

Programming for the Web with PHP

Programming for the Web with PHP

Trainer 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

Edition 1 - 2016



“ Any expansion is life
all contraction is death ”

For Aptech Center Use Only

Preface

The Trainer's Guide for **Programming for the Web with PHP** aims to teach application development for the Web using PHP. It begins with basics of PHP programming and then moves on to explore advanced concepts of PHP such as database handling, cookies, session management, and OOP fundamentals.

The faculty/trainer should teach the concepts in the theory class using the slides. This Trainer's Guide will provide guidance on the flow of the session and also provide tips and additional examples wherever necessary. The trainer can ask questions to make the session interactive and also to test the understanding of the students.

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

**“Learning how to learn is
life’s most important skill”**

For Aptech Center Use Only

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
7. Scalar Type Declarations
8. PHP Operators
10. Conditional Statements in PHP
12. Flow Control in PHP
14. Functions in PHP
16. Working with Arrays
18. Handling Databases with PHP
19. Working with Cookies
21. Session Management in PHP
22. Handling E-mail with PHP
24. OOP Concepts
26. Generator Delegation and Throwable Interface

Session 1 – Introduction to PHP

1.1 Pre-Class Activities

Prepare the background knowledge/summary to be discussed with students in the class. Familiarize yourself with the topics of this session in depth.

1.1.1 Objectives

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

- Explain the history of PHP
- Identify the need for PHP
- Explain PHP tools and setup
- Explain a simple PHP script
- Explain User Input/Output (I/O)
- Explain the use of PHP to generate HTTP headers
- Describe passing of variables using Universal Resource Locator (URL)

1.1.2 Teaching Skills

To teach this session successfully, you must know about the background and history of PHP, need for PHP, and PHP programming in general. You should also be aware of an overall view of scripting in PHP using the tools that are available for PHP.

You should teach the concepts in the theory class using the slides provided. For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students a brief overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *Explain the history of PHP*
- ◆ *Identify the need for PHP*
- ◆ *Explain PHP tools and setup*
- ◆ *Explain a simple PHP script*
- ◆ *Explain User Input/Output (I/O)*
- ◆ *Explain the use of PHP to generate HTTP headers*
- ◆ *Describe passing of variables using Universal Resource Locator (URL)*

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 2 of 41

Tell them that they will be introduced to PHP, learn the history, and explore the need for PHP. They will also be able to create a PHP script by learning about the tools and use of PHP. In addition, they will also understand how to pass variables using a Universal Resource Locator (URL).

1.2 In-Class Explanations

Slide 3

Let us introduce PHP.

Introduction

- ◆ PHP
 - ◆ Stands for Hypertext Preprocessor
 - ◆ Is an open source scripting language
 - ◆ Is used for developing dynamic Web pages
 - ◆ Is embedded in the HyperText Markup Language (HTML)
- ◆ PHP scripts are executed on the Web server

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 3 of 41

Use slide 3 to introduce PHP. Tell them that PHP is an open source scripting language and PHP scripting is done using the HTML tags. Also, tell the students that the PHP scripts are executed on the Web server. Once created, each page can be viewed in the browser by using local host on the Web server.



Key Points to Remember:

1. PHP Stands for Hypertext Preprocessor
2. PHP is an open source scripting language.

Slides 4 and 5

Let us understand the history of PHP.

History of PHP

1-2

- ◆ 1994
 - ◆ PHP was created by Rasmus Lerdorf
 - ◆ Later, incorporated with Form Interpreter (FI) to create PHP/FI
 - ◆ PHP/FI enables:
 - ◆ Communication with database
 - ◆ Development of dynamic Web application
- ◆ 1997
 - ◆ PHP/FI 2.0 version released
 - ◆ Lack of features led to the development of PHP 3.0
 - ◆ PHP 3.0 provided support for:
 - ◆ Object oriented syntax
 - ◆ Different databases
 - ◆ Protocols
 - ◆ Application Programming Interfaces (APIs)

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 4 of 41

History of PHP

2-2

- ◆ 2000
 - ❖ PHP 4.0 version released
 - ❖ Features supported in PHP 4.0 are as follows:
 - ❖ Multiple Web servers
 - ❖ HTTP session
 - ❖ Output buffering
 - ❖ Security for user inputs
- ◆ 2004
 - ❖ PHP 5.0 version released
- ◆ 2011
 - ❖ Current version PHP 5.3.6 released

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 5 of 41

Use slide 4 to explain the students about the history of PHP. You can also give them some more information about each version of PHP, so that the students can understand the differences and scope of current version of the PHP. Then using slide 5, continue explaining the history of PHP. Tell them the various upgradation done to the PHP versions in subsequent years and mention the latest version release using this slide.

Additional Information:

For more information on the history of PHP, visit the following links:

<http://scholar.lib.vt.edu/manuals/php3.0.6/intro-history.html>

<http://php.net/manual/en/history.php.php>

Slides 6 to 8

Let us learn more about PHP.

PHP  **1-3**

- ◆ Is a server-side scripting language
- ◆ Advantages are as follows:
 - ❖ Easy to learn, use, and implement
 - ❖ Freely available
 - ❖ Customizable
 - ❖ Executed on any Web server on any platform

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 6 of 41

PHP  **2-3**

- ◆ Uses of PHP are as follows:
 - ❖ **Application Control** – is used to control access logging for HTTP servers
 - ❖ **Database Access** – is used to read and write to any database using Structured Query Language (SQL) or Open Database Connectivity (ODBC)
 - ❖ **File Access** – is used for file and directory maintenance, generate files in Portable Document File (PDF) and HTML formats, and to process eXtensible Markup Language (XML) data

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 7 of 41

- ❖ **Graphics** – is used to create graphics, charts, and generate images in GIF and PNS formats
- ❖ **Server-Side Scripting** – is used to implement server-side scripting using PHP parser, Web server and Web browser
- ❖ **Command Line Scripting** – is used to execute scripts on Unix/Linux platforms
- ❖ **Desktop Applications** – is used to create GUI-based desktop applications

Using slide 6, explain to the students the advantages of the PHP. It is important to tell them the pros and cons of PHP, as there are several programming languages exist. You should make them understand what kind of applications PHP language is used to create. Using slide 7, explain the students about the uses of PHP. Using slide 6, you can compare other widely used languages such as Java or C++, and explain them the new features of PHP language. Explain them about the application control, database access, and file access. Using slide 8, continue explaining the uses of PHP. Explain them about graphics, server-side scripting, desktop applications, and command line scripting.

Slides 9 to 11

Let us learn about PHP tools.

PHP Tools  **1-3**

- ◆ Are text editors
- ◆ Are used for developing and designing Web pages
- ◆ Are implemented after installation of PHP

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 9 of 41

PHP Tools  **2-3**

- ◆ Are programming text editors that enable fast development of Web sites

Table lists tools used for developing dynamic Web sites

Tool	Description
PHPDebugger DBG	Enables step by step execution and debugging of a PHP script without changing the PHP code
ionCube Standalone PHP Encoder	Protects the PHP code and ensures security and runtime performance
Codelock	Enables you to encrypt both PHP and HTML code and protect Web pages
PHing	Is a PHP build system, which enables you to design your 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 you to create, debug, profile, deploy, and integrate PHP code

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 10 of 41

PHP Tools

3-3

Tool	Description
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
Simplewire PHP SMS Software Development Kit (SDK)	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
gedit	Is a GNOME based text editor for writing PHP scripts

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 11 of 41

Using slide 9, explain the students about PHP tools. First, give an introduction about the PHP tools with the points given in slide 9. Show the list of PHP tools along with their description, using slide 10. You can use the table to explain the students about the different tools. Using slide 11, continue explaining the PHP tools. After explaining the description, tell them the use of each tool that is available.

Slides 12 to 16

Let us understand installing PHP.

Installing PHP

1-5

- ◆ The installation of PHP requires:
 - ◆ A Web server
 - ◆ A database
- ◆ Web servers supported are as follows:
 - ◆ Internet Information Services (IIS)
 - ◆ Apache
 - ◆ Zeus
- ◆ Databases supported are as follows:
 - ◆ DB2
 - ◆ MSSQL
 - ◆ MySQL
 - ◆ Oracle
 - ◆ PostgreSQL

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 12 of 41

Installing PHP

2-5

- To install PHP, perform the following steps:

Step 1

Download the `php-5.3.6.tar.gz` file from
<http://www.php.net/downloads.php>

Step 2

Right-click `php-5.3.6.tar.gz` files and select Extract Here. The contents are extracted to the `php-5.3.6` folder

Step 3

Right-click `php-5.3.6` folder and select Open In Terminal

Installing PHP

3-5

Step 4

To configure the source code of PHP, enter the following at the command prompt:

```
./configure --with-apxs2=/usr/local/apache2/bin/apxs
```

Displays the following output:

```
root@localhost:~/php-5.3.6# ./configure --with-apxs2=/usr/local/apache2/bin/apxs
[...]
checking for Cygwin environment... (cached) no
checking for mingw32 environment... (cached) no
checking for grep... (cached) grep -E
checking for a sed that does not truncate output... (cached) /bin/sed
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking for gcc... (cached) gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-compiler... no
checking whether we are using GNU C... (cached) yes
checking whether gcc accepts -g... (cached) yes
checking how to run the C preprocessor... (cached) gcc -E
```

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 13 of 41

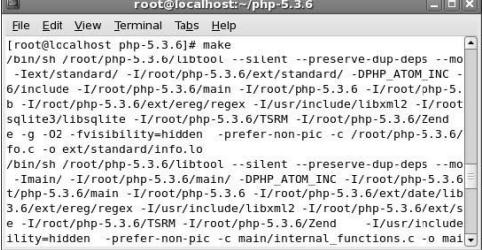
Installing PHP 4-5

Step 5

To build the compiled files, enter the following at the command prompt:

```
make
```

Displays the following output:



Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 15 of 41

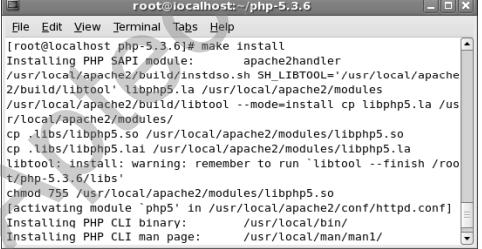
Installing PHP 5-5

Step 6

To install PHP, enter the following at the command prompt:

```
make install
```

Displays the following output:



Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 16 of 41

To start with PHP programs, the installation procedure plays an important role. Therefore, it is important to tell the students the process of installing PHP. Using slide 12, explain them the requirements for the installation procedure. Then, continue explaining the process in brief. Using slide 13, explain the students the steps to install the PHP. Using slide 14, you show them the output that is obtained after the completion of the process. Use slide 15 to explain the students about the make command show them the desired output. Using slide 16, explain them about install command and show them the output that is obtained.

Slide 17

Let us understand writing a simple PHP script.

Writing a Simple PHP Script

- ◆ Rules followed while creating 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

Snippet

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

This snippet is saved in a file with a `.php` extension.
The `echo` command displays "Hello World" in the browser when executed.

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 17 of 41

Using slide 17, tell the students the rules to be followed while creating a PHP script. Explain them how to write a simple PHP script. While explaining, tell them the use of each command and tags used. After explaining the procedure, give them a simple task and tell them to write a PHP script for performing the task.

Slide 18

Let us understand comments in a PHP script.

Comments in a PHP Script

- ◆ Comments are:
 - ❖ Not displayed in the output
 - ❖ Used to assist a programmer to interpret the meaning of a code
- ◆ Comments supported in PHP are:
 - ❖ Single-line
 - ❖ Multi-line
- ◆ Demonstrating the use of comments in a PHP script

Snippet

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

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 18 of 41

Slide 18 shows how to represent a comment in the script. Explain the points as given on slide 18. Also, define the single-line and multi-line comments. Explain the use of the comment and tell them

the importance of including comments to the script. It is to be noted, that the script that includes several function need revising at some time. The comments that are included in the script will help the programmer to assist them, by interpreting the meaning of code.

In-Class Question:

After you finish explaining advantages of PHP, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What are the advantages of PHP?

Answer:

- Easy to learn, use, and implement
- Freely available
- Customizable
- Executed on any Web server on any platform

Slides 19 to 24

Let us understand PHP scripts.

PHP Scripts 1-6

◆ Displaying current date using PHP script

- ◆ Open the gedit text editor
- ◆ Enter the following code snippet:

Snippet

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

- ◆ Save the file as date.php in the /usr/local/apache2/htdocs directory
- ◆ Open the Mozilla Firefox Web browser and enter <http://localhost/date.php> in the Address bar

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 19 of 41

PHP Scripts**2-6**

Displays the following output:



The `gmdate()` function used in the code snippet displays the current date and time in the browser.

The `gmdate("M d Y")` function takes three parameters to display the current date:

M – displays only three letters of the month

d – displays the current date

Y – displays four digits of the current year

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 20 of 41

PHP Scripts**3-6**

- ◆ Displaying a simple text in the browser using the PHP script
 - ◆ Open the gedit text editor
 - ◆ Enter the following code snippet:

Snippet

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

- ◆ Save the file as `stringdisp.php` in the `/usr/local/apache2/htdocs` directory
- ◆ Open the Mozilla Firefox Web browser and enter `http://localhost/stringdisp.php` in the Address bar

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 21 of 41

PHP Scripts

4-6

Displays the following output:

The screenshot shows a Mozilla Firefox window with the title bar "Mozilla Firefox". The address bar contains "http://localhost/stringdisp.php". The main content area displays the text "Hello Everybody". At the bottom of the browser window, there is a "Done" button.

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 22 of 41

PHP Scripts

5-6

- ◆ Displaying a text in the browser using a variable
 - ❖ Open the gedit text editor
 - ❖ Enter the following code snippet:

Snippet

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

- ❖ Save the file as stringname.php in the /usr/local/apache2/htdocs directory
- ❖ Open the Mozilla Firefox Web browser and enter <http://localhost/stringname.php> in the Address bar

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 23 of 41

PHP Scripts 6-6

Displays the following output:

The screenshot shows a Mozilla Firefox browser window with the title bar "Mozilla Firefox". The address bar contains "http://localhost/stringname.php". The main content area displays the text "My name is Samson". Below the content area is a "Done" button.

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.

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 24 of 41

Using slide 19, tell the students about scripting in PHP. Explain them how to start with the PHP scripts and explain them about the editor used to write a PHP script. Using slide 20, show them the output for the code mentioned in the previous slide 19. Tell them how the function is declared and defined in the script. Explain to the students about the parameters used in the function and tell them what the parameters represent.

Then, using slide 21, explain to the students the code to display the text ‘Hello Everybody’ using echo command. Also, tell them how to view the corresponding output in a Web browser. Using slide 22, show the desired output in the Web browser. Tell them how to include the URL to execute the code in the browser. Using slide 23, you start explaining the given code snippet. Here, the variable `str` has been declared and assigned a value as “My name is Samson”. The string `str` is displayed with the `echo` command.

Tell them the syntax for declaring the variable included in the script is '`$variable name=value;`'. Use slide 24 to explain the students about the text string that is assigned to a variable. Show them the output that is displayed on the browser.

Additional Information:

For more information on PHP scripting, visit the following links:

<http://php.net/manual/en/tutorial.firstpage.php>
<http://www.nbccs.rutgers.edu/~cje/php1/hello.html>



Key Points to Remember:

- The echo command displays the contents of the variable in the output.

In-Class Question:

After you finish explaining installation of PHP, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What does installation of PHP require?

Answer: Installation of PHP requires Web Servers and Database.

Slide 25

Let us understand about HTTP.



HTTP

- ◆ It is a network transmission protocol
- ◆ It transfers hypertext files
- ◆ It provides instructions for communication between the client and the server
- ◆ It runs on the Transmission Control Protocol/Internet Protocol (TCP/IP) suite

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 25 of 41

Using slide 25, start explaining about HTTP, which is a transmission protocol. Tell them that HTTP stands for HyperText Transfer Protocol. Explain the students the need for using the protocol. Then explain them about the TCP/IP protocol suite.

Slides 26 and 27

Let us understand about HTTP header.



HTTP Header

1-2

- ◆ **HTTP Header**
 - ◆ Is an Internet protocol
 - ◆ Contains instructions to transfer information between a Web client and a Web server
- ◆ The instruction are of two types:
 - ◆ Request - sent by the client to the server
 - ◆ Response - from the server to a client request
- ◆ Format for request or response contains following components:
 - ◆ A request or a response line
 - ◆ HTTP header lines
 - ◆ A blank line
 - ◆ A message body, which is optional

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 26 of 41

HTTP Header

2-2

- Format of an HTTP message is as follows:

Syntax

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

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 27 of 41

Using slide 26, explain the students about the HTTP headers. Start explaining the students about HTTP header. Then, continue to explain the instructions that are of two types. Explain them the format for request or response and the components included in the HTTP header instructions. Using slide 27, explain the students about the format of an HTTP message. Explain them the syntax that consists of three headers with three different values such as Value 1, Value 2, and Value 3. Continue explaining each line of it.

In-Class Question:

After you finish explaining about HTTP header, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What is meant by HTTP?

Answer:

HTTP is a network transmission protocol. It helps to transfer hypertext files and provides instructions for communication between the client and the server.

Slide 28

Let us understand Initial Request or Response Line.

Initial Request or Response Line

- ◆ Request line contains the following information separated by spaces:
 - ❖ An HTTP method name
 - ❖ Uniform Resource Identifier (URI)
 - ❖ The HTTP version being used

Snippet

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

- ◆ 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

Snippet

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

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 28 of 41

Using slide 28, explain them the request line and the response. Explain the students some real time examples to show the difference between request and response. Then explain them the code snippet given in slide 28.

Slides 29 and 30

Let us learn about Header Lines.

Header Lines

1-2

- ◆ Provide information about the request or response or the data sent in the message body

Syntax

```
Header-Name: value
```

- ◆ Categorized as follows:
 - ❖ General:
 - ❖ Control the processing of a message
 - ❖ Provide extra information to the receiver
 - ❖ Entity:
 - ❖ Provides information about the entity
 - ❖ Request or response:
 - ❖ Provides details about the client's request
 - ❖ Contains response header attached with the response sent by the server

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 29 of 41

Header Lines

2-2

- ◆ Displaying HTTP header lines

Snippet

```
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** - specifies requested file name and the version of HTTP used
- ◆ **User-agent** - specifies the name of the browser and the version
- ◆ **Last-Modified** - specifies the date and time when the resource was last modified
- ◆ **Accept-Language** - specifies the language preference as English

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 30 of 41

Using slide 29, explain the students about Header Lines. List the categories as General, Entity, and Request or Response. Then, explain each of the categories briefly using the points given in slide 29. Use slide 30 to explain about the HTTP Header Lines. Also, describe them each of the command given in the code snippet on slide 30. Tell them about GET, User-agent, Last-modified, and Accept-language.

Slide 31

Let us learn about the message body.

The Message Body

- ◆ Is the third and an optional component of an HTTP header
- ◆ Appears after the header lines
- ◆ Messages can be of two types:
 - ◆ **Request Message** - contains user data and uploaded files
 - ◆ **Response Message** - contains requested resource

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 31 of 41

Using slide 31, explain the message body and its types. While explaining the request and response message, give the students a real world example.

Slides 32 to 34

Let us learn about using HTTP headers in PHP.

Using HTTP Headers in PHP

1-3

- ◆ **header ()** function:
 - ❖ Used to generate the HTTP headers
 - ❖ Sends the HTTP commands to the server through HTTP protocols
 - ❖ Displays a blank line showing that the header information is complete after the execution of the `header ()` function

Syntax

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

Where,

- ◆ **string** – specifies the header string to be sent
- ◆ **replace** – is an optional parameter and indicates whether should be replaced or not
- ◆ **http_response_code** - is an optional parameter and forces the HTTP response code to the specified value

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 32 of 41

Using HTTP Headers in PHP

2-3

- ◆ Displaying an authentication header

Snippet

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

Authentication helps to identify if a client is allowed to access to a resource.

Authentication is a means of negotiating access to a secure resource.

Version 1.0 © Aptech Limited. Introduction to PHP / Session 1 / Slide 33 of 41

Using HTTP Headers in PHP

3-3

- ❖ Authentication schemes are as follows:
 - ❖ Http Basic Authentication
 - Sends an encoded string
 - Contains a user name and password
 - ❖ HTTP Digest Authentication
 - Is a challenge-response scheme
 - Server sends a data string to the client as a challenge
 - Client responds with a user name and password
 - ❖ NTLM
 - Is a challenge-response scheme
 - Uses Windows credentials to transform the challenge data
 - Requires multiple exchanges between the client and server
 - ❖ Negotiate
 - Selects between Kerberos and NTLM depending on their availability

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 34 of 41

Use slide 32 to explain the students about the `header()` function in brief. Then, explain them the syntax given in slide 32 and describe about `string`, `replace`, and `http_response_code`. Use slide 33 to explain the students about the authentication header. Tell them the use of authentication and meaning of it. Using slide 34, explain the students about the Authentication Schemes. There are four schemes namely, HTTP basic authentication, HTTP digest authentication, NTLM, and negotiate. Explain each of the schemes in brief.

Slides 35 and 36

Let us learn about `header()` function with `replace` option.

header() Function with replace Option

1-2

- ❖ The `replace` option specifies to replace the previous header or add a second header to the document
- ❖ If `false`, then new header will be added to the document

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

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 35 of 41

header() Function with replace Option

2-2

- Displaying addition of multiple headers to the document

Snippet

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

WWW-Authenticate - specifies the authentication string.

NTLM - specifies a challenge-response authentication mechanism.

false - defines the parameter of the replace option.

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 36 of 41

Use slide 35 to explain about the replace option. Tell them the syntax and explain them using an example for the syntax. Inform them that one can include multiple headers in single document. Use slide 36 to explain the students about the parameters specified as WWW-authenticate, NTLM, and false.

Additional Information:

For more information on header() function with replace option, visit the following links:

<http://php.net/manual/en/function.header.php>

<https://www.udemy.com/blog/php-header-xml/>

Slides 37 to 39

Let us learn about `header()` function with `http_response_code` option.

header() Function with http_response_code Option 1-3

- ◆ Displays the response of the Web server for a request
- ◆ The request can include the status or the location of the client

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

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 37 of 41

header() Function with http_response_code Option 2-3

- ◆ Displaying a PHP script to redirect the user from one Web page or URL to another Web site

Snippet

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

Location is a type of HTTP header redirecting the browser to the specified URL.

Version 1.0 © Aptech Limited.

Introduction to PHP / Session 1 / Slide 38 of 41

header() Function with http_response_code Option 3-3

- Displaying a PHP script with an HTTP response code

Snippet

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

- 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

Explain the students about the `http_response_code` option in brief using slide 37. Also, explain about `string`, `replace`, and `http_response_code` that are the parameters of the given header. Using slide 38, explain the students about the Location that is a type of HTTP Header and the given code snippet. Use slide 39 to explain the students about displaying a PHP script with an HTTP response code.

Slides 40 and 41

Let us summarize the session.

Summary

1-2

- 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 e-mails by retrieving data from documents and sending it through any standard mail protocol

Summary**2-2**

- ◆ PHP is used to render graphical images, such as GIF and PNG images
- ◆ PHP consists of tools for developing and designing Web pages. These tools are the program text editors. The popular text editor used for writing PHP scripts on Linux platform is gedit
- ◆ A PHP script starts with <?php tag and ends with the ?> tag. These scripts are embedded in the 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

Use slides 40 and 41 to list and explain the summary of this session.

The summary points are as follows:

- 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 e-mails by retrieving data from documents and sending it through any standard mail protocol.
- PHP is used to render graphical images, such as GIF and PNG images.
- PHP consists of tools for developing and designing Web pages. These tools are the program text editors. The popular text editor used for writing PHP scripts on Linux platform is gedit.
- A PHP script starts with <?php tag and ends with the ?> tag. These scripts are embedded in the 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.

1.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to Form Handling.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

For Aptech Center Use Only

Session 2 – Installing and Configuring PHP 7

2.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

2.1.1 Objectives

By the end of this session, learners 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.

2.1.2 Teaching Skills

To teach this session, you should be well versed with PHP and steps involved to install PHP 7. You should also be well versed with the steps to configure, create simple PHP scripts, the method to use HTTP headers in PHP, and similar concepts.

You should teach the concepts in the class using the code snippet provided. For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ 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.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 2 of 32

2.2 In-Class Explanations

Slides 3 and 4

Let us understand the pre-requisites that are required for installing PHP 7.

Pre-requisites for Installing PHP 7

1-2

- ◆ Installation of PHP requires:
 - A Web server
 - A database
- ◆ Web servers supported are as follows:
 - Internet Information Services (IIS)
 - Apache
 - Nginx

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 3 of 32



The slide is titled "Pre-requisites for Installing PHP 7" and is labeled "2-2". It features a list of databases supported by PHP 7, represented by colored circles and bars: DB2 (red), MS SQL (green), MySQL (purple), Oracle (teal), and PostgreSQL (orange). To the right, there is a diagram of a database cylinder with a query overlay. The query is:

