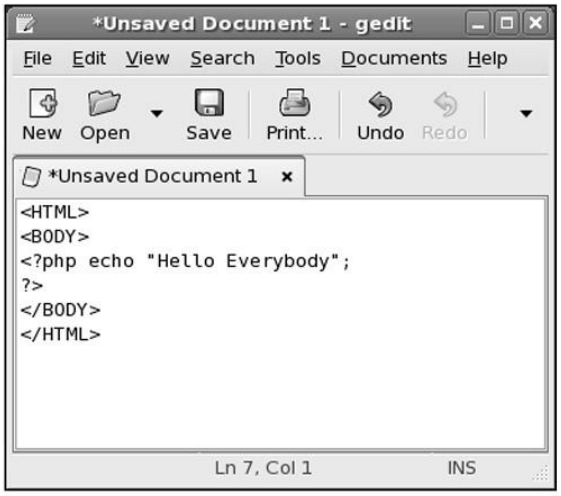


Figure 2.6: PHP Script for Displaying Message

Note: The `echo` command is used to display text. The text to be displayed is enclosed within double quotes.

Session 2

3. Save the file as `stringdisp.php` in the `/usr/local/apache2/htdocs` directory.

4. Open the Mozilla Firefox Web browser.

5. Type `http://localhost/stringdisp.php` in the Address bar and press **Enter**.

   Figure 2.7 displays the output of the script.

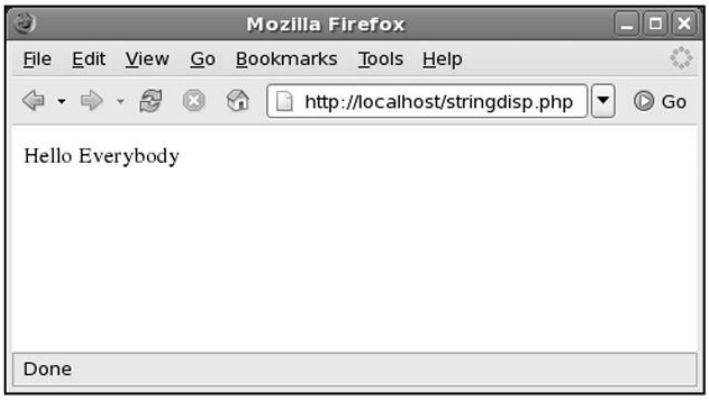


Figure 2.7: Displaying Text

The rules to be followed while using a variable in a PHP script are as follows:

- PHP variables must start with a dollar sign '$'
- PHP variables can contain strings, numbers, and arrays
- Variable names must start with a letter or an underscore ' _ '
- Variable names can only contain alpha-numeric characters and underscores and no spaces

Note: The data type for a variable in PHP need not be specified while declaring the variable. PHP automatically assigns the correct data type for a variable depending upon the value assigned to the variable.

Following example shows how to display text using a variable:

1. Open the gedit text editor.

2. Enter the code as shown in Code Snippet 5.

Session 2

Code Snippet 5:

```
<HTML>
<BODY>
<?php
$str = "My name is Samson";
echo $str;
?>
</BODY>
</HTML>
```

Figure 2.8 displays the text editor after adding the code.

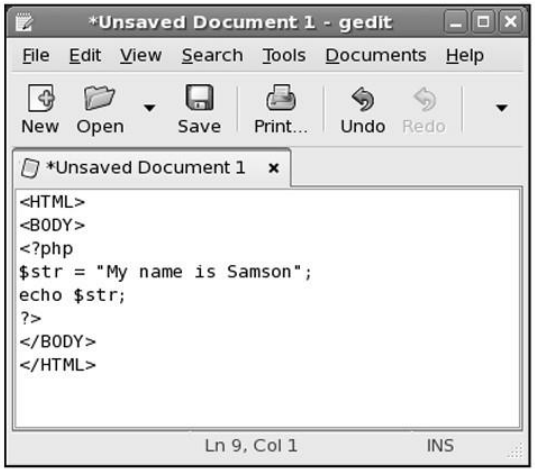


Figure 2.8: PHP Script - Displaying Text Using a Variable

In figure 2.8, the text string is assigned to a variable named **str**. The second instruction substitutes the value or content of the variable **str** to the `echo` command. The `echo` command displays the contents of the **str** variable in the output.

To execute the script,

1. Save the file as, `stringname.php` in the `/usr/local/apache2/htdocs` directory.

2. Open the Mozilla Firefox Web browser.

3. Type `http://localhost/stringname.php` in the Address bar and press `Enter`. Figure 2.9 displays the output.

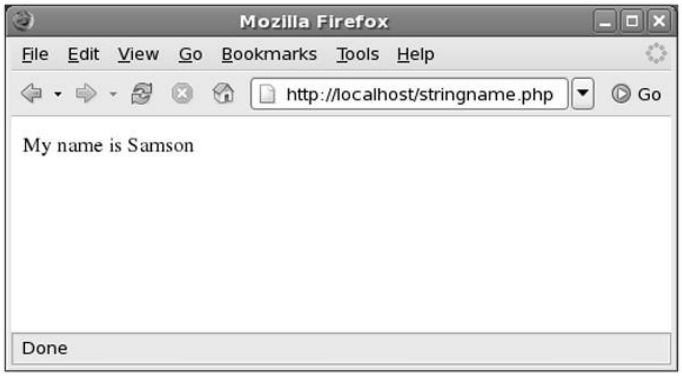


Figure 2.9: Displaying Text Using a Variable

2.8 Using HTTP Headers in PHP

In PHP, the `header()` function is used to generate the HTTP headers. The `header()` function sends the HTTP commands to the server through HTTP protocols. After the execution of `header()` function, it displays a blank line showing that the header information is complete.

To use a `header()` function, the syntax is as follows:

```
void header( string string [,bool replace [,int http_response_code]] )
```

where,

- `string` - is a required parameter and specifies the header string to be sent
- `replace` - is an optional parameter. Indicates whether the header should replace the previous or add a second header
- `http_response_code` - is an optional parameter. Forces the HTTP response code to the specified value

Code Snippet 6 displays the use of an authentication header.

Code Snippet 6:

```
<?php
header('WWW-Authenticate: Negotiate');
?>
```

Authentication helps to identify if a client is allowed to access to a resource. It is a means of negotiating access to a secure resource.

The initial request from a client will not contain any authentication information. An HTTP server application can deny the request indicating that authentication is required. The server application then sends WWW-Authentication headers with the supported authentication schemes. The commonly used authentication schemes are as follows:

- **HTTP Basic Authentication**: Sends an encoded string that contains a user name and password for the client.
- **HTTP Digest Authentication**: Is a challenge-response scheme. The server sends a data string to the client as a challenge. The client responds with a user name and password, among other additional information.
- **NTLM (defined by Microsoft)**: Is a challenge-response scheme that uses Windows credentials to transform the challenge data instead of sending the unencoded user name and password details. This scheme requires multiple exchanges between the client and server.

Negotiate (defined by Microsoft) has the following protocols:

- Kerberos
- NT LAN Manager (NTLM)

The Negotiate scheme selects between Kerberos and NTLM depending on their availability.

2.8.1 The header( ) Function with replace Option

The `replace` option in the `header()` function specifies to replace the previous header or add a second header to the document. The existing header is replaced with a new header if the `replace` option is not specified. If the `replace` option is `false` then new headers will be added to the document.

Concepts

To use the `header()` function with `replace` option, the syntax is as follows:

Syntax:

```
void header('string string', boolean replace)
```

where,

- `string` - defines the authentication parameters
- `replace` - substitutes the existing header or adds new headers to the document. The default value is set to `true`, so all similar headers are replaced.

Code Snippet 7 displays addition of multiple headers to the document.

Code Snippet 7:

```
<?php
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM', false);
```

where,

- `WWW-Authenticate` - specifies the authentication string
- `NTLM` - specifies a challenge-response authentication mechanism, where the client authenticate their identities with the server without sending a password
- `false` - defines the parameter of the `replace` option

2.8.2 The header() Function with http_response_code Option

The `http_response_code` option displays the response of the Web server for a request. The request can include the status or the location of the client.

To use `header()` function with `http_response_code` option, the syntax is as follows:

Syntax:

```
void header( string string, boolean replace, integer http_response_code)
```

where,

- `string` - defines the authentication parameters
- `replace` - indicates whether previous defined headers need to be replaced or not
- `http _ response _ code` - forces the HTTP response code to the specified value

There are different HTTP response status codes. These are three digit codes that determine the status of a response. The status codes are classified as follows:

- 1xx Informational codes
- 2xx Success codes
- 3xx Redirection codes
- 4xx Client Error codes
- 5xx Server Error codes

Code Snippet 8 displays a PHP script to redirect the user from one Web page or Uniform Resource Locator (URL) to another Web site.

Code Snippet 8:

```
header("Location: http://google.com");
```

The code must be included on the page that is to be redirected to the new location. `Location` is a type of HTTP header that redirects the browser to the specified URL. The header `Location` by default sends a 302 redirection status code to the browser unless you specifically send a different code to the browser. The status code 302 stands for 'Found'.

Code Snippet 9 displays a PHP script with an HTTP response code.

Code Snippet 9:

```
header("Location: http://google.com", true, 303);
```

Session 2

where,

- `Location` - is an HTTP header that redirects the browser to the specified URL.
- `True` - defines the parameter of the `replace` option.
- `303` - is a redirection response code which stands for 'See Other' implying that the resource you are looking for can be found under a different URL.

Session 2

Summary

- Before beginning installation of PHP, a Web server and database must be installed on the system.
- Any older version of PHP existing on the system must be uninstalled before installing PHP 7.x.
- After downloading the necessary files, you need to extract them, install the pre-requisite packages, and then install PHP.
- A PHP script starts with <?php.. ?> tag. These scripts are embedded within HTML tags.
- A HTTP message or protocol is divided into three parts, the request or response line, the HTTP header, and the body of the protocol.
- A PHP script starts with <?php tag and ends with the ?> tag. These scripts are embedded in the HTML tags.
- In PHP, the header() function is used to generate the HTTP headers.

Session 2

Check Your Progress

1. Which of the following options contain the correct sequence of steps to be followed before installing PHP 7.0.4?

   a. Web server is installed
      Database is installed
      Older version of PHP is uninstalled

   b. Web server is installed
      Database is installed

   c. Web server is installed
      Older version of PHP is uninstalled

   d. Older version of PHP uninstalled

2. Which of the following commands is used to verify the version of PHP?

   a. `echo PHPVERSION;`

   b. `echo VERSION_OF_PHP`

   c. `echo PHP_VERSION`

   d. `echo VERSION_PHP`

3. Which of the following is required to run PHP scripts?

   a. PHP, Web server, and Web browser

   b. Database and Web server

   c. Only Web browser

   d. Web server and database

Session 2

Installing and Configuring PHP 7

Answers

1. a
2. c
3. a

Session 3

New Features of PHP 7

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the `intdiv()` function.*
- *Explain spaceship operator.*
- *Explain null coalescing operator.*
- *Describe Scalar Type Declarations.*
- *Explain Uniform Variable Syntax in PHP programs.*
- *Describe `use` operator.*
- *Explain closure call methods in the PHP programs.*
- *Explain Generator delegation via `yield from`.*
- *Explain Levels parameter of the `dirname()` function.*

3.1 New Features of PHP 7 and their Usage

PHP 7, one of the most popular programming languages used by developers, was launched in 2015. It is an upgrade over the earlier version PHP 5.

This session explains the basic new features such as `intdiv ()`, `spaceship` operator (<=>), the null coalescing operator (`??`), and the uniform variable syntax. This session also explains the miscellaneous new features such as `use` operator, closure `call` method, Generator return expression, Generator delegation via `yield from`, and the `Levels` parameter of `dirname()`.

3.1.1 Integer Division

Prior to PHP 7, the division operator (/) was used for division operation. However, this operator always returns a `float` value. A roundabout method is used for integer division shown as follows:

```
<?php
 $x = 10;
 $y = 2;
 $z = (int) ($x / $y);
 echo $z;
?>
```

Session 3

If `$x` and `$y` are integer type variables, there would not be any issue in the division operator along with type casting (`int`). However, if `$x` and `$y` contain float values, then using division operator may not yield accurate results as some of the precision is lost. In order to avoid such issues of integer division, a new function has been introduced in PHP 7 that performs integer division. It accepts two parameters, where the first parameter is the dividend and the second is the divisor. This is more convenient for integer divisions.

The syntax for integer division function is:

```
intdiv(m, n)
```

where,

`m` is the dividend and `n` is the divisor.

This means that m will be divided by n, giving integer results.

Code Snippet 1 demonstrates the use of the `intdiv()` function and the behavior of operator '/'.

Code Snippet 1:

```
<HTML>
<BODY>
<?php
 echo 8/3, "\n";
 echo intdiv(8, 3), "\n";
?>
</BODY>
</HTML>
```

When the code in Code Snippet 1 is executed, the first `echo` statement displays 2.6666666666667, because the expression using division operator (/) returns a `float` value.

The second `echo` statement displays 2, because the expression is using a function `intdiv()` which returns integer value after division.

Figure 3.1 displays the output of Code Snippet 1.

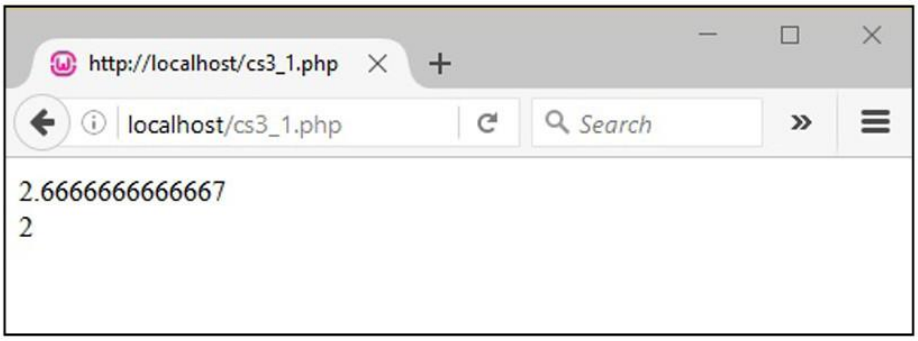


Figure 3.1: Output of Code Snippet 1

3.1.2 Spaceship Operator (<=>)

Spaceship operator is a single comparison operator, which compares operands against three rules, namely greater than, lesser than, or equal to. It combines more than one rule, therefore, it is also known as the combined comparison operator. Consider an example.

```
($op1 < $op2) ? -1 : (($op1 > $op2) ? 1 : 0);
```

This would be evaluated to -1, if `$op1` is less than `$op2`. On the other hand, it is evaluated to 0, if $op1 and $op2 are equal. It may also return 1, if `$op1` is greater than `$op2`. If you want to sort a large array and use this expression, the performance of the program suffers.

'Spaceship operator' is newly introduced in PHP 7 in order to improve the speed of program execution for sorting. This operator uses three individual operators: less than (<), equal to (=), and greater than (>). The sequence of operation starts from the 'less than' operator. If the value on the left of the operator is less than the value on the right of the operator, then it returns -1. Otherwise, it tests if the value on the left is equal to the value on the right of the operator. In that case, it returns 0 (zero). Then, it proceeds to the final evaluation to check if the value on the left of the operator is greater than the value on the right of the operator. If it is greater, it returns 1. This is mainly used to make chained comparisons more appropriate. The behavior of this operator is similar to `strcmp()` or `version_compare()`. The most common usage for this operator is in sorting.

Consider the following expression:

```
$x <=> $y
```

Session 3

This expression is evaluated to:

- -1 if `$x` is smaller than `$y`
- 0 if `$x` is equal to `$y`
- 1 if `$x` is greater than `$y`

Hands-On Exercise: Using Spaceship Operators

Code Snippet 2 demonstrates usage of spaceship operator on numbers. Usually, numbers are compared by their size.

Code Snippet 2:

```
<HTML>
<BODY>
<?php
$x = 1;
$y = 2;
echo $x.'<=>'.$y,' Returns ', $x <=> $y;
// This will output -1
echo '</br>';
$x = 10;
$y = 10;
echo $x.'<=>'.$y,' Returns ',$x <=> $y;
// This will output 0
echo '</br>';
$x = 10;
$y = 5;
echo $x.'<=>'.$y,' Returns ', $x <=> $y;
// This will output 1
?>
</BODY>
</HTML>
```

In Code Snippet 2, the program defines two variables `$x` and `$y`. These variables are assigned values 1 and 2 respectively. The spaceship operator is used between `$x` and `$y` which results in -1. Next, `$x` and `$y` have been assigned values 10 and 10 respectively. Again an `echo` statement is used to display the output of `$x <=> $y`, which results in 1. Finally, `$x` and `$y` have been assigned values 10 and 5 respectively. Then, the spaceship operator is used once again between `$x` and `$y`, which results in an output of 1.

The output of Code Snippet 2 is shown in figure 3.2.

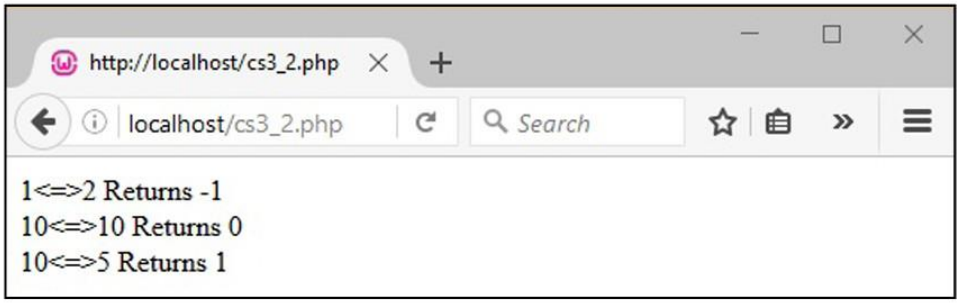


Figure 3.2: Output of Code Snippet 2

Strings are compared lexically, that is, by their alphabetical order. Code Snippet 3 uses spaceship operator in strings.

Code Snippet 3:

```
<HTML>
<BODY>
<?php
$x = "Cat";
$y = "Dog";
echo $x.'<=>'.$y,' // Returns ', $x <=> $y;
// This will output -1 because Cat is less than Dog.
 echo '</br>';
 $x = "PHP";
 $y = "PHP";
 echo $x.'<=>'.$y,' // Returns ',$x <=> $y;
// This will output 0 because both strings have same value.
echo '</br>';
 $x = "COMPUTE";
 $y = "APPLE";
 echo $x.'<=>'.$y,' // Returns ',$x <=> $y;
// This will output 1 because "COMPUTE" is greater than APPLE.
 echo '</br>';
?>
</BODY>
</HTML>
```

Session 3

Concepts

In Code Snippet 3, the variable $x and $y are assigned with string values, “Cat” and “Dog”. The statement with spaceship operator compares the two strings, “Cat” and “Dog”. The result of the comparison is -1 because the value of C in “Cat” is less than the value of D in “Dog”.

Next, the variables `$x` and `$y` are assigned with values “PHP” and “PHP”. The statement that uses the spaceship operator results in 0 because the two strings are same. Next, the variable `$x` is assigned with a value “COMPUTE” and `$y` is assigned with a value “APPLE”. The statement following this, which makes use of spaceship operator, results in 1. This is because the comparison between two strings occurs and “C” in “COMPUTE” is bigger than “A” in “APPLE”.

The output of Code Snippet 3 is shown in figure 3.3.

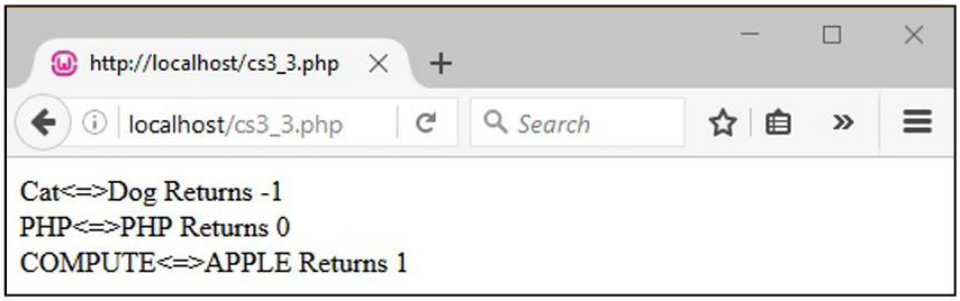


Figure 3.3: Output of Code Snippet 3

In case of arrays, corresponding array elements are compared by their size with one another. Code Snippet 4 uses spaceship operator on arrays.

Code Snippet 4:

```
<HTML>
<BODY>
<?php
$x = array();
$y = array();
echo 'array()'.'<=>'.'array()'.' Returns ', $x <=> $y;
// This will output 0
echo '</br>';
$m = array(1,2, 3);
$n = array(1,2, 3);
$p = array(1,2, 1);
$q = array(1,2, 4);
echo 'array(1,2,3)'.'<=>'.'array(1,2,3)'.' Returns ',$m <=> $n;
// This will output 0
```

```
echo '</br>';
echo 'array(1,2,3)'.'<=>'.'array()'.' Returns ', $m <=> $x;
// This will output 1
echo '</br>';
echo 'array(1,2,3)'.'<=>'.'array(1,2,4)'.' Returns ',$m <=> $q;
// This will output -1
?>
</BODY>
</HTML>
```

In the code, two arrays are declared without any elements while two other arrays are declared with values. They are compared with the spaceship operator.

The output of Code Snippet 4 is shown in figure 3.4.

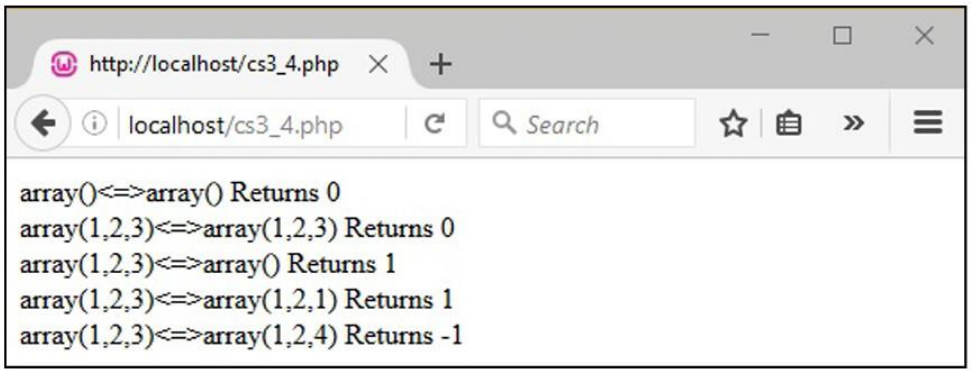


Figure 3.4: Output of Code Snippet 4

3.1.3 Null Coalescing Operator (??)

The null coalescing operator is a binary operator, which is part of the syntax for a basic conditional expression. The symbol for null coalescing operator is `??`. This operator checks strictly for a NULL value or a non-existent item such as variable, array, index, or property. This operator acts in the same way as the PHP function `isset()`. However, this operator basically improves the performance.

The `??` operator returns the result of its first operand if it exists and is NOT NULL, otherwise, it returns second operand.

Session 3

Concepts

Consider the following statement:

```
$name = $first_Name ?? "Guest";
```

If the variable `$first_Name` has some value assigned to it and is NOT NULL, then the statement assigns the value of `$first_Name` to `$name`. Else, it will assign the value "Guest" to the `$name`.

Code Snippet 5 demonstrates the use of the `??` operator. In this program, the variable `$first_name` is not assigned with any value and takes NULL value.

Code Snippet 5:

```
<HTML>
<BODY>
<?php
$name = $first_name ?? "Guest";
echo $name;
?>
</BODY>
</HTML>
```

In Code Snippet 5, `$first_name` is not assigned any value; hence, it stores NULL value. The expression using (`??`) null coalescing operator results in "Guest" because the value of `$first_name` is NULL. Hence, the variable `$name` is assigned the value "Guest".

The output of Code Snippet 5 is shown in figure 3.5.

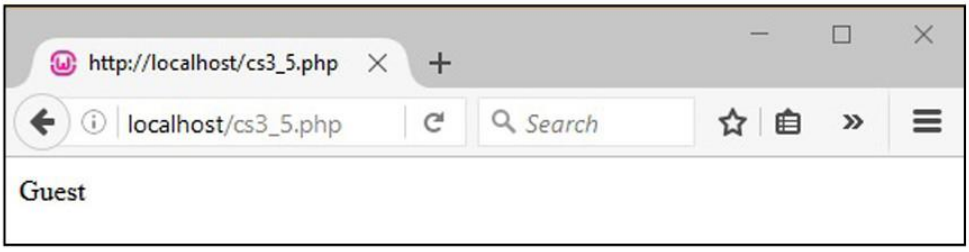


Figure 3.5: Output of Code Snippet 5

3.1.4 Scalar Type Declarations

Type Declaration involves explicitly specifying the data type of the variable, instead of allowing PHP to set this automatically. PHP is normally considered as weak type language, meaning that PHP does not require you to declare data types. In PHP 7, scalar type hints have been added, specifically `int`, `float`, `string`, and `bool`. By default, the scalar type declarations are not strict. This means, the programmer can change original type to match the type specified by the type declaration. PHP 7 allows enabling strict mode on a file by file basis.

By adding scalar type hints and enabling strict requirements, PHP program development can be made more precise.

3.1.5 Uniform Variable Syntax

Prior to PHP 7, it was not possible to use multiple operators in a given expression. The new 'uniform variable' syntax allows combinations of operators that were previously not allowed while old operations can be achieved in new code. The new syntax always follows a left-to-right evaluation order. The uniform variable syntax allows backward compatibility in some cases where old evaluation is used. For instance, calling a function from inside another function was not possible in earlier versions.

With the uniform variable syntax, all these inconsistencies are fixed and while it is not a backward-compatible change, it is easy to change your source code to be both backward-compatible and forward-compatible.

With the new syntax, there would not be any restrictions on nesting of dereferencing operations.

Hands-On Exercise: Using Uniform Variable Syntax

Code Snippet 6 demonstrates the use of the uniform variable syntax.

Code Snippet 6:

```
<?php
 function e() {
 echo "This is e() \n";
 };
 function f() {
 echo "This is f() \n";
 return e;
 };
 function g() {
 echo "This is g()\n";
return f;
```

Session 3

Concepts

```
 };
 g();
 echo "***********\n";
 g()();
 echo "***********\n";
 g()()();
?>
```

In Code Snippet 6, three functions have been defined. The function `f()` displays a message and calls function `e()`. The function `g()` also displays a message and calls function `f()`.

The next three statements call function `g()`. The first statement calls `g()`, which will only execute one statement inside the function. The next call of function `g()()` executes the function `g()` as well as calls the function `()` and executes it. The third call of function `g()()()` executes the function `g()` and executes the function `f()` which is called within function `g()`. Then the function `f()` is executed, which in turn calls another function `e()`. Finally, function `e()` is executed.

The output of the Code Snippet 6 is as follows:

```
This is g()

***********

This is g()

This is f()

***********

This is g()

This is f()

This is e()
```

3.2 Miscellaneous Features

There are a few other features that are newly introduced with PHP 7.

3.2.1 Use Operator

The ability to refer to an externally fully qualified name with an alias or importing is an important feature of namespaces. This is very similar to the Unix-based file systems to create symbolic links to a file or a directory. All versions prior to PHP 7 that support namespace support three kinds of aliasing or importing. They are:

- Aliasing a class name
- Aliasing an interface name
- Aliasing a namespace name

Aliasing is achieved by using the `use` operator. From PHP 7 onwards, classes, functions, and constants being imported from the same namespace can be grouped together in a single 'use' statement.

The `use` keyword allows grouping multiple declarations in one statement. Code Snippet 7 shows a portion of code where the `use` operator is being used.

Code Snippet 7:

```
<?php
 namespace aptech;
 class Boston {
function say()
{
echo "Boston\n";
}
}
class NewYork {
function say()
{
echo "NewYork\n";
}
}
function foo1()
```

Session 3


```
{
echo "This is foo1()\n";
}
function foo2()
{
echo "This is foo2()\n";
}
?>
```

Code Snippet 7 declares two classes, `Boston` and `NewYork`. It also declares functions `foo1()` and `foo2()`. A member function `say()`, which is the part of classes, `Boston` and `NewYork`, will display a message.

The program can be stored in a file called myfile.php. This file is included in ugroup.php, as shown in Code Snippet 8.

Code Snippet 8:

```
<?php
 include 'myfile.php';
 use aptech\{Boston, NewYork};
 use function aptech\{foo1, foo2};
  $d = new Boston();
 $d->say();
  $n = new NewYork();
 $n->say();
  foo1();
 foo2();
 ?>
```

The code can be saved in a file called ugroup.php.

Code Snippet 8 includes the myfile.php file, where namespace is defined as `aptech` and two classes as `Boston` and `NewYork`. A function is defined in each class by name `say()`. In this file, two individual functions are also defined. The code is stored in a file called ugroup.php. The code uses `use` for multiple declarations. `$d` and `$n` are two new objects where `$d` belongs to class Boston and `$n` belongs to class `NewYork`. The next two statements are calling the members of the class which is a function `say()`.

The two functions, which are defined in myfile.php are also being called in the ugroup.php program and these functions are executed and corresponding messages are displayed, as shown in figure 3.6.

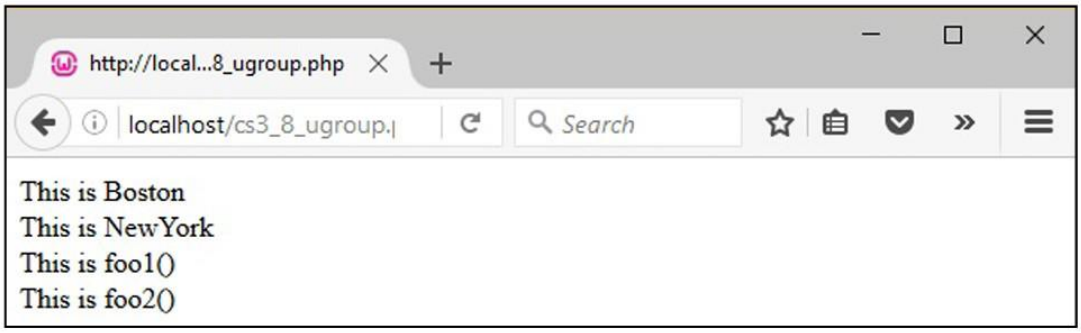


Figure 3.6: Output of Code Snippet 8

3.2.2 Closure call() Method

A closure is a record storing a function together with an environment. A closure, unlike a function, allows the function to access those captured variables, through close references to them. A closure is good when you want a local function that is only used for a specific purpose.

With the addition of `$this`, Closure gained two methods, the instance method `Closure->bindTo()` and the static method `Closure::bind()`. Both of these functions do the same task but the first is called on the closure instance itself, while the static version must be passed to the closure itself as the first argument. They both then take two arguments: the first is an object that `$this` will then point to and the second is the scope from which to access `$this`.

Code Snippet 9 creates a class named Greetings. It has a private variable $word and it is assigned with a value 'Hello'. An anonymous function is defined with a parameter $whom. This function accesses the member of the Greetings class, which is $word and prints the value that is concatenated with the argument of the function.

Code Snippet 9:

```
<?php
 class Greetings {
 private $word = "Hello";
 }
 $closure = function($whom) {
 echo "$this->word $whom\n";
 };
```

Concepts

```
  $obj = new Greetings();

 $closure->call($obj, 'John');
 $closure->call($obj, 'Kevin');
?>
```

The output of Code Snippet 9 is shown in figure 3.7.

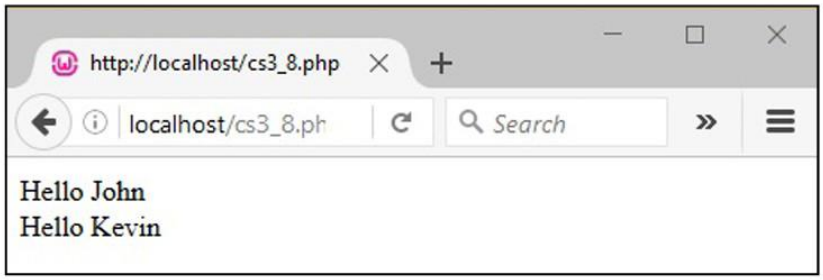


Figure 3.7: Output of Code Snippet 9

An object `$obj` is being created from class `Greetings` and function is being called with method having two arguments namely, `$obj` and actual value 'John'. It will call the function and return 'Hello John'. Similarly, the second call is done with closure, which will get the value 'Hello Kevin'.

3.2.3 Generator Return Expressions

A generator function is a means to implement an iterator and is similar to a regular function. It allows programmers to write code to perform tasks such as calculating and returning values that are in a loop. It was first introduced in PHP 5.5.

Generator return expressions is a new feature in PHP 7 that builds upon the generator feature in PHP 5.5.

A return statement can be used within a generator to enable a final expression to be returned. This value can be retrieved using the `getReturn()` method, which may only be used once the generator has finished yielding values.

Code Snippet 10 uses a random generator function. It will yield the value generated from the `rand()` function which is between 1 and 10. The `foreach` statement prints the value randomly that is generated from the `random_generator` function. Finally, `getReturn` will return -1.

Session 3

Code Snippet 10:

```
<?php
srand();
function random_numbers($k) {
 for ($i=0; $i<$k; $i++) {
 $r = rand(1, 10);
  yield $r;
  }
 return -1;
 }
 $rns = random_numbers(10);
 foreach ($rns as $r) {
 echo "$r";
 echo '</br>';
 }
echo $rns->getReturn() . PHP_EOL;
 ?>
```

In Code Snippet 10, the generator returns random values. At the end of the iteration, the generator returns -1. The output is shown in figure 3.8.

Figure 3.8: Output of Code Snippet 10

Concepts

3.2.4 Generator Delegation via `yield from`

Generator delegation allows you to yield values from another generator, Traversable object, or array by using the `yield from` keyword. Generator delegation promotes cleaner code and reusability.

Code Snippet 11 shows the usage of generator delegation via `yield from`.

Code Snippet 11:

```
<?php
 function f1() {
 yield from f2();
 yield "f1() 1";
 yield "f1() 2";
 yield from [3, 4];
 yield "f1() 3";
 yield "f1() end";
 }
 function f2() {
 yield "f2() 1";
 yield "f2() 2";
 yield "f2() 3";
 yield "f2() end";
 }
 $f = f1();
 foreach ($f as $val) {
 echo "$val\n";
 }
?>
```

In Code Snippet 11, two functions `f1` and `f2` are defined. Function `f1` yields the values from function `f2`. The `foreach` statement will display the value from function `f1`, in turn, yielding the values from function `f2`.

Figure 3.9 shows the results of Code Snippet 11.

Figure 3.9: Result of Code Snippet 11

3.2.5 Levels Parameter of `dirname()`

The `dirname()` function returns the parent's directory path. The new levels parameter tells how many parent directories to go up.

Code Snippet 12 uses `dirname` with two arguments. The first argument is the directory name and the second argument indicates how many levels of the directory name should be returned.

Code Snippet 12:

```
<?php
$path = '/foo/bar/bat/baz';
echo dirname($path, 2);
?>
```

Code Snippet 12 displays the levels of the directory path. Here, the directory path is `'/foo/bar/bat/baz'`. The `dirname` statement returns two levels of the path, which is `'/foo/bar'`.

The `dirname()` function can now accept a second parameter to set how many levels up it will go, meaning you can avoid nesting.

Session 3

New Features of PHP 7

Concepts

The output of Code Snippet 12 is shown in figure 3.10.

Figure 3.10: Result of Code Snippet 12

Session 3

Summary

- The new intdiv() performs integer division and is considered the inverse of the modulo operator %.
- Spaceship operator, also known as combined comparison operator, returns three distinct values, -1, 0 , or +1.
- null coalescing operator ?? will return the left operand if it is not null; otherwise, it returns the right operand.
- The Generator->getReturn() method allows you to retrieve the value returned by any return statement inside a generator.
- Group 'use' declarations allow a programmer to deduplicate common prefixes in use statements and just specify the unique parts within a block {}.
- With addition of $this, closure has gained two methods, the instance method Closure->bindTo() and the static method Closure::bind().
- The dirname() function can now accept a second parameter to set how many levels up it will go.

Session 3

Concepts

Check Your Progress

1. What is the output of the following program?

```
<?php
 echo intdiv(10,3);
?>
```

a. 3

b. 3.3333333

c. 1

d. 30

2. What is the output of the following program?

```
<?php
 $x = 10;
 $y = 20;
 echo $x <=> $y;
?>
```

a. +1

b. 0

c. -1

d. 200

Check Your Progress

3. What is the output of the following program?

```
<? php
 $password = $upassword?? "Password is not valid";
 echo $password;
?>
```

   a. NULL

   b. "Password Not valid"

   c. Value of $upassword

   d. "Password is valid"

4. Which of the following statements are true about the null coalescing operator?

   a. It performs integer division

   b. It returns the left operand if it is not null

   c. It retrieves the value returned by any return statement inside the generator

5. When does the spaceship operator return -1?

   a. If the left operand is less than right operand

   b. If the left and right operands are equal

   c. If the left operand is greater than right operand

Session 3

Answers

1. a
2. c
3. b
4. b
5. a

Onlinevarsity

INDUSTRY BEST PRACTICES

SYNC WITH THE INDUSTRY

Session 4

Form Handling in PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the use of the GET method.*
- *Explain the use of the POST method.*
- *Retrieve data from forms using the Form methods.*
- *Explain the use of hidden fields.*

4.1 Introduction

PHP supports handling of form data. PHP enables to retrieve data from a form using form methods, such as `GET` or `POST`. These methods define the data retrieval process. PHP also enables to pass data from hidden fields.

In this session, you will learn about the attributes of a form. You will also learn how to pass data from a form to the Web server using the `GET` and the `POST` method. In addition, you will learn how to retrieve data from the hidden fields in a form.

4.2 Forms

A form is a Web page that is used to pass data from a client to a server. Forms are common on the Internet. A form contains fields, where the user can enter information. This information is passed to the Web server, which processes the application data and stores the information to a database.

PHP is designed to handle HTML forms and process information entered in these forms. When a user enters information in an HTML form and submits it, the information is sent to the Web server. The Web server in turn passes the information to the PHP script engine. The script engine processes this information, manipulates it, and sends the output back to the Web browser.

4.2.1 HTML FORM tags

Forms add interactivity to a Web site and are used to pass data to a server. PHP simplifies the task of processing data generated from Web-based forms. A `<FORM>` tag is used to create an HTML form. Forms, in a Web page, are included within the `<FORM>` and `</FORM>` tags. The `FORM` elements have to be included within the `FORM` tags.

Session 4

There are several attributes associated with HTML forms, of which the required attributes are as follows:

- **Action** - It is a form-processing agent and defines the URI where the form data is sent after it has been submitted. The URI can be an HTML page, a PHP page, or any server-side script.

- **Method** - It defines the protocol that will be used to submit the form data set. There are two method protocols, `GET` and `POST`. By default, forms are submitted using the `GET` method.

4.3 Using Methods

The `method` attribute specifies the HTTP methods used to send the form data set to the processing agent. `GET` and `POST` are the two types of method protocols.

4.3.1 Using the 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 form data is a stream of name/value pairs separated by an ampersand (`&`) sign. 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.

An input variable will have the following structure:

```
Name=value
```

Multiple input variables are grouped as follows:

```
Name1=value1&Name2=value2&Name3=value3
```

Consider an example that shows how the `GET` method passes the variables to the processing agent by appending them at the end of the URL.

```
Name=john&age=18
```

The query string is appended with the following URL:

```
http://www.information.com/text.php?Name=john&age=18
```

The example shows multiple name/value pairs separated by the `&` sign.

Session 4

The browser automatically appends the encoded information to the URL with a ? symbol as a separator and this new URL is sent to the processing agent.

The GET method restricts form data set values to American Standard Code for Information Interchange (ASCII) characters. It is used to transfer limited amount of information, as the length of the query string is restricted to 255 characters.

4.3.2 Using the POST Method

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.

The information sent by the POST method is not encrypted. As a result, hackers can easily access it. In order to secure the information, it is necessary to establish a secure server connection.

In the POST method also, the form information is passed through variables. However, the variables are not appended to the URL.

4.3.3 Difference in the GET and POST Method

The GET and POST methods work almost identically. The main difference is the method of transmitting information to the server. Table 4.1 lists the differences between the GET and the POST methods.

GET	POST
Encodes the form data as a stream of name/value pairs and appends it in the URL making it visible in the browser	Sends the encoded form data through the body of an HTTP request
Form submissions can be bookmarked	Form submissions cannot be bookmarked
Is less secure as the information is displayed in the URL	Is more secure for transmitting passwords and other sensitive information, as the form data is embedded in the body of the HTTP request
The amount of data that can be sent is limited depending on the browser used	Does not have size limitations
This method is mainly used for displaying data such as searching, sorting, and pagination	This method is mainly used for data manipulation such as adding and editing data

Table 4.1: Differences between GET and POST Methods

Session 4

Concepts

4.4 Retrieving Data Using the GET Method

You can use the GET method to retrieve data from an HTML form.

The syntax to retrieve the data from a form using the GET method in PHP is as follows:

```
$varname = $_GET["variable"];
```

where,

varname - specifies the name of the variable in which the data is to be stored

$_GET["variable"] - specifies the name of the input variable

To create an HTML form to retrieve data using the GET method, perform the following steps:

1. Open a new file in the **gedit** text editor.
2. Enter the code as shown in Code Snippet 1.

Code Snippet 1:

```
<HTML>
<BODY>
<B>ENTER YOUR PERSONAL DETAILS</B>
<FORM METHOD=GET ACTION="Details.php">
FIRST NAME:
<INPUT NAME="n1text" TYPE="TEXT"><BR>
LAST NAME:
<INPUT NAME="n2text" TYPE="TEXT"><BR>
ADDRESS:
<TEXTAREA NAME="n3text" ROWS=1, COLUMNS=1000></TEXTAREA>
<BR>
CONTACT NO:
<INPUT NAME="n4text" TYPE="TEXT"><br>
<INPUT TYPE="SUBMIT" NAME="SUBMIT" VALUE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

Session 4

Figure 4.1 displays the text editor after adding the code.

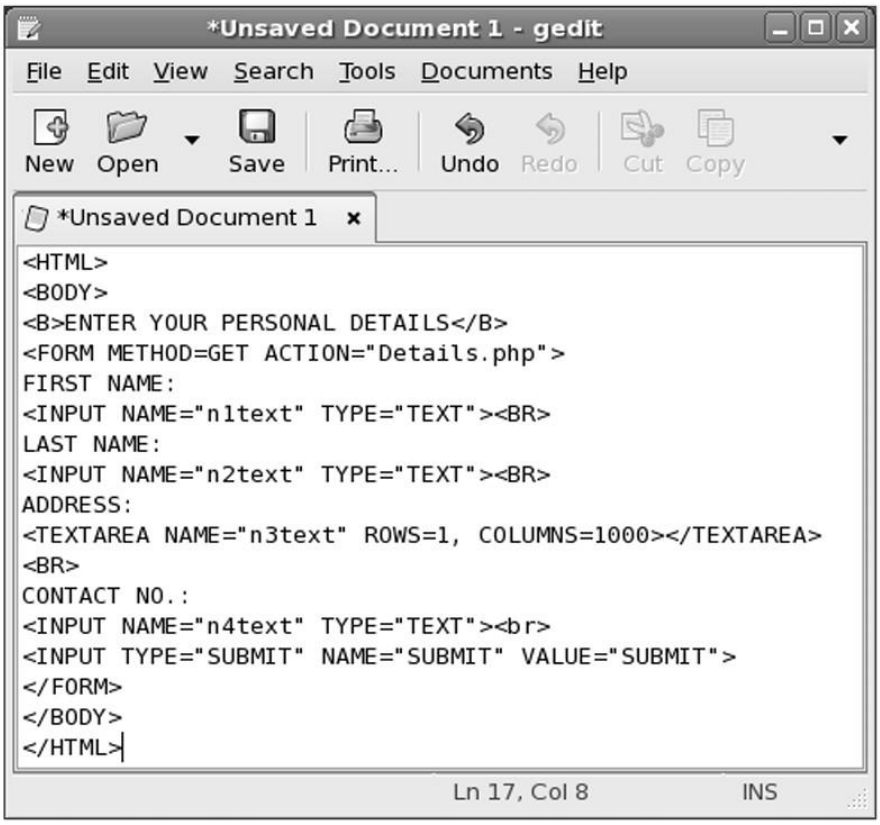


Figure 4.1: Script for Creating a Personal Details Form

3. Save the file as **`Details.html`** in the `/usr/local/apache2/htdocs` directory.

To create a PHP script, to retrieve and process the data entered in the HTML form, perform the following steps:

1. Open a new file in the **gedit** text editor.

2. Enter the code as shown in Code Snippet 2.

Code Snippet 2:

```
<?php
error_reporting(0);
$A = $_GET["n1text"];
```

Session 4


```
$B = $_GET["n2text"];
$C = $_GET["n3text"];
$D = $_GET["n4text"];
echo "YOUR PERSONAL DETAILS";
echo "<BR><BR>";
echo "FIRST NAME: $A <BR>";
echo "LAST NAME: $B <BR>";
echo "ADDRESS: $C <BR>";
echo "CONTACT NO.: $D <BR>";
?>
```

Figure 4.2 displays the text editor after entering the code.

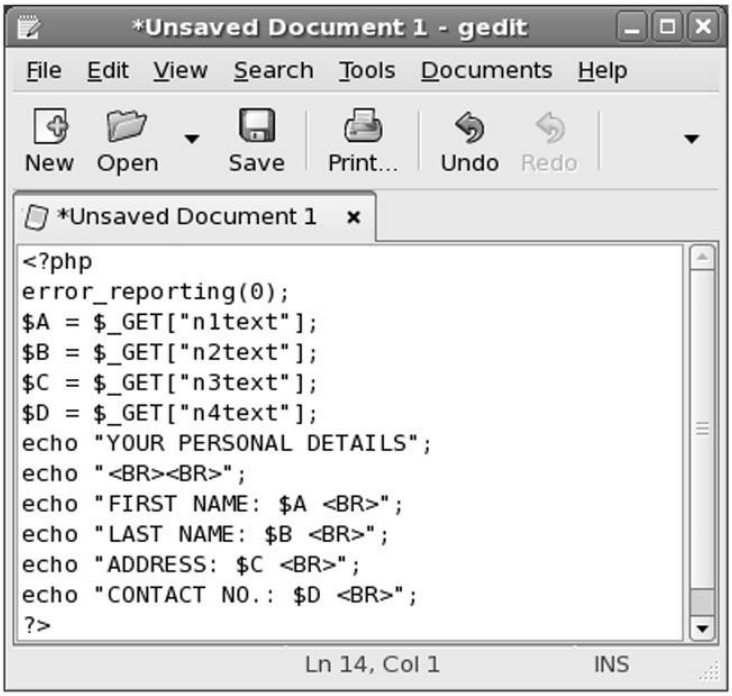


Figure 4.2: PHP Script Using the GET Method to Retrieve Data

3. Save the file as **`Details.php`** in the `/usr/local/apache2/htdocs` directory.

4. Open the Mozilla Firefox Web browser.

5. Enter `http://localhost/Details.html` in the Address bar and press **Enter**.

Figure 4.3 displays the Personal Details Form.

Figure 4.3: Personal Details Form

6. Enter **John** in the FIRST NAME box.

7. Enter **Simons** in the LAST NAME box.

8. Enter **32, Park Street, New Jersey** in the ADDRESS box.

9. Enter **45450** in the CONTACT NO. box.

10. Click **SUBMIT**.

Concepts

Figure 4.4 displays the output of the script.

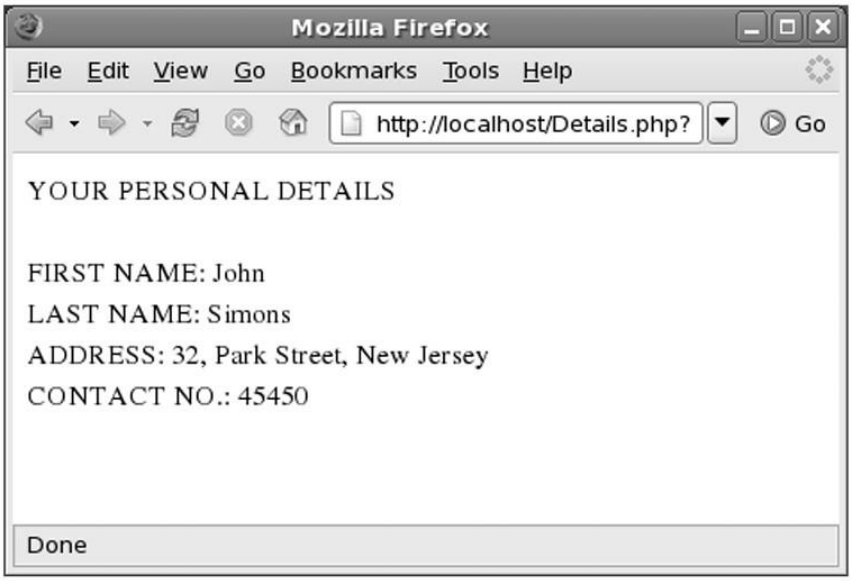


Figure 4.4: Displaying the Personal Details

4.5 Retrieving Data Using the POST Method

You can also use the `POST` method to retrieve data from an HTML form. The `POST` method transmits data through the body of an HTTP request.

The syntax to retrieve the data using the `POST` method in PHP is as follows:

```
$varname = $_POST["variable"];
```

where,

`varname` - specifies the name of the variable in which the data is to be stored

`$_POST["variable"]` - specifies the name of the input variable

Session 4

To retrieve form data in a PHP script using the POST method, perform the following steps:

1. Open a new file in the **gedit** text editor.

2. Enter the code as shown in Code Snippet 3.

Code Snippet 3:

```
<HTML>
<HEAD>
<TITLE>Employee Details</TITLE>
</HEAD>
<BODY>
<H4>Enter your details</H4>
<FORM METHOD=POST ACTION="EMP_DETAILS.php">
<TABLE>
<TR>
<TD>Employee ID</TD>
<TD><INPUT TYPE="text" NAME="empid"></TD>
</TR>
<TR>
<TD>Name</TD>
<TD><INPUT TYPE="text" NAME="Name"></TD>
</TR>
<TR>
<TD>Department</TD>
<TD>
<INPUT TYPE="radio" NAME="dept" VALUE="Finance">Finance
<INPUT TYPE="radio" NAME="dept" VALUE="Marketing">Marketing
<INPUT TYPE="radio" NAME="dept" VALUE="IT">IT
</TD>
</TR>
<TR>
<TD>Email</TD>
<TD><INPUT TYPE="text" NAME="email"></TD>
</TR>
</TABLE>
<BR>
<TD><INPUT TYPE="submit" VALUE="SUBMIT"></TD>
```

Session 4

Concepts

```
</FORM>
</BODY>
</HTML>
```

Figure 4.5 displays the text editor after adding the code.

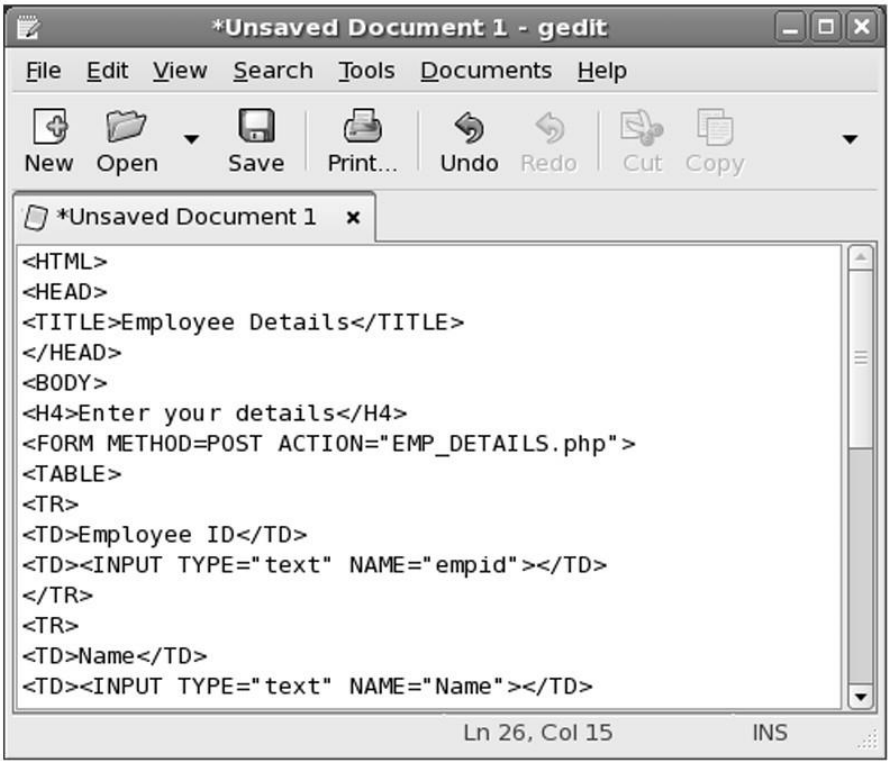


Figure 4.5: Script for Creating the Employee Details Form

3. Save the file as **`EMP _ DETAILS.html`** in the `/usr/local/apache2/htdocs` directory.

4. Open a new file in the **gedit** text editor.

Session 4

5. Enter the code as shown in Code Snippet 4.

Code Snippet 4:

```
<?php
error_reporting(0);
$A=$_POST["empid"];
$B=$_POST["Name"];
$C=$_POST["dept"];
$D=$_POST["email"];
echo "YOUR PERSONAL DETAILS";
echo "<BR><BR>";
echo "EMPID: $A <BR>";
echo "NAME: $B <BR>";
echo "DEPARTMENT NAME: $C <BR>";
echo "EMAIL: $D <BR>";
?>
```

Figure 4.6 displays the text editor after entering the code.

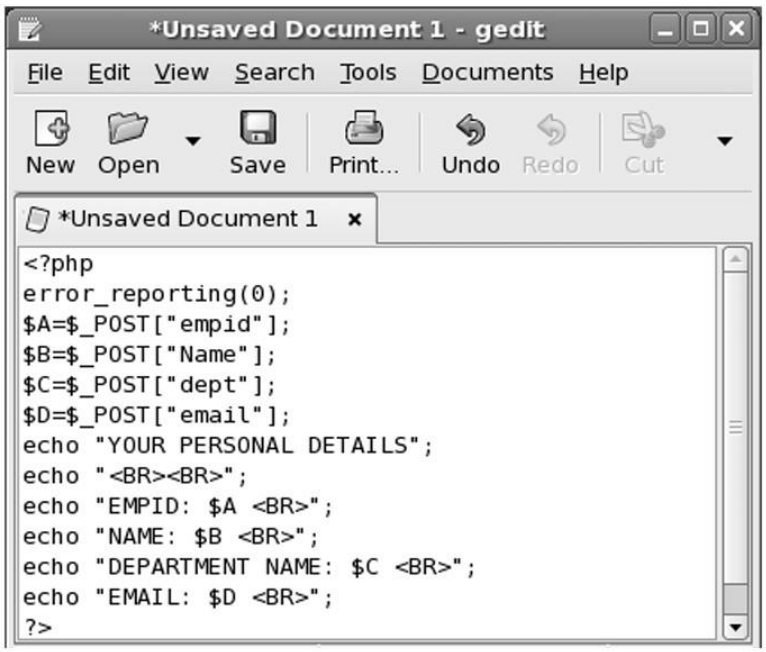


Figure 4.6: Using the POST Method to Retrieve Data

Session 4

6. Save the file as **EMP _ DETAILS.php** in the `/usr/local/apache2/htdocs` directory.

7. Open the Mozilla Firefox Web browser.

8. Enter `http://localhost/EMP _ DETAILS.html` in the Address bar and press **Enter**.

   Figure 4.7 displays the output of the HTML page.

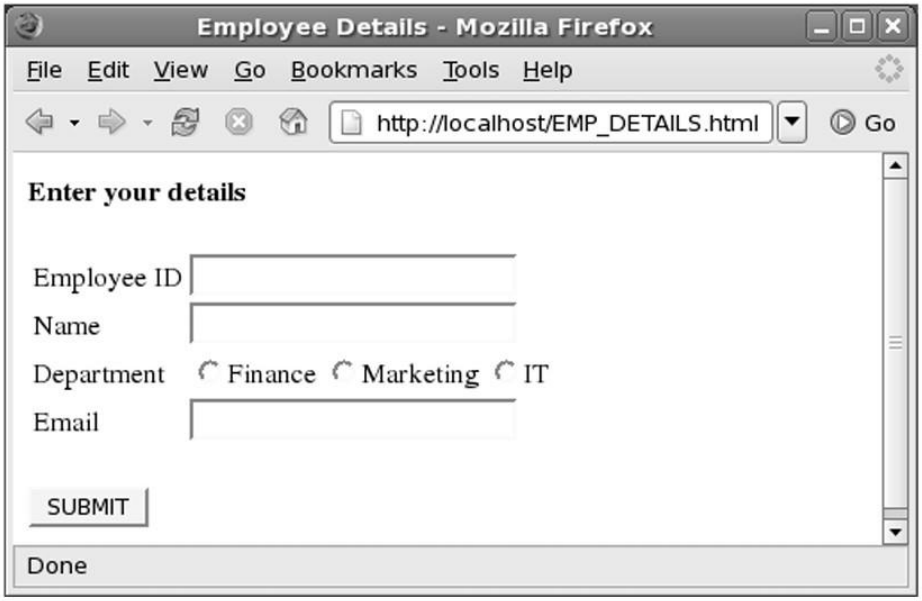


Figure 4.7: HTML Form to Accept Employee Details

9. Enter **A05** in the Employee ID textbox.

10. Enter **Peter Taylor** in the Name box.

11. Select **IT** as the Department.

12. Enter **peterA05@finfotech.com** in the Email box.

13. Click **SUBMIT**.

Figure 4.8 displays the output of the script.

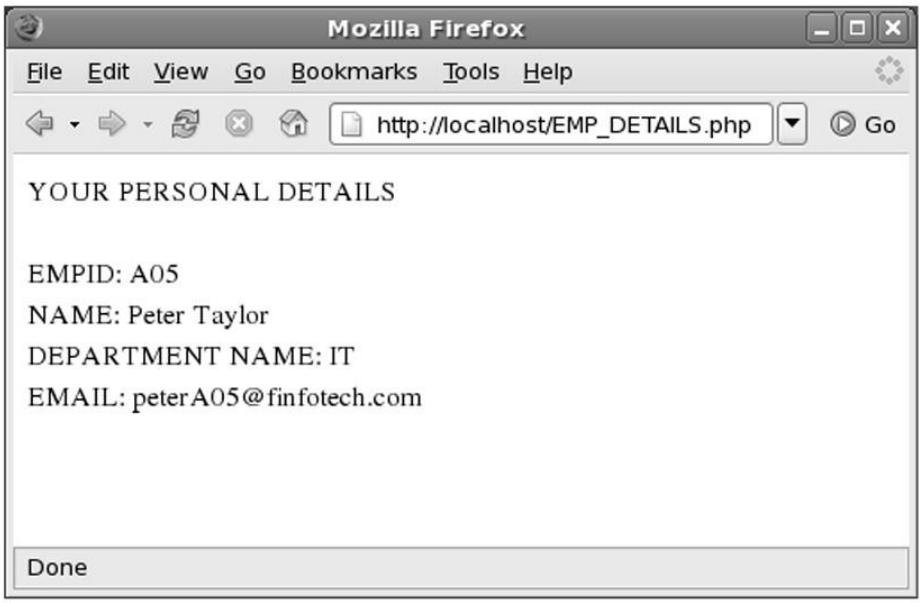


Figure 4.8: Displaying the Employee Details Using POST Method

4.6 Using Hidden Fields

A hidden field is similar to a text field. The difference is that the user cannot view the hidden field and its contents. The hidden fields in a form are embedded in the HTML source code of the form. A hidden field enables the user to pass variables with values from one form to another without requiring to re-enter the information.

The syntax to define a hidden field is as follows:

```
<INPUT TYPE=HIDDEN NAME=hidden1 VALUE="PHP MESSAGE">
```

where,

`INPUT TYPE` - specifies that the field is hidden

`NAME` - specifies the name of the hidden field

`VALUE` - specifies the value as it appears on the form

Session 4

Concepts

To use hidden fields and pass the name of the continents in a PHP script, perform the following steps:

1. Open a new file in the **gedit** text editor.

2. Enter the code as shown in Code Snippet 5.

Code Snippet 5:

```
<html>
<FORM METHOD='get' action='continent.php'>
Specify the continent:
<SELECT TYPE='LISTBOX' NAME='continent'>
<OPTION>ASIA</OPTION>
 <OPTION>AUSTRALIA</OPTION>
<OPTION>EUROPE</OPTION>
</SELECT><BR><BR>
<INPUT TYPE=HIDDEN NAME=Asia>
<INPUT TYPE=HIDDEN NAME=Australia>
<INPUT TYPE=HIDDEN NAME=Europe>
<BR><INPUT TYPE=SUBMIT>
</FORM> </html>
```

Figure 4.9 displays the text editor after entering the code.

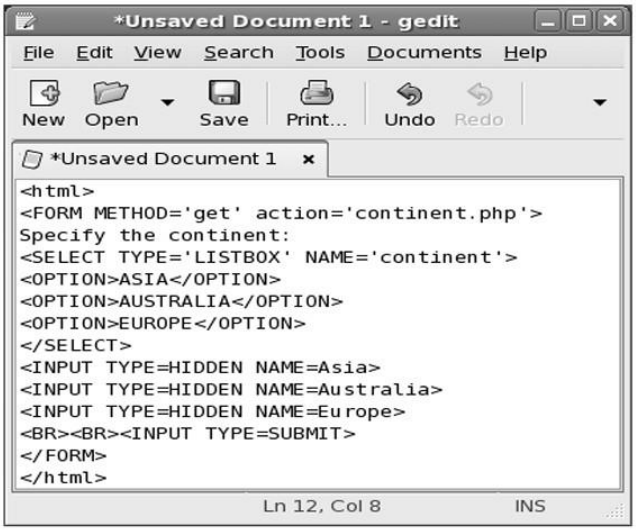


Figure 4.9: Creating a Form with Hidden Fields

Session 4

3. Save the file as **continent.html** in the /usr/local/apache2/htdocs directory.

4. Open a new file in the **gedit** text editor.

5. Enter the code as shown in Code Snippet 6.

Code Snippet 6:

```
<?php
$A=$_GET['Asia'];
$B=$_GET['Australia'];
$C=$_GET['Europe'];
$Name=$_GET['continent'];
echo "<BR>";
echo "Continents:<BR> <BR> Asia <BR> Australia <BR> Europe <BR> <BR>";
echo "The continent you have selected is: $Name";
?>
```

Figure 4.10 displays the text editor after entering the code.

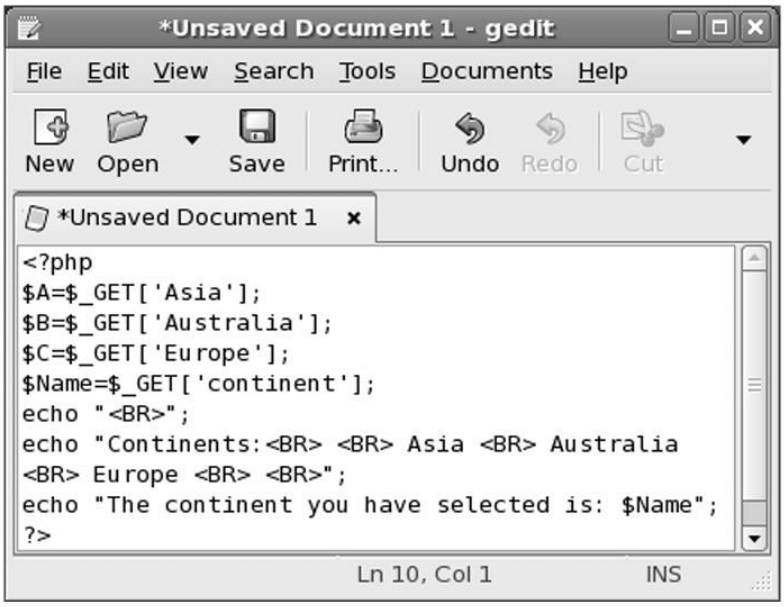


Figure 4.10: Retrieving Data from Hidden Fields

6. Save the file as **continent.php** in the /usr/local/apache2/htdocs directory.

Session 4

7. Open the Mozilla Firefox Web browser.

8. Enter `http://localhost/continent.html` in the Address bar and press **Enter**. Figure 4.11 displays the Continents Form page.

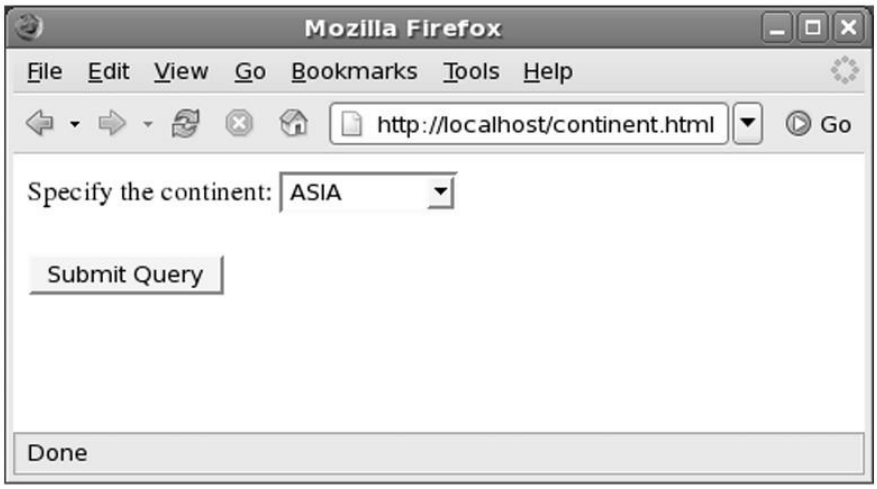


Figure 4.11: Continents Form

9. Select the required continent from the from the drop-down menu.

10. Click **Submit Query**.

Figure 4.12 displays the output of the script.

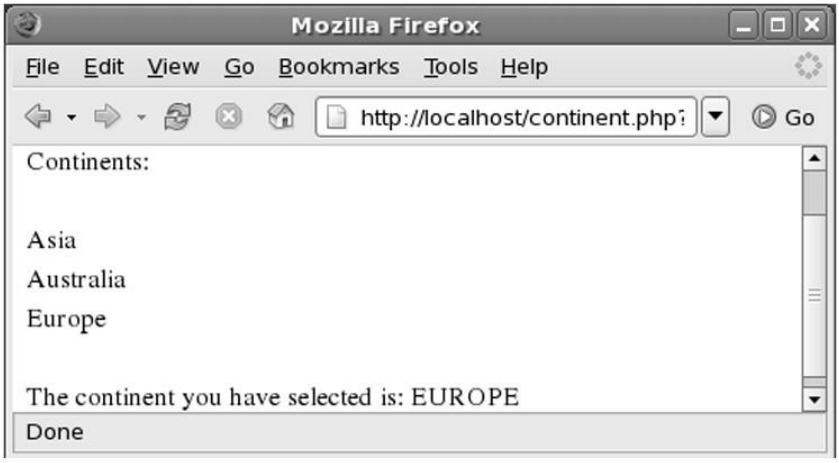


Figure 4.12: Using Hidden Fields

Session 4

Summary

- A form is a Web page that is used to pass data from a client to a server.
- PHP has a built-in support for collecting data from an HTML form.
- The attributes of a form are namely, action and method.
- The action attribute of a form specifies the URL that will process the form data and provide the feedback.
- The method attribute of the form defines the method of transmitting information to the URL.
- The GET method directs the Web browser to send the encrypted user information appended at the end of the URL, to the processing agent.
- 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.
- Hidden form fields are not visible to users and enable form developers to pass information from a form to a script or from one form to another, before being passed to a script.

Session 4

Concepts

Check Your Progress

1. Which of the following option is also known as the form-processing agent?

   a. Action

   b. Method

   c. GET

   b. POST

2. HTML stands for _____.

   a. Hyper Text Mode Language

   b. Hyper Text Markup Language

   c. Hyper Text Manual Language

   d. Hyper Text Mixed Language

3. 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

Check Your Progress

4. Which of the following is true for the POST method?

   a. Form data is sent as a URL string

   b. Does not have size limitations

   c. Form submissions can be bookmarked

   d. Form data is sent in the body of an HTTP request

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']

6. Which method sends the information in the body of the HTTP request?

   a. POST

   b. GET

   c. BODY

   d. HEADER

Session 4

Answers

1. a
2. b
3. c
4. b, d
5. c
6. a

LEARN
BETTER
@
Onlinevarsity

Session 5

Using Variables and Expressions in PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the use of identifiers.*
- *Explain the data types in PHP.*
- *Explain the use of variables and constants.*
- *Explain the scope of variables in PHP.*
- *Explain the use of HTTP environment variables.*

5.1 Introduction

A variable is a named memory location, which stores data that keeps on changing during the execution of a program. Different types of data can be stored in a variable. An expression is a combination of variables, constants, functions, and operators.

In this session, you will learn about the identifiers, data types, variables, constants, and expressions. In addition, you will also learn about the scope of variables and HTTP environment variables.

5.2 Identifiers

Identifiers are names given to various elements of a program such as variables, constants, arrays, functions, and classes. Rules to follow while naming identifiers are as follows:

- An identifier must begin with a letter
- An identifier can contain only letters (A to Z) or digits (0-9)
- An underscore (_) character can be used to add space in the identifier to make it more readable
- An identifier must not have any special characters including blank space

Some of the valid identifiers names are **`firstnum`**, **`lname`**, **`net_sal`**, **`add8num`**, and **`NewNum`**.

Table 5.1 displays some invalid identifiers.

Identifier	Invalid Because
first-num	Contains a hyphen
1num	First character is a number
first num	Contains a blank space
first&num	Contains an ampersand

Table 5.1: Invalid Identifiers

5.3 Variables and Data Types

A variable is an identifier whose value keeps changing throughout the execution of a program. A variable has a name and a data type. The name refers to the variable and the data type refers to the type of data that the variable can store. Variables are used to store user information, intermediate data such as calculated results, and values returned by the functions.

In PHP, few rules to be followed for a variable are as follows:

- Declaring a variable before assignment is not mandatory.
- A variable is automatically declared at the time of initialization and is of the same data type as the value stored in it, which can change during the execution of the program.
- All variable names in PHP are preceded by a dollar sign (`$`) and can only contain alphanumeric characters and underscores (a-z, A-Z, 0-9, and _).
- Value to a variable is assigned with the equal to (=) operator, with the variable on the left-hand side and the expression on the right-hand side.
- A variable created without any value assigned to it, takes the default value of `NULL`.
- A variable is case sensitive. A variable name, `$var _ name`, is different from a variable name, `$Var _ Name`.

Variables created within a function are local to their scope and available for the lifetime of the function.

Variables are used to store data values of different data types.

Session 5

PHP supports eight primitive data types that are categorized as follows:

5.3.1 Scalar Types

Scalar types are as follows:

- **Integer**: Stores whole numbers without decimal points. The value ranges from -2,147,483,648 to +2,147,483,647.

- **Float**: Stores floating-point numbers that are equivalent to the C compilers double data type. The common data type size is 8 bytes and the value approximately ranges from - 2.2E–308 to 1.8E+308. Floating point numbers can include a decimal point, +/- sign, and an exponential value.

- **String**: Stores a sequence of characters. A string is enclosed within single quotes or double quotes. In PHP, there is a difference between single quotes and double quotes. A single quote treats variables at its face value and does not recognize the special characters unless escaped. A double quote treats variables by expanding its value and recognizes all the special characters.

  To view how single and double quotes work in a PHP script, enter the code as shown in Code Snippet 1, in a script named **`scalar_string.php`**.

  Code Snippet 1:

```
<?php
$a = "hello\n"; //attempting multi-line output
$b = 'hello\n'; //attempting multi-line output
echo $a;
echo "<br>"; //Printing a new line in HTML output
echo $b;
?>
```

Concepts

Figure 5.1 displays the output of the script.

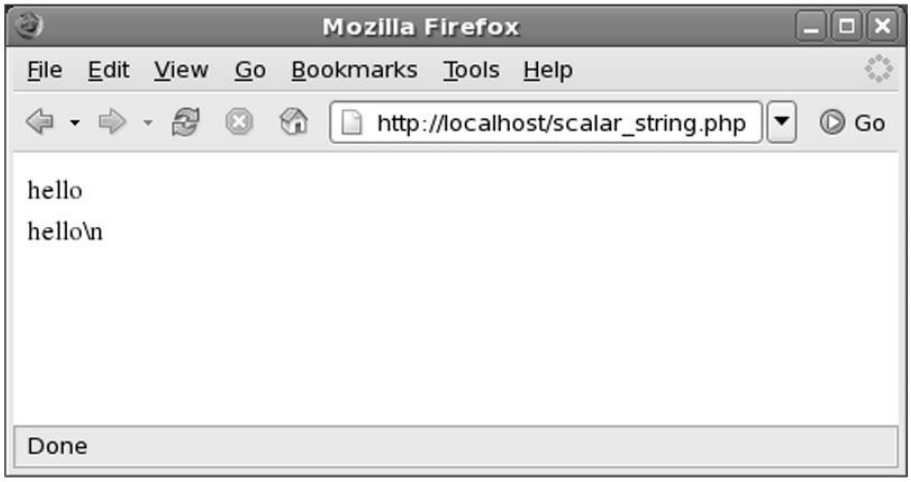


Figure 5.1: Viewing Single and Double Quotes

In the code, '`\n`' was not printed in the output of variable `$a`, but it is visible in the output of the variable `$b`. That is because when double quotes are used, the special characters used in the strings such as '`\n`' (linefeed) or a '`\t`' (horizontal tab) are processed by the PHP compiler, which does not happen when using a single quote. When double quotes are used, PHP will process escape sequence for special characters.

- **Boolean**: Stores one of the two values, True or False

5.3.2 Compound Types

Compound types are as follows:

- **Array**: Stores multiple values in one single variable. Each element in the array can be accessed using its index. In PHP arrays are classified as follows:
  - Numeric array - An array with a numeric index
  - Associative array - An array where each ID key is associated with a value
  - Multidimensional array - An array containing one or more arrays
- **Object**: Used for any object reference. An object is an instance of a user-defined class.

5.3.3 Special Types

Special types are as follows:

- **Resource**: Hold references to resources external to PHP, such as database connections, database query, open file, and other external types.
- **NULL**: Represents a variable with a NULL value. A variable is considered to be null if:
  - It has been assigned the constant value NULL
  - It has not been set to any value yet
  - It has been unset ()

The syntax to initialize a variable is as follows:

Syntax:

```
$variable_name = value;
```

where,

`$variable_name` - specifies the name of the variable

`value` - specifies the value the variable will store

Assigning a value to a variable indicates an assignment operation. An assignment operation is executed with the equal to (=) sign included between the variable and the value. The equal to (=) sign is called the assignment operator. The left-hand side of the assignment operator has the name of the variable and the right-hand side of the assignment operator has the value for the variable. The name on the left-hand side of the assignment operator can be the name of a constant or a variable. A value of an expression can also be assigned to a variable. An expression consists of variables and operators.

Session 5

Using Variables and Expressions in PHP

Concepts

The following examples illustrate the use of variables in PHP.

Consider the code as shown in Code Snippet 2, to store an integer value in a variable, in a script named **`int_variable.php`**.

Code Snippet 2:

```
<?php
$Salary = 5000;
echo $Salary;
?>
```

Figure 5.2 displays the output of the script.

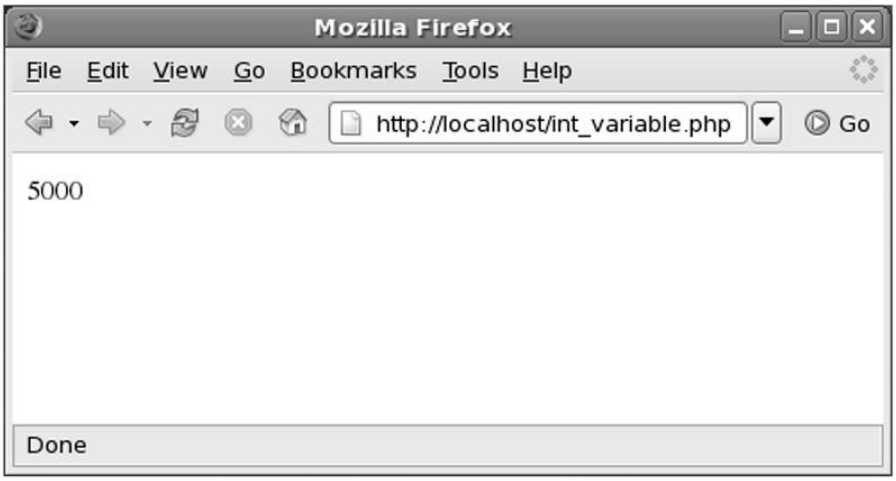


Figure 5.2: Storing an Integer Value in a Variable

In the code, the `$Salary` variable will be declared as the numeric variable because the value assigned to it is of the integer data type.

Session 5

To store a string value in a variable, enter the code as shown in Code Snippet 3, in a script named **str_variable.php**.

Code Snippet 3:

```
<?php
$message = "HELLO! How are you?";
echo $message;
?>
```

Figure 5.3 displays the output of the script.

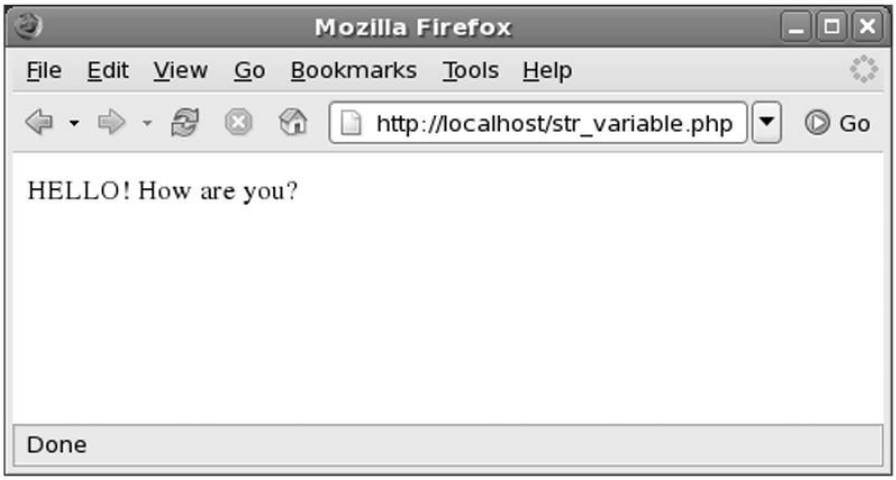


Figure 5.3: Storing a String Value in a Variable

In the code, the `$message` variable will be declared as the string variable because the value assigned to it is of string data type. The string is enclosed within the double quotes.

Session 5

Using Variables and Expressions in PHP

Concepts

To store the value of an expression in a variable, enter the code as shown in Code Snippet 4, in a script named **exp_var.php**.

Code Snippet 4:

```
<?php
$number1 = 1019;
$number2 = 126;
$number3 = $number1 + $number2;
echo $number3;
?>
```

Figure 5.4 displays the output of the script.

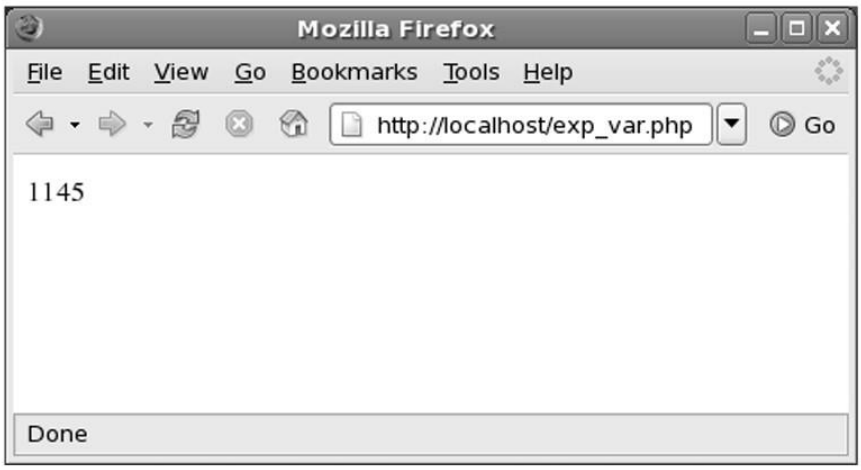


Figure 5.4: Storing the Value of Expression in a Variable

In the code, the **$number3** variable will be declared as an integer variable and the value of the addition expression (**$number1** + **$number2**) is assigned to it. Since **$number1** and **$number2** variables are assigned integer values, the value of the expression stored in the **$number3** variable also stores the integer data type.

The alternate method to assign a value to a variable is by referencing to another variable. In this, a variable stores data of another variable. This is done by providing a reference of a variable to another variable. The new variable is provided the reference of a variable using the equal to (=) symbol followed by an ampersand (&) symbol.

Session 5

Note: The ampersand (&) symbol signifies that the object is being passed by reference rather than by value. When an object is passed by reference, its original value can be changed in the function. If the object is passed by value, a copy of it is made so that the original object does not change.

The syntax to assign a value to a variable by referencing another variable is as follows:

Syntax:

```
$new_varname = & $old_varname
```

where,

`$new_varname` - defines the new variable name

`$old_varname` - specifies the other variable name whose value is to be assigned

To assign the value of one variable to another, enter the code as shown in Code Snippet 5, in a script named **`var_ref.php`**.

Code Snippet 5:

```
<?php
$Fname = "John";
$Lname = "Smith";
$name =& $Fname;
echo $name;
echo "<br>";
echo $Lname;
echo "<br>";
?>
```

Concepts

Figure 5.5 displays the output of the script.

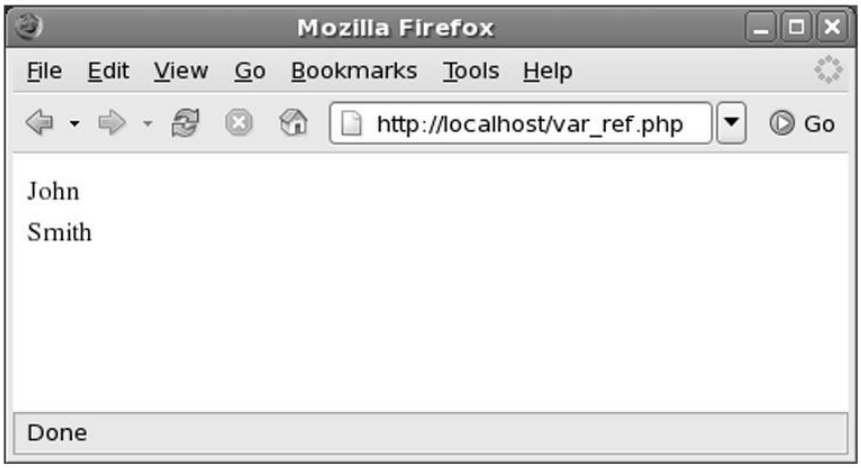


Figure 5.5: Assigning Value of One Variable to Another

In the code, the variables, **`$Fname`** and **`$Lname`** are string variables. The reference of the **`$Fname`** variable is assigned to the **`$name`** variable using the equal to and ampersand symbols.

5.4 Constants

Constants are identifiers that contain values that do not change throughout the execution of a program.

The points to be noted while using constants are as follows:

- Is case-sensitive
- Is static, meaning constants once defined cannot be changed
- Is declared using the `define( )`function
- Has a global scope of existence and can be accessed from anywhere in the script

Note: Constant names normally appear in uppercase letters in most programming languages including PHP. However, there is also an option to define the constants as case-insensitive, which does not require the code to use the correct casing while referring to a constant.

Session 5

The syntax to declare a constant using the `define()` function is as follows:

Syntax:

```
define(string_name, mixed_value)
```

where,

`string_name` - defines the variable name for the constant

`mixed_value` - specifies a numeric or string value that is assigned to the constant variable

To declare the constant, `NAME`, containing a string value, enter the code as shown in Code Snippet 6, in a script named **`dec_con.php`**.

Code Snippet 6:

```
<?php
//Enable error reporting
error_reporting(-1);
define("NAME", "John Smith");
echo NAME;
echo "<br>";
echo name;
echo "<br>";
?>
```

Figure 5.6 displays the output of the script.

Figure 5.6: Declaring a Constant

The code snippet declares a constant "`NAME`" and assigns a string value to it. On executing the statement, `echo NAME`, the string value stored in the constant will be displayed. The declaration of the constant is case sensitive. Therefore, the statement `echo` name displays the text `name`.

Case in-sensitive constants can also be declared using the `define()` function.

The syntax to declare a case-insensitive constant is as follows:

Syntax:

```
bool define ( string $name , mixed $value [, bool $case_insensitive])
```

where,

`$name` - defines the variable name for the constant

`$value` - specifies a numeric or string value that is to be made constant

`$case_insensitive` - specifies a Boolean value, `TRUE` or `FALSE`. If the value specified is `TRUE`, the constant name will be case insensitive.

Note: Two constants with the same name but in different cases represent different values.

Session 5

To declare a case insensitive constant, enter the code as shown in Code Snippet 7, in a script named **`case_ins.php`**.

Code Snippet 7:

```
<?php
define("NAME", "Michael Graff", TRUE);
echo NAME;
echo "<br>";
echo name;
echo "<br>";
?>
```

Figure 5.7 displays the output of the script.

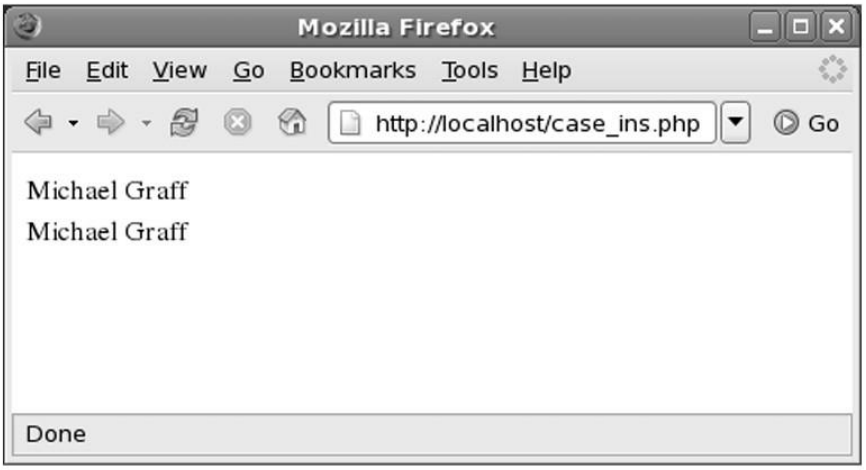


Figure 5.7: Declaring a Case-Insensitive Constant

The `TRUE` value in the `define()` function specifies that the constant declared is case-insensitive. The execution of both the statements, `echo NAME` and `echo name`, displays the string value stored in the constant "`NAME`".

Concepts

5.5 Scope of Variables

The scope of a variable is the context within which it is defined. It is the lifetime of a variable from the time the variable is created to the time the execution of the script ends. In PHP, the different scopes that a variable can have are as follows:

- Local
- Global
- Static

5.5.1 Local Variables

A variable initialized and used inside a function is called a local variable. The declaration of a local variable is similar to a normal variable. It is declared inside a function. The lifetime of a local variable begins when the function is called, and ends when the function is completed.

To display the use of local variables, enter the code as shown in Code Snippet 8, in a script named **`local_var.php`**.

Code Snippet 8:

```
<?php
$num2 = 0;
echo "The multiplication value of 14 * 15 is <br>";
function multiply()
{
$num1=14;
$num2=15;
$num2=$num1 * $num2;
echo $num2;
}
multiply();
echo " <br> The value of the local variable on exiting the function is
<br>";
echo $num2;
?>
```

Concepts

Figure 5.8 displays the output of the script.

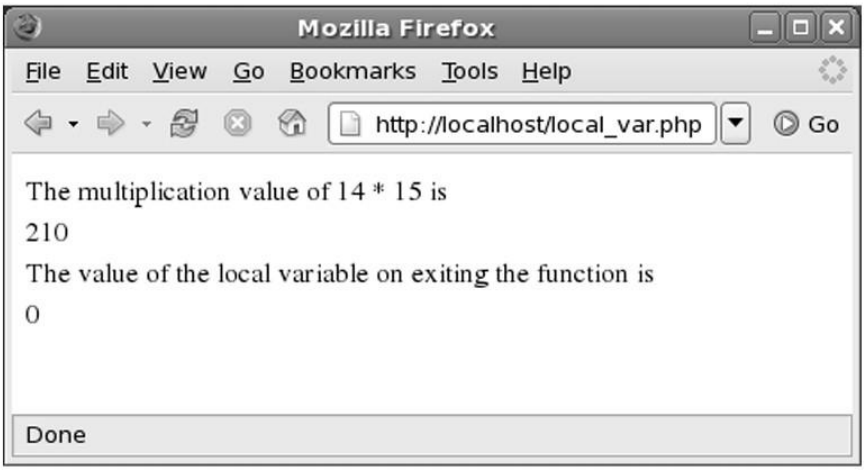


Figure: 5.8: Initializing Variables Inside a Function

In the code, the variables, **num1** and **num2** are declared as local variables, inside the function **multiply()**. They are initialized and used inside the **multiply()** function. The product of two variables, **num1** and **num2** is calculated in the **multiply()** function. This function executes when the PHP script invokes the function using the function name.

5.5.2 Global Variables

A variable that retains its value throughout the lifetime of a Web page is called a global variable. A global variable is declared with the help of the keyword `global`, within the function. As a result, it can be accessed from any part of the program.

The syntax to declare a global variable in a program is as follows:

Syntax:

```
global $var_name;
```

where,

`$var_name` - defines the global variable name

Session 5

Using Variables and Expressions in PHP

Concepts

To display multiplication of two numbers using global variables, enter the code as shown in Code Snippet 9, in a script named **multi_global.php**.

Code Snippet 9:

```
<?php
$var1 = 4;
$var2 = 15;
function multiply()
{
global $var1, $var2;
$var2 = $var1 * $var2;
echo $var2;
}
echo "The multiplication value of 4 * 15 =";
multiply();
?>
```

Figure 5.9 displays the output of the script.

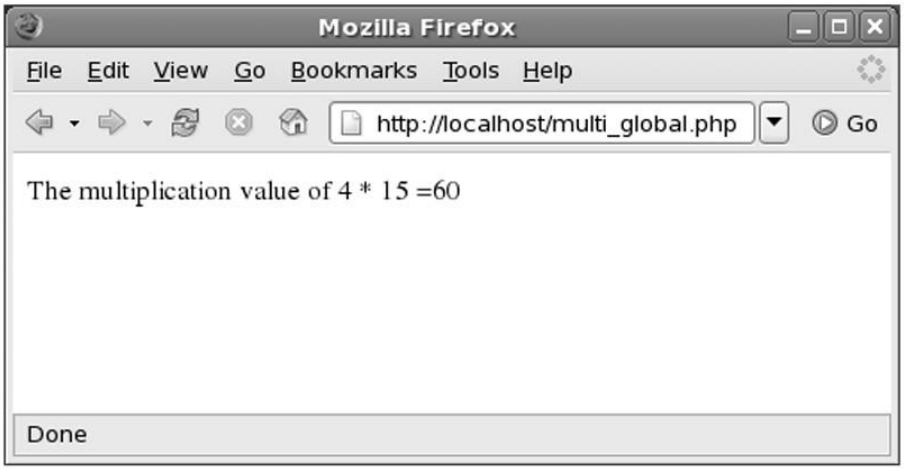


Figure 5.9: Multiplying Two Numbers Using Global Variables

In the code, the variables, **var1** and **var2** are declared as global variables. They are initialized outside the function and declared as global variables within the **multiply()** function. Therefore, all references to either variable will refer to the global value. The product of two variables, **var1** and **var2** is calculated in the **multiply()** function. This function executes when the PHP script invokes the function using the function name.

5.5.3 Static Variables

A static variable is similar to a local variable. The only difference is that the static variable retains its value even after the function terminates. When a function terminates, the local variables of that function lose their values. To retain the value of the variable throughout the execution of the program, use the `static` keyword with the variable name. Static variables are only accessible from within the function they are declared and their value remains intact between function calls. Static variables can be initialized during declaration and the static declarations are resolved during compile time. The static variable is commonly used in the recursive functions. A recursive function calls itself recursively within a function.

The syntax to declare a static variable is as follows:

Syntax:

```
static $var_name = value;
```

where,

`var_name` - defines the variable name

`value` - specifies the value of the variable

To illustrate the use of a static variable, enter the code as shown in Code Snippet 10, in a script named **`stat_var.php`**.

Code Snippet 10:

```
<?php
$var1;
function sum()
{
     static $var1 = 9;
     $var2 = $var1 + 12;

echo "The value of the variable is : $var1<br>";
echo "The addition value of 9 + 12 = ";
echo "$var2<br>";
}
sum();
?>
```

Concepts

Figure 5.10 displays the output of the script.

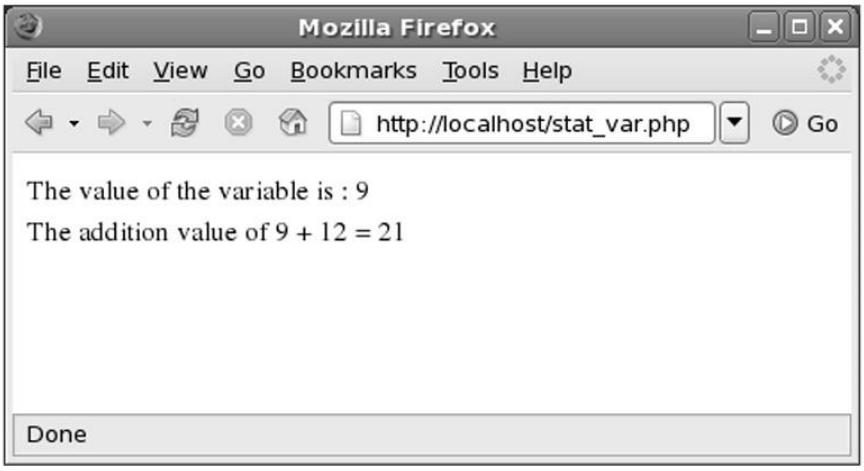


Figure 5.10: Using Static Variables

In the code, the first time the function **`sum()`** is called, the static variable **`$var1`** is set to zero, and incremented to display 1 as output. The value of **`$var1`** is maintained for subsequent calls. Therefore, the next function call will increment the value of **`$var1`** by one and print 2. The variable, **`var1`** is declared as a static variable. It is declared inside the **`sum()`** function. The variable **`var1`** is local to the **`sum()`** function but retains its value throughout the program.

5.6 HTTP Environment Variables

Environment variables are system-defined variables that can be used in any PHP script. These variables provide information about the transactions between the client and the server. They provide information about an HTTP request or response. The environment variables are similar to the user-defined variables because they also begin with the dollar ($) sign.

Following are some of the environment variables in PHP:

- **SERVER_SOFTWARE** - is a server identification string specified in the headers when responding to requests. It returns the name and the version of the server software.

To view the server software version, enter the code as shown in Code Snippet 11, in a script called **server_software.php**.

Code Snippet 11:

```
<?php
echo $_SERVER['SERVER_SOFTWARE'];
?>
```

Figure 5.11 displays the output of the script.

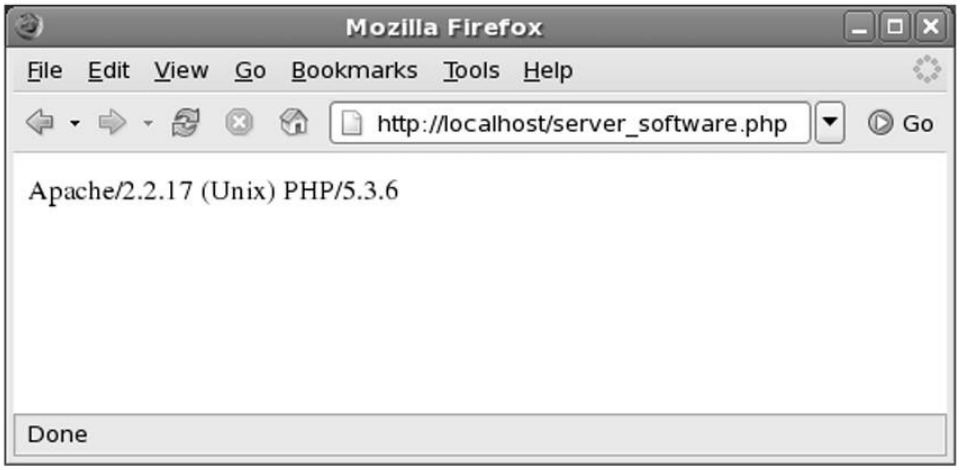


Figure 5.11: SERVER_SOFTWARE Environment Variable

- **SERVER_NAME** - Returns the name of the server host under which the current script is executing. The host name can be the Internet Protocol (IP) address or the Domain Name System (DNS) name of the server.

To view the server host name, enter the code as shown in Code Snippet 12, in a script called **server_name.php**.

Code Snippet 12:

```
<?php
echo $_SERVER['SERVER_NAME'];
?>
```

Concepts

Figure 5.12 displays the output of the script.

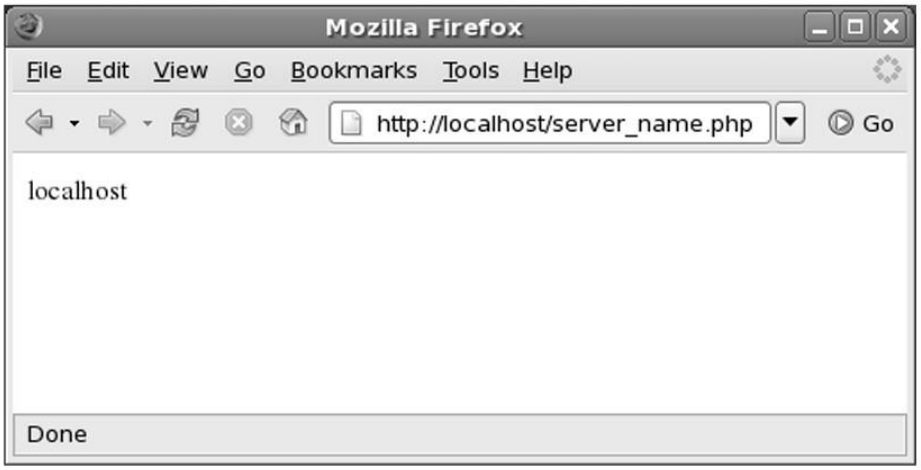


Figure 5.12: SERVER_NAME Environment Variable

- **SERVER_PROTOCOL** - Returns the name and version number of the protocol via which the page was requested.

  To view the server protocol, enter the code as shown in Code Snippet 13, in a script named **`server_protocol.php`**.

  Code Snippet 13:

```
<?php
echo $_SERVER['SERVER_PROTOCOL'];
?>
```

Figure 5.13 displays the output of the script.

Figure 5.13: SERVER_PROTOCOL Environment Variable

- **SERVER_PORT** - Returns the server port number for the script.

  To view the server port, enter the code as shown in Code Snippet 14, in a script called **`server_port.php`**.

  Code Snippet 14:

```
<?php
echo $_SERVER['SERVER_PORT'];
?>
```

Session 5

Figure 5.14 displays the output of the script.

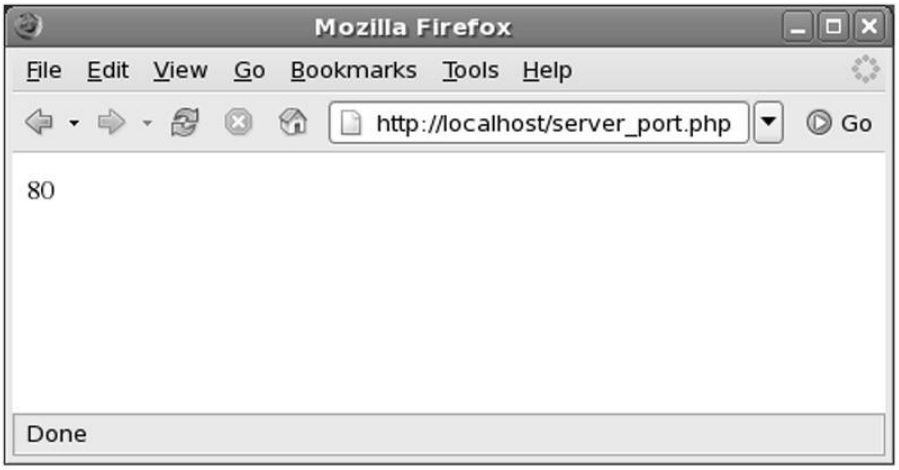


Figure 5.14: SERVER_PORT Environment Variable

- **$_COOKIE** - Returns the content of the recently used cookie. A cookie stores user information sent to the browser by the Web server. A cookie is saved as a text file. A cookie is sent back to the server when the browser requests to display a page.

  To set the value of a cookie, enter the code as shown in Code Snippet 15, in a script named, **`set_cookie.php`**.

  Code Snippet 15:

```
<?php
$Month = 86400 + time();
setcookie('Name', 'Jerry', $Month);
echo "The cookie has been set.";
?>
```

Figure 5.15 displays the output of the script.

Figure 5.15: Setting a Cookie

To retrieve the value of the cookie defined in Code Snippet 15, enter the code as shown in Code Snippet 16, in a script named, **`ret_cookie.php`**.

Code Snippet 16:

```
<?php
if(isset($_COOKIE['Name']))
     {
     $last = $_COOKIE['Name'];
     echo "Welcome back! <br> Your name is ". $last;
     }
else
     {
     echo "Welcome to our site!";
     }
?>
```

Concepts

Figure 5.16 displays the output of the script.

Figure 5.16: Retrieving the Value from a Cookie

- **HTTP_USER_AGENT** - Returns the name of the browser the client is using.

  To view the type of the browser, enter the code as shown in Code Snippet 17, in a script called **`http_user.php`**.

  Code Snippet 17:

```
<?php
echo $_SERVER['HTTP_USER_AGENT'];
?>
```

Figure 5.17 displays the output of the script.

Figure 5.17: HTTP_USER_AGENT Environment Variable

- **HTTP_ACCEPT** - Returns the contents of the Accept: header if there is a current request. It lists the media types the client will accept.

  To view the list of media types, enter the code as shown in Code Snippet 18, in a script called **`http_accept.php`**.

  Code Snippet 18:

```
<?php
echo $_SERVER['HTTP_ACCEPT'];
?>
```

Concepts

Figure 5.18 displays the output of the script.

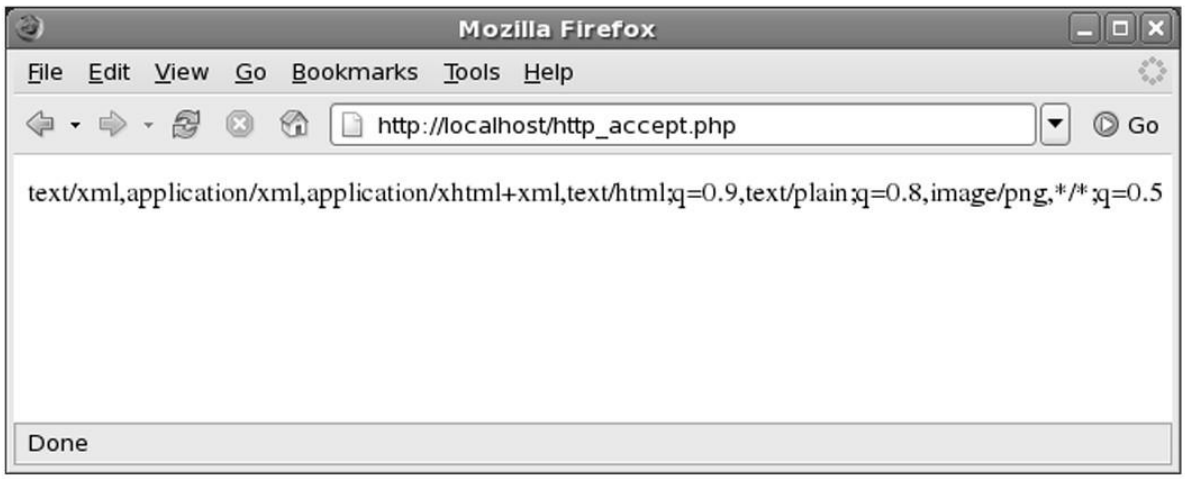


Figure 5.18: HTTP_ACCEPT Environment Variable

- **HTTP_FROM** - Returns the e-mail address of those users who have made request to the server.

Summary

- Identifiers are names given to different elements of a program such as, variables, constants, arrays, and classes.
- A variable is an identifier whose value keeps changing. Variables are used to store data values. A dollar ($) symbol must be included before a variable name.
- Constants are identifiers whose values do not change throughout the execution of a program. They are declared using the define() function.
- The scope of a variable defines the availability of the variable in a program in which it is declared. The different scopes of a variable are local, global, and static.
- A variable initialized and used inside a function is called a local variable. When a variable retains its value throughout the lifetime of the Web page, it is called the global variable. It is declared using the keyword global within the function.
- The static variable retains its value even after the function terminates. It is declared using the keyword static with the variable name.
- Environment variables are system-defined variables that can be used in any PHP script.
- The $_COOKIE environment variable returns the content of the recently used cookie.

Session 5

Concepts

Check Your Progress

1. An assignment operation is performed using the _____ sign.

   a. ,

   b. .

   c. =

   d. &

2. Which of the following function is used to declare a constant?

   a. constant()

   b. define()

   c. string

   d. static

3. A constant has a _____ scope of existence.

   a. Local

   b. Static

   c. Global

   d. Constant

4. The _____ environment variable returns the content of the recently used cookie.

   a. SERVER_SOFTWARE

   b. SERVER_PORT

   c. $_COOKIE

   d. HTTP_ACCEPT

Check Your Progress

5. Which of the following environment variable returns the name and version number of the protocol?

   a. SERVER_PROTOCOL

   b. SERVER_PORT

   c. HTTP_POST

   d. HTTP_FROM

6. The _____ data type stores true or false values.

   a. string

   b. integer

   c. floating-point

   d. boolean

7. The _____ variable retains its value throughout the lifetime of the program.

   a. Local

   b. Static

   c. Global

   d. Environment

8. Which of the following environment variable returns information about the port in use?

   a. SERVER_PROTOCOL

   b. SERVER_NAME

   c. POST_FILES

   d. SERVER_PORT

Session 5

Using Variables and Expressions in PHP

Concepts

Answers

1. c
2. b
3. c
4. c
5. a
6. d
7. b
8. d

Onlinevarsity

Session 6

Using Variables and Expressions in PHP (Lab)

Objectives

At the end of this session, the student will be able to:

- *Explain the use of the assignment operator.*
- *Explain the use of the local variables.*
- *Explain the use of the global variables.*
- *Explain the use of the static variables.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Configuring Apache Web Server to Execute PHP Scripts

In order to execute PHP scripts, Apache Web server must be configured. The configuration enables Apache Web server to interpret and execute PHP scripts. The changes to the configuration of the Apache Web server must be done in the `httpd.conf` file.

To edit the `httpd.conf` file, perform the following steps:

1. **Using the GUI in Linux, browse to the /usr/local/apache2/conf directory.**
2. **Right-click the httpd.conf file and select Open With Text Editor. The file opens in the text editor.**
3. **Locate the <Directory "/usr/local/apache2/htdocs>" section in the httpd.conf file.**
4. **Enter the following code:**

```
Options Indexes FollowSymLinks ExecCGI
AddHandler cgi-script .cgi .pl
AddHandler php5-script php
AddType text/html php
```

Session 6

5. **Click Save.**

6. **Restart Apache Web server.**

Using Assignment Operators

A variable is an identifier whose value changes as the program progresses. A variable name is used to access the value stored in the variable and the data type refers to the type of data that the variable can store. Variables can be used to store user information or intermediate data such as, calculated results, and values returned by the functions.

Assigning a value to a variable represents an assignment operation. An assignment operation is performed with the help of an equal to (=) sign, the assignment operator, which is placed between the variable and the value. The left-hand side of the assignment operator has the name of the variable and the right-hand side of the assignment operator has the value that is required to be stored in the variable. You can also assign a value of an expression to a variable.

The following examples display the use of assignment operators:

- `$number = 1;`
- `$string = "We are studying PHP. ";`

To assign a string value to a variable and display it, perform the following steps:

1. **To start the Apache Web server, enter the following command at the Terminal:**
```
/usr/local/apache2/bin/apachectl start
```

2. **Open a new file in the gedit text editor.**

3. **Enter the following code in the text editor:**
```
<HTML>
<HEAD>
   <TITLE>String Assignment Operation</TITLE>
</HEAD>
<BODY>
<?php
//Declaring the variable and assigning a value
$string_val = "This is John Cizzel.";
```

Session 6


```
//Displaying the value
echo $string_val;
?>
</BODY>
</HTML>
```

4. **Save the file as assignstr.php under the /usr/local/apache2/htdocs directory.**

5. **Open the Mozilla Firefox Web browser.**

6. **Type http://localhost/assignstr.php in the Address bar and press Enter.**

   Figure 6.1 displays the output of the script.

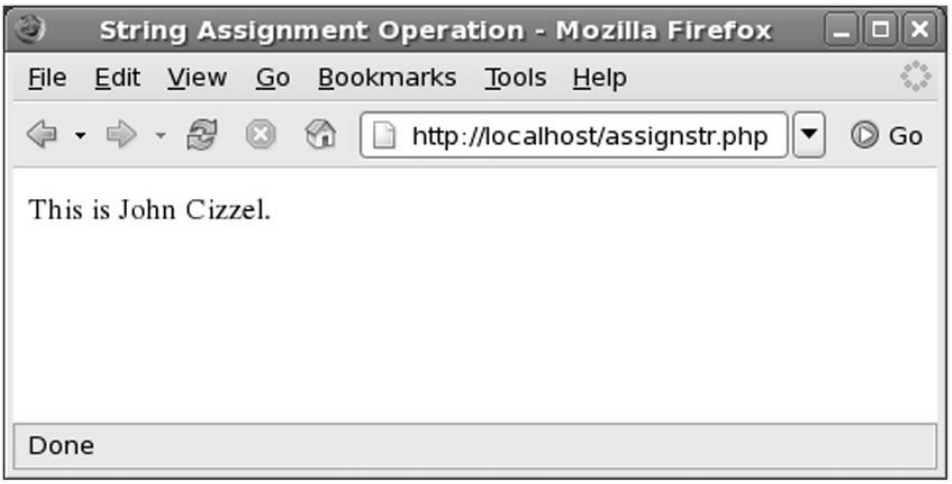


Figure 6.1: Assigning a String Value to a Variable and Displaying it

To calculate the product of two numbers, assign it to a variable, and display the output, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code in the text editor:**

```
<HTML>
<HEAD>
<TITLE>Integer Assignment Operation</TITLE>
</HEAD>
```

Session 6


```
<BODY>
<?php
//Declaring variables and assigning values
$int_value1 = 13;
$int_value2 = 45;
//Displaying the text
echo "The product of $int_value1 * $int_value2 is : ";
//Calculating the product
$int_value2 = $int_value1 * $int_value2;
//Displaying the result
echo $int_value2;
?>
</BODY>
</HTML>
```

3. **Save the file as assignint.php under the /usr/local/apache2/htdocs directory.**
4. **Open the Mozilla Firefox Web browser.**
5. **Type http://localhost/assignint.php in the Address bar and press Enter.**

   Figure 6.2 displays the output of the script.

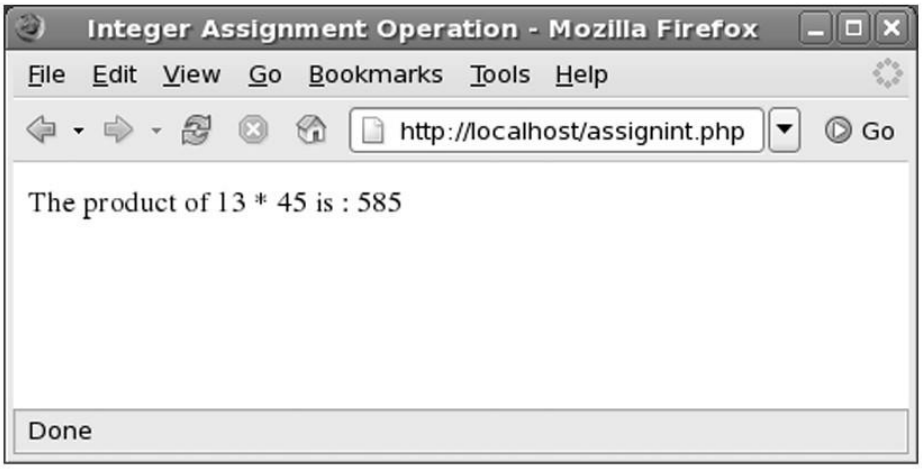


Figure 6.2: Displaying the Value of an Expression

Session 6

Using Variables and Expressions in PHP (Lab)

To subtract a decimal number from another and assign its value to a variable, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<HTML>
<HEAD>
<TITLE>Float Assignment Operation</TITLE>
</HEAD>
<BODY>
<?php
//Declaring and assigning values to variables
$float_value1 = 134.57;
$float_value2 = 12.87;
//Displaying the text
echo "The subtraction value of $float_value1 - $float_value2 is : ";
//Calculating the difference
$float_value2 = $float_value1 - $float_value2;
//Displaying the result
echo $float_value2;
?>
</BODY>
</HTML>
```

3. **Save the file as assignfloat.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

Session 6

5. **Type http://localhost/assignfloat.php in the Address bar and press Enter.**

   Figure 6.3 displays the output of the script.

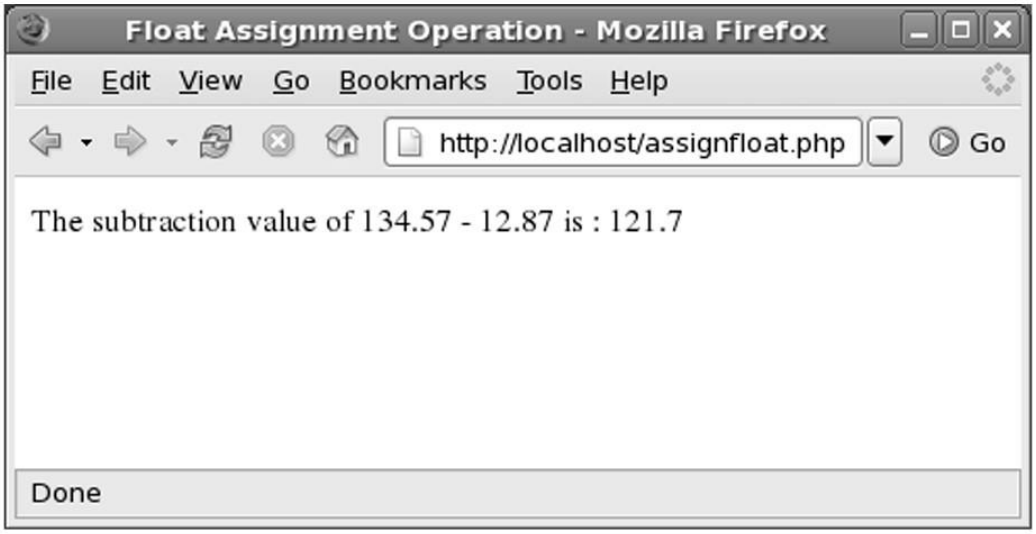


Figure 6.3: Displaying the Value of a Subtraction

Using Local Variables

A variable initialized and used inside a function is known as a local variable. A local variable is declared like any other variable. Variables are local to their scope and when created within a function, they are available only within a function. Thus, the scope of a local variable begins when the function is invoked and ends when the function is complete. If a local variable is used beyond its scope, an error message is displayed, which says that an undefined variable is declared.

To use local variables in the function that calculates sum of two variables, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<HTML>
<HEAD>
<TITLE>Local Variables</TITLE>
</HEAD>
<BODY>
<?php
```

```
//creating a function
function sum()
 {
 //Declaring and assigning values to variables
 $int1 = 43.4;
 $int2 = 13;
 $int3 = $int1 + $int2;
//Displaying text
 echo "The sum of $int1 + $int2 = $int3";
 }
//calling the function
sum();
?>
</BODY>
</HTML>
```

3. **Save the file as localvar.php under the /usr/local/apache2/htdocs directory.**
4. **Open the Mozilla Firefox Web browser.**
5. **Type http://localhost/localvar.php in the Address bar and press Enter.**

Figure 6.4 displays the output of the script.

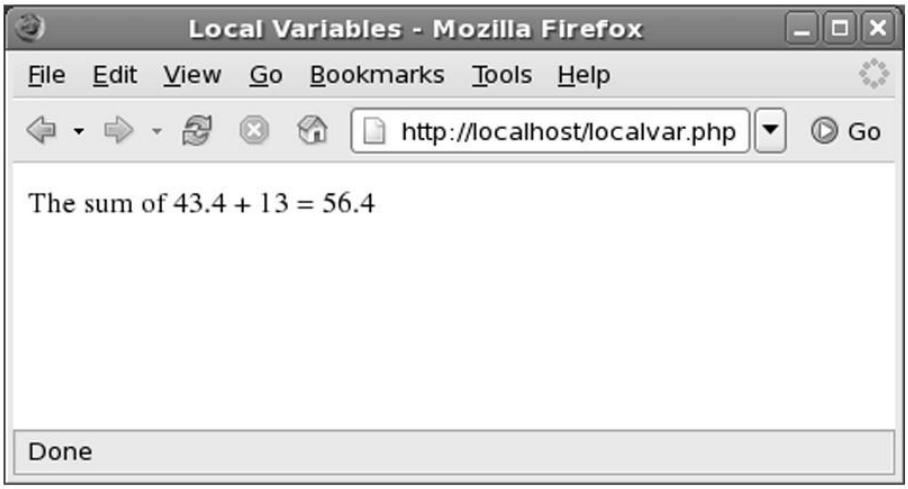


Figure 6.4: Using Local Variables

Session 6

Using Variables and Expressions in PHP (Lab)

Note: In the code, the variables, `$int1` and `$int2`, are declared inside the `sum()` function as the local variables of the function. They are initialized and used inside the function. The sum of two variables, `$int1` and `$int2`, is calculated in the `sum()` function. This function is executed when the PHP script invokes the function using the function name.

Lab Guide

Using Global Variables

A global variable is accessed from any part of the program. To modify a global variable, explicitly declare it as global in the function in which it is to be modified. This is achieved with the help of the keyword `global` within the function.

To use global variables in a function, and calculate the quotient of two integers, perform the following steps:

1. **Open a new file in the text editor.**

2. **Enter the following code:**

```
<HTML>
<HEAD>
<TITLE>Global Variables</TITLE>
</HEAD>
<BODY>
<?php
//Declaring and assigning values to variables
$int1 = 68;
$int2 = 50;
//Creating a function
function division()
{
 global $int1, $int2;
 //Storing the result in a variable
 $int3 = $int1/$int2;
 //Displaying text
 echo "The quotient for $int1/$int2 = $int3";
}
//Calling the function
division();
?>
</BODY>
</HTML>
```

3. **Save the file as globalvar.php under the /usr/local/apache2/htdocs directory.**
4. **Open the Mozilla Firefox Web browser.**
5. **Type http://localhost/globalvar.php in the Address bar and press Enter.**

   Figure 6.5 displays the output of the script.

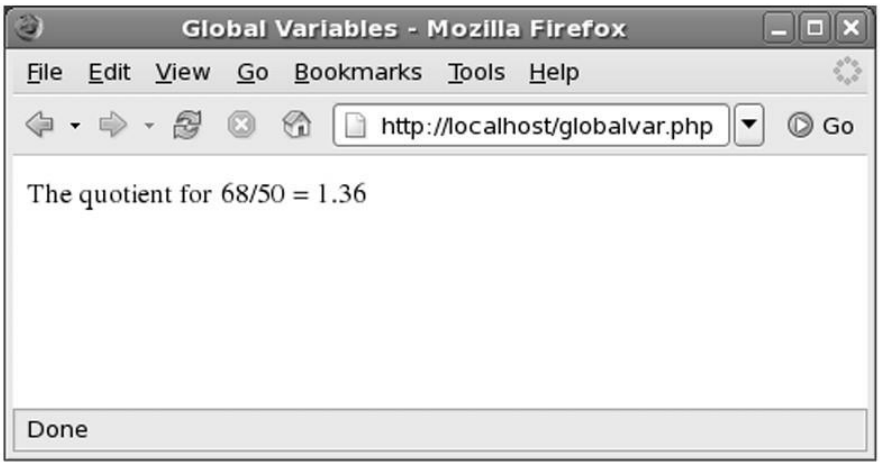


Figure 6.5: Using Global Variables

Using Static Variables

A static variable is similar to a local variable. The only difference is that the static variable retains its value even after the function terminates. When a function terminates, local variables declared in the function lose their values. To retain the value of the variable throughout the execution, use the `static` keyword with the variable name.

To display the use of static variable, decrement a value using a static variable. Decrementing a value means reducing a value by 1. This is done either by using the decrement operator or by simply performing a subtraction expression.

Session 6

Using Variables and Expressions in PHP (Lab)

Lab Guide

1. **Open a new file in the gedit text editor.**

2. **Enter the following code in the editor:**

```
<HTML>
<HEAD>
<TITLE>Static Variables</TITLE>
</HEAD>
<BODY>
<?php
//Creating a Function
function decrement()
{
 //Declaring the Variable and assigning the Value
 static $static_int1 = 99;
 //Decrementing the Value
 $static_int1-- ;
 //Displaying the Value
 echo "The decrement value is $static_int1";
}
//Calling the Function
decrement();
?>
</BODY>
</HTML>
```

3. **Save the file as staticdec.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Type the http://localhost/staticdec.php key in the Address bar and press Enter.**

   Figure 6.6 displays the output of the script.

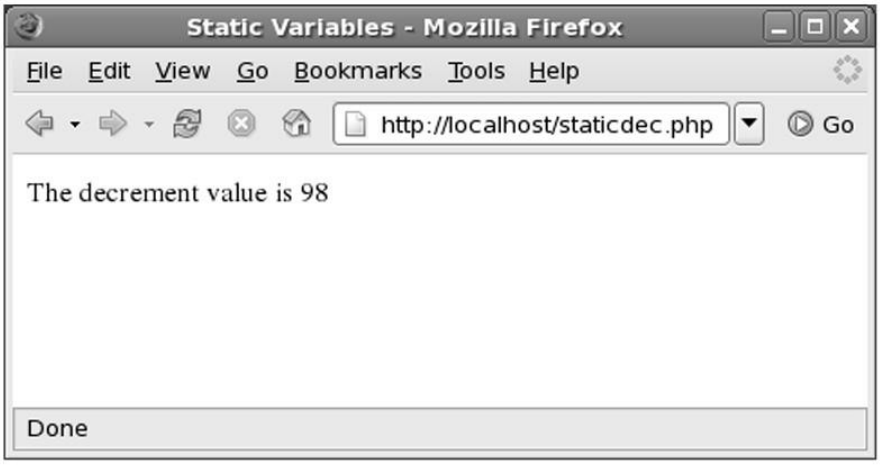


Figure 6.6: Using Static Variables

Note: In the code, the variable, `static_int1`, is declared as the static variable. It is declared inside the `decrement()` function. The `static_int1` variable is local to the `decrement()` function but it retains its value throughout the program.

Session 6

Using Variables and Expressions in PHP (Lab)

Lab Guide

Do It Yourself

1. Store the string, "`I am learning PHP 5`", in a variable and display it on the Web browser.
2. Create a form that accepts two numbers and displays its sum using the PHP script.

Onlinevarsity GROUPS

Session 7

Scalar Type Declarations

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain scalar type declarations.*
- *Describe usage of scalar type declarations in PHP programs.*
- *Explain scalar type hinting.*
- *Explain anonymous classes.*
- *Describe usage of anonymous classes in PHP programs.*

7.1 Introduction

The session introduces scalar type declarations and anonymous classes. The session explains how code written in older PHP versions can be impacted by scalar type declarations in PHP. Further, this session explains anonymous classes.

7.2 Scalar Data Type

Data types that hold single values are known as scalar data types. They could be any one of the following:

- Integer
- Float
- Boolean
- Strings

Scalar data types are also called as base data types.

7.2.1 Type Declarations in PHP

Type declaration refers to specifying the data type of the parameter while passing it to the function. In other words, it is specifying the data type of a parameter in the function. Type declaration is also known as type hinting.

Session 7

Type hinting refers to providing hints to a function, only to accept a given data type. Using this method, one can enforce a function to accept the desired data type. From type declaration, one can enforce the input parameter data type. This can be either strict or coercive.

Upon calling the function, PHP runtime raises an error and halts execution if the arguments are not of the specified type.

7.2.2 Explicit Declaration of Scalar Types

The syntax to declare a new function is as follows:

```
function function_name(type, p1)
```

where,

> `function` is a reserved word and `type` is data type of the function parameter. This can be any one of the scalar data types: `int`, `float`, `string`, or `bool`. `p1` is the parameter.

This new form of function declaration can be accommodated in the previous versions of PHP. You can add the following statement to the beginning of the source code file:

```
declare(string_type=1);
```

Note: This declaration has to be on the first line of the PHP source file and should not be declared at any other place in the PHP program.

7.2.3 Hands-On Exercise: Declaring and Using Scalar Types

Let us now take a few examples of scalar type declarations. In Code Snippets 1 to 4, each code defines a function and a parameter of a particular data type. However, when calling the function with a different data type value, the program would not fail. This is because there is no strict checking.

In Code Snippet 5, the code uses `declare` statement with `strict_types=1`. This statement would verify that the data type of function parameter and the data type of value that is being passed are the same. If they do not match, the program execution fails.

To begin with, let us write a function `test1` with a parameter whose data type is integer. A `float` value will be passed while calling the function. The function should display the value of its parameter that will be converted to `int` data type.

Session 7

Code Snippet 1 shows the code to achieve this.

Code Snippet 1:

```
<?php
function test1(int $x){
echo 'integer $x = '.$x;
}
test1(10.124);
//output: integer $x = 10
?>
```

After defining the function, it will be called and passed a `float` value (for example, 10.124 here). Observe the output that is shown in figure 7.1.

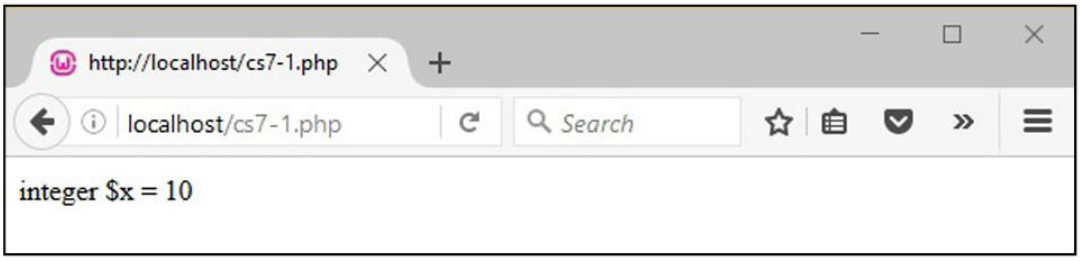


Figure 7.1: Output of Code Snippet 1

The code in Code Snippet 1 defines a function named `test1`. This function has a parameter `$x` and its data type is `int`. While calling function `test1`, we are passing a `float` value 10.124. As the function accepts integer values as parameter, the value 10.124 will be converted to integer value 10. The function prints the value of the parameter that is converted to `int`. Hence, the result is 10.

Now, let us take up another example. Code Snippet 2 displays the code for this example.

Code Snippet 2:

```
<HTML>
<BODY>
<?php
function test1(float $x){
echo 'float $x = '.$x;
}
test1(true);
 ?>
 </BODY>
 </HTML>
```

Session 7

Scalar Type Declarations

Concepts

Code Snippet 2 defines a function `test1`, which has a parameter with data type `float`. While calling the function `test1`, boolean value '`true`' is passed. This value is then converted to integer and 1 is displayed.

The output is shown in figure 7.2.

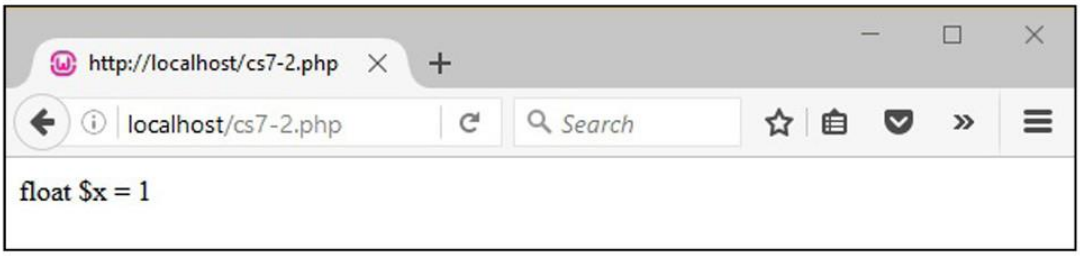


Figure 7.2: Output of Code Snippet 2

Here is another example to show the declaration and usage of data types. Code Snippet 3 shows a function `test1` that has a parameter with boolean data type. We are passing a `float` value and that is converted to boolean value. Any number other than zero is treated as `true`. The function should display the value of its parameter.

Code Snippet 3:

```
<HTML>
<BODY>
<?php
 function test1(bool $a){
 echo $a;
 }
 test1(10.34);//
?>
</BODY>
</HTML>
```

In Code Snippet 3, the function `test1` is defined. It has a parameter `$a` and its data type is `bool`. While calling the function, we are passing a `float` data type value that is 10.34. This value is converted to `bool` and the function prints the value of parameter that is passed.

Hence, the result is `true` as shown in figure 7.3. The result is true because we are passing a value 10.34 and it is converted to boolean value, `true`.

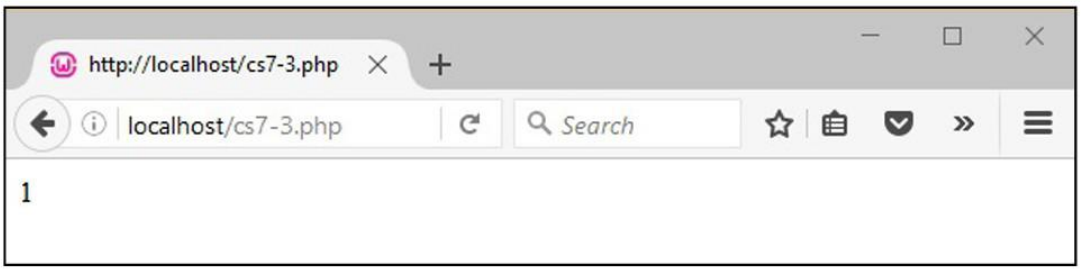


Figure 7.3: Output of Code Snippet 3

Code Snippet 4 shows an example of passing `string` with input parameter while providing declaration or type hint for `String`.

Here, you have a function `test1`, which has a parameter whose data type is string. The function should display the value of its parameter. Observe the output when you call the function and pass a float value.

Code Snippet 4:

```
<HTML>
<BODY>
<?php
 function test1(string $a){
 echo $a;
 }
 test1(10.1234);
?>
</BODY>
</HTML>
```

Session 7

The output of Code Snippet 4 is shown in figure 7.4.

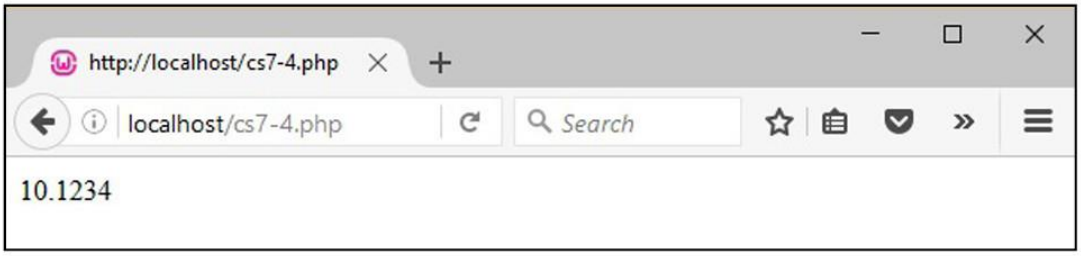


Figure 7.4: Output of Code Snippet 4

In Code Snippet 4, function `test1` is defined. It has a parameter `$x` of `string` data type. While calling the function, we are passing a `float` data type with value 10.34. This value is converted to `string` and the function prints the value of parameter that is passed. Hence, the result is as shown in figure 7.4.

In all the code snippets shown earlier, we did not use `strict_types=1` declaration. On account of this, the program is executed successfully even though we pass different data type value, other than the data type of the parameter used in the function.

Now to enforce the type declaration, use the `declare` statement with `strict_types=1` and explicitly declare the types of the scalar arguments. These are strictly checked.

Let us now take an example to demonstrate the use of the strict type declaration.

In Code Snippet 5, the first line is a `declare` statement with `strict_types = 1`. The code defines a function `test1`, which has a parameter with data type as integer. The function should display the value of its parameter. When you call the function and pass a boolean value, observe the output.

Code Snippet 5:

```
<?php declare(strict_types=1);
 function test1(int $a){
 echo $a;
 }
 test1(true);
?>
```

In Code Snippet 5, the first statement is a `declare` statement with `strict_types=1`. A function, `test1` is defined with an argument `$a` of data type integer. The function body has an `echo` statement that prints the value of the argument. When function `test1` is called with a boolean value `true`, the function does not accept arguments of values other than integer. As we used `strict_types=1`, the program verifies the argument value with the parameter data type. If it matches, the function would execute successfully otherwise, the program fails to execute and terminates prematurely.

Session 7

When Code Snippet 5 is executed, it will result in an error, as shown in figure 7.5.

```
Fatal error: Uncaught TypeError: Argument 1 passed to test1() must be of the
type integer, boolean given.
```

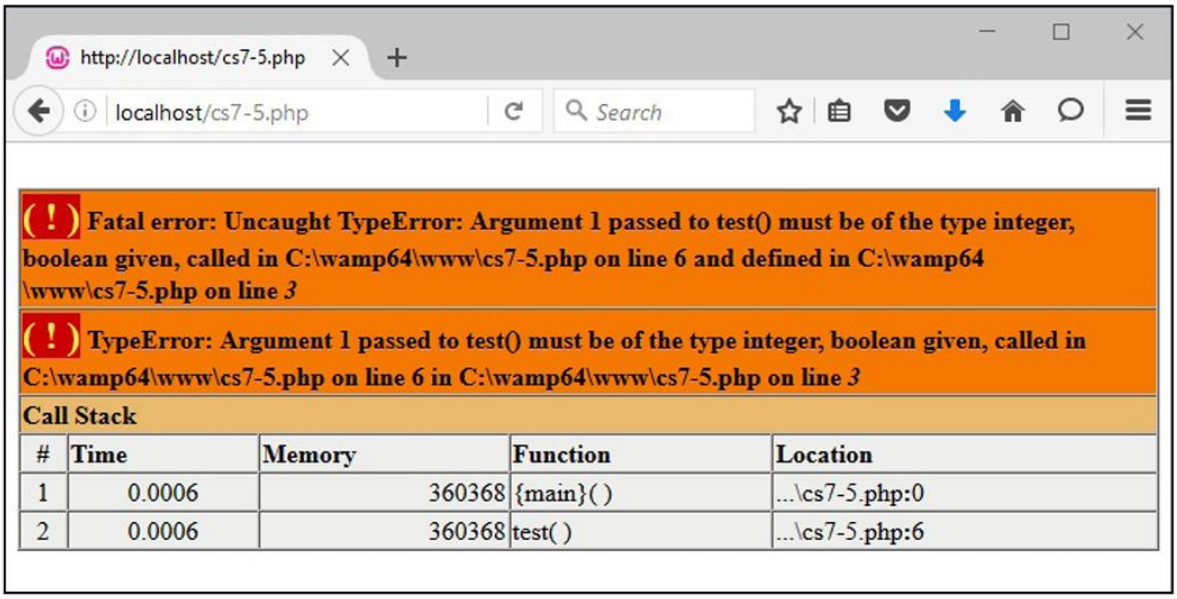


Figure 7.5: Output of Code Snippet 5

7.2.4 Behavior of Weak Type Checking and Type Conversion

By using the `declare(strict_types=0)` statement, weak type checking mode can be enforced.

However, there are few points to be considered while using this:

- Weak type checked calls to a built-in PHP function or to an extension have the same behavior as in previous PHP versions.
- The rules for weak type new scalar type declarations are almost the same as that of built-in PHP functions or to an extension of the PHP program.
- NULL is a special case in order to be consistent with the current type declarations for classes and arrays. By default NULL is not accepted, unless it is a parameter and is explicitly given a default value of NULL.

Concepts

Table 7.1 summarizes the implicit scalar conversion in weak mode.

Type Declaration	int	float	String	bool
`int`	yes	yes	yes	yes
`float`	yes	yes	yes	yes
`string`	yes	yes	yes	yes
`bool`	yes	yes	yes	yes

Table 7.1: Implicit Scalar Conversion in Weak Mode

7.2.5 Behavior of Strict Type Checking

At the beginning of the PHP file, use `declare (strict_types=1)` which corresponds to strict type check mode. In case of strictly type checked call to a built-in PHP function or to an extension, the result would be a catchable error. When `strict_types=1` is declared, then scalar typed parameters are strictly checked which are straightforward. If the value type matches with the type of the parameter declared, then it would be accepted. If not, it will not accept that value with type mismatch and results in an error.

Note: The only exception is that scalar type conversion is done implicitly for integer to `float`. This implies that parameters can also accept `int` data type along with declaring `float`.

7.3 Anonymous Classes

An anonymous class is nothing but a class that is defined without a name.

7.3.1 Defining Anonymous Classes

An anonymous class is defined just like any other class definition. Internally, a name is generated for the anonymous class. An anonymous class is used in the following scenarios:

- When the class need not be documented.
- When the class is used only once during execution.

7.3.2 Creating Anonymous Classes

A class definition with a name is known as a named class. To define a named class, you use the keyword `class` followed by the class name and enclose the property and methods definitions within curly braces as shown.

Session 7

Syntax to define a named class:

```
class class_Name {

 // defined properties and methods

};

$object = new class_Name( 'arguments' );
```

Syntax of an anonymous class:

```
$object = new class( 'arguments' ) {

 // defined properties and methods
};
```

An anonymous class is similar to a named class. The primary difference is that, an anonymous class is instantiated without a name.

To create an anonymous class, you simply combine the `new class($constructor, $args)` followed by a standard class definition. An anonymous class is instantiated always during its creation, giving you an object of that class.

Let us now see an example of creating an anonymous class.

First, an anonymous class is created with public variable and values are assigned to those variables. Then within the anonymous class, two functions are created. The first function should not have any arguments while the second function should have one argument. The first function would return some message to the user while the second function should return the value of the argument.

The code will display the values of the variables of the anonymous class.

Code Snippet 6:

```
<?php
// anonymous_class.php
// PHP 7

$anon_class_obj = new class{
```

Session 7

Scalar Type Declarations

Concepts

```
 public $greeting = 'hello';
 public $Id = 754;
 const SETT = 'some configuration';
 public function getValue()
 {
 // do some operation
 return 'some returned value';
 }
 public function getValueWithArg($str1)
 {
 // do some operation
 return 'returned value is '.$str1;
 }
 };
echo '</br>';
echo $anon_class_obj->greeting;
echo '</br>';
echo $anon_class_obj->Id;
echo '</br>';
echo $anon_class_obj::SETT;
echo '</br>';
echo $anon_class_obj->getValue();
echo '</br>';
echo $anon_class_obj->getValueWithArg('Aptech');
echo '</br>';
echo '</br>';
?>
```

Code Snippet 6 creates an anonymous class. This class has three members, namely, `$greeting`, `$Id` and a constant `SETT`. These members are assigned with values '`hello`', `754`, and '`some configuration`'. Two functions `getValue` and `getValueWithArgument` are created. The `getValue` function returns a message 'Some Returned Value', whereas the `getValueWithArgument` function will return the value of its argument. Then `set echo` statements are used to display the values of the anonymous class members and functions.

Session 7

The output is shown in figure 7.6.

Figure 7.6: Output of Code Snippet 6

Session 7

Summary

- With scalar type hints, one can design reliable PHP code that ensures correctness of the programs.
- In addition to the default behavior, PHP 7 and higher versions support strict type hints.
- In order to enable strict types, you can use the declare statement to declare strict_types directive.
- Any type mismatch with function arguments results in an error.
- The addition of anonymous classes gives the programmer flexibility in writing PHP code.
- In order to create an anonymous class, you can combine new class ($constructor, $args) followed by a standard class definition.
- Anonymous classes open up a number of architectural possibilities.

Check Your Progress

1. What is the output of the following program?

```
?>php
declare(strict_types=1);
function print1(bool $a){
echo $a;
}
print1(10.34);
?>
```

a. 10.34

b. true

c. false

d. error

2. What is the output of the following program?

```
<?php
 declare(strict_types=1);
     function print2(float $a){
 echo $a;
     }
 print(1.123);
?>
```

a. 1.123

b. 1

c. 123

d. error

Session 7

Check Your Progress

3. What is the output of the following program?

```
<?php
 $a_class_obj = new class{
 public $UserId = 'XYZ';
 public $pword= '<&as123';
 };
echo $a_class_obj->UserId;
echo '</br>';
echo $a_class_obj->pword;
echo '</br>';
?>
```

a. XYZ
<&as123

b. <&as123
XYZ

c. XYZ

d. <&as123

4. With strict scalar type hint, which directive is used in the program?

a. declare(strict_types=0);

b. declare(strict_types=-1);

c. declare(strict_types=1);

d. No directive is required

Check Your Progress

5. Which of the following is not true about anonymous classes?

   a. Anonymous classes are similar to named classes

   b. Anonymous class is used when it is required to execute it only once

   c. Anonymous class is always instantiated during its creation

   d. Anonymous class is a class definition without a name

Session 7

Scalar Type Declarations

Concepts

Answers

1. d
2. a
3. a
4. c
5. a

Onlinevarsity

ANYTIME | ANYWHERE

Session

8 PHP Operators

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the use of arithmetic operators.*
- *Explain the use of logical operators.*
- *Explain the use of relational operators.*
- *Explain the use of bitwise operators.*
- *Explain the use of assignment operators.*
- *Explain the use of string operators.*
- *Explain the use of increment and decrement operators.*
- *Explain the use of conditional or ternary operators.*
- *Explain the precedence of operators.*

8.1 Introduction

All programming languages use operators. An expression consists of operators and operands. Operators are pre-defined symbols that perform specific actions on objects called operands. The operators are assigned a precedence value that indicates the order in which the operators are evaluated in an expression.

In this session, you will learn about the arithmetic, logical, relational, bitwise, assignment, string, increment, decrement, and conditional operators. In addition, you will also learn about operator precedence.

8.2 Classification of Operators

In PHP, the operators are classified into the following categories:

- **Unary Operator** - acts on only one operand in an expression
- **Binary Operator** - has two operands and performs different arithmetic and logical operations

Concepts

- **Conditional or Ternary Operator** - has three operands and evaluates the second or third expression depending on the result of the first expression

8.3 Arithmetic Operators

Arithmetic operators are binary operators that work only on numeric operands. If the operands are non-numeric values such as strings, Booleans, nulls, or resources, they are converted to their numeric equivalents before evaluating the expression.

To display a mathematical expression, enter the code as shown in Code Snippet 1, in a PHP script named **`math-exp.php`**.

Code Snippet 1:

```
<?php
$var1 = 21-4*4;
echo "The result of 21-4*4 is $var1.";
?>
```

Figure 8.1 displays the output of the script.

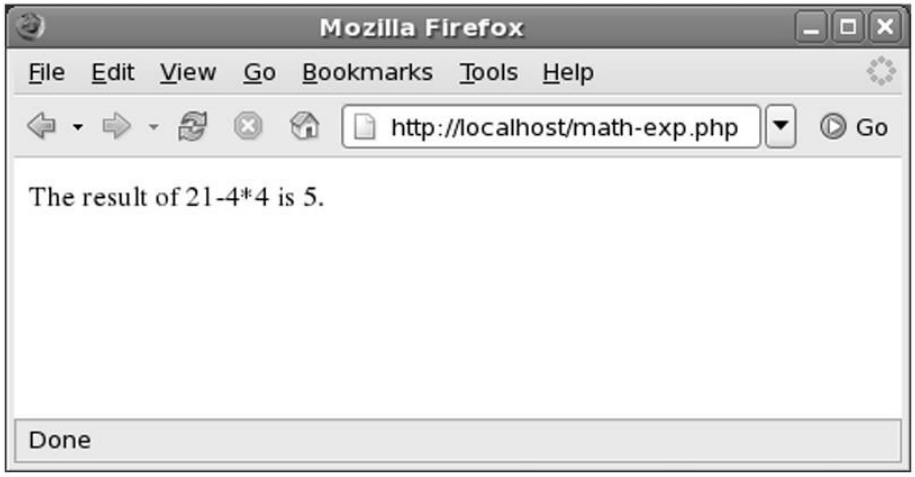


Figure 8.1: Mathematical Expressions in PHP

In the code, if you follow the order in which operands appear, the result is 68. However, if the Brackets Order Division Multiply Add Subtract (BODMAS) rule of mathematics is followed, the result will be 5. PHP follows the BODMAS rule for calculation.

Note: BODMAS is a precedence order followed while evaluating an expression.

Session 8

Table 8.1 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 8.1: Arithmetic Operators in PHP

To display the use of arithmetic operators, enter the code as shown in Code Snippet 2, in a PHP script named **`sumarithmatic.php`**.

Code Snippet 2:

```
<?php
$VAR1=5;
$VAR2=10;
//Addition
$SUM=$VAR1+$VAR2;
//Subtraction
$DIFFERENCE =$VAR1-$VAR2;
//Multiplication
$PRODUCT = $VAR1*$VAR2;
//Division
$QUOTIENT =$VAR1/$VAR2;
//Modulus
$REMAINDER = $VAR1%$VAR2;
echo "Addition of 5 and 10 is ".$SUM."<br>";
echo "Subtraction of 10 from 5 is ".$DIFFERENCE."<br>";
echo "Multiplication of 5 and 10 is ".$PRODUCT."<br>";
echo "Division of 5 and 10 is ".$QUOTIENT."<br>";
echo "Modulus of 5 and 10 is ".$REMAINDER."<br>";
?>
```

Figure 8.2 displays the output of the script.

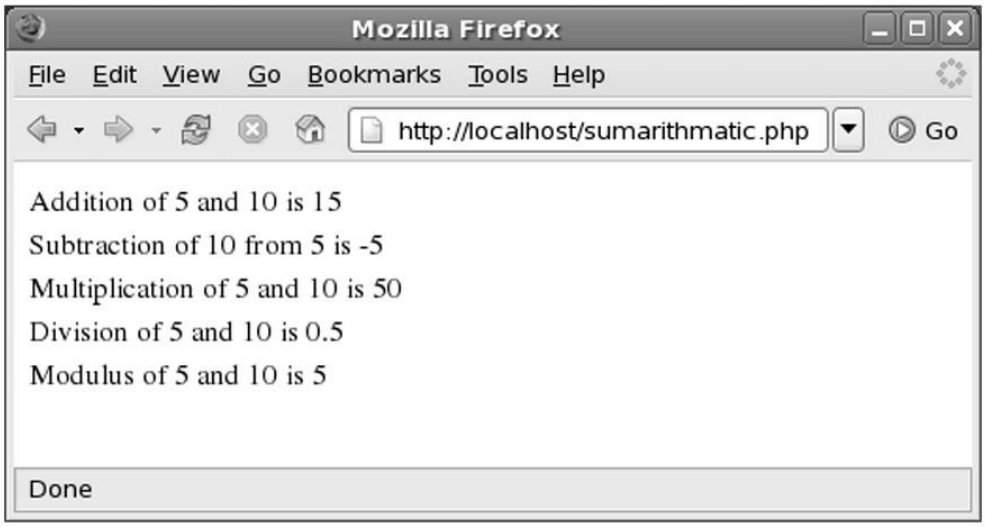


Figure 8.2: Using Arithmetic Operators

8.4 Relational Operators

Relational operators are also known as comparison operators. A relational operator compares two operands and returns either a `true` or a `false` value. In other words, it helps to determine the relationship between two operands. The operands can either be numbers or string values. Table 8.2 lists the relational operators supported in 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 (Introduced in PHP 4)
!=	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 (Introduced in PHP 4)
<	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

Session 8

Operator	Name	Description
>	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 8.2: Relational Operators in PHP

Note: When an integer is compared with a string, the string is first converted to a numeric value, and two numerical strings are compared as integers.

Code Snippets 3 and 4 show the difference between '== 'equal and '===' identical relational operators in PHP.

Code Snippet 3:

```
<?php
if("10" == 10)
    echo "YES";
else
    echo "NO";
?>
```

Figure 8.3 displays the output of the script.

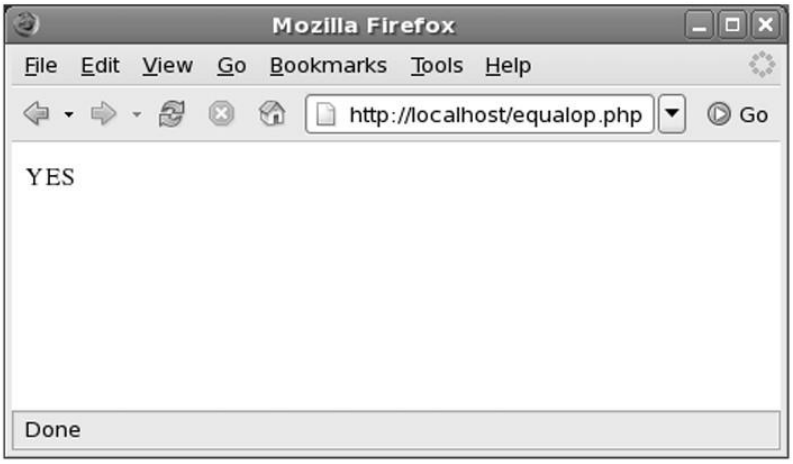


Figure 8.3: Using == Relational Operator

The code will print `YES` because the values of the operands are equal.

Concepts

Code Snippet 4:

```
<?php
if("10" === 10)
    echo "YES";
else
    echo "NO";
?>
```

Figure 8.4 displays the output of the script.

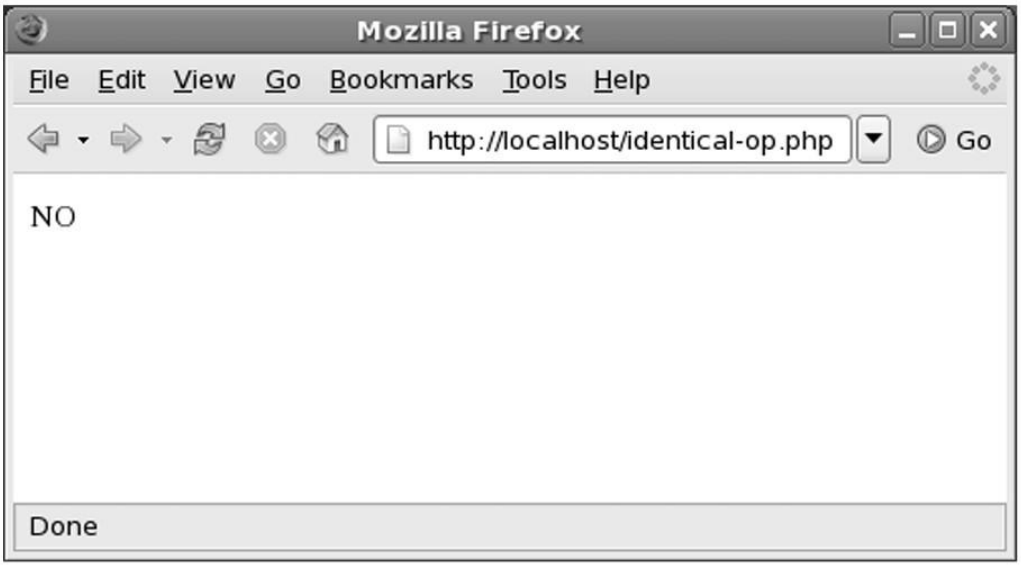


Figure 8.4: Using === Relational Operator

The code will print `NO` because although values of both operands are same, their data types are different. "10" is a string while 10 is an integer.

To check whether the given year is a leap year or not, using relational operators, enter the code as shown in Code Snippet 5, in a script named **`relational-op.php`**.

Code Snippet 5:

```
<?php
$y = 2011;
if(($y%4==0 && $y%100!=0) || ($y%400 == 0))
{
    echo "$y is a leap year. <br />";
}
else
{
    echo "$y is not a leap year. <br />";
}
?>
```

Figure 8.5 displays the output of the script.

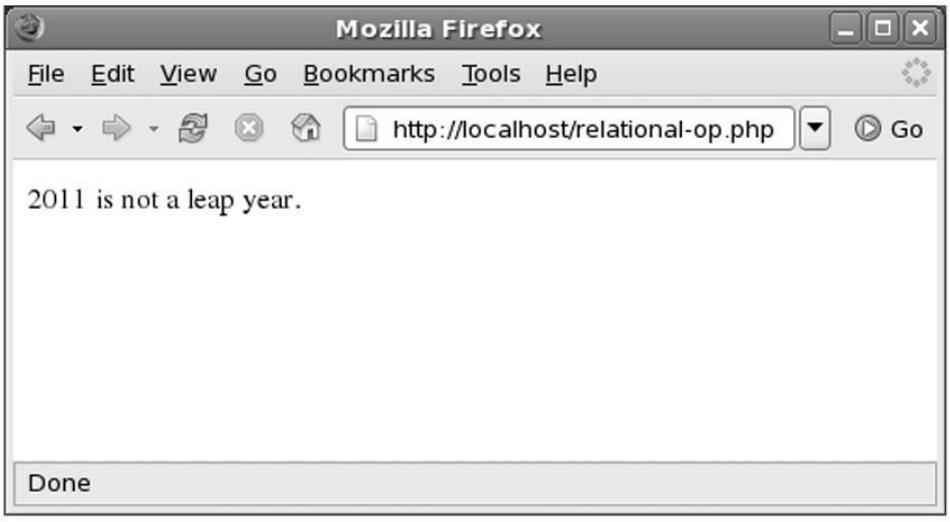


Figure 8.5: Using != Relational Operator

The code uses an `if` conditional statement to execute a block of code only when the specified condition is true else it executes the block of code in the body of the else statement. The output of the code will be `2011 is not a leap year.`

8.5 Logical Operators

Logical operators enable to combine two or more expressions in a condition. The operands are first converted to `Boolean` values before they are compared.

Session 8

The logical operator then evaluates the expression and returns a `Boolean` value of either true or false. Table 8.3 lists the logical operators supported by PHP.

Operator	Name	General Form	Description
`AND` `&&`	Logical AND operator	Expression1 `AND` Expression2 Expression1 `&&` Expression2	Returns true only if both the expressions are true
`OR` `||`	Logical OR operator	Expression1 `OR` Expression2 Expression1 `||` Expression2	Returns true if any one of the expression 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	`!`Expression	Returns true only if the condition is not true

Table 8.3: Logical Operators in PHP

The following examples explain the use of logical operators:

- To use a logical `AND` operator to check if a student's percentage is greater than 60 and the year of passing is 2003, enter the code as shown in Code Snippet 6, in a PHP script named **`logical _ and.php`**.

  Code Snippet 6:

```
<?php
$Percentage = 70;
$Year = "2003";
if ($Percentage>60 AND $Year=="2003")
{
    echo "Eligible for admission.";
}
?>
```

Session 8

Figure 8.6 displays the output of the script.

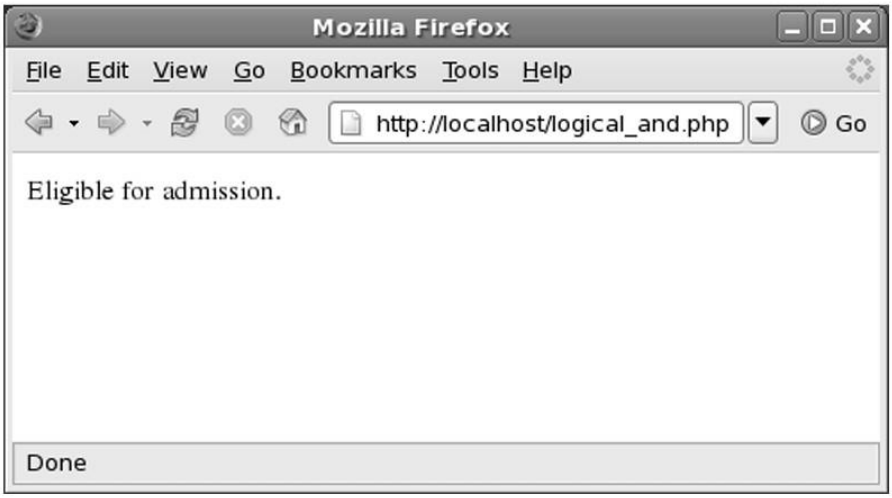


Figure 8.6: Using Logical AND Operator

In the code, the `AND` operator requires both the expressions to be true. Only then, the block of code following the `if` statement is executed.

➢ To display the use of the `OR` operator, enter the code as shown in Code Snippet 7, in a PHP script named, **`logical _ or.php`**.

Code Snippet 7:

```
<?php
$day1="Saturday";
if(($day1=="Saturday")||($day1=="Sunday"))
{
    echo "$day1 is a holiday.";
}
else
{
    echo "$day1 is a working day.";
}
?>
```

Concepts

Figure 8.7 displays the output of the script.

Figure 8.7: Using OR Logical Operator

In the code, the `OR` operator requires any one of the expressions to be true, and executes the block of code after the `if` statement.

8.6 Bitwise Operators

Bitwise operators operate on the bitwise representation of their operands. They are similar to the logical operators and work on small-scale binary representation of data. If the operand is a string, the operation is performed only after converting it to its corresponding integer representation. If both the operands are strings, the bitwise operator will first convert them to their ASCII values and operate between corresponding character offsets of the two strings.

Table 8.4 lists the bitwise operators supported by PHP.

Operator	Name	General Form	Description
&	AND	Operand1 & Operand2	Compares two bits and sets the result to 1 if both the bits are 1 and 0 otherwise
|	OR	Operand1 | Operand2	Compares two bits and sets the result to 1 if either of the bits are 1 and 0 otherwise
^	EXCLUSIVE-OR	Operand1 ^ Operand2	Compares two bits and sets the bit to 1 if the bits are different and 0 otherwise

Session 8

Operator	Name	General Form	Description
~	COMPLEMENT	~ Operand	Compares and sets the 0 bits to 1 and vice versa
<<	SHIFT LEFT	Operand1 << Operand2	Shifts the bits of Operand1, Operand2 steps to the left (each step means "multiply by two")
>>	SHIFT RIGHT	Operand1 >> Operand2	Shifts the bits of Operand1, Operand2 steps to the right (each step means " divide by two")

Table 8.4: Bitwise Operators in PHP

Table 8.5 displays the result of each bitwise comparison.

X	Y	X & Y	X | Y	~ X	~ Y	X ^ Y
1	1	1	1	0	0	0
1	0	0	1	0	1	1
0	1	0	1	1	0	1
0	0	0	0	1	1	0

Table 8.5: Binary Comparison of Bitwise Operators

Note: If the operands are strings, they are first converted to their integer representation. Then a bitwise operation is performed between two corresponding characters in the operands to give the resultant string. If each operand string varies in length, the result string will be truncated to the length of the shorter operand.

To display the use of bitwise operations, enter the code as shown in Code Snippet 8, in a PHP script named **`bitwise.php`**.

Code Snippet 8:

```
<?php
$x= 50;
$y= 5;
echo "\$x & \$y = ".($x & $y)."<br>";
echo "\$x | \$y = ".($x | $y)."<br>";
echo "\$x ^ \$y = ".($x ^ $y)."<br>";
echo "~(\$y) = ".~$y."<br>";
//x is divided by 2 y times
echo "\$x >> \$y = ".($x >> $y)."<br>";
```

Concepts

```
//x is multiplied by 2 y times
echo "\$x << \$y = ".($x << $y)."<br>";
//Converts the operands to their ASCII values first ('5'(ascii 53))^ ('9'
(ascii 57))
echo "The Bitwise result of 5 ^ 9 is:".(5 ^ 9)."<br>";
//Converts "8" to perform the operation (5 ^ ((int)"8")
echo "The result of 5 ^ 8 is: ".(5 ^ 8). "<br>";
?>
```

Figure 8.8 displays the output of the script.

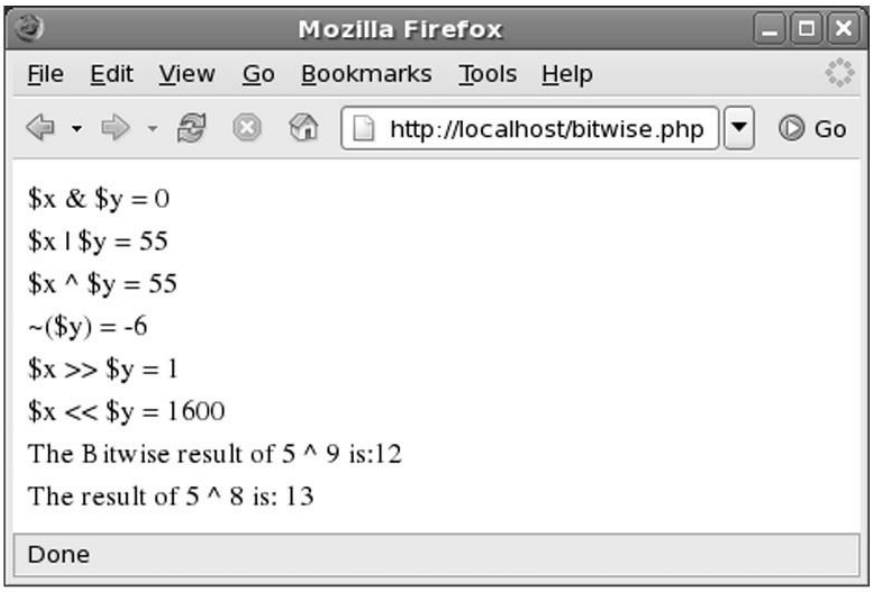


Figure 8.8: Using Bitwise Operators

8.7 Assignment Operators

Assignment operators enable to assign a value to a variable. The '=' sign is the assignment operator. It is different from the equal to '==' sign used in the relational operator.

The syntax for an assignment statement is as follows:

Syntax:

```
expression 1 =  expression 2;
```

Session 8

The value of `expression 2` is assigned to the variable in `expression 1`. The operand on the left hand side of the assignment operator '=' should be a variable. The assignment operation can be performed either by value or by reference. Assignment by value copies the value of `expression 2` and assigns it to `expression 1`. Assignment by reference, assigns a reference of `expression 2` to `expression 1`.

For example, the expression `$a = $b` assigns the value of `$b` to `$a`, and the expression `$a = &$b` sets `$a` to reference `$b`.

The basic assignment operator can also be used as a combined operator in conjunction with arithmetic and string operators. The combined operators are known as shorthand operators.

The list of shorthand operators are as follows:

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

Table 8.6 displays examples with the combination operators.

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

Table 8.6: Shorthand Operators

Session 8

PHP Operators

Concepts

To use shorthand operators, enter the code as shown in Code Snippet 9, in a PHP script named **assign_op.php**.

Code Snippet 9:

```
<?php
$a = 30;
$b = 6;
$a+= $b;
echo "The value of $a += $b is ". $a ."<br>";
$a-= $b;
echo "The value of $a -= $b is ". $a ."<br>";
$a*= $b;
echo "The value of $a *= $b is ". $a ."<br>";
$a/= $b;
echo "The value of $a /= $b is ". $a ."<br>";
$a%= $b;
echo "The value of $a %= $b is ". $a ."<br>";
$a.= " days in the month of June";
echo "The value of $a is: " . $a ."<br>";
?>
```

Figure 8.9 displays the output of the script.

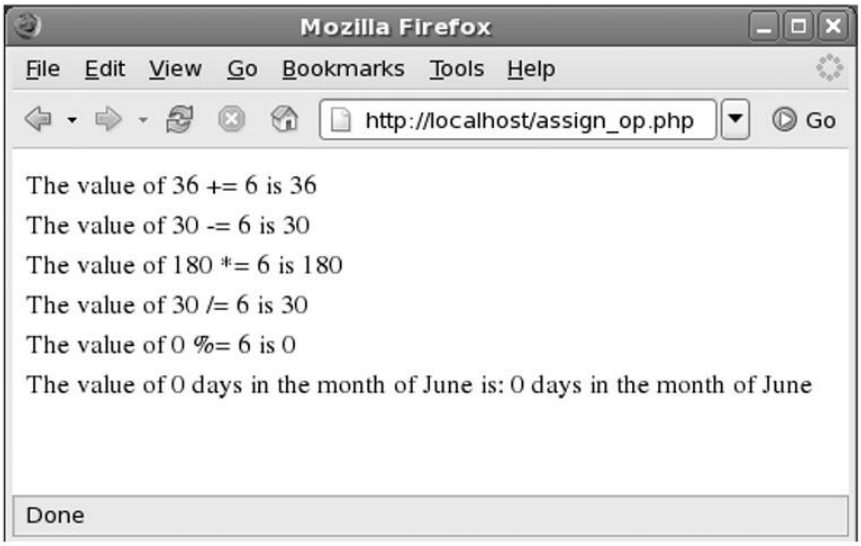


Figure 8.9: Using Assignment Operators

8.8 Increment and Decrement Operators

Increment and decrement operators operate only on variables. Apart from calculating the result value, these operators cause a variable to change its value as well. The increment operators increase the value of the operand by one. The decrement operators decrease the value of the operand by one. These operators are used mostly within conditional looping statements such as, `for`, `do while`, and `while`. Table 8.7 lists the increment and decrement operators supported by PHP.

Operand	Operator Name	Description
`++$a`	Pre-increment	Increments the operand value by one and then returns this new value to the variable
`$a++`	Post-increment	Returns the value to the variable and then increments the operand by one
`--$a`	Pre- decrement	Decrements the operand by one and then returns this new value to the variable
`$a--`	Post decrement	Returns the value to the variable and then decrements the operand by one

Table 8.7: Increment and Decrement Operators in PHP

To display the use of the increment and decrement operators on a variable `$A`, enter the code as shown in Code Snippet 10, in a PHP script named **`inc_dec.php`**.

Code Snippet 10:

```
<?php
$A=15;
echo "Pre increment value of A is ".++$A."<br/>";
echo "Post increment value of A is ".$A++."<br/>";
echo "Value of A post increment is ".$A."<br/>";
echo "Pre decrement value of A is ".--$A."<br/>";
echo "Post decrement value of A is ".$A--."<br/>";
echo "Value of A post decrement is ".$A--."<br/>";
?>
```

Concepts

Figure 8.10 displays the output of the script.

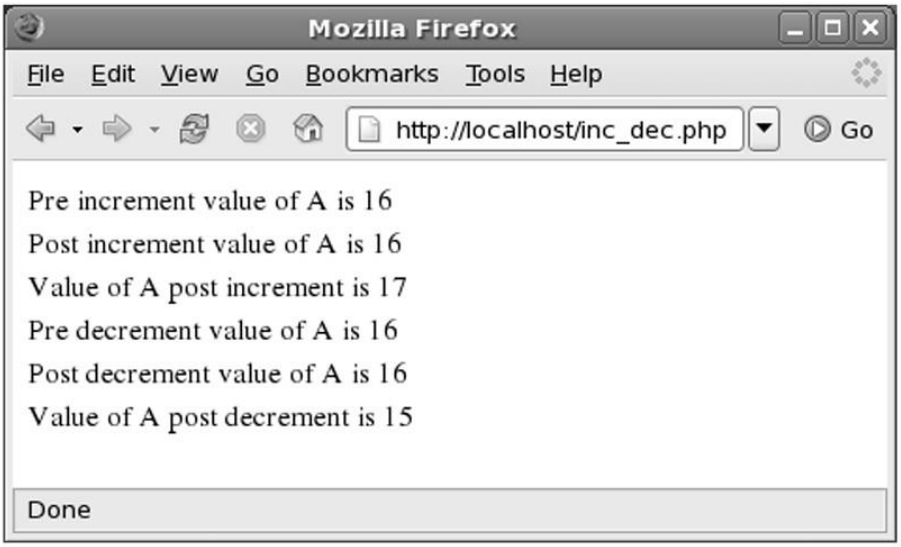


Figure 8.10: Using Increment and Decrement Operators

When working with increment operations on strings or character variables, PHP follows Perl's convention. If the last letter is alphanumeric, it is incremented by one.

Note: Character variables or non-numeric strings can only be incremented and not decremented and only ASCII characters (a-z and A-Z) are supported.

To display the use of increment operator on a character variable, enter the code as shown in Code Snippet 11, in a PHP script named **`inc_op.php`**.

Code Snippet 11:

```
<?php
$i = 'X';
for ($n=0; $n<5; $n++)
{
echo ++$i . "\n";
}
?>
```

Figure 8.11 displays the output of the script.

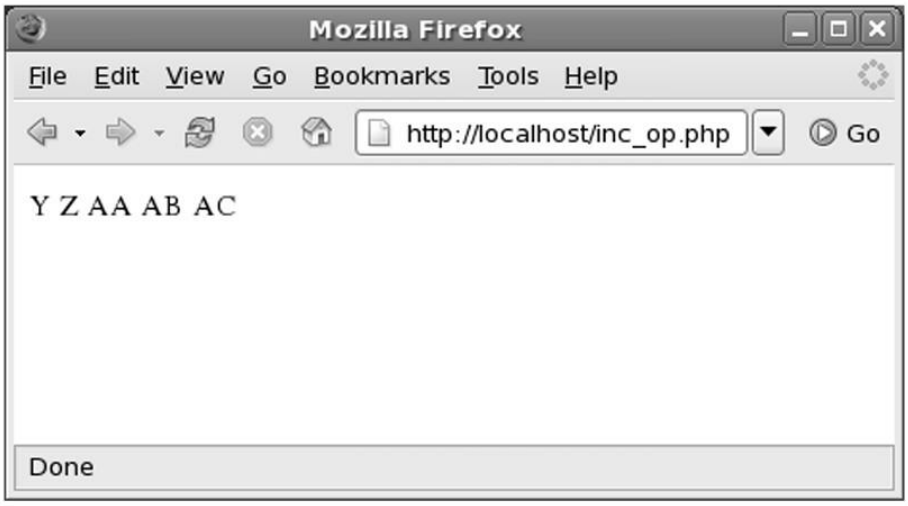


Figure 8.11: Displaying Five Characters after 'X'

In the code, the value of the variable `$i` is incremented five times using a `for` loop statement and each value is printed. Here `++$i` , i.e., `'X'+1` returns `'Y'` followed by `'Z'` and `'Z'+1` returns `'AA'` and `'AA'+1` returns `'AB'`.

8.9 String Operators

The string operators operate only on character data and any non-string operand is first converted before the operation is executed. Table 8.8 lists the string operators supported by PHP.

Operator	Name	Description
.	Concatenation	Returns a concatenated string
.=	Concatenating assignment	Appends the argument on the right side of the operator to the arguments on the left hand side

Table 8.8: String Operators in PHP

The following examples display the use of string operators in PHP:

- To display the concatenation of the strings WELCOME and FRIENDS using the Concatenation operator, enter the code as shown in Code Snippet 12, in a PHP script named **`concat.php`**.

Session 8

Code Snippet 12:

```
<?php
$A="WELCOME ";
$B="FRIENDS!";
$C=$A.$B;
echo "The concatenated string is $C";
?>
```

Figure 8.12 displays the output of the script.

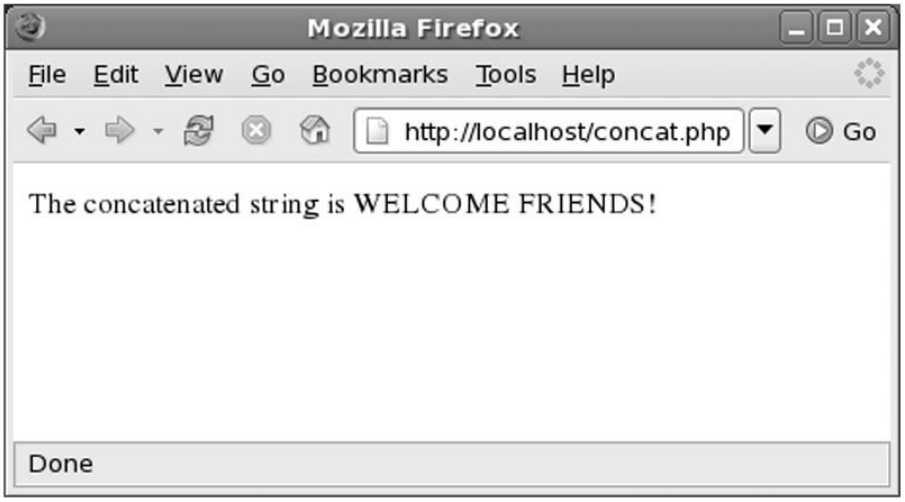


Figure 8.12: Using String Concatenating Operator

In the code, the values of the variables `$A` and `$B` are concatenated and stored in a variable `$C`. It returns the string `WELCOME FRIENDS!` as the output.

- To assign a value to a variable using the concatenating assignment operator, enter the code as shown in Code Snippet 13, in a PHP script named **`assign _ concat.php`**.

Code Snippet 13:

```
<?php
$A="WELCOME";
$A.= " FRIENDS!";
echo " The concatenated string is $A";
?>
```

Figure 8.13 displays the output of the script.

Figure 8.13: Using String Concatenating Assignment Operator

In the code, the argument `FRIENDS!` is appended on the right side of `$A` containing the argument `WELCOME`. The string returned is `WELCOME FRIENDS!`.

8.10 Conditional or Ternary Operators

A conditional operator is an alternative to the `if-else` statement. It evaluates an expression for a `true` or `false` value and then executes one of the two statements depending upon the result of evaluation.

The syntax for the conditional operator is as follows:

Syntax:

```
$var1 = ($var2 = value1) ? expr1 : expr2
```

The operator evaluates the `$var1` and if it is true, `expr1` is evaluated and if it is false, `expr2` is evaluated.

Concepts

To display the use of a conditional operator, enter the code as shown in Code Snippet 14, in a PHP script named **condition.php**.

Code Snippet 14:

```
<?php
$age = 15;
$category = ($age < 16) ? 'Child' : 'Adult';
echo "The result of the conditional operator is: ";
echo $category;
?>
```

Figure 8.14 displays the output of the script.

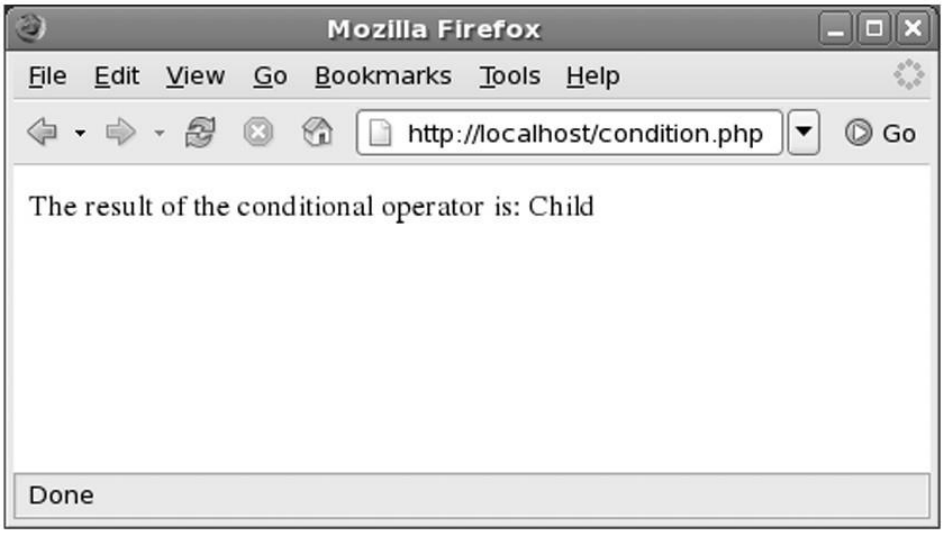


Figure 8.14: Using Conditional Operators

The code evaluates the condition `($age < 16)`. If it is true, then the value `Child` is assigned to `$category`, else the value `Adult` is assigned to `$category`.

8.11 Precedence of Operators

A complex expression can contain different operators. In such cases, PHP follows the order of precedence to evaluate the expression. The operators with higher precedence are evaluated first in an expression. However, operator precedence can be overridden by using parenthesis. The expression within the parenthesis is evaluated first. For example, (1 + 2) * 3 evaluates to 9. In addition, when two operators of the same precedence are encountered, the expression will be evaluated from left to right.

Session 8

PHP Operators

Table 8.9 lists the precedence of operators in PHP, with the highest precedence operators listed at the top. Operators with the same precedence are evaluated based on their associativity.

Precedence	Category	Operators	Associativity
	Unary	`!, ++ ,--`	Right to left
Highest Precedence	Multiplicative	`*, /, %`	Left to right
	Additive	`+, - , .`	Left to right
	Relational	`<, <=, >, >=`	Left to right
	Equality	`==, !=`	Left to right
	Logical AND	`&&`	Left to right
	Logical OR	`||`	Left to right
	Conditional	`?:`	Right to left
	Assignment	`=,+=,-=,*=,/=,%=`	Right to left
	Logical AND	`AND`	Left to right
	Logical XOR	`XOR`	Left to right
	Logical OR	`OR`	Left to right
Lowest Precedence	Comma	`,`	Left to right

Table 8.9: Precedence of Operators in PHP

The operators listed on the same row have the same precedence.

For example, consider the expression:

```
$A>60 OR $B==20
```

First the variables `$A` and `$B` are evaluated, whether they satisfy the required condition. Then the `OR` operator is used.

Session 8

Concepts

Summary

- Operator is any symbol that performs an operation on an operand.
- An operator enables to work on variables, strings, and numbers and control the program flow.
- The types of operators supported in PHP are arithmetic, logical, relational, bitwise, assignment, string, conditional, and increment and decrement operators.
- The arithmetic operators work with numbers and are used to execute mathematical operations. PHP follows the BODMAS rule for operator precedence.
- The relational operators compare two operands and determine the relationship between operands.
- The logical operators evaluate multiple conditions and combine two or more test expression in a condition, returning a Boolean value.
- The Bitwise operators enable comparison and manipulation of operands and operate on the bits of an operand.
- The assignment operator enables assignment of values to a variable and defines the operand on the left-hand side to the value on the right-hand side.
- The increment and decrement operators enable to increase or decrease value of an operand by one.
- The conditional or ternary operator is an alternative to the if-else statement. It evaluates a condition and executes one of the two statements depending on the true or false result of the condition.
- The concatenation operator combines two or more strings into a single string.
- In a complex expression, operator precedence indicates the order in which the operands must be evaluated. PHP evaluates operators with high precedence before operators with low precedence.

Check Your Progress

1. Which of the following arithmetic operator returns the remainder of a division operation?

   a. /

   b. %

   c. $

   d. \

2. _____ operators compare operands and determine relationship between operands.

   a. Relational

   b. Arithmetic

   c. Assignment

   d. Bitwise

3. The _____ operator compares Boolean values and returns true if either of the operand is true.

   a. ||

   b. |

   c. &

   d. !=

4. Which of the following operator is the concatenating assignment operator?

   a. .

   b. .=

   c. =.

   d. = =

Concepts

Check Your Progress

5. The _____ operator returns true only if all the expressions are true.

   a. AND

   b. OR

   c. XOR

   d. !

6. The << operator _____.

   a. shifts the bits of Operand2, Operand1 times to the right

   b. shifts the bits of Operand1, Operand2 times to the left

   c. shifts the bits of Operand1, Operand2 times to the right

   d. shifts the bits of Operand2, Operand1 times to the left

7. What will be the output of the following script?

```
<?php
$a = 5;
$b = &$a;
echo "$a=".$a. "<br>";
echo "$b=".$b. "<br>";
$a = 9;
echo "$a=".$a. "<br>";
echo "$b=".$b. "<br>";
   ?>
```

   a. 5=5 5=5 9=9 9=9

   b. 5=6 9=10

   c. 5=5 9=9

   d. 5 9

Check Your Progress

8. Which of the following expressions are true for the increment operation?

   a. x = x +2;

   b. x+=1;

   c. x =+1;

   d. x++;

9. What will be the output of the following script?

```
<?php
$b = 5;
$a = (( ++$b ) > 5 );
echo (int)$a;
echo "   ";
$b = 5;
$a = (( $b++ ) > 5 );
echo (int)$a;
?>
```

   a. 1 1

   b. 1 0

   c. 0 0

   d. 0 1

Session 8

PHP Operators

Concepts

Answers

1. b
2. a
3. a
4. b
5. a
6. b
7. a
8. a, b, d
9. b

Onlinevarsity

24 x 7
Access To
Learning

Session 9

PHP Operators (Lab)

Objectives

At the end of this session, the student will be able to:

- *Use Arithmetic operators.*
- *Use Relational operators.*
- *Use Assignment operators.*
- *Use Logical operators.*
- *Use Bitwise operators.*
- *Use String operators.*
- *Use Increment and Decrement operators.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Using Arithmetic Operators

Arithmetic operators are used to perform mathematical calculations. The arithmetic operators include operators for addition, subtraction, multiplication, division, and modulus.

For example, to create a form that accepts two numbers, and displays their product, using the PHP script, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form that accepts two numbers:**

```
<HTML>
<BODY>
<B>Multiplication of Two Numbers</B>
<FORM METHOD=GET ACTION = "multiplication.php">
```

Session 9

PHP Operators (Lab)


```
Enter the First Number
<INPUT TYPE="TEXT" NAME="n1text">
<BR>
Enter the Second Number
<INPUT TYPE="TEXT" NAME="n2text">
<BR>
<BR>
<INPUT TYPE="SUBMIT" VALUE="MULTIPLY"
<BR>
</BODY>
</HTML>
```

3. **Save the file as multiply.html in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the text editor.**

5. **Enter the following code to multiply the numbers:**

```
<?php
$A=$_GET['n1text'];
$B=$_GET['n2text'];
$C=$A*$B;
echo "The product of two numbers is $C.";
?>
```

6. **Save the file as multiplication.php in the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

Session 9

PHP Operators (Lab)

8. **Enter http://localhost/multiply.html in the Address bar and press Enter.**

   Figure 9.1 displays the output.

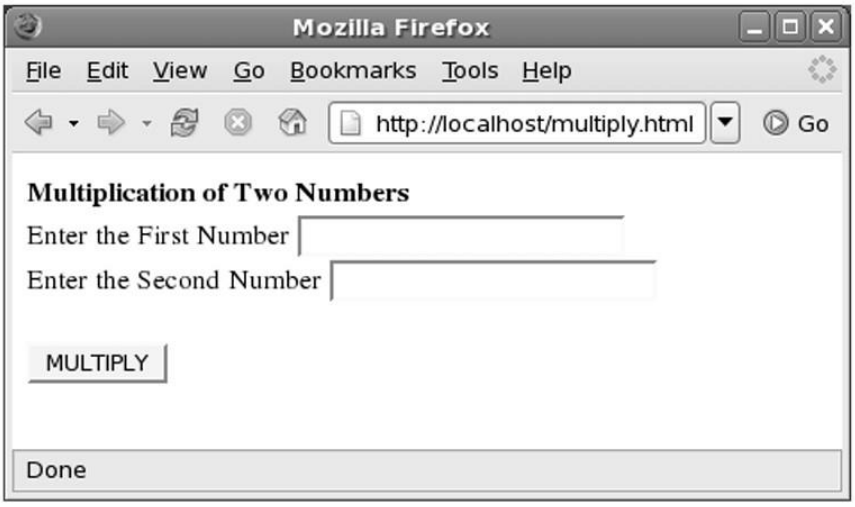


Figure 9.1: Multiplication Form

9. **Enter the first number as 76.**

10. **Enter the second number as 59.**

11. **Click MULTIPLY.**

    Figure 9.2 displays the output of the script.

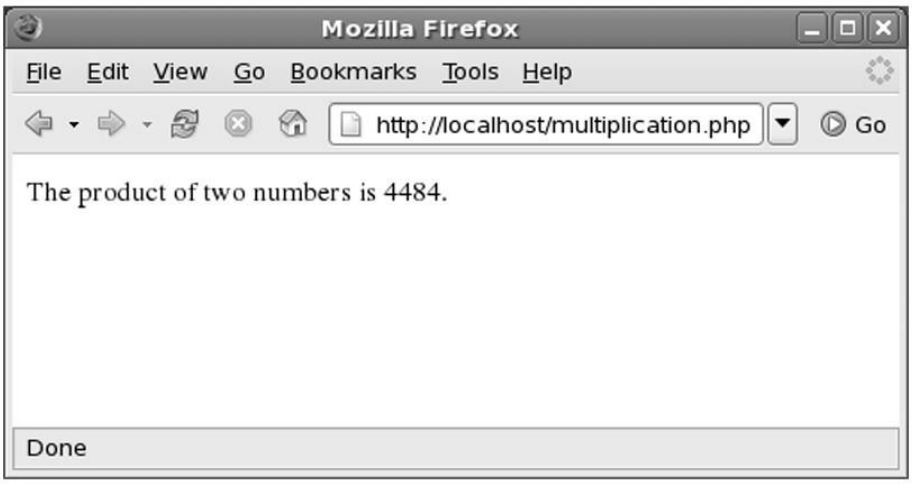


Figure 9.2: Displaying the Product

Using Relational Operators

A relational operator compares two operands to determine the relationship between the operands. Whenever two values are compared, the operator returns either true or false.

For example, to create a login form that accepts the username and password and authenticates the user, using the PHP script, perform the following steps:

1. **Open a new file in the text editor.**

2. **Enter the following code to create the authentication form:**

```
<HTML>
<BODY>
<H3>Sign In</H3>
<FORM METHOD=GET ACTION = "authentication.php">
Enter User Name
<INPUT TYPE="TEXT" NAME="NAME">
<BR><BR>
Enter Password   
```

Session 9

PHP Operators (Lab)


```
<INPUT TYPE="TEXT" NAME="passtext">
<BR><BR>
<INPUT TYPE="SUBMIT" NAME = "CONFIRM" VALUE="LOG IN">
<BR>
</BODY>
</HTML>
```

3. **Save the file as authentication.html in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to confirm the password:**

```
<?php
$A=$_GET['passtext'];
if($A=="pass")
{
  echo "You are a Valid User.";
}
else
{
  echo "Sorry, you are an Invalid User.";
}
?>
```

6. **Save the file as authentication.php in the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/authentication.html in the Address bar and press Enter.**

   Figure 9.3 displays the authentication form.

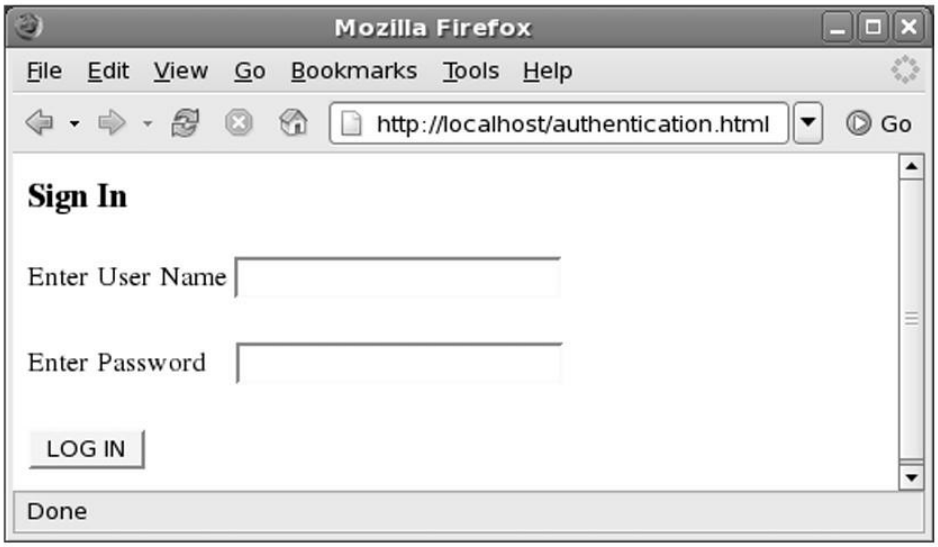


Figure 9.3: Authentication Form

9. **Enter John as the user name.**

10. **Enter pass as the password.**

Session 9

11. **Click LOG IN.**

    Figure 9.4 displays the output of the script.

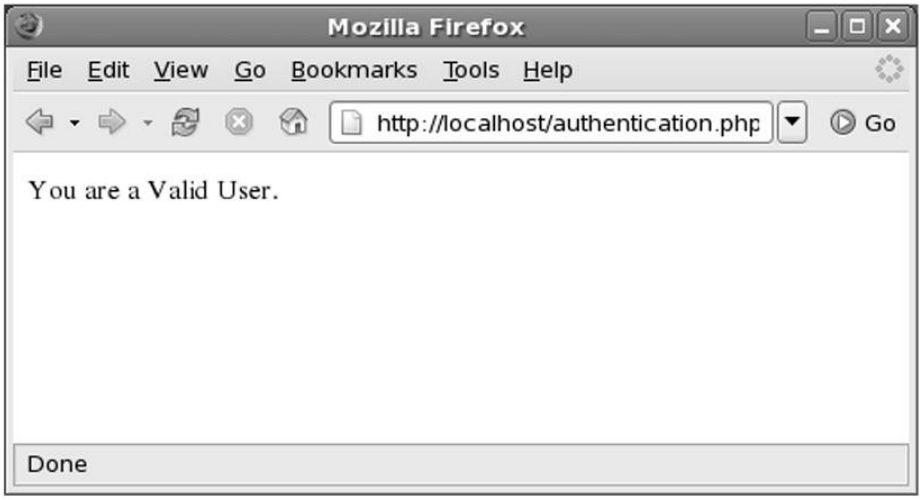


Figure 9.4: Using Relational Operators

Using Logical Operators

The Logical operators enable to combine two or more test expressions in a condition. They evaluate expressions and returns a Boolean value.

For example, to create a form that accepts the percentage scored by the student, and displays the grade depending upon it, using the PHP script, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create the form:**

```
<HTML>
<BODY>
<H3>Check Your Grade:</H3>
<FORM METHOD=GET ACTION = "grade.php">
ENTER YOUR PERCENTAGE  
<INPUT TYPE="TEXT" NAME="NAME" VALUE =">
<BR><BR>
```

Session 9

PHP Operators (Lab)

Lab Guide

```
<INPUT TYPE="SUBMIT" NAME="GRADE" VALUE="SUBMIT">
<BR>
</BODY>
</HTML>
```

3. **Save the file as grade.html in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to assign grades:**

```
<?php
$A = $_GET['NAME'];
if($A>60)
{
     echo "Congrats! ";
     echo "You have got Grade 1.";
}
else if($A < 60 AND $A > 40)
{
      echo "Congrats! ";
      echo "You have got Grade 2.";
}
else
{
     echo "Sorry. You have failed.";
}
?>
```

6. **Save the file as grade.php in the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

Session 9

8. **Enter http://localhost/grade.html in the Address bar and press Enter.**

   Figure 9.5 displays the output of the script.

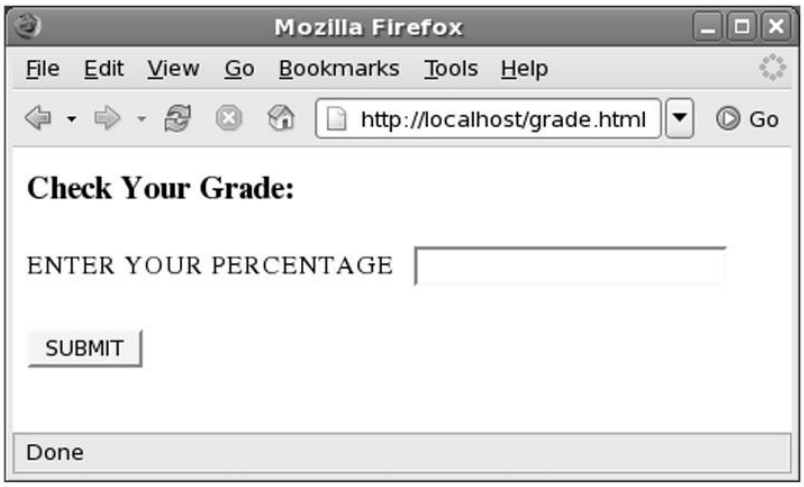


Figure 9.5: Grade Form

9. **Enter 70 in the percentage box.**
10. **Click SUBMIT.**

    Figure 9.6 displays the output of the script.

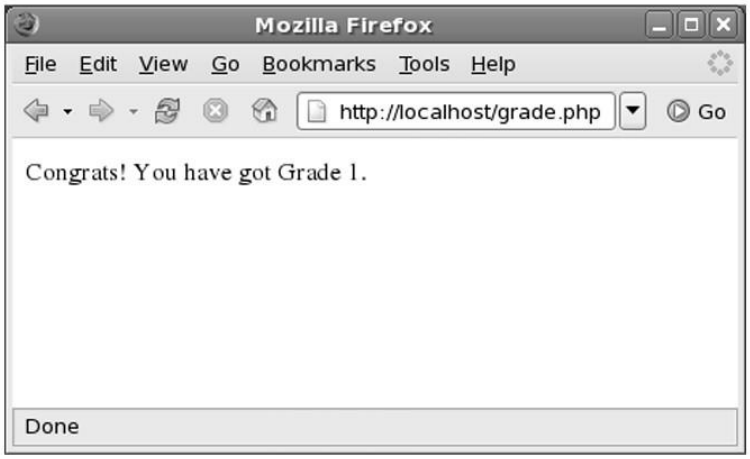


Figure 9.6: Using Logical Operators

11. **Similarly, enter 50 and 35 as the percentage scored and note the output.**

Session 9

PHP Operators (Lab)

Using Assignment Operators

The assignment operator enables to assign the operand on the left-hand side of the operator to the value of the expression given on the right-hand side. The assignment operator is represented by '=' sign. It is different from the equal to '==' sign used as the relational operator.

For example, to create a form that accepts the principal amount, interest rate, and number of years and calculates the simple interest using the PHP script, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form:**

```
<HTML>
<BODY>
<B>Simple Interest</B>
<FORM METHOD=GET ACTION = "simpleinterest.php">
Enter the Principal Amount:
<INPUT TYPE="TEXT" NAME="n1text">
<br><br>
Enter the Rate of Interest:    
<INPUT TYPE="TEXT" NAME="n2text">
<br><br>
Enter the Number of Years: 
<INPUT TYPE="TEXT" NAME="n3text">
<br><br>
<INPUT TYPE="SUBMIT" NAME = "CALCULATE" VALUE="CALCULATE">
<br>
</BODY>
</HTML>
```

3. **Save the file as simpleinterest.html in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to calculate the simple interest:**

```
<?php
$P=$_GET['n1text'];
$R=$_GET['n2text'];
```

```
$N=$_GET['n3text'];
$SI=($P*$R*$N)/100;
echo "The calculated simple interest is  $SI";
?>
```

6. **Save the file as simpleinterest.php in the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Web Firefox browser.**

8. **Enter http://localhost/simpleinterest.html in the Address bar and press Enter.**

   Figure 9.7 displays the output of the HTML form.

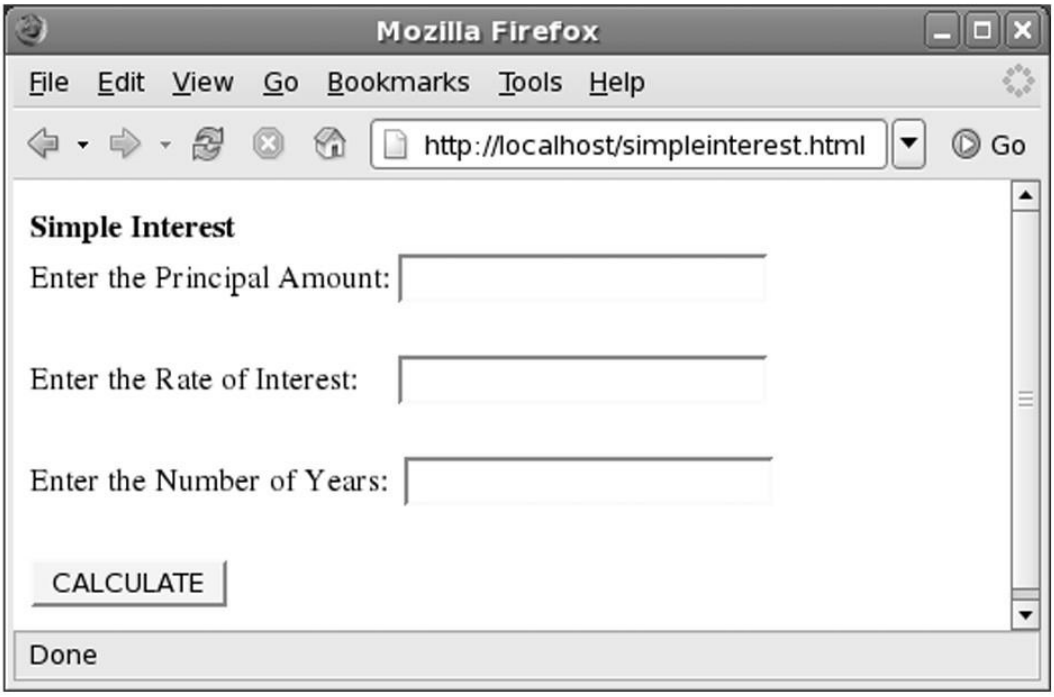


Figure 9.7: Simple Interest Form

9. **Enter 30000 in the Principal Amount box.**

10. **Enter 3.5 in the Rate of Interest box.**

11. **Enter 5 in the Number of Years box.**

12. **Click CALCULATE.**

    Figure 9.8 displays the output of the script.

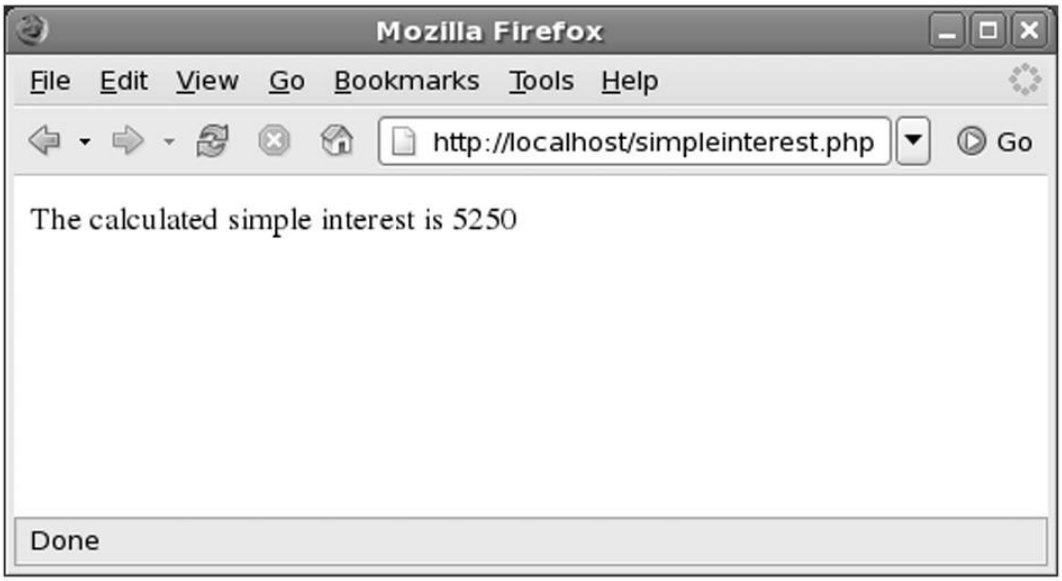


Figure 9.8: Using the Assignment Operator

Using Bitwise Operators

The bitwise operators operate on the bits of an operand. They are similar to the logical operators. They work on small-scale binary representation of data.

For example, to use the OR operator on 32 and 5, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
echo "The answer of 32|5 is:<BR> ";
echo 32|5;
?>
```

3. **Save the file as bitwise.php in the /usr/local/apache2/htdocs directory.**

Session 9

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/bitwise.php in the Address bar and press Enter.**

   Figure 9.9 displays the output of the script.

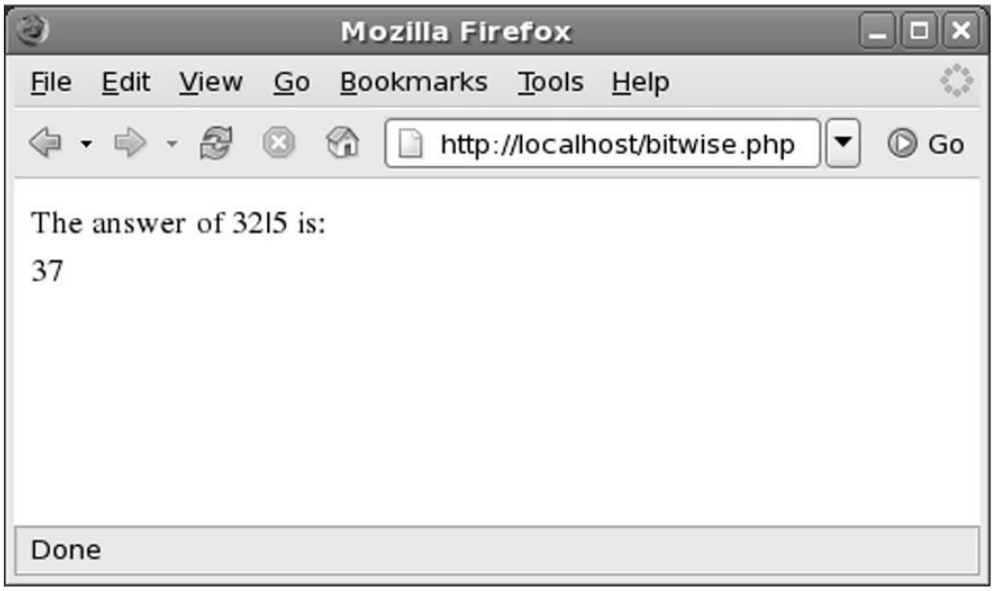


Figure 9.9: Using Bitwise Operator

Using String Operators

The string operators operate on character data. PHP provides two string operators to concatenate two or more strings.

For example, to append text to the existing string, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
$String = "HELLO";
$String.= " FRIENDS!";
echo $String;
?>
```

3. **Save the file as string.php in the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Web Firefox browser.**

5. **Enter http://localhost/string.php in the Address bar and press Enter.**

   Figure 9.10 displays the output of the script.

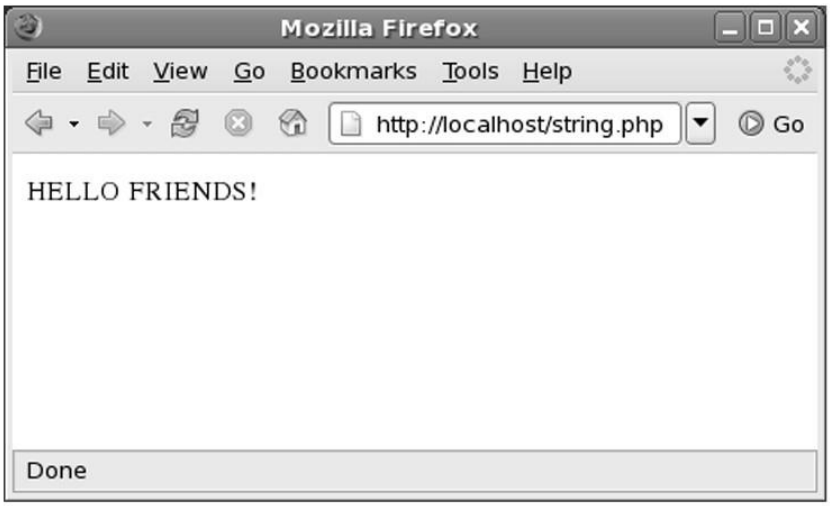


Figure 9.10: Using String Operators

Using Increment and Decrement Operators

The increment operator increases the value of the operand by one. The decrement operator decreases the value of the operand by one.

For example, to display the value of the variable before and after using the pre decrement operator, perform the following steps:

1. **Open a new file in the gedit text editor.**

Session 9

PHP Operators (Lab)

2. **Enter the following code:**

```
<?php
$A = 100;
echo "The value of A is $A.<BR>";
--$A;
echo "The decremented value of A is $A.";
?>
```

3. **Save the file as decrement.php in the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Web Firefox browser.**

5. **Enter http://localhost/decrement.php in the Address bar and press Enter.**

   Figure 9.11 displays the output of the script.

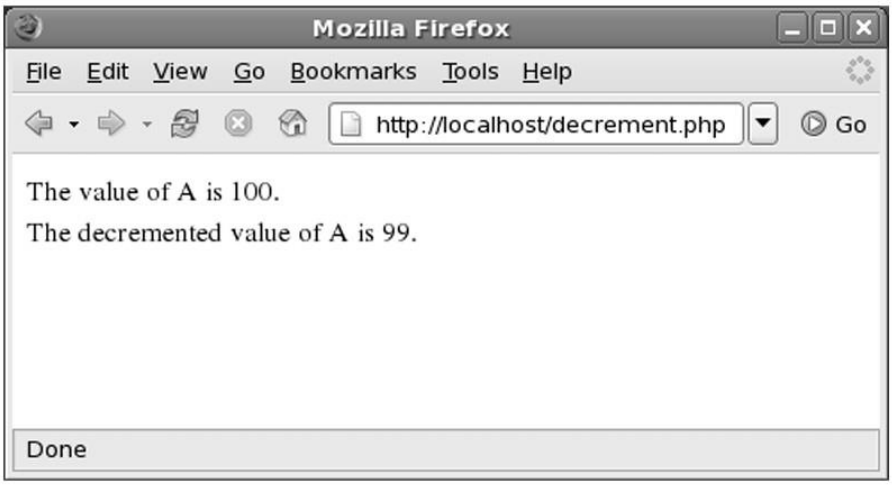


Figure 9.11: Using the Decrement Operator

Do It Yourself

1. Create a form that accepts the first name, last name, and the basic salary of the employee.

2. Write a PHP script to display the full name and basic salary of the employee.

3. Also, calculate and display Dearness Allowance (DA), House Rent Allowance (HRA), and Tax. The DA is calculated at 2% of the basic salary, while HRA is calculated at 5% of the basic salary. The tax is deducted from the salary based on the range of the basic salary as shown in table 9.1.

Range of the Basic Salary	Tax to be Deducted
4000-5000	250
5001-6000	500
6001 and above	700

Table 9.1: Salary and Tax Details

4. In addition, calculate the net salary of the employee using the following formula:

   Net Salary = Basic Salary + DA+ HRA - TAX

Session 10

Conditional Statements in PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the use of the if statement.*
- *Explain the use of the switch statement.*
- *Explain the use of the ternary (?) operator.*

10.1 Introduction

A statement is a smallest element of any programming language and normally consists of command given by a programmer to the computer. A PHP script consists of a series of statements. These statements can be an assignment, a function call, a conditional statement, or even an empty statement that does nothing. A statement usually ends with a semicolon and can be either an individual statement or a group of statements within curly braces. A group of statements is also considered as a statement by itself.

The statements in a program are executed in a sequence starting from the first to the last. However, conditional control structures can be used to change the order of control flow in a program. The conditional control structures control the flow of a program as they execute or skip code based on certain criteria.

In this session, you will learn how to use the different types of conditional statements, such as `if` and `switch` statement. In addition, you will also learn how to use the ternary operator in PHP.

10.2 Conditional Control Structures

Conditional control structures help to direct the flow of execution in a program at run time, based on a condition. The conditional control structures supported by PHP are as follows:

- `if` statement
- `switch` statement

10.2.1 The if Statement

The `if` statement is the most common conditional control structure in all programming languages. The expression in an `if` statement is called the truth expression. If the truth expression evaluates to true, the statement or the block of code following the `if` statement is executed and ignored otherwise.

Session 10

Conditional Statements in PHP

Concepts

The syntax for `if` statement is as follows:

Syntax:

```
if(truth expression)
{
   Statements to be executed;
}
```

The `if` keyword is followed by the truth expression in parentheses. The truth expression can be a Boolean variable, a constant, or an expression that evaluates to `TRUE`, `FALSE`, or `NULL`. If the truth expression evaluates to `TRUE`, the statements following the `if` statement are executed. If the truth expression evaluates to `FALSE` or `NULL` the statements are not executed.

Consider the code in Code Snippet 1 to check whether a triangle is valid using the `if` statement, in a script named **`validity.php`**.

Code Snippet 1:

```
<html>
<body>
<?php
$a=60;
$b=60;
$c=60;
if($a+$b+$c == 180)
echo "The triangle is valid.";
?>
</body>
</html>
```

Concepts

Figure 10.1 displays the output of the script.

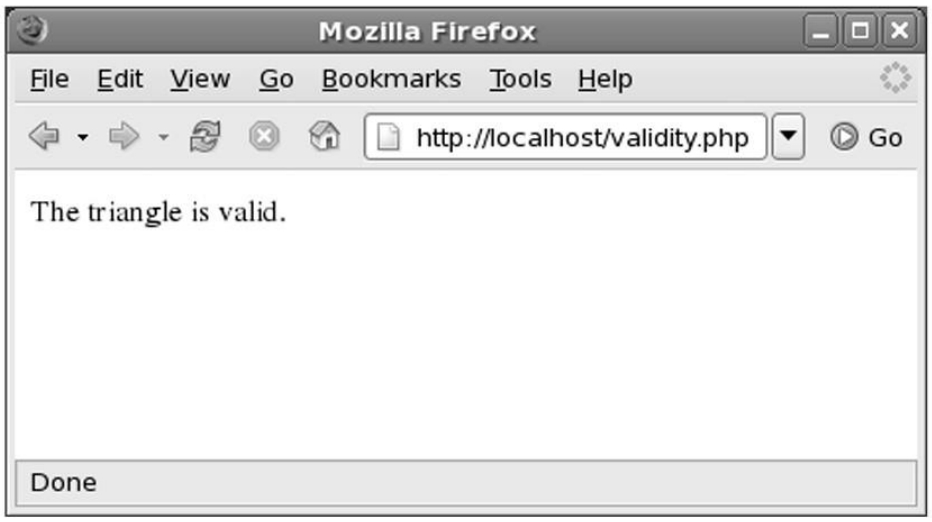


Figure 10.1: Using if Statement

In the code, if the sum of the degrees `$a`, `$b`, and `$c` are equal to `180` then the statement following the `if` condition is executed. If more than one statement is to be executed when a condition is `true`, enclose the body of the statements within curly braces.

Consider the code in Code Snippet 2 and Code Snippet 3 to accept and display the bonus at the rate of 10% and total salary when the salary of the employee is greater than $850, in a file named `salBonus.html`.

Code Snippet 2 accepts user inputs in a file named `salBonus.html` and Code Snippet 3 calculates the bonus and total salary in a file named **`salBonus.php`**.

Code Snippet 2:

```
<html>
<body>
<form action="salBonus.php" method="GET">
<table>
<tr>
<td>Salary   </td>
<td><input type="text" name="sal"></td>
</tr>
</table>
```

Session 10


```
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Figure 10.2 displays the output of the file.

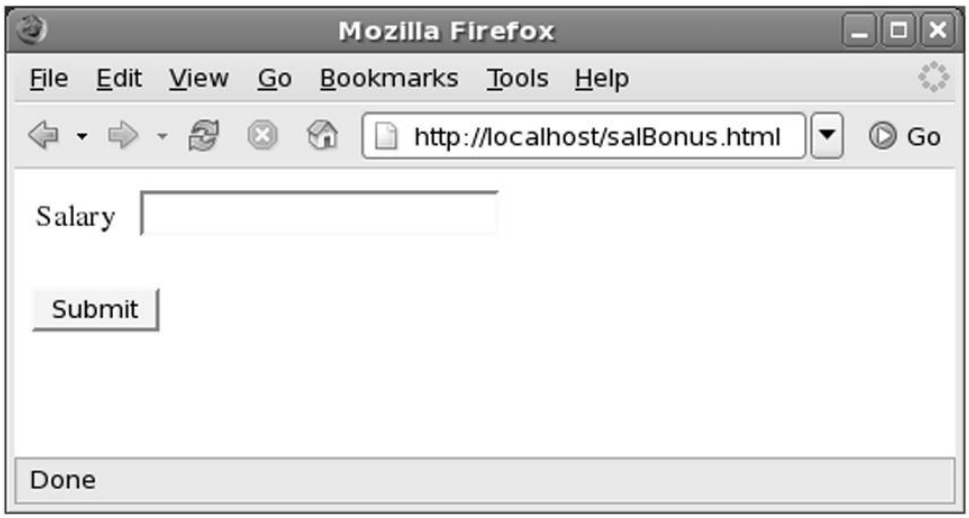


Figure 10.2: Accepting User Input

Consider the code in Code Snippet 3 to process the salary and calculate the bonus, in a script named **`salBonus.php`**.

Code Snippet 3:

```
<?php
$sal = $_GET['sal'];
echo "Salary before bonus : $";
echo $sal;
echo "<br>";
if ($sal > 850)
{
   $bonus = $sal * .1;
   echo "Bonus : $$bonus";
   echo "<br>";
```

```
    $sal = $sal + $bonus;
    echo "Total Salary : $$sal";
}
?>
```

Figure 10.3 displays the output of the script.

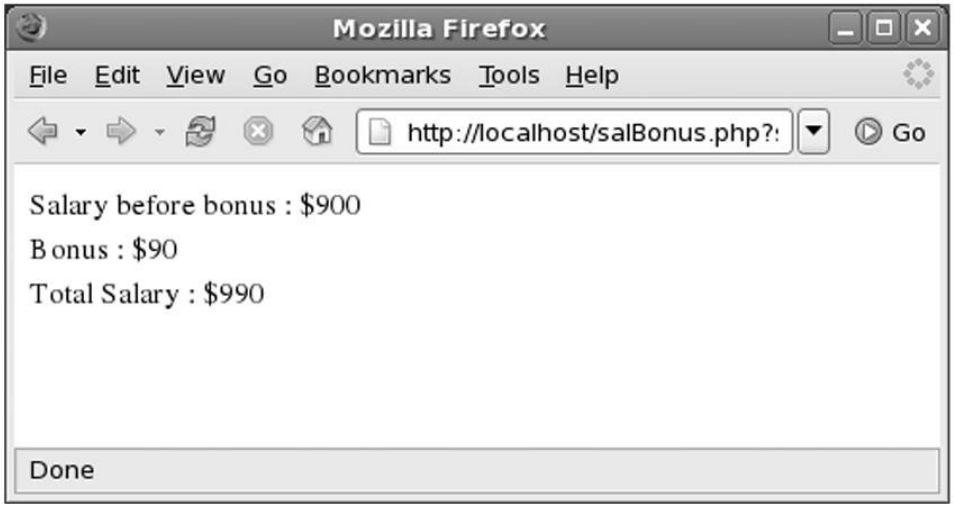


Figure 10.3: Using if Statement with Block of Code

In the code, if the salary of the employee is less than $850, the statements within the curly braces are not executed. The control jumps to the statement following the closing curly brace (`}`).

10.2.2 The if...else Statement

You can also execute a block of code when a specified condition is `false`. This can be done using the `else` statement. The `else` statement is used along with the `if` statement.

Session 10

Conditional Statements in PHP

Concepts

The syntax for `if…else` statement is as follows:

Syntax:

```
if(truth expression)
{
   Statements to be executed if the condition evaluates to true;
}
else
{
   Statements to be executed if the condition evaluates to false;
}
```

Code Snippet 4 displays a block of code with an `if...else` statement. This code calculates commission at the rate of 10% when the sale is greater than $2000 and 5% when the sale is less than $2000.

Code Snippet 4:

```
<?php
$sales = 2750;
if($sales > 2000)
{
   $comm = $sales * .1;
   echo "Sales: $$sales <br> Commission : $$comm";
}
else
{
   $comm = $sales * .05;
   echo "Sales: $$sales <br> Commission : $$comm";
}
?>
```

Figure 10.4 displays the output of the script.

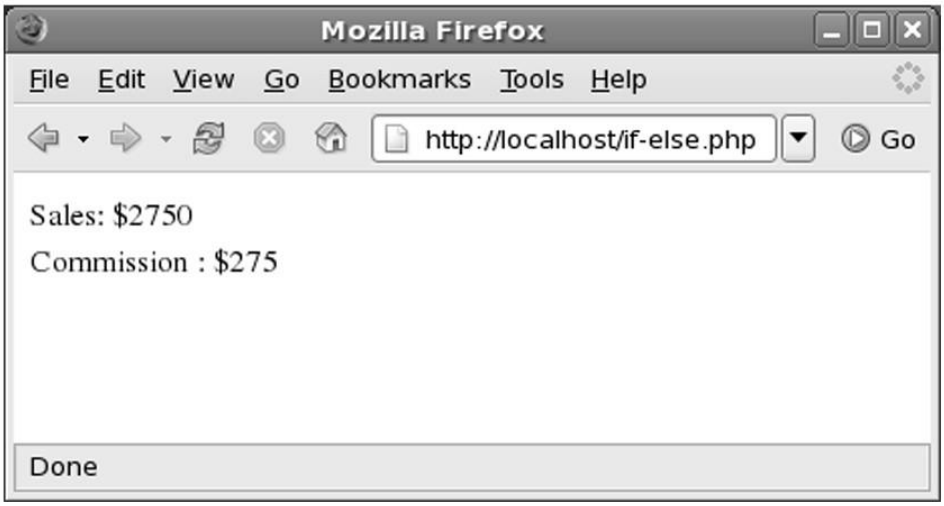


Figure 10.4: Using a Simple if...else Statement

In the code, the `$sales` variable stores the sales amount and the `$comm` stores the amount of commission. When the sales amount exceeds $2000, the program executes the body of the `if` statement and calculates commission at the rate of 10%. When the sales amount is less than $2000, the program executes the body of the `else` statement.

An `else if` clause can also be used along with the `if` statement. The `else if` clause is an optional clause that allows testing alternative conditions. The `else if` construct performs a series of checks against multiple conditions and only executes the code following the first condition that is met. It is executed before the `else` statement.

The following Code Snippet demonstrates the use of the `else if` clause while calculating a commission based on the rates specified in table 10.1.

Sales	Commission Rate
Greater than 50000	10%
Greater than 20000 and less than equal to 50000	7%
Lesser than 20000	5%

Table 10.1: Commission Rate Based on Sales

Session 10

Conditional Statements in PHP

Concepts

Code Snippet 5 uses an HTML form to accept the sales amount from the user, in a file named `SaleComm.html` and Code Snippet 6 calculates the commission using an `else if` construct, in a script named **`SaleComm.php`**.

Code Snippet 5:

```
<html>
<body>
<form action="SaleComm.php" method="GET">
<table>
<tr>
<td>Total Sales : </td>
<td><input type="text" name="sal"></td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Figure 10.5 displays the output of the script.

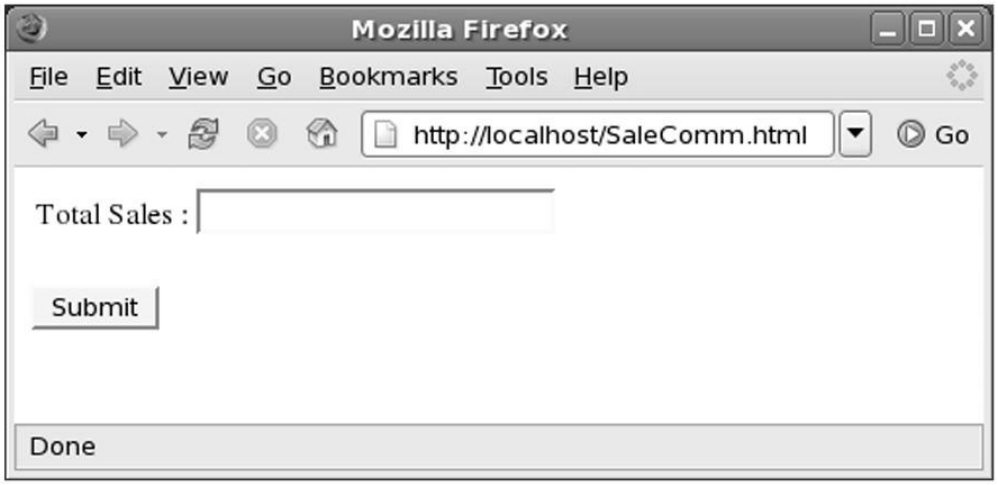


Figure 10.5: Accepting Sales Information

Session 10

Conditional Statements in PHP

Consider the code in Code Snippet 6 to process the sales amount and calculate the commission based on the value entered by the user, in a script named **`SaleComm.php`**.

Code Snippet 6:

```
<?php
$sal=$_GET['sal'];
echo "Total Sales : $";
echo $sal;
echo "<br>";
if ($sal > 50000)
{
   $comm = $sal * .10;
   echo "Commission : $$comm";
   echo "<br>";
}
else if ($sal > 20000 and $sal <= 50000)
{
   $comm = $sal * .07;
   echo "Commission : $$comm";
   echo "<br>";
}
else if ($sal < 20000)
{
   $comm = $sal * .05;
   echo "Commission: $$comm";
   echo "<br>";
}
?>
```

Concepts

Figure 10.6 displays the output of the script.

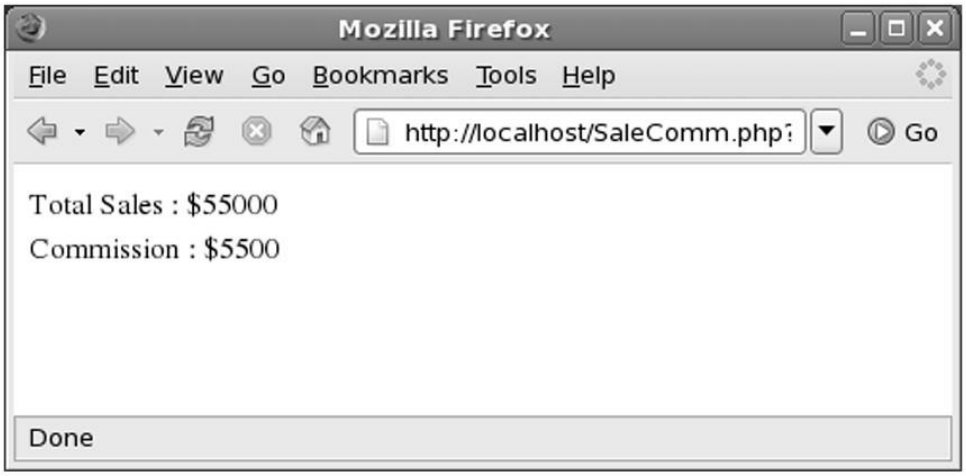


Figure 10.6: Using Multiple if..else Statements

In the code, commission is calculated according to the sales amount the user enters.

10.2.3 Nested if Statements

You can also include an `if` statement within an `if` statement or an `else` statement. These are known as nested `if` statements.

The following examples display the use of nested `if` statements, to calculate the electricity charges based on the units of electricity consumed as specified in table 10.2.

Units	Rate ($)
Above 1000	3
Greater than 500 but less than equal to 1000	2
Less than 500	1.5

Table 10.2: Electricity Charges Based on Units

In addition, a service charge of 10% is added when the amount exceeds 200.

Session 10

Code Snippet 7 uses an HTML form to accept the number of units consumed by the user in a file named `elecBill.html` and Code Snippet 8 calculates the electricity charges, in a file named **`elecBill.php`**.

Code Snippet 7:

```
<html>
<body>
<form action="elecBill.php" method="GET">
<table>
<tr>
<td>Electricity Units Consumed : </td>
<td><input type="text" name="units"></td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Figure 10.7 displays the output of the script.

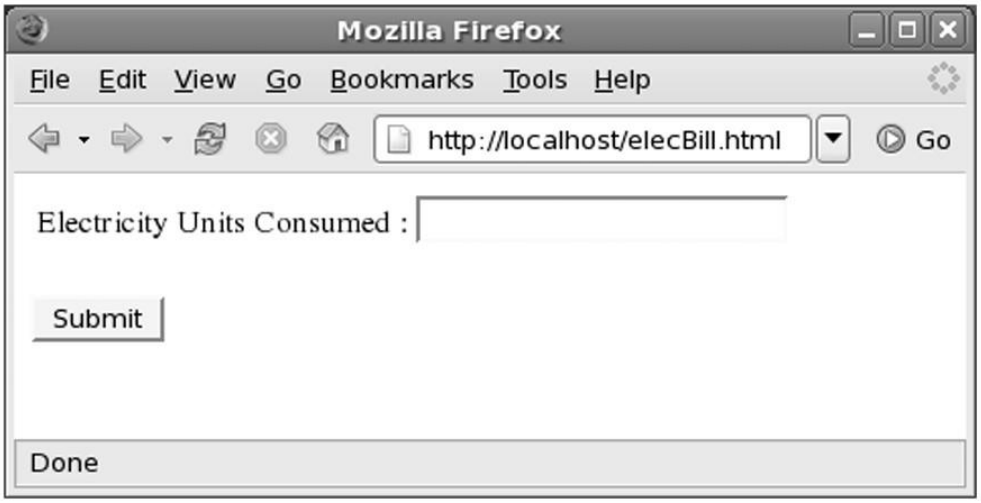


Figure 10.7: Using Nested if Statements

Session 10

Code Snippet 8 calculates total electricity bill, in a script named **elecBill.php**.

Code Snippet 8:

```
<?php
$units=$_GET['units'];
echo "Number of Units Consumed : ";
echo $units;
echo "<br>";
if ($units > 1000)
{
   $rate = $units * 3;
   $service = $rate * .1;
   echo "Service Charge added for Units above 1000 : $$service";
   echo "<br>";
   $totalbill = $rate + $service;
   echo "Total Electricity Bill : $$totalbill";
}
else
{
   if ($units > 500 and $units <= 1000)
    {
       $rate = $units * 2;
       echo "Total Electricity Bill : $$rate";
    }
    else
    {
       $rate = $units * 1.5;
       echo "Total Electricity Bill : $$rate";
    }
}
?>
```

Figure 10.8 displays the output of the script.

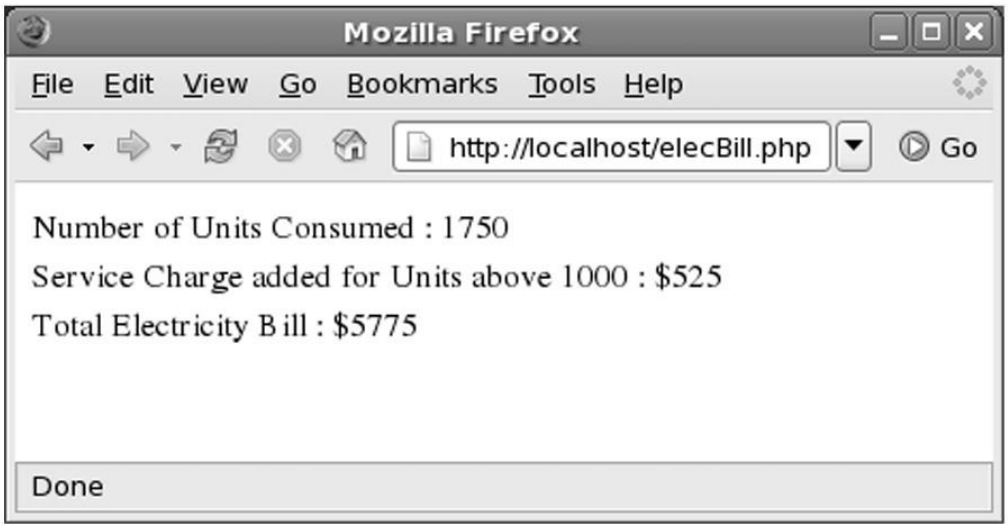


Figure 10.8: Another Example of Using Nested if Statements

In the code, if the user enters `1500` as input for Code Snippet 7, PHP code in Code Snippet 8 first calculates the rate and stores the value in the `$rate` variable. To calculate the service charge to be levied on the electricity bill, Code Snippet 8 calculates the service charge at the rate of 10% and stores the value in the `$service` variable. It then stores the total amount in the `$totalbill` variable.

If the user enters number of units as `400`, the total electricity bill would be calculated at the rate of `1.5`. The output will be, `Total Electricity Bill: $600`.

10.3 The switch Statement

A `switch` construct can be used as an alternative to a lengthy `if...else` construct. A `switch` statement consists of an expression that is compared to all possible case expressions listed in its body. On finding a match, it executes the block of code ignoring any further case lines. The comparison to find the right match is done internally using the equality operator (==) and not the identical operator (===). You can also use a `break` statement to halt the execution of the switch statement and transfer the control to the code following the `switch` construct.

Session 10

Conditional Statements in PHP

Concepts

The syntax for the `switch` statement is as follows:

Syntax:

```
switch(variable){
   case value1:
Code executes if condition equals value1
   break;
   case value2:
Code executes if condition equals value2
   break;
.
.
.
   default:
Code executes if the variable does not matches any specified value
}
```

The `switch` statement is followed by a variable in parenthesis. The `case` keyword is followed by a `case` constant. The `case` constant can be an integer or a string constant. The data type of the `case` constant must match the data type of the `switch` variable.