```
SELECT Name, S.StudentID, E.CourseNum, C.CourseName  
FROM Students S, Enrolled In E, Courses C  
WHERE S.StudentID = E.StudentID  
AND E.CourseNum = C.CourseID  
AND E.CourseNum = "231"
```

Version 3.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 4 of 32.

Using slide 3, explain that the foremost pre-requisite for installing PHP 7 is a Web server and a database that should be installed in the system.

There are a list of Web servers that support PHP 7, such as Apache, Microsoft Internet Information Service (IIS), and Nginx.

A few databases that could be used with PHP are DB2, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL.

It is important that a Web browser is also installed in addition to a Web server in your system to run PHP 7.

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

Additional Information:

Download the relevant package from <http://php.net/downloads.php> Website.

Slides 5 and 6

Let us now see how to install the packages.

Installing the Packages 1-2

- ◆ PHP can be installed on:

- ◆ Download the relevant package from <http://php.net/downloads.php> Website.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 5 of 32

Installing the Packages 2-2

- ◆ To extract PHP packages on Linux, perform these steps:
 - ◆ Enter the following command at the command prompt:

```
# tar xjvf php-7.0.4.tar.gz
# tar -xf php-7.0.4.tar
# cd php-7.0.4/
```

 - ◆ To install the development package, enter 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
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 6 of 32

Using slide 5, explain that PHP can be installed on Linux, Windows, or Mac OS systems. Again, specify that there are various downloads pertaining to the stable releases of PHP on <http://php.net/downloads.php> Website.

Using slide 6, explain the steps to be followed to extract PHP packages.

In-Class Question:

After you finish explaining the prerequisites and databases that support PHP 7, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



List a few databases that are used while installing PHP 7.

Answer:

DB2, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL.

Slides 7 and 8

Let us now understand the steps to configure PHP 7.0.4.

Configuring PHP 7.0.4 1-2

- ◆ To configure PHP 7.0.4, perform the following steps:
 - ◆ Enter the following command 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
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 7 of 32

Configuring PHP 7.0.4 2-2

Displays the following output:

```
root@localhost:~/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
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for a sed that does not truncate output... /usr/bin/sed
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for cc... cc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether cc accepts -g... yes
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 8 of 32

Using slide 7, explain that the code has to be typed at the command prompt to configure PHP 7.0.4. Using slide 8, show the output of the Configure command.

Slides 9 to 11

Let us now understand the steps in installing PHP 7.0.4.

Installing PHP 7.0.4

Step 1

To extract the packages, install the pre-requisites, and configure the PHP files, enter the following command at the command prompt:

```
# make
```

Installing PHP 7.0.4

Displays the following output:

```
root@localhost:~/php-7.0.4# make
/bin/sh /root/php-7.0.4/libtool -silent -preserve-dup-deps --mode=compile cc -OZEND_ENABLE_STATIC_TSRLMS_CACHE=1 -lext/opcache -I/root/php-7.0.4/ext/opcache -DPHP_ATOM_INC -I/Root/php-7.0.4/include -I/root/php-7.0.4/main -I/root/php-7.0.4 -I/root/php-7.0.4/ext/dbstring/libxml2 -I/usr/include/freetype2 -I/root/php-7.0.4/ext/mbstring/oniguruma -I/root/php-7.0.4/ext/mbstring/libmbfl -I/root/php-7.0.4/ext/sqlite3/libsqlite -I/root/php-7.0.4/ext/xml/xmlwriter -I/root/php-7.0.4/TSRM -I/root/php-7.0.4/zend -I/usr/include -g -O2 -fvisibility=hidden -c /root/php-7.0.4/ext/opcache ZendAccelerator.c -o ext/opcache/ZendAccelerator.o
/bin/sh /root/php-7.0.4/libtool -silent -preserve-dup-deps --mode=compile cc -OZEND_ENABLE_STATIC_TSRLMS_CACHE=1 -lext/opcache -I/root/php-7.0.4/ext/opcache -DPHP_ATOM_INC -I/Root/php-7.0.4/include -I/root/php-7.0.4/main -I/root/php-7.0.4 -I/root/php-7.0.4/ext/dblib/lib -I/usr/include/libxml2 -I/usr/include/freetype2 -I/root/php-7.0.4/ext/mbstring/oniguruma -I/root/php-7.0.4/ext/mbstring/libmbfl -I/root/php-7.0.4/ext/sqlite3/libsqlite -I/root/php-7.0.4/ext/xml/xmlwriter -I/root/php-7.0.4/TSRM -I/root/php-7.0.4/zend -I/usr/include -g -O2 -fvisibility=hidden -c /root/php-7.0.4/ext/opcache ZendAccelerator.blklist.c -o ext/opcache/send_accelerator_blklist.o
/bin/sh /root/php-7.0.4/libtool -silent -preserve-dup-deps -DZEND_ENABLE_STATIC_TSRLMS_CACHE=1 -lext/opcache -I/root/php-7.0.4/ext/opcache -DPHP_ATOM_INC -I/Root/php-7.0.4/include -I/root/php-7.0.4/main -I/root/php-7.0.4 -I/root/php-7.0.4/ext/dblib/lib -I/usr/include/libxml2 -I/usr/include/freetype2 -I/root/php-7.0.4/ext/mbstring/oniguruma -I/root/php-7.0.4/ext/mbstring/libmbfl -I/root/php-7.0.4/ext/sqlite3/libsqlite -I/root/php-7.0.4/ext/xml/xmlwriter -I/root/php-7.0.4/TSRM -I/root/php-7.0.4/zend -I/usr/include -g -O2 -fvisibility=hidden -c /root/php-7.0.4/ext/opcache ZendAccelerator.blklist.o
This command determines large files and issues commands to recompile those large files.
```

Installing PHP 7.0.4

Step 2

To install PHP, enter the following command at the command prompt:

```
# make install
```

Displays the following output:

```
root@localhost:~/php-7.0.4# make install
/bin/sh /root/php-7.0.4/ext/json_scanner/json_scanner_defs.h -no-generation-data -bcl -o /root/php-7.0.4/ext/json/json_scanner.c /root/php-7.0.4/ext/json_scanner.c
Installing shared extensions:      /usr/local/php7/lib/php/extensions/no-debug-no-zts
Installing PHP binary:            /usr/local/php7/bin/
Installing PHP CLI man page:     /usr/local/php7/man/man1/
Installing PHP FPM binary:        /usr/local/php7/sbin/
Installing PHP CGI binary:        /usr/local/php7/cgi/
Installing PHP FPM config page:   /usr/local/php7/php/fpm/
Installing PHP FPM status page:   /usr/local/php7/php/fpm/status
Installing phpdb binary:          /usr/local/php7/bin/
Installing phpdb man page:        /usr/local/php7/php/man/man1/
Installing PHP CGI config page:   /usr/local/php7/php/cgi/
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/
Installing man pages:             /usr/local/php7/man/man1/
program: php-config
Installing man pages:
page: phpcgi.i
page: php-config.i
```

Explain that only after you have extracted the packages, installed the prerequisites, and configured the PHP files, you can install PHP.

Using slide 9, explain that the first step in installing PHP is to run the command from the command prompt.

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

Using slide 10, show the output of the `make` command.

Using slide 11, explain that the second step is to run the command. Show the output given on the slide and explain it.

Slide 12

Let us now understand how to set up Apache in order to use PHP.

The slide has a blue header bar with the title "Setting up Apache to Use PHP". Below the header, there are four numbered steps:

- Step 1**: Open the `httpd.conf` file.
- Step 2**: Add the following directives in the file:

```
AddHandler application/x-httpd-php .php
LoadModule php7_module C:\php7.dll
AddType application/x-httpd-php .php
PHPIniDir C:\php
```
- Step 3**: If required, change the path of the PHP installation folder.
- Step 4**: Save and restart the Apache Web server.

At the bottom of the slide, there is a small footer with the text "Version 1.0 © Aptech Limited." and "Programming for the Web with PHP / Session 2 / Slide 12 of 32".

Explain that after successfully installing PHP, you need to set up your Web server to work with PHP. If you are using Apache as the Web server, then the below steps need to be followed.

Using slide 12, explain the steps that are used in setting up an Apache server.

Slide 13

Let us now understand how to write a simple PHP script.

The screenshot shows a slide titled "Writing a Simple PHP Script". It contains a list of rules for writing PHP scripts and a code snippet example.

Rules followed while writing 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

Snippet

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

This snippet is saved in a file with a `.php` extension.
The `echo` command displays "Hello World" in the browser when executed.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 13 of 32

Using slide 13, explain that a PHP file can include simple text, HTML tags, and PHP script. Also, specify there are a few rules that need to be followed when creating a PHP script.

Explain that `.php` extension enables the Web server to process the file as a PHP file.

Using slide 13, show the syntax to write a simple PHP script.

Explain that the `echo` command 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.

Slides 14 to 21

Let us now understand how to add comments in a PHP script.

Comments in a PHP Script

1-8

- ◆ Comments are:
 - ❖ Not displayed in the output
 - ❖ Used to assist a programmer to interpret the meaning of a code
- ◆ Comments supported in PHP are:
 - ❖ Single-line
 - ❖ Multi-line
- ◆ Demonstrating the use of comments in a PHP script

Snippet

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

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 14 of 32

Comments in a PHP Script

2-8

- ◆ Displaying current date using PHP script
 - Open the gedit text editor
 - Enter the following code snippet:

Snippet

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

- Save the file as date.php in the /usr/local/apache2/htdocs directory
- Open Mozilla Firefox Web browser and enter <http://localhost/date.php> in the Address bar

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 15 of 32

Comments in a PHP Script

3-8

Displays the following output:



The echo command used in the code is used to display the output in a Web browser.

The gmdate() function used in the code snippet displays the current date and time in the browser.

The gmdate("M d Y") function takes three parameters to display the current date:

M – displays only three letters of the month

d – displays the current date

Y – displays four digits of the current year

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 16 of 32

Comments in a PHP Script

4-8

- ◆ Displaying a simple text in the browser using the PHP script
 - ❖ Open the gedit text editor
 - ❖ Enter the following code snippet:

Snippet

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

- ❖ Save the file as `stringdisp.php` in the `/usr/local/apache2/htdocs` directory
- ❖ Open Mozilla Firefox Web browser and enter `http://localhost/stringdisp.php` in the Address bar

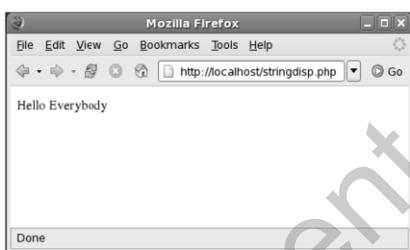
Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 17 of 32

Comments in a PHP Script

5-8

Displays the following output:



Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 18 of 32

Comments in a PHP Script

6-8

- ◆ Rules followed while using a variable in a PHP script are as follows:
 - ❖ Variables:
 - ❖ Must start with a dollar sign '\$'
 - ❖ Can contain strings, numbers, and arrays
 - ❖ Variables names:
 - ❖ Must start with a letter or an underscore '_'
 - ❖ Can only contain alpha-numeric characters and underscores without spaces

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 19 of 32

Comments in a PHP Script 7-8

- ◆ Displaying a text in the browser using a variable:
 - ❖ Open the gedit text editor
 - ❖ Enter the following code snippet:

Snippet

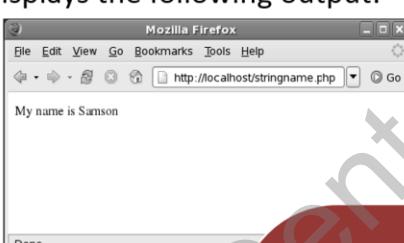
```
<HTML>
<BODY>
<?php
$str = "My name is Samson";
echo $str;
?>
</BODY>
</HTML>
```

- ❖ Save the file as stringname.php in the /usr/local/apache2/htdocs directory
- ❖ Open the Mozilla Firefox Web browser and enter <http://localhost/stringname.php> in the Address bar

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 20 of 32

Comments in a PHP Script 8-8

Displays the following output:



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.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 21 of 32

Explain that codes are not displayed in the output and are used to assist a programmer to interpret the meaning of a code. PHP supports both single-line and multi-line comments.

Use slide 14 to explain the way to add a comment in a PHP script. Using slide 15 explain the steps to display current date using PHP script.

Explain that 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.

Using slide 16 show the output where date is displayed. Using slide 17, explain the steps required to display a simple text using PHP script.

Explain that again the `echo` command is used to display text. The text to be displayed is enclosed within double quotes.

Using slide 18, show the output where the text is displayed. Using slide 19, explain the various rules that need to be followed when using a variable in a PHP script.

Using slide 20, explain the steps that are required to display a text in a browser using a variable.

Explain that in the mentioned code, 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.

Using slide 21, show the output where the text is displayed.

Slides 22 to 24

Let us now understand how to use HTTP headers in PHP.

The slide has a blue header bar with the title "Using HTTP Headers in PHP" and a small logo. In the top right corner, it says "1-3". The main content area has a white background with a blue sidebar on the left. The sidebar contains the following text:

- ◆ `header()` function:
- ◆ Used to generate HTTP headers
- ◆ Sends HTTP commands to the server through HTTP protocols
- ◆ Displays a blank line showing that the header information is complete after the execution of the `header()` function

A blue button labeled "Syntax" is visible. Below it is a code block:

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

Underneath the code block, the word "where," is followed by a list of parameters:

- ◆ `string` – specifies the header string to be sent
- ◆ `replace` – is an optional parameter and indicates whether it should be replaced or not
- ◆ `http_response_code` - is an optional parameter and forces the HTTP response code to the specified value

At the bottom of the slide, there is a footer bar with the text "Version 1.0 © Aptech Limited" and "Programming for the Web with PHP / Session 2 / Slide 22 of 32".

Using HTTP Headers in PHP

2-3

◆ Displaying an authentication header

Snippet

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

Authentication helps to identify if a client is allowed to access to a resource.

Authentication is a means of negotiating access to a secure resource.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 23 of 32

Using HTTP Headers in PHP

3-3

◆ Authentication schemes are as follows:

- ◆ Http Basic Authentication
 - Sends an encoded string
 - Contains a user name and password
- ◆ HTTP Digest Authentication
 - Is a challenge-response scheme
 - Server sends a data string to the client as a challenge
 - Client responds with a user name and password
- ◆ NTLM
 - Is a challenge-response scheme
 - Uses Windows credentials to transform the challenge data
 - Requires multiple exchanges between the client and server
- ◆ Negotiate
 - Selects between Kerberos and NTLM depending on their availability

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 24 of 32

Explain that 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.

Using slide 22 explain that the `header()` function is used to generate the HTTP headers. Using slide 23 explain that the following code is used to display an authentication header. Using slide 24 explain the process of authentication schemes.

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.

Slides 25 and 26

Let us now understand how to use the `header()` with a `replace` option.

1-2

2-2



header() Function with replace Option

- ◆ The `replace` option specifies to replace the previous header or add a second header to the document
- ◆ If `false`, then new header will be added to the document

Syntax

```
void header('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 that all similar headers are replaced



header() Function with replace Option

- ◆ Displaying addition of multiple headers to the document

Snippet

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

WWW-Authenticate - specifies the authentication string.
NTLM - specifies a challenge-response authentication mechanism.
false - defines the parameter of the replace option.



Programming for the Web with PHP / Session 2 / Slide 25 of 32

Programming for the Web with PHP / Session 2 / Slide 26 of 32

Using slide 25, explain what does a `replace` option does and the syntax.

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.

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.

Using slide 26, show the code snippet of adding multiple header to the document.

Slides 27 to 30

Let us now understand the `header()` with `http_response_code` option.

header() Function with http_response_code Option 1-4

- ◆ Displays the response of the Web server for a request
- ◆ The request can include the status or the location of the client

Syntax

```
void header( 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

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 27 of 32

header() Function with http_response_code Option 2-4

- ◆ An HTTP response codes consists of three digits that determine the status of a response.
- ◆ The status codes are classified as follows:
 - ◆ 1xx codes are informational codes
 - ◆ 2xx are success codes
 - ◆ 3xx are redirection codes
 - ◆ 4xx are client error codes
 - ◆ 5xx are server error codes

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 28 of 32

header() Function with http_response_code Option

3-4

- ◆ Displaying a PHP script to redirect the user from one Web page or URL to another Web site

Snippet

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

Location is a type of HTTP header redirecting the browser to the specified URL.

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 29 of 32

header() Function with http_response_code Option

4-4

- ◆ Displaying a PHP script with an HTTP response code

Snippet

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

- ◆ 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

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 2 / Slide 30 of 32

Using slide 27, explain the importance of the option and the syntax to use the option.

Explain that 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

Using slide 28, explain that there are different HTTP response status codes.

Using slide 29 show the syntax of the http_response_code option.

The syntax to redirect the user from one Web page or URL to another Website is as follows:

```
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'.

Using slide 30, show the syntax displays a PHP script with an HTTP response code.

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

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.

In-Class Question:

After you finish explaining the use of authentication headers in PHP, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Which is the significance of an authentication header?

Answer:

An authentication header is a means of negotiating access to a secure resource.

Slides 31 and 32

Summarize the session. Explain each of the following points in brief.

Summary  1-2

- ◆ A Web server and a database is required before installing PHP 7.
- ◆ Any older version of PHP must be uninstalled before installing PHP 7.
- ◆ A PHP file includes simple text, HTML tags, and PHP script.
- ◆ PHP supports both single-line and multiple line comments.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 31 of 32

Summary  2-2

- ◆ PHP automatically assigns the correct data type for a variable depending upon the value assigned to the variable.
- ◆ A PHP script starts with <?php tag and ends with the ?> tag. These scripts are embedded in the 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.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 2 / Slide 32 of 32

Using slides 31 and 32, tell them that:

- 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.

2.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

For Aptech Center Use Only

Session 3 – New Features of PHP 7

3.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

3.1.1 Objectives

By the end of this session, learners 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.

3.1.2 Teaching Skills

To teach this session, you should be well versed with the new features that have been added in PHP 7 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()`.

You should teach the concepts in the class using the code snippet provided. For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

The slide has a blue header bar with the word "Objectives" and a small green circular icon. The main content area contains a bulleted list of nine items, each starting with a diamond symbol and followed by a descriptive sentence. At the bottom of the slide, there is a thin blue footer bar with the text "Version 1.0 © Aptech Limited." on the left and "Programming for the Web with PHP / Session 3 / Slide 2 of 43" on the right.

- ◆ 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.

3.2 In-Class Explanations

Slides 3 to 6

Let us understand the `intdiv()` function.

The slide has a blue header bar with the title "Integer Division" and a small green circular icon. On the right side of the header, it says "1-4". The main content area contains a bulleted list of three items, each starting with a diamond symbol and followed by a descriptive sentence. Below this is a "Snippet" section with a dark blue header containing the word "Snippet". Inside this section is a code block with PHP code. At the bottom of the slide is a red callout box with white text. The footer bar at the bottom contains the text "Version 1.0 © Aptech Limited." on the left and "Programming for the Web with PHP / Session 3 / Slide 3 of 43" on the right.

- ◆ Is a division operator
- ◆ Is represented as `/`
- ◆ In earlier versions, always returned a `float` value and one had to use workarounds to get integers such as:

Snippet

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

`$x` and `$y` contain float values. Therefore, using the division operator may not provide accurate results. Here, the value returned is cast to integer using `int()` function.

Integer Division

2-4

- ◆ In PHP 7, the integer division function accepts two parameters:
 - ◆ Dividend
 - ◆ Divisor
- ◆ It returns an integer result

Syntax`intdiv(m, n)`

Where,

- ◆ **intdiv** – is the function
- ◆ **m** – is the dividend
- ◆ **n** – is the divisor

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 4 of 43

Integer Division

3-4

- ◆ Demonstrating the use of the `intdiv()` function and the behavior of operator '/'.

Snippet

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

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 5 of 43

Integer Division

4-4

- ◆ Displays the following output:

http://localhost/cs3_1.php

2.66666666666667
2

The first statement displays the value 2.66666666666667 as the expression uses the division operator that returns a float value.

The second statement displays the value 2 as the expression is using the function `intdiv()` that returns an integer value after division.

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 6 of 43

Using slide 3, explain that prior to PHP 7, the division operator (/) was used for division operation. However, this operator always returns a `float` value.

Explain the snippet shown on the slide.

Let us consider the following example:

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

Explain that 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.

Using slide 4, explain that 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.

The syntax for integer division function is:

`intdiv(m, n)`

where,

- `intdiv` is the function
- `m` is the dividend
- `n` is the divisor

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

Using slide 5, show the code on how to use the `intdiv()` operator.

Explain that when the code is executed, then the first `echo` statement displays 2.66666666666667, 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.

Display slide 6 and show the output.

Slides 7 to 17

Let us now understand the spaceship operator `<=>`.

Spaceship Operator (`<=>`) 1-11

- ◆ Is a single comparison operator
- ◆ Is mainly used for sorting
- ◆ Is represented as `<=>`
- ◆ Compares operands against three rules, namely:

	Greater than
	Lesser than
	Equal to

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 7 of 43

Spaceship Operator (`<=>`) 2-11

- ◆ Demonstrating the use of spaceship operator

Syntax

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

- ◆ This expression displays the result:
 - ◆ -1 if \$x is smaller than \$y
 - ◆ 0 if \$x is equal to \$y
 - ◆ 1 if \$x is greater than \$y

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 8 of 43

Spaceship Operator (`<=>`) 3-11

- ◆ Demonstrating the use of spaceship operator on numbers

Snippet

```
<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
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 9 of 43

Spaceship Operator (<=>)

4-11

```

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

```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 10 of 43

Spaceship Operator (<=>)

5-11

- ◆ Displays the output:

```

http://localhost/cs1_2.php
localhost/cs1_2.php
1<=>2 Returns -1
10<=>10 Returns 0
10<=>5 Returns 1

```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 11 of 43

Spaceship Operator (<=>)

6-11

- ◆ Demonstrating the use of spaceship operator on strings

Snippet

```

<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.

```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 12 of 43

Spaceship Operator (<=>)

7-11

```

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>

```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 13 of 43

Spaceship Operator (<=>)

8-11

- ◆ Displaying the output:

```

http://localhost/cs1_3.php
localhost/cs1_3.php
Cat<=>Dog Returns -1
PHP<=>PHP Returns 0
COMPUTE<=>APPLE Returns 1

```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 14 of 43

Spaceship Operator (<=>)

9-11

- ◆ Demonstrating the use of spaceship operator on arrays

Snippet

```

<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

```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 15 of 43

Spaceship Operator (<=>) 10-11

```
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>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Side 16 of 43

Spaceship Operator (<=>) 11-11

- ◆ Displaying the output:

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Side 17 of 43

Using slide 7, define a spaceship operator and explain its functionality.

Explain that a 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. The most common usage for this operator is in sorting.

Take the following example of a syntax to explain about the operator:

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

Explain that the syntax 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.

Using slide 8, show another code on how to use the spaceship operator.

Now, let us understand how to use the spaceship operator on numbers, strings, and arrays.

Using slides 9 and 10, explain how to use the spaceship operator on numbers. Usually, numbers are compared by their size.

Explain that 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.

Using slide 11, display the output of the code where the spaceship operator is used on numbers.

Using slides 12 and 13, explain how to use the spaceship operator on strings. Explain that the strings are compared lexically, that is, by their alphabetical order.

Explain that 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".

Using slide 14, display the output of the code where the spaceship operator is used in strings.

Using slides 15 and 16, explain how to use the spaceship operator on arrays. In case of arrays, corresponding array elements are compared by their size with one another.

Explain that in the code, two arrays `$x`, `$y` are declared without any elements. They are compared with the spaceship operator. Here the output is 0 (zero) because the two arrays are empty and hence, they are treated as equal. In the next two statements, two arrays `$m`, `$n` are defined with exact number of elements and the elements are same. The next statement using spaceship operator returns a value 0(zero) because the two arrays `$m` and `$n` are identical. Next two statements declared two arrays `$p`, `$q` with different elements in the arrays. The next statement used spaceship operator between `$m` and `$x`. It returns 1, because the `$m` is greater than `$x`. The next statement uses spaceship operator between `$m` and `$q`. It returns -1 because `$m` is less than `$q`. Here each elements one array `$m` is compared with corresponding element of array `$q`.

Using slide 17, display the output of the code where the spaceship operator is used on arrays.

In-Class Question:

After you finish explaining the new features of PHP, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



How the strings are compared using the spaceship operator?

Answer:

The strings are compared lexically, that is, by their alphabetical order.

Slides 18 to 20

Let us now understand the null coalescing operator (??)

Null Coalescing Operator (??) 1-3

- ◆ Is used to check for a NULL value
- ◆ Is represented as ??
- ◆ Returns:

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 18 of 43

Null Coalescing Operator (??) 2-3

- ◆ Demonstrating the use of null coalescing operator

Snippet
<pre><HTML> <BODY> <?php \$name = \$first_name ?? "Guest"; echo \$name; ?> </BODY> </HTML></pre>

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 19 of 43

Null Coalescing Operator (??) 3-3

- ◆ Displaying the output:

Guest

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 20 of 43

Using explain 18, explain that the null coalescing operator is a binary operator, which is part of the syntax for a basic conditional expression.

Explain that this operator checks strictly for a NULL value or a non-existent item such as variable, array, index, or property. The ?? operator returns the result of its first operand if it exists and is NOT NULL, otherwise, it returns second operand.

Let us understand this with the help of an example using slide 19.

Explain that in the code \$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".

Using slide 20, display the output of the code.

Slides 21 and 22

Let us now understand the scalar type declarations.

Scalar Type Declarations 1-2

- ◆ Data types that hold single data type are known as scalar data types.
- ◆ These can be:

Integer

Float

Boolean

Strings

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 21 of 43

Scalar Type Declarations

2-2

- ◆ Refers to specifying the data type of a parameter in a function
- ◆ Also referred to as type hinting
- ◆ Provides hints to a function
- ◆ Enforces to input the parameter data type, that can be either strict or coercive

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 22 of 43

Using slide 21, define and list the types of scalar type declarations. Using slide 22, explain further on scalar type declarations.

Slides 23 to 25

Let us now understand about the uniform variable syntax.

Uniform Variable Syntax

1-3

- ◆ Allows to use multiple operators in a given expression
- ◆ Follows a left-to-right evaluation order
- ◆ Allows backward compatibility in old evaluation technique
- ◆ Allows nesting of dereferencing operations

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 23 of 43

Uniform Variable Syntax

2-3

- Demonstrating the use of uniform variable syntax

Snippet

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

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 24 of 43

Uniform Variable Syntax

3-3

- Displays the following output:

```
This is g()
*****
This is g()
This is f()
*****
This is g()
This is f()
This is e()
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 25 of 43

Using slide 23, explain the advantages of using uniform variable syntax.

Using slide 24, explain the use of uniform variable syntax.

Explain that in the code 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.

Using slide 25, display the output of the code.

Slides 26 to 29

Let us now understand the function of use operator.

Use Operator 1-4

- ◆ Is used to achieve aliasing
- ◆ Allows grouping of multiple declarations
- ◆ Enables to group classes, functions, and constants imported from the same namespace

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 26 of 43

Use Operator 2-4

- ◆ Demonstrating the use of `use` operator

Snippet

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

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

Save the file as myfile.php.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 27 of 43

Use Operator 3-4

- Include myfile.php in ugroup.php as shown:

Snippet

```
<?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();
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 28 of 43

Use Operator 4-4

- Displays the following output:

This is Boston
This is NewYork
This is foo1()
This is foo2()

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 29 of 43

Explain that the use function is used to achieve aliasing. Tell them that before familiarizing more with the function of the use operator, let us first understand what is meant by aliasing.

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

Using slide 26, explain that the use operator is used to achieve aliasing. 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.

Using slides 27 and 28 display the code to show how to use the use operator.

Explain that there are two classes that are declared, namely `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. Tell that this program can be stored in a file called `myfile.php`, which will be included in slide 27.

Explain that the code 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.

Using slide 29, show the output of the code.

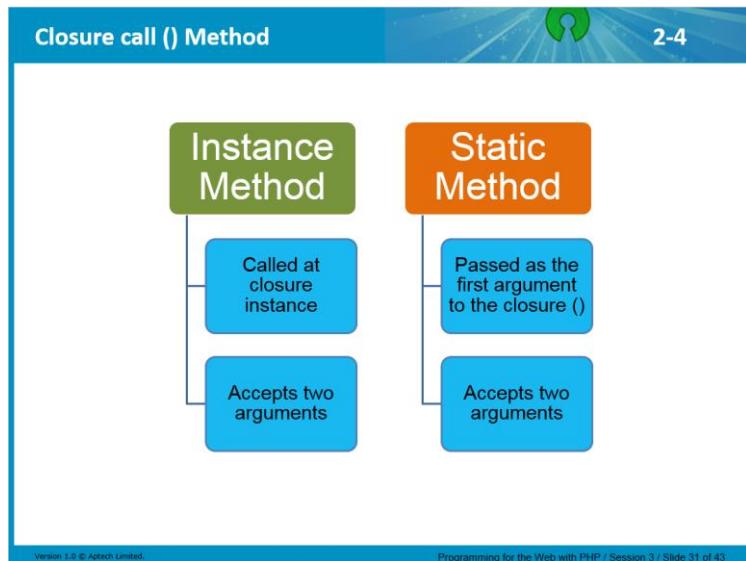
Slides 30 to 33

Let us now understand the function of closure call () method.

The screenshot shows a presentation slide with a blue header bar. The title 'Closure call () Method' is at the top left, and a green circular icon with a person symbol is at the top right. The slide number '1-4' is in the top right corner. The main content area contains a bulleted list of four points:

- ◆ A closure record stores a function together with an environment.
- ◆ A closure method allows the functions to access the captured variables.
- ◆ Adding `$this` parameter to the closure method results in two new methods:
 - ◆ The Instance method - `Closure->bindTo()`
 - ◆ The Static method - `Closure::bind()`

At the bottom of the slide, there is a small footer bar with the text 'Version 1.0 © Aptech Limited.' on the left and 'Programming for the Web with PHP / Session 3 / Slide 30 of 43' on the right.



Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 31 of 43

Closure call () Method

◆ Demonstrating the use of closure ()

Snippet

```

<?php
class Greetings {
    private $word = "Hello";
}

$closure = function($whom) {
    echo "$this->word $whom\n";
};

$obj = new Greetings();

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

?>

```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 32 of 43

Closure call () Method

◆ Displays the following output

A screenshot of a web browser window displaying the output of the provided PHP code. The browser address bar shows `http://localhost/cs3_9.php`. The page content area displays the following text:

```
Hello John
Hello Kevin
```

Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 3 / Slide 33 of 43

Using slide 30, explain the meaning of the closure.

Say that 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.

Further explain, that by adding the `$this` parameter, Closure gained two methods, the instance method `Closure->bindTo()` and the static method `Closure::bind()`.

Using slide 31, explain that 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`.

Using slide 32, display the code to show how to use the closure call () function.

Explain that in the code a class named `Greetings` is created. 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.

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'.

Using slide 33, show the output of the code.

Slides 34 to 36

Let us now understand the generator return expressions.

Generator Return Expressions 1-3

- ◆ A statement used within a generator to return the final expression.
- ◆ The value can be retrieved using the `getReturn()` method.
- ◆ The method should be used only after the generator has finished yielding results.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 34 of 43

Generator Return Expressions 2-3

- ◆ Demonstrating the use of `getReturn()` expression

Snippet

```
<?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;
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 35 of 43

Generator Return Expressions 3-3

- ◆ Displaying the output:

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 36 of 43

Using slide 34, explain that 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.

Using slide 35, display the code to show how to use the `getReturn()` function.

Explain that in the code the generator returns random values. At the end of the iteration, the generator returns -1.

Using slide 36, show the output of the code.

Slides 37 and 38

Let us now understand the generator delegation via `yield from` function.

Generator Delegation via `yield from` 1-2

- Demonstrating the use of `yield from` keyword

Snippet

```
<?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";
}
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 37 of 43

Generator Delegation via `yield from` 2-2

- Displaying the following output:

http://localhost/cs2_6.php

f2() 1
f2() 2
f2() 3
f2() end
f1() 1
f1() 2
3
4
f1() 3
f1() end

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 38 of 43

Explain that 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.

Using slide 37, display the code to show how to use the `yield from` function.

Explain that in the code, 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`.

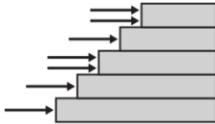
Using slide 38, show the output of the code.

Slides 39 to 41

Let us now understand the `levels` Parameter of `dirname()`

levels Parameter of dirname() 1-3

- ◆ The `dirname()` function returns the parent's directory path.
- ◆ The `dirname()` now accepts a second parameter that specifies the number of levels a parent directory can go up.
- ◆ The `levels` parameter tells how many parent directories can go up.



Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 39 of 43

levels Parameter of dirname() 2-3

- ◆ Demonstrating the use of `dirname()` function

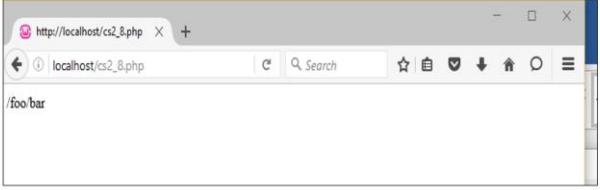
Snippet

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

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 40 of 43

levels Parameter of dirname() 3-3

- ◆ Displays the following output:



Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 41 of 43

Using slide 39, explain that the `dirname()` function returns the parent's directory path. Explain that the new `levels` parameter tells how many parent directories to go up. The `dirname()` function can now accept a second parameter to set how many levels up it will go, meaning you can avoid nesting.

Using slide 40, display the code to show how to use the `dirname()` function.

Explain that the `dirname` function has two arguments. The first argument is the directory name and the second argument indicates how many levels of the directory name should be returned.

Using slide 41, show the output of the code.

Slides 42 and 43

Summarize the session.

Summary 1-2

- ◆ 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`.
- ◆ The null coalesce operator `??` returns the left operand if it is not null; otherwise, it returns the right operand.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 42 of 43

Summary

2-2

- ◆ Group 'use' declarations allow to deduplicate common prefixes in 'use' statements and specify unique parts within a block {}.
- ◆ Generator->getReturn() method allows you to retrieve value returned by any return statement inside the generator.
- ◆ 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.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 3 / Slide 43 of 43

Using slides 42 and 43, explain each of the following points in brief:

- The intdiv() function performs the division function and accepts two parameters, where the first parameter is the dividend and the second is the divisor.
- The spaceship operator <=> is used for sorting and compares operands against three rules, namely: greater than, lesser than, and equal to.
- The null coalescing operator ?? is used to check for a null value and it returns the result of its first operand if it exists and is NOT NULL, otherwise, it returns second operand.
- The use operator is used to achieve aliasing and allows grouping of multiple declarations.
- A closure method allows functions to access the captured variables. When adding \$this parameter, Closure gained two methods, the instance method Closure->bindTo () and the static method Closure::bind().
- The generator delegation allows you to yield values from another generator, Traversable object, or array by using the yield from keyword.
- The dirname () function returns the parent's directory path and the new levels parameter tells how many parent directories to go up.

3.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

Session 4 – Form Handling in PHP

4.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

4.1.1 Objectives

At the end of this session, the learners 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

4.1.2 Teaching Skills

To teach this session, you should be well-versed with Form handling mechanism in PHP. You should be able to explain the use of GET and POST methods. You should be able to describe the steps to retrieve data from forms using the Form method. You should familiarize yourself with Form method and the hidden fields.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 2 of 32

Tell them that they will be introduced to GET and POST method and learn the use of methods. They will also taught about retrieving data from forms using the Form methods. In addition, they will also understand about hidden fields.

4.2 In-Class Explanations

Slide 3

Let us introduce Form Handling in PHP.

Introduction

- ◆ Form data is passed to the Web server using following methods:
 - ◆ GET
 - ◆ POST
- ◆ Web server
 - ◆ Accepts the information
 - ◆ Processes the application data
 - ◆ Stores it to the database

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 3 of 32

With slide 3, introduce a Form on a Web page and the role of the Web Server in Form data handling.

Before explaining the GET and POST methods, tell the students the definition of a Form and explain how it is used. Tell them that the Form is an organized set of elements that consists of several Form attributes such as text box, buttons, and so on. The basic use of a Form is to combine several input elements of different types and process the input data to get a particular result. Some of the real world examples of Forms include Registration Form, Login Form, and so on.

It is important to understand that any Form element will be automatically available to your PHP scripts.

Using slide 3, also explain the students about the Web server and its role in Form data handling.

Additional Information:

For more information on forms, visit the following links:

<http://php.net/manual/en/tutorial.forms.php>

<http://www.the-art-of-web.com/php/form-handler/>

Slides 4 and 5

Let us see the basic Form elements.

The slide displays a screenshot of a web page titled "Form". The page content is a "Contact us!" form with the following fields:

- Name * : [Input field]
- Website * : [Input field]
- E-Mail * : [Input field]
- Phone_Number * : [Input field]
- Subject : [Input field]
- Message * : [Text area]

Below the form is a CAPTCHA section with the text "Captcha:" followed by a CAPTCHA image showing the numbers "4 6 5 4" and a button labeled "Daten absenden". The slide has a blue header bar with the title "Form" and a slide number "1-2". The footer of the slide shows "Version 1.0 © Aptech Limited." and "Form Handling in PHP / Session 2 / Slide 4 of 32".

Form

2-2

- ◆ Steps for handling HTML forms and process information are as follows:
 - ❖ User enters information in an HTML form and sends it to the Web server
 - ❖ Web server passes the information to the PHP script engine for:
 - ❖ Processing the information
 - ❖ Sending output back to the Web browser

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 5 of 32

Slide 4 shows a sample Form image that consists of several attributes in it. Tell the students that each Form gets certain inputs from the client and sends it to the server. Using slide 5, tell them about the steps for handling HTML forms. The process includes that the HTML Form is redirected to the page using the POST method from the PHP Form handler. The Form action is the action that is performed with the client-server architecture.

After you finish explaining about Form handling and forms, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.

In-Class Question:



Why does the Web server pass the information in the Form to the PHP script engine?

Answer:

The Web server passes the information in the Form to the PHP script engine for:

- Processing the information
- Sending output back to the Web browser

Slide 6

Let us study the HTML <FORM> tag.

HTML <FORM> Tag

- ◆ **HTML <FORM> tag is:**
 - ❖ Used to create HTML form
 - ❖ Included within the <FORM> and </FORM> tag
- ◆ **Attributes of an HTML form tag are:**
 - ❖ **Action** - defines URI where the form data is sent after it has been submitted
 - ❖ **Method** - defines protocols that are used to submit the form data set
- ◆ **Method protocols are of two types:**
 - ❖ GET
 - ❖ POST

Version 1.0 © Aptech Limited. Form Handling in PHP / Session 2 / Slide 6 of 32

Slide 6 shows the FORM tag and its attributes. To create a Form, it is necessary to have a Form tag that in turn includes the all other attributes embedded into it. Some of the Form attributes are input type, name, action, method, and so on. Explain these points to the students.

Also, explain them about the GET and POST method, which is used to get data and send it to the server.

Slides 7 to 9

Let us explore using the GET method.

Using the GET Method

1-3

- ◆ **GET method**
 - ◆ Directs the Web browser to send the encoded user information to the processing agent
 - ◆ Appends the encoded information at the end of the URL by a question mark (?) which separates URL and form information
- ◆ The form data sent in the URL is a stream of name/value pair separated by ampersand (&)

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 7 of 32

Using the GET Method

2-3

- ◆ An input variable will have following structure:
 - ◆ Name=value
- ◆ Multiple input variables are grouped as follows:
 - ◆ Name1=value1&Name2=value2&Name3=value3
- ◆ The following example shows the multiple name/value pair separated by the & sign:
 - ◆ Name=john&age=18
- ◆ The query string is appended with the following URL:
 - ◆ <http://www.information.com/text.php?Name=john&age=18>

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 8 of 32

Using the GET Method3-3

- ◆ The restrictions of GET method are as follows:
 - ◆ Form data set values are restricted to American Standard Code for Information Interchange (ASCII) characters
 - ◆ Amount of information transferred is limited
 - ◆ Length of the query string is restricted to 255 characters

Using slides 7 to 9, explain about the GET method. Before explaining the GET method, tell the students about the encoding of data. You need to explain only the necessity of encoding data and not the complete process. Encoding the data is done, to change the corresponding data into a machine understandable format.

Using slide 8, explain the students about the general format for defining the Form tag attributes. With the help of the example, show them how to define the Form attributes. Also, using slide 9, tell them about the restrictions of GET method. It is to be noted that the information sent from a Form using GET method is visible to everyone and the data is limited while entering in the Form.

Slide 10

Let us learn about using the POST method.



Using the POST Method

- ◆ POST method
 - ❖ Directs the Web browser to send all the user information to the processing agent
 - ❖ Uses message body of an HTTP request to send the information
 - ❖ Has capacity to transmit more information as:
 - ❖ No physical limit on the amount of information passed in the HTTP request message body
 - ❖ Uses variables to pass form information
- ◆ The drawback of POST method is as follows:
 - ❖ Information sent is not encrypted, so hackers can easily access it

Version 1.0 © Aptech Limited. Form Handling in PHP / Session 2 / Slide 10 of 32

Use slide 10 to explain the students about the POST method. The main use of POST method is to send the information that is received from the Form to the server. Here, you can mention that the information sent from a Form with the POST method is invisible to other users because all names/values are embedded within the body of the HTTP request. Also, there are no limits forth data as in the GET method.

Tell them that the POST method supports advanced functionality such as, support for multi-part binary input while uploading files to the server. Using slide 10, also explain the drawback of the POST method.

While explaining all these points, it is important to tell them the general format of the POST method and how it is represented in the HTML Form scripting.

Slide 11

Let us see the differences between the GET and POST methods.

Difference in the GET and POST Method

- ◆ GET and POST methods work almost identically

Table lists the difference between the GET and POST method

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

Version 1.0 © Aptech Limited. Form Handling in PHP / Session 2 / Slide 11 of 32

The table given in slide 11 describes the main differences between the GET and POST methods. By default both the GET and POST method are treated as `$_GET` and `$_POST`. These methods are invoked irrespective of the access specifiers. That is, these methods are said to be the superglobals that means that they are always accessible from any function, class, or file regardless of scope.

After you finish with explaining the overview of GET and POST methods, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.

In-Class Question:

What is the drawback of POST method?

Answer: Information sent is not encrypted, so hackers can easily access it.

Slides 12 to 18

Let us study the retrieving data using the GET method.

Retrieving Data Using the GET Method

1-7

- ◆ Retrieving data from an HTML form using the GET method

Syntax

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

Where,

- ◆ **varname** - specifies the name of the variable in which data is to be stored
- ◆ **\$_GET["variable"]** - specifies the name of the input variable

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 12 of 32

Retrieving Data Using the GET Method

2-7

- ◆ Steps for creating an HTML form to retrieve data using the GET method are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **Details.html** in the `/usr/local/apache2/htdocs` directory

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 13 of 32

Retrieving Data Using the GET Method

3-7

Snippet