Before executing the `switch` statement, a value should be assigned to the `switch` variable. The `switch` statement implements the code line by line. The `switch` statement executes the code only when the value of the `case` constant matches the value of the `switch` variable. The program continues to execute the statements until the end of the `switch` statement or until a `break` statement is encountered.

The `break` statement is used to move the control to the statements following the `switch` statement. It instructs a program to halt execution, come out of the switch statement, and execute statements following the `switch` construct. In the absence of the `break` statement, the program executes all the statements including the statements of the following cases in the `switch` construct.

The `default` is a special case. It is used when none of the `case` constants match the value of the `switch` variable. The `default` case is placed at the end of all the cases in a `switch` statement.

A `switch` statement evaluates a condition only once and the result is compared with each case statement to obtain a match. However, an `if...else` statement evaluates a condition more than once. For complex conditions, a `switch` statement is preferred to an `if...else` statement.

Session 10

Code Snippet 9 displays a `switch` statement without any `break` statement in a script named **`switch.php`**.

Code Snippet 9:

```
<?php
$day = 1;
switch ($day)
{
   case 1:
   echo "It is Sunday";
   echo "<br>";
   case 2:
   echo "It is Monday";
   echo "<br>";
   case 3:
   echo "It is Tuesday";
   echo "<br>";
   case 4:
   echo "It is Wednesday";
   echo "<br>";
   case 5:
   echo "It is Thursday";
   echo "<br>";
   case 6:
   echo "It is Friday";
   echo "<br>";
   case 7:
   echo "It is Saturday";
   echo "<br>";
   default:
   echo "There are Seven Days in a Week";
   echo "<br>";
}
?>
```

Concepts

Figure 10.9 displays the output of the script.

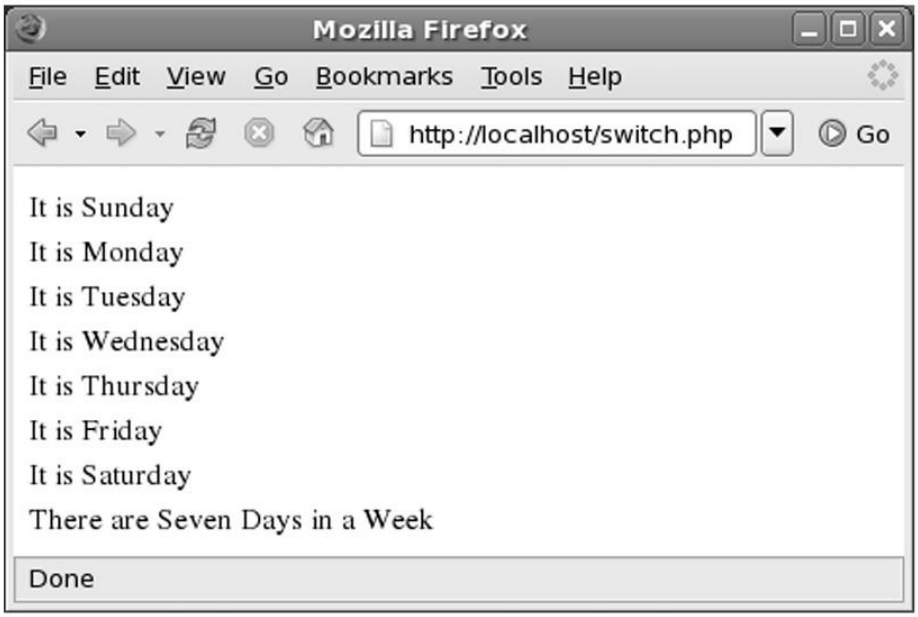


Figure 10.9: Using switch Statement

In the code, the weekday is 1, the program displays the message related to `case 1`. Due to the absence of a `break` statement, it also displays the messages related to the subsequent cases until it reaches the end of the `switch` statement.

Session 10

Conditional Statements in PHP

Code Snippet 10 is similar to Code Snippet 9, but contains a break statement in a script named **break.php**.

Code Snippet 10:

```
<?php
$day = 1;
switch ($day)
{
   case 1:
   echo "It is Sunday";
   echo "<br>";
   break;
   case 2:
   echo "It is Monday";
   echo "<br>";
   break;
   case 3:
   echo "It is Tuesday";
   echo "<br>";
   break;
   case 4:
   echo "It is Wednesday";
   echo "<br>";
   break;
   case 5:
   echo "It is Thursday";
   echo "<br>";
   break;
   case 6:
   echo "It is Friday";
   echo "<br>";
   break;
   case 7:
   echo "It is Saturday";
   echo "<br>";
   break;
   default:
   echo "There are Seven Days in a Week";
```

Concepts

```
    echo "<br>";
    break;
}
?>
```

Figure 10.10 displays the output of the script.

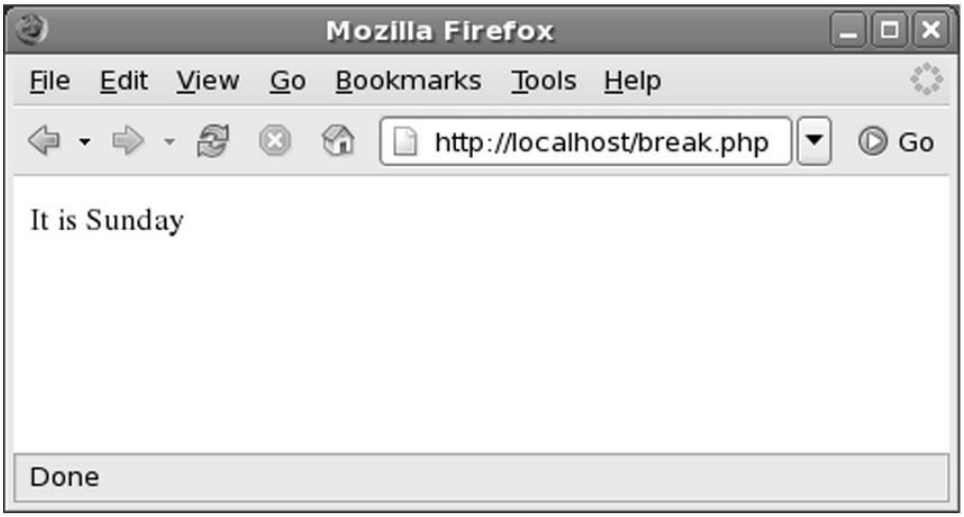


Figure 10.10: Using break Statement

In the code, the weekday is 1, the program displays only the message related to `case 1`. If the value assigned is any other value apart from numbers `1` to `7`, the program displays `There are Seven Days in a Week`.

10.4 Ternary (?) Operator

The ternary operator is also known as a conditional operator. It simplifies complex conditions into one-line statements.

The syntax for the ternary operator is as follows:

Syntax:

```
truth_exp ? expr1 : expr2;
```

A ternary operator requires three operands. The operator evaluates the `truth_expr` and if it is true, `expr1` is evaluated and if it is false, `expr2` is evaluated.

Session 10

The ternary operator is considered as an alternative for the `if…else` statement. This is demonstrated in the following code snippets to find the greatest of two numbers using the `if...else` statement as well as a ternary operator.

You can also display the output using the ternary operator as shown in Code Snippet 11 in the script named **`ternary-op.php`**.

Code Snippet 11:

```
<?php
$x = 100;
$y = 50;
$disp = ($x > $y) ? "X is greater than Y" : "Y is greater than X";
echo $disp;
?>
```

Figure 10.11 displays the output of the script.

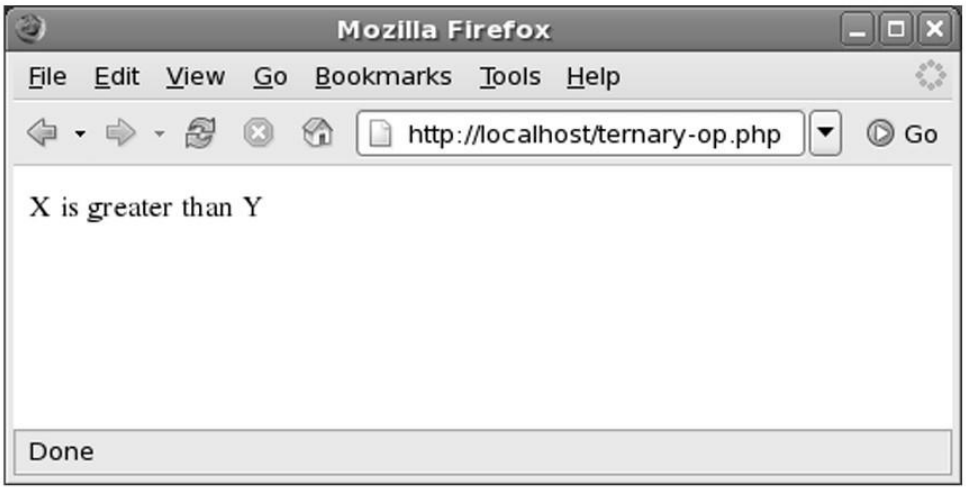


Figure 10.11: Using Ternary Operator to Compare Variables

The code displays, `X is greater than Y`.

Session 10

Concepts

Summary

- Conditional statements execute a set of statements only when a specified condition is satisfied and modify the order of flow in a program.
- The if statement executes a block of code only when the specified condition is true.
- In a nested if statement, you can include an if statement within another if statement or an else statement.
- A switch...case statement checks a single variable against multiple values and executes a block of code based on the value it matches.
- The break statement is used to transfer the control to the statements following the switch...case statement.
- The default statement is used when none of the case statements matches the value of the switch variable.
- Ternary operator is also known as a conditional operator. It simplifies complex conditions into one-line statements.

Check Your Progress

1. The if keyword is followed by the _____ in parentheses.

   a. condition

   b. operator

   c. variable

   d. declaration

2. Which of the following statements executes a set of code only when a condition is true?

   a. if statement

   b. break statement

   c. Assignment statement

   d. Ternary operator

3. Which of the following conditional statement is used when there are many conditions with discrete values?

   a. if statements

   b. switch statements

   c. Conditional statements

   d. Ternary operators

4. Which of the following statements is used when none of the case statements matches the value of the switch variable?

   a. Ternary operator

   b. switch statement

   c. break statement

   d. default statement

Concepts

Check Your Progress

5. A program continues to execute the statements in a switch statement until the end of the switch block or until it encounters the _____statement.

   a. if

   b. Default

   c. Continue

   d. break

6. What helps in simplification of complex conditional statements into one-line statements?

   a. if statement

   b. switch statement

   c. Conditional statement

   d. Ternary operator

Check Your Progress

7. What will be the output of the following code?

```
<?php
$num = 2750;
if($num % 2 == 0)
{
echo "num is even.";
}
else
{
echo "num is odd.";
}
?>
```

a. 2750 is even.

b. num is even.

c. num is odd.

d. 2750 is odd.

Concepts

Check Your Progress

8. What will be the output of the following code?

```
<?php
 $year = 2001;
 $month = 4;
 $days;
 switch ($month)
    {
      case 2:  $days = ($year % 4 == 0) ? 28 : 29;
               break;
      case 4:
      case 6:
      case 9:
      case 11:$days = 30;
break;
      default:$days = 31;
    }
 echo $days
?>
```

a. No display

b. 30

c. 31

d. Syntax error

Answers

1. a
2. a
3. b
4. d
5. d
6. d
7. b
8. b

Session 11 Conditional Statements in PHP (Lab)

Objectives

At the end of this session, the student will be able to:

- *Use the if statement.*
- *Use the switch statement.*
- *Use the Ternary (?) operator.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Using the if Statement

The `if` statement executes a block of code only when the specified condition is true. If the condition is false, the program ignores the block of code within the `if` body and executes the statements after the `if` body.

You will create a form that accepts name and age from the user. You will also perform validations to the data present in the form and check if the fields in the form are blank using `if…else` statement.

To create the form to accept name and age from the user, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form:**

```
<HTML>
<HEAD>
<TITLE>Personal Details</TITLE>
</HEAD>
<BODY>
<H4>Please enter the information</H4>
<FORM ACTION = "dispDet.php" METHOD = "POST">
<table>
<tr>
```

Session 11

Conditional Statements in PHP (Lab)

Lab Guide

```
<td>Name: </td>
<td><input type="text" name="myname"> </td>
</tr>
<tr>
<td>Age: </td>
<td><input type="text" name= "myage"> </td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
</FORM>
</BODY>
</HTML>
```

3. **Save the file as info.html under the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to retrieve the information from the form and store them in variables:**

```
<HTML>
<HEAD>
<TITLE> User Information </TITLE>
</HEAD>
<BODY>
<?php
$myname = $_POST['myname'];
$myage = $_POST['myage'];
if($myname==="")
{
    echo "Please enter your name";
}
else
{
  if ($myage==="")
  {
        echo $myname;
        echo ",you did not enter your age!";
  }
```

Session 11


```
        else
          {
              echo "Hi ";
              echo $myname;
              echo ". ";
              echo "Your age is ";
              echo $myage;
              echo ".";
          }
      }
      ?>
      <br>
      <a href="info.html"> Back </a>
      </BODY>
      </HTML>
```

Figure 11.1 displays the text editor containing the PHP Script to display user information.

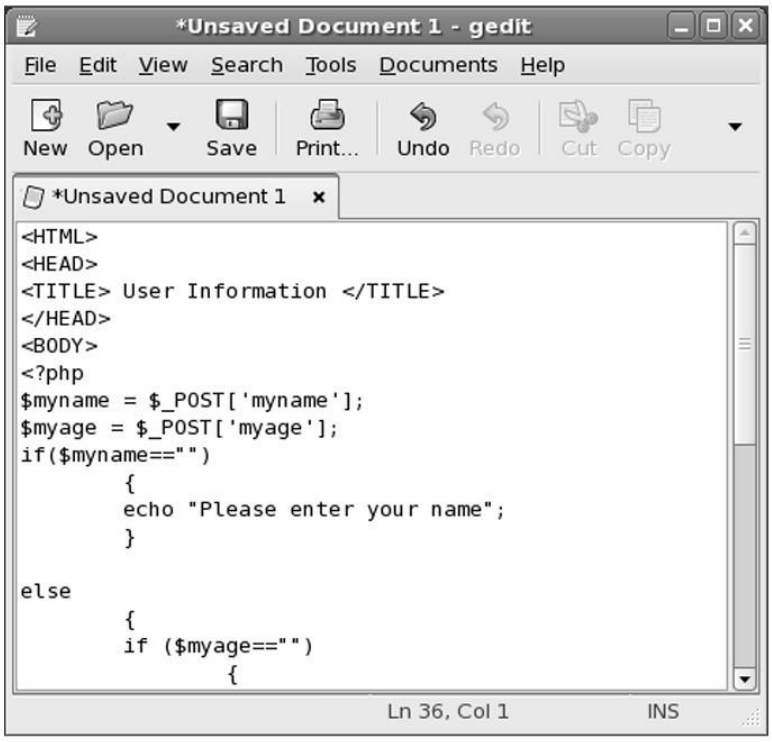


Figure 11.1: PHP Script to Display User Information

Session 11

6. **Save the file as dispDet.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/info.html in the Address bar and press Enter.**

   Figure 11.2 displays the output of the script.

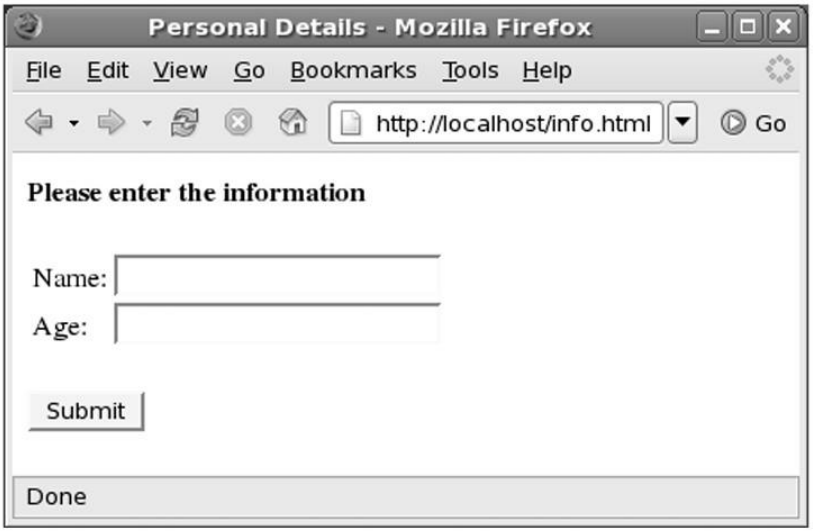


Figure 11.2: Personal Details Form

9. **Enter John in the Name box.**

10. **Enter 44 in the Age box.**

11. **Click Submit.**

Lab Guide

Figure 11.3 displays the output of the script.

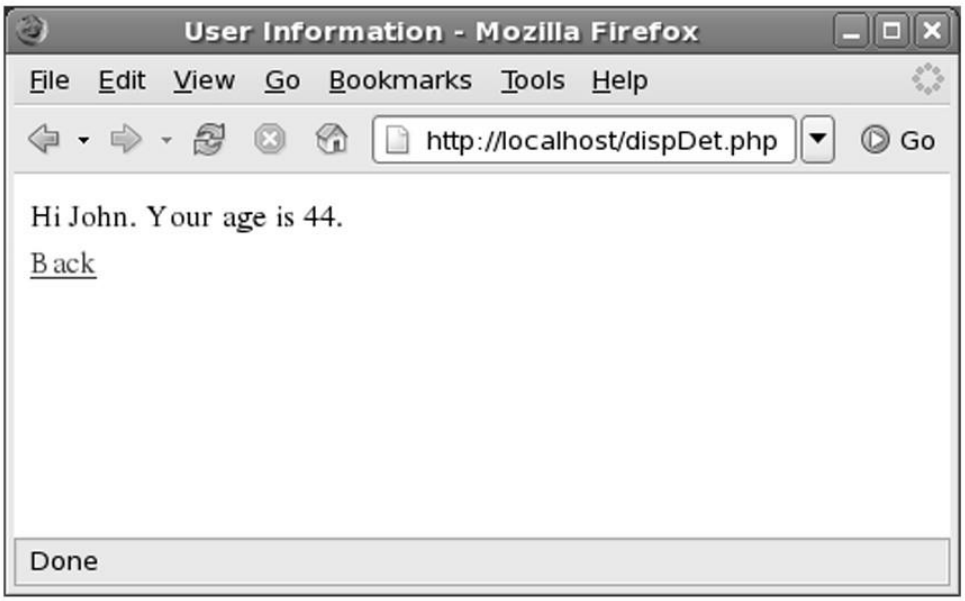


Figure 11.3: Displaying Personal Details

Using the switch Statement

A `switch` statement checks the value present in a variable against multiple values and executes a block of code based on the value it matches. The value can be an integer or string constant. The `switch` statement is easier to read than multiple if statements.

Consider an example, where an organization provides various facilities and perks to their employees depending on the grade assigned to them. Table 11.1 lists all the facilities available for the respective grades.

Grade	Facilities
A	1. Increments are based on 30% of their basic salary 2. Bonus at the rate of 20% of their basic salary 3. Traveling allowance worth $500 4. Medical allowance worth $500
B	1. Increments are based on 20% of their basic salary 2. Bonus at the rate of 10% of their basic salary 3. Traveling allowance worth $300 4. Medical allowance worth $300

Session 11

Conditional Statements in PHP (Lab)

Grade	Facilities
C	1. Increments are based on 10% of their basic salary 2. Bonus at the rate of 5% of their basic salary 3. Traveling allowance worth $100 4. Medical allowance worth $100

Table 11.1: Facilities Based on Grades

To display facilities according to the grades using the `switch` statement, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following HTML code to create a form that accepts the name and grade of an employee:**

```
<HTML>
<HEAD>
   <TITLE>Grade Details</TITLE>
</HEAD>
<BODY>
     <H4>Please enter the information</H4>
     <FORM ACTION = "calcPerks.php" METHOD = "GET">
          <TABLE>
          <TR>
          <TD>Name: </TD>
          <TD><INPUT TYPE="text" NAME="myname"> </TD>
          </TR>
          <TR>
          <TD>Enter your Grade:</TD>
          <TD><INPUT TYPE="text" NAME="mygrade"> </TD>
          </TR>
          </TABLE>
     <BR>
     <INPUT TYPE="submit" VALUE="Submit">
     </FORM>
</BODY>
</HTML>
```

3. **Save the file as perks.html under the /usr/local/apache2/htdocs directory.**

Session 11

Conditional Statements in PHP (Lab)

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to retrieve values entered by the user in the PHP variables:**

```
<HTML>
<HEAD>
<TITLE> Grade Details </TITLE>
</HEAD>
 <BODY>
 <?php
 $myname=$_GET['myname'];
 $mygrade=$_GET['mygrade'];
 echo "<br>";

 if($myname==="")
 {
     echo "Please enter your name";
 }
 else
 {
     switch($mygrade)
     {
         case "":
                 echo $myname;
                 echo ", you did not enter your Grade!";
                 break;
         case "A":
                 echo "Facilities for $myname (Grade A)";
                 echo "<BR><BR>";
                 echo "Increment = 30% of basic salary";
                 echo "<BR>";
                 echo "Bonus = 20% of basic salary";
                 echo "<BR>";
                 echo "Traveling allowance = $500";
                 echo "<BR>";
                 echo "Medical allowance = $500";
                 echo "<BR>";
                 break;
         case "B":
```

Session 11

Conditional Statements in PHP (Lab)


```
                    echo "Facilities for $myname (Grade B)";
                    echo "<BR><BR>";
                    echo "Increment = 20% of basic salary";
                    echo "<BR>";
                    echo "Bonus = 10% of basic salary";
                    echo "<BR>";
                    echo "Traveling allowance = $300";
                    echo "<BR>";
                    echo "Medical allowance = $300";
                    echo "<BR>";
                    break;
        case "C":
                    echo "Facilities for $myname (Grade C)";
                    echo "<BR><BR>";
                    echo "Increment = 10% of basic salary";
                    echo "<BR>";
                    echo "Bonus = 5% of basic salary";
                    echo "<BR>";
                    echo "Traveling allowance = $100";
                    echo "<BR>";
                    echo "Medical allowance = $100";
                    echo "<BR>";
                    break;
        default:
                    echo "$myname, Please enter the correct Grade (A, B,
                           or C)";
                    break;
        }
    }
?>
<A HREF="perks.html"> Back </A>
</BODY>
</HTML>
```

6. **Save the file as calcPerks.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/perks.html in the Address bar and press Enter.**

Lab Guide

Figure 11.4 displays the output of the script.

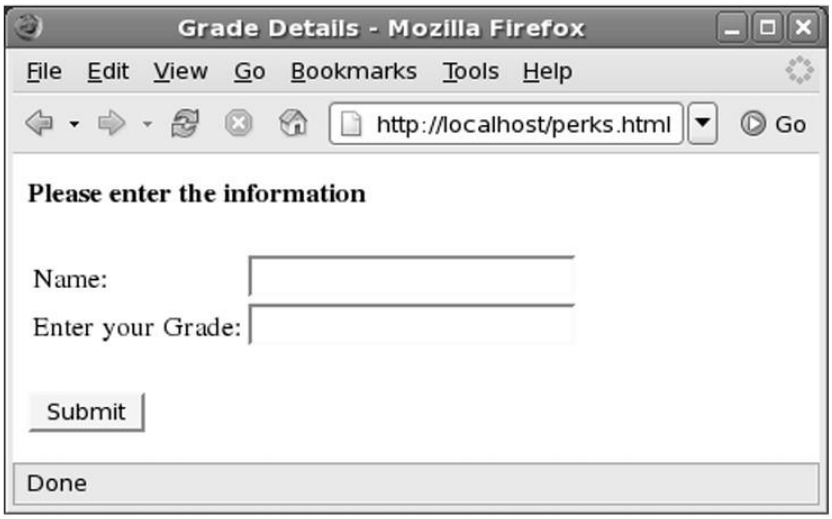


Figure 11.4: Grade Details Form

9.	**Enter John in the Name box.**
10.	**Enter A in the Grade box.**
11.	**Click Submit.**

Lab Guide

Figure 11.5 displays facilities available for Grade A.

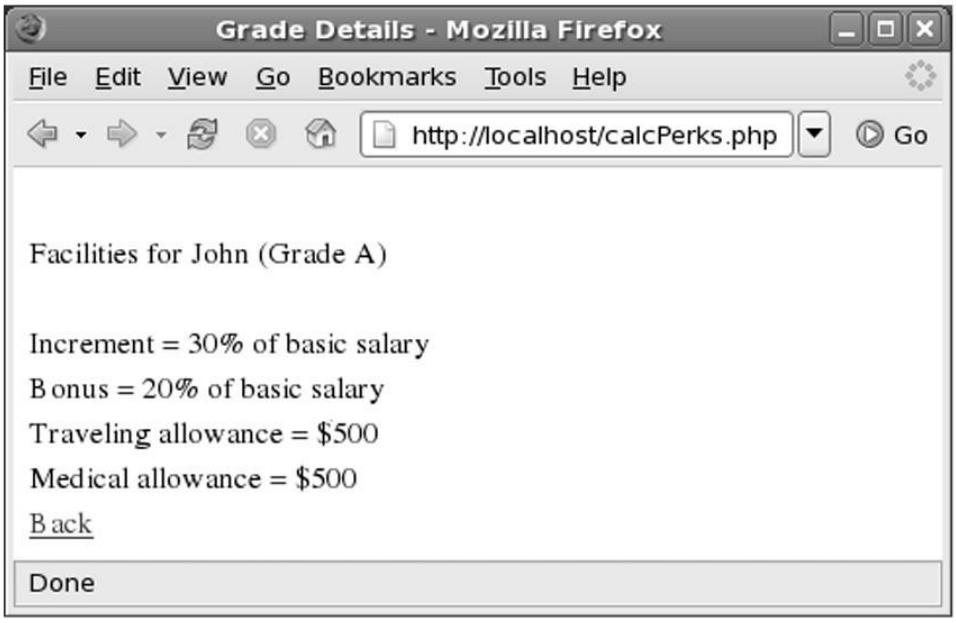


Figure 11.5: Displaying Facilities for Grade A

Using the Ternary Operator

Ternary operator is also known as conditional operator. It simplifies complex conditions into single line statements. Ternary operator evaluates the expression for a True or False value and executes one of the two given statements. It is considered as a shortcut for `if...else` statement.

You can modify the **`dispDet.php`** file to display the same message using the ternary operator.

To use the ternary operator, perform the following steps:

1. **Open the dispDet.php file in the text editor.**

2. **Replace the PHP script in the code with the following script:**

```
<HTML>
<HEAD/>
<BODY>
<TITLE> User Information </TITLE>
```

Session 11

Conditional Statements in PHP (Lab)


```
<?php
$myname=$_POST['myname'];
$myage=$_POST['myage'];
if ($myname==="")
 {
     echo "Please enter your name!";
 }
else
{
     $mesg = ($myage==="") ? "$myname, please enter your age!" :
     "Hi $myname. Your age is $myage.";
     echo $mesg;
}
?>
<br>
<a href="info.html"> Back </a>
</BODY>
</HTML>
```

3. **Save the dispDet.php file.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/info.html in the Address bar and press Enter. The HTML page is displayed.**

6. **Enter John in the Name box.**

7. **Enter 44 in the Age box.**

8. **Click Submit.**

Lab Guide

Figure 11.6 displays the output of the script.

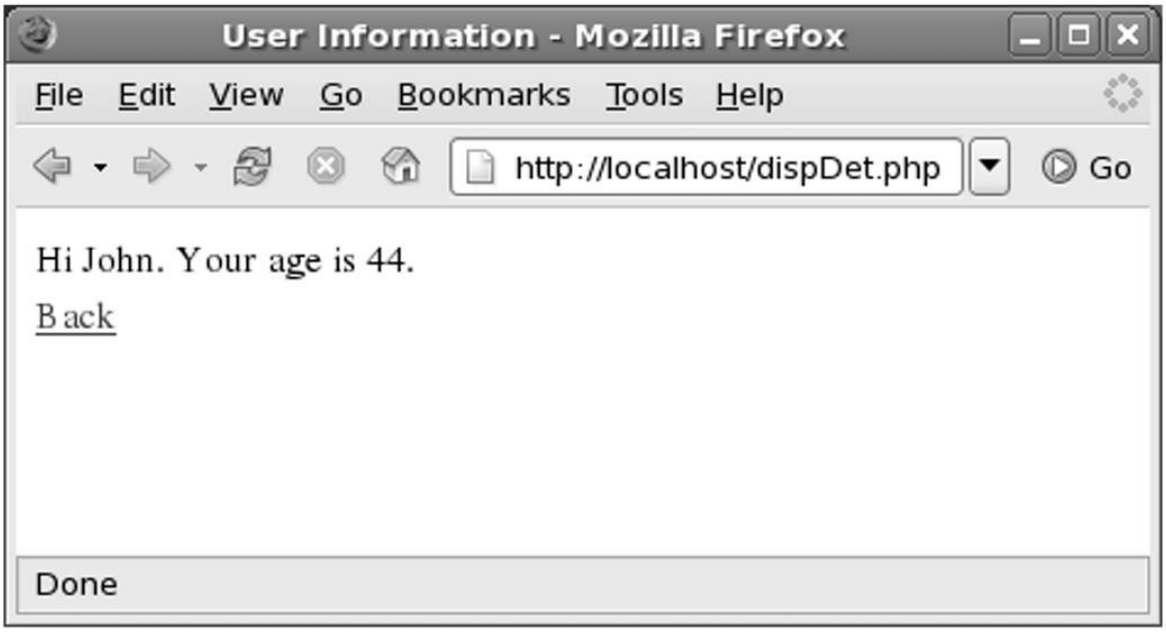


Figure 11.6: Using the Ternary Operator

Do It Yourself

1. Create a form for an online shopping mall that accepts details such as the Customer Name, Membership Number, Type of Membership, and the Points in Hand. There are two types of membership as shown in table 11.2 - Basic and Privileged.

Points in Hand	Additional Points	
	Basic Member	Privileged Member
0-250	5%	10%
251-500	7.5%	12.5%
501-750	10%	15%
751-1000	12.5%	17.5%

Table 11.2: Types of Membership

Calculate and display the additional amount of points available and the total amount of points. Use the `if…else` statement.

2. Create a job application form for an organization that accepts the first name, last name, address, phone number, qualifications, and experience of the candidate. It must also display different departments such as IT, Finance, and Sales as shown in table 11.3 for which the candidate wants to apply. Depending on the choice of department, different posts that are available under that department must be displayed. Use the `switch` statement.

Department	Post Available
IT	Software Developer, Graphic Designer, Web Designer, and Technical Support
Finance	Cost Analyst, Chartered Accountant, Company Secretary, Accountant, and Auditor
Sales	Sales Executive, Sales Manager, and Sales Analyst

Table 11.3: Department

Onlinevarsity

ANYTIME | ANYWHERE

Session 12

Flow Control in PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the use of loops.*
- *Explain the use of jump statements.*

12.1 Introduction

Loops are useful when you need to perform repetitive tasks such as retrieving information stored in databases, sending mails to multiple users, or reading contents of an array. The loop statements control the flow of the statement execution. The various loop statements provided by PHP are `while` loop, `do-while` loop, and `for` loop. The jump statements also control the execution of the loop. The different types of jump statements are `break`, `continue`, and `exit`.

In this session, you will learn to explain and use the various types of loop and jump statements in PHP.

12.2 Working with Loops

A loop executes a block of code repetitively. In this control structure, the specified condition is tested, and the statements present in the body of the loop are executed repetitively if the condition is true. If the condition is false, the loop ends, and the control is transferred to the statement following the loop. The continuous execution of statements inside the loop is called iteration.

PHP provides the following loop statements:

- `while`
- `do-while`
- `for`

12.2.1 The while Loop

A `while` loop executes the statements in the loop body as long as the condition within the parentheses is true. In `while` loop, the validity of the condition is checked before the loop is executed. So, if the condition is false, the body of the loop is not executed.

Session 12

Flow Control in PHP

Concepts

The syntax for a `while` loop is as follows:

Syntax:

```
while(condition)
{
These statements are executed only if the condition is true;
}
These statements are executed irrespective of the condition;
```

The `while` keyword is followed by the `condition` in parentheses. The `condition` is the test expression that consists of variables and operators. If the `condition` is satisfied, the statements present in the loop body are executed. If the `condition` is false, the statements present in the loop body is skipped, and the control moves to the statements following the loop.

The following examples illustrate the use of the `while` loop:

- Consider the code as shown in Code Snippet 1 to display the first five multiples of `5`, in a script named **`while1.php`**.

  Code Snippet 1:

```
<?php
$counter=1;
$number=5;
while($counter <= 5)
{
   $result=$number*$counter;
   echo "<br>$result";
   $counter=$counter+1;
}
?>
```

Session 12

Figure 12.1 displays the output of the script.

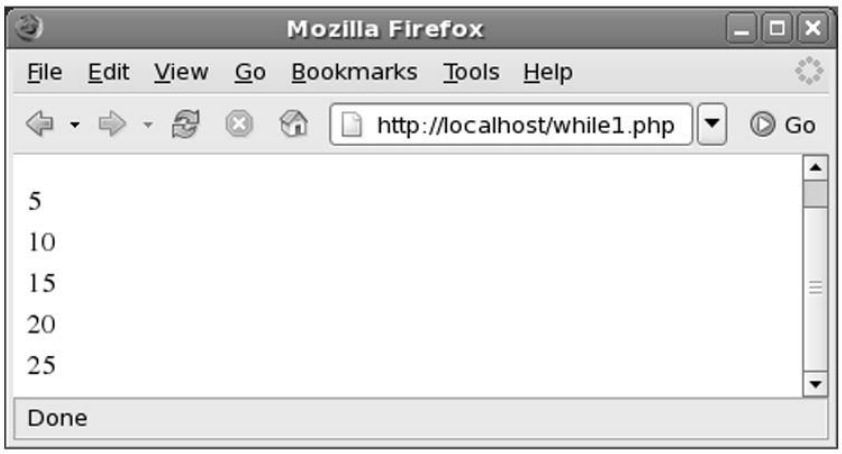


Figure 12.1: Using the while Loop

In the code, the result is displayed until the counter reaches `5`. The loop stops once the counter exceeds `5`.

- Consider the code as shown in Code Snippet 2 to display the odd numbers between `1` and `10`, in a script named **`while _ odd.php`**.

Code Snippet 2:

```
<?php
$number=1;
echo "The odd numbers between 1 and 10 are:";
while($number <= 10)
{
   echo "<br>$number";
   $number=$number+2;
}
?>
```

Concepts

Figure 12.2 displays the output of the script.

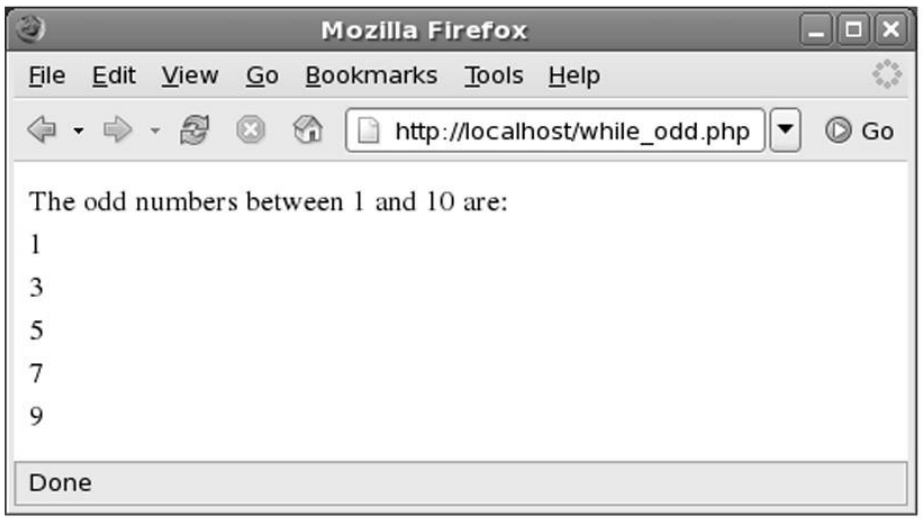


Figure 12.2: Displaying Odd Numbers Using while Loop

In the code, the number is always incremented by 2 until the number reaches 10. This is because `$number` is initialized at 1 and every alternate number is odd.

12.2.2 The do-while Loop

In a `while` loop, the condition is evaluated at the beginning of the loop. If the condition is false, the statements in the loop body are not executed at all. The main difference between `do-while` and `while` loops is that in `do-while` loops, the condition is checked at the end of the loop instead of at the beginning.

To execute the loop body at least once, use the `do-while` loop. The `do-while` loop works similar to the `while` loop. The only difference between the `while` and `do-while` loop, is that in the `do-while` loop the condition is placed at the end of the loop and it is executed atleast once.

The syntax for a `do-while` loop is as follows:

Syntax:

```
do{
       <These statements are executed if the condition is true;>
}while(condition)
<These statements are executed irrespective of the condition;>
```

Session 12

In a `do-while` loop, the loop body follows the `do` keyword. The loop body is followed by the `while` keyword and the condition in the parentheses. If the condition evaluates to true, the loop executes. If the condition evaluates to false, the loop terminates and the statements following the `while` keyword are executed.

The following examples illustrate the use of `do-while` loop:

- Consider the code as shown in Code Snippet 3 to display the odd numbers between 1 and 10, in a script named **`do-while _ odd.php`**.

  Code Snippet 3:

```
<?php
$number=1;
echo "The odd numbers between 1 and 10 are:";
do{
    echo "<br>$number";
    $number=$number+2;
}
while($number <= 10);
echo "<br>The loop ends because the condition is satisfied.";
?>
```

  Figure 12.3 displays the output of the script.

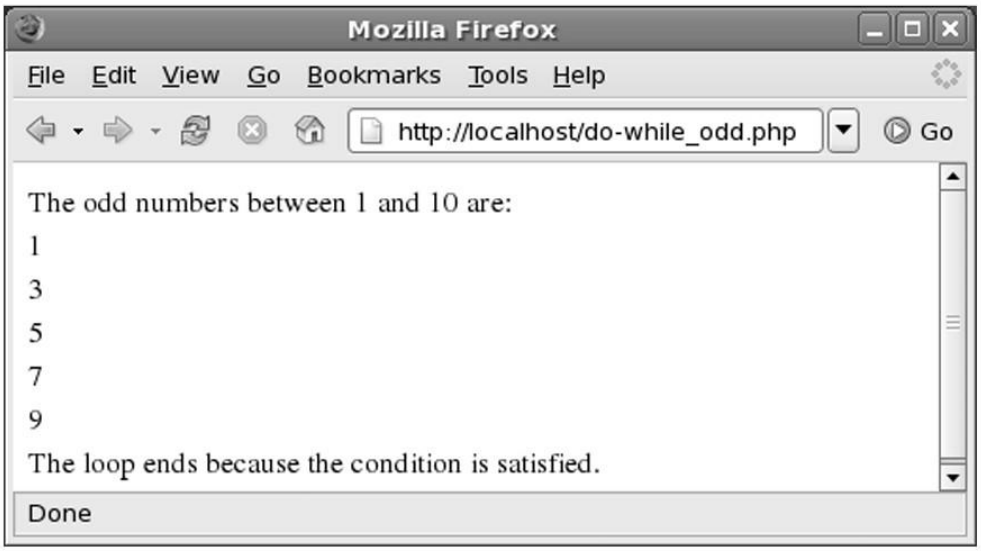


Figure 12.3: Displaying Odd Numbers Using do-while Loop

Session 12

Concepts

In the code, odd numbers between 1 and 10 are displayed using the `do-while` loop. Before the iteration of the loop starts, the value is displayed once. Then the execution of the loop starts. In the code, `$number` is initialized at 1 and is incremented by 2, since the odd numbers are required to be displayed.

The execution of the loop continues until the counter reaches 10. The loop stops execution once the condition is satisfied.

- Consider the code as shown in Code Snippet 4 to display even numbers from 1 to 10, in a script named **`do-while _ even.php`**.

Code Snippet 4:

```
<?php
$counter=0;
echo "The even numbers are: <br>";
do {
echo "$counter<br>";
$counter=$counter+2;
}
while($counter <= 10);
echo "The loop ends because the condition is satisfied.";
?>
```

Figure 12.4 displays the output of the script.

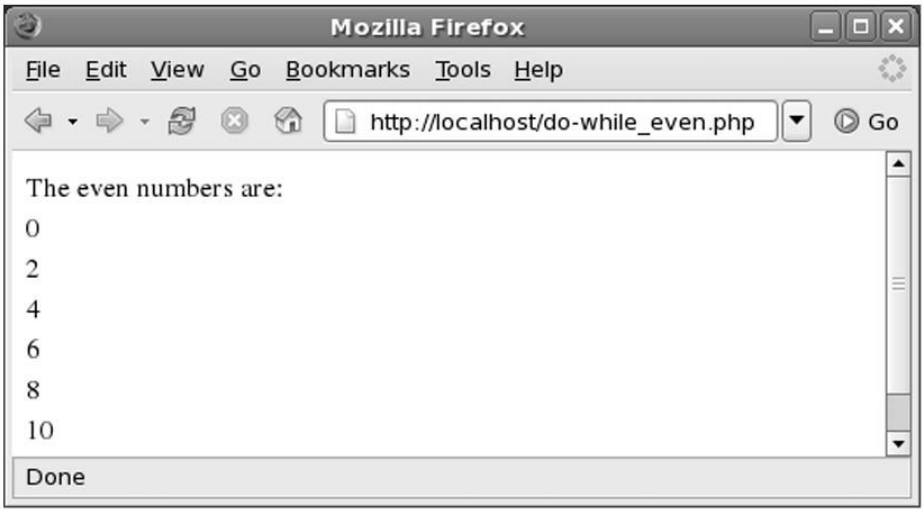


Figure 12.4: Displaying Even Numbers Using the do-while Loop

In the code, the even numbers are displayed using the `do-while` loop. Before the iteration of the loop starts, the value is displayed once. Then the execution of the loop starts. In the code, `$number` is initialized at 0 and is incremented by 2, since the even numbers are required to be displayed.

The execution of the loop continues until the counter reaches 10. The loop stops execution once the condition is satisfied.

12.2.3 The for Loop

The `for` loop enables to execute a block of code repetitively for a fixed number of times. The statements in the body of the loop are executed as long as the condition is satisfied and stops only when the condition is not satisfied.

The syntax for the `for` loop is as follows:

Syntax:

```
for (expr1; expr2; expr3)
{
    These statements are executed if the condition is true;
}
These statements are executed irrespective of the condition;
```

The `for` loop consists of the initialization expression, text expression, and the re-initialization expression. The initialization expression initializes the value of the counter. The condition expression specifies the test expression that is evaluated for each loop iteration. The re-initialization expression increases or decreases the value in the counter variable. The code inside the `for` loop is executed only when the condition returns true. The re-initialization expression helps in the iteration of the loop.

When the values in a `for` loop are not initialized, it goes into endless iteration mode. Such loops are known as infinite loops.

Session 12

The following examples illustrate the use of the `for` loop:

- Consider the code in Code Snippet 5 to display the double of the number given, in a script named **`for.php`**.

Code Snippet 5:

```
<?php
$number=6;
for($counter=1; $counter <= 3; $counter++)
{
   echo "$number<br>";
   $number=$number*2;
}
echo "The loop ends because the condition is satisfied.";
?>
```

Figure 12.5 displays the output of the script.

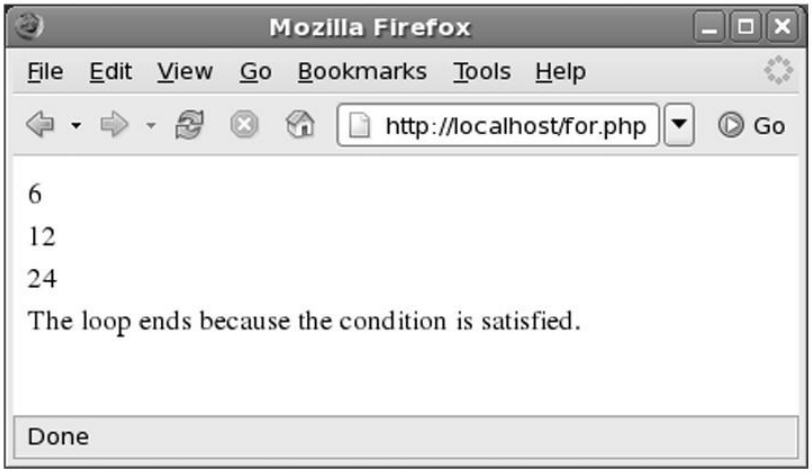


Figure 12.5: Multiplying Numbers Using for Loop

In the code, the `for` loop displays the double of the given number.

The variable, `$number` is initialized with a value of 6. When the loop starts, 6 is multiplied by 2 and the value stored in the variable, `$number` is 12. The loop executes thrice since the terminating condition has been set when the counter value reaches 3. Once the counter value reaches 3, the loop stops executing.

Session 12

- Consider the code in Code Snippet 6 to display the first five odd numbers in the reverse order, in a script named **for _ rev.php**.

Code Snippet 6:

```
<?php
echo "The odd numbers in reverse order are:";
for($i=5;$i>=1;$i--)
{
   $number=$i * 2 - 1;
   echo "<br>$number";
}
echo "<br>The loop ends because the condition is satisfied.";
?>
```

Figure 12.6 displays the output of the script.

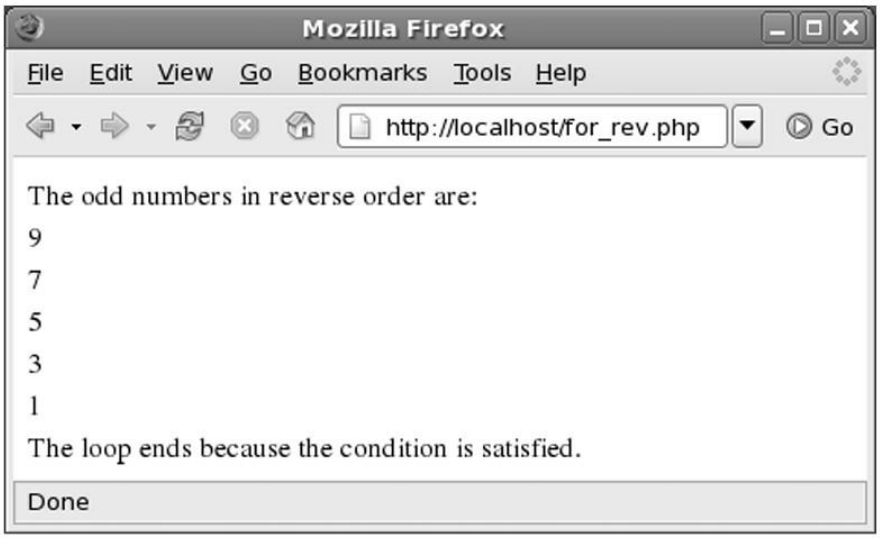


Figure 12.6: Displaying Odd Numbers in Reverse Order Using for Loop

In the code, the `for` loop declares a counter variable, which is initialized at 5. The re-initialization expression decrements the counter every time the `for` loop is executed.

Concepts

12.3 Jump Statements

The jump statements control the execution of the loop and conditional statements. PHP provides the following jump statements:

- `break`
- `continue`
- `exit`

12.3.1 The break Statement

The `break` statement stops the execution of the loops and conditional statements. The control is then transferred either to the beginning of the next loop or to the statement following the loop. This statement can be used with the `if` statement, `switch` statement, `for` loop, `while` loop, and `do-while` loop.

The following examples illustrate the use of `break` statements:

- Consider the code in Code Snippet 7 to display the consecutive numbers from 1 to 5, in a script named **`break _ consec.php`**.

  Code Snippet 7:

```
<?php
for($i=1;;$i++) {
if($i>5)
{
   break;
}
echo "<br>$i";
}
?>
```

Figure 12.7 displays the output of the script.

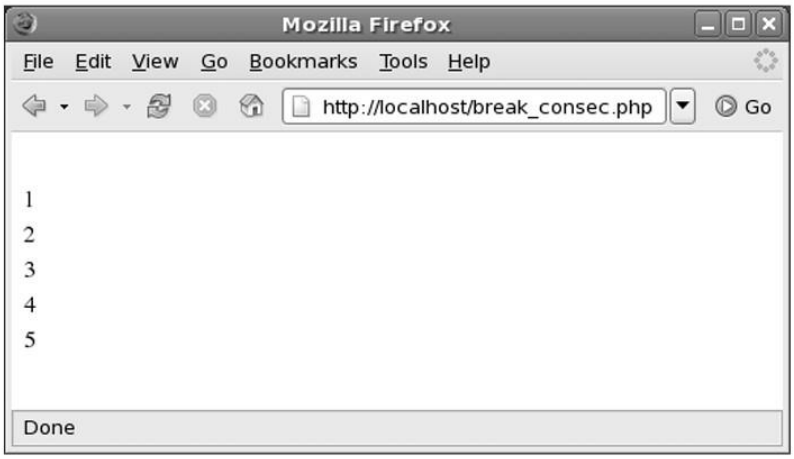


Figure 12.7: Displaying Consecutive Numbers Using the break Statement

In the code, the `break` statement is used within the `for` loop. The `for` loop does not include any terminating condition. The terminating condition is specified within the `if` statement using the `break` statement. If the `break` statement is not used, it will become an infinite loop.

- Consider the code as shown in Code Snippet 8 to check whether the alphabet is a vowel using `switch` statement, in a script named **`break _ vowel.php`**.

Code Snippet 8:

```
<?php
$alphabet='u';
switch($alphabet) {
case 'a':
echo "<br>The alphabet is a vowel.";
break;
case 'A':
        echo "<br>The alphabet is a vowel.";
        break;
case 'e':
        echo "<br>The alphabet is a vowel.";
        break;
case 'E':
        echo "<br>The alphabet is a vowel.";
        break;
```

Concepts

```
case 'i':
        echo "<br>The alphabet is a vowel.";
        break;
case 'I':
        echo "<br>The alphabet is a vowel.";
        break;
case 'o':
        echo "<br>The alphabet is a vowel.";
        break;
case 'O':
        echo "<br>The alphabet is a vowel.";
        break;
case 'u':
        echo "<br>The alphabet is a vowel.";
        break;

case 'U':
        echo "<br>The alphabet is a vowel.";
        break;

default:
        echo "<br>The alphabet is not a vowel.";
}
?>
```

Figure 12.8 displays the output of the script.

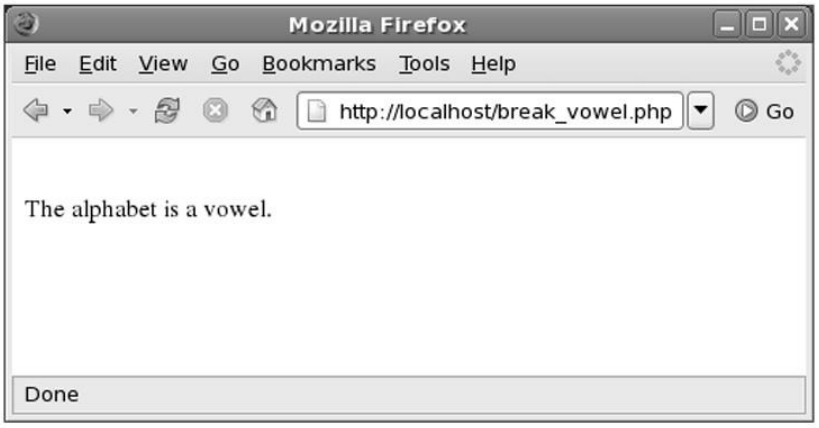


Figure 12.8: Checking Alphabets Using the break Statement

In the code, the `break` statement is used in the `switch` statement. The `break` statement moves the control to the statements following the `switch` statement. If the `break` statement is not used, PHP will execute all the statements including the statements present in the following `case` statement.

12.3.2 The continue Statement

The `continue` statement is used within the loop statements. It skips the code following the `continue` statement in the loop body and executes the next iteration of the loop. This statement is used in nested loops. The `continue` statement can be used with the `if` statement, `for` loop, `while` loop, and `do-while` loop.

- Consider the code as shown in Code Snippet 9 to display the consecutive numbers from 1 to 5 using the `while` loop in a script named **`while _ continue.php`**.

Code Snippet 9:

```
<?php
$counter = 0;
while($counter<5)
{
   $counter++;
   if($counter==3)
   {
     echo "Continues the loop<br>";
     continue;
   }
   echo "$counter<br>";
}
echo "The loop ends here";
?>
```

Concepts

Figure 12.9 displays the output of the script.

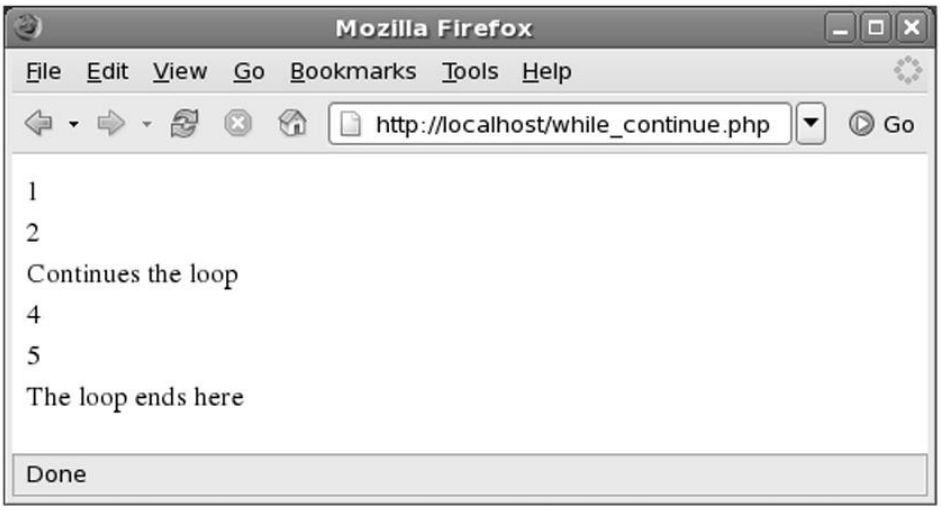


Figure 12.9: Displaying Consecutive Numbers Using the continue Statement

In the code, the `continue` statement is used in the `if` statement. Here, the counter is initialized to 0. The loop continues until the counter reaches 3. When the counter reaches the value of 3, the loop skips the `if` body and executes the next iteration of the loop. The loop continues until the condition becomes false.

12.3.3 The exit Statement

The `exit` statement ends the loop and the control is transferred to the statement following the loop body.

Consider the code as shown in Code Snippet 10, to calculate the HRA, in a script named **`hra.php`**.

Code Snippet 10:

```
<?php
$salary=8000;
if($salary<6000)
{
   echo "Basic : $salary<br>";
   echo "Salary below 6000 is not entitled for HRA.";
   exit;
}
```

```
else
{
   echo "Basic : $salary<br>";
   $hra=$salary * 0.8;
   echo "HRA : $hra";
}
?>
```

Figure 12.10 displays the output of the script.

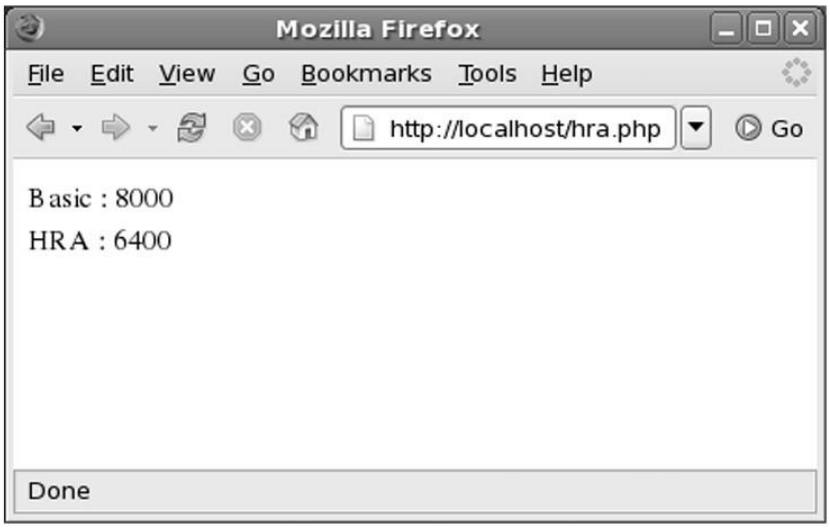


Figure 12.10: Calculating HRA Using the exit Statement

In the code, HRA is calculated based on the basic salary. If the basic salary is less than 6000, the `if` statement exits. If the basic salary is greater than or equal to 6000, HRA is calculated.

Session 12

Summary

- A loop executes a block of code repetitively.
- A while loop executes the statements in the loop body as long as the condition is true.
- The do-while loop is similar to the while loop. In this loop structure the condition is placed at the end of the loop.
- A for loop enables the execution of a block of code repetitively for a fixed number of times.
- The jump statements control the execution of the loop statements.
- The break statement stops the execution of the loop. The control is then passed either to the beginning of the next loop or to the statement following the loop.
- The continue statement skips the code following the continue statement in the loop body and executes the next iteration of the loop.
- The exit statement ends the loop and the control is passed to the statement following the loop body.

Check Your Progress

1. The _____statement executes a set of code only when a condition is true.

   a. if

   b. switch

   c. conditional

   d. while

2. In the _____ loop, the condition is placed at the end of the loop.

   a. while

   b. do-while

   c. for

   d. for-while

3. A _____ loop enables to execute a block of code repetitively for a fixed number of times.

   a. for

   b. while

   c. do-while

   d. for-each

4. The _____ statement stops the execution of the loop.

   a. continue

   b. exitloop

   c. break

   d. else

Concepts

Check Your Progress

5. The _____ loop consists of the initialization expression, text expression, and the re-initialization expression.

   a. do-while

   b. while

   c. if

   d. for

6. What will be output of the following code script?

```
<?php
for ($i=0; $i<5;)
  {
   do {
echo $i;
break;
      }
   while ($i > 0);
   $i++;
   }
?>
```

   a. 0

   b. 012345

   c. 01234

   d. Syntax error

Check Your Progress

7. What changes should be incorporated in the code script to display numbers from 15 to 50?

```
<?php
for($num=15;;$num++) {
if($num>50)
{
_______________
}
echo "<br>$num";
}
?>
```

a. break;

b. continue;

c. stop;

d. echo "stop at 50";

Session 12

Flow Control in PHP

Concepts

Answers

1. d
2. b
3. a
4. c
5. d
6. c
7. a

Onlinevarsity

Session 13

Flow Control in PHP (Lab)

Objectives

At the end of this session, the student will be able to:

- *Use the while loop.*
- *Use the do-while loop.*
- *Use the for loop.*
- *Use the jump statement.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Using the while Loop

A `while` loop executes the statements in the loop body as long as the condition evaluates to true.

To generate numbers in the Fibonacci series with the help of the `while` loop, perform the following steps:

1. **Open a new file in the gedit text editor.**
2. **Enter the following code for displaying the Fibonacci series:**

```
<?php
$fib1=0;
$fib2=1;
$sum=0;
echo "The Fibonacci series : <BR><BR>";
echo "$fib1<BR>";
while($sum<=15)
{
    echo "$fib2<BR>";
    $sum = $fib1+$fib2;
```

Session 13


```
    $fib1 = $fib2;
        $fib2 = $sum;
    }

    echo "<BR>";
    return 0;
    ?>
```

3. **Save the file as fibo.php in the /usr/local/apache2/htdocs/ directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/fibo.php in the Address bar and press Enter.**

Figure 13.1 displays the output of the script.

Figure 13.1: Viewing the Fibonacci Series

Session 13

Flow Control in PHP (Lab)

Using the do-while Loop

The `do-while` loop works similar to the `while` loop. The difference between a `while` and a `do-while` loop is that in the `do-while` loop the condition is placed at the end of the loop and the loop is executed at least once.

To display odd numbers in the reverse order using the `do-while` loop, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to display the first eight odd numbers in the reverse order:**

```
<?php
echo "Odd numbers in reverse order:<BR>";
$i=8;
do
{
    $num=$i*2-1;
    echo "<BR>$num";
    $i--;
}while($i>=1);
?>
```

3. **Save the file as odd.php under the /usr/local/apache2/htdocs/ directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/odd.php in the Address bar and press Enter.**

Figure 13.2 displays the output of the script.

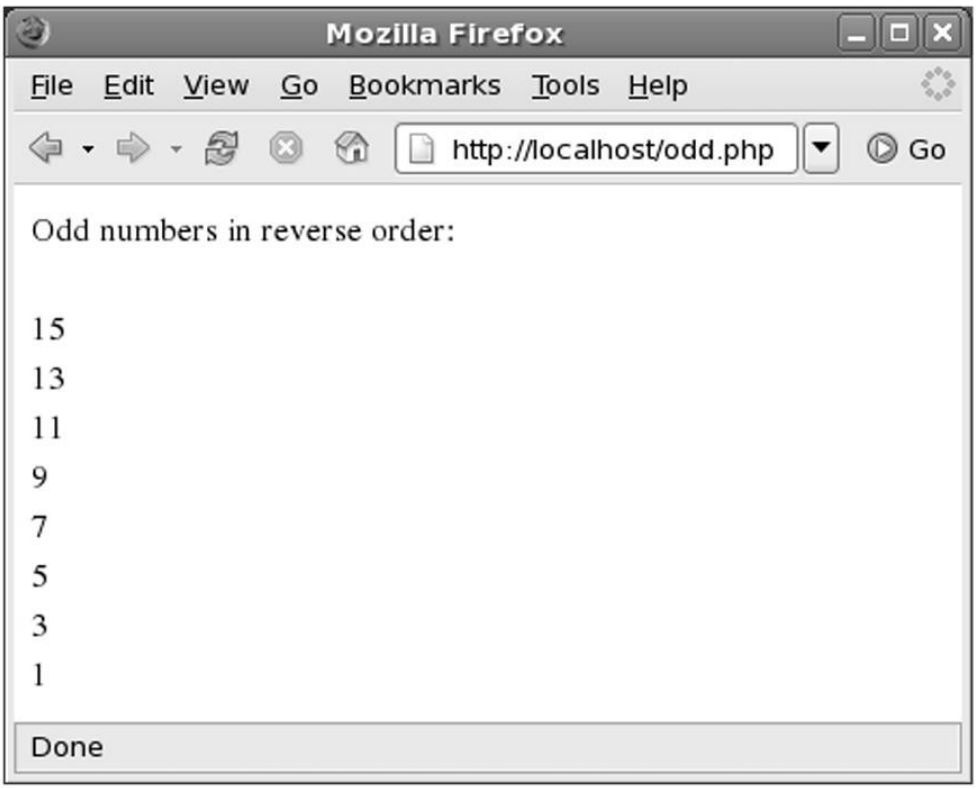


Figure 13.2: Viewing Odd Numbers in Reverse Order

Using the for Loop

A `for` loop enables to execute a block of code repetitively for a fixed number of times. The `for` loop consists of the initialization expression, test expression, and the re-initialization expression.

To display the square of the first ten numbers using the `for` loop, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
echo "The square of first ten numbers are :<BR><BR>";
```

```
for($i=1;$i<=10;$i++)
{
    $square = $i*$i;
    echo "$square";
    echo "<BR>";
}
?>
```

3. **Save the file as square.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/square.php in the Address bar and press Enter.**

Figure 13.3 displays the output of the script.

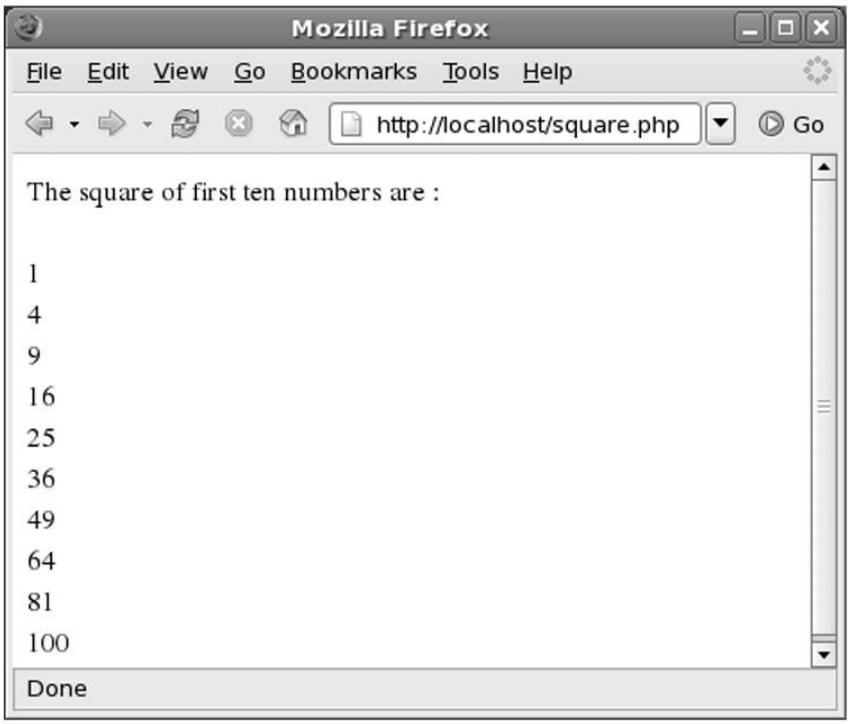


Figure 13.3: Using the for Loop

Session 13

Flow Control in PHP (Lab)

Using the Jump Statement

The jump statement controls the execution of the loop statements. The various jump statements are namely, `break`, `continue`, and `exit`.

The `break` statement stops the execution of the current loop and the control moves to the next loop. The `continue` statement skips the code following the `continue` statement in the loop body and forcibly executes the next iteration of the loop. The `exit` statement ends the loop and the control is passed to the statement following the loop body. These statements are used in the `if` statement, `for` loop, `while` loop, and `do-while` loop.

To calculate and display the inverse of a number, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form for accepting a number:**

```
<HTML>
<BODY>
<FORM METHOD = "GET" ACTION = "breakinv.php">
Enter number:
<INPUT TYPE = "TEXT" NAME = "num">
<BR><BR>
<INPUT TYPE = "SUBMIT" NAME = "submit" VALUE = "INVERSE">
</FORM>
</BODY>
</HTML>
```

3. **Save the file as break.html under the /usr/local/apache2/htdocs/ directory.**

4. **Open a new file in the text editor.**

5. **Enter the following code to display the inverse of the number:**

```
<?php
$number = $_GET['num'];
if($number==0)
{
    exit;
}
$ans = 1/$number;
```

Session 13

Flow Control in PHP (Lab)


```
echo "<BR>Inverse of the entered number is 1/$number";
echo "<BR><BR>Its decimal equivalent is $ans";
echo "<BR><BR><a href=break.html>Go Back</a>";
?>
```

6. **Save the file as breakinv.php under the /usr/local/apache2/htdocs/ directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/break.html in the Address bar and press Enter.**

   Figure 13.4 displays the output of the script.

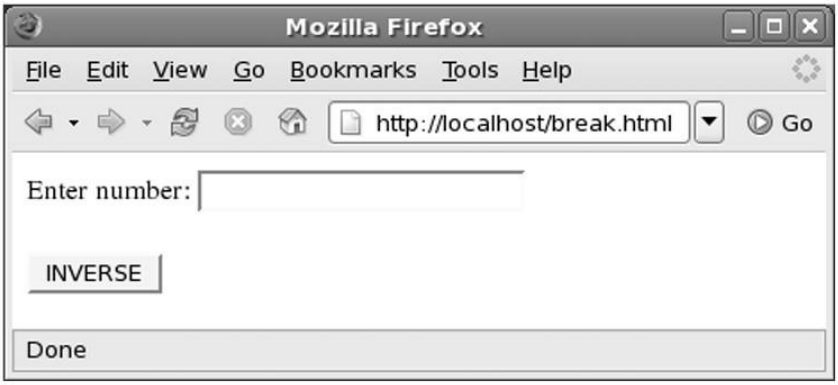


Figure 13.4: Displaying the Form

9. **Enter 5 in the number box.**

10. **Click INVERSE.**

Session 13

Flow Control in PHP (Lab)

Figure 13.5 displays the output of the script.

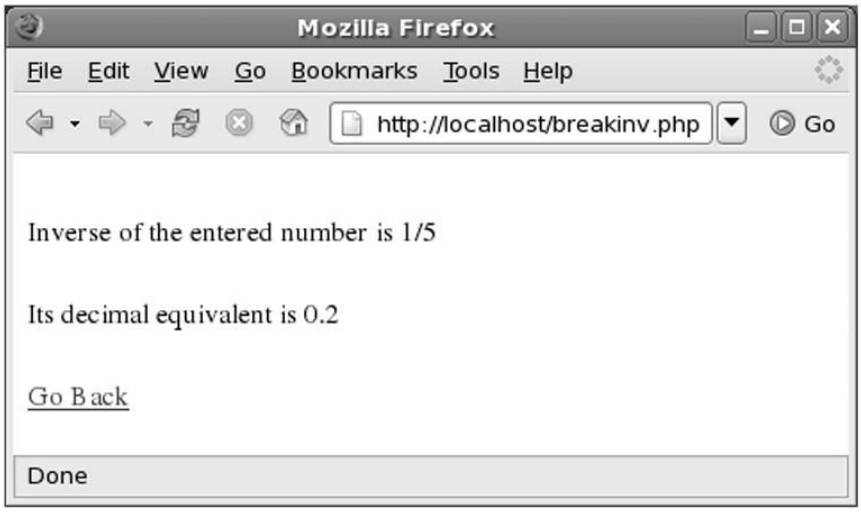


Figure 13.5: Using the jump Statement

Session 13

Flow Control in PHP (Lab)

Do It Yourself

1. Write a program to display the cube of the first 10 numbers.

2. Write a program to check whether the user is an adult, senior, youth, teenager, or minor by accepting the age of the user.

Onlinevarsity

LEARN @ YOUR COMFORT

Session 14

Functions in PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain functions in PHP.*
- *Describe the built-in functions in PHP.*
- *Explain the process of creating a user-defined function.*
- *Explain the process of passing arguments to a function.*
- *Explain the process of returning values from a function.*
- *Explain the use of recursive functions.*

14.1 Introduction

Functions can be defined as a named section of a program that performs a specific task. They enable to split a program into modules. Since, you do not have to go through the entire code and make changes everywhere, it is easier to modify a program containing functions. This means that you would only have to modify the code in the function, thus saving a lot of time.

In this session, you will learn about functions. You will also learn how to use different built-in functions and how to create a user-defined function in PHP. In addition, you will learn how to pass arguments and return values from the function.

14.2 Functions

Sometimes in a program, there are blocks of code that are used repetitively. Instead of writing the same code several times, you can use a function. A function groups together a number of statements into a single unit. This group of statements performs a specific task. It enables the developer to reuse the same piece of code in the program without typing it again.

Functions help in organizing a program, that is, they enhance the logical flow in a program by dividing complicated code sequences into smaller modules. In any software, a module is always part of a program. Programs consist of one or several modules that are developed independently, which are combined only when the program is linked. A single module can contain one or more routines.

Functions enable to write a piece of code and assign a name for it. The function can be executed or invoked using the assigned name anywhere in the program.

Session 14

Parameters are included within a function to add more functionality. Parameters are like variables and are specified within the parentheses after the name of a function.

PHP provides different built-in functions. PHP also allows to define a function, create a function, and assign a specific task to it. These functions are known as user-defined functions.

14.3 Built-in PHP Functions

PHP provides various built-in functions, that can be grouped into following categories:

- Mathematical functions
- String functions
- Date and time functions
- Error handling functions
- Database functions
- Array functions
- Mail functions

Built-in functions can be included in the PHP script.

14.3.1 Mathematical Functions

Mathematical functions operate on numerical data. Table 14.1 lists and describes some of the mathematical functions in PHP.

Function Name	Syntax	Description
abs	abs(arg)	Returns the absolute value of the argument
max	max(arg1,arg2,....)	Returns the largest value from the specified arguments. It also allows comparing multiple arrays
min	min(arg1,arg2,....)	Returns the smallest value from the specified arguments. It also allows comparing multiple arrays
sqrt	sqrt(arg)	Returns the square root of the argument
pow	pow(base, exp)	Returns the value of the base raised to the power of the exponential
round	round(number)	Returns the nearest integer of the specified number

Session 14

Function Name	Syntax	Description
rand()	rand(min, max)	Returns a random integer
ceil()	ceil(x)	Returns the value of a number rounded upwards to the nearest integer
floor()	floor(x)	Returns the value of a number rounded downwards to the nearest integer

Table 14.1: Mathematical Functions in PHP

Code Snippet 1 illustrates the use of mathematical functions, in a script named **`maths.php`**.

Code Snippet 1:

```
<?php
echo "Mathematical Functions in PHP";
echo "<br>";
echo "The absolute value of 100 is: ";
echo abs(100);
echo "<br>";
echo "The nearest integer of 99.6 is: ";
echo round(99.6);
echo "<br>";
echo "The largest number from 99.9, 9.9999, and 99.99 is: ";
echo max(99.9,9.9999,99.99);
echo "<br>";
echo "The smallest number from 99.9, 9.9999, and 99.99 is: ";
echo min(99.9,9.9999,99.99);
echo "<br>";
echo "The square root of 256 is: ";
echo sqrt(256);
echo "<br>";
?>
```

Concepts

Figure 14.1 displays the output of the script.

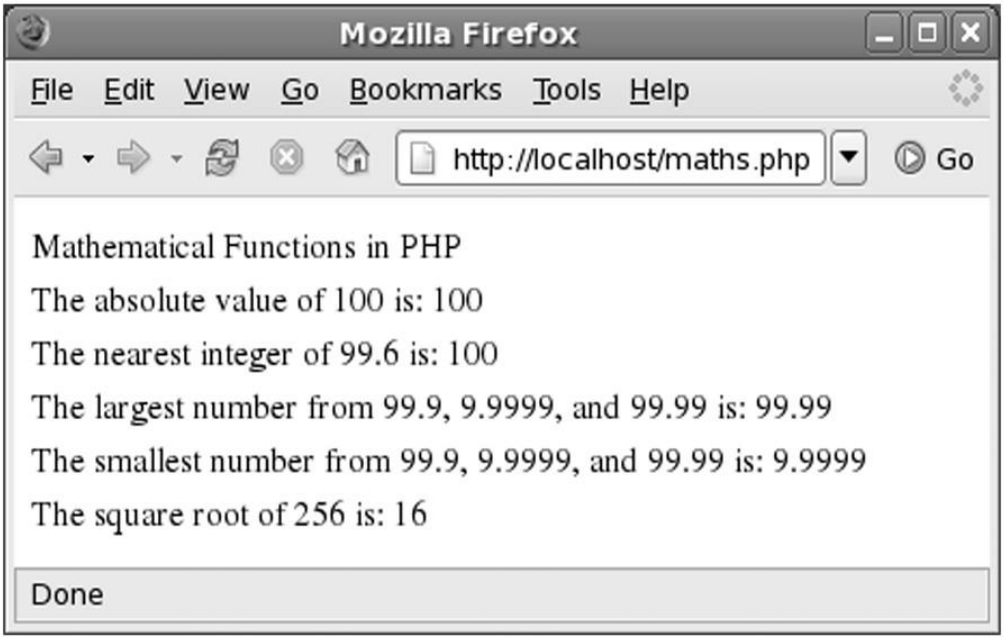


Figure 14.1: Using Mathematical Functions in PHP

14.3.2 String Functions

String functions operate on character type of data. It allows the developers to manipulate strings. Table 14.2 lists some of the string functions in PHP.

Name	General Form	Description
chr	chr(ascii)	Returns the character equivalent to the specified ASCII code
bin2hex	bin2hex(string)	Converts a string of ASCII characters to hexadecimal values
strtolower	strtolower(string)	Converts the specified string to lower case
strlen	strlen(string)	Returns the length of the string specified as an argument
strcmp	strcmp(string1,string2)	Compares two strings. Returns zero if string1 is equal to string2. It returns less than zero, if string1 is less than string2. Otherwise, it returns greater than zero when string1 is greater than string2.
strtoupper	strtoupper(string)	Converts the specified string to upper case
strrev	strrev(string)	Returns the reverse of the string

Session 14

Name	General Form	Description
stristr()	stristr(string,search)	Finds the first occurrence of a string inside another string (case-insensitive)
strrchr()	strrchr(string,char)	Finds the last occurrence of a string inside another string
strrpos()	strrpos(string,find,start)	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strncmp()	strncmp(string1,string2,length)	String comparison of the first n characters (case-sensitive)

Table 14.2: String Functions in PHP

Code Snippet 2 illustrates the use of string functions in PHP in a script named **`string.php`**.

Code Snippet 2:

```
<?php
echo "String Functions in PHP";
echo "<br>";
echo "The length of the string HELLO is: ";
echo strlen("HELLO");
echo "<br>";
echo "The comparison of Hello world and Hello Everybody is: ";
echo strcmp("Hello world!","Hello Everybody");
echo "<br>";
echo "The lowercase of the term PHP is: ";
echo strtolower("PHP");
echo "<br>";
echo "The uppercase of the term php is: ";
echo strtoupper("php");
echo "<br>";
echo "The ASCII value of A is: ";
echo bin2hex("A");
echo "<br>";
echo "The reverse of the term ECNALUBMA is: ";
echo strrev("ECNALUBMA");
?>
```

Concepts

Figure 14.2 displays the output of the script.

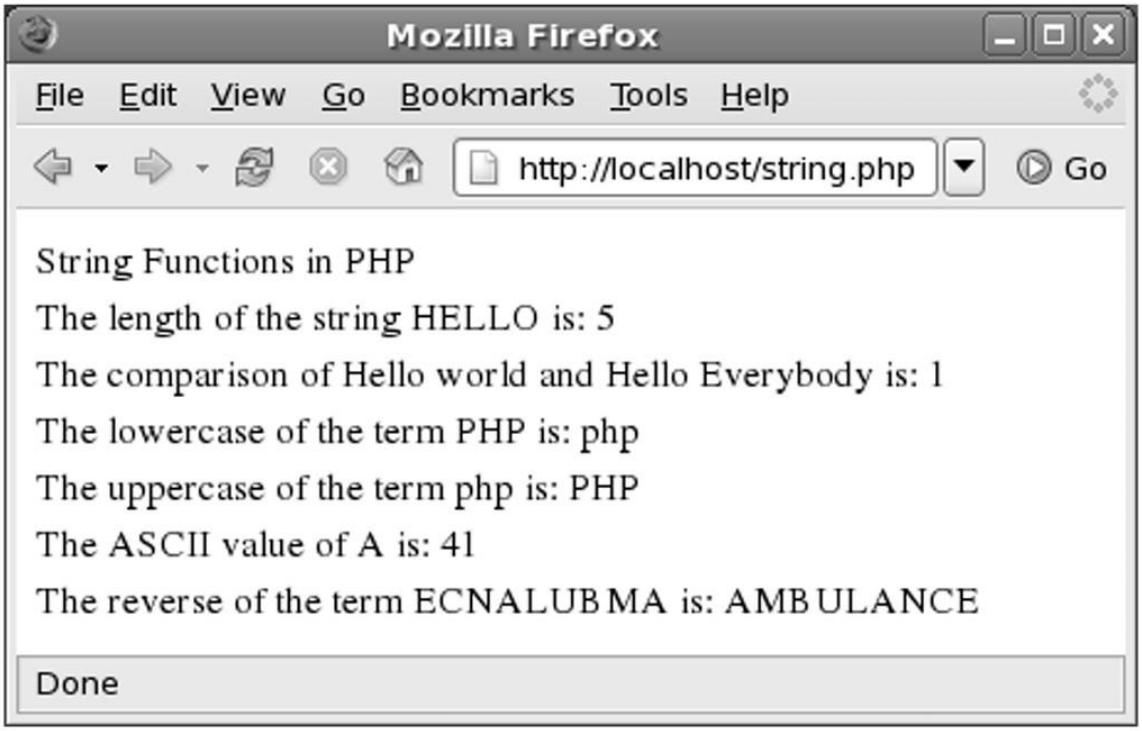


Figure 14.2: String Functions in PHP

14.3.3 Date and Time Functions

The date and time functions enable to calculate the date and time on the system and allows to extract and format the date and time on the server. Table 14.3 lists and describes some of the date and time functions.

Name	General Form	Description
checkdate	checkdate(month,day,year)	Returns the value as 1 if the specified date is valid. A valid date contains: ➢ month between 1 and 12 ➢ day within the range of days for the specified month ➢ year between 1 and 32767

Session 14

Name	General Form	Description
getdate	getdate(timestamp)	Returns an array containing date and time information. The information is returned for a Unix timestamp
time	time()	Returns the current time measured in the number of seconds
strtotime	strtotime(string time [,now])	Parses an English textual date or time into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT)
Date	date(format, timestamp)	Returns a string depending on the timestamp or the current local time, if the timestamp is not specified Some of the formats that can be used are as follows: d – day of the month D – textual representation of the day j – day of the month without leading zeros (1 to 31) m – numeric representation of the month M – textual representation of the month y – a two digit representation of the year Y – a four digit representation of the year a – Lowercase am or pm h – 12 hour format of an hour H – 24 hour format of an hour i – minutes with leading zeros s – seconds with leading zeros

Table 14.3: Date and Time Functions in PHP

Code Snippet 3 illustrates the use of common date and time functions in PHP, in a script named **`date_time.php`**.

Code Snippet 3:

```
<?php
date_default_timezone_set('Asia/Calcutta');
echo "Today is : " .date("l");
$Today_Date=getdate();
```

Concepts

```
$current_month=$Today_Date['month'];
echo "<br>";
echo "Current month is: ";
echo $current_month;
?>
```

Figure 14.3 displays the output of the script.

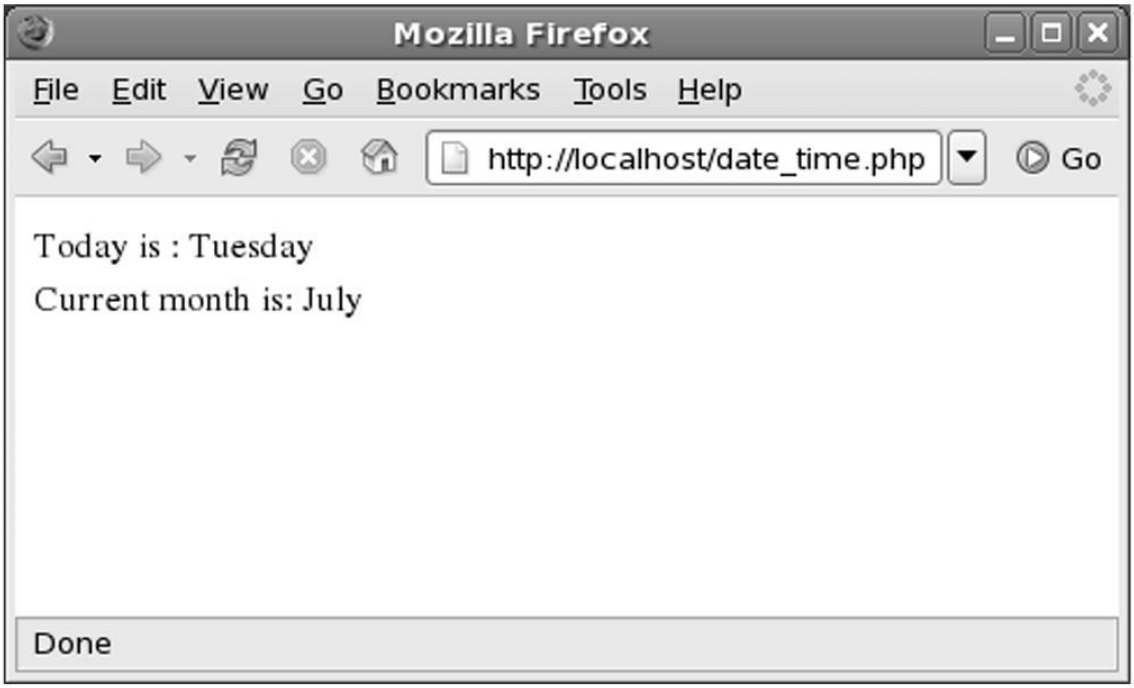


Figure 14.3: Date and Time Functions in PHP

In the code, the "l" is the argument string, which corresponds to the textual representation of the day of the week and will return the day.

14.3.4 Error Handling Functions

Error handling is the process of troubleshooting errors as they arise during the execution of a program. PHP provides error handling functions to define the error handling rules and modify the way the errors are handled.

Table 14.4 lists the error handling functions.

Function Name	General Form	Description
trigger_error	trigger_error(error_msg [,error_type])	Generates an error message, that is, it defines an error message at a specified condition as specified by the user. User-defined function such as set_error_handler() inbuilt error handler can be used with it
set_error_handler	set_error_handler(error_handler)	Handles errors during runtime by using a user-defined function
error_reporting	error_reporting(Constant)	Specifies which PHP errors are reported. PHP provides many levels of errors. You can use this function to set a level during the run-time of the script.

Table 14.4: Error Handling Functions in PHP

The `error_reporting()` function requires constants as arguments.

Table 14.5 lists some of the error constants.

Constant Name	Value	Description
E_ERROR	1	Displays errors which are not recoverable
E_WARNING	2	Displays non-fatal run-time errors, however, this does not halt the execution of the script
E_PARSE	4	Displays the errors generated by the parser during the compilation
E_NOTICE	8	Displays run time error notices
E_COMPILE_ERROR	64	Displays compile time errors
E_USER_ERROR	256	Displays user generated error message
E_USER_WARNING	512	Displays user generated warning message
E_CORE_ERROR	16	Displays fatal errors during PHP start up
E_CORE_WARNING	32	Displays non-fatal errors during PHP start up
E_COMPILE_WARNING	128	Displays errors generated by the Zend Scripting Engine
E_ALL	8191	Displays all errors and warnings that are supported

Table 14.5: Error Constants

Session 14

Concepts

Code Snippet 4 illustrates the use of error handling functions. The code generates a user-defined error, in a script named **`user_error.php`**.

Code Snippet 4:

```
<?php
$num1=0;
if($num1==0)
{
     echo "Dividing by zero";
     trigger_error("Cannot divide by zero", E_USER_ERROR);
}
else
{
     $B=100/$num1;
}
?>
```

Figure 14.4 displays the output of the script.

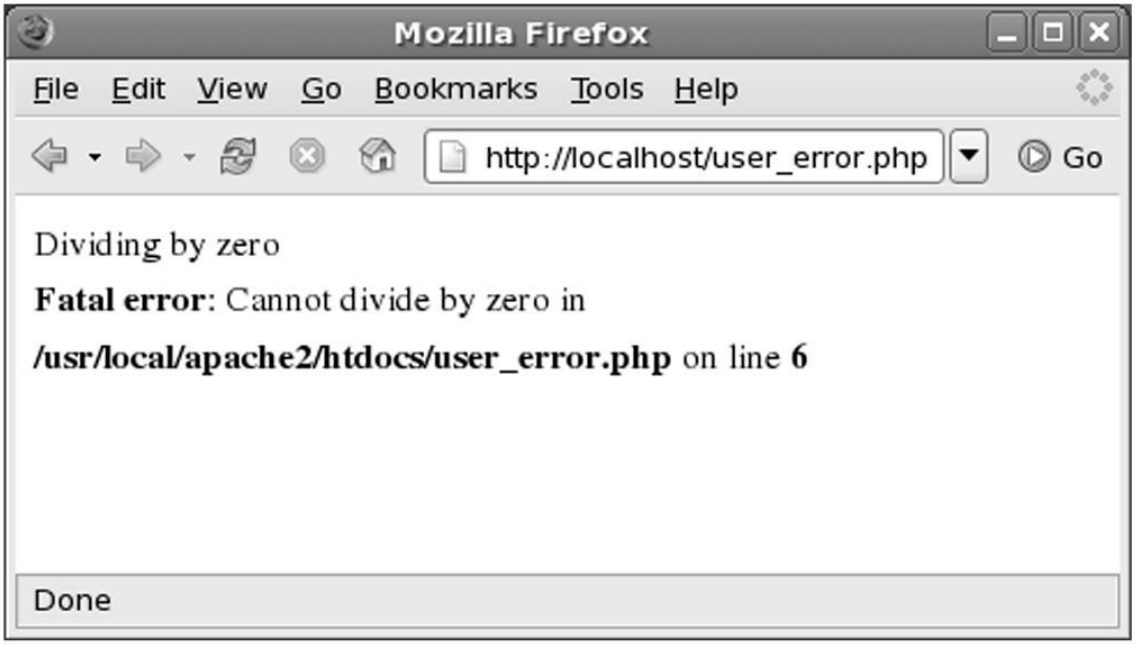


Figure 14.4: Generating User Defined Error

In the code, the value of **$num1** variable is tested. Since, the value of **$num1** is equal to zero the code in the `if` body is executed.

14.4 User-defined Functions

You can also define or create a function. A function has to be defined before it can be used. The function definition contains the code to be executed.

The syntax for defining a function is as follows:

Syntax:

```
function function_name (arg1, arg2, arg3, …)
{
      statement list
}
```

The `return expr` statement within the body of the function is used to return a value from a function. The `return` keyword stops the execution of the function and returns specified `expr` as the value.

Code Snippet 5 accepts an argument, `$x`, and returns its square.

Code Snippet 5:

```
function square ($x)
{
     return $x*$x;
}
```

After defining the function, the developer can invoke the function like an expression wherever required.

Code Snippet 6 demonstrates how to use the function defined in Code Snippet 5.

Code Snippet 6:

```
print "The square of 12 is" square(12);
```

Session 14

Functions in PHP

Concepts

Invoke the function in order to execute it. The syntax to invoke a function is as follows:

Syntax:

```
fun_name();
```

where,

`fun_name` – specifies the name of the function

A PHP code can be included inside a function, other functions, and class definitions. The rules to define function names are similar to the rules for defining labels in PHP. A valid function name should begin with a letter or underscore, followed by any number of letters, numbers, or underscores.

It is not required to define an entire function before referencing it. However, it is essential to conditionally define a function before referencing it. When a function is defined n a conditional manner, the script must first process the function definition before invoking the function.

Code Snippet 7 illustrates the use of user defined functions in PHP in a script named **`user_func.php`**.

Code Snippet 7:

```
<?php
//A function to calculate the sum of two variables
function addition()
{
     $A=100;
     $B=200;
     $C=$A+$B;
     echo "The sum of 100 and 200 is: $C";
}
addition();
echo "<br>";
// A function to display the text
function Display()
{
     echo "LEARNING PHP IS FUN";
}
Display();
?>
```

Figure 14.5 displays the output of the script.

Figure 14.5: User Defined Functions in PHP

14.5 Passing Arguments to Functions

PHP supports passing of arguments to a function. The three different ways of passing arguments to a function are as follows:

- Passing arguments by value
- Passing arguments by reference
- Setting default values for arguments

The function definition determines the method of passing arguments to the function. A function can be defined to accept multiple arguments. The syntax to define a function with arguments is as follows:

Syntax:

```
function fun_name(arg1,arg2,...)
{
     Code to be executed
}
```

where,

`arg` - specifies the argument that is passed to the function

Concepts

14.5.1 Passing Arguments by Value

When an argument is passed to the function as a value, the value of the argument remains unchanged outside the function. The arguments are prefixed with the dollar ($) sign in the function definition to indicate that the argument will be passed by the value.

The argument can be any valid expression, the expression is evaluated, and its value is assigned to the corresponding variable in the function.

Code Snippet 8 illustrates the process of passing arguments by value, in a script named **arg_value.php**.

Code Snippet 8:

```
<?php
//Creating a function to calculate the square of a number

function Square($A)
{
   //The argument is passed by Value in the function definition
   //using the $ sign.
   //Calculating the square of the number

   $A=$A*$A;
   //Displaying the square of the number
   echo $A;
}

//Assigning a value to the variable
$A=5;

//Displaying the text
echo "The square of $A is: ";

//Calling the function
Square($A);

// Creating a function to subtract one variable from another
function subtraction($A,$B)
{
```

```
    //Calculating the difference
    $C=$A-$B;
    //Displaying the text
    echo "<br>The difference of $A and $B is: $C";
    }
//Calling the function and assigning values to the argument
subtraction(90,45);
?>
```

Figure 14.6 displays the output of the script.

Figure 14.6: Passing Arguments by Value

In the code, the argument is passed by value. The variable **$A** is passed to the **Square()** function, which multiplies **$A** with **$A** and stores the result in the same variable. When the function is called, the output is displayed as 25. The value of variable, **$A** outside the function **Square($A)** is 5 because the execution of a function does not affect the value of the variable outside the function. Two arguments are passed to the function **subtraction()**. The **subtraction()** function subtracts the variable **$B** from **$A** and stores the resultant value in **$C**.

Concepts

14.5.2 Passing Argument by Reference

Values can be passed to the argument by reference. When a value is passed by reference to the argument, the value of the argument changes outside the function. While defining a function, prefix the arguments with the ampersand (&) sign to indicate that the value is passed by reference.

When you pass an argument to a function by reference, the argument must be a variable. When you modify the value of the variable within a function, the instance of the variable outside the function is also modified. You must pass arguments to a function by reference in order to enable the function to modify the argument. The ampersand (&) symbol is prefixed to the argument name in the function definition in order to pass an argument to the function by reference.

Code Snippet 9 illustrates the process of passing values to the function by reference, in a script named **`arg_ref.php`**.

Code Snippet 9:

```
<?php
//Defining a function and passing value to the arguments by reference
function Square(&$A)
{
   //Calculating the square of the number and storing it in a variable
   $A=$A*$A;
   //Displaying the result
   echo $A;
}
//Assigning value outside the function
$A=5;
//Displaying text
echo "The square of $A is: ";
//Executing the function by passing value to argument by reference
Square($A);
//Defining a function and passing value to the arguments by reference
function multiplication(&$A,&$B)
{
    //Calculating the multiplication of two numbers and storing it in a
    //variable
    $C=$A*$B;
    //Displaying text
    echo "<br><br>The multiplication of $A and $B is: $C";
}
```

```
//Assigning value outside the function
$A=25;
$B=30;
//Executing the function by passing value to argument by reference
multiplication($A,$B);
?>
```

Figure 14.7 displays the output of the script.

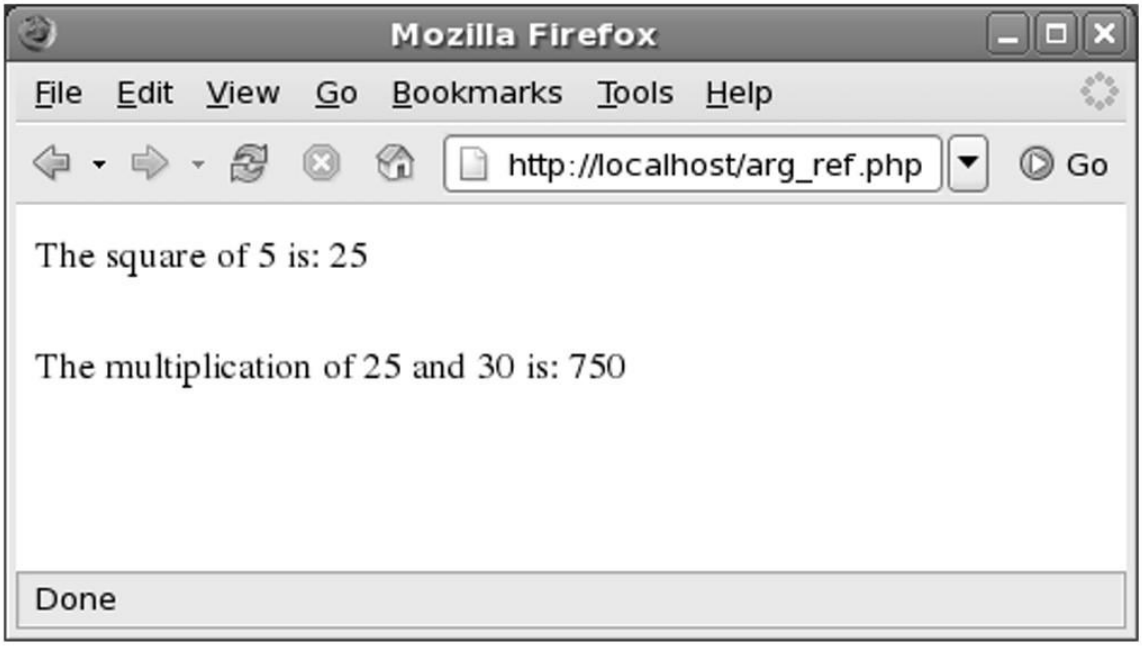


Figure 14.7: Passing Arguments by Reference

In the code, the value has been passed to the argument by reference. When the variable **$A** is passed to the **Square()** function, it multiplies **$A** with **$A** and stores the resultant value in the same variable. Hence, the output displayed is 25. In addition, when the value of **$A** is displayed outside the function, the output will be 25. This is because the execution of function changes the value of the variable even outside the function.

Two arguments are passed to the function **multiplication()**. The **multiplication()** function calculates the product of two variable **$A** and **$B** and stores the value in **$C**.

Concepts

14.5.3 Setting Default Values

You can also assign default values to the arguments. The default values for the argument must be assigned in the function definition. The function would process only the default value, when a default value is defined for the argument in the function definition.

PHP enables to assign default values to an argument in a function. The default values enable the developer to initialize the function parameters when the function is invoked without any value being passed. The default value assigned can be any one of the following:

- Constant
- Scalar
- Array with scalar values or constant

Code Snippet 10 illustrates the use of default parameters.

Code Snippet 10:

```
function increment(&$num, $increment = 1)
{
     $num += $increment;
}
$num = 4;
increment($num);
increment($num, 3);
```

In the code, **$num** is incremented by 8. Initially **$num** is incremented by 1 because the default increment is defined as 1. The variable **$num** is then incremented by 4.

Code Snippet 11 illustrates the process of assigning default values for an argument, in a script named **travel.php**.

Code Snippet 11:

```
<?php
//Creating a function and assigning default value to the argument
function T_ALLOWANCE($BASIC_SAL=100000)
{
     //Calculating the travel allowance and storing it in a variable
     $T_ALLOWANCE=0.25*$BASIC_SAL;
```

```
    //Displaying text
    echo "The travel allowance is: $T_ALLOWANCE";
}

//Executing the function

T_ALLOWANCE();
?>
```

Figure 14.8 displays the output of the script.

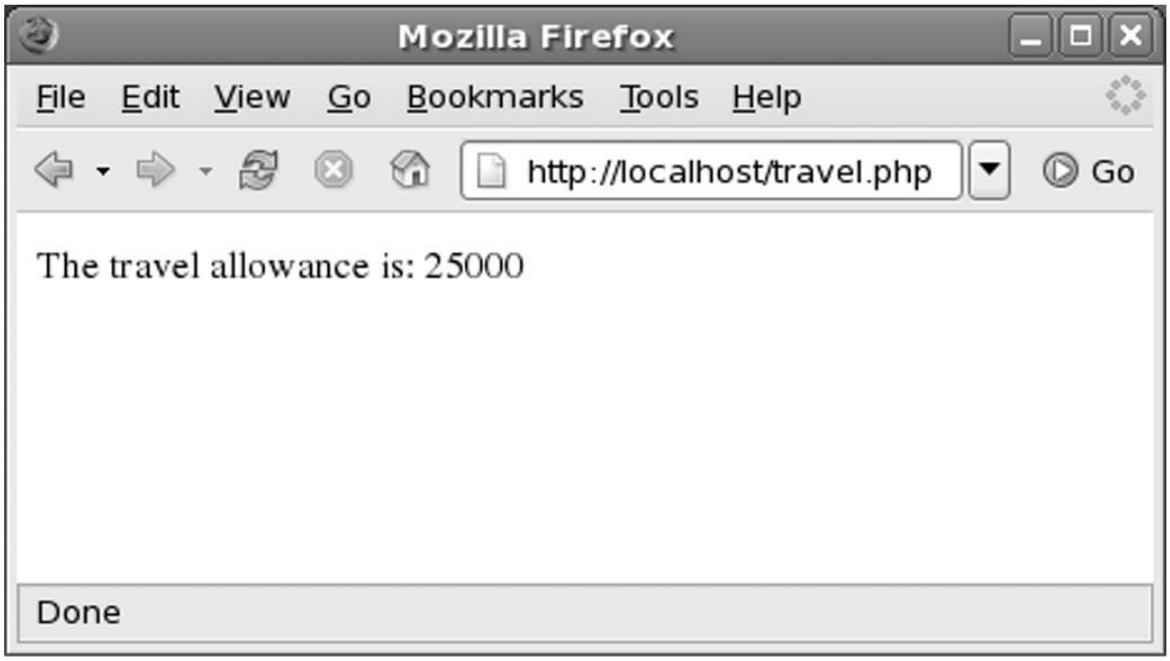


Figure 14.8: Assigning Default Values for an Argument

In the code, the **`T_ALLOWANCE()`** function calculates and displays the traveling allowance. It calculates the traveling allowance at the rate of 25% on the basic salary.

14.5.4 Returning Values from Functions

A function can also return values. The `return` statement in a function returns the value from the function.

Session 14

Functions in PHP

Concepts

The value returned can be an array or an object. The `return` keyword causes the function to stop execution and pass the control to the line from which it was invoked.

It is essential to use the reference operator `&` while declaring a function as well as when assigning the returned value to the variable. This is required in order to return a reference from a function.

Code Snippet 12 illustrates the use of `return` keyword to calculate the house rent allowance, in a script named **`hra_func.php`**.

Code Snippet 12:

```
<?php
//Creating a function
function HRA($Basic_Sal)
{
     //Calculating the HRA and storing it in a variable
     $HRA=0.25*$Basic_Sal;
     //Returning the value stored in the variable
     return $HRA;
}
//Storing the output of the function in a variable after setting a
//default value
$B=HRA(20000);
//Displaying the text
echo "The HRA is: ";
//Displaying the output
echo $B;
?>
```

Figure 14.9 displays the output of the script.

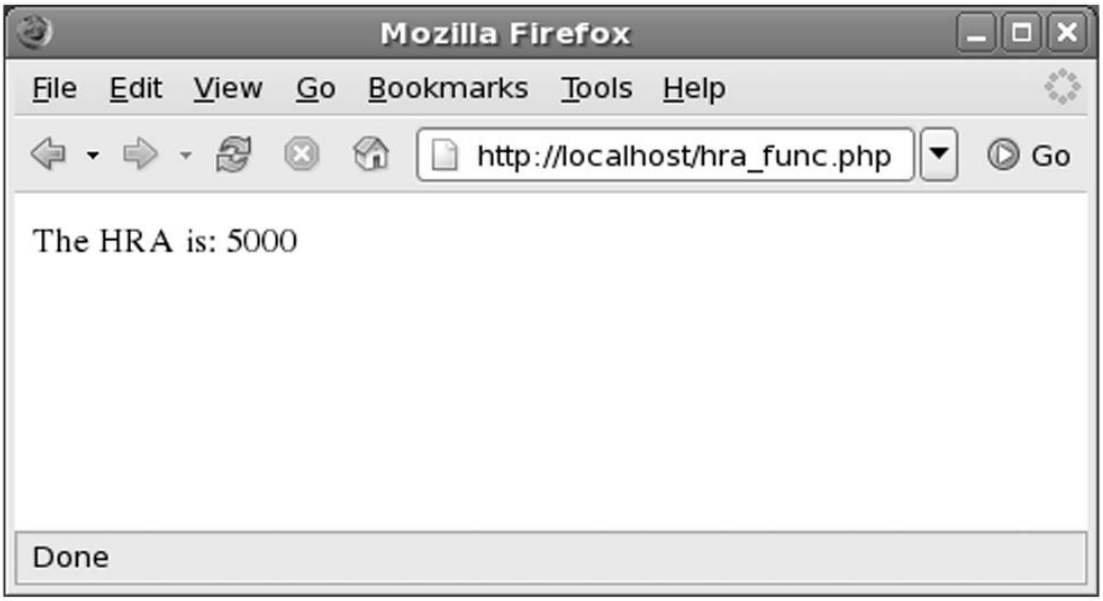


Figure 14.9: Returning Values from Functions

In the code, the **HRA()** function returns the house rent allowance, calculated at the rate of 25% of basic salary. The return **$HRA** statement returns the HRA calculated in the **HRA()** function.

14.5.5 Nesting of Functions

There can be different functions in a program. One function can be dependent on another function in the program. When one function is being executed, it can call or execute another function. The execution of one function inside another function is called nesting of functions.

Code Snippet 13 illustrates the use of nested functions in a script named **nested.php**. In the script, define a function named **Net_Salary()**, which in turn calls functions, such as **HRA()**, **TAX()**, and **DA()**.

Code Snippet 13:

```
<?php
//Assigning value to variable
$Basic_Sal=75000;
//Creating a function and passing value by reference
```

Session 14

Functions in PHP

Concepts

```
function HRA($Basic_Sal)
{
    //Calculating the HRA
    $HRA=3/10 * $Basic_Sal;
    //Displaying Text
    echo "Your HRA is:  ";
    //Displaying the computed HRA
    echo $HRA;
    echo "<br>";
}
//Creating a function and passing value by reference
function TA($Basic_Sal)
{
    //Calculating the TA
    $TA=1/4*$Basic_Sal;
    //Displaying Text
    echo "Your TA is: ";
    //Displaying the computed TA
    echo $TA;
    echo "<br>";
}
//Creating a function and passing value by reference
function TAX($Basic_Sal)
{
    //Calculating the tax
    $TAX=1/10*$Basic_Sal;
    //Displaying Text
    echo "Your TAX is:  ";
    //Displaying the computed tax
    echo $TAX;
    echo "<br>";
}
//Creating a function and passing value by reference
function Net_Salary($Basic_Sal)
{
   //Storing tax, HRA and TA in variables
   $A=3/10 * $Basic_Sal;
   $B=1/4*$Basic_Sal;
   $C=1/10*$Basic_Sal;
```

```
    //Calculating the Net Salary
    $Net_Sal=75000+$A+$B-$C;
    //Displaying Text
    echo "Your Net Salary is:";
    //Displaying the Net Salary
    echo $Net_Sal;
}
//Calling the functions
HRA($Basic_Sal);
echo "<br>";
TA($Basic_Sal);
echo "<br>";
TAX($Basic_Sal);
echo "<br>";
Net_Salary($Basic_Sal);
echo "<br>";
?>
```

Figure 14.10 displays the output of the script.

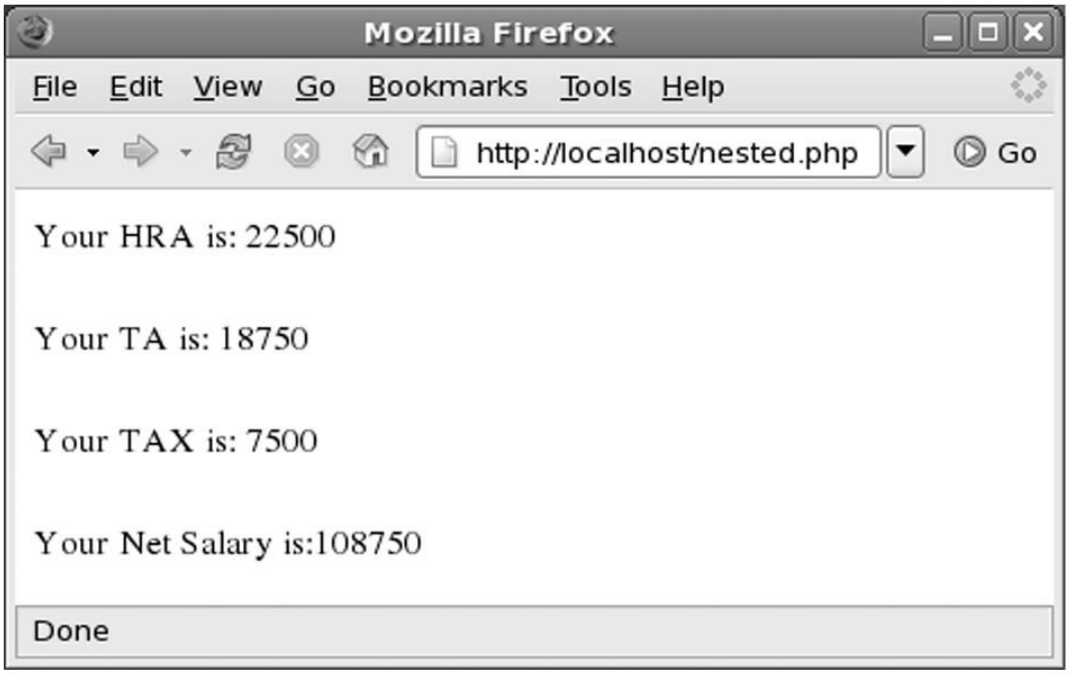


Figure 14.10: Nesting of Functions

Concepts

In the code, the **Net_Salary()** function calls three functions, **HRA()**, **TA()**, and **TAX()**.
The **Net_Salary()** function calculates the net salary using the values returned by the three functions.

14.6 Recursion

The execution of a function within another function is called nested functions. When a function executes itself repeatedly it is known as recursion. When a function calls itself, the same block of code is executed.

You can use recursive functions in programs to calculate the factorial, as shown in Code Snippet 14, in a script named **recursion.php**.

Code Snippet 14:

```
<?php
//Assigning Value to a variable
$A=10;
//Creating a function to calculate the factorial
function factorial($A)
{
     //Calculating the factorial
     if($A<=1)
     {
         return 1;
     }
     else
     {
         return $A * factorial($A-1);
     }
}
//Displaying Text
echo "The factorial of $A is: ";
//Assigning the result to a variable
$B = factorial($A);
//Displaying the result
echo $B;
?>
```

Figure 14.11 displays the output of the script.

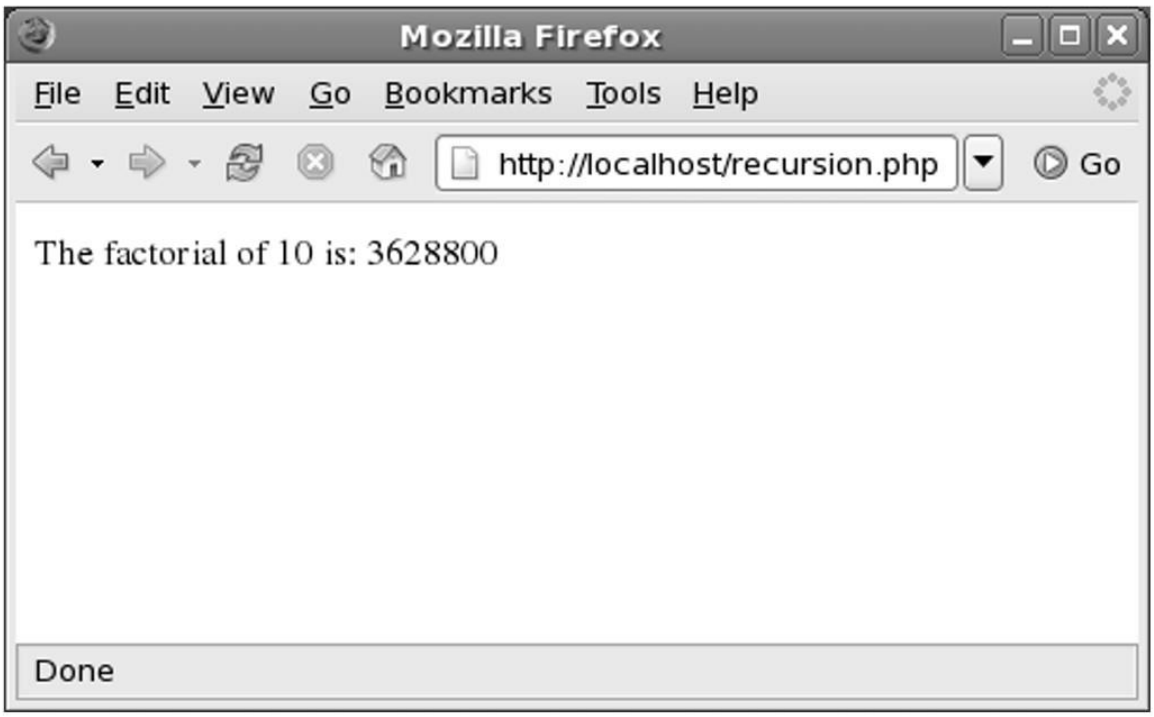


Figure 14.11: Using Recursive Functions

In the code, the **`factorial()`** function returns 1 if the input is 0 or 1. In any other condition, the **`factorial()`** function executes itself to calculate the factorial. The function in Code Snippet 14 will return 24 as the output.

The code written using recursive functions can be interpreted easily. The only disadvantage of recursive functions is that they require more time to be executed as compared to non-recursive functions. You can also use non-recursive functions instead of recursive functions for the factorial example. In a non-recursive function, use a loop instead of recursion.

Session 14

Concepts

Summary

- Functions can be used to implement execution of repetitive lines of code.
- The built-in functions in PHP are mathematical, string, date and time, error handling, database, array, and mail functions.
- Mathematical functions operate on numerical data.
- String functions operate on character type of data.
- Date and time functions are used to display the system date and time.
- Error handling functions are used to define the error handling rules.
- Arguments can be passed to a function by value and reference. Default values can also be passed to a function.
- Recursive functions have the ability to call themselves repeatedly.

Session 14

Check Your Progress

1. Which of the following function returns the ASCII value of the first character of the string?
   a. chr()
   b. bin2hex()
   c. ord()
   d. ASCII()

2. The time() function displays the time in _____ .
   a. hours
   b. minutes
   c. seconds
   d. hours:minutes:seconds

3. The trigger_error() function is used to _____.
   a. generate a user defined error
   b. generate the PHP back trace
   c. set a user function to handle errors in script
   d. generate an error

4. When you pass the arguments by value, the arguments value _____.
   a. remains unchanged outside the function
   b. changes outside the function
   c. remains constant inside the function
   d. changes inside the function

Concepts

Check Your Progress

5. Which of the following signs are used to pass the arguments by reference to the function?

   a. $

   b. &

   c. |

   d. !

6. Which of the following code snippet will be used to compare two strings?

   a. echo strcmp('Hello world!','Hello Everybody');

   b. echo strcmp("Hello world!","Hello Everybody");

   c. echo strcmp('Hello world!','Hello Everybody'):

   d. echo strcmp(("Hello world!"),("Hello Everybody"));

Session 14

Answers

1. a
2. c
3. a
4. a
5. b
6. b

Session 15

Functions in PHP (Lab)

Objectives

At the end of this session, the student will be able to:

- *Use the built- in functions.*
- *Create user-defined functions.*
- *Pass arguments to a function.*
- *Return values from a function.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Using Mathematical Functions

PHP provides built-in mathematical functions that operate on numerical data.

For example, to accept a number and display the square root of the number using the PHP script, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form that accepts a number:**

```
<HTML>
<BODY>
<FORM METHOD = "GET" ACTION = "squareroot.php">
Enter a Number:
<INPUT TYPE = "TEXT" NAME = "n1text">
<BR>
<BR>
<INPUT TYPE="SUBMIT" NAME = "SQUARE ROOT" VALUE = "SQUARE ROOT">
</BODY>
</HTML>
```

Session 15

3. **Save the file as squareroot.html in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the text editor.**

5. **Enter the following code to calculate the square root of a number:**

```
<?php
$A=$_GET['n1text'];
echo "The square root of $A is ";
echo sqrt($A);
echo "<BR><BR><a href=squareroot.html>Go Back</a>";
?>
```

6. **Save the file as squareroot.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/squareroot.html in the Address bar and press Enter.**

   Figure 15.1 displays the output.

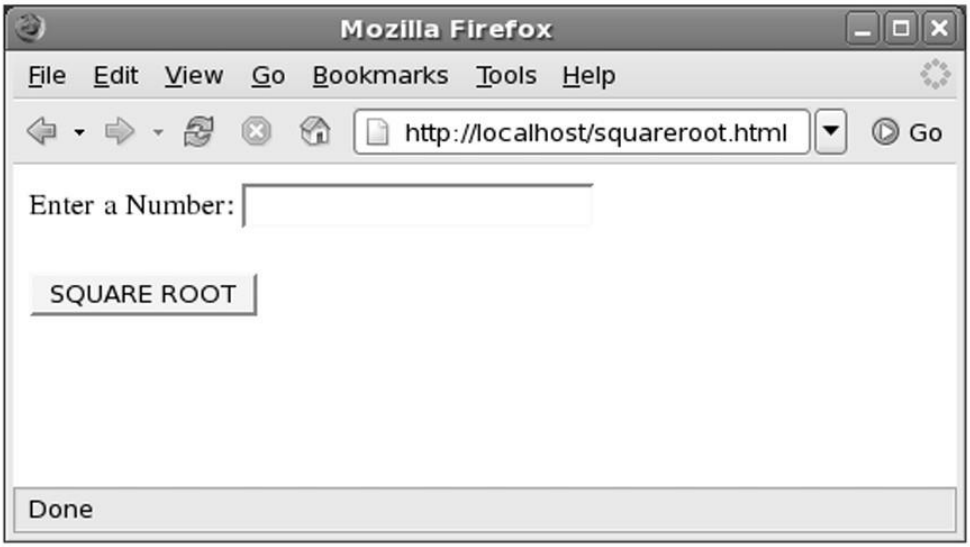


Figure 15.1: Square Root Form

9. **Enter 64 in the Enter a Number: box.**

10. **Click SQUARE ROOT.**

    Figure 15.2 displays the output of the script.

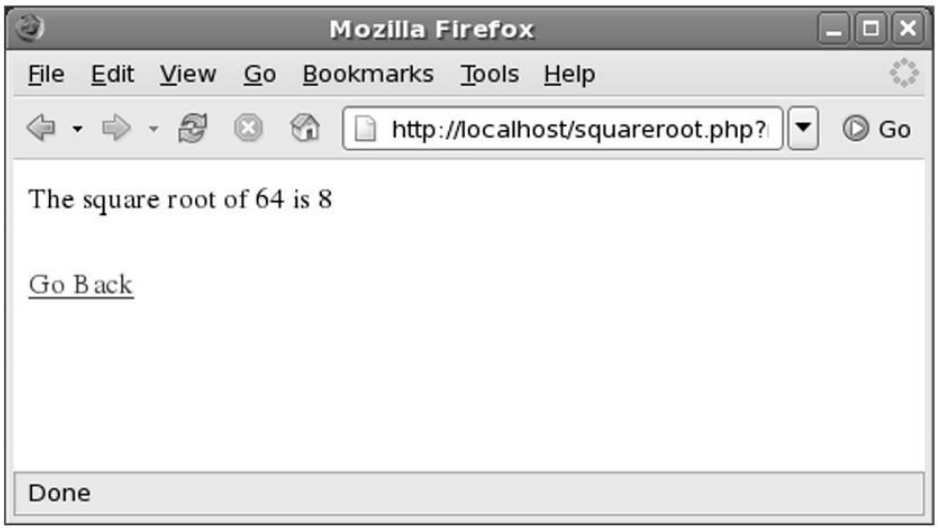


Figure 15.2: Calculating the Square Root

Using String Functions

PHP provides string functions that operate on character data. For example, to obtain the ASCII equivalent of character A using the PHP script, perform the following steps:

1. **Open a new file in the text editor.**

2. **Enter the following code:**

```
<?php
$str = "A";
$asciival = ord($str);
echo "The ASCII equivalent of A is $asciival.";
?>
```

3. **Save the file as convert.php in the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/convert.php in the Address bar and press Enter.**

   Figure 15.3 displays the output of the script.

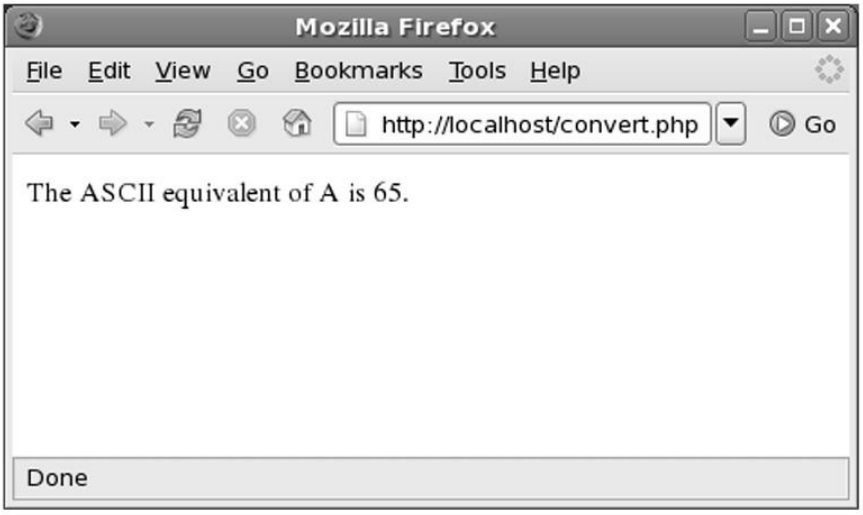


Figure 15.3: Displaying ASCII Equivalent

Using Date and Time Functions

The date and time functions in PHP enable to retrieve the date and time on the system.

For example, to display the current time in seconds using PHP script, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
$B = time();
echo $B;
?>
```

3. **Save the file as time.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

Session 15

5. **Enter http://localhost/time.php in the Address bar and press Enter.**

   Figure 15.4 displays the output of the script.

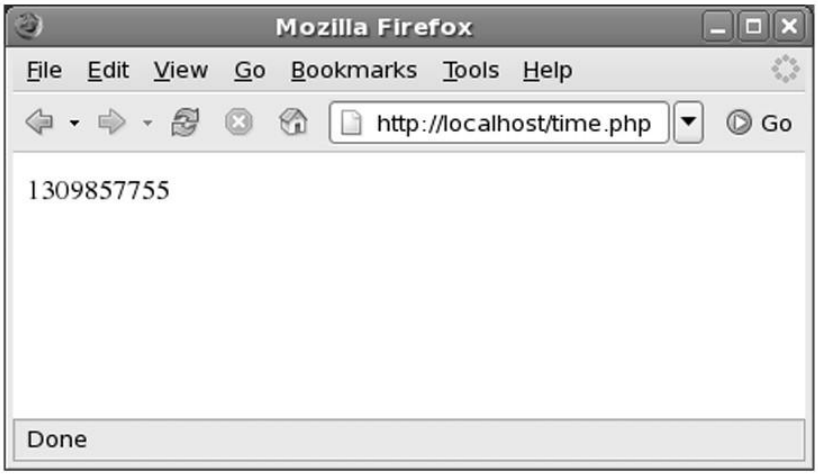


Figure 15.4: Using Date and Time Functions

Creating User-Defined Functions

In addition to the built-in functions, you can also define or create functions to perform tasks.

For example, to create a function that calculates the remainder, perform the following steps:

1. **Open a new file in the gedit text editor.**
2. **Enter the following code:**

```
<?php
function remainder()
{
   $A=59;
   $B=6;
   $C=$A%$B;
   echo $C;
}
echo "The remainder is ";
remainder();
?>
```

Session 15

3. **Save the file as remainder.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/remainder.php in the Address bar and press Enter.**

   Figure 15.5 displays the output of the script.

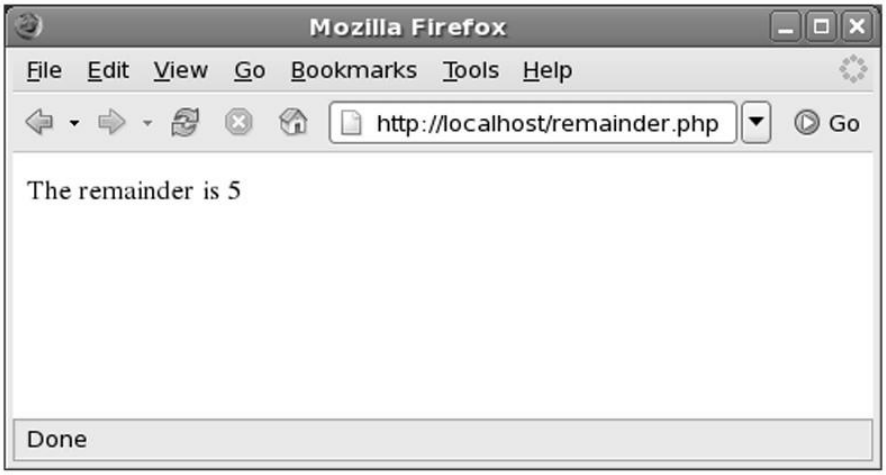


Figure 15.5: Displaying the Remainder

Passing Argument to a Function by Value

When you pass the argument to a function by value, the value of the variable is modified only inside the function. The change in the value of the variable is not reflected outside the function.

For example, to accept two numbers and display their product using a function, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form that accepts two numbers:**

```
<HTML>
<BODY>
<FORM METHOD = GET ACTION = "multiply.php">
Enter the first number:
<INPUT TYPE = "TEXT" NAME = "num1text">
<BR>
```

Session 15

Functions in PHP (Lab)


```
Enter the second number:
<INPUT TYPE = "TEXT" NAME = "num2text">
<BR>
<BR>
<INPUT TYPE="SUBMIT" NAME = "MULTIPLY" VALUE = "MULTIPLY">
<BR>
</BODY>
</HTML>
```

3. **Save the file as multiply.html under the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to multiply two numbers:**

```
<?php
$A = $_GET['num1text'];
$B = $_GET['num2text'];
function multiply($A,$B)
{
   $C = $A*$B;
   echo "The multiplication of $A and $B: $C";
}
multiply($A,$B);
?>
```

6. **Save the file as multiply.php under the /usr/local/apache2/htdocs directory**

7. **Open the Mozilla Firefox Web browser.**

Lab Guide

8. **Enter http://localhost/multiply.html in the Address bar and press Enter.**

   Figure 15.6 displays the output of the script.

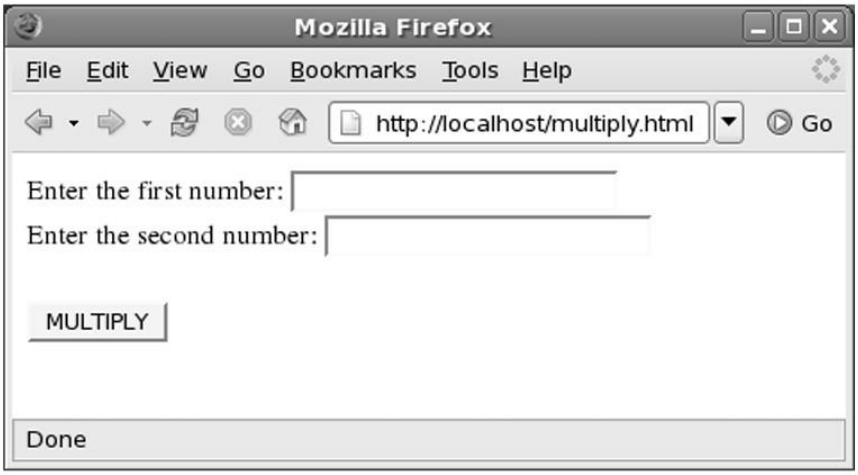


Figure 15.6: Multiplication Form

9. **Enter 62 as the first number.**

10. **Enter 66 as the second number.**

11. **Click MULTIPLY.**

    Figure 15.7 displays the output of the script.

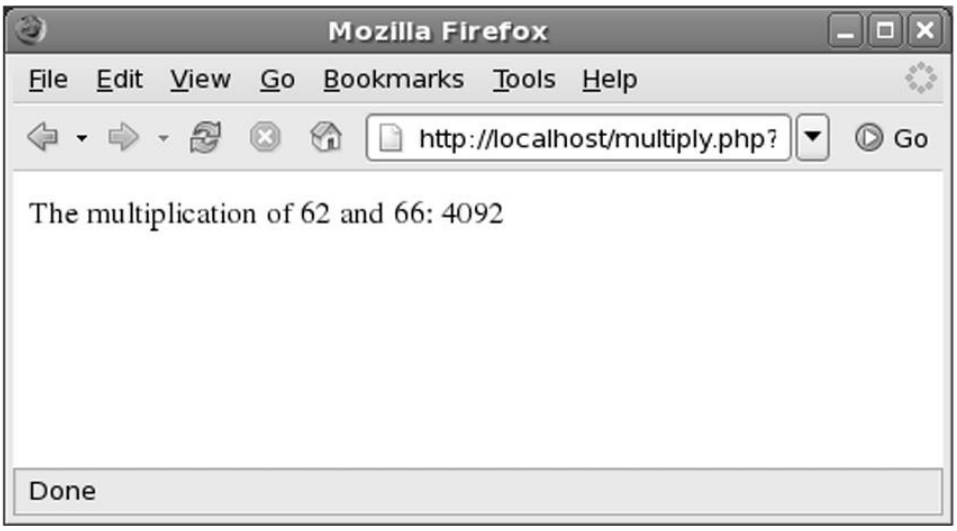


Figure 15.7: Passing Argument to a Function by Value

Passing Argument to a Function by Reference

The second method of passing argument to a function is by passing the reference. When you pass the arguments by reference, the value of the variable is also modified outside the function.

For example, to create a function that calculates the dearness allowance, perform the following steps:

1. **Open a new file in the text editor.**

2. **Enter the following code to create a form that accepts the basic salary:**

```
<HTML>
<BODY>
<H3>Calculate Dearness Allowance</H3>
<FORM METHOD = GET ACTION = "dearnessallowance.php">
Enter your basic salary in USD:
<INPUT TYPE = "TEXT" NAME = "num1text">
<BR><BR>
```

Session 15

Functions in PHP (Lab)

Lab Guide

```
<INPUT TYPE = "SUBMIT" NAME = "Dearness Allowance" VALUE = "Dearness
Allowance">
<BR>
</BODY>
</HTML>
```

3. **Save the file as dearnessallowance.html under the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to calculate the dearness allowance:**

```
<?php
$A=$_GET['num1text'];
function DA(&$A)
{
    $B = 0.1*$A;
    echo "<BR>Your salary is $A USD";
    echo "<BR>Dearness Allowance is $B USD";
}
DA($A);
?>
```

6. **Save the file as dearnessallowance.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/dearnessallowance.html in the Address bar and press Enter.**

   Figure 15.8 displays the output of the script.

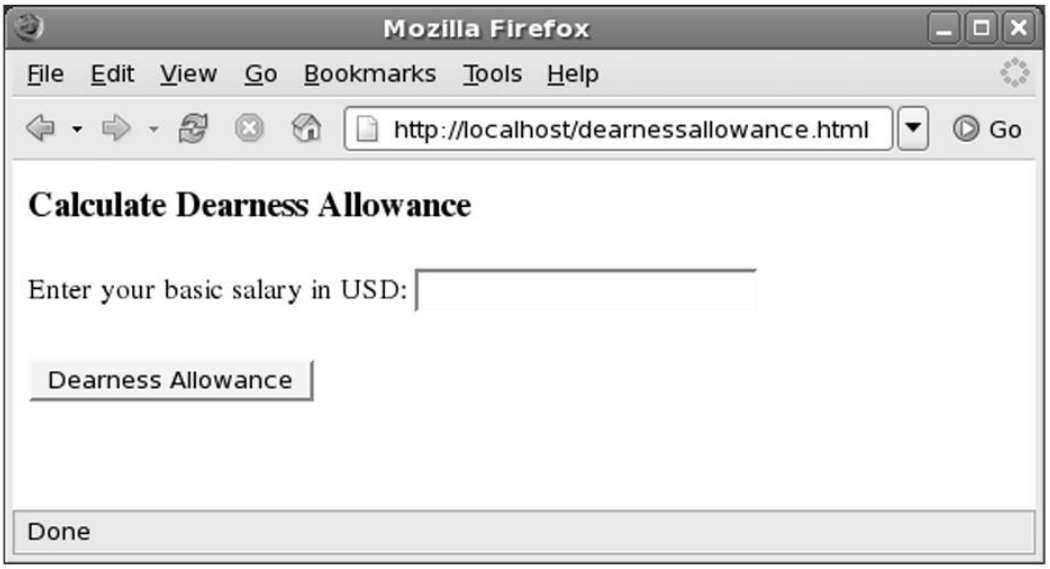


Figure 15.8: Dearness Allowance Form

9. **Enter 20000 as the basic salary.**

Session 15

10. **Click DEARNESS ALLOWANCE.**

    Figure 15.9 displays the output of the script.

Figure 15.9: Passing Argument to a Function by Reference

Setting Default Values for an Argument

It is possible to assign default values to the arguments. A default value must be assigned to the argument only in the function definition. When you attempt to pass different arguments, the function would process only the default value.

For example, to create a function that has default values for its arguments, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to calculate the sum of 50 and 100:**

```
<?php
function addition($A=50, $B=100)
{
     $C = $A+$B;
     echo "The addition of $A and $B is $C";
}
```

Session 15


```
addition();
?>
```

3. **Save the file as addition.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/addition.php in the Address bar and press Enter.**

   Figure 15.10 displays the output of the script.

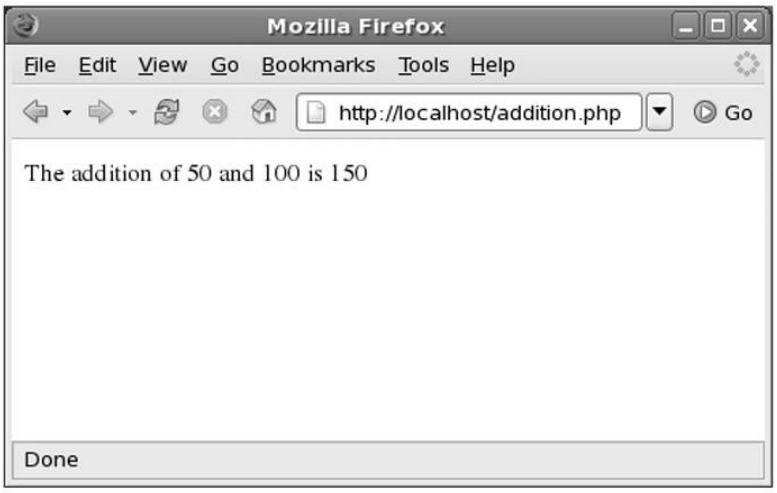


Figure 15.10: Assigning Default Values to a Function

Returning Values from Functions

A function can return values. The `return` statement returns the value from the function.

For example, to return the cube of a number from the function, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form that accepts a number:**

```
<HTML>
<BODY>
<FORM METHOD = GET ACTION = "cube.php">
Enter the number:
<INPUT TYPE="TEXT" NAME="n1text">
```

Session 15

Functions in PHP (Lab)

Lab Guide

```
<BR>
<BR>
<INPUT TYPE="SUBMIT" NAME = "CUBE" VALUE="CUBE">
<BR>
</BODY>
</HTML>
```

3. **Save the file as cube.html in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to calculate the cube of the number:**

```
<?php
$A = $_GET['n1text'];
function cube($A)
{
     $cube = $A*$A*$A;
     return $cube;
}
$B = cube($A);
echo "The cube of $A is ";
echo $B;
?>
```

6. **Save the file as cube.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

Session 15

8. **Enter http://localhost/cube.html in the Address bar and press Enter.**

   Figure 15.11 displays the output.

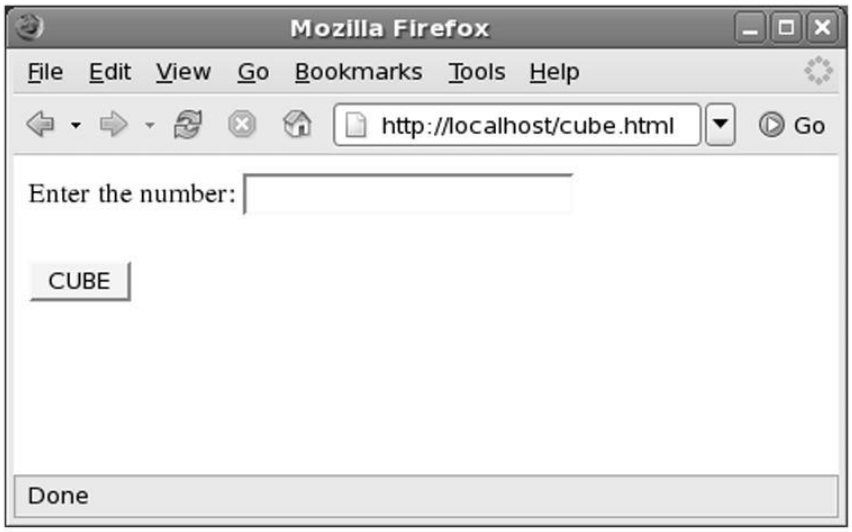


Figure 15.11: Cube Form

9. **Enter 10 in the Number text box.**

Lab Guide

10. **Click CUBE.**

    Figure 15.12 displays the output of the script.

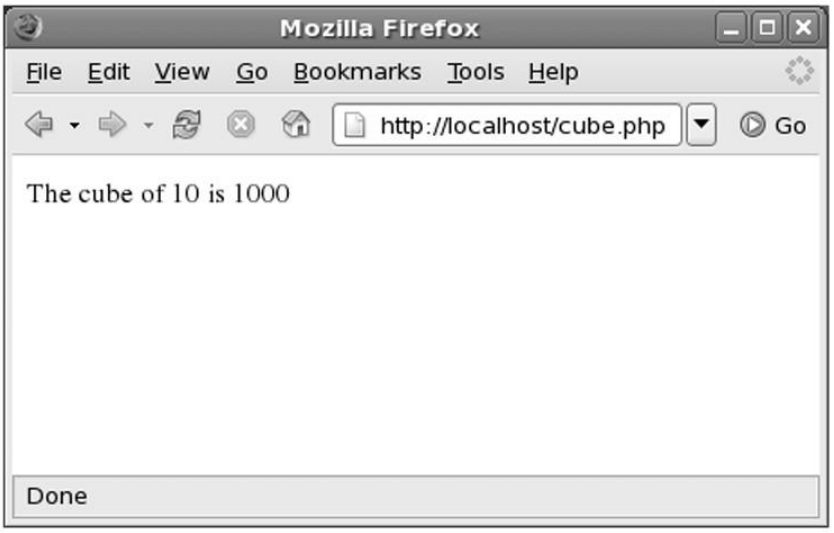


Figure 15.12: Returning Values from Functions

Creating Recursive Functions

It is possible for a function to invoke or call itself. When a function calls itself several times, the process is termed as recursion.

For example, to generate the Fibonacci series using the recursive function, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
$A = 0;
function fib($A)
{
   if ($A<=1)
   {
      return $A;
   }
```

Session 15


```
        else
        {
            return fib($A-1)+fib($A-2);
        }
    }
    while ($A!=8)
    {
       $B=fib($A);
       echo "$B<BR>";
       $A++;
    }
    ?>
```

3. **Save the file as fibonacci.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/fibonacci.php in the Address bar and press Enter.**

   Figure 15.13 displays the output of the script.

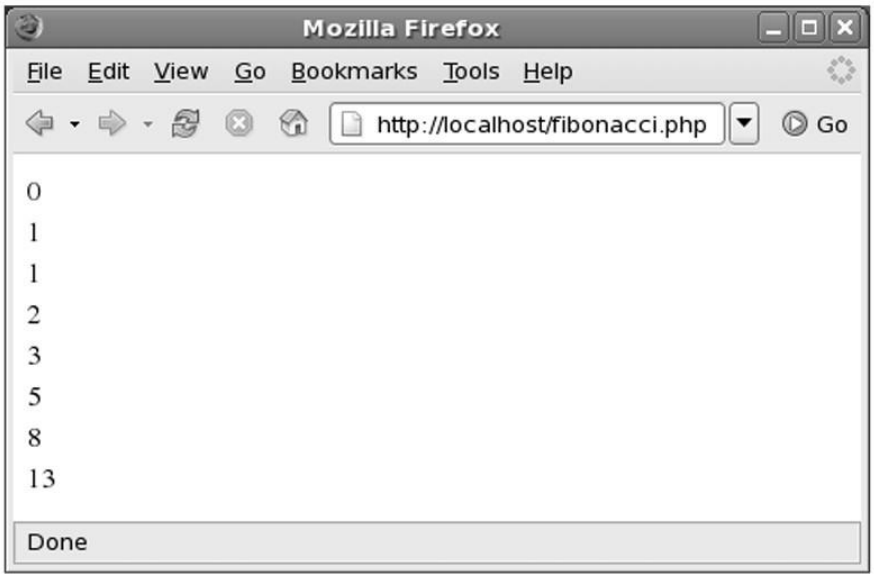


Figure 15.13: Using Recursive Functions

Session 15

Functions in PHP (Lab)

Do It Yourself

1. Find and display the minimum number from the following:

   55.5, 5555.5, 5.55555555, and 555.55555

2. Find and display the length of the string, WELCOME.

3. Create a form that accepts the principal amount. Create a user-defined function to calculate the compound interest.

Session

16 Working with Arrays

Concepts

Objectives

At the end of this session, the student will be able to:

- *Define an array.*
- *Explain the use of arrays.*
- *Explain the process of merging arrays.*
- *Explain the use of single and multi-dimensional arrays.*
- *Explain the use of array-related functions.*

16.1 Introduction

Programming languages use variables to store values. An array is a variable that can store a list of values, that is, an ordered map. Arrays can be single-dimensional or multi-dimensional. All the values are referred by the same array name.

This session introduces creation and use of arrays and array-related functions. The session also describes how to initialize arrays, use single-dimensional, and multi-dimensional arrays.

16.2 Defining an Array

A variable can store only one value at a time. You can change the value of a variable many times in a program. However, you can store only one value in a variable. An array is a variable that can store a set of values of the same data type.

Each value in an array is termed as an element. All elements in an array are referenced by a common name. Each element of an array can be referred by an array name and an index. An array index is used to access an element. An index can be a number or a string. If the index is a string, the array is an associative array. If the index is a number, the array is an indexed array.

By default, the index value in an array starts at zero. As a result, the index number of the last element in an array is one less than the total number of elements. For example, in an array of five elements, the index number of the last element will be four.

Concepts

16.3 Initializing an Array

PHP provides two ways of initializing an array and they are as follows:

- `array()` function - assigns value to all the elements of an array
- `array` identifier - assigns value to a specific element of an array

16.3.1 The array() Function

You can create an array and define the initial values of array elements using the `array()` function. The `array()` function uses key value pairs separated by a comma to create an array. The number of key value pairs in the `array()` function determines the number of elements in an array.

The syntax for creating an array using the `array()` function is as follows:

Syntax:

```
$array_name = array([key => ] value, [key => ] value)
```

where,

`array_name` - specifies the array name

`key` - specifies the index value of the array element. The key can be a string or an integer.

`value` - specifies the value of the element

Using the `array()` function, you can initialize both indexed and associative arrays.

- **Indexed Arrays** - An indexed array includes an integer as the index type. By default, PHP creates an indexed array, if the index type is not specified at the time of creating an array. The index value can start with any integer, such as 1, 20, or 123.

  To create an indexed array named department, enter the code as shown in Code Snippet 1 in a script named **`dep_ind_arr.php`**.

Session 16

Code Snippet 1:

```
<?php
// Creating an array and storing values
$department = array (1 => 'Accounts', 2 => 'Economics', 3 =>
                  'Computers', 4 => 'Marketing');
// Displaying the element of the array
echo $department [1];
?>
```

Figure 16.1 displays the output of the script.

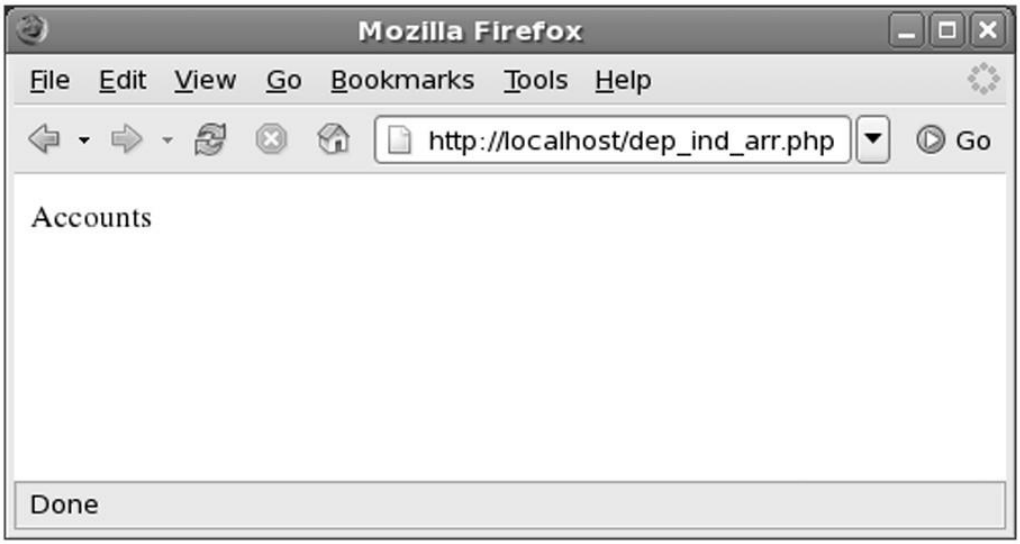


Figure 16.1: Using Indexed Arrays

In the code, the array index type is an integer. The department array contains four values, Accounts, Economics, Computers, and Marketing. When the first element of the array is called, the output returned is **`Accounts`**.

- **Associative Arrays** - An associative array is an array where the index type is a string. In associative arrays, the index value must be specified within double quotes.

Concepts

To create an associative array named **associate**, enter the code as shown in Code Snippet 2 in a script named **assoc_arr.php**.

Code Snippet 2:

```
<?php
$associate = array("a" => 'Finance', "b" => 'Sales', "c" => 'HR',
                    "d" => 'Purchase');
echo "The value of the associative array is: ";
echo $associate["c"];
?>
```

Figure 16.2 displays the output of the script.

Figure 16.2: Using Associative Arrays

In the code, the array index type is a string. The index value starts from `a`. The index values are specified within double quotes. The department array contains four values **Finance**, **Sales**, **HR**, and **Purchase**. The statement `echo $associate["c"];` displays the value associated with the index `"c"`.

16.3.2 Array Identifier

The array identifier enables to initialize the value of a specific element in an array.

Session 16

Working with Arrays

The syntax for creating an array using the array identifier is as follows:

Syntax:

```
$array_name[key] = "element_value";
```

where,

`array_name` - Specifies the name of the array

`key` - Specifies the index value of the array element. The `key` type can be a string or an integer.

`element_value` - Specifies the value assigned to the element of the array

Code Snippet 3 illustrates the use of array identifiers to create an array named department containing four values, **`Finance`**, **`Sales`**, **`HR`**, and **`Purchase`**, in a script named **`array_identify.php`**.

Code Snippet 3:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
   {
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
   }
?>
```

Concepts

Figure 16.3 displays the output of the script.

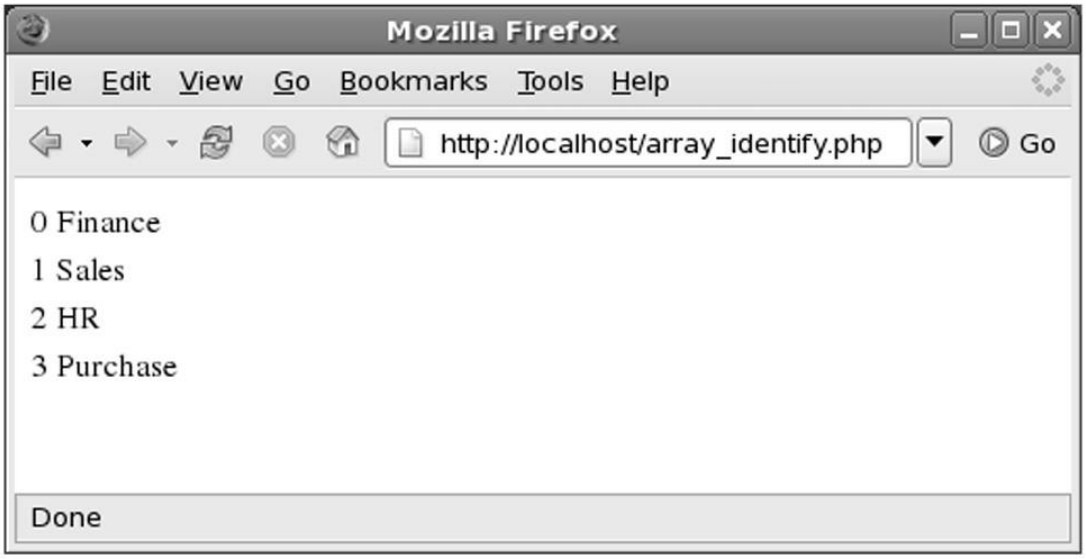


Figure 16.3: Using Array Identifiers

In the code, the `count()` function calculates the number of elements in the array and the result is stored in `$no_of_element` variable. The `each()` function retrieves each key value pair of an array and stores the result in the `$rec` variable. The `for` statement will continue to retrieve values of the array, till it reaches the last key value pair.

16.4 Merging Arrays

Merging of arrays is the process of combining the element values of two or more arrays. The `array_merge()` function is used to combine the element values of two or more arrays.

The syntax for merging arrays is as follows:

Syntax:

```
$merged_array_name = array_merge($first_array, $second_array);
```

where,

`$merged_array_name` - specifies the name of the new array that will contain the merged element values

`$first_array` and `$second_array` - specifies the names of the arrays whose elements are to be merged

Session 16

Working with Arrays

Concepts

Consider an example where there are two arrays, **ITdept** and **Salesdept**. The array, **ITdept,** contains values, such as, **Testing** and **Training**. The second array, **Salesdept**, contains values, such as, **Advertising** and **Marketing**.

Code Snippet 4 illustrates the merging of the arrays **ITdept** and **Salesdept**, in a script named **merge_array.php**.

Code Snippet 4:

```
<?php
$ITdept = array(0 => "Testing", 1 => "Training");
$Salesdept = array(0 => "Advertising", 1 => "Marketing");
$no_of_element = count($ITdept);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($ITdept);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
}
echo "<br>";
$num_of_element = count($Salesdept);
for ($i=0; $i< $num_of_element; $i++)
{
    $rec = each($Salesdept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
$AdminDept = array_merge($ITdept, $Salesdept);
$num1_of_element = count($AdminDept);
for ($i=0; $i< $num1_of_element; $i++)
{
   $rec = each($AdminDept);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
}
echo "<br>";
$AdminDept = array_merge($Salesdept, $ITdept);
$num2_of_element = count($AdminDept);
```

Session 16


```
for ($i=0; $i< $num2_of_element; $i++)
{
    $rec = each($AdminDept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
?>
```

Figure 16.4 displays the output of the script.

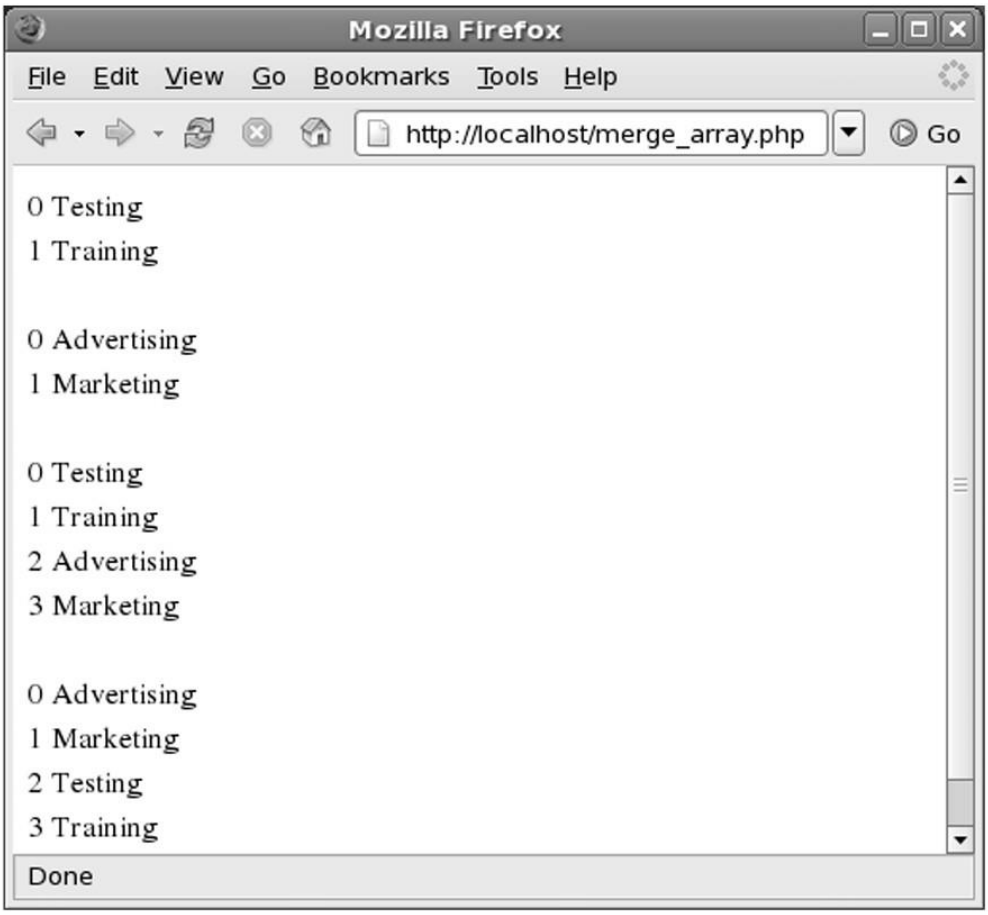


Figure 16.4: Merging Arrays

In the code, the array **AdminDept**, will include values, such as **Testing**, **Training**, **Advertising**, and **Marketing**. The element of the array that is mentioned first in the `array_merge()` function gets the first index number. By default, PHP allots zero as the index number to the first array element. The first element value of the next array, **Salesdept**, is allotted an index number after the first array mentioned in the function has been allotted an index number.