```
<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>
```

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 14 of 32

Retrieving Data Using the GET Method

4-7

- ◆ Steps for creating PHP script to retrieve and process the data entered in the HTML form are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **Details.php** in `/usr/local/apache2/htdocs` directory

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 15 of 32

Retrieving Data Using the GET Method

5-7

Snippet

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

$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>";
?>
```

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 16 of 32

Retrieving Data Using the GET Method

6-7

- ◆ Steps for displaying **details.html** page are as follows:

 1. Open the Mozilla Firefox Web browser
 2. Enter **http://localhost/Details.html** in the Address bar and press **Enter**
 3. Enter **John** in the FIRST NAME box
 4. Enter **Simons** in the LAST NAME box
 5. Enter **32, Park Street, New Jersey** in the ADDRESS box
 6. Enter **45450** in the CONTACT NO. box
 7. Click **SUBMIT**



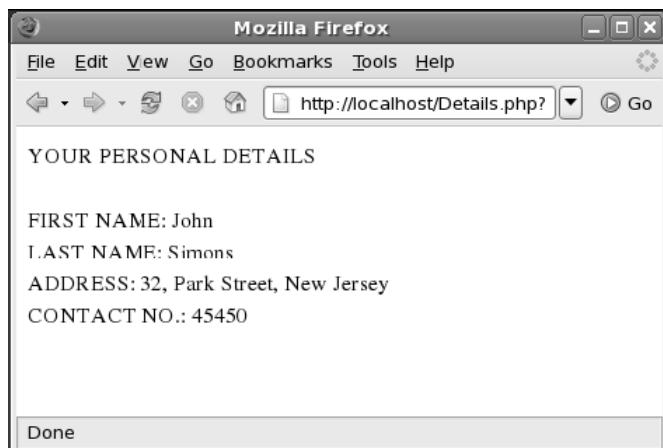
Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 17 of 32

Retrieving Data Using the GET Method

7-7

Displays the following output:



Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 18 of 32

Using slide 12, explain the students the general format for the GET method. Then, explain the data retrieval process using the GET method. Use an example that completely describes the data retrieval process.

The first step is to create a Form with a set of needed attributes using HTML code. To create a Form, it is important to know about the `<FORM>` tag and the Form elements. To explain the Form elements in a better way, show a table for each of the Form elements and the description. Then, using those elements, create a Form that consists of textboxes and buttons. Use slides 13 and 14 to explain the students about the HTML code to create a Form.

The next step is to write a PHP script to process the data entered in the Form. Using slides 15 and 16, explain them the PHP script code that is used to retrieve and process the corresponding data. Now, use slide 17 to explain the steps to execute the corresponding code. Open the browser to view the Form design that is created. Tell the students to enter the appropriate values in the field. Show them the output that is displayed on slide 18. Then tell them some additional Form elements that enhance the look and feel of it.

Slides 19 to 25

Let us study the retrieving data using the POST method.

Retrieving Data Using the POST Method

1-7

- ◆ Retrieving data from an HTML form using the POST method

Syntax

```
$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

Retrieving Data Using the POST Method

2-7

- ◆ Steps for creating an HTML form to retrieve data using the POST method are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **EMP_DETAILS.html** in the `/usr/local/apache2/htdocs` directory

Retrieving Data Using the POST Method

3-7

Snippet

```
<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>
```

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 21 of 32

Retrieving Data Using the POST Method

4-7

Snippet

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

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 22 of 32

Retrieving Data Using the POST Method

5-7

- ◆ Steps for creating a PHP script to retrieve and process the data entered in the HTML form are as follows:

1. Open a new file in the **gedit** text editor
2. Enter the code and save the file as **EMP_DETAILS.php** in the **/usr/local/apache2/htdocs** directory

Snippet

```
<?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>";
?>
```

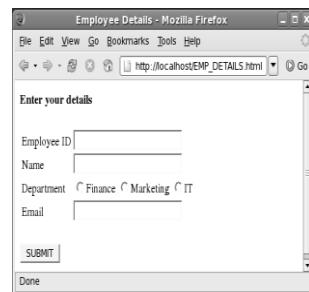
Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 23 of 32

Retrieving Data Using the POST Method

6-7

- ◆ Steps for displaying **EMP_DETAILS.html** page are as follows:
1. Open the Mozilla Firefox Web browser
 2. Enter **http://localhost/EMP_DETAILS.html** in the Address bar and press **Enter**
 3. Enter **A05** in the Employee ID textbox
 4. Enter **Peter Taylor** in the Name box
 5. Select **IT** as the Department
 6. Enter **peterA05@infotech.com** in the Email box
 7. Click **SUBMIT**



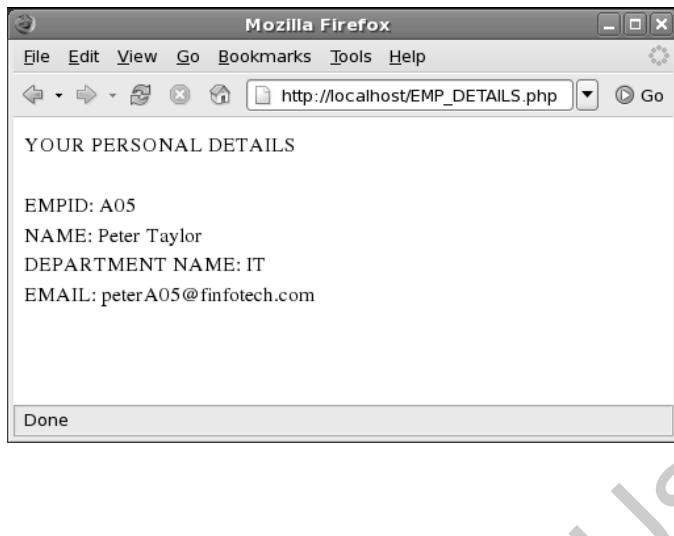
Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 24 of 32

Retrieving Data Using the POST Method

7-7

Displays the following output:



Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 25 of 32

Using slide 19, explain the process of retrieving the data using POST method. It is to be noted that all HTML Form elements, apart from the unmarked check boxes and radio buttons will emerge in the `$_POST` array. The null value is also stored in an array. The Form handler once checks all the fields that are available in the Form including the button that appears in POST array. After that, the testing is done by extracting the necessary values. Testing is nothing but validating the data. There are several types of testing. You can explain them the testing types and its uses correspondingly.

Using slides 20 and 23, explain the students the example code that describes the data retrieval using POST method. Explain the each line of code elaborately to the students. The execution process is same as explained earlier. Using slides 24 and 25, show the output for the code.

Slides 26 to 30

Let us see how to use Hidden Fields.

Using Hidden Fields

1-5

- ◆ **Hidden Field**

- ◆ Is embedded in the HTML source code of the form
- ◆ Enables the user to pass variables with values from one form to another without requiring to re-enter the information
- ◆ Contents cannot be viewed by the user

Syntax

```
<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

Using Hidden Fields

2-5

- ◆ Steps for passing the names of the continents in a PHP script using hidden fields are as follows:

1. Open a new file in the **gedit** text editor
2. Enter the code and save the file as **continent.html** in the **/usr/local/apache2/htdocs** directory

Snippet

```
<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>
```

Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 27 of 32

Using Hidden Fields

3-5

- ◆ Steps for creating a PHP script to retrieve and process the data entered in the HTML form are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **continent.php** in the `/usr/local/apache2/htdocs` directory

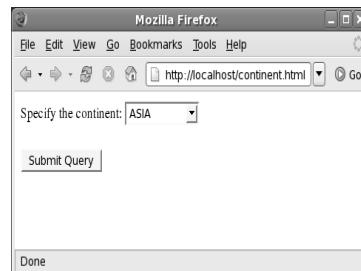
Snippet

```
<?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";
?>
```

Using Hidden Fields

4-5

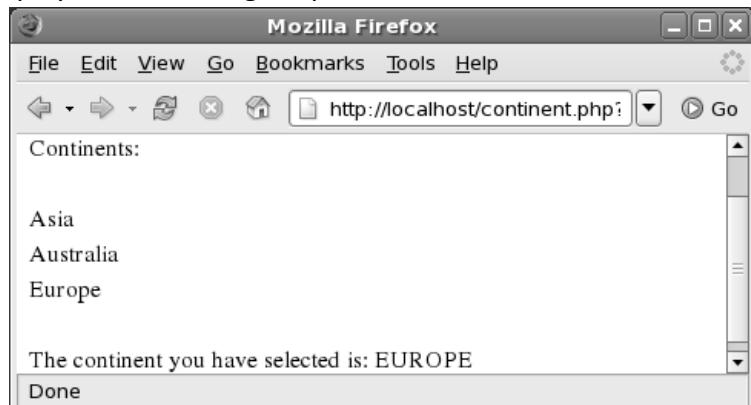
- ◆ Steps for displaying **continent.html** page are as follows:
 1. Open the Mozilla Firefox Web browser
 2. Enter `http://localhost/continent.html` in the address bar and press **Enter**
 3. Select the required continent from the drop-down menu
 4. Click **Submit Query**



Using Hidden Fields

5-5

Displays the following output:



Version 1.0 © Aptech Limited.

Form Handling in PHP / Session 2 / Slide 30 of 32

The hidden fields are given to specify the corresponding value that is not viewed by the user and directly retrieved to the output of the code. It transfers the hidden data from one Form to the other that is not re-entered into the Form.

The `<input>` tag is used to specify the hidden field. Explain them the general format using slide 26 and give them a brief view over the hidden fields with the example code given in slides 27 and 28.

Then, explain them the output by showing them slides 29 and 30. Also, show them the difference between the general Form field and the hidden field. It is to be noted that the hidden fields in HTML do not offer any security as the values are directly passed to another Form without encoding the data. As the values are directly specified, the PHP handler does not hold the hidden field values. So that the encoding of those values are not done at once.

Additional Information:

For more information on the hidden fields in PHP, visit the following links:

<http://www.html-form-guide.com/php-form/php-form-variables.html>

<http://www.echoecho.com/htmlforms07.htm>

Slides 31 and 32

Let us summarize the session.

Summary	1-2
<ul style="list-style-type: none"> ◆ 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 	
<small>Version 1.0 © Aptech Limited. Form Handling in PHP / Session 2 / Slide 31 of 32</small>	
Summary	2-2
<ul style="list-style-type: none"> ◆ 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 	
<small>Version 1.0 © Aptech Limited. Form Handling in PHP / Session 2 / Slide 32 of 32</small>	

Use slides 31 and 32 to list and explain the summary of this session.

The summary points are as follows:

- 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.

4.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to using variables and expressions in PHP.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 5 – Using Variables and Expressions in PHP

5.1 Pre-Class Activities

Prepare the background knowledge/summary to be discussed with students in the class. Familiarize yourself with the topics of this session in depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

5.1.1 Objectives

At the end of this session, the learners 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

5.1.2 Teaching Skills

To teach this session, you should be well-versed with identifiers, variables, constants, and data types in PHP. You must also be familiar with HTTP environment variables.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 2 of 42

Tell the students that they will be introduced to identifiers and learn the data types in PHP. Inform them that they will also be taught the use of variables and constants. In addition, they will also understand scope of variables and use of HTTP environment variables.

5.2 In-Class Explanations

Slide 3

Let us give an introduction to use variables and expressions in PHP.

Introduction

- ◆ A variable
 - ❖ Is a named memory location
 - ❖ Stores different types of data
 - ❖ Contains data that keep on changing during execution
- ◆ An expression is a combination of:
 - ❖ Variables
 - ❖ Constants
 - ❖ Functions
 - ❖ Operators

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 3 of 42

Using slide 3, give a brief introduction to variables and expressions.

The variables and expressions are the core of programming in general. A variable is defined as a named area of storage or somewhere you can store a value. Accordingly, a variable can be declared with different data types. Here, the variable name is case-sensitive.

Expressions are the most important building stones of PHP. An expression is a combination of constants and variables.

Describe each of these terms: variables, constants, functions, and operators.

Give some examples for variables such as, `student_name` (variable), `first_score` (variable), `productprice2014`, and so on. Tell them that examples of expressions can include the following:

```
marks1 + marks2  
costprice>sellingprice  
seats * seats
```

Additional Information:

For more information on variables in PHP, visit the following links:

<http://php.net/manual/en/language.variables.php>

http://www.tutorialspoint.com/php/php_variable_types.htm

<http://webcheatsheet.com/php/variables.php>

Slide 4

Let us learn about identifiers.

Identifiers

- Are names given to various elements of a program such as variables, constants, arrays, functions, and classes
- Rules to be followed while naming identifiers are as follows:
 - It must begin with a letter
 - It must contain only letters (A to Z) or digits (0-9)
 - It must not have any special characters including blank space
 - An underscore (_) character can be used to add space in the identifier to make it more readable
- Valid identifiers names are `firstnum`, `lname`, `net_sal`, `add8num`, and `NewNum`

Identifier is the term applied to variables, functions, and various other user-defined objects. There are several rules that PHP identifiers must abide by. These rules are listed in slide 4. Give the students examples about identifiers that satisfy all the rules specified in slide 4.

Slides 5 to 7

Let us see the variables and data types.

Variables and Data Types

1-3

- ◆ Variable is an identifier whose value keeps changing throughout the execution of a program
- ◆ Variable has:
 - ❖ **Name** - refers to the variable
 - ❖ **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
 - ❖ Values returned by the functions

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 5 of 42

Variables and Data Types

2-3

- ◆ Rules to be followed for a variable are as follows:
 - ❖ Its not mandatory to declare a variable before assignment
 - ❖ A variable is automatically declared at the time of initialization
 - ❖ A variable is of the same data type as the value stored in it
 - ❖ Variable name is preceded by a dollar sign (\$) and can only contain alpha-numeric characters and underscores
 - ❖ Value to the variable is assigned with the equal to (=) operator, with the variable on the left side and the expression on the right side
 - ❖ A variable created without any value assigned to it, takes the default value of NULL
 - ❖ A variable is case-sensitive

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 6 of 42

Variables and Data Types

3-3

- ◆ PHP supports primitive data types that are categorized as follows:
 - ◆ Scalar Types
 - ◆ Integer
 - ◆ Float
 - ◆ String
 - ◆ Boolean
 - ◆ Compound Types
 - ◆ Array
 - ◆ Object
 - ◆ Special Types
 - ◆ Resource
 - ◆ Null
 - ◆ Constants

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 7 of 42

Slides 5 to 7 describe about the variables and data types. Slide 5 describes in detail what constitutes a variable. Besides satisfying the rules for an identifier, variables must also satisfy few other rules. These are listed in slide 6. Also, tell the students that there are some special and predefined variable names that cannot be used. These are called reserved words. List and describe a few such reserved words. Tell them that variables are highly case sensitive.

Using slide 7, tell the students about the definition of a data type and the scope of using data types. Explain them the different types of data that are available. Also, explain them the categorized set of data types. Tabulate the data types with the attributes, data type, name, and description.

Additional Information:

For more information on reserved words, visit the following link:

<http://php.net/manual/en/reserved.keywords.php>

In-Class Question:

After you finish explaining about variables and data types, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What is meant by a variable?

Answer:

A variable is an identifier whose value keeps changing throughout the execution of a program.

Slide 8

Let us understand the Scalar Types.

Scalar Types

- ◆ **Integer**
 - ◆ Stores whole numbers without decimal points
 - ◆ Range from -2,147,483,648 to +2,147,483,647
- ◆ **Float**
 - ◆ Data type size is 8 bytes
 - ◆ Range from -2.2E-308 to 1.8E+308
 - ◆ Can include a decimal point, +/- sign, and an exponential value
- ◆ **String**
 - ◆ Is a sequence of characters
 - ◆ Is enclosed within single quotes or double quotes
- ◆ **Boolean**
 - ◆ Stores one of the two values, True or False

Version 1.0. © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 8 of 42

Slide 8 describes scalar data types. They are the integer, float, string, and boolean types. An integer denotes the whole numbers, which corresponds to both positive and negative. The format of the integer data types extends to decimal, octal, and hexadecimal. Similarly, define the float, string, and boolean types.

Tell them the base value of each format and give them some examples of values of each of these types.

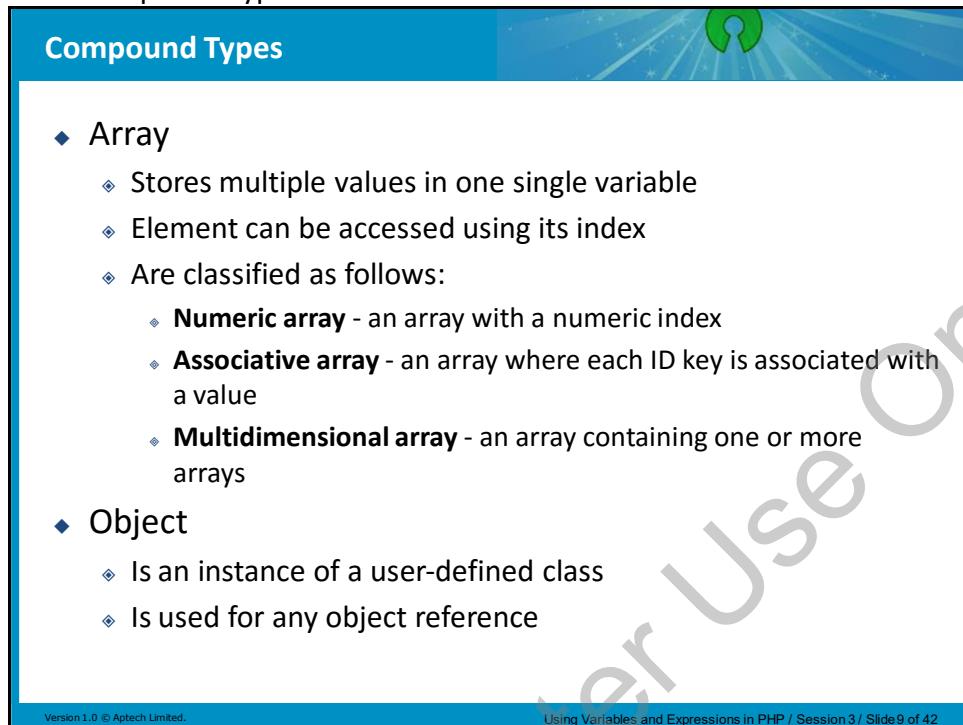
Additional Information:

For more information on scalar data types, visit the following links:

<http://php.net/manual/en/language.types.intro.php>
<http://php.net/manual/en/language.types.integer.php>
<http://php.net/manual/en/language.types.boolean.php>
<http://php.net/manual/en/language.types.float.php>
<http://php.net/manual/en/language.types.string.php>

Slide 9

Let us understand compound types.