16.5 Multi-dimensional Arrays

A multi-dimensional array contains one array stored within another. A single-dimensional array, has only one level of key value pairs for each element. The element in a single-dimensional array requires an array name and an index. In a multi-dimensional array, each element is an array. Each element in a multi-dimensional array requires an array name and multiple set of indices. For example, in case of a two-dimensional array, each element requires array name and two set of indices instead of one.

The syntax for creating a multi-dimensional array is as follows:

Syntax:

```
$array_name = array(array(key => value), array(key => value));
```

where,

`$array_name` - specifies the name of the multi-dimensional array

`key` - specifies the index number of the array element

`value` - specifies the value of the array element

Note: In the syntax, there is one `array()` function within another.

Code Snippet 5 illustrates the use of multi-dimensional arrays where an array named **country_mdlist** contains another array that includes information, such as **capital** and **currency** of each country, in a script named **multi-dimensional.php**.

Concepts

Code Snippet 5:

```
<?php
$country_mdlist = array(
        "USA" => array(
              "Capital" => "Washington D.C.",
              "Currency" => "US Dollar"),
        "England" => array(
              "Capital" => "London",
              "Currency" => "Pound Sterling"),
        "Australia" => array(
              "Capital" => "Canberra",
              "Currency" => "Australian Dollar"),
        "New Zealand" => array(
              "Capital" => "Wellington",
              "Currency" => "NZ Dollar"));
        echo $country_mdlist["Australia"]["Currency"];
?>
```

Figure 16.5 displays the output of the script.

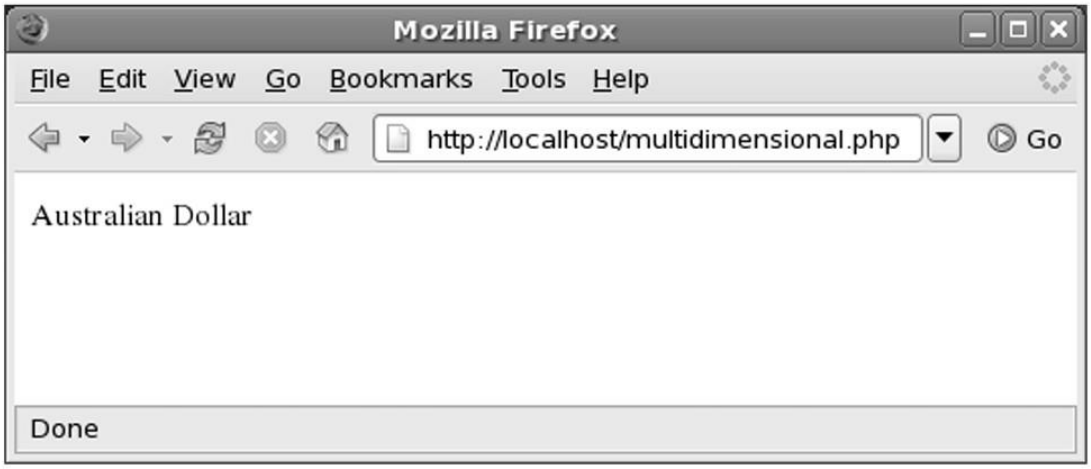


Figure 16.5: Using Multi-dimensional Arrays

In the code, **`country_mdlist`** is a multi-dimensional associative array. It contains key indices such as **`USA`**, **`England`**, **`Australia`**, and **`New Zealand`**. Each array element of a multi-dimensional array includes another array within it containing key indices such as **`Capital`** and **`Currency`**.

Session 16

Working with Arrays

You can also create a multi-dimensional indexed array or a combination of associative and indexed arrays. For example, to create an array that stores commission details of all the employees of an organization earned during the last six months, use a two-dimensional array. The first level of the array will store the names of the employees and the second level of the array will store the commission of the employee earned during the last six months.

To create the array that stores the employee commission details, consider the code as shown in Code Snippet 6, in a script named **emp_comm.php**.

Code Snippet 6:

```
<?php
$employee_det = array(
       "Employee 1" => array(
             1 => "$100",
             2 => "$150",
             3 => "$100",
             4 => "$160",
             5 => "$250",
             6 => "$148"),
       "Employee 2" => array(
             1 => "$180",
             2 => "$195",
             3 => "$200",
             4 => "$130",
             5 => "$280",
             6 => "$218"));
echo "The commission is: ";
echo $employee_det["Employee 2"][5];
?>
```

Concepts

Figure 16.6 displays the output of the script.

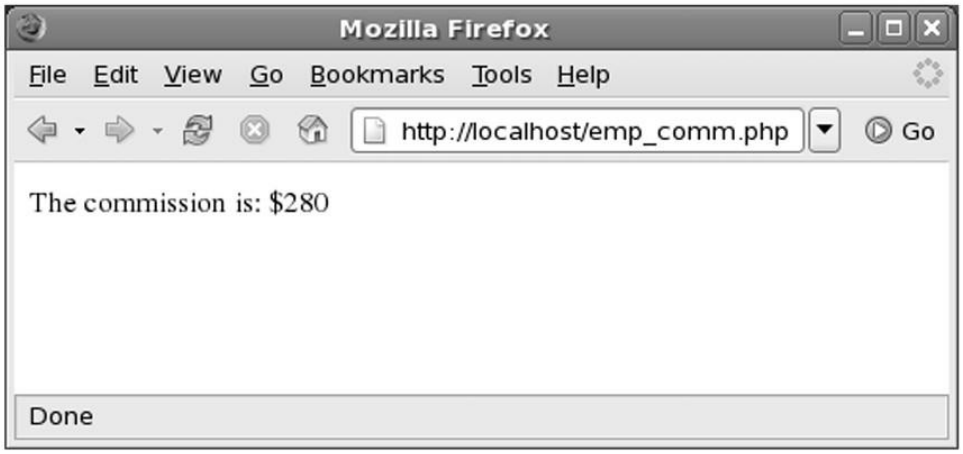


Figure 16.6: Using Multi-dimensional Indexed Arrays

In the code, both associative and indexed indices are used to create a multi-dimensional array.

16.6 Array-Related Functions

PHP provides different functions to manipulate arrays. You can perform different tasks using these functions, such as change the order of the element values, or the key indices and swap the element values between key indices.

16.6.1 The sort() Function

PHP enables to sort an array based on the element value. The `sort()` function arranges the element values in alphabetical order.

The syntax for the `sort()` function is as follows:

Syntax:

```
sort(ArrayName)
```

where,

`sort` - arranges the element values in alphabetical order

`ArrayName` - specifies the name of the array whose elements are to be sorted

Session 16

Working with Arrays

Code Snippet 7 illustrates the sorting of the department array, in a script named **sort_array.php**.

Code Snippet 7:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
sort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($department);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
}
?>
```

Figure 16.7 displays the output of the script.

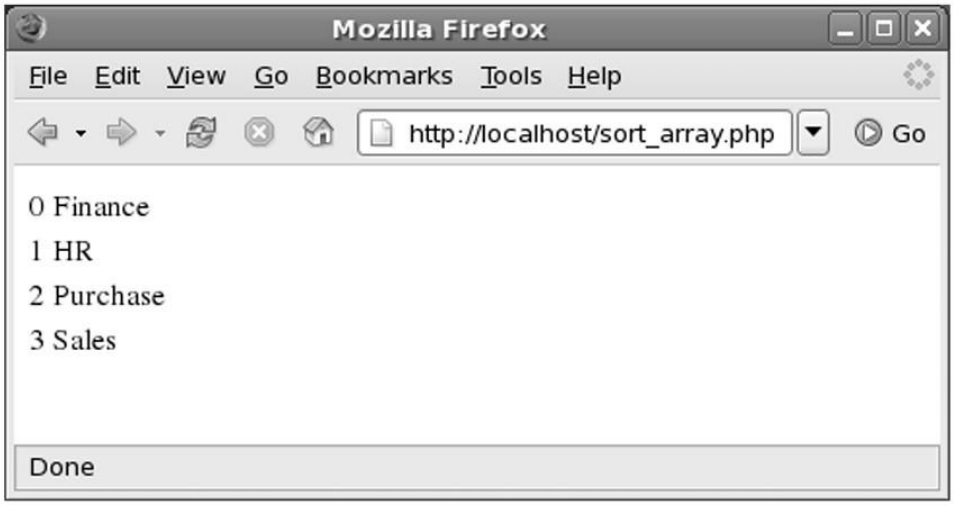


Figure 16.7: Sorting Elements of an Array

Figure 16.7 displays the elements of the array in alphabetical order. The order of the element values have changed. However, the order of the index values is constant.

Concepts

16.6.2 The rsort( ) Function

The `rsort()` function is similar to the `sort()` function. The only difference between the two is that instead of sorting in ascending alphabetical order, the `rsort()` function sorts the element values in descending alphabetical order.

The syntax for the `rsort()` function is as follows:

Syntax:

```
rsort(ArrayName)
```

where,

`rsort` - arranges the element values in descending alphabetical order

`ArrayName` - specifies the name of the array whose elements are to be sorted

Code Snippet 8 illustrates the use of the `rsort()` function to display the values of the department array in the descending alphabetical order, in a script named **`rsort_array.php`**.

Code Snippet 8:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
rsort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
     $rec = each($department);
     echo "$rec[key] $rec[value] ";
     echo "<br>";
}
?>
```

Figure 16.8 displays the output of the script.

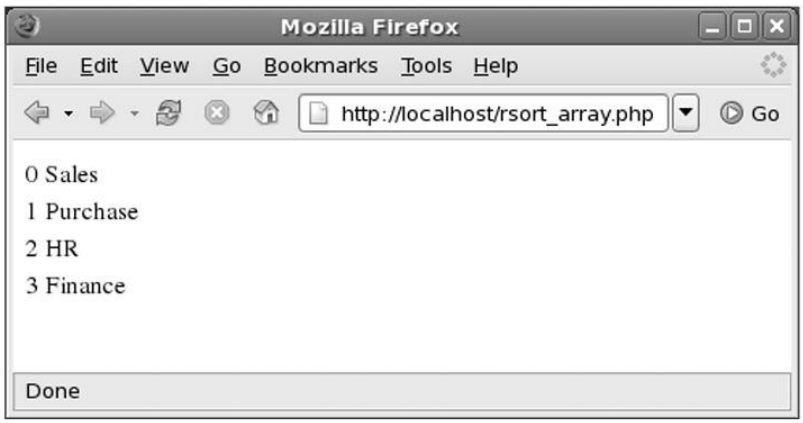


Figure 16.8: Sorting Array Elements in Descending Alphabetical Order

Figure 16.8 displays the elements of the array in descending alphabetical order. The order of the element values have changed. However, the order of the index values is constant.

16.6.3 The arsort() Function

The `arsort()` function is similar to `rsort()` function. The only difference between `rsort()` and `arsort()` function is that the `arsort()` function can sort both associative and indexed arrays.

The syntax for the `arsort()` function is as follows:

Syntax:

```
arsort(ArrayName)
```

Code Snippet 9 illustrates the use of the `arsort()` function on the array, **`department`**, in a script named **`arsort_array.php`**.

Code Snippet 9:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
arsort($department);
```

Concepts

```
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Figure 16.9 displays the output of the script.

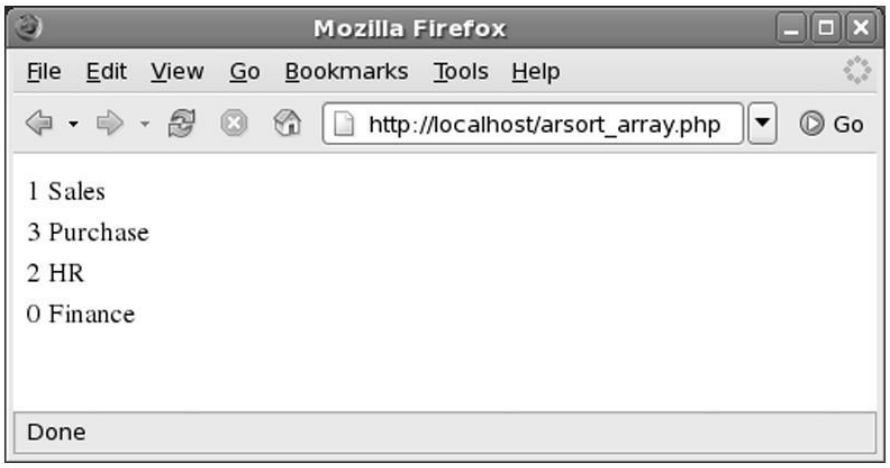


Figure 16.9: Sorting Associative and Indexed Arrays

16.6.4 Other Array-related Functions

Apart from the previously mentioned functions that enable sorting of arrays, PHP provides different functions to manipulate arrays. Table 16.1 lists the functions to manipulate arrays.

Function Name	Description
`count()`	Returns the number of elements in an array
`sizeof()`	Returns the number of elements in an array. You can use this function instead of `count()`
`array_count_values()`	Maintains a count of the occurrences of same element values in an array. It returns the number of occurrences of same element values
`array_flip()`	Converts the element values to index values and vice versa

Session 16

Function Name	Description
array_intersect()	Identifies and returns the common element value among a group of arrays
array_keys()	Displays all the key indices of the specified array
array_reverse()	Reverses the order of the array elements
array_shift()	Returns and removes the first element of an array
array_key_exists()	Identifies whether or not a given key or index exists in an array
array_push()	Adds one or more elements to the end of an array
array_pop()	Pops and returns the last value of an array

Table 16.1: Functions to Manipulate Arrays

The following examples illustrate the use of other array functions:

- To flip the element values to the index values and the index values to the element values of the department array, enter the code as shown in Code Snippet 10, in a script named **array _ flip.php**.

Code Snippet 10:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_flip($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($dept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Concepts

Figure 16.10 displays the output of the script.

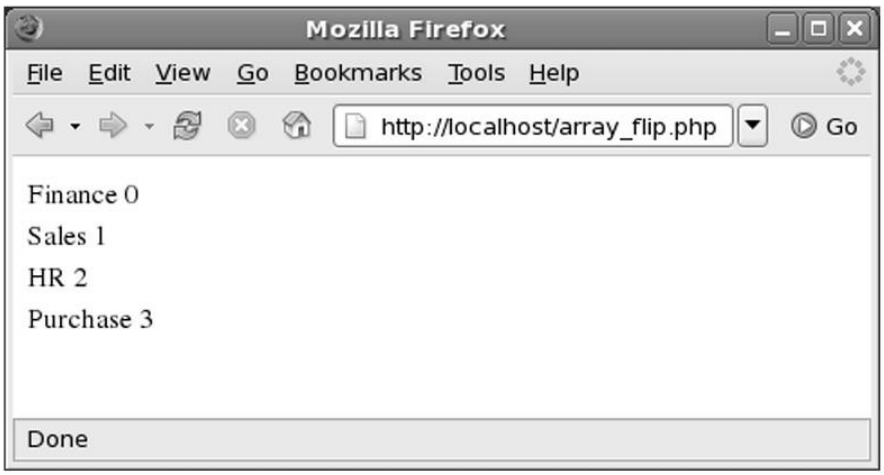


Figure 16.10: Using array_flip Function

- To reverse the order of elements of the department array, enter the code as shown in Code Snippet 11, in a script named **`array _ rev.php`**.

Code Snippet 11:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_reverse($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($dept);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
}
?>
```

Concepts

Figure 16.11 displays the output of the script.

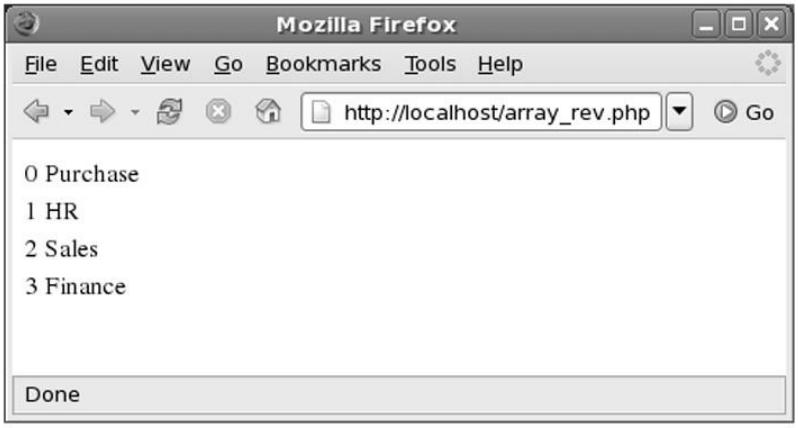


Figure 16.11: Using array_reverse Function

➢ To view all the key values of the department array, enter the code as shown in Code Snippet 12, in a script named **`array _ keys.php`**.

Code Snippet 12:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_keys($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($dept);
    echo "$rec[value] ";
    echo "<br>";
}
?>
```

Concepts

Figure 16.12 displays the output of the script.

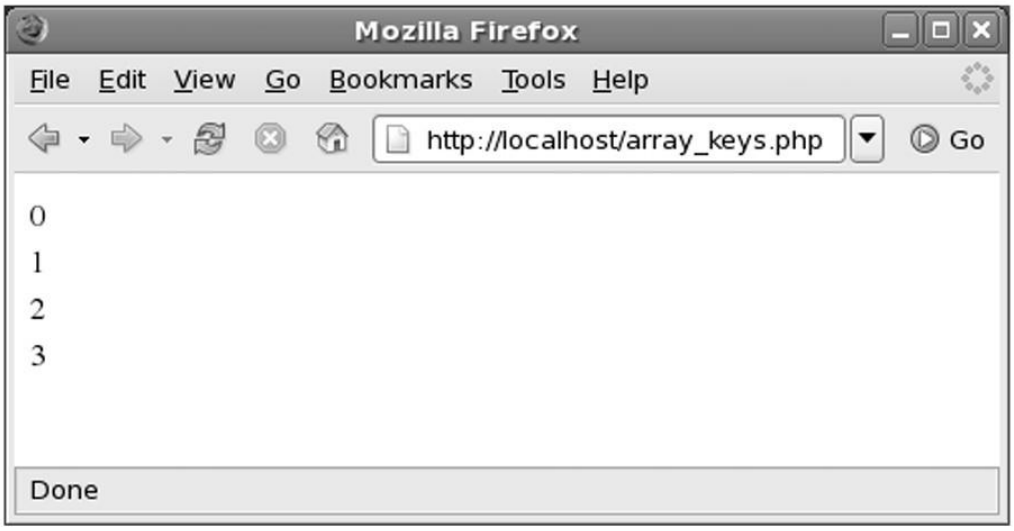


Figure 16.12: Using array_keys Function

➢ To remove an element value from the department array, enter the code as shown in Code Snippet 13, in a script named **`array _ pop.php`**.

Code Snippet 13:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
array_pop($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($department);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
}
?>
```

Figure 16.13 displays the output of the script.

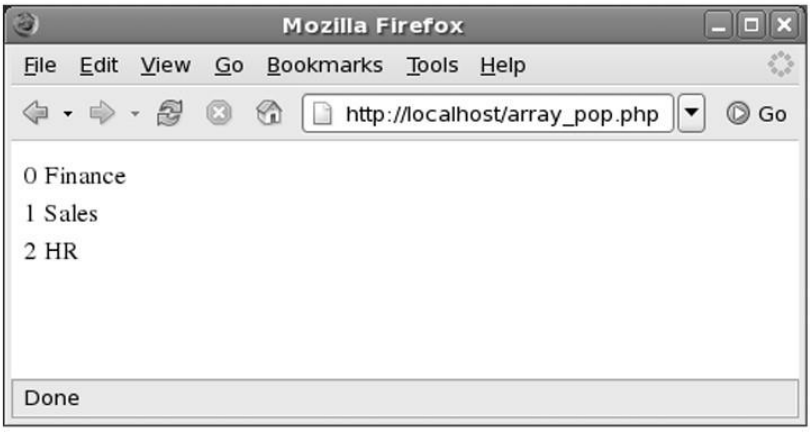


Figure 16.13: Using array_pop Function

➢ To add an element value to the department array, enter the code as shown in Code Snippet 14, in a script named **`array _ push.php`**.

Code Snippet 14:

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
array_push($department, "Marketing");
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
     $rec = each($department);
     echo "$rec[key] $rec[value] ";
     echo "<br>";
}
?>
```

Concepts

Figure 16.14 displays the output of the script.

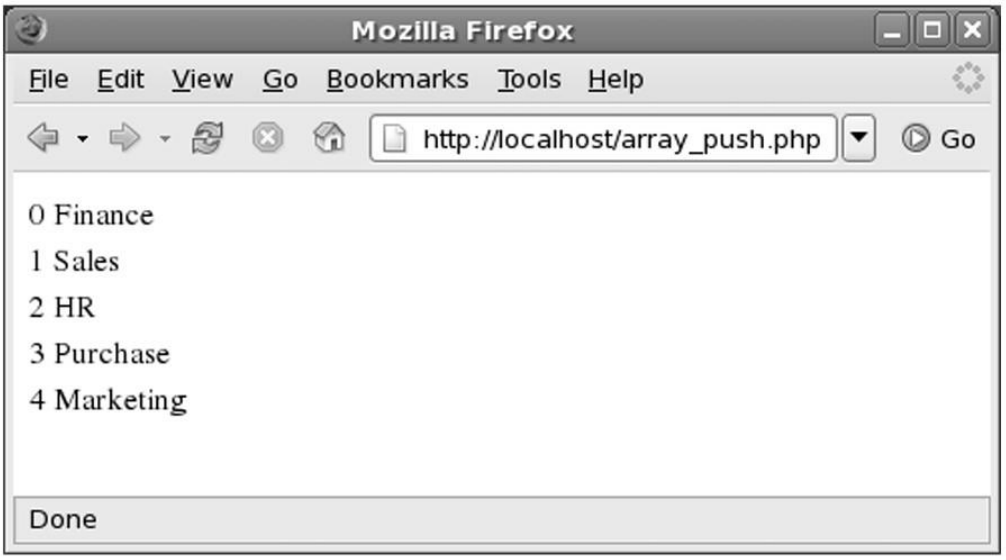


Figure 16.14: Using array_push Function

Session 16

Summary

- An array is a variable that can store a set of values of the same data type.
- All the elements in an array are referenced by a common name.
- An array index is used to access an element.
- Merging arrays is the process of combining element values of two or more arrays.
- In a single-dimensional array, the element includes only one level of key value pairs. In a multi-dimensional array, each element is an array. Each element requires an array name and multiple set of indices.
- An indexed array is an array where the index type is integer and an associative array is an array where the index type is string.
- The sort() function arranges the element values in alphabetical order, the rsort() function sorts the element values in descending alphabetical order, and the arsort() function sorts both associative and indexed arrays.

Concepts

Check Your Progress

1. Which of the following is used to initialize the value of a specific element in an array?

   a. array() function

   b. array identifier

   c. count() function

   d. sort() function

2. Which of the following would you use to assign values for all the elements in an array at the same time?

   a. array() function

   b. array identifier

   c. count() function

   d. sort() function

3. Which of the following functions combine the element values of two arrays?

   a. array_merge()

   b. merge_array()

   c. array_combine()

   d. merge()

4. The _____ function arranges the element values into an alphabetical ascending order.

   a. arsort()

   b. ksort()

   c. sort()

   d. asort()

Check Your Progress

5. _____ function keeps a count of the occurrences of the same element values in an array.

   a. array_count()

   b. count()

   c. count_values()

   d. array_count_values()

6. Which function identifies the common element value among a specified group of array?

   a. array_intersect()

   b. array_merge()

   c. array_group()

   d. array_count_values()

7. _____ function returns all the key values of an array.

   a. array_intersect_keys()

   b. array_keys()

   c. array_key_exist()

   d. array_count_values()

Session 16

Answers

1. b
2. a
3. a
4. c
5. d
6. a
7. b

Onlinevarsity
LEARN BETTER

Session 17

Working with Arrays (Lab)

Objectives

At the end of this session, the student will be able to:

- *Create and use arrays.*
- *Merge arrays.*
- *Use single and multi-dimensional arrays.*
- *Use array-related functions.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Using the array() Function

You can create an array and set the initial values of array elements using the `array()` function. The `array()` function uses key value pairs separated by a comma to create an array. The number of key value pairs in the `array()` function determines the number of elements in an array. Using `array()` function, you can create both indexed and associative arrays.

Indexed array stores the values with reference to an unsigned integer value. The first index is always the number 0, and the index increments by one.

The syntax for indexed array is as follows:

Syntax:

```
$array[0] = "value";
```

Associative array have a key which is associated with a value. Hence, the key can be any data that you want to store, for example, employee's name and their salary.

Session 17

Working with Arrays (Lab)

The syntax for associative array is as follows:

Syntax:

```
$array["key"] = value;
```

To create an associative array that contains names of various programming languages such as, VB, Java, Perl, PHP, VC++, .NET, and Delphi, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **To create an array named $lang, enter the following code:**

```
<?php
$lang = array(
    1 => " VB",
    2 => " Java",
    3 => " Perl",
    4 => " PHP",
    5 => " VC++",
    6 => " .NET",
    7 => " Delphi");
$elmts = count ($lang);
for ($item = 0; $item < $elmts; $item++)
 {
    $row = each ($lang);
    echo "$row[key]  $row[value] ";
    echo "<br>";
 }
?>
```

The `each()` function retrieves each key value pair of an array and stores the result in the `$row` variable. The `for` statement will continue to retrieve values of the array until it reaches the last key value pair.

3. **Save the file as lang.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/lang.php in the Address bar and press Enter.**

Figure 17.1 displays the output of the script.

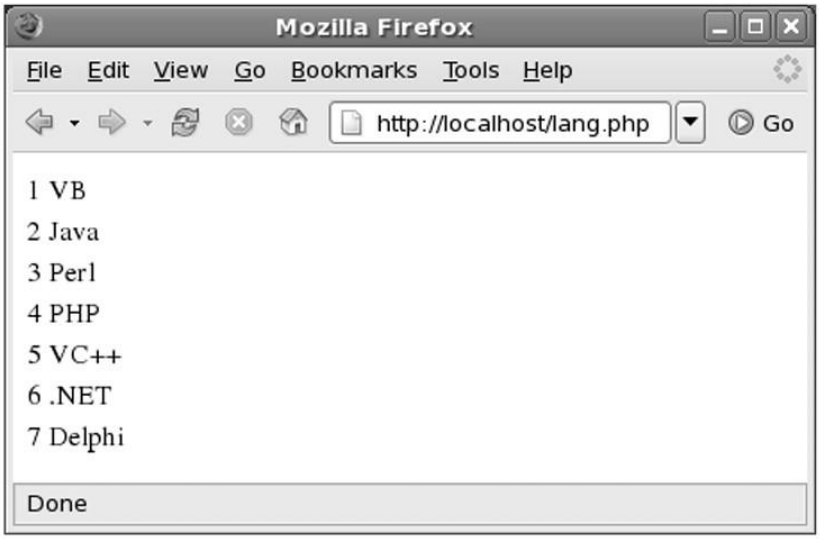


Figure 17.1: Viewing Contents of an Array

Using the Array Identifier

You can initialize value of a specific element in an array using the array identifier. The array identifier also helps to create both associative and indexed arrays.

To use the array identifier in a PHP script, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **To create the array named $lang using the array identifier, enter the following code:**

```
<?php
$lang[]= "VB";
$lang[]= "Java";
$lang[]= "Perl";
$lang[]= "PHP";
$lang[]= "VC++";
$lang[]= ".NET";
$lang[]= "Delphi";
```

Session 17

Working with Arrays (Lab)


```
/* The following code calculates the total number of elements in the
array and stores it in a variable: */

$elmts = count($lang);

/* The following PHP script displays all the values of the
lang() array along with their respective index values: */

for($item=0; $item < $elmts; $item++)
 {
     $row = each($lang);
     echo "$row[key]  $row[value] ";
     echo "<br>";
 }
?>
```

Note: The index value starts from 0. PHP automatically assigns the index value from 0 when you do not specify the index value.

3. **Save the file as mylang.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/mylang.php in the Address bar and press Enter.**

Figure 17.2 displays the output of the script.

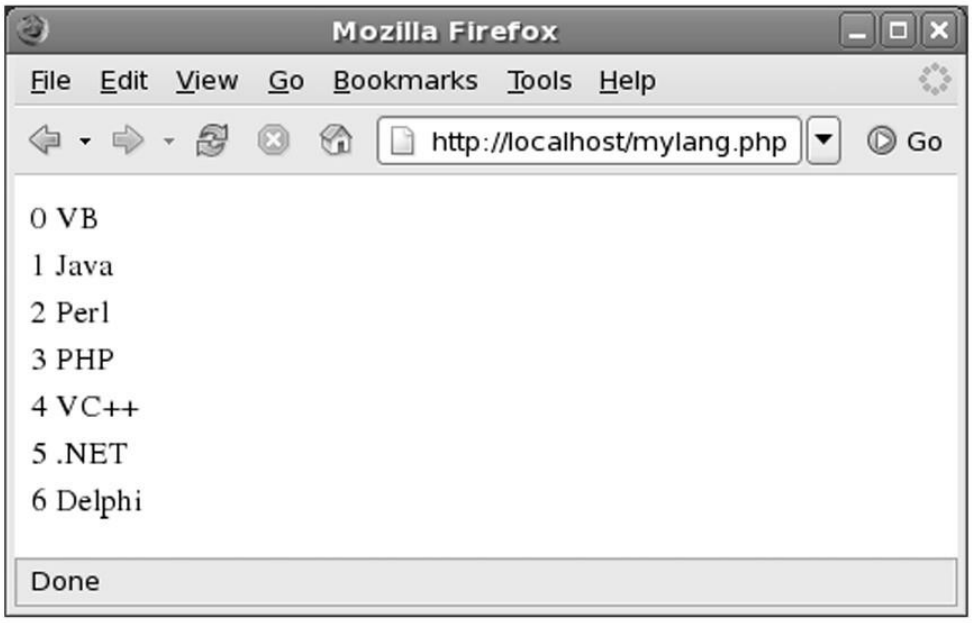


Figure 17.2: Viewing Array Contents

Merging Arrays

PHP enables to merge the element values of two or more arrays. You can merge indexed arrays and associative arrays.

Consider an example of two arrays, **`$serverOS`** and **`$applicationOS`**, that contain names of the operating systems. To merge these arrays, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
$serverOS=array(0 => "Windows NT", 1 => "Windows 2000", "Windows
                                                          2003");
$applicationOS=array(1 => "Windows 95", 2 => "Windows 98", 3 =>
                                                   "Windows ME");
/* To display the contents of the $serverOS array, enter the
following code after creating the two arrays */
```

```
$server_count = count($serverOS);
echo "Server Operating systems";
echo "<br>";
for($i = 0; $i < $server_count; $i++)
{
    $rec = each ($serverOS);
    echo "$rec[key]  $rec[value] ";
    echo "<br>";
}
/* To display the content of the $applicationOS array, enter the
following code: */
echo "<br>";
echo "Application Operating systems";
echo "<br>";
$app_count = count($applicationOS);
for ($i = 0; $i < $app_count; $i++)
{
    $rec = each ($applicationOS);
    echo "$rec[key]  $rec[value] ";
    echo "<br>";
}
/* To combine the element values of these two arrays into one, enter
the following code: */

$OptSys = array_merge($serverOS, $applicationOS);

// To display the merged array, enter the following code:
echo "<br>";
echo "Merged array of Operating systems";
echo "<br>";
$total = count($OptSys);
for ($i = 0; $i < $total; $i++)
 {
    $rec = each ($OptSys);
    echo "$rec[key]  $rec[value] ";
    echo "<br>";
}
?>
```

Session 17

Working with Arrays (Lab)

The `each()` function retrieves each key value pair of the `$serverOS` array and stores the result in the `$rec` variable. The `for` statement will continue to retrieve values of the array until it reaches the last key value pair.

3. Save the file as optSyst.php under the /usr/local/apache2/htdocs directory.

4. Open the Mozilla Firefox Web browser.

5. Enter http://localhost/optSyst.php in the Address bar and press Enter.

Figure 17.3 displays the output of the script.

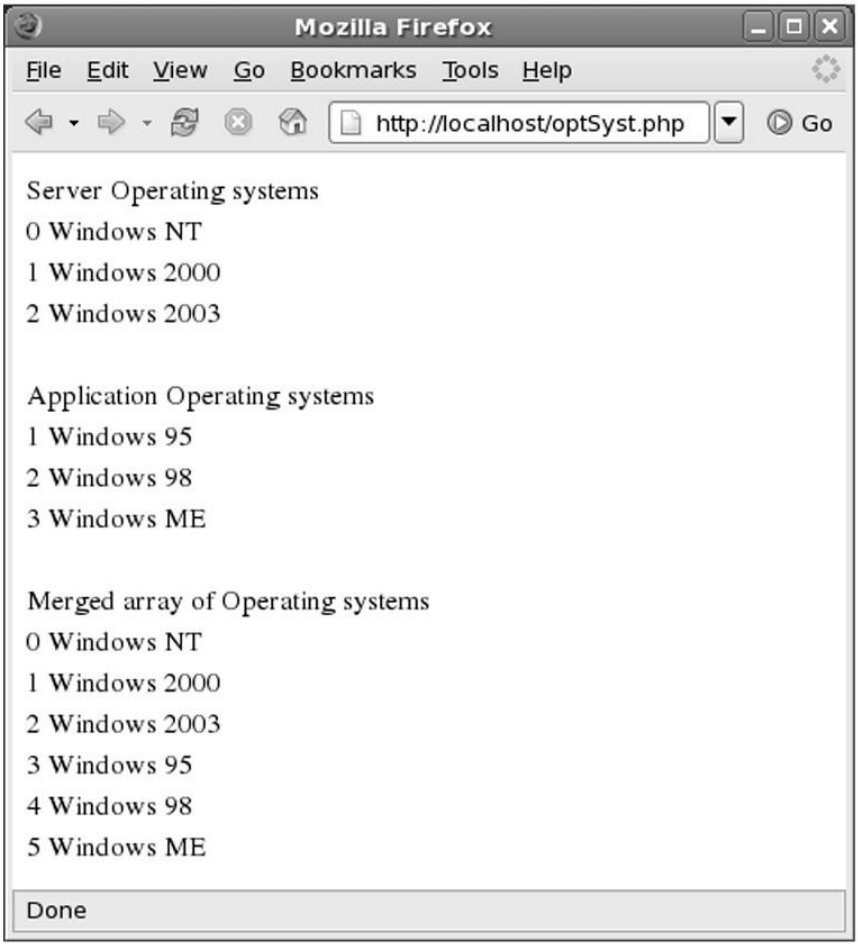


Figure 17.3: Merging Arrays

Session 17

Using Multi-dimensional Array

In a multi-dimensional array, one array is stored within another. Each element in the multi-dimensional array is an array. Each element is referenced by an array name and multiple sets of indices. For example, in case of a two-dimensional array, each element requires array name and two set of indices instead of one.

For example, to create and display a two-dimensional array to store the information of student's name, address, email addresses, and grade, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
$student = array(
    001 => array(
    "Name" => "Chris Edwards",
    "Email" => "chris.e@flaymore.edu",
    "Grade" => "A",
    "Gender" => "Male"),
     002 => array(
     "Name" => "Martina Lake",
     "Email" => "martina.l@flaymore.edu",
     "Grade" => "A+",
     "Gender" => "Female"),
     003 => array(
     "Name" => "Luce Grace",
     "Email" => "luce.g@flaymore.edu",
     "Grade" => "B",
     "Gender" => "Female"),
     004 => array(
     "Name" => "Jack Thompson",
     "Email" => "jack.t@flaymore.edu",
     "Grade" => "B+",
     "Gender" => "Male")
);
/* To display the information stored in the $student array, enter
the following code: */
```

Session 17


```
$total_elmt = count($student);

for($i = 0; $i<$total_elmt; $i++)
 {
     $row = each($student);
     $val = $row["key"];
     echo "00$val:";
     echo "<BR>";
     $valcount = count($student[$val]);
     for($t = 0; $t<$valcount; $t++)
     {
         $rec = each($student[$val]);
         echo "$rec[key] : $rec[value]";
         echo "<BR>";
     }
  echo "<BR>";
}
?>
```

Note: The $student array is a multi-dimensional array. 001, 002, 003, and 004 are the index values of the elements that in turn are arrays that includes student information, such as the name, email address, grade, and gender.

3. **Save the file as studentDet.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/studentDet.php in the Address bar and press Enter.**

Lab Guide

Figure 17.4 displays the output of the script.

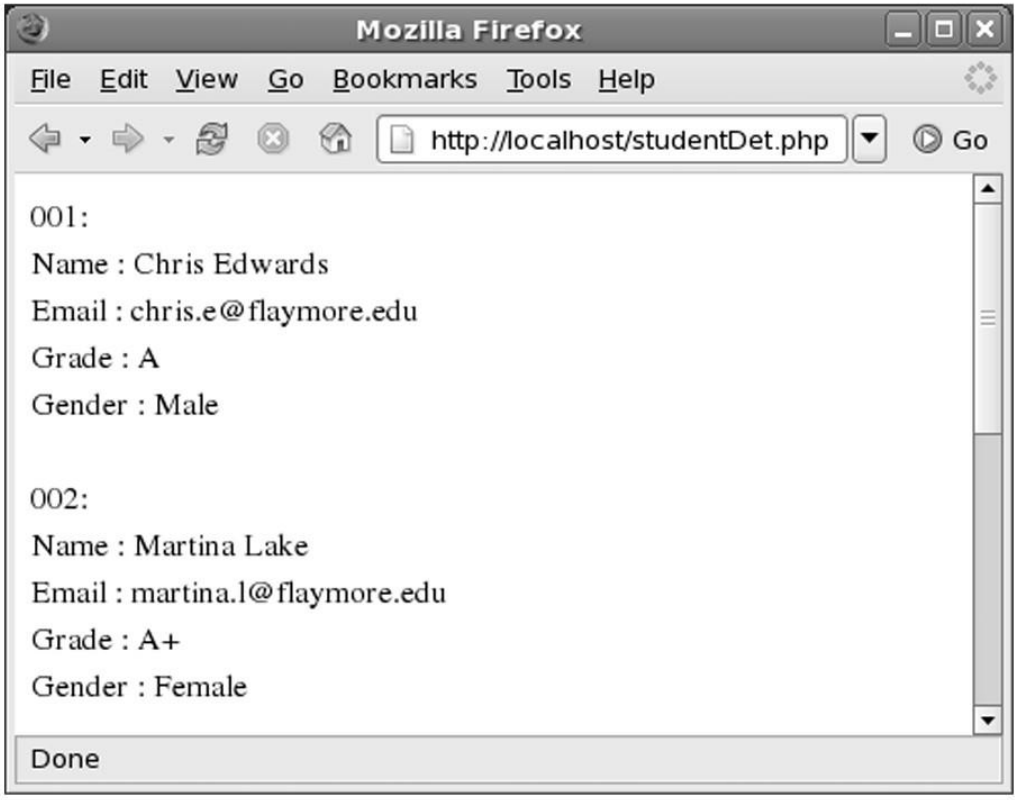


Figure 17.4: Using Multi-dimensional Array

Sorting Arrays

PHP provides different functions, such as `sort()`, `rsort()`, and `arsort()` to sort the element values of an array.

To create an array containing values such as VB, Java, PHP, and Delphi, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<?php
// The $lang arranges the array in the alphabetical order.
$lang = array(1 => "VB", 2 => "Java", 3 => "PHP", 4 => "Delphi");
```

Session 17

Working with Arrays (Lab)


```
/* The following PHP script displays all the elements of the
$lang array after sorting the array */
$total_elmt = count($lang);
echo "Original array:";
echo "<br>";
for($i=0; $i<$total_elmt; $i++)
{
     $rec = each($lang);
     echo "$rec[key] $rec[value]";
     echo "<br>";
}
echo "<br>";
sort($lang);
$total_elmt = count($lang);
echo "Modified array:";
echo "<br>";
for($i=0; $i<$total_elmt; $i++)
{
    $rec = each($lang);
    echo "$rec[key] $rec[value]";
    echo "<br>";
}
?>
```

3. **Save the file as modlang.php under the /usr/local/apache2/htdocs directory.**

4. **Open the Mozilla Firefox Web browser.**

5. **Enter http://localhost/modlang.php in the Address bar and press Enter.**

Lab Guide

Figure 17.5 displays the output of the script.

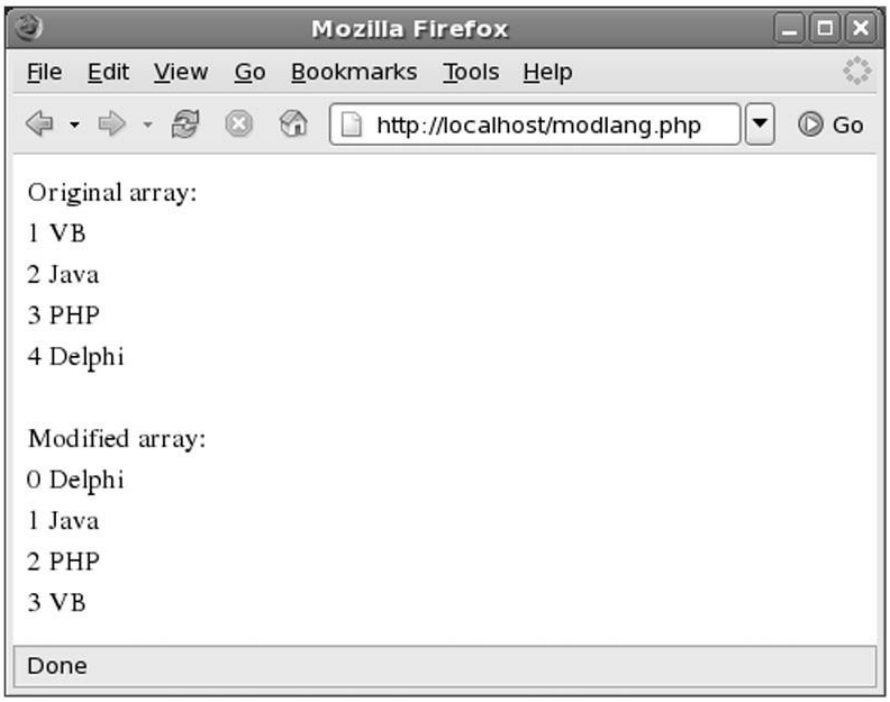


Figure 17.5: Sorting Elements of an Array

6. **Open the modlang.php file in the text editor.**
7. **Replace the code sort($lang) with rsort($lang).**
8. **Save the modlang.php file.**
9. **Open the Mozilla Firefox Web browser.**
10. **Enter http://localhost/modlang.php and press Enter.**

Figure 17.6 displays the output of the script.

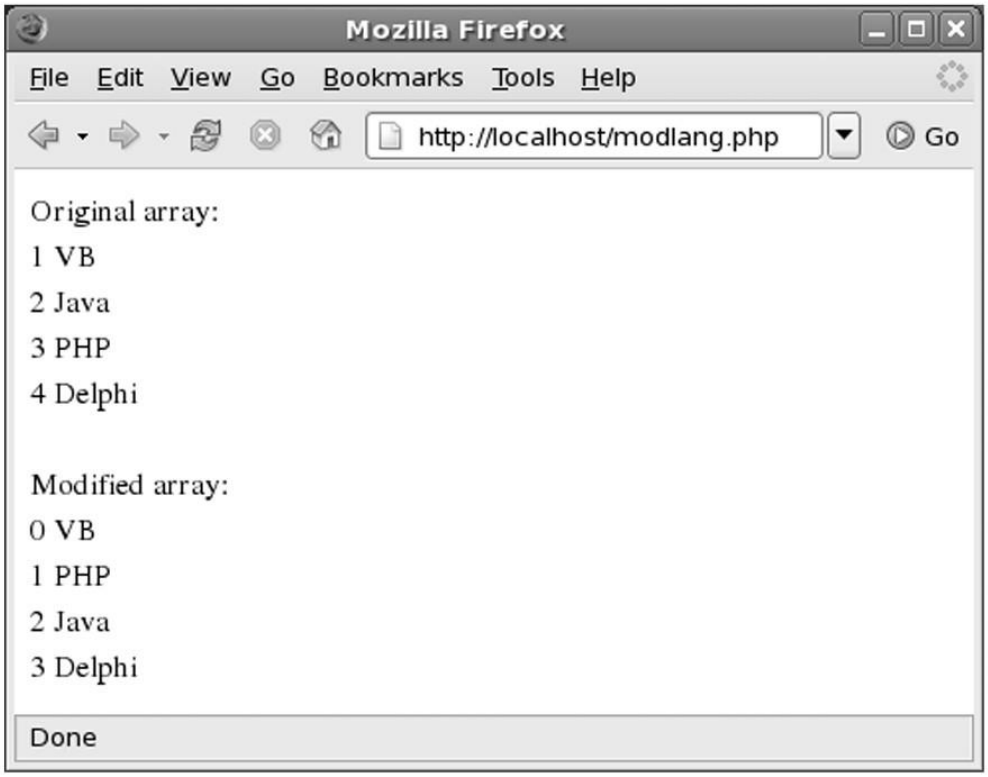


Figure 17.6: Sorting an Array

Session 17

Working with Arrays (Lab)

Do It Yourself

1. Venture Capital Inc. is a trading company that deals in automobile parts. It has branches in USA, Australia, Russia, Japan, Africa, China, and New Zealand.

2. Create an associative array that includes the names of the seven countries where the branches of Venture Capital Inc. are located using array() function. Also, display the contents of the associative array.

3. Create an indexed array that includes the names of the seven countries where the branches of Venture Capital Inc. are located using array identifier. Also, display the contents of the indexed array.

4. Sort the contents of the indexed array in the alphabetical order.

5. Sort the contents of the associative array in the descending alphabetical order.

6. Create and display the contents of a multi-dimensional array that contains the information as shown in table 17.1.

Country	HO	Other Branches
Australia	Canberra	Sydney, Melbourne
New Zealand	Wellington	Christchurch, Hamilton
USA	Washington DC	New Jersey, Miami
Japan	Tokyo	Yokohama, Osaka

Table 17.1: Branch Details

Session 18

Handling Databases with PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Describe Database APIs.*
- *Explain the process of connecting to a database.*
- *Explain the use of data access functions.*
- *Explain SQL queries using PHP.*
- *Explain HTML tables using SQL queries.*

18.1 Introduction

A database is used to store data. A relational database stores data in tables that are linked with common fields, known as keys. A Relational Database Management System (RDBMS) also stores data and the relationships between the data in tables. The different examples of RDBMS are MS Access, Oracle, and MySQL. In the Linux operating system, MySQL is mostly preferred because it is open source software, and is easy to use.

In this session, you will learn how to connect to a database through PHP. You will also learn how to use the data access functions and perform SQL queries using PHP. In addition, you will learn to display the records of the table in HTML tables using SQL queries.

18.2 Database APIs

Database APIs allows developers to write applications that are movable or easily accessible between the database products. Some of the common database APIs are Native-Interface, Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), and Common Object Request Broker Architecture (CORBA). It is a method for accessing data from the database using any application.

PHP supports MySQL database for accessing data from the database.

- **Connecting to a Database**

  A Web site connects to a database to access and store information. A connection is established with the help of a data source name. A data source name is a structure containing information required to connect to the database.

Concepts

The steps for connecting to a database are as follows:

- Open the database connection
- Work with the database
- Close the database connection

18.2.1 Connecting to the MySQL Server

PHP and MySQL are automatically installed while customizing the installation of Linux operating system.

You have to use MySQLi Extension (MySQL Improved) which is a relational database driver to connect to the MySQL server. You can establish a connection using the procedural method `mysqli_connect()` or by using the object method, `new mysqli()`.

You have to establish a connection to the MySQL server and PHP with the help of **`mysql_connect()`** function. This function takes three arguments - the name of the machine on which the database is running, the database username, and the database user password.

The syntax to connect to the MySQL server is as follows:

Syntax:

```
$link_id = new mysqli('host_name','user_name','password');

OR

$link_id =   mysqli_connect('host_name','user_name','password','dbname');
```

where,

`host_name` - specifies the name of the server on which the database is running. The default location of MySQL server is localhost.

`user_name` - specifies the user name.

`password` - specifies the password to connect to the database. This is an optional argument because a user account can be created without a password.

`link_id` - stores the return value of the connection. This variable determines whether the connection is established or not.

Optionally, you may also specify

`db_name`: The name of the database to which connection is being established.

For example, to connect to the MySQL server, consider the code in a script named **`mysql.php`**.

```
<?php
$link_id = mysqli_connect('localhost','root','abc123');
?>
```

In the code, **`root`** is the username, **`localhost`** is the server name, and the password is **`abc123`**.

18.2.2 Working with the Database

Before starting work with the database, you have to establish a connection with the MySQL server. PHP provides the following functions to work with the MySQL database:

- **mysqli_select_db()**: This function defines the default database that will be used for the connection.

  The syntax for `mysqli_select_db()` function is as follows:

  Syntax:

  ```
  mysqli_select_db($link_id, "database_name");
  ```

  where,

  `database_name` - specifies the database name.

  `link_id` - specifies the return value of the connection.

Concepts

For example, to connect to the MySQL database, enter the code as shown in Code Snippet 1 in a script named **mysql_select_db.php**.

Code Snippet 1:

```
<?php
$connect_mysql=mysqli_connect('localhost','root','abc123','customers');
$mysqli_db = mysqli_select_db($connect_mysql, "Current");

if(!$mysqli_db)
{
    die("Connection failed");
}
else
{
    echo "Current Database is selected";
}
?>
```

Figure 18.1 displays the output of the script.

Figure 18.1: Displaying Confirmation of Database Selection

In the code, the Current database is defined as the default database. If the database is present on the server, the message with the `echo` command is displayed. If the connection fails with the server, the message with the `die()` function is displayed. A `die()` function is equivalent to the `exit()` function. This function terminates the current program.

Session 18

The `mysqli_query()` function can be used to determine list of all tables. It executes a MySQL query.

For example, to list all the tables of the MySQL database, enter the code as shown in Code Snippet 2 in a script named **`listables.php`**.

Code Snippet 2:

```
<?php
$dbname = 'mysql';
if (!mysqli_connect('127.0.0.1', 'root', ''))
{
     echo 'Could not connect to mysql';
     exit;
}
    $sql = "SHOW TABLES FROM $dbname";
    $connect_mysql=mysqli_connect('127.0.01','root',' ');
    $result = mysqli_query($connect_mysql, $sql);
    if (!$result)
    $result = mysqli_query($sql);
    echo " The tables from the database are: <br><br>";
    if (!$result)
    {
    echo "DB Error, Unable to list tables<br>";
    echo 'MySQL Error: ' . mysqli_error();
    exit;
    }
    while ($row = mysqli_fetch_row($result))
    {
    echo "Table: {$row[0]}<br>";
    }
    ?>
```

Concepts

Figure 18.2 displays the output of the script.

Figure 18.2: Displaying the List of Tables

In the code, connection is established with the server. All the tables available in the MySQL database are listed and stored in the `$result` variable. If the database contains the tables, they will be displayed. If the table does not exist in the database, the message with the `echo` command is displayed and the execution will stop.

- **mysqli_num_rows()**: This function displays the number of rows present in the specified table.

The syntax for the `mysqli_num_rows()` function is as follows:

Syntax:

```
mysqli_num_rows("query_result");
```

where,

`query_result` - Specifies the query resultset.

Concepts

For example, to list the number of rows from a table in a database, enter the code as shown in Code Snippet 3 in a script named **`list_rows.php`**.

Code Snippet 3:

```
<?php
$connect = mysqli_connect("localhost", "root", "");
if ($result = mysqli_query($connect, "SELECT * FROM Employees"))
{
$rows = mysqli_num_rows($result);
echo "The table contains $rows rows.<br>";
}
?>
```

Figure 18.3 displays the output of the script.

Figure 18.3: Displaying the Number of Rows in a Table

In the code, the numbers of rows from the table are listed. If there are no rows in the table, the message specified with the `echo` command is displayed. If rows are present in the table, the total numbers of rows are displayed.

18.2.3 Closing the Connection

The connection with MySQL server can be closed with the help of the `mysqli_close()` function.

The syntax for the `mysqli_close()` function is as follows:

Syntax:

```
mysqli_close($link_id);
```

Concepts

where,

`link_id` - specifies the return value of the connection.

For example, to close the connection to the MySQL database, enter the code as shown in Code Snippet 4 in a script named **`conn_close.php`**.

Code Snippet 4:

```
<?php
$connect_mysql = mysqli_connect("localhost", "root", "");
$result = mysqli_query($connect_mysql, "SELECT * FROM Employees");
$rows = $result->num_rows;
echo "The table contains $rows rows.<br>";
mysqli_close($connect_mysql);
echo "The connection to the database has been closed.";
?>
```

Figure 18.4 displays the output of the script.

Figure 18.4: Displaying the Termination of Connection

18.3 Data Access Functions

PHP provides the following functions for accessing data from the database:

- **mysqli_query()** - executes MySQL query for retrieving data from tables. This function sends queries to the active database. The MySQL commands such as `SELECT`, `SHOW`, `EXPLAIN`, and `DESCRIBE` can be used with this function.

Session 18

The syntax for the `mysqli_query()` function is as follows:

Syntax:

```
mysqli_query(link_id, query);
```

where,

`link_id` - specifies the return value of the connection.

`query` - specifies the MySQL query.

The MySQL queries must be enclosed within double quotes and must not end with a semicolon (;) symbol.

- **mysqli_fetch_row()** - returns the resultant rows as an array. Each row of the table is placed in an array. These rows are accessed with the index numbers starting from 0.

The syntax for the `mysqli_fetch_row()` function is as follows:

```
mysqli_fetch_row("query_result");
```

For example, to fetch row, enter the code as shown in Code Snippet 5.

Code Snippet 5:

```
$result = mysqli_query($connect_mysql, "SELECT * FROM Employees");
   while ($row = mysqli_fetch_row($result)) {
        printf ("%s (%s)\n", $row[0], $row[1]);
    }
```

- **mysqli_fetch_array()** - retrieves the rows of the table and saves it as an array. It is an extended version of `mysqli _ fetch _ row()` function.

The syntax for the `mysqli_fetch_array()` function is as follows:

Syntax:

```
mysqli_fetch_array("query_result");
```

Concepts

where,

`query_result` - Specifies the query resultset.

- **mysqli_fetch_field()** - displays the details of the column, such as the column name, table name, the maximum length of the column, column constraints, and the column type.

The syntax for the `mysqli_fetch_field()` function is as follows:

Syntax:

```
mysqli_fetch_field("query_result");
```

where,

`query_result` - Specifies the query resultset.

- **mysqli_num_fields()** - displays the number of fields in the specified table.

The syntax for the `mysqli_num_fields()` function is as follows:

Syntax:

```
mysqli_num_fields("query_result");
```

where,

`query_result` - Specifies the query resultset.

18.4 Executing SQL Queries in PHP

Before executing the SQL queries in PHP, you must establish the database connection.

Create a table named `USER_CONTACT` in the USER database with the fields as shown in table 18.1.

Field Name	Data Type	Constraint
USER_ID	INT	NOT NULL PRIMARY KEY
USER_NAME	CHAR(25)	NOT NULL
USER_EMAIL_ID	CHAR(25)	

Table 18.1: USER_CONTACT Table

Session 18

To create a table using the SQL commands in PHP, enter the code as shown in Code Snippet 6 in a PHP script named **mysqltable.php**.

Code Snippet 6:

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysqli_connect($server, $username, $password 'USER');
if($connect_mysql)
{
     echo "Connection established<BR>";
}
else
{
     die("Unable to connect<BR>");
}
$mysql_db = mysqli_select_db($connect_mysql, "USER");
if($mysql_db)
{
     echo "Connected to the database<BR>";
}
else
{
     die("Unable to connect to the database<BR>");
}
$sql_table = "CREATE TABLE USER_CONTACT("."USER_ID INT NOT NULL PRIMARY
KEY,"."USER_NAME CHAR(25) NOT NULL, "."USER_EMAIL_ID CHAR(25)".")";

if(mysqli_query($connect_mysql, $sql_table))
{
     echo "Table is created<BR>";
}
else
{
     die("Unable to create the table<BR>");
}
?>
```

Concepts

Figure 18.5 displays the output of the script.

Figure 18.5: Creating USER_CONTACT Table

In the code, the `USER_CONTACT` table is created in the `USER` database. The table is created using the `CREATE` command in MySQL.

After the table is created, records must be inserted. The records are inserted in the table with the HTML `FORM` method.

To insert records in the `USER_CONTACT` table, enter the code as shown in Code Snippet 7 in a PHP script named **`usercontact.php`**.

Code Snippet 7:

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysqli_connect($server, $username, $password);
if($connect_mysql)
{
     echo "Connection established.";
}
else
{
     die("Unable to connect");
}
$db = "user";
```

```
$mysql_db = mysqli_select_db($connect_mysql, $db);

if($mysql_db)
{
     echo "<BR><BR>Connected to the database.";
}
else
{
     die("Unable to connect to the database");
}
$sql_insert = "INSERT INTO user_contact (user_id, user_name, user_email_id)
VALUES (101,'John','john@mail.com')";
$result = mysqli_query($connect_mysql, $sql_insert);
if($result)
{
     echo "<BR><BR>The records have been added to the table.";
}
else
{
echo "Unable to insert records.";
mysqli_error();
}
?>
```

Figure 18.6 displays the output of the script.

Figure 18.6: Inserting Data in USER_CONTACT Table

Session 18

In the code, records are inserted in the table using the `INSERT` command. If there exists any error, the `printf("Error message: %s\n",mysqli_error($connect_mysql));_error()` function returns the error message that is sent by MySQL server.

The `SELECT` command enables to access data from the tables.

To display the records of the `USER_CONTACT` table from the `USER` database, enter the code as shown in Code Snippet 8 in a PHP script named **`displaytable.php`**.

Code Snippet 8:

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysqli_connect($server, $username, $password);
if($connect_mysql)
{
     echo "Connection established<br>";
}
else
{
     die("Unable to connect<br>");
}
$mysql_db = mysqli_select_db($connect_mysql, "USER");
if($mysql_db)
{
     echo "Connected to the database<br>";
}
else
{
     die("Unable to connect to the database<br>");
}
$sql_disp=("SELECT * FROM USER_CONTACT;");
echo "<br>Displaying Records from the USER_CONTACT table:<br>";
$result = mysqli_query($connect_mysql,  $sql_disp);
while ($row = mysqli_fetch_array($result))
{
echo "$row[USER_ID] ";
     echo "$row[USER_NAME] ";
     echo "$row[USER_EMAIL_ID] <br>";
}
 ?>
```

Session 18

Figure 18.7 displays the output of the script.

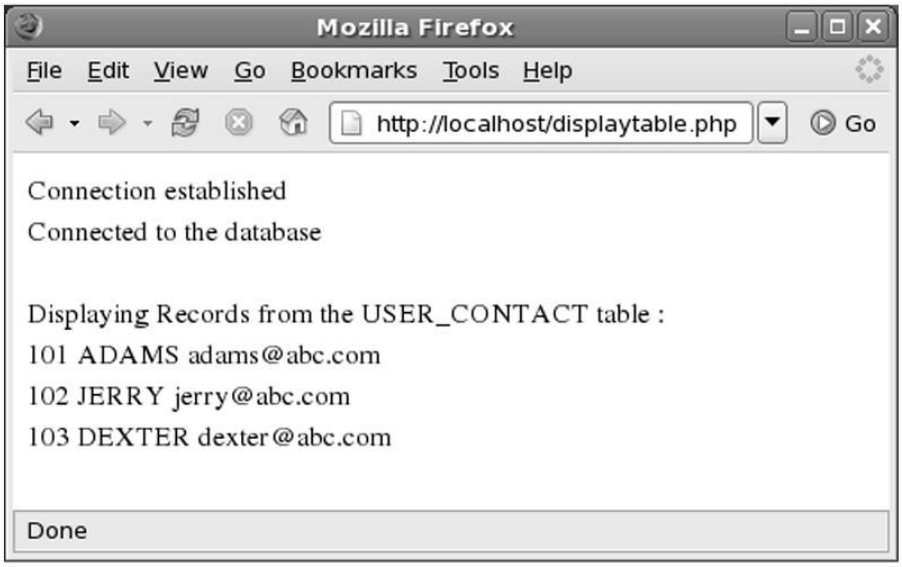


Figure 18.7: Displaying Records of USER_CONTACT Table

In the code, the records such as `USER_ID`, `USER_NAME`, and `USER_EMAIL_ID` from the `USER_CONTACT` table are displayed.

The `DELETE` and `UPDATE` commands enable to modify the contents of the table.

For example, to delete a record from the table, enter the code as shown in Code Snippet 9 in a PHP script named **`delete_record.php`**.

Code Snippet 9:

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysqli_connect($server, $username, $password);
if($connect_mysql)
{
     echo "Connection established<br>";
}
```

Concepts

```
else
{
     die("Unable to connect<br>");
}
$mysql_db = mysqli_select_db($connect_mysql, "USER");

if($mysql_db)
{
     echo "Connected to the database<br>";
}
else
{
     die("Unable to connect to the database<br>");
}
$sql_delete=("DELETE FROM USER_CONTACT WHERE USER_ID = '101'");

$result=mysqli_query($connect_mysql, $sql_delete);
if($result)
{
     echo "Records Deleted: $result<br>";
}
else
{
     echo "RECORDS NOT FOUND IN THE TABLE<br>";
      printf("Error message: %s\n",mysqli_error($connect_mysql));
}
?>
```

Figure 18.8 displays the output of the script.

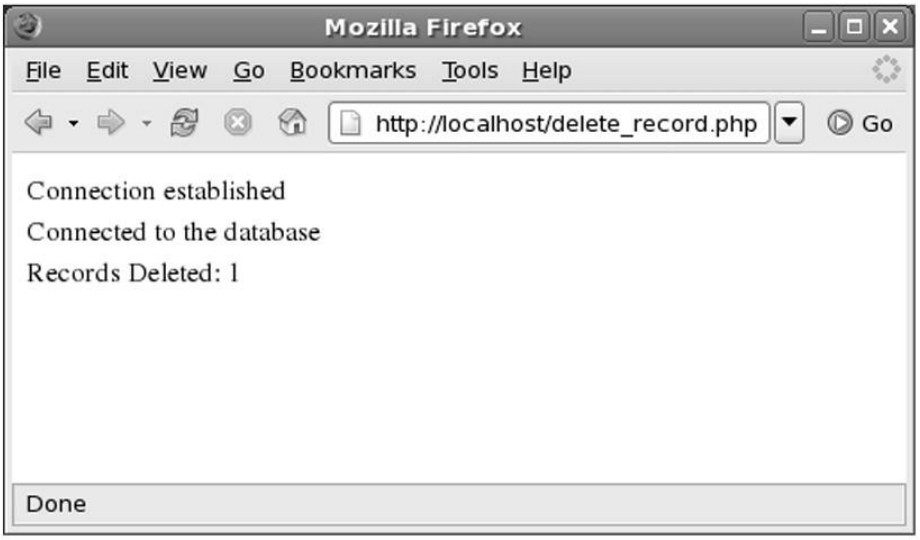


Figure 18.8: Deleting a Record from the USER_CONTACT Table

Note: After deleting the records from the table, check the table contents at the MySQL command prompt using the SELECT command.

To update a record in the table, enter the code as shown in Code Snippet 10 in a PHP script named **`update_record.php`**.

Code Snippet 10:

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysqli_connect($server, $username, $password);
if($connect_mysql)
{
     echo "Connection established<br>";
}
else
{
     die("Unable to connect<br>");
}
```

Concepts

```
$mysql_db = mysqli_select_db($connect_mysql, "USER");
if($mysql_db)
{
     echo "Connected to the database<br>";
}
else
{
     die("Unable to connect to the database<br>");
}
$sql_update=("UPDATE USER_CONTACT SET USER_NAME ='David'
WHERE USER_ID ='102'");

$result=mysqli_query($connect_mysql, $sql_update);
if($result)
{
     echo "RECORDS UPDATED: $result<br>";
}
else
{
     echo "UNABLE TO UPDATE RECORDS<br>";
     printf("Error message: %s\n",mysqli_error($connect_mysql));
}
?>
```

Figure 18.9 displays the output of the script.

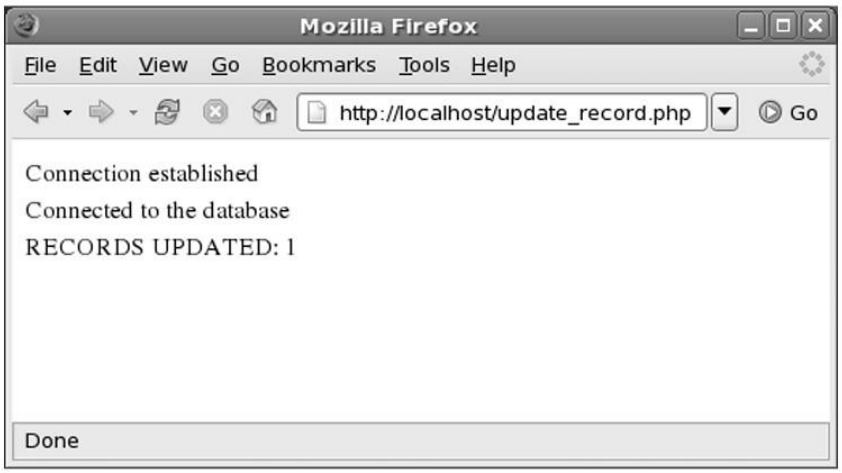


Figure 18.9: Updating Data in a Table

Note: After updating the records in the table, check the table contents at the MySQL command prompt by using the SELECT command.

18.5 Building HTML Tables Using SQL Queries

HTML supports database application components for accessing the database. The contents of the SQL tables can be displayed on the Web browser by building an HTML table structure. The HTML table structure will display the contents of the table along with its field names.

For example, to display all the records of the `user_contact` table using the HTML table structure, enter the code as shown in Code Snippet 11 in a PHP script named **`display_records.php`**.

Code Snippet 11:

```
<HTML>
<BODY>
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysqli_connect($server, $username, $password);
if($connect_mysql)
     echo "Connection established";
$mysql_db = mysqli_select_db($connect_mysql, "USER");

if($mysql_db)
     echo "<BR><BR>Connected to the database<BR><BR>";

echo "<TABLE BORDER BGCOLOR="WHITE">";
echo "<TR><TH>USER_ID<TH><TH>USER_NAME<TH><TH>USER_EMAIL_ID </TH>";
echo "<DBQUERY q> select * FROM USER_CONTACT";
echo  "<DBROW><TR><TD><?  q.USER_ID></TD><TD><?  q.USER_NAME></TD><TD><?
q.USER_EMAIL_ID></TD></TR>";
echo "</DBQUERY>";
echo "</TR>";
echo "</TABLE>";
?>
</BODY>
</HTML>
```

Concepts

Figure 18.10 displays the output of the script.

Figure 18.10: Displaying the Table

The code displays the records of the `user_contact` table on the Web browser in a tabular format. The `DBQUERY` and the `DBROW` are the HTML tags. The `DBQUERY` tag executes records for the SQL query. The `DBROW` tag is used for placing text in the row.

Summary

- Database APIs enable developers to write applications that are movable or easily accessible between the database products.
- The common databases APIs are Native-Interface, ODBC, JDBC, and CORBA.
- PHP is connected to MySQL using three arguments: the MySQL server host name, the MySQL user name, and the MySQL password.
- The connection with the server is established with the help of mysqli_connect() function.
- The basic PHP functions that are used with respect to the database are: mysqli_select_db(), mysqli_list_tables(), and mysqli_num_rows().
- The mysqli_close() function closes the connection with the MySQL server.
- The data access functions used in PHP are: mysqli_query(), mysqli_fetch_array(), mysqli_fetch_row(), mysqli_fetch_field(), mysqli_field_len(), and mysqli_num_fields().

Concepts

Check Your Progress

1. Which of the following function connects the database server to PHP?

   a. mysqli_select_db()

   b. mysqli_connect()

   c. connect_mysqli()

   d. connect()

2. The _____ function displays the total number of rows of the specified table.

   a. mysqli_fetch_array()

   b. mysqli_select_db()

   c. mysqli_list_tables()

   d. mysqli_num_rows()

3. The _____ function displays the details of the columns of the table.

   a. mysqli_fetch_field()

   b. mysqli_fetch_row()

   c. mysqli_fetch_array()

   d. mysqli_num_rows()

4. The MySQL queries are executed by using_____ function.

   a. Mysqli_Query()

   b. mysqli_connect()

   c. mysqli_query()

   d. connect_mysqli()

Check Your Progress

5. Using the _____ command in PHP, records can be inserted in the table.

   a. INSERT

   b. UPDATE

   c. SELECT

   d. DELETE

6. Records can be modified in the table using the _____ command in PHP.

   a. DELETE

   b. INSERT

   c. SELECT

   d. UPDATE

Session 18

Handling Databases with PHP

Concepts

Answers

1. b
2. d
3. a
4. c
5. a
6. d

Session 19

Working with Cookies

Concepts

Objectives

At the end of this session, the student will be able to:

- *Describe the process of setting a cookie.*
- *Explain the process of retrieving a cookie in PHP.*
- *Explain the process to delete a cookie.*
- *Identify the drawbacks associated with cookies.*

19.1 Introduction

Web sites store user information in databases to maintain a track of their visits. However, there are users who do not register with the Web site but frequently visit the Web site. Cookies are used to store information of such visitors. Cookies enable Web sites to store user information.

PHP supports HTTP cookies. In this session, you will learn how to set, retrieve, and delete cookies in PHP. You will also learn about the different security issues related to cookies.

19.2 Introducing a Cookie

HTTP is a stateless protocol, because the execution of the current command is completed without the knowledge of commands that came before it. When a Web browser requests for a static Web page, the server completes the request by sending the required file. This process does not involve any interaction with the user. The user simply clicks the hyperlink on the Web site and accesses the Web page containing the content.

Session 19

Working with Cookies

Concepts

Figure 19.1 displays the transfer of static data from Web server to the Web browser.

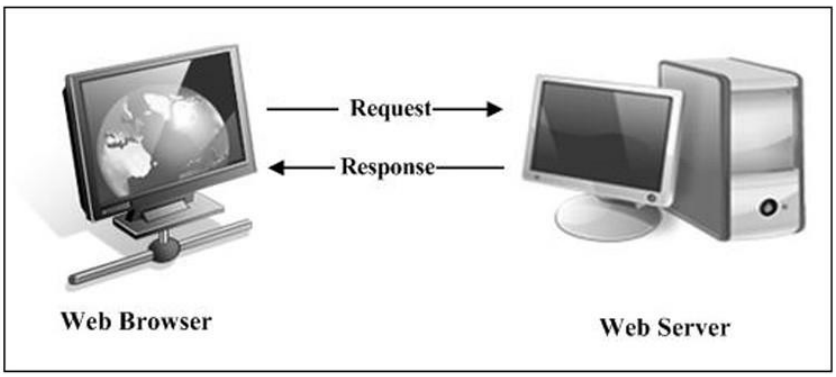


Figure 19.1: Transfer of Static Web Page

For dynamic Web pages that require user interaction, scripting languages such as JavaScript, PHP, and Active Server Pages (ASP) are used. Dynamic Web pages accept information from the user and record it for further processing. For example, in an online shopping Web site, the user navigates through the Web site and the products purchased by the user are added to the shopping cart. On completion of the order, the Web site prompts the user to enter transaction details for order confirmation and delivery. Figure 19.2 displays the transfer of dynamic Web pages.

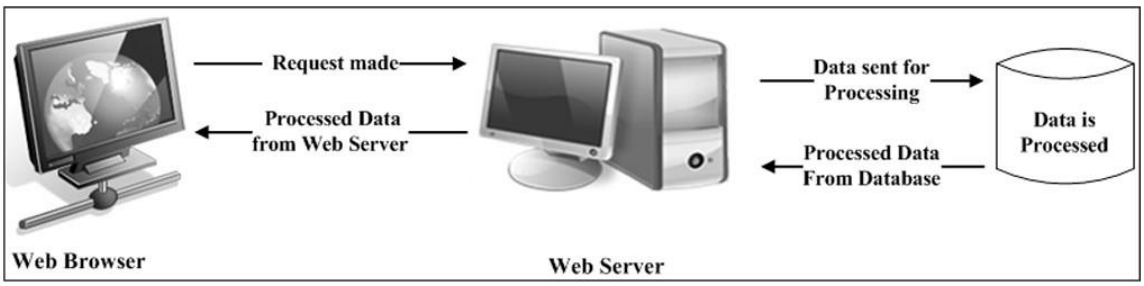


Figure 19.2: Transfer of Dynamic Web Page

Web sites store two types of data, 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. Web sites use two types of cookies and they are as follows:

- **Persistent** - exist in the Web browser for a period specified at the time of its creation
- **Non-persistent** - deleted from the Web browser as soon as the user exits the browser

For example, in an online shopping Web site, a user selects few products and adds them to a shopping cart. The user then navigates to the next page and selects additional products.

Session 19

In such a situation, the user information needs to be stored before the user navigates to the next page. Although the user has not completed the transaction, the Web site stores the data in a temporary variable called cookie. Figure 19.3 displays the temporary data stored in a cookie.

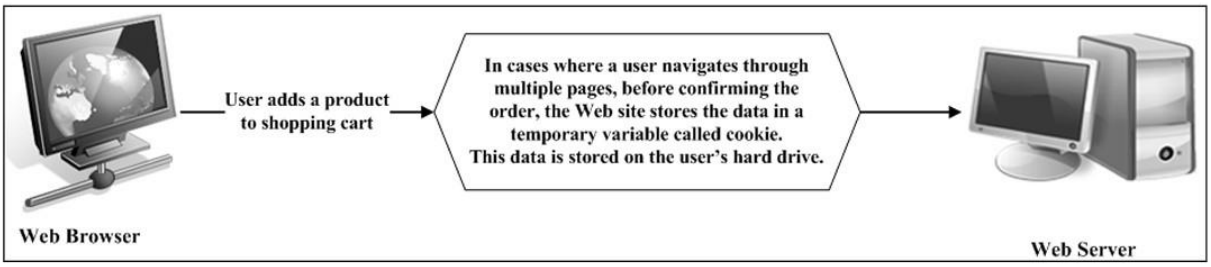


Figure 19.3: Storage of Temporary Data

When the order is cancelled, the products added to the cart are no longer required to the Web server and the Web browser. Such temporary data stored in the cookie is deleted as soon as the order is cancelled or the Web browser is closed. Figure 19.4 displays the deletion of the temporary cookie when the order is cancelled.

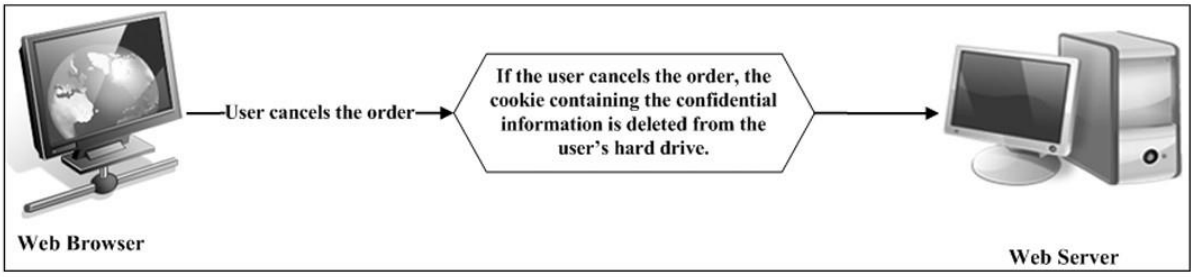


Figure 19.4: Deletion of Temporary Data

Figure 19.4 illustrates an example of order cancellation. When the user places an order, the user enters transaction information, such as the name, address, credit card details on order confirmation. Such confidential information is stored in the hard drive of the client. However, when the user cancels the order, the cookie containing the confidential information is deleted.

Web sites use cookies to determine the following:

- number of times the user has visited the Web site
- number of new visitors
- number of regular users

Session 19

- frequency of a user visiting the Web site
- date on which the user had last visited the Web site
- customized Web page settings for a user

When a user visits the Web site 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 Web site in subsequent requests. The Web server can read the information stored in the cookie only when it is related to the Web site. 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 Web site to keep a track on the user visiting the Web site. The information about the user is generally stored in the name-value pair.

Web servers and Web browsers send cookies to each other in HTTP headers. The Web server sends the cookie to the browser in the `setcookie` 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.

Consider the HTTP response header as shown in Code Snippet 1.