The slide has a blue header bar with the title "Compound Types". The main content area contains two bullet points: "Array" and "Object". The "Array" point has three sub-points: "Stores multiple values in one single variable", "Element can be accessed using its index", and "Are classified as follows:". Under "Are classified as follows:", there are three more sub-points: "Numeric array - an array with a numeric index", "Associative array - an array where each ID key is associated with a value", and "Multidimensional array - an array containing one or more arrays". The "Object" point has two sub-points: "Is an instance of a user-defined class" and "Is used for any object reference". At the bottom of the slide, there is a footer bar with the text "Version 1.0 © Aptech Limited." on the left and "Using Variables and Expressions in PHP / Session 3 / Slide 9 of 42" on the right.

- ◆ Array
 - ◆ Stores multiple values in one single variable
 - ◆ Element can be accessed using its index
 - ◆ 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
 - ◆ Is an instance of a user-defined class
 - ◆ Is used for any object reference

Using slide 9, explain the students about compound types of data that are classified as array and object. An array in PHP is actually an ordered map. A map is a type that associates values to keys. For example to store 50 numbers, it is easy to define an array of 50 length instead of defining 50 variables. Explain them the three different kinds of arrays and the term array index. Using slide 9, explain about the object that is an instance of a user-defined class.

Additional Information:

For more information on compound data types, visit the following links:

<http://php.net/manual/en/language.types.intro.php>
<http://php.net/manual/en/language.types.array.php>
<http://php.net/manual/en/language.types.object.php>

Slide 10

Let us understand special types.

The screenshot shows a slide titled "Special Types" with a blue header and a decorative background. The slide content is organized into two main sections:

- ◆ **Resource**
 - ◆ Hold references to resources external to PHP, such as:
 - ◆ Database connections
 - ◆ Database query
 - ◆ Open file
 - ◆ Other external types
- ◆ **NULL**
 - ◆ Is a variable with NULL value
 - ◆ A variable is considered to be NULL in following cases:
 - ◆ Variable has been assigned the constant value NULL
 - ◆ Variable has not been set to any value yet
 - ◆ Variable has been `unset()`

At the bottom of the slide, there are two small text elements: "Version 1.0 © Aptech Limited." on the left and "Using Variables and Expressions in PHP / Session 3 / Slide 10 of 42" on the right.

In addition to the data types discussed so far, there are also special types supported in PHP. Use slide 10 to explain about special data types. A resource is a special type of variable that holds a reference to an external resource. These resources are created and used by special functions. Also, explain them about other external types.

The special NULL value represents a variable with no value. NULL is the only possible value of type null. Explain the students the use of these special types.

Additional Information:

For more information on special data types in PHP, visit the following links:

<http://php.net/manual/en/language.types.intro.php>
<http://php.net/manual/en/language.types.resource.php>
<http://php.net/manual/en/language.types.null.php>

Slides 11 to 15

Let us understand working with variables.

Working with Variables
1-5

Initializing a variable

Syntax

```
$variable_name = value;
```

Where,

- ❖ **\$variable_name** - specifies the name of the variable
- ❖ **value** - specifies the value the variable will store

Version 1.0 © Aptech Limited.
Using Variables and Expressions in PHP / Session 3 / Slide 11 of 42

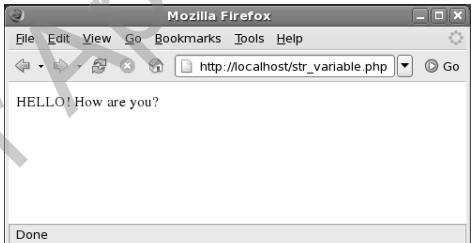
Working with Variables
2-5

- ❖ Storing a string value in a variable
 - ❖ Enter the code and save it in a script named **str_variable.php**

Snippet

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

Displays the following output:



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.

Version 1.0 © Aptech Limited.
Using Variables and Expressions in PHP / Session 3 / Slide 12 of 42

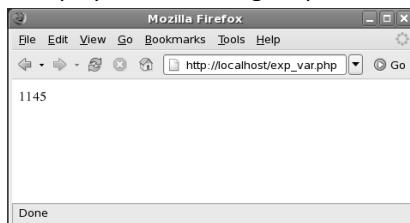
Working with Variables**3-5**

- ◆ Storing the value of an expression in a variable
 - ❖ Enter the following code and save in a script named **exp_var.php**

Snippet

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

Displays the following output:



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.

Working with Variables**4-5**

- ◆ To assign a value to a variable by referencing another variable

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

Working with Variables

5-5

- ◆ Assigning the value of one variable to another variable
 - ❖ Enter the following code and save in a script named **exp_var.php**

Snippet

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

Displays the following output:



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.

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 15 of 42

Using slides 11 and 12, explain how to initialize a variable. A variable name is the one that can be set and used dynamically. It is said that a variable is a holder for a type of data. So, based on the variable type, it can hold numbers, strings, booleans, objects, resources, or it can even be NULL. In PHP, all variables begin with a dollar sign '\$' and the value can be assigned using the '=' operator. The '\$' symbol is not technically part of the variable name, but it is required as the first character for the PHP parser to recognize the variable as such.

It is important to note that in PHP, all the statements must end with a semicolon '. In PHP, the variable type need not to be specified. This is because it takes the data type of the assigned value. To declare a variable, the data type must be included in the script. In a single statement, a variable can be declared and assigned to a value.

Using slides 13 to 15, explain them the code that describes the variable declaration and about assigning a value to a variable.

Slides 16 to 19

Let us understand constants.



1-4

Constants

- ◆ Constants
 - ❖ Are identifiers containing values that do not change throughout the execution of a program
 - ❖ Are case-sensitive
 - ❖ Are static, meaning constants once defined cannot be changed
 - ❖ Are declared using the `define()` function
 - ❖ Are accessed from anywhere in the script

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 16 of 42



2-4

Constants

- ◆ Declaring a Constant using `define()` function

Syntax

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

Where,

- ❖ **string_name** - defines the variable name for the constant
- ❖ **mixed_value** - specifies a numeric or string value

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 17 of 42

Constants**3-4**

- ◆ Declaring the constant, **NAME**, containing a string value
 - ❖ Enter the code as shown, in a script named **dec_con.php**

Snippet

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

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 18 of 42

Constants**4-4**

Displays the following output:



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.

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 19 of 42

In general, a constant is said to be an identifier for a fixed value. The label and the constant name in PHP follow the same rules as for identifiers. There is a difference between constants and variables; it is that the constant value cannot be changed during the course of a program. The constants may be of mathematical constants, passwords, paths to files, and so on. By default, a constant is case sensitive. By convention, constant identifiers are always uppercase.

Using slides 16 and 17, explain to the students about the constants and the general format to declare a constant.

Using slides 18 and 19, explain them the lines of code that describes declaration and use of a constant.

Slide 20

Let us see the scope of variables.

The slide has a blue header bar with the title 'Scope of Variables'. The main content area contains a bulleted list explaining what scope is and the different types it can have. At the bottom, there is a footer bar with copyright information.

Scope of Variables

- ◆ Is the context within which a variable is defined
- ◆ Is the lifetime of a variable from the time the variable is created to the time the execution of the script ends
- ◆ Different scopes that a variable can have are as follows:
 - ◆ Local
 - ◆ Global
 - ◆ Static

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 20 of 42

The scope of a variable is the context within which it is defined. This scope spans included and required files as well. Tell them that any variable that is used inside a function is, by default, limited to the local function scope. In PHP, global variables must be declared global inside a function if they are going to be used in that function.

Using slide 20, explain the students about the scope of the variables and its types.

Slides 21 to 23

Let us learn about local variables.

Local Variables
1-3

- ◆ Local variable
 - ❖ Is a variable initialized and used inside a function
 - ❖ Is similar to a normal variable
 - ❖ Is declared inside a function
 - ❖ Its lifetime begins when the function is called, and ends when the function is completed

Version 1.0 © Aptech Limited.
Using Variables and Expressions in PHP / Session 3 / Slide 21 of 42

Local Variables
2-3

- ◆ Displaying the use of local variables

Snippet

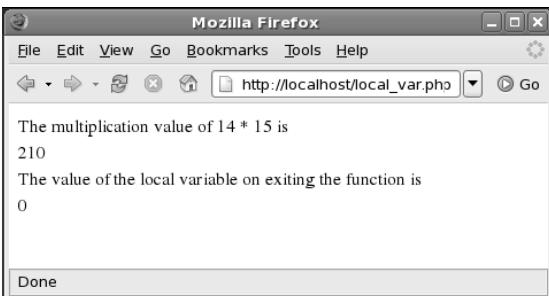
```
<?php
$num2 = 0;
echo "The multiplication value of 14 * 15 is <br>";
function multiply()
{
$num1=14;
$num2=15;
$num2=$num1 * $num2;
echo $num2;
}
multiply();
echo " <br> The value of the local variable on exiting the function is
<br>";
echo $num2;
?>
```

Version 1.0 © Aptech Limited.
Using Variables and Expressions in PHP / Session 3 / Slide 22 of 42

Local Variables

3-3

Displays the following output:



The multiplication value of `14 * 15` is
210
The value of the local variable on exiting the function is
0

Done

The variable `num1` and `num2` are declared as local variables inside the `multiply()` function

The `multiply()` function is executed when PHP script invokes the function

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 23 of 42

The variable that is declared in a function is considered as local. It can be referenced solely in that function. Here, any assignment that is outside of the function will be considered to be an entirely different variable from the one contained in the function. Using slide 21, describe about local variables.

Then, using slides 22 and 23, explain them the lines of code and its functionality. Show them corresponding output for the code that is given in slide 23. Here, the variables `num1` and `num2` are considered as local variables as they are declared in the `multiply()` function.

In-Class Question:

After you finish explaining scope of variables, you will ask the students an In-Class question.

This will help you in reviewing their understanding of the topic.



Can you describe about the scope of the variables?

Answer:

- A variable is defined within the context.
- Its lifetime of a variable from when the variable is created to the time when execution of script ends.

Slides 24 to 26

Let us understand global variables.

Global Variables

1-3

- ◆ Global Variable
 - ❖ Is a variable retaining its value throughout with the lifetime of a Web page
 - ❖ Is declared within the function using global keyword
 - ❖ Is accessed from any part of the program
- ◆ Declaring a global variable

Syntax

```
global $var_name;
```

Where,

- ❖ **\$var_name** - defines the global variable name

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 24 of 42

Global Variables

2-3

- ◆ Displaying multiplication of two numbers using global variables
 - ❖ Enter the code in a script named **multi_global.php**

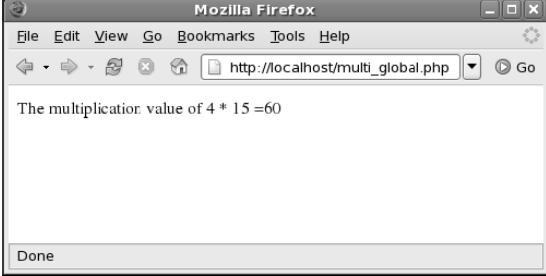
Snippet

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

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 25 of 42

Global Variables 3-3

Displays the following output:



The multiplication value of 4 * 15 =60

Done

The variables **var1** and **var2** are initialized outside the function and declared as global variables within the **multiply()** function

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 26 of 42

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished by placing the keyword GLOBAL in front of the variable. This is evident when you explain the code given in slide 25, after first defining about the global variables using slide 24. The output for the given code is that the multiplied value of the global variables. Thus, `var1` and `var2` are said to be global.

Show the students the output of the corresponding code using slide 26.

Slides 27 to 29

Let us understand static variables.

Static Variables

1-3

◆ Static Variables

- ❖ Retains its value even after the function terminates
- ❖ Are only accessible from within the function they are declared and their value remains intact between function calls
- ❖ Can be initialized during declaration and the static declarations are resolved during compile time
- ❖ Are commonly used in the recursive functions

Syntax

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

- ❖ **var_name** - defines the variable name
- ❖ **value** - specifies the value of the variable

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 27 of 42

Static Variables

2-3

◆ Illustrating the use of a static variable

- ❖ Enter the code as shown in a script named **stat_var.php**

Snippet

```
<?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();
?>
```

Version 1.0 © Aptech Limited.

Using Variables and Expressions in PHP / Session 3 / Slide 28 of 42

Static Variables

3-3

Displays the following output:

The value of the variable is : 9
The addition value of 9 + 12 = 21

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. The variable `var1` is local to the `sum()` function but retains its value throughout the program.

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 29 of 42

Using slides 27 to 29, explain the students in brief about static variables.

Declaring class properties or methods as static makes them accessible without needing an instantiation of the class. A property declared as static cannot be accessed with an instantiated class object; though a static method can be accessed.

Like any other PHP static variable, static properties may only be initialized using a literal or constant. Here, expressions are not allowed.

Slides 30 and 31

Let us understand HTTP environment variables.

HTTP Environment Variables 1-2

- ◆ Environment variables are:
 - ❖ System-defined variables
 - ❖ Similar to the user-defined variables and begin with dollar (\$) sign

- ◆ Environment variables provide information about:
 - ❖ Transactions between the client and the server
 - ❖ HTTP request or response

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 30 of 42

HTTP Environment Variables 2-2

- ◆ Environment variables are as follows:
 - ❖ SERVER_NAME
 - ❖ SERVER_PROTOCOL
 - ❖ SERVER_PORT
 - ❖ \$_COOKIE
 - ❖ HTTP_USER_AGENT
 - ❖ HTTP_ACCEPT
 - ❖ HTTP_FROM

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 31 of 42

An associative array of variables is passed to the current script via the environment method. These variables are imported into PHP's global namespace from the environment under which the PHP parser is running. Using slides 30 and 31, explain the students about environment variables in brief.

Slide 32

Let us understand the SERVER_SOFTWARE environment variable.

SERVER_SOFTWARE

- ◆ Server identification string specified in the headers when responding to requests
- ◆ Returns the name and the version of the server software

Snippet

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

- ◆ Displays the following output:

A screenshot of a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The address bar shows "http://localhost/server_software.php". The main content area displays the text "Apache/2.2.17 (Unix) PHP/5.3.6". At the bottom of the browser window, there is a "Done" button.

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 32 of 42

Using slide 32, explain the students about the SERVER_SOFTWARE environment variable. Tell them that `$_SERVER` is an array containing information such as headers, paths, and script locations. The entries in this array are created by the Web server. Not all Web servers provide these, some servers may omit them.

Additional Information:

For more information on SERVER_SOFTWARE environment variable, visit the following links:

<http://www.techrecite.com/10-essential-server-variables-php-developer-must-know/>
<http://www.webtrafficexchange.com/php-server-environment-variable>

Slide 33

Let us understand the SERVER_NAME environment variable.

The slide has a blue header bar with the title 'SERVER_NAME'. Below the header, there is a bulleted list defining the SERVER_NAME environment variable. A 'Snippet' box contains a PHP code example. To the right, a screenshot of a Mozilla Firefox browser window shows the output of the code.

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

Snippet

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

Output:

A screenshot of a Mozilla Firefox browser window. The address bar shows 'http://localhost/server_name.php'. The main content area displays the word 'localhost'. At the bottom, a status bar shows 'Done'.

Using slide 33, explain the students about the SERVER_NAME environment variable. It states the name of the host server. Explain them the code snippet that shows the corresponding server name. If you run the code given on slide 33 on localhost, by default, it will show the server name as localhost. If the script is running on a virtual host set, it will return the server name.

Additional Information:

For more information on SERVER_NAME environment variable, visit the following links:

<http://www.techrecite.com/10-essential-server-variables-php-developer-must-know/>
<http://www.webtrafficexchange.com/php-server-environment-variable>

Slide 34

Let us understand the SERVER_PROTOCOL environment variable.

SERVER_PROTOCOL

- ◆ Returns the name and version number of the protocol via which the page was requested

Snippet

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

- ◆ Displays the following output:

The screenshot shows a Mozilla Firefox window with the URL http://localhost/server_protocol.php in the address bar. The page content displays "HTTP/1.1" and a "Done" button at the bottom. The browser interface includes standard menu options like File, Edit, View, Go, Bookmarks, Tools, and Help.

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 34 of 42

Using slide 34, explain the students about the SERVER_PROTOCOL. Describe the code snippet along with the output shown on slide 34.

Additional Information:

For more information on SERVER_PROTOCOL environment variable, visit the following links:

<http://www.techrecite.com/10-essential-server-variables-php-developer-must-know/>
<http://www.webtrafficexchange.com/php-server-environment-variable>

Slide 35

Let us understand the SERVER_PORT environment variable.

SERVER_PORT

- ◆ Returns the server port number for the script

Snippet

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

- ◆ Displays the following output:

The screenshot shows a Mozilla Firefox browser window. The address bar contains "http://localhost/server_port.php". The main content area displays the number "80". At the bottom of the browser window, there is a "Done" button.

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 35 of 42

Use slide 35 to explain the students about the SERVER_PORT environment variable. It returns the server port number for the script. Explain the given code snippet as well as the output shown on slide 35.

Additional Information:

For more information on SERVER_PORT environment variable, visit the following links:

<http://www.techrecite.com/10-essential-server-variables-php-developer-must-know/>
<http://www.webtrafficexchange.com/php-server-environment-variable>

Slides 36 to 38

Let us understand the `$_COOKIE` environment variable.

\$_COOKIE

1-3

- ◆ Returns the content of the recently used cookie which are:
 - ❖ Saved as a text file
 - ❖ Sent back to the server when the browser requests to display a page

Snippet

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

◆ Displays the following output:

The cookie has been set.

Done

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 36 of 42

\$_COOKIE

2-3

- ◆ Retrieving the value of the cookie

Snippet

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

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 37 of 42

3-3

\$_COOKIE

Displays the following output:

Welcome back!
Your name is Jerry

Done

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 38 of 42

Using slide 36, explain the students about `$_COOKIE`. Use an example to describe it more appropriately. An associative array of variables is passed to the current script via HTTP Cookies. These are bits of data that a Web browser stores on your visitor's computer.

Using slides 37 and 38, explain the students the code snippet that is given for retrieving the value of the cookie. It is used to store things such as your visitor's preferences or login data or other things that are specific to a particular visitor.

Additional Information:

For more information on `$_COOKIE`, visit the following links:

<http://php.net/manual/en/reserved.variables.php>

<http://webcheatsheet.com/php/variables.php>

Slide 39

Let us learn about HTTP_USER_AGENT environment variable.

HTTP_USER_AGENT

- ◆ The following code returns the name of the browser to the client:

Snippet

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

Displays the following output:

Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.9) Gecko/20070118
Red Hat/1.5.0.9-10.el5 Firefox/1.5.0.9 pango-text

Done

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 39 of 42

Using slide 39, explain the students the code returns the name of the browser to the client. Show them the output that is displayed in the browser window.

Additional Information:

For more information on HTTP_USER_AGENT, visit the following links:

http://php.about.com/od/learnphp/p/http_user_agent.htm

Slide 40

Let us learn about HTTP_ACCEPT environment variable.

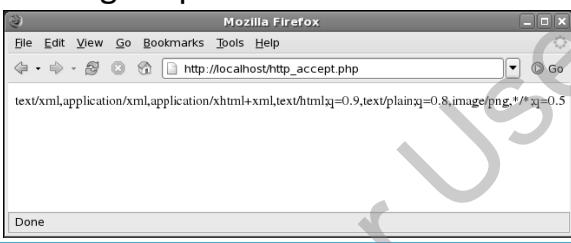
HTTP_ACCEPT

- ◆ Returns the contents of the Accept: header if there is a current request
- ◆ Lists the media types the client will accept

Snippet

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

Displays the following output:



Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/http_accept.php
text/xml,application/xml,application/xhtml+xml+xml;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Done

Version 1.0 © Aptech Limited. Using Variables and Expressions in PHP / Session 3 / Slide 40 of 42

Slide 40 describes the HTTP_ACCEPT that returns the contents of the Accept. Explain them the code snippet along with the corresponding output that is displayed.

Additional Information:

For more information on HTTP_ACCEPT, visit the following links:

<http://php.net/manual/en/reserved.variables.server.php>

[http://www.w3resource.com/php/super-variables/\\$_SERVER.php](http://www.w3resource.com/php/super-variables/$_SERVER.php)

Slides 41 and 42

Let us summarize the session.

Summary**1-2**

- ◆ 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

Summary**2-2**

- ◆ 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

Use slides 41 and 42 to list and explain the summary of this session.

The summary points are as follows:

- 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.

5.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to PHP operators.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 7 – Scalar Type Declarations

7.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

7.1.1 Objectives

By the end of this session, learners 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.

7.1.2 Teaching Skills

To teach this session, you should be well versed with the 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.

You should teach the concepts in the class using the code snippet provided. For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ 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.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 2 of 26

7.2 In-Class Explanations

Slide 3

Let us understand the scalar data types.

Scalar Data Type

- ◆ The data types that hold single data type are known as scalar data types.
- ◆ The data types can be:

Integer
Float
Boolean
Strings

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 3 of 26

Using slide 3, explain that scalar data types are data types that hold single values and then list the scalar types shown.

Slide 4

Let us now understand the meaning of type declarations in PHP.

Type Declarations in PHP

- ◆ Refers to specifying the data type of a parameter in a function
- ◆ Also referred to as type hinting
- ◆ Provides hints to a function
- ◆ Enforces the input the parameter data type, that can be either strict or coercive

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 4 of 26

Using slide 4, explain that type declarations refers to specifying the data type of the parameter while passing it to the function. A function is a program code that can take one or more inputs as a parameter and does some processing. After processing, a function may return a value to the calling program. Once a function is written, it can be invoked any number of times as per the requirements. In other words, it is specifying the data type of a parameter in the function. Type declaration is also known as type hinting.

Add further that, 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.

Slide 5

Let us now understand how to explicitly declare scalar types.

The screenshot shows a slide titled "Explicit Declarations of Scalar Types". It contains a bulleted list: "◆ Declaring a new function". Below this is a "Syntax" section with a code example: "function function_name(type, p1)". A note below the code says "Where," followed by a list: "◆ **function** – is a reserved word", "◆ **type** – is the data type of the function parameter", and "◆ **p1** – is the parameter". At the bottom of the slide, it says "Version 1.0 © Aptech Limited." and "Programming for the Web with PHP / Session 7 / Slide 5 of 26".

Using slide 5, display the syntax to declare a new function.

Read out the syntax and explain that each parameter is different 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` and `p1` is the parameter.

Further explain that, 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.

Slides 6 to 15

Let us now understand how to declare and use scalar types.

Declaring and Using Scalar Types 1-10

- ◆ Displaying the value of a parameter in `int` data type

Snippet

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

◆ `test1` - is the function name
◆ `int` - is the data type
◆ `$x` - is the parameter

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 6 of 26

Declaring and Using Scalar Types 2-10

Displays the following output:

The screenshot shows a web browser window with the URL `http://localhost/cs7-1.php`. The page content displays the text `integer $x = 10`. The browser interface includes a back button, forward button, search bar, and other standard navigation controls.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 7 of 26

Declaring and Using Scalar Types

3-10

- ◆ Displaying the value of a parameter in `float` data type

Snippet

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

- ◆ `test1` – is the function name
- ◆ `float` – is the data type
- ◆ `$x` – is the parameter

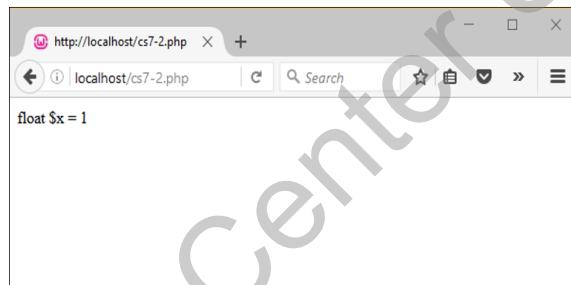
Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 7 / Slide 8 of 26

Declaring and Using Scalar Types

4-10

Displays the following output:



Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 7 / Slide 9 of 26

Declaring and Using Scalar Types

5-10

- ◆ Displaying the value of a parameter in `boolean` data type

Snippet

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

- ◆ `test1` - is the function name
- ◆ `bool` – is the data type
- ◆ `$a` – is the parameter

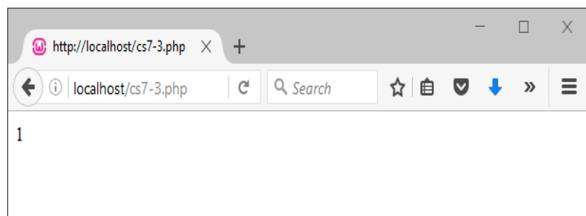
Version 1.0 © Aptech Limited.

Programming for the Web with PHP / Session 7 / Slide 10 of 26

Declaring and Using Scalar Types

6-10

Displays the following output:

**Declaring and Using Scalar Types**

7-10

- ◆ Displaying the value of a parameter in string data type

Snippet

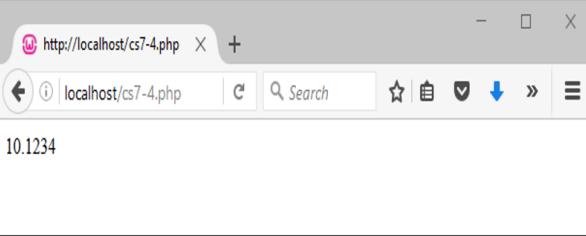
```
<HTML>
<BODY>
<?php
    function test1(string $a) {
        echo $a;
    }
    test1(10.1234);
?>
</BODY>
</HTML>
```

- ◆ test1 – is the function name
- ◆ string – is the data type
- ◆ \$a – is the parameter

Declaring and Using Scalar Types

8-10

Displays the following output:



Declaring and Using Scalar Types  **9-10**

- ◆ Demonstrating strict type declaration

Snippet

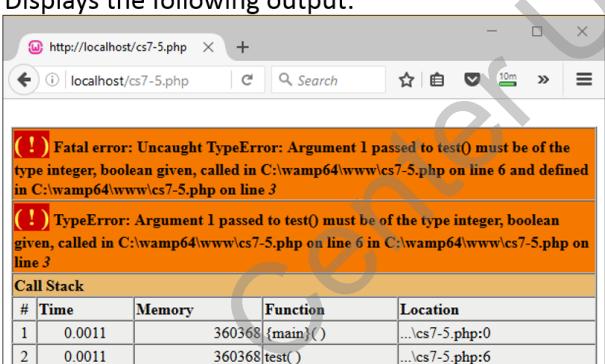
```
<?php declare(strict_types=1);
function test1(int $a){
    echo $a;
}
test1(true);
```

The `declare` statement explicitly declares the scalar type, that are strictly checked.
The function `test1` is the function name that has an `int` data type.
The `echo` statement prints the value of the argument.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 14 of 26

Declaring and Using Scalar Types  **10-10**

Displays the following output:



(!) Fatal error: Uncaught TypeError: Argument 1 passed to test() must be of the type integer, boolean given, called in C:\wamp64\www\cs7-5.php on line 6 and defined in C:\wamp64\www\cs7-5.php on line 3
(!) TypeError: Argument 1 passed to test() must be of the type integer, boolean given, called in C:\wamp64\www\cs7-5.php on line 6 in C:\wamp64\www\cs7-5.php on line 3

Call Stack				
#	Time	Memory	Function	Location
1	0.0011	360368	{main}()	...cs7-5.php:0
2	0.0011	360368	test()	...cs7-5.php:6

The program fails to execute and terminates prematurely because the argument types does not match the parameter type.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 15 of 26

Tell the students slides 6 to 15 show a few examples of using scalar types. Explain that 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.

Let us now take a look at the first example where the value of the parameter is `int` data type.

Using slide 6, display the syntax and give the examples.

Explain that in the example a function is defined with the 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`. Using slide 7, display the output.

Now, let us take a look at another example where the value of the data type is `float`.

Using slide 8, explain that 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. Using slide 9, display the output of the syntax.

Now, let us look at the third example where the parameter has a `boolean` data type. Using slide 10, explain that 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 slide 11. The result is `true` because we are passing a value `10.34` and it is converted to boolean value, `true`.

Let us now see another example where the value of the parameter is `string` data type.

Using slide 12, explain that a function `test1` is defined, 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. Using slide 13, display the output.

Now, let us understand strict type declarations.

Using slide 14, display the syntax for strict type declaration and explain that 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.

Using slide 15, show the error that is displayed after executing the code.

In-Class Question:

After you finish explaining the use and the method to declare scalar data types, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What are the different scalar data types?

Answer:

Scalar data types are types that can hold more than one data type. The data types are `integer`, `boolean`, `string`, and `float`.

Slides 16 and 17

Let us now understand the weak type checking mode and the various type conversions that can be used.

Behavior of Weak Type Checking and Type Conversion 1-2

- ◆ Weak type conversion is enforced using the `declare(strict_types=0)` statement
- ◆ Rules to be considered while using weak type checking:
 - Calls to a built-in PHP function or to an extension have the same behavior as earlier versions
 - Weak type new scalar type declarations are same as that of built-in PHP function or to an extension
 - NULL is not accepted unless it is a parameter and is explicitly given a default value

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 16 of 26

Behavior of Weak Type Checking and Type Conversion 2-2

- ◆ An implicit scalar conversion in a 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

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 17 of 26

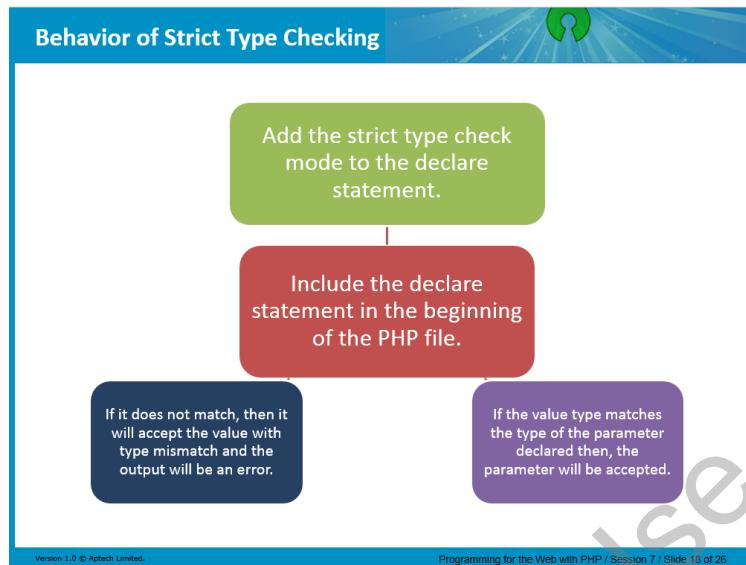
Using slide 16, explain that by using the `declare(strict_types=0)` statement, weak type checking mode can be enforced. However, there are few points to be considered while using this:

- 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.

Using slide 17, display the implicit scalar conversions types that are possible in a weak mode.

Slide 18

Let us now understand the behavior of strict type checking.



Using slide 18, explain 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.

Further add that 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`.

Slide 19

Let us now understand about anonymous classes.

The diagram shows two blue circles connected by a line, representing an object. To the right of the top circle is a red box containing the text "When no documentation of the class is required". To the right of the bottom circle is a blue box containing the text "When the class is used only once while execution".

Anonymous Class

- ◆ Is a class that does not have a name
- ◆ Can be used in the following scenarios:

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 19 of 26

Using slide 19, define the term anonymous class and explain scenarios when they can be used. Say that an anonymous class is also a kind of class, but without a name. There is no difference between a named class and anonymous class in terms of object functionality.

Slides 20 to 24

Let us now understand how to create anonymous classes.

The diagram shows two blue boxes labeled "Step 1" and "Step 2". To the right of Step 1 is a blue box containing the text "Use the keyword class before the class name". To the right of Step 2 is a blue box containing the text "Enclose the property and method definitions within curly braces". Below these is a code block showing the syntax for creating an anonymous class.

Creating Anonymous Classes

1-5

- ◆ Demonstrating creation of a named class

Step 1 Step 2

Use the keyword `class` before the class name
Enclose the property and method definitions within curly braces

```
class class_Name {
    // defined properties and methods
};

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

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 20 of 26

Creating Anonymous Classes

2-5

Step 1	Create a public variable.
Step 2	Assign values to the public variables.
Step 3	Create a function within the anonymous class without any arguments.
Step 4	Create a second function within the anonymous class with one argument.

The first function will return a message to the user whereas the second function will return the value of the argument.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 21 of 26

Creating Anonymous Classes

3-5

Snippet

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

$anon_class_obj = new class{
    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;
    }
};
```

\$greeting, \$Id, and SETT are members of the anonymous class.
 'hello', 754, and 'some configuration' are the values assigned to these members.
 getValue and getValueWithArgument are the functions within the class.
 The getValue function will display the message 'Some Returned Value'
 The getValueWithArgument function will display the value of its argument.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 22 of 26

Creating Anonymous Classes

4-5

Snippet

```
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>';
?>
```

The echo statements will display the values of the members and functions of the anonymous class.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 23 of 26

Creating Anonymous Classes 5-5

Displays the following output:

http://localhost/cs7-6.php

localhost/cs7-6.php

Search

hello
754
some configuration
some returned value
returned value is Aptech

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 24 of 26

Explain that 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. Using slide 20, display the syntax to create a named class.

Using slide 21, display the steps involved in creating an anonymous class. Explain that an anonymous class is similar to a named class. The primary difference is that, an anonymous class is instantiated without a name.

Let us now take an example to see how to display the values of the variables of the anonymous class. Using slide 22, display the syntax to create anonymous class.

Explain that in the syntax the 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.

Using slide 24, display the output.

In-Class Question:

After you finish explaining the various type comparisons, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you define an anonymous class?

Answer:

A class that does not have a name.

Slides 25 and 26

Summarize the session.

Summary  **1-2**

- ◆ Scalar data types are those types that hold single values. These data types can either be integer, string, Boolean, or float.
- ◆ Type declaration refers to process of specifying the data type of the parameter when passing it to a function. They are also known as type hinting.
- ◆ Scalar type hints help design reliable PHP code.
- ◆ To enable strict type checking, use the declare statement to declare strict_types directive. Any type mismatch with function arguments results in an error.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 25 of 26

Summary  **2-2**

- ◆ An anonymous class is a class that is defined without a name.
- ◆ To create anonymous class, combine new class (\$constructor, \$args) followed by a standard class definition.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 7 / Slide 26 of 26

Using slides 25 and 26, explain each of the following points in brief t:

- Scalar data types are data types that hold single values and the different types are integer, boolean, strings, and float.
- Type declarations refers to specifying the data type of a parameter in the function and is also known as type hinting.
- The syntax to declare a new function is `declare (string_type=1);`. 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.
- The `declare (strict_types=0)` statement enables to enforce weak type checking mode and the `declare (strict_types=1)` enables to enforce strict type checking mode.
- An anonymous class is also a kind of class without a name. There is no difference between a named class and anonymous class in terms of object functionality.

7.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session.

For Aptech Center Use Only

Session 8 – PHP Operators

8.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

8.1.1 Objectives

At the end of the session, the learners 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

8.1.2 Teaching Skills

To teach this session, you should be well-versed with the use of arithmetic operators and logical operators in PHP. You should be able to explain all other operators associated with the operator types in PHP. Also, tell them the use of increment and decrement operators. Describe them about the use of conditional or ternary operators and also the precedence of operators.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 2 of 44

Tell the students that they will be introduced to operators in PHP and learn the use of increment and decrement operators. In addition, they will also understand the precedence of operators in detail.

8.2 In-Class Explanations

Slide 3

Let us introduce operators in PHP.

Introduction

- ◆ All programming languages use operators
- ◆ Operators
 - ◆ Are pre-defined symbols that perform specific actions on objects called operands
 - ◆ Are assigned a precedence value indicating the order in which the operators are evaluated

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 3 of 44

Using slide 3, give a brief introduction to operators in PHP.

An operator is something that takes one or more values and yields another value, so that the construction itself becomes an expression. In general, the operators are used to perform operations on variables and values.

Additional Information:

For more information on operators in PHP, visit the following links:

<http://php.net/manual/en/language.operators.php>

http://www.tutorialspoint.com/php/php_operator_types.htm

http://www.w3schools.com/php/php_operators.asp

Slide 4

Let us learn about classification of operators.

The slide has a blue header bar with the title "Classification of Operators". The main content area is white with a decorative blue and green starburst graphic in the top right corner. A large, semi-transparent watermark reading "For Aptech Center Only" is diagonally across the slide. At the bottom, there is a thin blue footer bar with small text: "Version 1.0 © Aptech Limited" on the left and "PHP Operators / Session 5 / Slide 4 of 44" on the right.

- ◆ Operators are classified as follows:
 - ❖ **Unary Operator** - acts on only one operand in an expression
 - ❖ **Binary Operator** - has two operands and performs different arithmetic and logical operations
 - ❖ **Conditional or Ternary Operator** – has three operands and evaluates the second or third expression depending on the result of the first expression

Operators can be categorized according to the number of values they take. Unary operators take only one value while the binary operators take two values, such as addition operator (+) and subtraction operator (-). In addition, there is a ternary operator that takes three values. Use slide 4 to explain the students this classification.

Slides 5 to 8

Let us see the arithmetic operators and their uses.

Arithmetic Operators

1-4

- ◆ Are binary operators that work only on numeric operands
- ◆ If operands are non-numeric values such as strings, Booleans, nulls, or resources, they are converted to their numeric equivalents

Table 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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 5 of 44

Arithmetic Operators

2-4

- ◆ Displaying a mathematical expression
 - ❖ Enter the code as shown in a PHP script named `math-exp.php`

Snippet

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

Displays the following output:

In the code, if you follow the Brackets Order Division Multiply Add Subtract (BODMAS) rule of mathematics, the result will be 5.
PHP follows the BODMAS rule for calculation.

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 6 of 44

Arithmetic Operators

3-4

- ◆ Displaying the use of arithmetic operators
 - ❖ Enter the code in a PHP script named **sumarithmetic.php**

Snippet

```
<?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;
```

Version 1.0 © Aptech Limited.

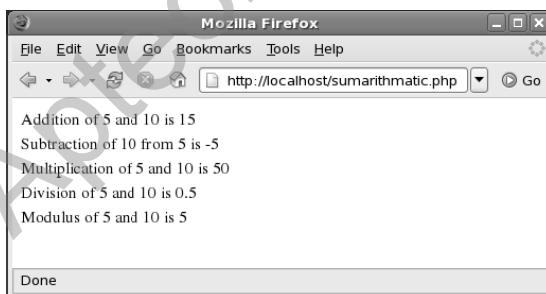
PHP Operators / Session 5 / Slide 7 of 44

Arithmetic Operators

4-4

```
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>";
?>
```

Displays the following output:



Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 8 of 44

Slides 5 to 8 describe the use of arithmetic operators. The arithmetic operators include the basic arithmetic operators such as +, -, *, /, and %. These are used to perform the corresponding operations such as addition, subtraction, multiplication, division, and modulus. Besides these, there exist negation and exponentiation operators that in turn give the resultant accordingly. The negation is the reverse of the given operand and is denoted by – symbol preceded to the variable. The exponentiation is given by ** symbol in between the operands.

In-Class Question:

After you finish explaining about arithmetic operators, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What are the classifications of operators based on the number of operands?

Answer:

Operators are classified as follows based on number of operands:

- **Unary Operator** - acts on only one operand in an expression
- **Binary Operator** - has two operands and performs different arithmetic and logical operations
- **Conditional or Ternary Operator** – has three operands and evaluates the second or third expression depending on the result of the first expression.

Slides 9 to 12

Let us understand the relational operators in PHP.

Relational Operators
1-4

◆ Compares two operands and returns either a true or a false value

◆ The operands can either be numbers or string values

Table lists relational operators along with its description

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
>	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

Version 1.0 © Aptech Limited.
PHP Operators / Session 5 / Slide 9 of 44

Relational Operators

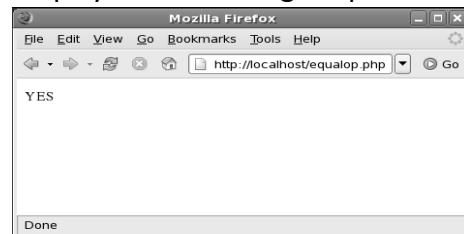
2-4

- Showing the difference between '==' 'equal' and '===' 'identical' relational operators in PHP

Snippet

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

Displays the following output:



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

Relational Operators

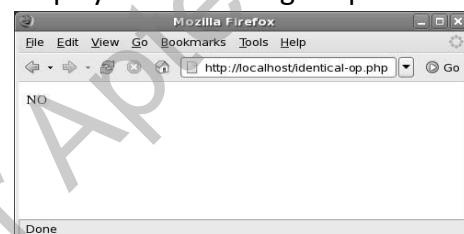
3-4

- Showing the difference between '==' 'equal' and '===' 'identical' relational operators in PHP

Snippet

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

Displays the following output:



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.

Relational Operators

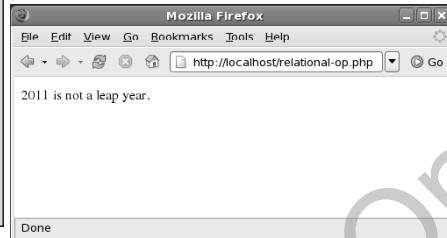
4-4

- ◆ Checking whether the given year is a leap year or not:
 - ❖ Enter the code as shown in a script named **relational-op.php**

Snippet

```
<?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 />";
}
?>
```

Displays the following output:



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.

Version 1.0 © Aptech Limited.
PHP Operators / Session 5 / Slide 12 of 44

Slide 9 describes the relational operators in PHP. The relational operator is also called as comparison operator as it used to compare two values. When comparison is made between a number and a string or the comparison involves numerical strings, then each string is converted to a number and the comparison is performed numerically. Also, tell them that the type conversion does not take place when the comparison is **==** or **!=** as it involves comparing the type as well as the value.

Using slides 10 and 11, explain the example with the given lines of code. Show them the output window given in slide 12, to explain the resultant that is obtained.

Slides 13 to 17

Let us understand about logical operators.

Logical Operators

1-5

- ◆ Enable to combine two or more expressions in a condition
- ◆ Evaluates the expression and returns a Boolean value of either true or false

Table lists various logical operators

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

Version 1.0 © Aptech Limited. PHP Operators / Session 5 / Slide 13 of 44

Logical Operators

2-5

- ◆ Using 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 a PHP script named **logical_and.php**

Snippet

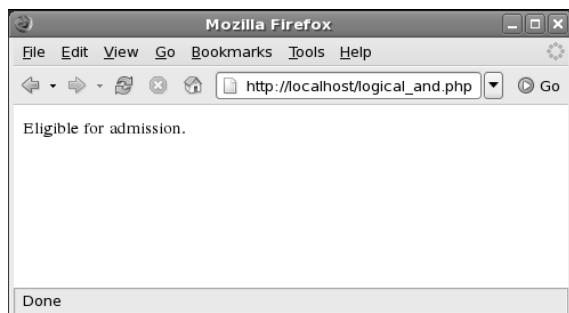
```
<?php
$Percentage = 70;
$Year = "2003";
if ($Percentage>60 AND $Year=="2003")
{
    echo "Eligible for admission.";
}
?>
```

Version 1.0 © Aptech Limited. PHP Operators / Session 5 / Slide 14 of 44

Logical Operators

3-5

Displays the following output:



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.