Code Snippet 1:

```
HTTP/2.0 200
Content-Length: 8451
Content-Type: text/html
Date: Mon, 27 Dec 2010 05:29:24 GMT
Expires: Mon, 27 Dec 2010 05:29:44 GMT
setcookie: city=east-coast-usa
```

In the code, the following information is displayed:

- Version number of the HTTP protocol
- Size of the content
- Type of the content
- Date and time of response
- Expiry date and time of the cookie
- Cookie header

The Web browser records the cookie information and saves it to the hard disk of the system.

In subsequent requests made to the Web server for a Web page, the information is sent along with the HTTP request header.

Consider the HTTP request header as shown in Code Snippet 2.

Code Snippet 2:

```
GET /usa/florida.php HTTP/2.0
Connection: Keep-Alive
Cookie: city=east-coast-usa
Host: www.Webworldmaps.com
Referrer: http://www.Webworldmaps.com/
```

Observe the cookie shown in Code Snippet 2. This cookie can be defined using the `setcookie` function. Code Snippet 2 displays a subsequent request that the Web browser sends to the Web server.

19.3 Setting a Cookie

Cookies are incorporated in HTTP request and response headers. Setting a cookie is sending the cookie to the browser. PHP uses two functions, `setcookie()` and `setrawcookie()` to set a cookie. Programmers prefer using the `setcookie()` function because the `setrawcookie()` function sends a cookie without encoding the cookie value.

Note: There are certain special characters that cannot appear in a URL and have to be encoded to retrieve information. Urlencoding is a process where the Web browser takes special characters, such as a tab, space, exclamation mark, hash, and quotes and replaces them with code values.

The `setcookie()` function generates the cookie header field that is sent along with the rest of the header information. The syntax for the `setcookie()` function is as follows:

Syntax:

```
setcookie(name, value, expiry date, path, domain, secure)
```

where,

`name` - defines the name of the cookie. This is a mandatory attribute.

`value` - defines the value of the cookie that is stored on the client system. This is also a mandatory attribute.

`expiry date` - defines the date and time (UNIX timestamp) when the cookie will expire. The cookie is not used once the expiry date is reached. This is an optional attribute.

Session 19

Note: UNIX timestamp signifies the time and date that the `time()` function returns. The time is measured in the number of seconds elapsed from 1st January 1970 00:00:00 GMT.

`path` - defines the location on the server where the cookie will be stored. It specifies the subset of the URLs present in the domain where the cookie is applicable. If the path attribute is not specified in the `setcookie()` function, the path of the document present in the header is taken.

`domain` - defines the domain name where the cookie is made available.

`secure` - defines the type of HTTP connection that the cookies will pass through.

Note: When the value of the secure parameter is set to 1, the cookie will be set only if a secure HTTP connection exists.

When the cookie is set, the value is automatically encoded in the URL. When the script retrieves a cookie, it automatically decodes the value from the URL. PHP executes codes in a specific sequence. HTTP headers are executed before the scripts. Cookies are a part of the HTTP header. There can be more than one cookie in the header but it should relate to the same domain or Web site. The code related to the cookies must be specified before the following:

- HTTP header
- Displaying any content
- Any white space

If any content is displayed before calling the `setcookie()` function, the function will fail and return `False`. So before displaying any output or even the white space, `setcookie()` function must be called. If the `setcookie()` function runs successfully, the function returns `True`.

Note: The `setcookie()` function returning True does not indicate that the Web browser has accepted the cookie.

Consider an example where a Web site displays country maps when a user enters a country name in the search feature of the Web site. To set a cookie that expires in one day, enter the code as shown in Code Snippet 3 in a PHP script.

Code Snippet 3:

```
$mapname = $_GET['fmapname'];
setcookie("mycookie", $mapname, time()+86400, "/Webmap/", ".Webworldmaps.
com");
```

In the code, `fmapname` is the variable that contains the country name that the user enters. The `$mapname` variable stores the value that the `GET` method retrieves from the form. The `setcookie()` function includes the following:

- `mycookie` - defines the name of the cookie
- `time()+86400` - specifies the time when the cookie will expire
- `/Webmap` - defines the location where the cookie will be stored
- `.Webworldmaps.com` - specifies the domain that the cookie will use

To create a cookie that expires when the Web browser window is closed, enter the code as shown in Code Snippet 4 in a PHP script named **`create_cookie.php`**.

Code Snippet 4:

```
$val = $_GET['uname'];
setcookie("uname",$val);
```

In the code, `uname` is the variable that contains a value. The `$val` variable stores the value of `uname` that the `GET` method retrieves. The `setcookie()` function in the code snippet sets a cookie named `uname`. The value of `$val` is assigned to the cookie, `uname`.

PHP script can include more than one `setcookie()` function. Multiple calls to a cookie in the same script can be made in a specific order depending on the version of PHP. In PHP, multiple `setcookie()` function within the same script are called in the specified order.

For example, if a cookie is to be deleted before creating another, the `setcookie` statement must be stated before the `delete` command.

19.4 Retrieving Cookies in PHP

Cookies are useful only when the Web server can retrieve the information from it. The cookie is available only when the next page is reloaded. The Web browser matches the URL against a list of all the cookies present on the client system. If the Web browser finds a match, a line containing the name value pairs of the matched cookie is included in the HTTP header.

Session 19

Working with Cookies

Concepts

The document that created the cookie can access it. All the documents that are present in the same directory can also access the cookie. The documents outside the directory need to include the path or the domain name of the cookie to access the cookie.

PHP provides three ways of retrieving a cookie value and they are as follows:

- Passing a variable as the cookie name - To retrieve the cookie value, use the variable as the cookie name. The following code snippet displays a cookie value:

```
echo $cookie_name;
```

  This method of retrieving the cookie value is not recommended. This is because, PHP will start searching all the variables present in the client system. The `register_globals` option must be enabled in the configuration file to use this method to retrieve cookie value.

- Using `$_COOKIE` array - PHP uses cookie name as a variable to retrieve the cookie value. PHP can also use an associative array called `$_COOKIE` to retrieve the cookie value. The `$_COOKIE` is a global variable that reads a value of the cookie. The name of the cookie must be passed as the key to the `$_COOKIE` array. An example of this is shown as follows:

```
echo $_COOKIE [$cookie_name];
```

  This is more reliable and faster than retrieving the cookie value through a variable.

- Using `$_COOKIE[]` variable - You can also use the following to retrieve the cookie value:

```
$_COOKIE['$cookie_name'];
```

  This method of retrieving the cookie value is recommended and considered the best method. It is also known to be simpler and more secured than using the deprecated `$HTTP_COOKIE_VARS[]` associative array.

  To retrieve a cookie value using the `$_COOKIE` global variable, enter the code as shown in Code Snippet 5 in a script named **`retrieve_cookie.php`**.

  Code Snippet 5:

```
<?php
$cookieval = $_COOKIE ['uname'];
?>
<HTML>
<BODY>
<?php
if (isset($cookieval))
{
```

```
    echo "Welcome $cookieval";
}
else
{
    echo "You need to log in";
}
?>
</BODY>
</HTML>
```

Figure 19.5 shows the output of the script.

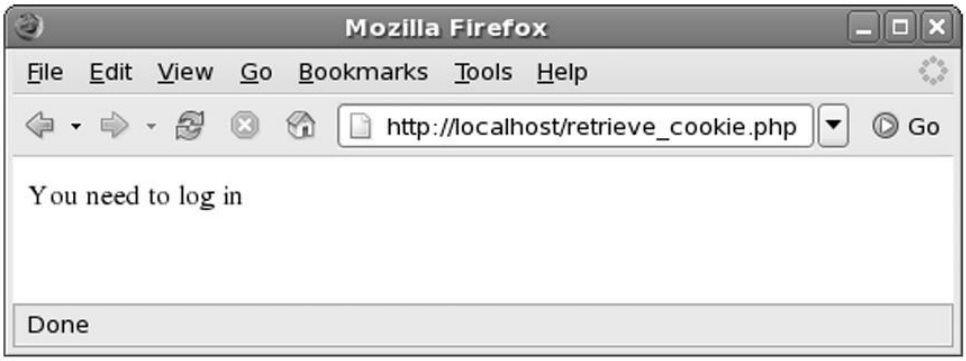


Figure 19.5: Retrieving a Cookie

In the code, `$cookieval` stores the cookie value. The `isset()` function checks whether the cookie is set. If the cookie is set, the `echo` statement will display a welcome message along with the cookie value. If the cookie is not set, a message will appear, prompting the user to log in.

19.5 Deleting a Cookie

Cookies can be deleted automatically or manually. There are two ways to delete a cookie, which are as follows:

- Resetting the expiry time of the cookie to a time in the past
- Resetting a cookie by specifying the name of the cookie

When you create a cookie that has the same name and time as an existing cookie, the existing cookie is deleted from the hard drive of the client.

Concepts

To delete a cookie with a date in the past, enter the code as shown in Code Snippet 6 in a PHP script.

Code Snippet 6:

```
setcookie("$cookie_name", "", time()-8000);
```

In the code, `$cookie_name` refers to the name of the cookie. The value of the cookie is not specified and the `time()` function accepts the expiration date in the past.

Note: Expiry date can be any negative number.

This process is called as deconstructing the variable. Use the following syntax to delete a cookie through deconstruction:

Syntax:

```
setcookie($cookie_name);
```

For example, to delete the cookie named `uname`, use the code as shown in Code Snippet 7.

Code Snippet 7:

```
setcookie($uname);
```

19.6 Problems with Cookies

Web sites store user-related information on the client system. Cookies are not secure and reliable because the user-related information can be accessed by anyone who has full access to the client system. The user-related information can contain sensitive information, such as credit card information, passport number, password, or an identification number.

Following are some of the drawbacks of cookies:

- Cookies cannot contain more than a certain amount of information. The cookies work well when the size limits to 4 KiloBytes (KB). Web sites cannot use cookies to store a large amount of data.
- Only a maximum of 20 cookies of a domain can be maintained.
- A browser can maintain a maximum of 300 cookies. Older cookies that were stored earlier are deleted to accommodate the newer cookies of the different Web site.

Session 19

Working with Cookies

- Storing large number of cookie files slows down the system. To enhance the performance of the system, users often delete temporary files that contain cookies. Web site statistics may go wrong when users delete such cookies to improve the performance.

- Some users disable cookies while accessing Web sites. The Web sites that depend on cookies lose information of such users.

- There can be multiple users using the same system visiting the same Web site. Web sites assign cookies to the system and not to the user. This can hamper the number of visitor's statistics.

- A cookie is created whenever you access a Web page. Therefore, a cookie can contain large amount of information. Retrieving larger amount of information on each page requires repetitive coding across the pages.

Session 19

Working with Cookies

Concepts

Summary

- Web sites use cookies, stored on the hard disk of the client system, to store user-specific information.
- Dynamic Web pages gets information from the user and record it for further processing.
- Persistent cookies are stored in the Web browser for a period specified during the time of its creation and non-persistent cookies are deleted from the Web browser as soon as the user exits the browser.
- The HTTP header, transmitted between the Web server and the Web browser, contains cookies.
- A cookie can be retrieved by passing a variable as a cookie name and using the $_COOKIE[] variable.
- PHP uses the setcookie() and setrawcookie() functions to set a cookie.
- The two ways to delete a cookie are resetting the expiry time of the cookie to a time in the past and resetting the cookie by specifying the name of the cookie.
- The maximum number of cookies that can be maintained for a domain is 20. A browser can maintain a maximum of 300 cookies.

Check Your Progress

1. _____ is a global variable that reads a value of the cookie.

   a. $_COOKIE[]

   b. $HTTP_COOKIE_VARS[]

   c. setcookie()

   d. isset()

2. Which of the following is used to retrieve cookie value?

   a. $_COOKIE[]

   b. $HTTP_COOKIE_VARS[]

   c. setcookie()

   d. isset()

3. A maximum of _____ cookies of a domain can be maintained.

   a. 20

   b. 30

   c. 200

   d. 300

4. Which function does PHP use to remove the session variable from a session?

   a. session_destroy()

   b. gc_maxlifetime()

   c. gc_probability()

   d. session_unregister()

Concepts

Check Your Progress

5. The _____ function checks whether the cookie is set.

   a. $_COOKIE[]

   b. set()

   c. setcookie()

   d. isset()

6. The _____ option must be enabled in the configuration file to pass a variable as cookie name.

   a. register_cookies

   b. enable_register_cookies

   c. register_globals

   d. enable_register_globals

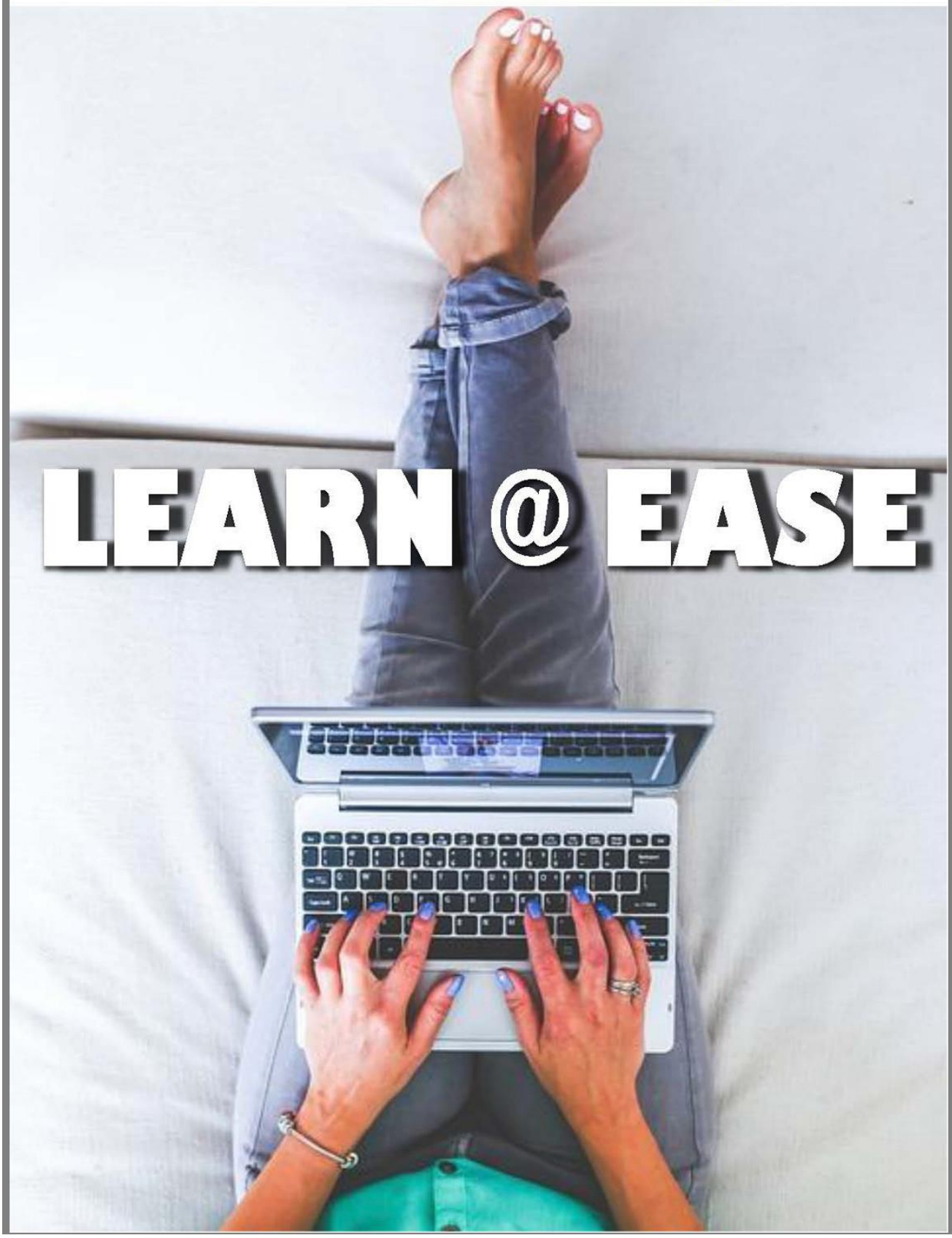
Session 19

Working with Cookies

Answers

1. a
2. a
3. a
4. a
5. d
6. c

Onlinevarsity



Session 20 Working with Cookies (Lab)

Objectives

At the end of this session, the student will be able to:

- *Set a cookie.*
- *Retrieve a cookie.*
- *Delete a cookie.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Setting a Cookie

Cookies are incorporated in HTTP request and response headers. Setting a cookie means sending the cookie to the browser. The setcookie() function generates the cookie header field that is sent along with the rest of the header information.

Suppose a Web site needs a user to enter login name and password before they can shop online. The Web site saves the user information in a cookie. The subsequent Web pages retrieve the user details from the cookie.

To create a form that accepts the login name and password, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code:**

```
<HTML>
<HEAD>
<TITLE> Login Page </TITLE>
</HEAD>
<BODY>
<H4> Please enter your details </H4>
<FORM ACTION = "validate.php" METHOD = "GET" >
<TABLE>
```

Session 20

Working with Cookies (Lab)

Lab Guide

```
<TR>
<TD>Login name</TD>
<TD><INPUT TYPE = "text" NAME = "logname"></TD>
</TR>
<TR>
<TD>Password</TD>
<TD><INPUT TYPE = "password" NAME = "pass"></TD>
</TR>
</TABLE>
<BR>
<INPUT TYPE = "submit" VALUE = "LOGIN">
<FORM>
</BODY>
</HTML>
```

3. **Save the file as information.html under the /usr/local/apache2/htdocs directory.**

To validate the login name and password that the user enters in the form, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to retrieve information from the form and store them in the variables:**

```
<?php
$val1 = $_GET['logname'];
$val2 = $_GET['pass'];

/* Enter the following code to set a cookie for logname that does
not expire for a week. The code shown earlier sets a cookie named
logname that expires as soon as the user closes the browser. The
value that gets stored in the $val variable is assigned to the
cookie. */

setcookie("logname",$val1);

/* Enter the following code to validate whether the Login Name and
Password fields are left blank: */
```

Session 20

Working with Cookies (Lab)


```
if ($val1=="")
{
     echo "Please enter the name!";
     echo "<HTML>";
     echo "<HEAD>";
     echo "<TITLE> Validate</TITLE>";
     echo "</HEAD>";
     echo "<BODY>";
     echo "<A HREF='information.html'> Back </A>";
     echo "</BODY>";
     echo "</HTML>";
}
else if($val2=="")
{
     echo "Please enter the password!";
     echo "<HTML>";
     echo "<HEAD>";
     echo "<TITLE> Validate</TITLE>";
     echo "</HEAD>";
     echo "<BODY>";
     echo "<BR>";
     echo "<A HREF='information.html'> Back </A>";
     echo "</BODY>";
     echo "</HTML>";
}

/* The code snippet uses the Header() function to redirect the
information to homepage.php that is the home page of the Web site. */
else
{
     Header("Location: homepage.php");
}
?>
```

3. **Save the file as validate.php in the /usr/local/apache2/htdocs directory.**

Session 20

Working with Cookies (Lab)

To create the home page of the Web site, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to retrieve the login name from the cookie and store it in the variable:**

```
<?php

// Store the value of the cookie logname
$logcookie = $_COOKIE['logname'];

// Display the various items available for shopping:
echo "<HTML>";
echo "<HEAD>";
echo "<TITLE> Homepage</TITLE>";
echo "</HEAD>";
echo "<BODY>";
echo "<ALIGN='right'>";

// Display the value of the cookie logcookie
echo "Welcome $logcookie ";
echo "<BR><A HREF='logout.php'>Logout</A>";
echo "<CENTER>";
echo "<H3> Shopper's Paradise </H3>";
echo "<H5> Shop till you drop!!! <H5>";
echo "<HR>";
echo "<BR>";
echo "<TABLE>";
echo "<TR ALIGN='center'>";
echo "<TD><A HREF='conf.php'>Confectionery</A></TD>";
echo "</TR>";
echo "<TR ALIGN='center'>";
echo "<TD><A HREF='flowers.php'>Flowers</A></TD>";
echo "</TR>";
echo "<TR ALIGN='center'>";
echo "<TD><A HREF='access.php'>Accessories</A></TD>";
echo "</TR>";
echo "<TR ALIGN='center'>";
```

Session 20

Working with Cookies (Lab)


```
echo "<TD><A HREF='perf.php'>Perfumes</A></TD>";
echo "</TR>";
echo "<TR ALIGN='center'>";
echo "<TD> <A HREF='apparel.php'>Apparel</A></TD>";
echo "</TR>";
echo "</TABLE>";
echo "</CENTER>";
echo "</BODY>";
echo "</HTML>";
?>
```

The code displays the cookie value stored in the variable, **`$logcookie`**. It also displays items such as, Confectionery, Flowers, Accessories, Perfumes, and Apparel on the Web page.

3. **Save the file as homepage.php in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to enable the user to log out from the Web site:**

```
<?php
$logcookie = $_COOKIE['logname'];

// Deletes logname cookie
setcookie("logname");

// Redirects users to the main information.html page for login
Header("Location: information.html");
?>
```

6. **Save the file as logout.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

Session 20

8. **Enter http://localhost/information.html in the Address bar and press Enter.**

   Figure 20.1 displays the Login page.

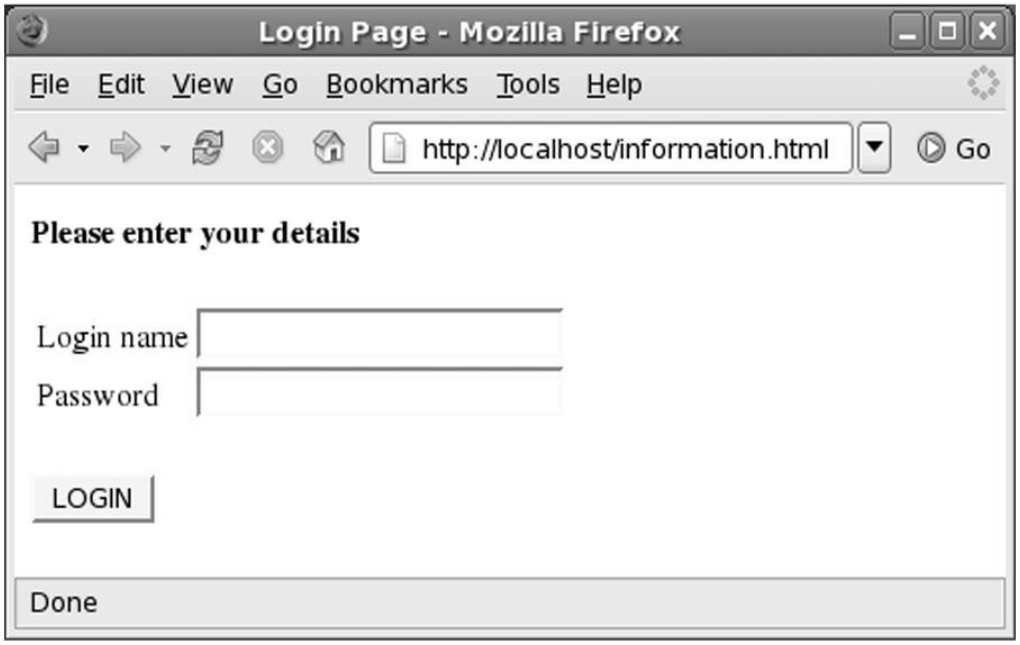


Figure 20.1: Login Page

9. **Enter the login name and password.**

Session 20

10. **Click LOGIN.**

    Figure 20.2 displays the Homepage.

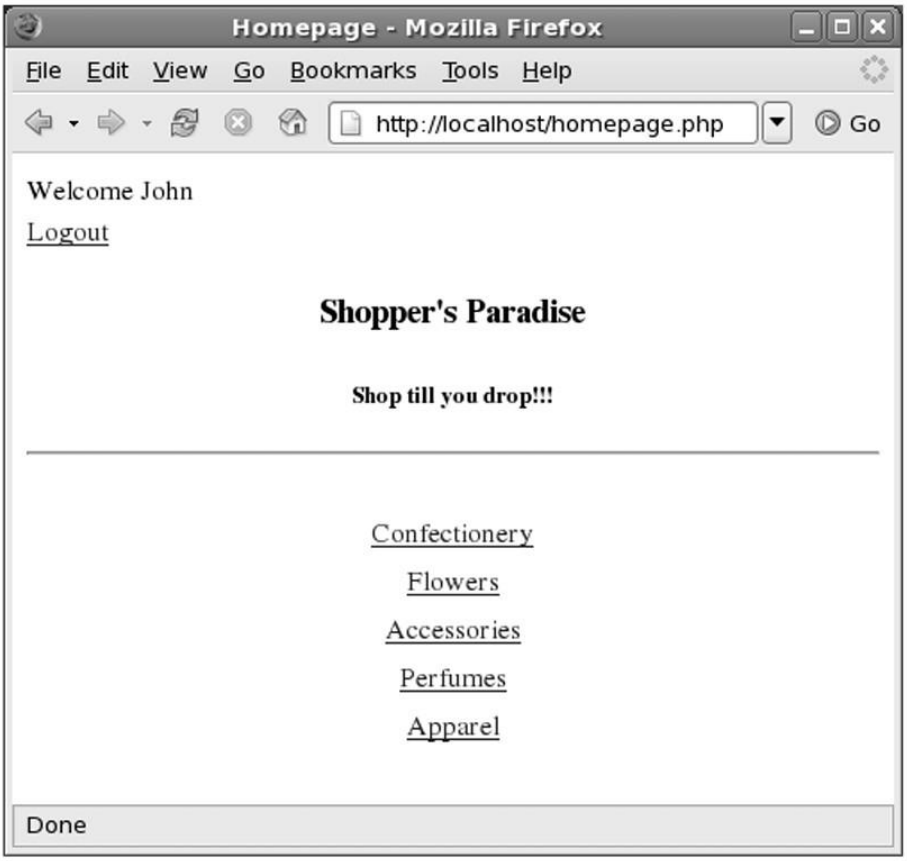


Figure 20.2: Homepage Details

To create the subsequent pages of the Web site, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to display the various items for the Confectionery category:**

```
<HTML>
<HEAD>
<TITLE> Confectionery Details </TITLE>
</HEAD>
```

Session 20

Working with Cookies (Lab)

Lab Guide

```
<BODY>
<?php
$logcookie = $_COOKIE['logname'];
echo "</HEAD>";
echo "<BODY>";
echo "<CENTRE>";
echo "<ALIGN='right'>";
echo "Welcome $logcookie ";
echo "<BR><A HREF='logout.php'>Logout</A>";
echo "<CENTER>";
echo "<H3> Shopper's Paradise </H3>";
echo "<H5> Shop till you drop!!! <H5>";
echo "<HR>";
echo "<BR>";
echo "<TABLE BORDER='1'>";
echo "<TR ALIGN='center'>";
echo "<TH>Code</TH>";
echo "<TH>Name</TH>";
echo "<TH>Price</TH>";
echo "</TR>";
echo "<TR ALIGN='center'>";
echo "<TD> C001 </TD>";
echo "<TD><A HREF=''> Vanilla Crush </A></TD>";
echo "<TD> $4.00 </TD>";
echo "</TR>";
echo "<TR ALIGN='center'>";
echo "<TD> C002 </TD>";
echo "<TD><A HREF=''> Choco Chunks </TD>";
echo "<TD> $7.00 </TD>";
echo "</TR>";
echo "<TR ALIGN='center'>";
echo "<TD> C002 </TD>";
echo "<TD><A HREF=''> Strawberry Delight </A></TD>";
echo "<TD> $9.00 </TD>";
echo "</TR>";
echo "</TABLE>";
echo "</CENTER>";
?>
</BODY></HTML>
```

3. **Save the file as conf.php under the /usr/local/apache2/htdocs directory.**

4. **Enter http://localhost/information.html in the Address bar and press Enter. The Login page is displayed.**

5. **Enter John and john as the username and password.**

6. **Click LOGIN. The Homepage is displayed.**

7. **Click Confectionery. Figure 20.3 displays the Confectionery details page.**

Figure 20.3: Confectionery Details Page

Session 20

Working with Cookies (Lab)

Similarly, create Web pages to display various products available for sale under the Flowers, Accessories, Perfumes, and Apparel categories. Table 20.1 list all the products and their details under these categories.

Product Category	Product Code	Name	Price
Flowers	F004	Tulip Bouquet	$6.00
	F010	Red Rose Bouquet	$8.00
	F011	Lilacs	$10.00
Accessories	A001	Diamond Bracelet	$95.00
	A006	Diamond Ring	$80.00
	A012	Diamond Earrings	$50.00
Perfumes	P002	Charles	$38.00
	P008	Maui Rain	$30.00
	P018	Night Mist	$25.00
Apparel	AP001	Three piece Suit	$200.00
	AP018	Wrinkle-free pant	$190.00
	AP020	Wrinkle-free shirt	$120.00

Table 20.1: Product Details

Session 20

Working with Cookies (Lab)

Do It Yourself

1. Shopper's Paradise is a shopping company that deals in selling goods online. The Web site offers items, such as Confectionery, Flowers, Apparels, Accessories, and Perfumes for sale. The company requires users to provide the following information to place orders:

   - Full name
   - Address
   - Email Address
   - Credit Card Number
   - Telephone Number

   a. Create a cookie that stores all the user information.

   b. Store the orders placed by the user in the database.

Session 21

Session Management in PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Define a session.*
- *Explain the procedure to work with a session.*
- *Describe the methods to start a session.*
- *Explain the method to register a session.*
- *Describe the method to end a session.*
- *Explain the use of the php.ini file.*

21.1 Introduction

Sessions are similar to cookies and enable the functionality of storing temporary user information. The only 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.

In this session, you will learn about sessions and session variables. You will also learn how to register and work with PHP session IDs. Additionally, you will learn about the `php.ini` file.

21.2 Sessions

Web browsers and Web servers have a stateless interaction and do not maintain track of user sessions. HTTP is a stateless protocol that enables Web browsers to communicate with Web servers. This protocol has no methods or functions to maintain the status of a particular user; even the Web server cannot distinguish user specific data. However, users can navigate and find information using the hyperlinks.

Session 21

Figure 21.1 displays the interaction between the Web browser and the Web server.

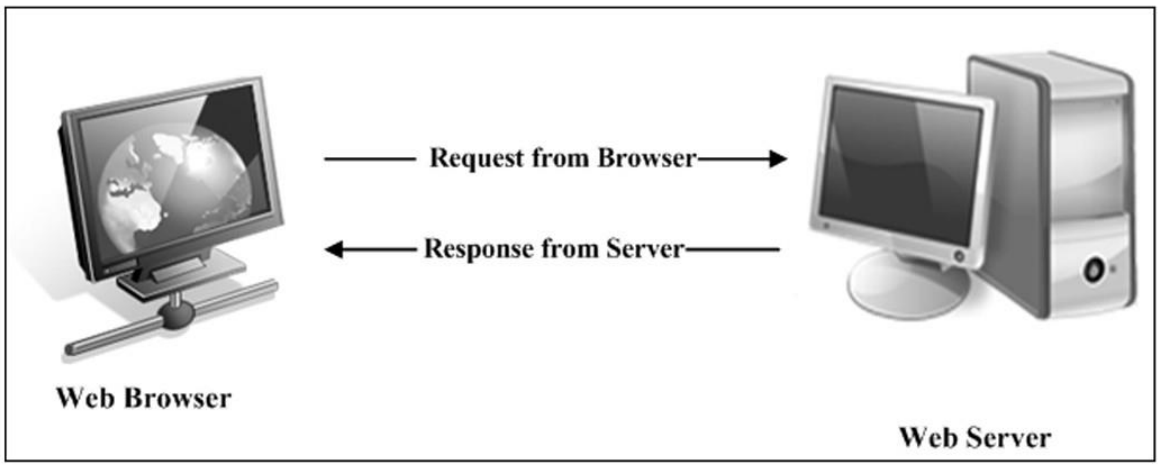


Figure 21.1: Interaction between the Web Browser and the Web Server

Web sites that require complex user interaction need session tracking and cannot depend on the HTTP or Web servers. Sessions allow Web sites to store user requests and information on the Web server. A session refers to the total time the user accesses information on a particular Web site before exiting the Web site. Session management is used to manage data for a particular user in a specific session.

PHP sessions enable distinguishing user specific information for the entire duration of the session.

21.3 Importance of a Session

Consider an example, in a particular Web site, the user has to first register and then log on to access any information. For such authentication procedures, the state of the user has to be maintained across the Web site. Web sites traditionally use `GET` and `POST` methods to pass user information from one script to another. When these methods are used, PHP assigns user information variables in the following format:

```
$name = $_GET['name'];
```

In the code, the variable `$name` stores the value that the script retrieves from the HTML form. This process of transferring user information is time consuming and not essential for large Web sites.

Session 21

Figure 21.2 displays the code to be used across all the Web pages of the Web site.

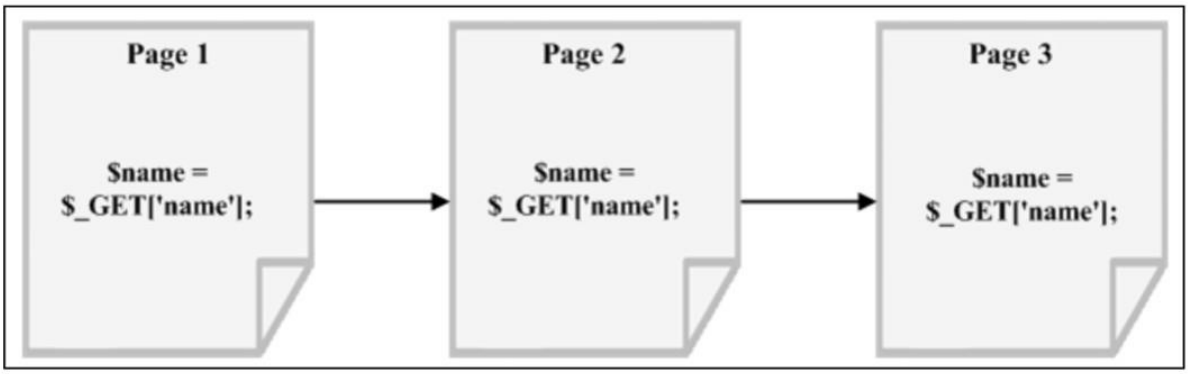


Figure 21.2: Traditional Transfer of User Information

The image shows the transfer of user information from one page to another. Consider a scenario, where you have to store and retrieve information for 20 or more users across 10 different pages. Due to this disadvantage of using the `GET` and `POST` methods, Web developers prefer using cookies. Figure 21.3 displays the assignment of cookies by the Web server to the browser.

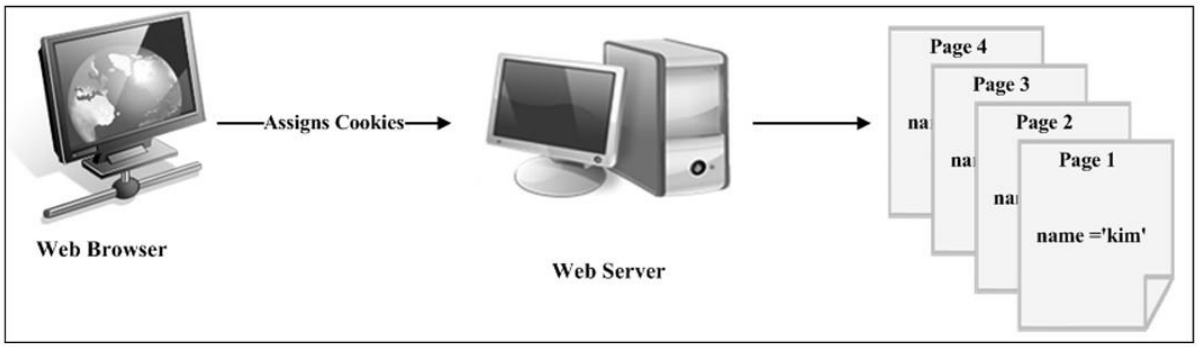


Figure 21.3: Assignment of Cookies to the Web Browser

Cookies enable to store data into a variable and access it across all the pages of the Web site. Cookies are prone to security risks because the user information is saved at the client-end. The risks involved are greater when users access Web sites from a public computer or a shared computer.

For example, a user purchases an item from a Web site from a shared computer. While placing the order, the user enters all his personal information, such as name, age, address, and credit card information. All these personal details are stored in cookies on the shared computer leaving it vulnerable to misuse by another user.

Concepts

The 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 the temporary file folders. Users often delete the temporary Internet files to improve the performance of a system. Then, the Web sites allot a new cookie to the user.

- **Multiple cookies to the same user** - Cookies enable Web sites to identify users according to their computers. Web sites allot a different cookie to the same person every time the user accesses the Web site from different computers. The statistics of the Web site 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 Web site.

- **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** - Web sites 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.

Sessions play an important role in such situations. The security of the Web site increases because unauthorized users cannot access the information. Sessions eliminate deletion and assignment of new cookies to the same user. The size of a cookie does not affect the performance of a Web site. Both the Web server and Web browser benefit because statistical information in the server database is accurate. The user information is not lost from the server database.

Table 21.1 explains the difference between cookies and sessions.

Cookies	Sessions
Stores user information on the client system (Web Browser)	Stores user information on the Web server
Available even after the user exits the Web browser	Destroyed when the user exits the Web browser
Users can disable cookies	Users cannot disable sessions
Have size limits	Do not have size limits

Table 21.1: Difference Between Cookies and Sessions

21.4 Working with Sessions

A session commences when a user accesses a session-enabled Web site. Web server assigns a unique session ID to each user when the user starts a session.

Session 21

Session Management in PHP

The scripts store and access user information through the session ID depending on the following two situations:

- **Cookies enabled** - Web server allots a session ID to the Web browser through a cookie, using the `setcookie()` function. Cookies enable transfer of user information between the browser and the server. PHP stores session IDs in cookies. The scripts access the required information through cookies.

- **Cookies disabled** - Web server allots a session ID to the browser using the URL. The URL transfers user information from the browser to the server. PHP stores the session variables in a file and names the file based on the session ID. The scripts access the required user information by retrieving it through the URL.

While using a session, PHP stores all the user information in a file on the Web server. The file includes a session ID that is related the user's session variable. Each session ID identifies a different user and relates to a file that belongs to that user. Therefore, a session ID is referred to as a key that links user and user data. PHP destroys the session file once the user exits the Web site.

Figure 21.4 shows the working of a session.

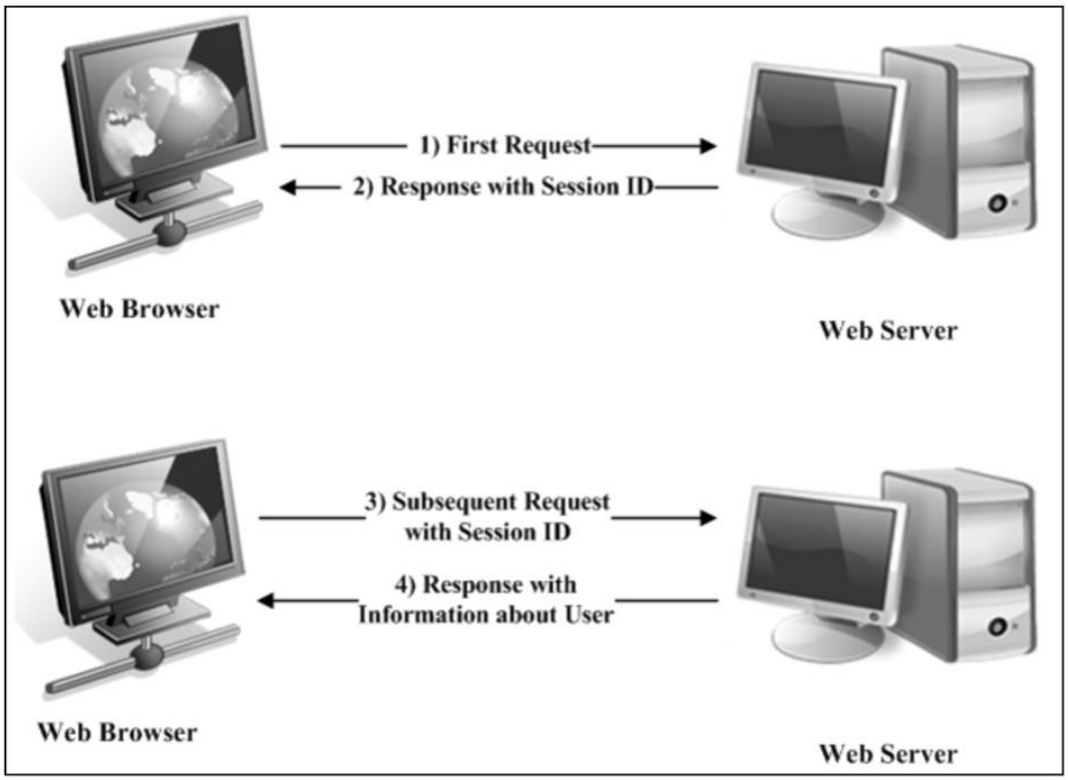


Figure 21.4: Working of a Session

Session 21

PHP works with sessions in the following sequence:

1. User accesses a session-enabled Web site.
2. The Web site checks the user identity, if the user is a new visitor or an ongoing session user.
3. If the user is a new visitor, the Web site allocates a unique session ID to the user. The Web site saves a cookie containing the session ID on the Web browser. The PHP engine creates a file that stores the session-related variables on the Web server.
4. The Web browser records the cookie that holds the session ID. The browser uses the same cookie to retrieve the session ID and record all the session-related information.
5. The session file is destroyed from the Web server, when the user exists from the Web site.

21.5 Lifecycle of a Session

There are three stages in the lifecycle of a session based on the communication between the Web browser and the Web server. They are as follows:

- Starting the session
- Registering the session variable
- Ending the session

21.5.1 Starting the Session

A session starts when a user logs on to the Web site. In PHP, the `session_start()` function enables to start a session. The process of starting a session is also called as initializing a session.

PHP creates a session file on the Web server when a new session starts. The session file is created in the `/tmp` directory. PHP assigns a name to this file based on the unique session identifier value generated by the PHP engine. The session identifier is also known as the session ID. The session ID is a hexadecimal string of 32 digits.

The file naming convention for the session file is as follows:

```
sess_<32_digit_hexadecimal_value>
```

The session file name is always preceded by sess_ and is followed by a random 32 digit hexadecimal value. For example, `sess_denkhu7869jhnkh789jas543hk87p5u3` is a session file name with session ID `denkhu7869jhnkh789jas543hk87p5u3`.

Session 21

The Web server passes the session ID as a response to the browser. The `setcookie` header field is sent along with the session ID. The response sets up a session cookie in the browser with the name `PHPSESSID` and the value of the identifier.

The `session_start()` function must be specified on the top of every Web page or before the start of the actual coding. The `session_start()` function always returns True. When the session starts, PHP checks for the validity of the session. If the session is valid and existing, it activates the frozen variables of the session. If the session is invalid or does not exist, it creates a session ID for the new session.

The scripts can use the session variables only when the variables are registered with the session library. The syntax for the `session_start()` function is as follows:

Syntax:

```
session_start();
```

Note: While using cookie-based sessions, call `session_start()` before anything is sent to the browser as an output.

To start a session, perform the following steps:

1. Open a new file in **gedit** text editor.

2. Enter the code as shown in Code Snippet 1.

   Code Snippet 1:

```
<?php
// initializing a session
session_start();
/* The session_id() function displays the session id that PHP allots to
a user. */
echo "The Session id is " .session_id(). ".<br>";
?>
```

3. Save the script as session1.php in the `/usr/local/apache2/htdocs/` directory.

4. Open the Mozilla Firefox Web browser.

5. Enter `http://localhost/session1.php` in the Address bar and press Enter.

Session 21

Figure 21.5 displays the output of the script.

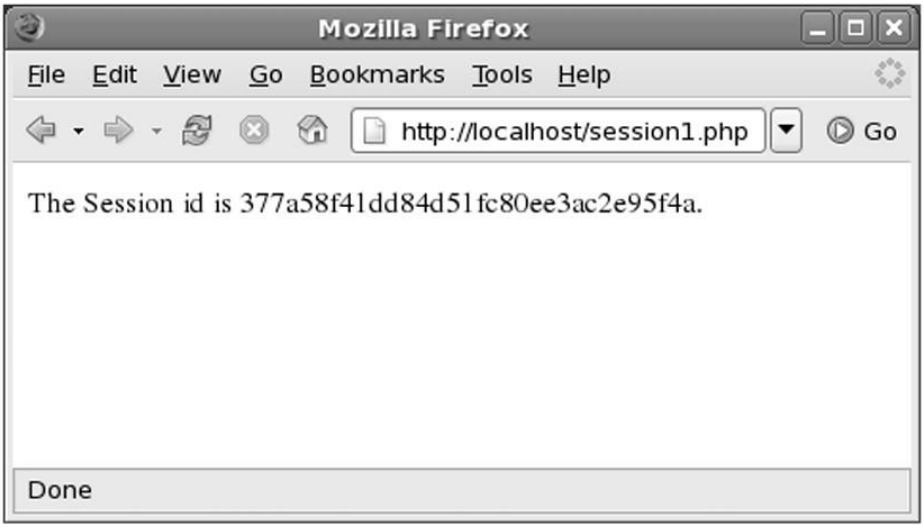


Figure 21.5: Displaying a Session ID

21.5.2 Registering the Session Variable

Variables in a session file contain user specific information. To work with the sessions across all the Web pages, session variables need to be registered with the session library. Session library enables creation, serialization, and storage of session data. The methods used to set a session variable are as follows:

- `$_SESSION[]` - recommended for PHP 4.1.0
- `$HTTP_SESSION_VARS[]` - Recommended for PHP 4.0.6 or less
- `session_register()` - not recommended as it has been deprecated

Session variables can be of any data type such as integer, string, Boolean, or object. PHP stores the session variables in a session file by serializing the values. PHP automatically handles the process of serializing the session variables.

Note: Serializing enables preserving information across requests. It is a process of transforming variables to a byte-code representation and is stored as a normal string.

Session 21

To register the value of a session variable, perform the following steps:

1. Open a new file in **gedit** text editor.

2. Enter the code as shown in Code Snippet 2.

   Code Snippet 2:

```
<?php
session_start();
$_SESSION['myname']= "Jessica";
?>
<HTML>
<HEAD> <TITLE> Session </TITLE></HEAD>
<BODY>
<A HREF = "mypage.php"> Homepage of MyPage.com </A>
</BODY>
</HTML>
```

3. Save the file as **`sessionstart.php`** in the `/usr/local/apache2/htdocs/` directory.

To display the value of the session variable, perform the following steps:

1. Open a new file in **gedit** text editor.

2. Enter the code as shown in Code Snippet 3.

   Code Snippet 3:

```
<?php
session_start();
$myname = $_SESSION['myname'];
?>
<HTML>
<HEAD> <TITLE> Homepage </TITLE></HEAD>
<BODY>
Welcome <?php echo $myname ?> to MyPage.com <br>
</BODY>
</HTML>
```

3. Save the file as `mypage.php` in the `/usr/local/apache2/htdocs/` directory.

Concepts

4. Open the Mozilla Firefox Web browser.

5. Enter `http://localhost/sessionstart.php` in the Address bar and press **Enter**.

   Figure 21.6 displays the output of the script.

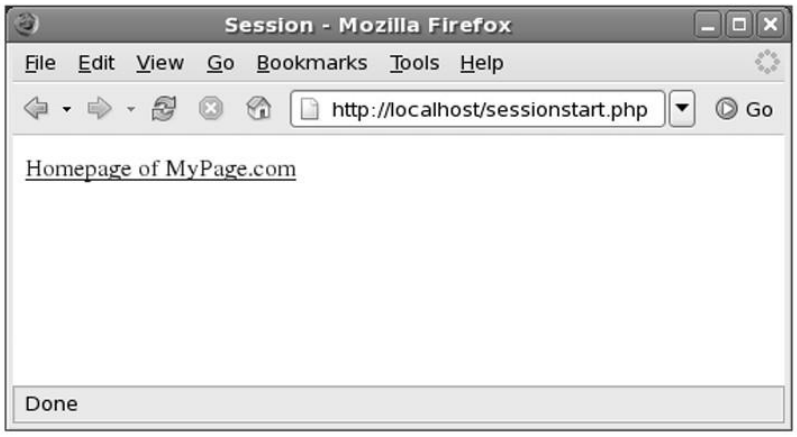


Figure 21.6: Displaying Link to MyPage.com

6. Click `Homepage of MyPage.com`. The Homepage appears with the message, `Welcome Jessica to MyPage.com`.

   Figure 21.7 displays the output of the script.

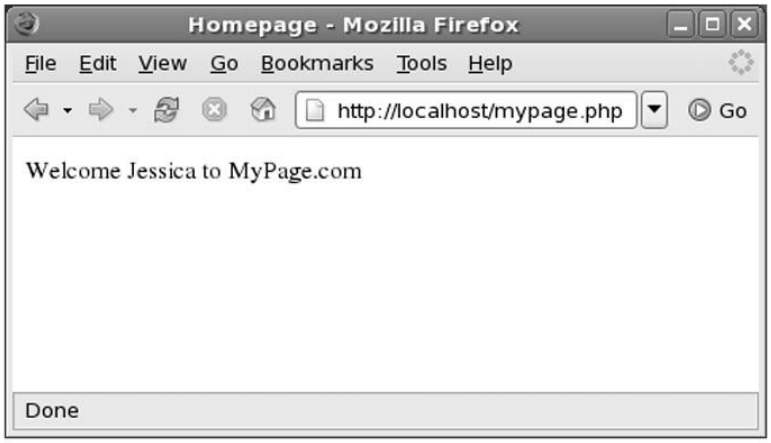


Figure 21.7: Displaying the Message on Homepage screen

Session 21

21.5.3 Ending the Session

When the user logs out of the Web site, the PHP script executes the `session_destroy()` function. This function removes the session file from the system. Although the session file is deleted, the `$PHPSESID` cookie is not removed from the Web browser.

The session must be initialized before the `session_destroy()` function is called.

Note: You can alter the lifetime of a session cookie by modifying the default value in the PHP configuration file.

The syntax for the `session_destroy()` function is as follows:

Syntax:

```
session_destroy();
```

The Web server performs several steps to determine that the session has ended. PHP uses the following configuration directives when a session ends:

- `gc _ maxlifetime()` - enables PHP to determine the time to wait before ending a session after the user exits the browser. This process of cleaning up the old session is called as garbage collection.
- `gc _ probability()` - enables PHP to determine with what probability the garbage collection routine must be invoked. If the value of this directive is set to 100%, then the process of garbage collection is performed on every request that the Web browser makes.

To destroy a session, perform the following steps:

1. Open a new file in **gedit** text editor.
2. Enter the code as shown in Code Snippet 4.

   Code Snippet 4:

```
<?php
session_start();
$myname = $_SESSION['myname'];
// The session_unset() function unregisters a session variable.
session_unset();
session_destroy();
echo "Session destroyed!";
?>
```

Session 21


```
<HTML>
<HEAD> <TITLE> Session </TITLE></HEAD>
<BODY>
<br>
<A HREF = "mypage.php"> Homepage of MyPage.com </A>
</BODY>
</HTML>
```

3. Save the file as **sessiondestroy.php** in the `/usr/local/apache2/htdocs/` directory.

4. Open the Mozilla Firefox Web browser.

5. Enter `http://localhost/sessiondestroy.php` in the Address bar and press **Enter**.

   Figure 21.8 displays the output of the script.

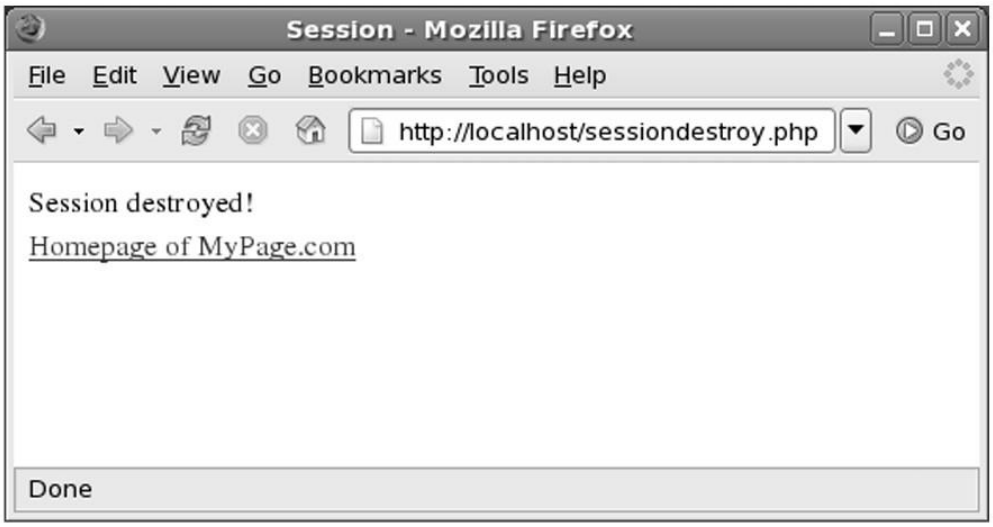


Figure 21.8: Ending the Session

6. To view the execution of the `session_destroy()` function, click the `Homepage of MyPage.com` hyperlink.

Figure 21.9 displays the output.

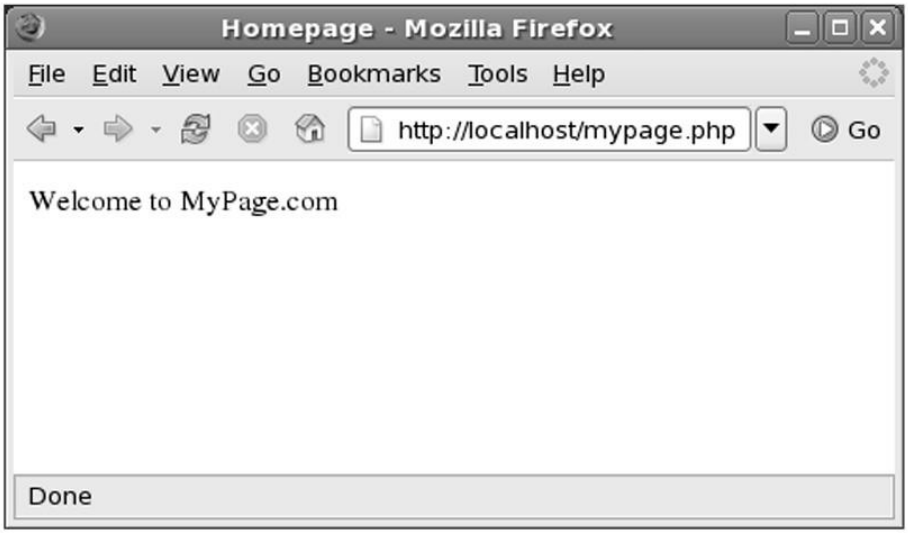


Figure 21.9: Destroying Session

The name `Jessica` does not appear in the message because the variable has been cleared.

21.6 Working with the php.ini File

A Web server can contain multiple `php.ini` files. The `locate` command in Linux can be used to identify the `php.ini` file used by PHP. You can create a `php.ini` file if it does not exist on the Web server. The complete source code must be downloaded to create a new `php.ini` file.

PHP interpreter works according to the instructions included in the `php.ini` file. The Web server searches sequentially for the `php.ini` file in the following locations:

- Directory where the PHP script was called
- Root of the Web directory
- Directory containing the `default.ini` file on the Web server

If a client system does not have a custom configuration file, the PHP configuration file of the Web server is used. On the Web server, the `php.ini` file is located in the `/etc` directory.

Session 21

Figure 21.10 displays the PHP configuration file.

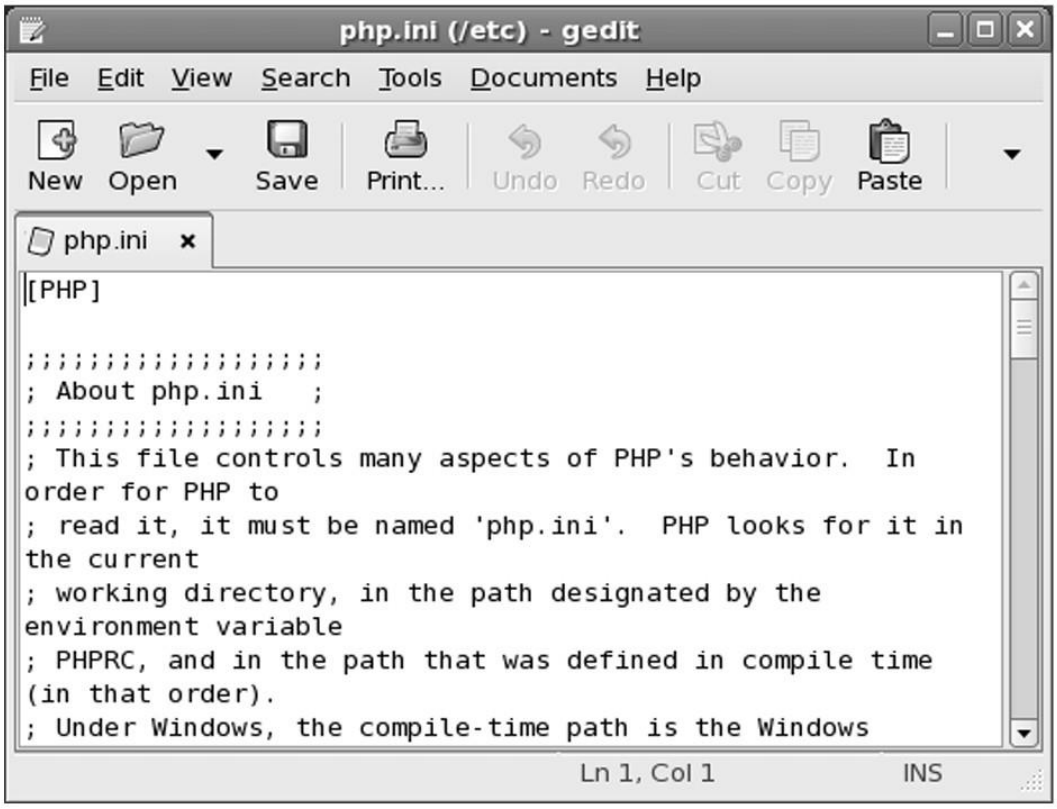


Figure 21.10: PHP Configuration File

The `php.ini` file contains directive listed in the `directive = value` format. Use a semicolon to add a comment to the file. PHP ignores lines that begin with semicolon or a single white space. Edit the `php.ini` file to customize the settings.

Table 21.2 lists the categories containing various options available in the `php.ini` file.

Categories	Options
Language Options	Enable PHP scripting language engine under Apache
	Allow ASP style tags
	Enforce year 2000 compliance
Safe Mode	Perform a UID compare check when opening files
	Allow executables under specific directories to be executed via exec family
	Allow user set environment variables that begin with PHP prefix

Session 21

Categories	Options
Font Colors	Indicate the colors that PHP uses for highlighting syntax
Misc	Indicate whether or not PHP discloses the fact that it is installed on the server
Resource Limits	Indicate the maximum time for script execution
	Indicate the maximum time for parsing request data
	Indicate the maximum amount of memory a script requires
Error handling and logging	Report all errors and warnings
	Report fatal compile time errors
	Report fatal run-time errors
	Report non-fatal error
	Report fatal errors that occur during initial startup of PHP
	Report user generated errors, warnings, and messages
Data Handling	Control list of separators used in PHP generated URLs to separate arguments
	Describe the order in which PHP registers Get, Post, Cookie, Environment, and built-in variables
Magic Quotes	Set magic quotes for incoming Get, Post, and Cookie data
	Use Sybase style magic quotes
	Automatically adds file before or after any PHP document
Path and Directories	Specifies the name of the directory under which PHP opens the script
	Specifies the name of the directory under which the loadable extensions exist
File Uploads	Indicate whether or not to allow HTTP file uploads
	Indicate temporary directory for HTTP uploaded files
	Indicate the maximum allowed size for upload files
Session	Store and retrieve data
	Indicate whether or not cookies should be used
	Initializes session on request startup
	Serializes data

Table 21.2: Categories of the PHP Configuration File

Several options can modify the functionality of the PHP session. Session category of the `php.ini` file include options, which are as follows:

- `session.save_handler` - specifies how PHP stores and retrieves session variables. Use either of the following values for this option:
  - `files`: indicates the use of the session files. This is the default value.

Session 21

- mm: stores and retrieves data from a shared memory
- user: stores and retrieves variables with custom defined handlers

➢ session.save _ path - specifies the name of the directory where the session files will be stored. By default, the session files are saved in the /tmp directory.

➢ session.use _ cookies - indicates whether PHP must send a session ID to the Web browser through a cookie. By default, a cookie stores the session ID. The value to enable a cookie to store a session ID is 1.

➢ session.use _ only _ cookies - indicates whether the modules can use only cookies for storing session IDs. By default, this option is disabled.

➢ session.cookie _ lifetime - specifies the lifetime of the cookie. The value is specified in seconds. By default, the lifetime of a cookie is set to 0 which means the cookie is destroyed once the browser is closed.

➢ session.name - manages the cookie name and form attributes such as GET and POST that holds the session ID. By default, the value of this option is PHPSESSID.

➢ session.auto _ start - enables sessions to automatically initialize if the session ID is not found in the browser request. It is recommended to disable the auto start feature because it increases overheads if no scripts require the session ID.

➢ session.cookie _ secure - specifies whether the cookies must be sent over secured connections. By default, the cookies are not sent through secured connections.

You can also modify other settings in the php.ini file, such as:

➢ register _ globals - controls the functioning of server, forms, and environment variables. When you enable this option, you can directly access the forms, server, and environment variables by their names.

If this option is disabled, you can retrieve variables using the GET or POST methods as follows:

```
$_POST['$variable_name'];

$_GET['$variable_name'];
```

If register_globals is enabled, you can directly access the variable using the variable name as follows:

```
$storeValue = $variable_name;
```

Session 21

In the code, `$variable_name` is the name of the variable that contains the session data of another Web page. The variable `$storeValue` stores the information included in the variable `variable_name`.

- `upload_tmp_dir` - sets the location of the temporary file that is uploaded with the HTML form. Any user can access the uploaded files in the default location of the Web server. You can create a directory and specify the path for storing the uploaded files in the `php.ini` file.

- `display_errors` and `display_startup_errors` - enables PHP to display errors on the Web browser. It is recommended to disable these variables, especially while creating dynamic Web pages. Dynamic Web pages may display user sensitive information, such as password, credit card details in event of an error.

- `log_errors` and `error_log` - enables PHP to display error logs. The `error_log` variable stores the path of the directory where the logs get stored. It is recommended to enable the `log_errors` variable, when the `display_errors` and `display_startup_errors` variables are disabled.

Session 21

Summary

- Cookies provide the functionality of storing temporary user information on the local computer.
- Sessions enable PHP to store user information on the Web server.
- HTTP is considered as a stateless protocol that enables Web browsers to communicate with the Web servers.
- Session refers to the total time a user accesses information on a particular Web site before exiting the Web site.
- A PHP script can access the session ID when the Web user enables or disables cookies.
- The three stages in the lifecycle of a session are: starting a session, registering a session variable, and ending a session.

Session 21

Check Your Progress

1. Which of the following options indicate whether the modules will use only cookies for storing session IDs?

   a. session.cookies

   b. session.cookie_lifetime

   c. session.use_cookies

   d. session.use_only_cookies

2. Which function enables PHP to determine the time to wait before ending a session after the user closes the browser?

   a. session_destroy()

   b. gc_maxlifetime()

   c. gc_probability()

   d. session_unregister()

3. Which function does PHP use to remove the session variable from a session?

   a. session_destroy()

   b. gc_maxlifetime()

   c. gc_probability()

   d. session_unregister()

4. 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

Concepts

Check Your Progress

5. The _____ option specifies whether the cookies should be sent over secured connections.

   a. session.cookie_secure

   b. session.secure

   c. session.secure_connection

   d. session.use_secure_connection

6. _____ controls the functioning of server, forms, and environment variables.

   a. register_globals

   b. uploads_tmp_dir

   c. session.control_variables

   d. session.register_variables

Session 21

Session Management in PHP

Answers

1. d
2. b
3. a
4. b
5. a
6. a

Session 22

Handling E-mail with PHP

Concepts

Objectives

At the end of this session, the student will be able to:

- *Describe the process of sending an e-mail using PHP.*
- *xplain the process of attaching files with an e-mail using PHP.*

22.1 Introduction

E-mail is the fastest way of communicating with people across the world. It takes only few minutes to send and receive messages as compared to the usual hand written letters. PHP provides the facility to send an e-mail. In this session, you will learn how to send an e-mail within PHP and also learn to attach files with the e-mail.

22.2 Sending an E-mail

In PHP, you can send an e-mail using the `mail()` function. You must specify the location of the current local mail server in the `php.ini` configuration file to use the `mail()` function. The `php.ini` file is located in the `/etc/` directory of the Linux operating system. Figure 22.1 displays most basic configuration in the `php.ini` file for implementing mail function.

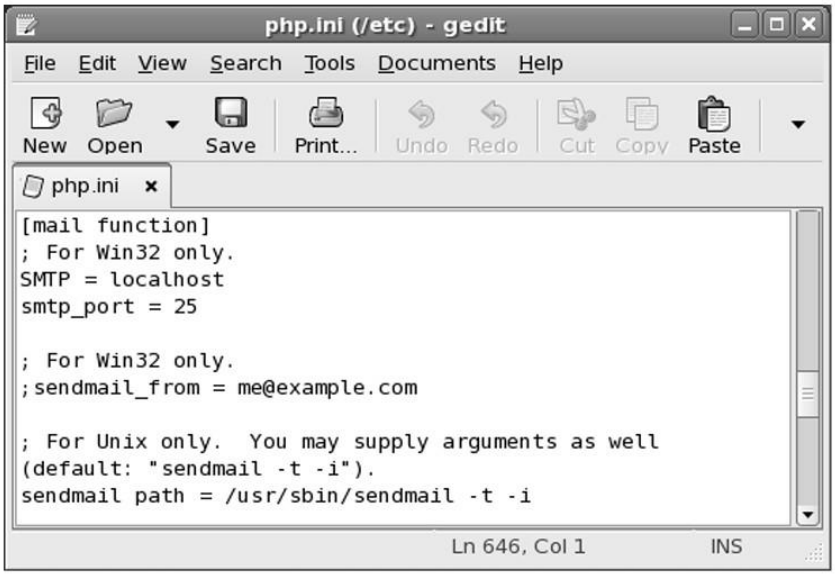


Figure 22.1: php.ini Configuration File

Session 22

Also, modify the sendmail.ini file as follows:

```
smtp_server=localhost

smtp_port=25

error_logfile=error.log

debug_logfile=debug.log

auth_username=<username>

auth_password=<password>
```

The syntax for the `mail()` function is as follows:

Syntax:

```
mail(to, subject, message , [additional_headers])
```

where,

`to` – specifies the recipient's e-mail address

`subject` – specifies the subject for the e-mail

`message` – specifies the message that is to be written in the e-mail or the body of the e-mail

`additional headers` – specifies additional information such as the e-mail address of the sender and attachments

Note: The recipient e-mail address is compulsory. The subject and the body arguments of the mail() function are optional.

If the e-mail is delivered to the recipient address, the control is transferred to the statement following the `mail()` function. Otherwise, PHP displays an error message.

Session 22

For example, to send an e-mail using the `mail()` function, enter the code as shown in Code Snippet 1, in a script named **`mail.php`**.

Code Snippet 1:

```
<HTML>
<BODY>
<?php
$to = "recipient@example.com";
$from = "yourname@example.com";
$subject = "Test e-mail";
$body = "This is an example for showing the usage of the mail() function.";
$send = mail($to, $subject, $body, $from);
if($send)
{
     echo "Mail sent to $to address!!!";
}
else
{
     echo "Mail could not be sent to $to address!!!";
}
?>
</BODY>
</HTML>
```

Figure 22.2 displays the output of the script.

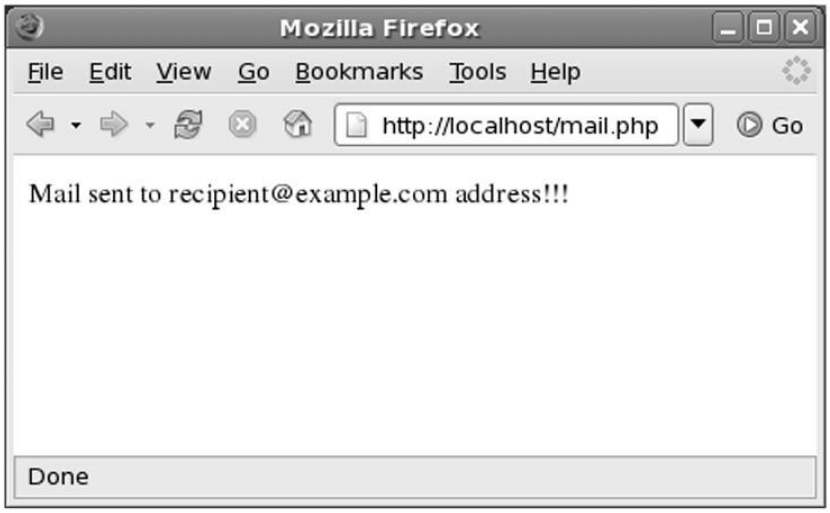


Figure 22.2: Sending an E-mail to a Single E-mail Address

Session 22

Handling E-mail with PHP

Concepts

The `mail()` function is also used for sending mails to more than one recipient. A comma (,) symbol is used for separating the e-mail addresses of the recipients.

For example, to send an e-mail to multiple recipients, enter the code as shown in Code Snippet 2, in a script named **`multi_rec.php`**.

Code Snippet 2:

```
<HTML>
<BODY>
<?php
$to = "recipient1@example.com, recipient2@example.com,
recipient3@example.com";
$from = "yourname@example.com";
for($i=0;$i<count($to);$i++)
{
      $to[$i] = trim($to[$i]);
      $subject = "An example";
      $body = "This is an example for showing the usage of the mail()
      function.";
      $send = mail($to, $subject, $body, $from);
      if($send)
      {
            echo "Mail was sent to all the addresses!!!";
      }
}
?>
</BODY>
</HTML>
```

Figure 22.3 displays the output of the script.

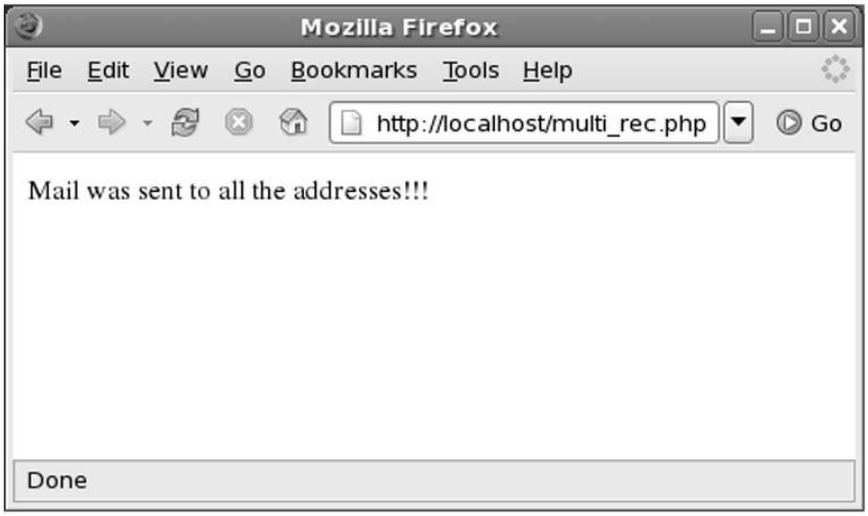


Figure 22.3: Sending an E-mail to Multiple E-mail Addresses

In the code, the `mail()` function is used to send an e-mail. If there is any blank space in between two recipient's e-mail address, the `trim()` function is used to remove it. The comma symbol is used to separate recipient addresses. The `for` statement is used for reading the number of recipient addresses.

PHP also enables to read the e-mail addresses of the recipients from the text file and send an e-mail to them.

For example, to send e-mail to multiple recipients whose addresses are stored in a text file, perform the following steps:

1. Open a new file in the **gedit** text editor.
2. Enter the following text:

   recipient1@example.com

   recipient2@example.com

   recipient3@example.com
3. Save the file as **`email_list.txt`** in the `/usr/local/apache2/htdocs` directory.
4. Assign read, write, and execute permissions to the **`email_list.txt`** file.
5. Open a new file in the **gedit** text editor.
6. Enter the code as shown in Code Snippet 3.

Session 22

Code Snippet 3:

```
<HTML>
<BODY>
<?php
error_reporting(0);
$multi = "/usr/local/apache2/htdocs/email_list.txt";
$to_mail = file('$email_list.txt');
$from = "yourname@example.com";

for($i=0;$i<count($to_mail);$i++)
{
     $to_mail[$i] = trim($to_mail[$i]);
     $to = implode(",",$to_mail);
     $subject = "An example";
     $body = "This is an example for the mail() function.";
     mail($to, $subject, $body, $from);
     echo "Mail was sent to all the addresses!!!";
}
?>
</BODY>
</HTML>
```

7. Save the file as **`multi _ email.php`** in the `/usr/local/apache2/htdocs` directory.

8. Open the Mozilla Firefox Web browser.

9. Enter **`http://localhost/multi _ email.php`** in the Address bar and press Enter.

Concepts

Figure 22.4 displays the output of the script:

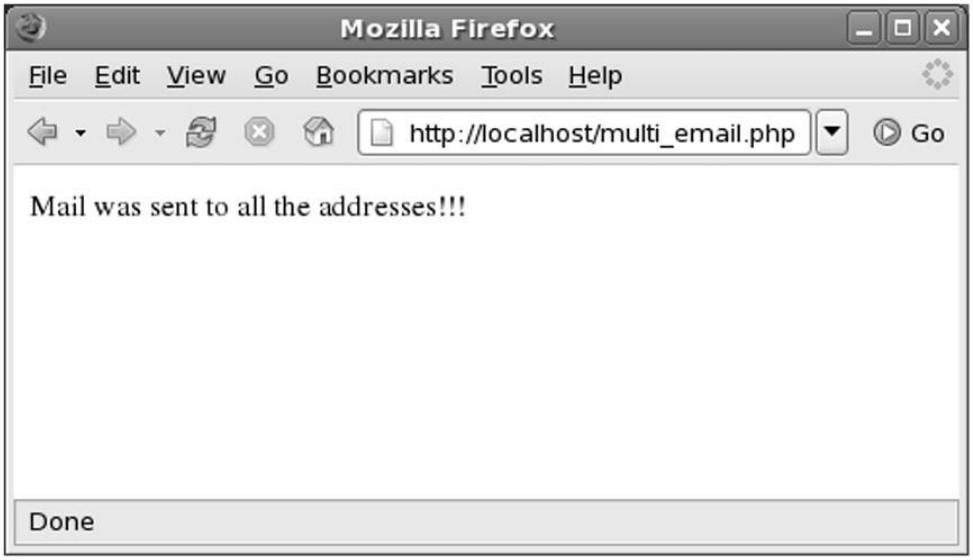


Figure 22.4: Sending E-mail to Multiple E-mail Addresses Stored in a Text File

In the code, the functions, such as `file()`, `trim()`, and `implode()` are used. The `file()` function reads the e-mail addresses from the **`email_list.txt`** file. It stores all the e-mail addresses in an array. The `trim()` function removes the blank spaces inserted between the e-mail addresses. The `implode()` function joins all the e-mail addresses in the array by separating each of them using a comma.

22.3 Attaching Files in an E-mail

You can also send attachments with an e-mail. An attachment is any file that you want to send along with the e-mail. An attachment file can be a simple text file, a document file, or a graphics file. A file is attached with the e-mail by setting up the header.

To send an e-mail with an attachment, perform the following steps:

1. Open a new file in the **gedit** text editor.

2. Enter the following text:

   This is a sample attachment file.