Logical Operators

4-5

- ◆ Displaying the use of the OR operator
 - ❖ Enter the code as shown in a PHP script named, **logical_or.php**

Snippet

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

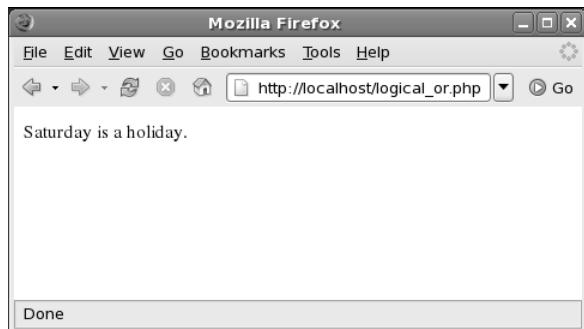
Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 16 of 44

Logical Operators

5-5

Displays the following output:



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.

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 17 of 44

The standard logical operators and, or, not, and xor are supported by PHP. Logical operators first convert their operands to boolean values and then perform the respective comparison. Slides 13 and 14 describe about the logical operators in detail. Also, explain them the example code snippet and the corresponding output using slides 15 to 17.

Slides 18 to 22

Let us understand about bitwise operators.

Bitwise Operators

1-5

- ◆ Operate on the bitwise representation of their operands
- ◆ 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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 18 of 44

Bitwise Operators**2-5**

Table lists bit-wise operators

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
~	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")

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 19 of 44

Bitwise Operators**3-5**

Table lists binary comparisons of bit-wise operators

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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 20 of 44

Bitwise Operators

4-5

- ◆ Displaying the use of bitwise operations
 - ◆ Enter the code in a PHP script named **bitwise.php**

Snippet

```
<?php
$x= 50;
$y= 5;
echo "\$x & \$y = ".($x & $y)."  
";
echo "\$x | \$y = ".($x | $y)."  
";
echo "\$x ^ \$y = ".($x ^ $y)."  
";
echo "~(\$y) = ~$y."  
;
//x is divided by 2 y times
echo "\$x >> \$y = ".($x >> $y)."  
;
//x is multiplied by 2 y times
echo "\$x << \$y = ".($x << $y)."  
;
//Converts the operands to their ASCII values first
('5'(ascii 53))^( '9' (ascii 57))
echo "The Bitwise result of 5 ^ 9 is:".(5 ^ 9)."  
;
//Converts "8" to perform the operation (5 ^ ((int)"8"))
echo "The result of 5 ^ 8 is: ".(5 ^ 8). " "  
;
?>
```

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 21 of 44

Bitwise Operators

5-5

Displays the following output:

The screenshot shows the Mozilla Firefox browser window with the URL `http://localhost/bitwise.php` in the address bar. The page content displays the following output:

```
$x & $y = 0
$x | $y = 55
$x ^ $y = 55
~($y) = -6
$x >> $y = 1
$x << $y = 1600
The Bitwise result of 5 ^ 9 is:12
The result of 5 ^ 8 is: 13
```

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 22 of 44

Slides 18 and 19 describe about the bitwise operator in PHP. It allow evaluation and manipulation of specific bits within an integer. Bit shifting in PHP is arithmetic. Bits shifted off either end are discarded. Left shifts have zeros shifted in on the right, while the sign bit is shifted out on the left, meaning the sign of an operand is not preserved. Right shifts have copies of the sign bit shifted in on the left, which means that the sign of an operand is preserved. Use parentheses to ensure the desired precedence.

Use slide 20 to show the tabulated values for binary conversion of bitwise operator. Using slides 21 and 22, explain the students the given code snippet and the displayed output.

Additional Information:

For more information on bitwise operators in PHP, visit the following links:

<http://php.net/manual/en/language.operators.bitwise.php>

<http://blog.code-head.com/how-to-write-a-permission-system-using-bits-and-bitwise-operations-in-php>

Slides 23 to 26

Let us understand about assignment operators.

The slide has a blue header bar with the title 'Assignment Operators' on the left and a green circular icon with a white question mark on the right. To the right of the icon is the number '1-4'. Below the header is a white content area with a dark blue sidebar on the left. The sidebar contains two bullet points: 'Enables to assign a value to a variable' and 'The '=' sign is the assignment operator'. Below this is a dark blue box labeled 'Syntax' containing the text 'expression 1 = expression 2;'. To the right of the sidebar is a large list of bullet points. These include: 'Operand on the left side of the assignment operator '=' should be a variable', 'The assignment operation can be performed by: Value - copies the value of expression 2 and assigns it to expression 1, Reference - assigns a reference of expression 2 to expression 1', 'The basic assignment operator can also be used as a shorthand operators (combined operator) in conjunction with arithmetic and string operators', and 'The shorthand operators are +=, -=, *=, /=, %=, and .='. At the bottom of the slide, there is a thin blue footer bar with the text 'Version 1.0 © Aptech Limited.' on the left and 'PHP Operators / Session 5 / Slide 23 of 44' on the right.

Assignment Operators

2-4

Table displays various examples with shorthand operators

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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 24 of 44

Assignment Operators

3-4

- ◆ Using shorthand operators
 - ❖ Enter the code in a PHP script named `assign_op.php`

Snippet

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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 25 of 44

Assignment Operators

4-4

Displays the following output:

```
The value of 36 += 6 is 36
The value of 30 -= 6 is 30
The value of 180 *= 6 is 180
The value of 30 /= 6 is 30
The value of 0 %= 6 is 0
The value of 0 days in the month of June is: 0 days in the month of June
```

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 26 of 44

Use slides 23 and 24 to explain the students about assignment operators in brief. The basic assignment operator is '='. Using this operator, the left operand is given the value of the expression on the right. The value of an assignment expression is the value that is assigned. For arrays, assigning a value to a named key is performed using the '>=' operator. The precedence of this operator is the same as other assignment operators.

In addition to the basic assignment operator, there are 'combined operators' for all of the binary arithmetic, array, union, and string operators that allow you to use a value in an expression and then set its value to the result of that expression.

Using slides 25 and 26, explain the students the examples given and along with the examples, show the output window that is displayed.

Slides 27 to 31

Let us understand about increment and decrement operators.

Increment and Decrement Operators

1-5

- ◆ Operate only on variables
- ◆ Cause a variable to change its value
- ◆ The increment operators increase the value of the operand by one
- ◆ The decrement operators decrease the value of the operand by one

Table lists increment and decrement operators

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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 27 of 44

Increment and Decrement Operators

2-5

- ◆ Displaying the use of the increment and decrement operators on a variable \$A
 - ◆ Enter the code as shown in a PHP script named **inc_dec.php**

Snippet

```
<?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/>";
?>
```

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 28 of 44

Increment and Decrement Operators

3-5

Displays the following output:

```
Pre increment value of A is 16
Post increment value of A is 17
Value of A post increment is 17
Pre decrement value of A is 16
Post decrement value of A is 16
Value of A post decrement is 15
```

Done

Increment and Decrement Operators

4-5

- Displaying the use of increment operator on a character variable
 - Enter the code as shown in a PHP script named **inc_op.php**

Snippet

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

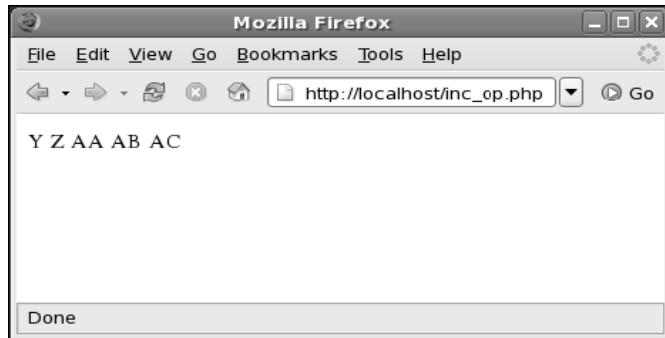
Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 30 of 44

Increment and Decrement Operators

5-5

- ◆ Displays the following output:



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'.

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 31 of 44

Using slides 27 to 29, explain the students about the increment and decrement operators in PHP. The increment and decrement operator is again classified as pre and post increment and decrement operators.

Generally, the increment and decrement operators only affect numbers and strings. Also, note that the character variables can be incremented, but not decremented and even then, only plain ASCII alpha nets and digits are supported.

Tell them that incrementing or decrementing other character variables has no effect, the original string is unchanged. The example code with the output is given in slides 30 and 31. Explain the code and the output.

Additional Information:

For more information and examples on increment/decrement operators in PHP, visit the following link:

<http://www.tizag.com/phpT/operators.php>

Slides 32 to 36

Let us see the string operators.

String Operators

1-5

- ◆ Operate only on character data
- ◆ Non-string operand is first converted before the operation is executed

Table lists string operators

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 side

Version 1.0 © Aptech Limited. PHP Operators / Session 5 / Slide 32 of 44

String Operators

2-5

- ◆ Displaying the concatenation of the strings WELCOME and FRIENDS using the Concatenation operator
 - ❖ Enter the code as shown in a PHP script named **concat.php**

Snippet

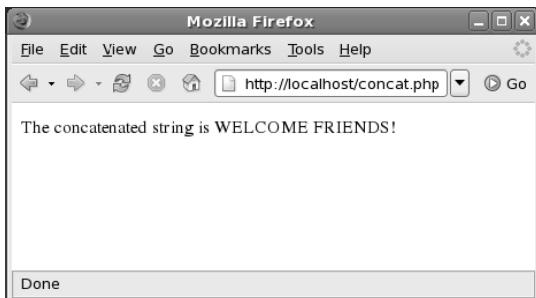
```
<?php
$A="WELCOME ";
$B="FRIENDS!";
$C=$A.$B;
echo "The concatenated string is $C";
?>
```

Version 1.0 © Aptech Limited. PHP Operators / Session 5 / Slide 33 of 44

String Operators

3-5

- ◆ Displays the following output:



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.

String Operators

4-5

- ◆ Assigning a value to a variable using the concatenating assignment operator
 - ◆ Enter the code as shown in a PHP script named **assign_concat.php**

Snippet

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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 35 of 44

String Operators

5-5

- ◆ Displays the following output:



In the code, the argument FRIENDS! is appended on the right side of \$A containing the argument WELCOME.

The string returned is WELCOME FRIENDS!.

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 36 of 44

There are two types of string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side. Use slide 32 to explain the students about string operator in brief.

Using slides 33 and 34, explain the code that describes about the concatenation operator. Using slides 35 and 36, explain the code snippet that shows the use of concatenating assignment operator.

Slides 37 to 39

Let us learn about conditional or ternary operators.

Conditional or Ternary Operators

1-3

- ◆ An alternative to the if-else statement
- ◆ Evaluates an expression for a true or false value and then executes one of the two statements depending upon the result of evaluation

Syntax

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

Where,

- ◆ The operator evaluates the \$var1 and if it is true, expr 1 is evaluated and if it is false, expr 2 is evaluated

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 37 of 44

Conditional or Ternary Operators

2-3

- ◆ Displaying the use of a conditional operator
 - ❖ Enter the code as shown in a PHP script named **condition.php**

Snippet

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

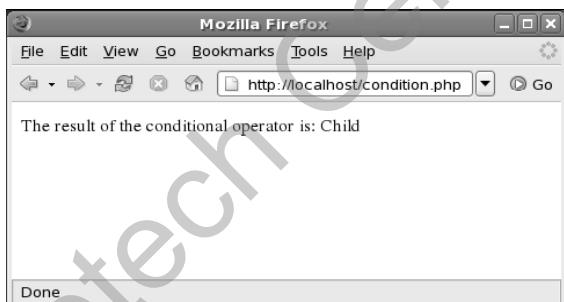
Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 38 of 44

Conditional or Ternary Operators

3-3

- ◆ Displays the following output:



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`.

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 39 of 44

Use slides 37 to 39 to explain about the conditional and ternary operators. It consists of three parts. The ternary operators uses three expressions separated by a question mark and colon. The question mark follows the test expression and the colon and then separates the two possible values, the first of which will be chosen if the test expression is true, and the second if the test expression is false.

In-Class Question:

After you finish with the scope of variable, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Describe about bitwise operators.

Answer:

- Bitwise operator operates on the bitwise representation of their operands.
- It works 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.

Slides 40 and 41

Let us understand the precedence of operators.

Precedence of Operators

1-2

- ◆ The order of precedence to evaluate the expression in PHP is:
 - ❖ Operators with higher precedence are evaluated first in an expression
 - ❖ Operator precedence can be overridden by using parenthesis. The expression within the parenthesis is evaluated first
 - ❖ If two operators of the same precedence are encountered, the expression will be evaluated from left to right

Version 1.0 © Aptech Limited. PHP Operators / Session 5 / Slide 40 of 44

Precedence of Operators

2-2

Table lists the precedence of operators in PHP, with the highest precedence operators listed at the top

Precedence	Category	Operators	Associativity
Highest Precedence	Unary	!, ++, --	Right to left
	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
Lowest Precedence	Logical XOR	XOR	Left to right
	Logical OR	OR	Left to right
Comma	,		Left to right

Use slides 40 and 41 to explain the operator precedence. The operator precedence specifies how the two expressions to be bound together. Explain them the precedence of each operators so far discussed, with the appropriate examples. When the operators have equal precedence, their associativity decides how the operators are grouped. Use of parentheses, even when not strictly necessary, can often increase readability of the code by making grouping explicit rather than relying on the implicit operator precedence and associativity.

Additional Information:

For more information on operator precedence, visit the following links:

<http://php.net/manual/en/language.operators.precedence.php>

<http://www.homeandlearn.co.uk/php/php3p12.html>

Slides 42 to 44

Let us summarize the session.

Summary

1-3

- ◆ Operator is any symbol that performs an operation on an operand
- ◆ An operator enables you 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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 42 of 44

Summary

2-3

- ◆ 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

Version 1.0 © Aptech Limited.

PHP Operators / Session 5 / Slide 43 of 44

Summary

3-3

- ◆ 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

Use slides 42 to 44 to list and explain the summary of this session.

The summary points are as follows:

- Operator is any symbol that performs an operation on an operand.
- An operator enables you 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 side to the value on the right 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.

8.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to conditional statements in PHP.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 10 –Conditional Statements in PHP

10.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

10.1.1 Objectives

At the end of this session, the learners 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

10.1.2 Teaching Skills

To teach this session, you should be well-versed with the use of the if statement in PHP. You should be able to explain the use of the switch statement in PHP. Also, tell them about ternary operator. Describe them about the use of ternary operator.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities:

Overview of the Session

Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

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

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 2 of 35

Tell the students that in this session they will learn the use of `if` statement in PHP. They will be able to describe about the use of `switch` statement. They will also learn the use of ternary operator.

10.2 In-Class Explanations

Slide 3

Let us give an introduction to statements in PHP.

Introduction

- ◆ A statement
 - ❖ Is a smallest element of any programming language
 - ❖ Consists of commands given by a programmer to a computer
 - ❖ Can be an individual statement or a group of statements within curly braces
 - ❖ Usually ends with a semicolon
- ◆ PHP script consists of a series of statements which are as follows:
 - ❖ An assignment
 - ❖ A function call
 - ❖ A conditional statement
 - ❖ An empty statement that does nothing

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 3 of 35

Using slide 3, introduce statements that can be written in PHP.

The statement is a smallest element of any programming language. Tell the students that the statement consists of commands and give some examples. Explain them various types of statements that are present in a PHP script.

Additional Information:

There are some fundamental differences between statements and expressions. The following table lists a key difference between the two.

Statements	Expressions
<p>Do not necessarily return values. For example, consider the following statement:</p> <pre>oddNum = 5;</pre> <p>The statement only stores the value 5 in the <code>oddNum</code> variable.</p>	<p>Always evaluates to a value. For example, consider the following expression:</p> $100 * (25 * 10)$ <p>The expression evaluates to the value 2500.</p>

For more information on statements in PHP, visit the following links:

http://www.w3schools.com/php/php_if_else.asp

<http://php.net/manual/en/control-structures.if.php>

<http://www.codingunit.com/php-tutorial-if-else-statements>

Slide 4

Let us learn about conditional control structures.

Conditional Control Structures

- ◆ Control the flow of a program on execution or skip code based on certain criteria
- ◆ Are of two types:
 - ◆ `if` statement
 - ◆ `switch` statement

Version 1.0 © Aptech Limited. Conditional Statements in PHP / Session 7 / Slide 4 of 35

Tell the students about the conditional control structures using slide 4. Explain that conditional statements are the set of commands used to perform different actions based on different conditions.

Slides 5 to 10

Let us learn in detail about the `if` statement.

if Statement

1-6

- ◆ It is a common control structure
- ◆ It contains an expression called as truth expression
- ◆ The truth expression:
 - ❖ Can be a Boolean, variable, constant or an expression
 - ❖ Evaluates to `true`, `false` or `NULL`
 - ❖ If evaluates to `true`, following statements are executed
 - ❖ If evaluates to `false` or `NULL`, statements are not executed

Syntax

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

Where,

`If` keyword is followed by truth expression in parenthesis

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 5 of 35

if Statement

2-6

- ◆ Checking whether a triangle is valid using the `if` statement
 - ❖ Enter the following code in a script named **validity.php**

Snippet

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

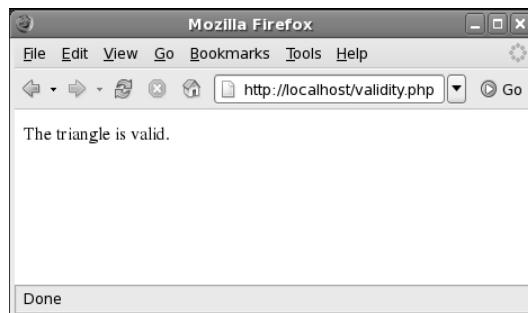
Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 6 of 35

if Statement

3-6

Displays the following output:



If the sum of the degrees \$a, \$b, and \$c are equal to 180 then the statement following the if condition is executed

if Statement

4-6

- ◆ Accepting and displaying the salary of an employee based on the salary and bonus
 - ❖ **salBonus.html** – accepts user inputs

Snippet

```
<html>
<body>
<form action="salBonus.php" method="GET">
<table>
<tr>
<td>Salary &nbsp; </td>
<td><input type="text" name="sal"></td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 8 of 35

if Statement

5-6

- Accepting and displaying the salary of an employee based on the salary and bonus

♦ **salBonus.php** – process the salary and calculate the bonus

Snippet

```
<?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";
}
?>
```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 9 of 35

if Statement

6-6

salBonus.html displays the following output:

salBonus.php displays the following output:

The code accepts and displays the bonus at the rate of 10% and total salary when the salary of the employee is greater than \$850.

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 10 of 35

Slide 5 describes the use of **if** construct. The **if** construct allows the conditional execution of code fragments. PHP features an **if** structure similar to that of C. Generally, expressions are evaluated to a boolean value. If the expression evaluates to true, then statement will be executed, and if it evaluates to false, then the statement will be ignored.

Using slides 6 and 7, explain the students how to use the **if** statement with an example. Three values for the corners of a triangle representing angles are assigned and then, their sum is calculated. Then, the code checks the sum to see whether the triangle is valid or not.

Here, the condition to be satisfied is that the sum of angles must be 180, so that the triangle is considered valid. Mention that multiple `if` statements can be nested infinitely within other `if` statements, which provides you with complete flexibility for conditional execution of various parts of your program.

Then, using slides 8 and 9, explain them the next example saying that it calculates the total salary based upon the given condition. Show them the output of the corresponding code using slide 10.

Additional Information:

Following is the syntax for the `if` statement:

Syntax:

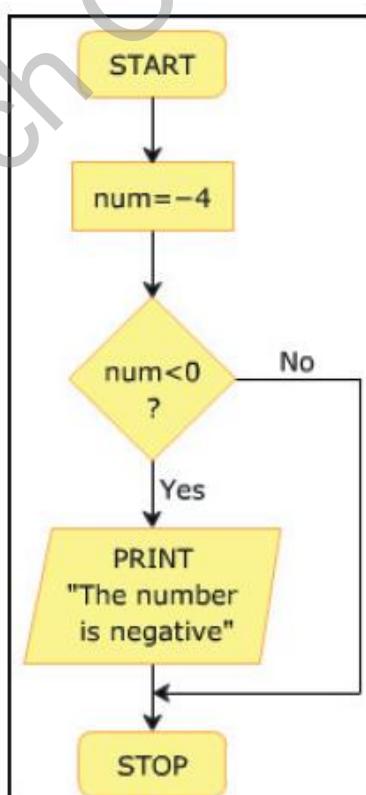
```
if (condition)
{
    // one or more statements;
}
```

where,

condition: Is the boolean expression.

statements: Are set of executable instructions executed when the boolean expression returns true.

You can use the following example to show the working of the `if` statement.



Here, `num` is initialized to value -4. The `if` statement is executed and the value of `num` is checked to see if it is less than 0. The condition evaluates to `true` and therefore, the output 'The number is negative' is displayed.

In-Class Question:

After you finish explaining about the `if` statement, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you define the truth expression of the `if` statement?

Answer:

The truth expression:

- Can be a Boolean, variable, constant, or an expression.
- Evaluates to true, false, or NULL.
- The statements enclosed within the `if` construct will be executed, if it evaluates to true.
- If it evaluates to false or NULL, statements are not executed.

Slides 11 to 18

Let us understand the `if...else` statement.

if...else Statement
1-8

◆ Is used along with `if` statement

◆ Is executed when a specified condition is false

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;
}

```

Version 1.0 © Aptech Limited.
Conditional Statements in PHP / Session 7 / Slide 11 of 35

if...else Statement

2-8

- ◆ Displaying a block of code with an if...else statement

- ◆ Enter the following code:

Snippet

```
<?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";
}
?>
```

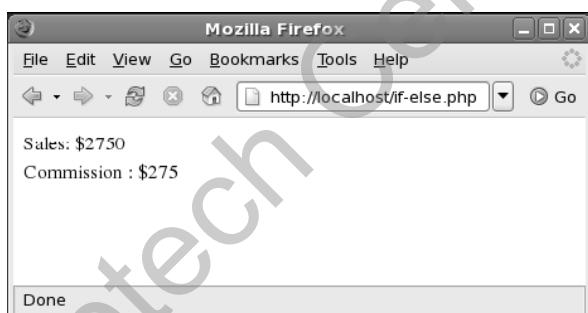
Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 12 of 35

if...else Statement

3-8

Displays the following output:



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.