3. Save the file as **`example.txt`** in the `/usr/local/apache2/htdocs` directory.

4. Assign read, write, and execute permissions to the **`example.txt`** file.

Session 22

5. Open a new file in the **gedit** text editor.

6. Enter the code as shown in Code Snippet 4.

Code Snippet 4:

```
<?php
// filenames to be sent as attachment
$files = "/usr/local/apache2/htdocs/example.txt";

// e-mail fields: to, from, subject, and so on
$to = "recipient@example.com";
$from = "yourname@example.com";
$subject ="Test e-mail";
$message = "Sample message";
$headers = "From: $from";

// boundary
$semi_rand = md5(time());
$mime_boundary = "==Multipart_Boundary_x{$semi_rand}x";

// headers for attachment
$headers .= "\nMIME-Version: 1.0\n" ."Content-Type: multipart/mixed;
            \n" . " boundary=\"{$mime_boundary}\"";

// multipart boundary
$message = "This is an example for mail attachment in MIME format.
            \n\n" . "--{$mime_boundary}\n" .
            "Content-Type: text/plain; charset=\"iso-8859-1\"\n" .
             $message . // "\n\n";
$message .= "--{$mime_boundary}\n";
// preparing attachments
$file = fopen($files,"rb");
$content = fread($file,filesize($files));
fclose($file);
$message .= "Content-Type: {\"application/octet-stream\"};\n" . "
           name=\"$files\"\n" .  "Content-Disposition: attachment;\n"
             . " filename=\"$files\"\n" .  $content . "\n\n";
```

```
$message .= "--{$mime_boundary}\n";
// send
$send_mail = @mail($to, $subject, $message, $headers);
if ($send_mail)
{
     echo "<p>Mail sent to $to!!!!</p>";
}
else
{
     echo "<p>Mail could not be sent to $to!!!!</p>";
}
?>
```

7. Save the file as **`mailattach.php`** in the `/usr/local/apache2/htdocs` directory.

8. Open the Mozilla Firefox Web browser.

9. Enter **`http://localhost/mailattach.php`** in the Address bar and press Enter.

Figure 22.5 displays the output of the script.

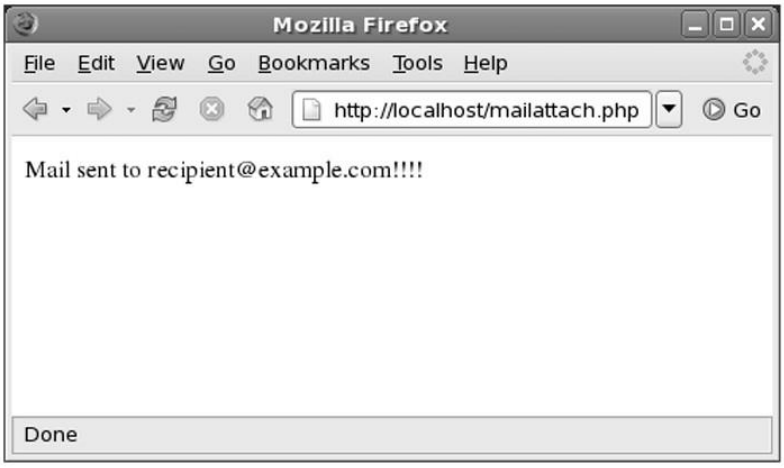


Figure 22.5: Sending an E-mail with an Attachment

In the code, a boundary is created between the actual e-mail and the attachments. The content-type header indicates the type of the file that will be attached with the e-mail. The `fopen()` function opens the file that is attached with the e-mail in the read mode. The `fread()` function reads the contents of the opened file.

Session 22

The `base64_encode()` and `chunk_split()` are the two built-in PHP functions.

The `base64_encode()` function is used for encoding the specified string. The `chunk_split()` function is used for splitting the data into a series of smaller parts. These two functions can be used in the codes for reading the file content.

Note: The base64_encode() and chunk_split() functions are optional.

22.4 Other Options for Sending Mail

The PHP mail() function does not support SMTP authentication. To overcome this, you can use the PEAR Mail package for SMTP Authentication and SSL Connection.

This free package offers flexibility needed and authenticates with your desired outgoing mail server. It also supports enhanced security through encrypted SSL connections.

Session 22

Summary

- The mail() function is used for sending an e-mail.
- The trim() function removes blank spaces included in between two recipient e-mail addresses.
- The implode() function is used to join all the addresses present in the array with a comma operator.
- An attachment can also be sent with an e-mail by setting up the header of the mail() function.
- The uniqid() function assigns a unique identifier to the original mail and creates a boundary between the e-mail and the attachment.
- The base64_encode() function is used for accepting the data in the form of a string and returns the data in the original form to the user.
- The chunk_split() function is used for separating the data into smaller parts.

Session 22

Check Your Progress

1. Which of the following function does PHP use for sending an e-mail?
   a. implode()
   b. sendmail_from()
   c. mail()
   d. file()

2. The _____ symbol is used for separating recipient addresses.
   a. /
   b. &
   c. ,
   d. :

3. The trim() function removes _____ from the recipient address.
   a. ascii characters
   b. comma operators
   c. blank spaces
   d. wide spaces

4. The _____ function joins all the e-mail addresses in the array.
   a. file()
   b. trim()
   c. mail()
   d. implode()

Check Your Progress

5. The _____ function opens the attached file in read mode.

   a. fread()

   b. implode()

   c. uniqid()

   d. fopen()

6. The _____ function splits the data into smaller parts.

   a. chunk_split()

   b. fread()

   c. base64_encode()

   d. fopen()

Session 22

Handling E-mail with PHP

Concepts

Answers

1. a
2. c
3. c
4. d
5. d
6. a

BE AHEAD OF EVERYONE ELSE

READ ARTICLES

Onlinevarsity

Session 23

Handling E-mail with PHP (Lab)

Objectives

At the end of this session, the student will be able to:

- *Send an e-mail.*
- *Attach files with an e-mail.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Sending E-mail

PHP provides the facility to send an e-mail. In PHP, you can send an e-mail using the `mail()` function.

To send an e-mail to multiple recipients, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form that accepts the recipient address, subject, and e-mail message:**

```
<HTML>
<BODY>
<FORM METHOD = "GET" ACTION = "multiusers.php">
From:<INPUT TYPE = "TEXT" NAME = "from" VALUE = jack@bluemountain.com
SIZE=30><br><br>
To:        
<INPUT TYPE = "TEXT" NAME = "to" SIZE = 30><br>
Cc:        
<INPUT TYPE = "TEXT" NAME = "cc" SIZE = 30><br>
Bcc:      
<INPUT TYPE = "TEXT" NAME = "bcc" SIZE = 30><br>
Subject:
```

Session 23

Handling E-mail with PHP (Lab)


```
<INPUT TYPE = "TEXT" NAME = "subject" SIZE = 30>
<br><br>
<TEXTAREA NAME = "mail_body" ROWS = "6"COLUMNS = "50"></TEXTAREA>
<br>
<INPUT TYPE = "SUBMIT" NAME = "send" VALUE = "Send">
</FORM>
</BODY>
</HTML>
```

3. **Save the file as multiusers.html in the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to accept the recipient address, subject, and e-mail message:**

```
<HTML>
<BODY>
<?php
$mail_to = $_GET['to'];
$mail_from = $_GET['from'];
$mail_cc = $_GET['cc'];
$mail_bcc = $_GET['bcc'];
$mail_header = $mail_from;
$mail_header = $mail_cc;
$mail_header = $mail_bcc;
for($i=0;$i<count($mail_to);$i++)
{
    $mail_to[$i] = trim($mail_to[$i]);
    $mail_subject = $_GET['subject'];
    $body = $_GET['mail_body'];
    $confirm = mail($mail_to,$mail_subject,$body,$mail_header);
    if($confirm)
     {
         echo "To: $mail_to  <BR>";
         echo "From: $mail_from <BR><BR>";
         echo "Cc: $mail_cc  <BR>";
         echo "Bcc: $mail_bcc <BR><BR>";
         echo "Subject: $mail_subject  <BR><BR>";
         echo $body;
```

```
            echo "<BR><BR>Mail sent to all the addresses.";
        }
            else
            {
                echo "<BR><BR>Mail could not be sent.";
            }
    }
    ?>
    </BODY>
    </HTML>
```

6. **Save the file as multiusers.php in the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/multiusers.html in the Address bar and press Enter.**

   Figure 23.1 displays the output of the script.

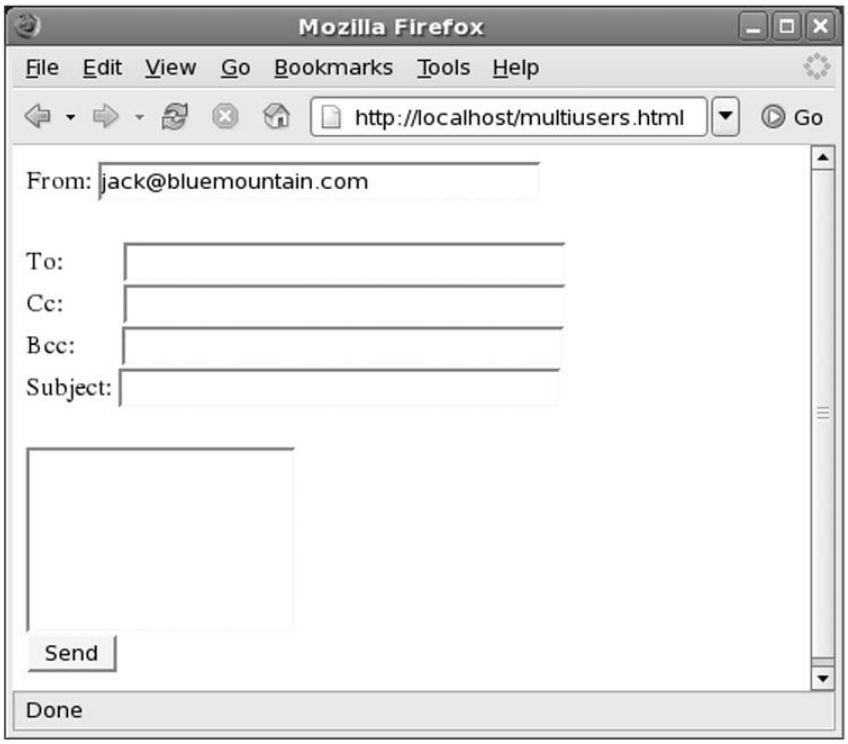


Figure 23.1: E-mail Form

Lab Guide

9. **Enter john@mail.com, jenny@syvalley.com, and samson@bluemountain.com in the To box.**

10. **Enter Birthday Wishes in the Subject box.**

11. **Enter the message Wish you many many happy returns of the day! Take care. Regards, Jack.**

12. **Click Send.**

Figure 23.2 displays the output of the script.

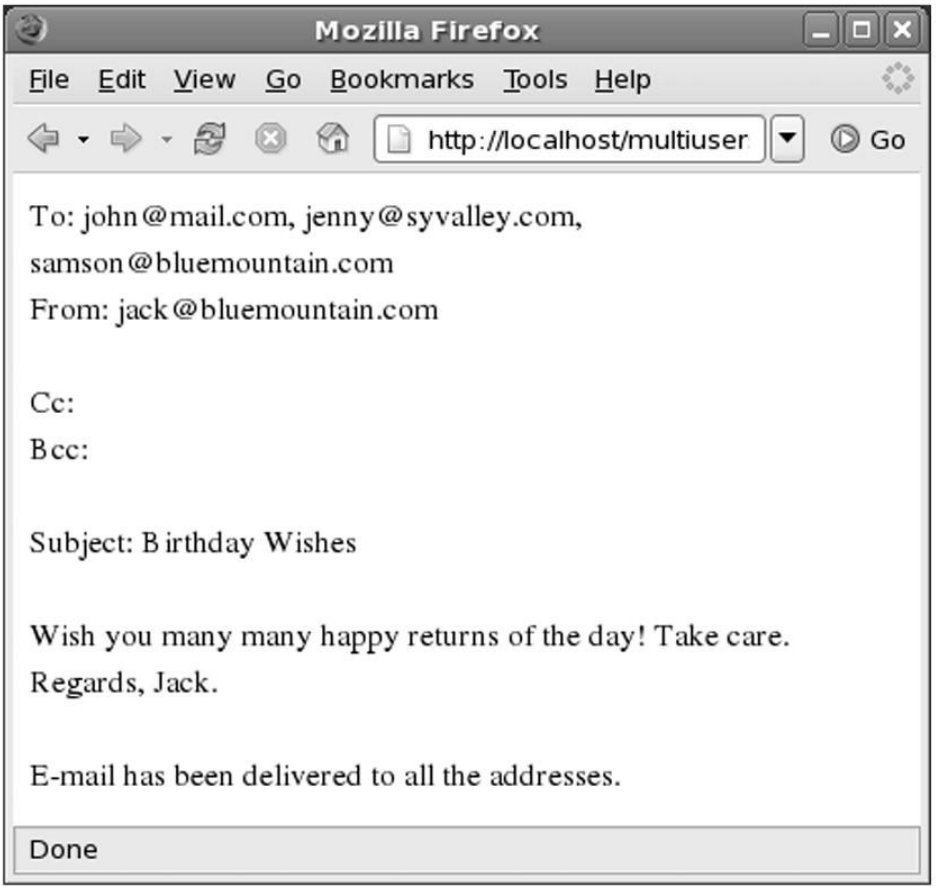


Figure 23.2: Viewing the E-mail Information

Session 23

Handling E-mail with PHP (Lab)

Attaching Files in an E-mail

An attachment such as a file can also be sent with an e-mail. An attachment file can be any simple text file, document file, or a graphics file. A file is attached with the e-mail by setting up the header.

To attach a file with an e-mail, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form for sending an e-mail with the attachment:**

```
<HTML>
<BODY>
<FORM METHOD = "GET" ACTION = "mail.php">
From:<INPUT TYPE = "TEXT" NAME = "from" SIZE=30
VALUE="jack@bluemountain.com" ><br>
To:        
<INPUT TYPE = "TEXT" NAME = "to" SIZE=30><br>
Cc:        
<INPUT TYPE = "TEXT" NAME = "cc" SIZE = 30><br>
Bcc:      <INPUT TYPE = "TEXT" NAME =
"bcc" SIZE = 30><br>
Subject: <INPUT TYPE = "TEXT" NAME = "subject" SIZE=30> <br>
Attachment: <INPUT TYPE = "TEXT" NAME = "attach" VALUE = "job.txt">
<br><br>
<TEXTAREA NAME = "mail_body" ROWS = "6" COLUMNS = "50"></TEXTAREA>
<br><br>
<INPUT TYPE = "SUBMIT" NAME = "send" VALUE = "Send">
</FORM>
</BODY>
</HTML>
```

3. **Save the file as mail.html under the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to send an e-mail to the receiver with an attachment:**

```
<?php
ini_set('auto_detect_line_endings', TRUE);
```

```
$file = "/usr/local/apache2/htdocs/job.txt";
$fp = fopen($file, "w");
fwrite($fp,"Job opening for students in XYZ dept. Send in your
                                                    resume.");
fclose($fp);
$to = $_GET['to'];
$from = $_GET['from'];
$subject = $_GET['subject'];
$body = $_GET['mail_body'];
$mime_boundary = "" . md5(uniqid(mt_rand(), 1));
$header = "";
$header.= $from;
$header.= $to;
$header.= "X-mailer: PHP/" . phpversion(). "\n";
$header.= "MIME-Version: 1.0\n";
$header.= "Content-Type: multipart/related";
$header.= "boundary=\".$mime_boundary.";
$header.= "type=\"text/plain\"\n";
$message = "";
$message.= "Urgent Requirement.";
$message.= "\r\n\r\n";
$message.= "—".$mime_boundary."\r\n";
$message.= "Content-type:text/plain;charset=\"iso-8859-1\"\r\n";
$message.= "—".$mime_boundary."\r\n";
$message.= "Content-Disposition:attachment;\r\n";
$message.= "name=\"filename.extn\"\r\n";
$message.= "\r\n\r\n";
$message.= $file;
$message.= "\r\n\r\n";
$message.= "--".$mime_boundary."\r\n";
$message.= "<P>.$message</P>";
$message.= str_replace(chr(10),"<P></P>",$message);
$mail_send = mail($to, $subject, $body, $header);
if($mail_send)
 {
     echo "To : $to  <BR> From : $from <P><BR>";
     echo "Subject : $subject<P><BR>";
     echo "$body";
     echo "<BR><BR>The mail is sent to $to from $from.";
```

Session 23

Handling E-mail with PHP (Lab)


```
 }
else
{
    echo "<BR><BR>Mail could not be sent.";
}
?>
```

Note: The attachment file (job.txt) must have read, write, and execute permissions.

6. **Save the file as mail.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

8. **Enter http://localhost/mail.html in the Address bar and press Enter.**

   Figure 23.3 displays the output of the script.

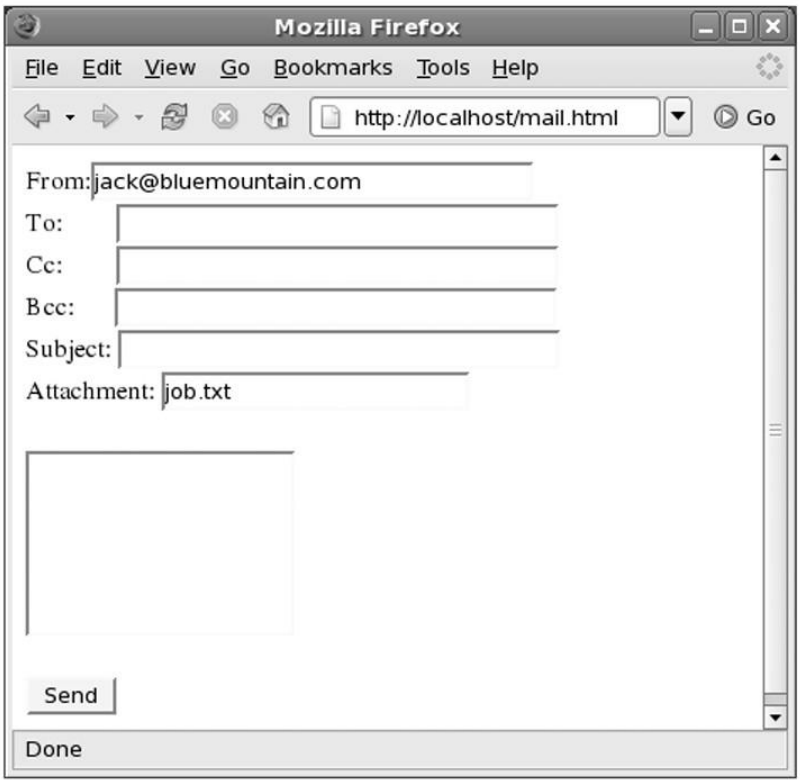


Figure 23.3: E-mail Form with the Attachment Field

Lab Guide

9. **Enter samson@bluemountain.com as the recipient address.**

10. **Enter An urgent opening in the subject box.**

11. **Enter the message Please check the job details in the attached file.**

12. **Click Send.**

Figure 23.4 displays the output of the script.

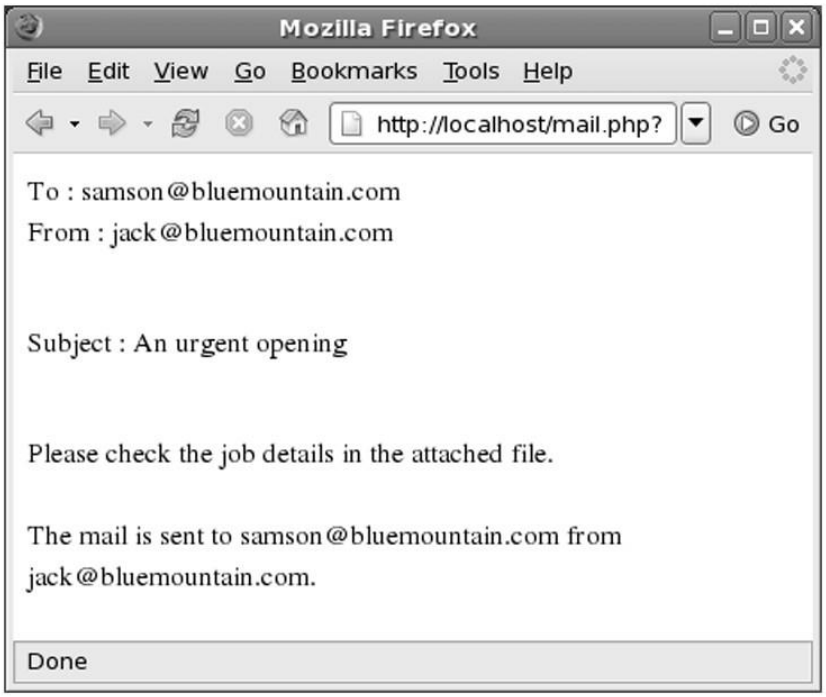


Figure 23.4: Viewing the E-mail Information Sent with Attachment

Session 23

Handling E-mail with PHP (Lab)

Do It Yourself

1. Mark celebrated his 21st birthday recently. His friends threw him a huge party. To thank his friends for this gesture, he made a collage of his birthday pictures. Write a program for Mark to send the picture file as an attachment to the following addresses:

   - sammy@mymail.com
   - monica@yourmail.com
   - antony@yourmail.com
   - martha@mymail.com
   - angela@mymail.com

Onlinevarsity

LEARN @ YOUR COMFORT

Session 24

OOP Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain the OOP concepts.*
- *Explain inheritance.*
- *Explain the use of classes.*
- *Describe the use of constructors.*
- *Explain the use of objects.*

24.1 Introduction

Object-Oriented Programming (OOP) is a term used to characterize a programming language. It is one of the most widely accepted paradigm for programming languages as compared to others because of its flexibility.

OOP uses three basic concepts. These are classes, objects, and methods. Additionally, inheritance, abstraction, polymorphism, event handling, and encapsulation are concepts in object-oriented programming languages.

In OOP, a program is a collection of distinct objects. These objects are self-contained collection of data structures and functions that interact with each other. C++ and Java are the common examples of object-oriented programming languages.

In this session, you will learn to create a class in PHP and derive a new class from a base class. In addition, you will also learn to create a constructor for a class.

24.2 Object-Oriented Programming

In object-oriented programming, you define both the data type of the data structure and the type of functions to be performed on the data structure. Objects are the units of execution in an object-oriented system. It is a combination of data and functions in a single unit.

Session 24

The basic concepts of an object-oriented programming are as follows:

- **Object** - Consists of data structures and functions for manipulating the data. Data structure refers to the type of data while function refers to the operation applied to the data structures. An object is a self-contained run-time entity. An analysis of the programming problem is done in terms of objects and nature of communication between them. During execution, the objects interact with each other by sending messages.

- **Class** - Contains a collection of similar types of objects. It has its own properties. Once a class is specified, a number of objects can be created that belong to this category.

- **Abstraction** - Defines the process of selecting the common features from different functions and objects. The functions those perform same actions can be connected into a single function using abstraction.

- **Encapsulation** - Specifies the process of combining data and objects into a single unit. The data cannot be accessed directly; it can be accessed only through the functions present inside the unit. It enables data encryption.

- **Polymorphism** - Defines the use of a single function or an operator in different ways. The behavior of that function will depend on the type of the data used in it.

- **Inheritance** - Specifies the process of creating a new class from the existing one. The new class that is formed is called the derived class while the existing class from which the new class is derived is called the base class.

24.3 Inheritance

Inheritance means creating a new class with the help of a base class. Inheritance therefore, is the process in which objects of one class derive the properties of objects of another class. For example, a car is a part of the sedan, or hatchback class, which is again a part of the class Vehicles.

A base class is called the parent class and the derived class is called the child class. A class is an object that contains variables and functions of different data types. The properties, such as variables, functions, and methods of the base class are transferred to the newly derived class. You have to use the `extends` keyword for inheriting a new class from the base class. A derived class can also have its own variables and functions.

The different types of inheritances are as follows:

- **Single Inheritance** - Contains only one base class. The derived class inherits the properties of the base class. In single inheritance, the derived class works with properties derived from the base class along with its own properties.

- **Multiple Inheritance** - Contains more than one base class. The derived class inherits the properties of all base classes. A class derived from this inheritance works with the derived properties along with its own properties.

- **Hierarchical Inheritance** - Contains one base class. In hierarchical inheritance, the properties of a single base class can be used multiple times in the sub-multiples. The derived class contains its own properties. It also uses the derived properties of all the base classes.

- **Multilevel Inheritance** - Contains one base class. The base class of the derived class can be a class derived from another base class. The properties of a base class are inherited to another base class and the derived class can become a base class for another derived class.

- **Hybrid Inheritance** - Uses the combination of two or more inheritances. This inheritance is normally a combination of multiple and multilevel inheritance.

Note: In PHP, multiple inheritance is not supported.

24.4 Creating a Class

A class is a collection of variables and functions that operate on data. A class can be inherited from a base class to create a new derived class. The new derived class uses all the properties of the base class including its own properties.

The syntax for declaring a class is as follows:

Syntax:

```
class class_name
{
    function function_name($var1, $var2)
    {
        return $var1 + $var2;
    }
}
```

where,

`class` - defines the keyword used to declare a class.

`class_name` - specifies the class name.

`function` - specifies the keyword to define a function.

Session 24

`var1, var2` - specifies the variable name.

`function_name()` - specifies the function name.

`return` - returns the value after performing the specified mathematical function.

The definition for a class is included within curly braces. The variables defined in the class are local to the class. A class can also use global variables. The variable inside a class is declared with the keyword `var`. The functions inside a class may use its own local variables or may use the class variables.

For example, to create a class named `empdetail`, consider the code as shown in Code Snippet 1.

Code Snippet 1:

```
<?php
class empdetail
{
    var $empid;
    var $empname;
    var $empcity;
    var $empdept;
    var $empdesign;
    function enteremp($id, $name, $city)
    {
        $this->empid=$id;
        $this->empname=$name;
        $this->empcity=$city;
    }
    function enterdet($dept, $design)
    {
        $this->empdept=dept;
        $this->empdesign=design;
    }
}
?>
```

Save this file with an extension `.inc`. In the code, the `enteremp()` and `enterdet()` functions are user defined. These functions are defined in the `empdetail` class, and are used for entering data of an employee. The `enteremp()` function accepts the employee id, employee name, and city. The `enterdet()` function accepts the employee department and the designation.

Session 24

The `filename.inc` file is included in the **`filename.php`** file with the help of the `include` keyword. This is done in order to create an instance for the classes those are defined in the `filename.inc` file. The `include` keyword enables to incorporate any type of file with the main file.

The syntax to include a file in the program is as follows:

Syntax:

```
include "filename.ext";
```

Assuming that the code in Code Snippet 1 is saved as `empdetail.inc`, now use the code in Code Snippet 2 to invoke it.

Code Snippet 2:

```
<?php
include "empdetail.inc";
echo "The Employee details: <BR><BR>";
$empdet = new empdetail();
$empdet->enteremp(101, "John Williams", "Miami");
echo $empdet->empid,"<BR>";
echo $empdet->empname,"<BR>";
echo $empdet->empcity;
?>
```

Save this file as **`emp.php`** and open it in Mozilla Firefox Web Browser. The output of this code is shown in figure 24.1.

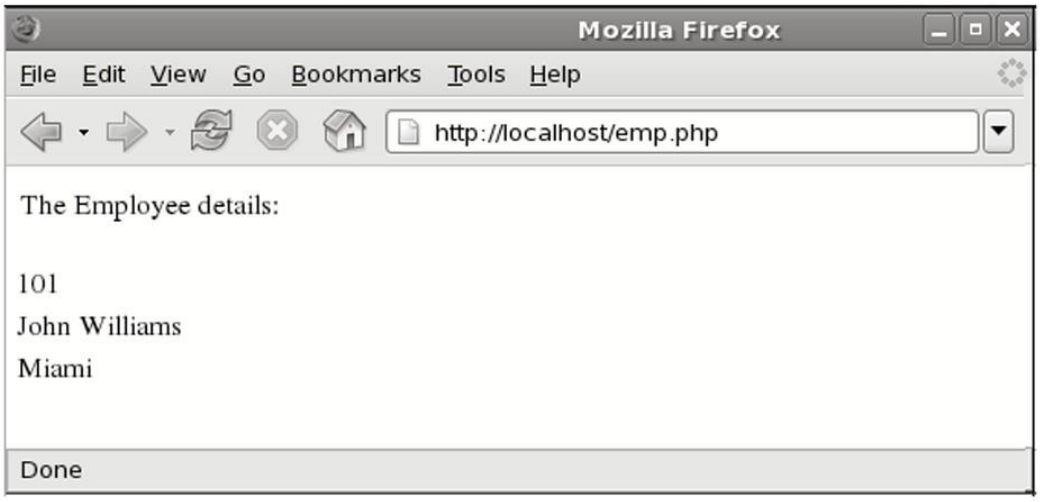


Figure 24.1: Creating a Class

Session 24

A new class is inherited from an existing class. The new class uses the properties of the parent class along with its own properties.

The syntax for inheriting a new class is as follows:

Syntax:

```
class new_class extends class_name
{
    var class_variable;
    function function_name()
    {
        . . .
    }
}
```

where,

`new_class` - specifies the name of the derived class.

`extends` - defines the keyword used to derive a new class from the base class.

`class_name` - specifies the name of the class from which the new class is to be derived.

For example, to derive the `net_salary` class from the `salary` class in PHP, consider the code as shown in Code Snippet 3 in a file named **`salary.inc`**.

Code Snippet 3:

```
<?php
class salary
{
    public  $hra;
    public  $ta;
    public  $tax;
    public function hra_calc($basic)
    {
        $hra = $basic * 0.25;
        return $hra;
    }
```

```
    public function travelallow_calc($basic)
    {
        $ta = $basic * 0.08;
        return $ta;
    }
    public function tax_calc($basic)
    {
        $tax = $basic * 0.05;
        return $tax;
    }
}
class net_salary extends salary
{
    function net($basic)
    {
        $hra = $this->hra_calc($basic);
        $ta = $this->travelallow_calc($basic);
        $tax = $this->tax_calc($basic);
        return $basic + ($hra + $ta)-$tax;
    }
}
?>
```

In the code, the `net_salary` class is derived from the base class, `salary`. The `net_salary` class uses the functions defined in the `salary` class along with its own functions.

The `hra_calc()`, `travelallow_calc()`, and `tax_calc()` functions are the user-defined functions. The `hra_calc()` function calculates HRA from the basic salary of the employee. The `travelallow_calc()` function calculates traveling allowance from the basic salary of the employee. The `tax_calc()` function calculates the tax from the basic salary of the employee. The `net()` function calculates the net salary of the employee.

This class is used in a file named **`salarydemo.php`** as shown in Code Snippet 4.

Code Snippet 4:

```
<?php
include "salary.inc";
echo "The Salary details of the employee: <BR><BR>";
$sal = new net_salary();
```

```
echo $sal->net(372500);
?>
```

The output is shown in figure 24.2.

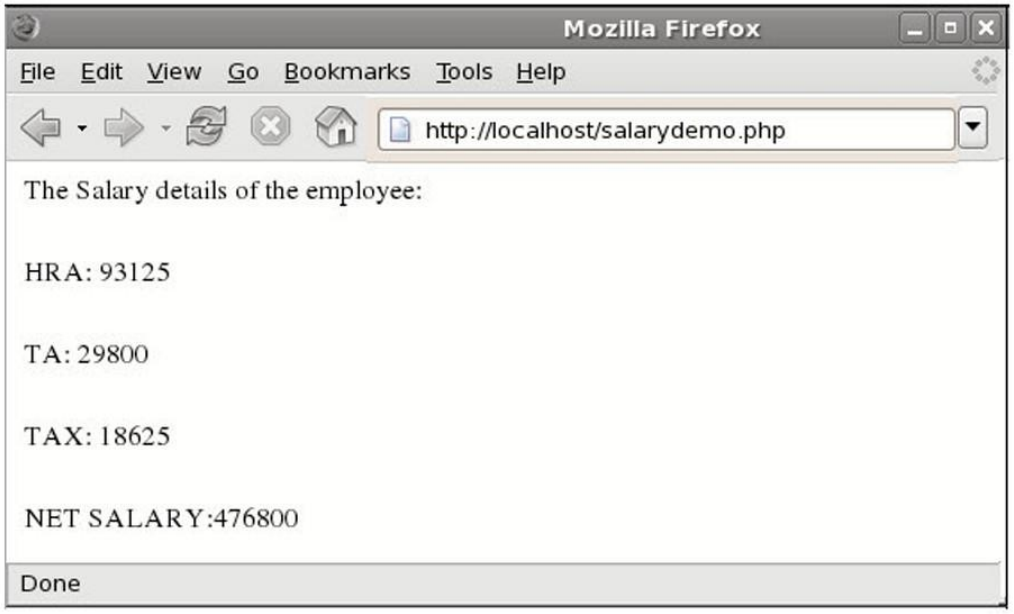


Figure 24.2: Demonstrating Inheritance

24.5 Creating a Constructor

A constructor is a special function that is a member of the class. This function has the same name as that of its class name. A constructor is called in the main program by using the `new` operator. You can use the constructor to initialize objects of the class of which it is a member. The constructor function is invoked whenever an object of the class is created.

When a constructor is declared, it provides a value to all the objects that are created in the class. This process is known as initialization. This allots a specific amount of memory to each object.

Session 24

The syntax for creating a constructor is as follows:

Syntax:

```
class class_name
{
    var class_variable;
    function constructor_name()
}
```

where,

`constructor_name` - specifies the name of the constructor.

For example, to create a constructor for the class named `empdetail`, consider the code as shown in Code Snippet 5, in a script named `emp_constructor.inc`.

Code Snippet 5:

```
class empdetail
{
    var $empid;
    var $empname;
    var $empcity;
    var $empdept;
    function empdetail($id, $name, $city, $dept)
    {
        $this->empid=$id;
        $this->empname=$name;
        $this->empcity=$city;
        $this->empdept=$dept;
    }
}
?>
```

Once the constructor is defined, call the constructor of the class using the `new` operator. Before calling the constructor of the class, include the file that contains the class.

Session 24

To call the constructor of the `empdetail` class, consider the code as shown in Code Snippet 6, in a script named **`emp_constructor.php`**.

Code Snippet 6:

```
<?php
include('emp_constructor.inc');
echo "The Employee details: <BR><BR>";
$empdet = new empdetail(101, "John Williams", "Miami", "Accounts");
echo $empdet->empid,"<BR>";
echo $empdet->empname,"<BR>";
echo $empdet->empcity,"<BR>";
echo $empdet->empdept;
?>
```

The `new` operator creates a `new` object of the `empdetail` class. It calls the functions defined in the `empdetail` class. Figure 24.3 shows the output of the code.

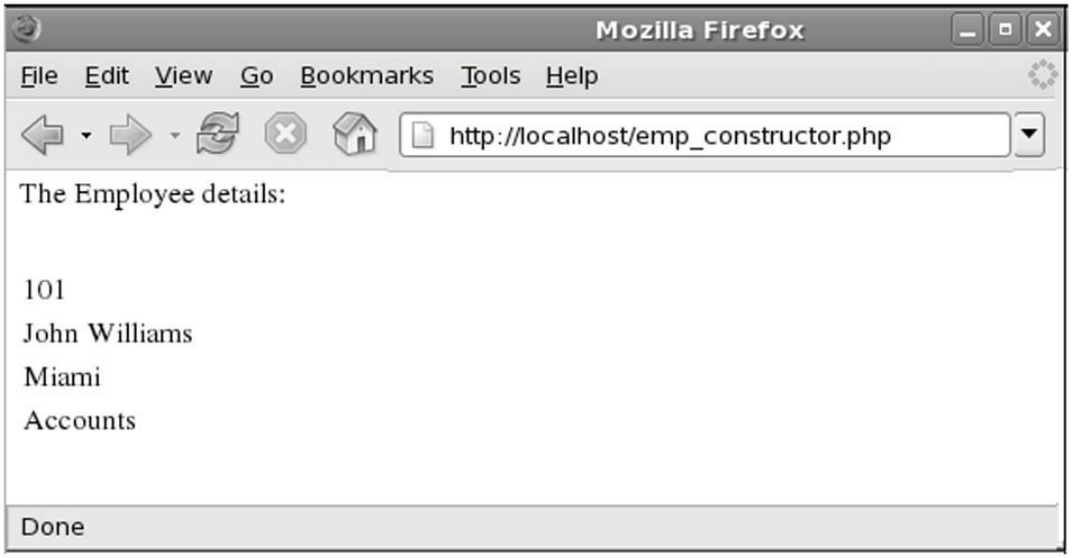


Figure 24.3: Using a Constructor

Session 24

For example, to call a constructor from the derived class, consider the code as shown in Code Snippet 7, in a script named **stringtest.php**.

Code Snippet 7:

```
<?php
class string
{
    function string()
    {
        echo "This function is a constructor.";
    }
    function stringdisp()
    {
        echo "This is function.";
    }
}
class display extends string
{
    function display()
    {
        echo "This is a constructor of a new class.";
    }
}
$disp1 = new string;
echo "<BR><BR>";
$disp = new display;
?>
```

In the code, the `display` class is inherited from the `string` class. As a result, the `display` class uses all the properties of the `string` class. Here, the `string()` function of the `string` class is the constructor. As the `display` class is inherited from the `string` class, therefore the `display` class will behave as a constructor for the `string` class.

The output is shown in figure 24.4.

Figure 24.4: Calling Constructor of Base Class from Derived Class

24.6 Creating and Using Objects

An object is a self-contained entity that includes both data and procedures to manipulate the data. You use objects in programming because an object makes the codes easier to understand and can be used more than once. It maintains the codes of the program. An object is an instance of a class. It provides a reference to the class. An object can be manipulated as required. The `new` operator can be used to initialize the object. There can be more than one object to one class.

The syntax for creating an object is as follows:

Syntax:

```
$object_name = new class_name;
```

where,

`object_name` - specifies the name of the object.

`new` - initializes the object.

`class_name` - specifies the class name.

Session 24

For example, to create an object for the class, `usermail`, consider the code as shown in Code Snippet 8, in a script named **`usermail.php`**.

Code Snippet 8:

```
<?php
class usermail
{
    var $username = "john";
    var $password = "abc123";
    function dispuser ()
    {
        echo $this->username,"<BR><BR>";
        echo $this->password, "<BR><BR>";
    }
}
class userdetails extends usermail
{
    var $secretquery = "Favorite food";
    var $answer = "Chinese";
    function dispdetail()
    {
        echo "<BR><BR>";
        echo $this->secretquery, "<BR><BR>";
        echo $this->answer, "<BR><BR>";
    }
}
$mail = new userdetails;
$mail1 = new usermail;
$disp1 = $mail->dispdetail();
$disp2 = $mail1->dispuser();
?>
```

In the code, the `$mail` object is created for the class, `userdetails`. The `userdetails` class is an extended class of the `usermail` class.

Figure 24.5 shows the output of the code.

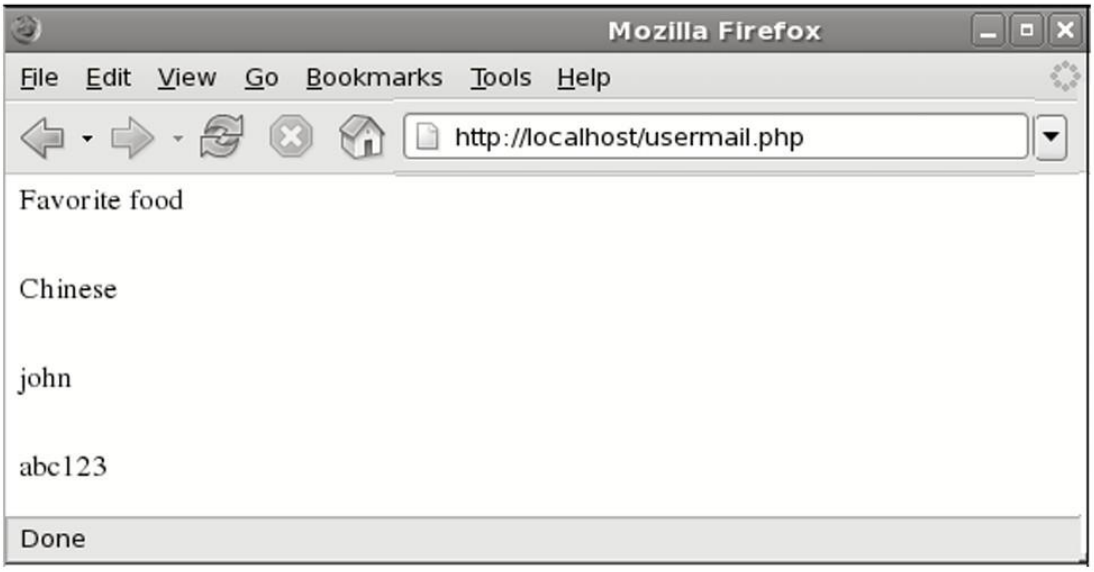


Figure 24.5: Creating and Using Objects

Summary

- Object-oriented programming defines the data type of the data structure and the type of functions that can be performed on the data structure.
- The main concepts of an OOP are objects, class, abstraction, encapsulation, polymorphism, and inheritance.
- In hierarchical inheritance, the properties of a single base class can be used multiple times in multiple subclasses.
- A class is a collection of variables and functions that operate on that data. A class can be inherited from a base class to create a new derived class.
- A constructor is a special function that has the same name as that of its class name, and is called in the main program by using the new operator.
- An object is a self-contained entity that consists of both data and procedures to manipulate the data. It is an instance of a class and can be manipulated as needed.

Session 24

Check Your Progress

1. _____ is the process of selecting the common features from different functions and objects.

   a. Inheritance

   b. Polymorphism

   c. Encapsulation

   d. Abstraction

2. The process of using a single function with its operators and variables in different ways is termed as _____.

   a. Abstraction

   b. Encapsulation

   c. Polymorphism

   d. Inheritance

3. _____ inheritance is a combination of multiple and multilevel inheritances.

   a. Hybrid

   b. Single

   c. Hierarchical

   d. Derived

4. The process of declaring a constructor is termed as _____.

   a. declaration

   b. encapsulation

   c. initialization

   d. abstraction

Check Your Progress

5. Which of the following is a self-contained entity?

   a. Object

   b. Constructor

   c. Class

   d. Function

6. An object is a/an _____ of a class.

   a. entity

   b. reference

   c. instance

   d. constructor

7. Which of the following makes the codes easier to understand and can be used repeatedly?

   a. reference

   b. object

   c. instance

   d. constructor

8. Which type of inheritance contains more than one base class?

   a. multilevel

   b. multiple

   c. hybrid

   d. hierarchical

Session 24

OOP Concepts

Answers

1. a
2. c
3. a
4. c
5. a
6. c
7. b
8. b

Session 25 OOP Concepts (Lab)

Objectives

At the end of this session, the student will be able to:

- *Create a class.*
- *Create a derived class.*
- *Create a constructor.*
- *Create an object of a class.*

The steps given in the session are detailed, comprehensive, and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. You must follow the steps carefully.

Part I - 120 Minutes

Creating a Derived Class

In object-oriented programming, you not only define the data type of the data structure but also the type of functions that can be performed on the data structure. It combines the data and functions into a single unit. Such a unit is called an object. A function of an object can be used to access the data.

A class is a collection of variables and functions that operate on data. A class can be inherited from a base class to create a new derived class. A new derived class uses all the properties of the base class including its own properties.

For example, to create a calculator, perform the following steps:

1. **Open a new file in the gedit text editor.**

2. **Enter the following code to create a form for the calculator:**

```
<HTML>
<BODY>
<FORM method = GET action = "calculate.php">
<B>Calculator</B><br>
First Number:     
<INPUT type="TEXT" name="num1">
```

Session 25

OOP Concepts (Lab)

Lab Guide

```
<br><br>
Second Number:
<INPUT type="TEXT" name="num2">
<br><br>
Calculate: <br>
<INPUT type= "radio" name= "add" value= "Addition">Addition  
<INPUT type= "radio" name= "sub" value= "Subtraction">
Subtraction 
<INPUT type= "radio" name= "mul" value= "Multiplication">
Multiplication
<INPUT type= "radio" name= "div" value= "Division">Division<br>
<INPUT type= "radio" name= "sin" value= "Sine">Sine  
<INPUT type= "radio" name= "cos" value= "Cosine">Cosine  
<INPUT type= "radio" name= "log" value= "Logarithm">Logarithm <br><br>
<INPUT type="SUBMIT" name="submit" Value = "Submit">
</FORM>
</BODY>
</HTML>
```

3. **Save the file as calculate.html under the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code to create classes that contain the functions to perform mathematical operations:**

```
<?php
class calci
{
   function addition($first, $second)
   {
      return $first + $second;
   }
   function subtraction($first, $second)
   {
      return $first - $second;
   }
   function multiply($first, $second)
   {
      return $first * $second;
   }
```

```
        function division($first, $second)
        {
            return $first / $second;
        }
    }
    class calculator extends calci
    {
        function sinvalue($first)
        {
            return sin($first);
        }
        function cosvalue($first)
        {
            return cos($first);
        }
        function logarithm($first)
        {
            return log($first);
        }
    }
    ?>
```

6. **Save the file as calculator.inc under the /usr/local/apache2/htdocs directory.**

7. **Open a new file in the gedit text editor.**

8. **Enter the following code for calculating the values using the class:**

```
<?php
error_reporting(E_ALL ^ (E_NOTICE | E_WARNING));
$add = "";
$sub = "";
$mul = "";
$div = "";
$sin = "";
$cos = "";
$log = "";
$num1 = "";
$num2 = "";
$first = "";
```

Session 25


```
$second = "";
include('calculator.inc');
$first = $_GET['num1'];
$second = $_GET['num2'];
$n1 = $_GET['add'];
$n2 = $_GET['sub'];
$n3 = $_GET['mul'];
$n4 = $_GET['div'];
$n5 = $_GET['sin'];
$n6 = $_GET['cos'];
$n7 = $_GET['log'];
$calculate = new calculator();
$calculate1 = new calci();
if($n1)
{
   $add = $calculate1->addition($first, $second);
   echo "The sum of $first & $second is: $add";
}
elseif($n2)
{
   $sub = $calculate1->subtraction($first, $second);
   echo "<br><br>";
   echo "The difference between $first & $second is: $sub";
}
elseif($n3)
{
   $prod = $calculate1->multiply($first, $second);
   echo "<br><br>";
   echo "The product of $first & $second is: $prod";
}
elseif($n4)
{
   $div = $calculate1->division($first, $second);
   echo "<br><br>";
   echo "The quotient for $first & $second is: $div";
}
```

Session 25


```
elseif($n5)
{
    $sine = $calculate->sinvalue($first);
    echo "<br><br>";
    echo "The sine value of $first is: $sine";
}
elseif($n6)
{
    $cosine = $calculate->cosvalue($first);
    echo "<br><br>";
    echo "The cosine value of $first is: $cosine";
}
elseif($n7)
{
    $logvalue = $calculate->logarithm($first);
    echo "<br><br>";
    echo "The log of $first is: $logvalue";
}
echo "<br><br><a href=calculate.html>Go Back</a>";
?>
```

In this example, **`$calculate`** and **`$calculate1`** are the instances of the **`calculator`** and **`calci`** classes. The functions defined in these classes are called by using the class instances with the help of the '->' sign.

9. **Save the file as calculate.php under the /usr/local/apache2/htdocs directory.**

10. **Open the Mozilla Firefox Web browser.**

Session 25

11. **Enter http://localhost/calculate.html in the Address bar and press Enter.**

    Figure 25.1 displays the output of the script.

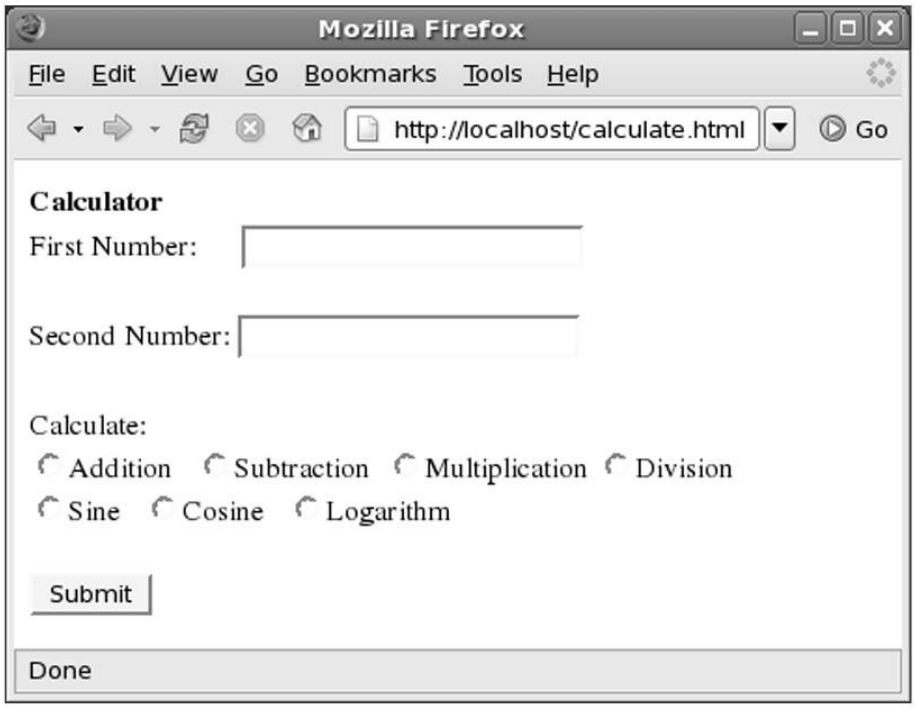


Figure 25.1: Calculator Form

12. **Enter 48 in the First Number box.**

13. **Enter 6 in the Second Number box.**

14. **Select Division.**

Session 25

15. **Click Submit.**

    Figure 25.2 displays the output of the script.

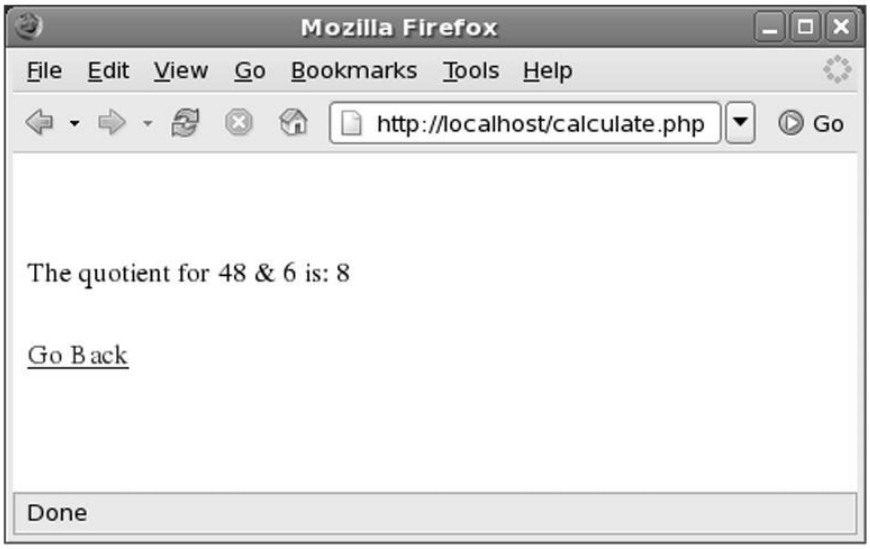


Figure 25.2: Displaying the Result for Division

16. **Click the Go Back hyperlink.**

17. **Enter 47 in the First Number box.**

18. **Select Cosine.**

19. **Click Submit.**

    Figure 25.3 displays the output of the script.

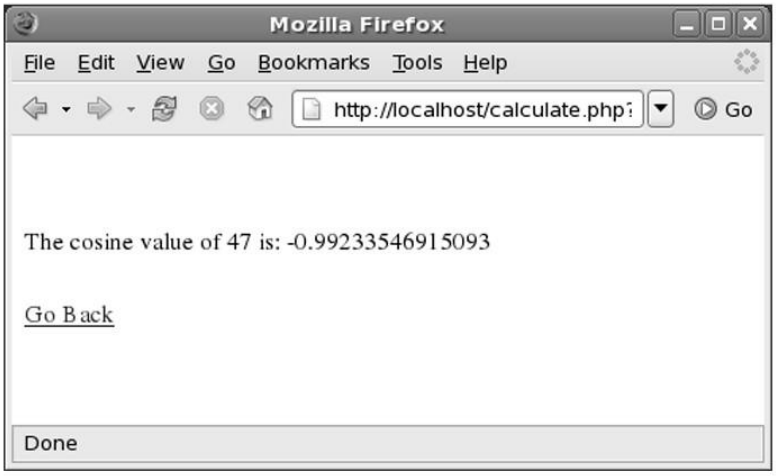


Figure 25.3: Displaying the Cosine Value

Creating a Constructor

A constructor is a special function that is a member of the class. This function has the same name as that of its class name. A constructor is called in the main program by using the new operator. You use the constructor to initialize objects of the class of which it is a member. The constructor function is invoked whenever an object of the class is created. When a constructor is declared, it provides a value to all the objects that are created in the class. This process is known as initialization. This allots a specific amount of memory to each object.

For example, to display the odd numbers in the reverse order using the constructor, perform the following steps:

1. **Open the gedit text editor.**

2. **Enter the following code for creating a constructor:**

```
<?php
class oddnum
{
  function oddnum()
    {
       $i=5;
```

Session 25

OOP Concepts (Lab)


```
            do
            {
                $num = $i*2-1;
                echo "<br>$num";
                $i--;
            }while($i>=1);
        }
    }
    ?>
```

In this example, **oddnum** is the user-defined class. This class has a function named, **oddnum**. The **oddnum** function is the constructor of the **oddnum** class.

3. **Save the file as construct.inc under the /usr/local/apache2/htdocs directory.**

4. **Open a new file in the gedit text editor.**

5. **Enter the following code for displaying the odd numbers in reverse order:**

```
<HTML>
<BODY>
<?php
include ("construct.inc");
echo "Displaying odd numbers in reverse order<br>";
$x = new oddnum();
?>
</BODY>
</HTML>
```

The `new` operator is used for calling the **oddnum** constructor. It is assigned to a variable **$x**. The variable **$x** is created as an object for the **oddnum** class.

6. **Save the file as constructor.php under the /usr/local/apache2/htdocs directory.**

7. **Open the Mozilla Firefox Web browser.**

Lab Guide

8. **Enter http://localhost/constructor.php in the Address bar and press Enter.**

   Figure 25.4 displays the output of the script.

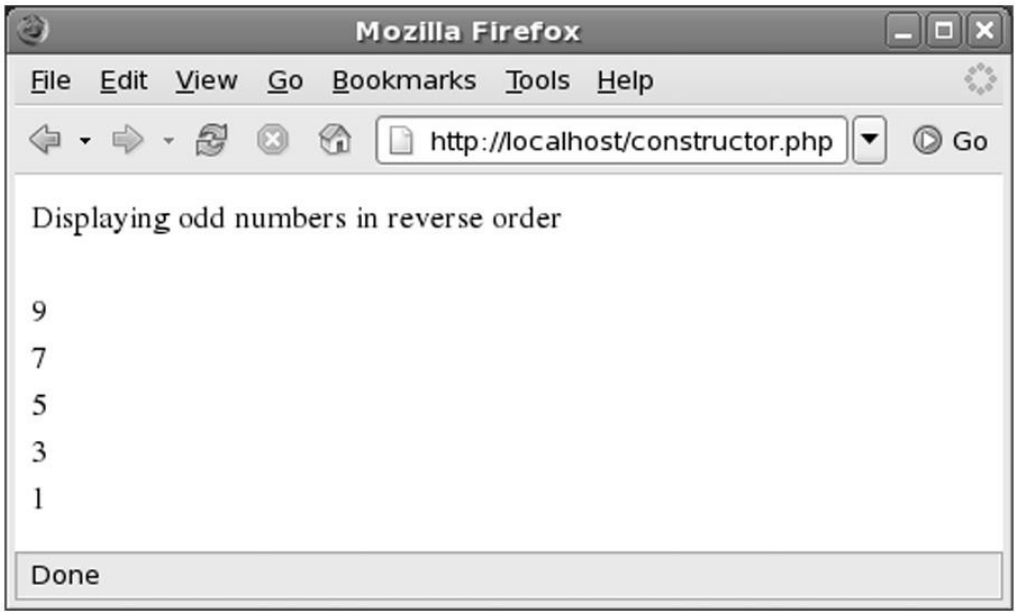


Figure 25.4: Displaying Odd Numbers in Reverse Order

Session 25

OOP Concepts (Lab)

Do It Yourself

1. Create a class named `Person` with properties such as name, age, and nationality. In the `Person` class, create functions named `getDetails` and `displayDetails` to accept and display the details of the person. Derive a class named `Student` from `Person` class. In the derived class, create properties specific to student such as roll no, class, and marks and add functions to accept and display the student details.

Session 26 Generator Delegation and Throwable Interface

Concepts

Objectives

At the end of this session, the student will be able to:

- *Explain generators and generator return expressions.*
- *Describe Generator Delegation.*
- *Explain exception handling and changes in exceptions in PHP 7.*
- *Explain the use of* `E_ERROR` *and* `E_RECOVERABLE_ERROR`*.*
- *Explain the* `Throwable` *Interface.*

26.1 Generators and Generator Return Expressions

Generator return expression makes it easier for you to implement simple iterators, without any overhead on the system.

26.1.1 Generators

As mentioned in earlier session 3, a generator function is a means to implement an iterator and is similar to a regular function. The difference between a normal function and a generator is that, a function always returns a single value whereas a generator can return multiple values from the function. Instead of computing values at once, a generator would return/yield the values on demand. This will help to alleviate memory limitations. Inside a generator function, programmers need to use the `yield` keyword that returns data.

An example of a generator is shown in Code Snippet 1. It prints a range of values.

Code Snippet 1:

```
<?php
function print_range_of_values($min_val, $max_val)
{
 for ($i= $min_val; $i <= $max_val; $i++)
 {
 yield $i;
 }
 }
 foreach(print_range_of_values(1,10) as $key=>$value)
 {
 echo "$key => $value", PHP_EOL;
 }
?>
```

Session 26

The output of Code Snippet 1 is shown in figure 26.1.

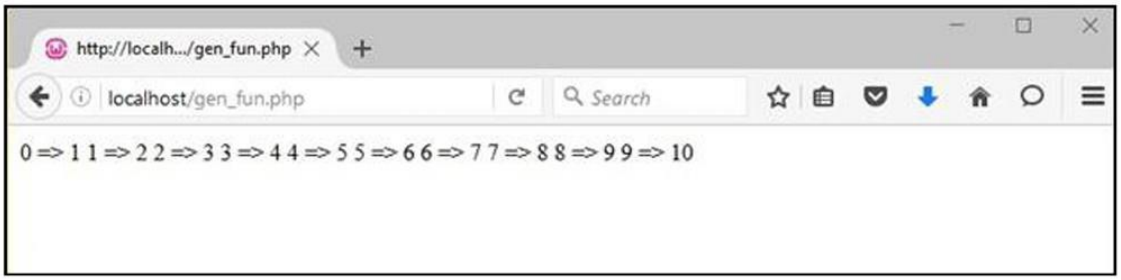


Figure 26.1: Output of Code Snippet 1

A generator allows programmers to write code that uses the `foreach` keyword to iterate over a set of data. This way, a programmer need not build an array in memory. If a programmer builds an array in the memory of a program, there is a possibility of exceeding the memory limit. It may also require a considerable amount of time to process and generate.

26.1.2 Generator Return Expressions

Generator return expressions is a new feature in PHP 7 that builds upon the generator feature introduced in PHP 5.5. Generator return expressions allow you to 'return' a value upon successful completion of a generator using the `return` keyword and the `Generator::getReturn()` method. In earlier versions of PHP, if you attempted to `return` anything from a generator through the return keyword, it would result in an error.

Programmers can write a generator function that returns (yields) as many values as possible by iterating within the function. Let us understand this with an example, as shown in Code Snippet 2.

Code Snippet 2 simulates the `range()` function, which generates random numbers between the starting number and maximum number that you specify.

Code Snippet 2:

```
<?php
function Sim_range($start_num, $max_limit, $increment_by = 1) {
 if ($start_num < $max_limit) {
 if ($increment_by <= 0) {
 throw new LogicException('Step must be +ve');
 }
 for ($x = $start_num; $x <= $max_limit; $x += $increment_by) {
 yield $x;
```

Session 26


```
}
 } else {
 if ($increment_by >= 0) {
 throw new LogicException('Step must be -ve');
 }
 for ($x = $start_num; $x >= $max_limit; $x += $increment_by) {
 yield $x;
  }
 }
}
echo 'Single digit odd numbers from range():<br> ';
foreach (range(1, 11, 2) as $num) {
 echo "$num ";
}
echo "<br>";

echo 'Single digit odd numbers from Sim_range():<br>';
foreach (Sim_range(1, 11, 2) as $num) {
 echo "$num ";
}
?>
```

Code Snippet 2 displays numbers between the start number and the maximum limit. The numbers are incremented by 2. If the increment by value is less than 0, the program throws a logical error.

The `foreach` loop iterates through the generator function and it iterates through the `for` loop and returns/yields the value. A built-in function `range()` is available in PHP 7 and the function `Sim_range()` defined here simulates the built-in function.

The output of Code Snippet 2 is shown in figure 26.2.

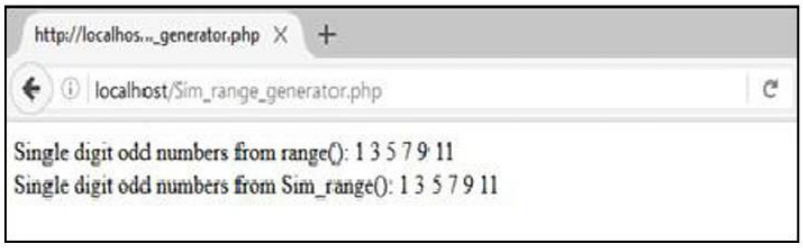


Figure 26.2: Output of Code Snippet 2

Session 26

Concepts

26.2 Generator Delegation

With the help of Generator Delegation, programmers can write a generator that can yield other generators, arrays, and traversable objects. The syntax is shown as follows:

Syntax:

```
yield from <thing_to_iterate>
```

Code Snippet 3 contains a generator function that, in turn, yields another generator function.

Code Snippet 3:

```
<?php
 function DisplayHello(){
 yield "Hello";
 yield " ";
 yield "World!";
 yield from DisplayGoodbye();
 }
 function DisplayGoodbye() {
 yield "Goodbye";
 yield " ";
 yield "Mars!";
 }
 $gen = DisplayHello();
 foreach ($gen as $value) {
 echo $value;
 echo '</br>';
 }
?>
```

Code Snippet 3 has two generator functions, `DisplayHello()` and `DisplayGoodbye()`. The generator functions yield (return) the values 'Hello', a blank line, and then 'World'. `DisplayHello()` also yields a Generator Delegation which calls another generator function `DisplayGoodbye()` and yields values from that. The values 'Hello', a blank line, and 'World' are yielded from generator `DisplayHello()` and values 'Goodbye', a blank line, and 'Mars!' are yielded from another generator `DisplayGoodbye()` within the generator function `DisplayHello()`.

The output of Code Snippet 3 is shown in figure 26.3.

Figure 26.3: Output of Code Snippet 3

26.3 Exceptions

PHP 5 and later versions support exceptions, which are a mechanism for error handling. Exceptions are supported by several object-oriented programming languages.

Exception handling code include:

- `try`: A function that uses exception handling must be enclosed in a `try` block. If exceptions do not occur, the code executes normally and exits successfully. However, if an exception is encountered, then the exception is 'caught' by the accompanying catch block, which may contain action to handle the exception.

- `catch`: A `catch` block normally handles the exception. The `catch` block takes an object as parameter that represents the exception information. Action statements to be executed in case an exception occurs are included in the `catch` block.

- `throw`: This is the way an exception is manually triggered. Each `throw` should have a corresponding `catch`.

- `finally`: From PHP 5.5 onwards, a `finally` block may optionally be included in the code after or instead of `catch` blocks. Code within the `finally` block will always be executed irrespective of whether an exception has been thrown and before normal execution continues. It is usually executed after the `try` and `catch` blocks.

Session 26

Code Snippet 4 demonstrates an example that uses `try-catch-finally` statements.

Code Snippet 4:

```
<?php
function computeDiv($num) {
 if (!$num) {
 throw new Exception('Division by zero.');
 }
 return 1/$num;
}

try {
echo computeDiv(0).'<br><br>';
echo 'Result of division';
} catch (Exception $e) {
 echo 'Caught exception: ', $e->getMessage(), '<br><br>';
} finally {
 echo "Now in finally block.<br><br>";
}

// Continue execution
echo "Program execution continues<br><br>";
?>
```

As the parameter passed to computeDiv is 0, an exception will be generated, hence, the statement '`Result of division`' is never displayed. The flow of control goes to the catch block upon encountering the exception. The finally block will be executed in any case, whether there was an exception or not.

The output will be as follows:

```
Caught exception: Division by zero.

Now in finally block.

Program execution continues
```

Concepts

26.4 Changes in PHP 7 for Exceptions and Errors

While executing a PHP program, when a runtime error occurs, an error message is displayed on the monitor and the program stops running if the error is fatal. If those runtime errors that are not fatal errors are handled properly, then the execution of program is successful and appropriate error processing statements would be executed.

In PHP 7, an exception is raised whenever a fatal error or a recoverable error occurs. The advantage of this is that you can 'catch' the exception in a graceful manner and display custom error messages. If an exception is not handled, it will still be considered as a fatal error. Note that only recoverable errors and fatal errors throw exceptions. For certain conditions such as 'out of memory', however, fatal errors would still exist. In such cases, the program would terminate its execution.

26.4.1 E_ERROR

This is a runtime error, which is fatal. This error is not recoverable. The script or program terminates prematurely whenever this type of error occurs. `E_ERROR` does not break backwards compatibility, which implies that every code that was working previously would continue to work in PHP 7.

Consider a program to demonstrate `E_ERROR`, which will not handle fatal errors. In Code Snippet 5, a function `my_function` is being called without being defined.

Code Snippet 5:

```
<?php
 error_reporting(E_ERROR);
 function handle_error($err_no, $err_str,$err_file,$err_line) {
 echo "<b>Error:</b> [$err_no] $err_str - $err_file:$err_line";
 echo "</br>";
 echo "Terminating PHP Script";
 die();
}
 //set error handler
 set_error_handler("handle_error", E_ERROR);
 //trigger error
 my_function();
?>
```

Concepts

In Code Snippet 5, `my_function` is not defined however is being called. It results in a fatal error. This error could not be handled even with `E_ERROR` as shown in Code Snippet 5.

The output of Code Snippet 5 is shown in figure 26.4.

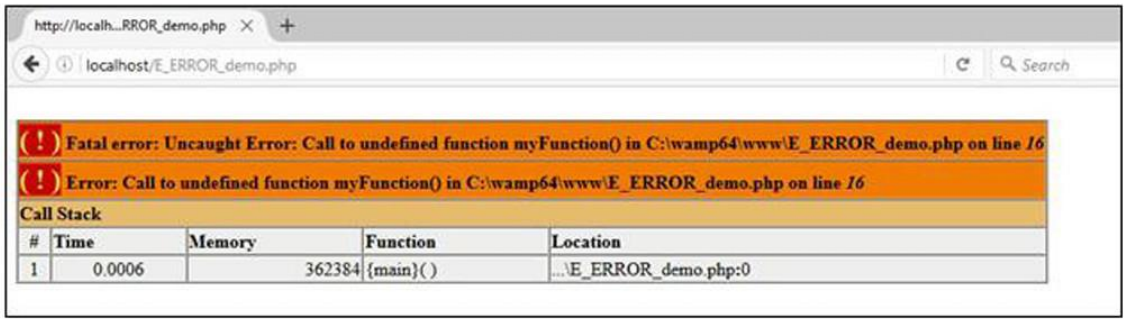


Figure 26.4: Output of Code Snippet 5

26.4.2 E_RECOVERABLE_ERROR

This error is similar to `E_ERROR`, however, can be handled by a user-defined exception. When `error_reporting` is mentioned as `E_RECOVERABLE_ERROR`, the exception can be handled. The program gracefully exits without reporting any error.

Let us write a program to demonstrate the `E_RECOVERABLE_ERROR`, which will handle fatal errors.

In Code Snippet 6, a function `myFunction` is not defined and it is being called. The error would not be reported to the user.

Code Snippet 6:

```
<?php
 error_reporting(E_RECOVERABLE_ERROR);
 function handle_error($errno, $errstr,$error_file,$error_line) {
 echo "<b>Error:</b> [$errno] $errstr - $error_file:$error_line";
 echo "</br>";
 echo "Terminating PHP Script";
 die();
}
 //set error handler
set_error_handler("handleError", E_RECOVERABLE_ERROR);
```

```
 //trigger error
 myFunction();
?>
```

Code Snippet 6 uses `error_reporting` as `E_RECOVERABLE_ERROR`. Here, we are calling an undefined function, `myFunction()`. Normally, this will cause a fatal error. With `E_RECOVERABLE_ERROR`, we can handle this error and the program can terminate appropriately. Hence, the output is a blank screen.

The output of Code Snippet 6 is shown in figure 26.5.

Figure 26.5: Output of Code Snippet 6

In PHP 7, all errors including fatal errors are engine exceptions. Engine Exceptions are a special type of exception where some fatal errors can also be handled. This is possible because an unhandled exception would result in a traditional fatal error. A few benefits of handling fatal errors as exceptions are as follows:

- We can handle fatal errors by using `try-catch` block.
- It makes easy to handle catchable fatal errors.
- It is easier to debug.

26.4.3 Throwable Interface

In PHP 7, both `Exception` and `Error` classes implement the new `Throwable` interface.

Previous versions of PHP did not include this interface and the root of the hierarchy was `\Exception`.

The hierarchy of classes and interfaces is shown here:

- Interface `Throwable`
  - `Exception` implements `Throwable`

- LogicException (extends Exception)
- `InvalidArgumentException` (extends `LogicException`)
- ...

- `Error` implements `Throwable` (Replaces `EngineException`)
  - `TypeError` extends `Error`
  - `ParseError` extends `Error`
  - ...

Using the `throw` statement, you could also throw objects that implement the `Throwable` interface.

Let us take an example of a program, in which we define a function `add` that has two integer type parameters namely, `$left_op` and `$right_op`. The function `add` returns the addition of the two parameters. When calling this function with string values 'two' and 'three', the program throws a fatal error, the exception is not caught, and an error will be shown in the output. Code Snippet 7 shows the program.

Code Snippet 7:

```
<?php
function add(int $left_op, int $right_op) {
 return $left_op + $right_op;
}
 try {
 echo add('two', 'three');
} catch (Exception $e) {
 // Handle or log exception.
 echo "Error occurred in the program due to parameter datatype mismatch";
}
?>
```

Session 26

The output of Code Snippet 7 is shown in figure 26.6.

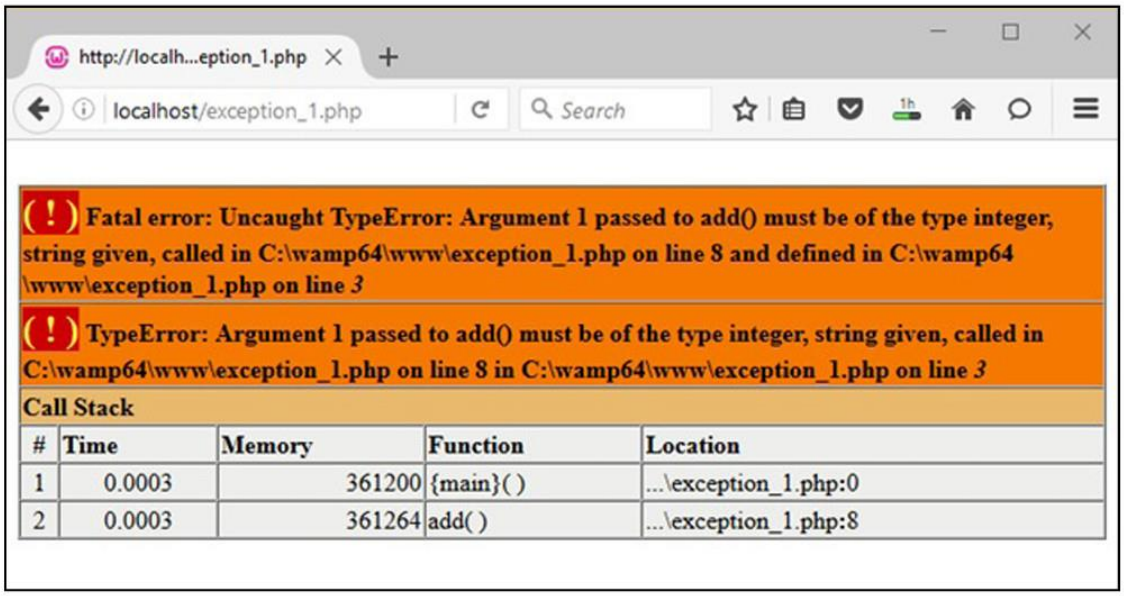


Figure 26.6: Output of Code Snippet 7

Code Snippet 7 will not catch the exception because we are passing string values instead of integers. Datatype mismatch results in a `TypeError`, which is a kind of `Error` and not `Exception`. Hence, the error could not be handled.

To catch the exception, Code Snippet 7 can be rewritten as shown in Code Snippet 8. Here, along with code to catch an `Exception`, the program also includes code to catch an `Error`.

Code Snippet 8:

```
<?php
function add(int $left_op, int $right_op) {
 return $left_op + $right_op;
}
 try {
 echo add('two', 'three');
} catch (Exception $e) {
 // Handle or log exception.
 echo "Exception occurred in the program";
}
catch (Error $e) { // Log error and end gracefully
echo "Error occurred in the program";
}
?>
```

Session 26

The output of Code Snippet 8 is shown in figure 26.7.

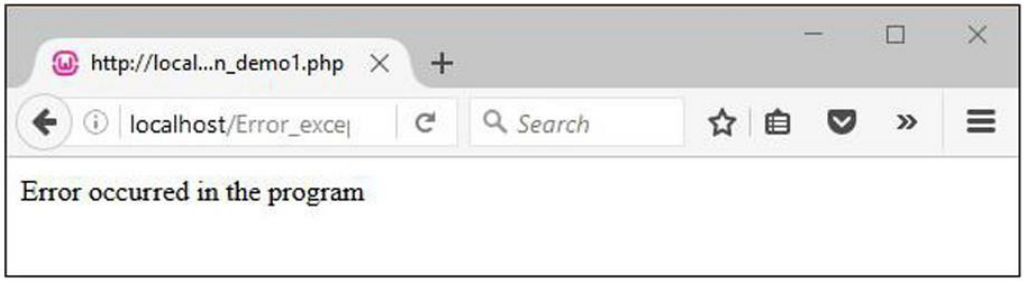


Figure 26.7: Output of Code Snippet 8

In this program, we call the function `add` with actual values 'two' and 'three' which are `string` type values. The function expects two `integer` type values; however, we have passed `string` values. On account of this, an error occurs and the control of the program goes to the `catch` section. In the `catch` block, we trap any `Error` instances that are encountered. `Error` is the base class for errors and implements the `Throwable` interface. The `echo` statement in the `catch` block is then executed. Hence, the output is shown as 'Error occurred in the program'. The program exits and the message is displayed to the user.

Session 26

Summary

- Generator return expressions is a new feature in PHP 7 that allows you to 'return' a value on completion of a generator using `return` keyword and `Generator::getReturn()` method.
- Generator Delegation promotes reusability and cleaner code.
- Generator Delegation allows getting/yielding values from other generators using `yield from` clause.
- PHP supports error and exception handling mechanisms that allow programs to continue or exit gracefully instead of exiting abruptly.
- In PHP 7, all errors including fatal errors are engine exceptions.
- `Exception` and `Error` classes implement the new `Throwable` interface, which is newly introduced in PHP 7.

Session 26

Check Your Progress

1. What is the output of the following program?

```
<?php
function DisplayWelcome () {
yield "Hello";
yield " ";
yield "World!";
return "Welcome Venus!";
}
$generator = DisplayWelcome();
foreach ($generator as $value) {
echo $value;
}
echo $generator->getReturn();
?>
```

a. Hello World! Welcome Venus!

b. Welcome Venus

c. Hello World

d. Error

Check Your Progress

2. What is the output of the following program?

```
<?php
        function Display() {
             yield "Hello";
             yield " ";
             yield "World!";
             yield from Welcome();
         }
        function Welcome() {
            yield "Welcome";
            yield " ";
            yield "Venus!";
        }
        $generator = Display();
        foreach ($generator as $value) {
                 echo $value;
         }
?>
```

a. Welcome Venus

b. Welcome Venus Hello World

c. Hello World

d. Hello World Welcome Venus

Session 26

Check Your Progress

3. What is the output of the following program?

```
<?php
try {
              non_existent_function();
 } catch (\Error $e) {
// handle error
         echo "There is no such function defined in the program";
}
?>
```

a. Fatal error

b. No output as the function is not defined

c. There is no such function defined in the program

d. Some output as the function is defined

4. Exceptions in PHP implement the _____ interface.

a. Throwable

b. ThrowError

c. ThrowException

d. BaseException

Session 26

Generator Delegation and Throwable Interface

Answers

1. a
2. d
3. c
4. a