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 13 of 35

if...else Statement

4-8

- ◆ **else if clause is:**
 - ❖ Used along with **if** statement
 - ❖ An optional clause that allows testing alternative conditions

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 14 of 35

if...else Statement

5-8

- ◆ Demonstrating the use of **else if** clause
 - ❖ **saleComm.html** - accepts the sales amount from the user

Snippet

```
<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>
```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 15 of 35

if...else Statement

6-8

- ❖ **saleComm.php** - processes the sales amount and calculates the commission

Snippet

```
<?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>"; }
?>
```

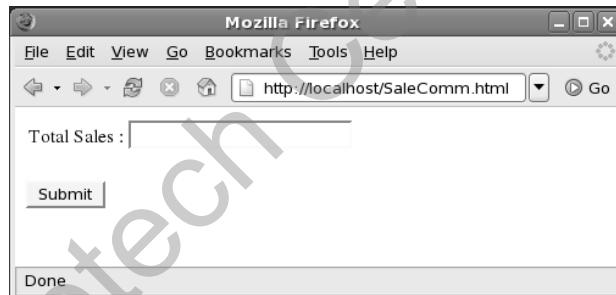
Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 16 of 35

if...else Statement

7-8

saleComm.html displays the following output:

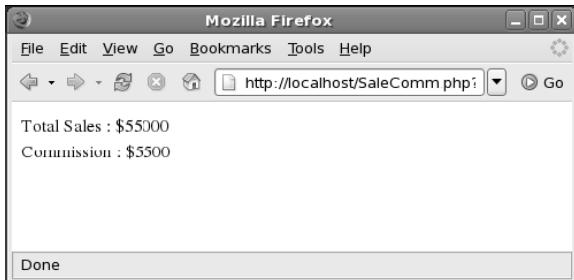


Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 17 of 35

if...else Statement

8-8

saleComm.php displays the following output:

In the code, commission is calculated according to the sales amount the user enters.

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 18 of 35

Slide 11 describes about the `if...else` construct. The `if...else` statement is used to execute a block of code if a condition is true and another block of code if the condition is false. Explain the students the detailed working of `if...else` construct using slides 12 and 13. In the given example, the condition to be satisfied is that the salary should be greater than 2000. This condition is specified using an `if` statement. When the condition becomes true, the statement inside the `if` statement is executed. When it becomes false, the statement inside the `else` is executed.

Using slides 14 to 16, explain them the code snippet that is given to explain the construct in brief. The output for the corresponding code is given in slides 17 and 18.

Additional Information:

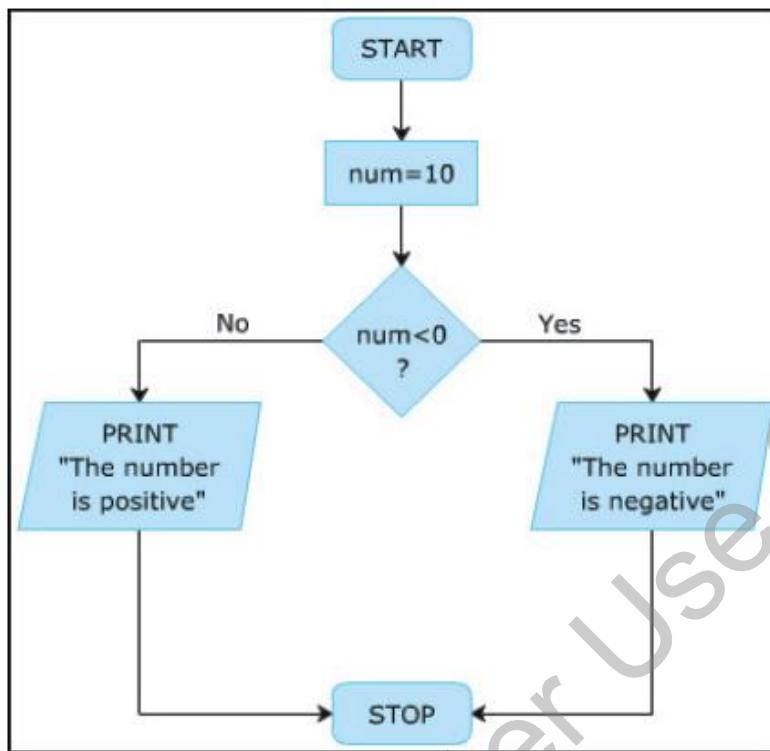
Following is the syntax for the `if...else` statement:

Syntax:

```
if (condition)
{
    // one or more statements;
}

else
{
    //one or more statements;
}
```

Following figure demonstrates the `if-else` construct with an example.



Here, `num` is initialized to value 10. The `if` statement is executed and the value of `num` is checked to see if it is less than 0. The condition evaluates to `false` and the program control is passed to the `else` block and the output 'The number is positive' is displayed.

Slides 19 to 23

Let us understand about the nested `if` statement.

Nested if Statement

1-5

- An `if` statement within an `if` statement or an `else` statement is known as nested `if` statement

Nested if Statement

2-5

- ◆ Calculating the electricity charges based on the units of electricity consumed
 - ❖ **elecBill.html** - accepts the number of units consumed

Snippet

```
<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>
```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 20 of 35

Nested if Statement

3-5

- ◆ Calculating the electricity charges based on the units of electricity consumed
 - ❖ **elecBill.php** - calculates total electricity bill

Snippet

```
<?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";
}
```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 21 of 35

Nested if Statement

4-5

```

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";
    }
}
?>

```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 22 of 35

Nested if Statement

5-5

elecBill.html – displays the following output:

Electricity Units Consumed :

Done

elecBill.php – displays the following output:

Number of Units Consumed : 1750

Service Charge added for Units above 1000 : \$525

Total Electricity Bill : \$5775

Done

In the code, if the user enters 1500 as input, PHP code in **Code** 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 **elecBill.php** 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.

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 23 of 35

Several conditional **if** statements can be nested within each other. Explain about the nested **if...else** statement using slide 19. Explain them the example given in slides 20 to 22. Describe each line of code given and the process carried out. Tell them that the inner statement is first executed and the resultant outcome becomes the input for the immediate inner statement that finally executes the outer statement. Show them the output for the given code using slide 23.

Slides 24 to 31

Let us understand about `switch` statement.

switch Statement

1-8

- ◆ Used as an alternative to a lengthy `if...else` construct
- ◆ 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
- ◆ Uses a `break` statement to halt the execution of the `switch` statement and transfer the control to the code following `switch`

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 24 of 35

switch Statement

2-8

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
}
```

Where,

- ◆ `case` keyword is followed by a case constant
- ◆ `default` is a special case executed when none of the case constants is matching

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 25 of 35

switch Statement

3-8

- ◆ Displaying a switch statement without any break statement

- ❖ Enter the code in a script named **switch.php**

Snippet

```
<?php  
$day = 1;  
switch ($day)  
{  
    case 1:  
        echo "It is Sunday";  
        echo "<br>";  
    case 2:  
        echo "It is Monday";  
        echo "<br>";
```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 26 of 35

switch Statement

4-8

```
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>";  
}  
?>
```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 27 of 35

switch Statement**5-8**

Displays the following output:

```
It is Sunday
It is Monday
It is Tuesday
It is Wednesday
It is Thursday
It is Friday
It is Saturday
There are Seven Days in a Week
```

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.

switch Statement**6-8**

- ◆ Displaying a switch statement without any break statement
 - ❖ Enter the code in a script named **break.php**

Snippet

```
<?php
$day = 1;
switch ($day)
{
    case 1:
        echo "It is Sunday";
        echo "<br>";
        break;
    case 2:
        echo "It is Monday";
        echo "<br>";
        Break;
```

switch Statement

7-8

```

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;

```

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 30 of 35

switch Statement

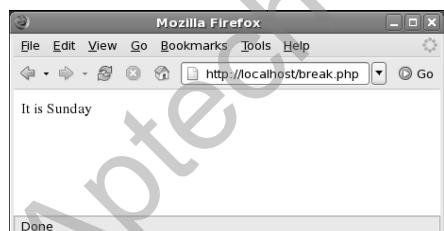
8-8

```

default:
echo "There are Seven Days in a Week";
echo "<br>";
break;
}
?>

```

Displays the following output:



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.'

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 31 of 35

Slide 24 is used to describe about the `switch` statement in detail. The `switch` statement is similar to a series of `if` statements on the same expression. While writing a script, in some situations, the comparison is done with the same variable for many different values and a different set of statements are executed depending upon the value it equals to.

Slide 25 shows the syntax for the `switch` statement while the corresponding example is given in slides 26 and 27. The code in slide 26 describes how to specify `switch` and `case` keywords.

Show them the output produced for the corresponding code that is displayed in slide 28. Also, explain them the example given in slides 29 and 30 along with the output displayed in slide 31. Show them the differences between each example with the given code.

In-Class Question:

After you finish with the explanation of `switch` statement, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you describe the `switch` statement?

Answer:

The `switch` statement is:

- Used as an alternative to a lengthy `if...else` construct.
- 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.
- Uses a `break` statement to halt the execution of the `switch` statement and transfer the control to the code following `switch`.

Slides 32 and 33

Let us understand the ternary operator.

Ternary (?) Operator
1-2

- ◆ It is also known as a conditional operator
- ◆ It simplifies complex conditions into one-line statements
- ◆ It is considered as an alternative for the `if...else` statement

Syntax

```
truth_expr ? expr1 : expr2;
```

Where,

- ◆ `truth_expr` is evaluated and if it is `true`, `expr1` is evaluated
- ◆ If it is `false`, `expr2` is evaluated

Version 1.0 © Aptech Limited.

Conditional Statements in PHP / Session 7 / Slide 32 of 35

Ternary (?) Operator

2-2

- ◆ Displaying the output using the ternary operator
 - ❖ Enter the code as shown in code in the script named **ternary-op.php**

Snippet

```
<?php
$x = 100;
$y = 50;
$disp = ($x > $y) ? "X is greater than Y" : "Y is
greater than X";
echo $disp;
?>
```

Displays the following output:

Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ternary-op.php
X is greater than Y
Done

Version 1.0 © Aptech Limited. Conditional Statements in PHP / Session 7 / Slide 33 of 35

Use slide 32 and 33 to explain about the ternary operator. The ternary operator consists of three operands that are, a condition, a result for true, and a result for false. The ternary operator is the shorthand for the `if` statement. The ternary operator will return either the first value if the condition is true or the second value if the condition is false. This is shown in the example given in slide 33.

Additional Information:

For more information on the `switch` statement, visit the following links:

<http://www.tizag.com/phpT/switch.php>

<http://php.net/manual/en/control-structures.switch.php>

Slides 34 and 35

Let us summarize the session.

Summary 1-2

- ◆ 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

Version 1.0 © Aptech Limited. Conditional Statements in PHP / Session 7 / Slide 34 of 35

Summary 2-2

- ◆ 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 conditional operator. It simplifies complex conditions into one-line statements

Version 1.0 © Aptech Limited. Conditional Statements in PHP / Session 7 / Slide 35 of 35

Use slides 34 and 35 to list and explain the summary of this session.

The summary points are as follows:

- 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 match the value of the switch variable.
- Ternary operator is also known as conditional operator. It simplifies complex conditions into one-line statements.

10.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to flow control in PHP.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 12 –Flow Control in PHP

12.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

12.1.1 Objectives

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

- Explain the use of loops
- Explain the use of jump statements

12.1.2 Teaching Skills

To teach this session, you should be well-versed with the use of loops in PHP. You should be able to explain the use of jump statements in PHP.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *Explain the use of loops*
- ◆ *Explain the use of jump statements*

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 2 of 32

Tell the students that they will be introduced to loops in PHP and learn the use of loops in PHP. In addition, they will also understand the use of jump statements.

12.2 In-Class Explanations

Slide 3

Let us introduce loops in PHP.

Introduction

- ◆ Loops
 - ❖ Perform repetitive tasks such as:
 - ❖ Retrieving information stored in databases
 - ❖ Sending mails to multiple users
 - ❖ Reading contents of an array
- ◆ Loops provided by PHP are as follows:
 - ❖ While
 - ❖ Do-while
 - ❖ For

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 3 of 32

Using slide 3, briefly introduce loops in PHP.

In programming, it is often necessary to repeat the same block of code a given number of times, or until a certain condition is met. This can be made easier using looping statements. Loop statements will reduce the lines of code as well make it easier to debug and maintain the programs. PHP has three major types of looping statements `while`, `do-while`, and `for`. The `for` statements are best used when you want to perform a loop a specific number of times. The `while` statements are best used to perform a loop an undetermined number of times.

Additional Information:

For more information on loops in PHP, visit the following links:

<http://webcheatsheet.com/php/loops.php>

<http://www.tizag.com/phpT/forloop.php>

http://www.tutorialspoint.com/php/php_loop_types.htm

Slide 4

Let us learn about working with loops.

The screenshot shows a slide titled "Working with Loops". The slide content is as follows:

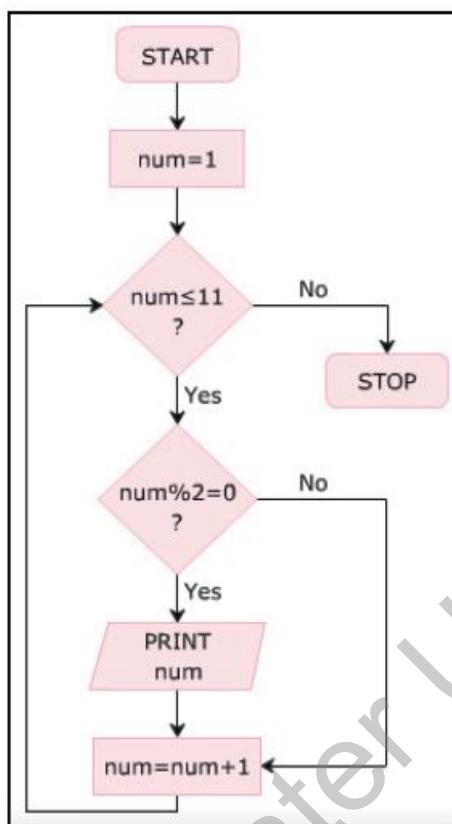
- ◆ Executes a block of code repetitively
- ◆ Tests the specified condition
 - ❖ If `true`, the statements present in the body of the loop are executed repetitively
 - ❖ If `false`, the loop ends, and the control is transferred to the statement following the loop
- ◆ The continuous execution of statements inside the loop is called iteration

At the bottom of the slide, there is a footer bar with the text "Version 1.0 © Aptech Limited." and "Flow Control in PHP / Session 9 / Slide 4 of 32".

Tell the students about the working process of loop in PHP. Using slide 4, explain them how the loop is executed in various situations. Explain them how it tests the specified condition.

Generally, loops in PHP are used to execute the same block of code a specified number of times. Slide 4 illustrates the idea of a loop in PHP.

You can use the following illustration and example to explain to students the working of a loop.



In this example, `num` is declared as an integer variable and initialized to value 1. The condition in the loop is checked, which specifies that the value of `num` variable should be less than or equal to 11. If this condition is true, the value of the `num` variable is divided by 2 and the remainder is checked to see if it is 0. If the remainder is 0, the value of the variable `num` is displayed in the console window and the variable `num` is incremented by 1. Then, the program control is passed to the `while` statement to check the condition again. When the value of `num` becomes 12, the `while` loop terminates as the loop condition becomes false.

Slides 5 to 9

Let us learn about while loop.

while Loop

1-5

- ◆ Validity of the condition is checked before the loop is executed:
 - ◆ If condition is true, statements are executed in the loop body
 - ◆ If condition is false, the body of the loop is not executed

Syntax

```

while(condition)
{
  These statements are executed only if the condition is true;
}
These statements are executed irrespective of the condition;

```

Where,

- ◆ The condition is the test expression consisting of variables and operators

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 5 of 32

while Loop

2-5

- ◆ Displaying the first five multiples of 5

Snippet

```

<?php
$counter=1;
$number=5;
while($counter <= 5)
{
$result=$number*$counter;
echo "<br>$result";
$counter=$counter+1;
}
?>

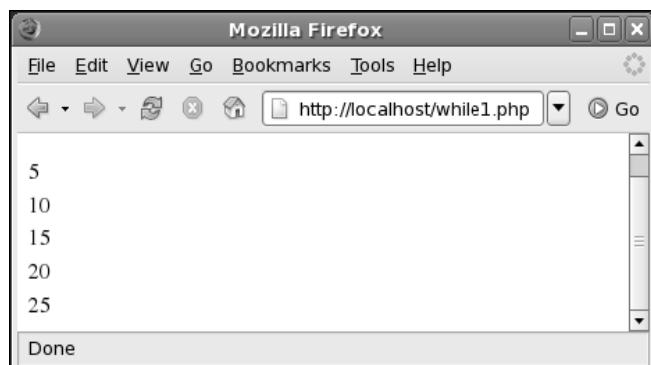
```

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 6 of 32

while Loop

3-5

Displays the following output:



A screenshot of a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The address bar shows "http://localhost/while1.php". The main content area displays the following text:
5
10
15
20
25
A "Done" button is at the bottom right of the content area.

In the code, the result is displayed until the counter reaches 5.

The loop stops once the counter exceeds 5.

while Loop

4-5

- ◆ Displaying the odd numbers between 1 to 10

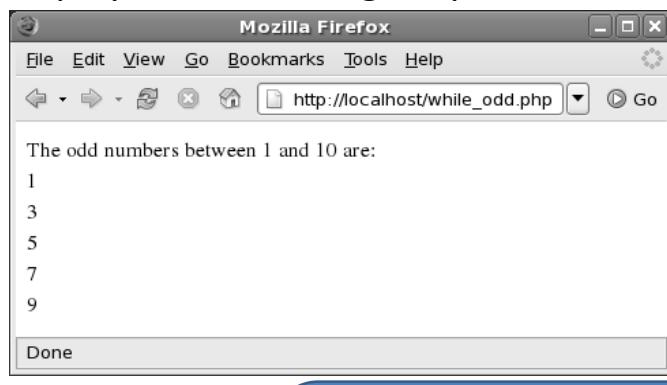
Snippet

```
<?php
$number=1;
echo "The odd numbers between 1 and 10 are:";
while($number <= 10)
{
echo "<br>$number";
$number=$number+2;
}
?>
```

while Loop

5-5

Displays the following output:



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.

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 9 of 32

Slide 5 describes the use of `while` loops in PHP. The simplest type of loop in PHP is the `while` loop. This loop behaves similar to its C counterpart. Using a `while` loop, the statements in the body of the loop are executed repeatedly as long as the `while` expression evaluates to true. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statements in the body of the loop block, execution will not stop until the end of the iteration. Sometimes, if the `while` expression evaluates to false at the beginning, the nested statement will not be run even once.

Using slides 6 to 9, explain the `while` statement with the examples given along with the output window.

In-Class Question:

After you finish explaining about the `while` loop, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What is meant by loops?

Answer:

Loops perform repetitive tasks such as:

- Retrieving information stored in databases.
- Sending mails to multiple users.
- Reading contents of an array.

Slides 10 to 12

Let us understand the do-while loop.

do-while Loop

1-3

- ◆ Condition is checked at the end of the loop
- ◆ Executes the loop body at least once
- ◆ Works similar to the while loop

Syntax

```
do{
    <These statements are executed if the condition is true;>
}while(condition)
<These statements are executed irrespective of the condition;>
```

Where,

- ◆ The loop body is followed by the while keyword and the condition in parenthesis

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 10 of 32

do-while Loop

2-3

- ◆ Displaying the odd numbers between 1 to 10 using do-while loop

Snippet

```
<?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.";
?>
```

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 11 of 32

do-while Loop

3-3

Displays the following output:

The screenshot shows a Mozilla Firefox browser window displaying the output of a PHP script. The title bar says "Mozilla Firefox". The address bar shows "http://localhost/do-while_odd.php". The main content area displays the following text:
The odd numbers between 1 and 10 are:
1
3
5
7
9
The loop ends because the condition is satisfied.
Done

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.

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 12 of 32

Slide 10 describes about the `do-while` loop. `Do-while` loops are very similar to `while` loops, except that the conditional expression is checked at the end of each iteration instead of in the beginning. The main difference from the `while` loop is that the first iteration of a `do-while` loop is guaranteed to run, regardless of whether the condition is true or false, whereas it may not necessarily run with a regular `while` loop. If the conditional expression in a `while` loop evaluates to false right at the beginning, the loop execution would end immediately.

Using slides 11 and 12, explain them how the `do-while` loop is executed based upon the condition. Use the examples and show them the corresponding output. While explaining the code, show them the process that is performed by each line of code to generate the desired value at the end.

Slides 13 to 18

Let us understand `for` loop.

for Loop 1-6

- ◆ Executes block of code repetitively for a fixed number of times
- ◆ Statements in the loop body are executed as long as the condition is satisfied
- ◆ Stops the execution only when the condition is not satisfied

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 13 of 32

for Loop 2-6

Syntax

```

for (expr1; expr2; expr3)
{
    These statements are executed if the condition is true;
}
These statements are executed irrespective of the condition;

```

Where,

- ◆ **expr1** – is an initialization expression that initializes the value of the counter
- ◆ **expr2** – is a test expression that is evaluated for each loop iteration
- ◆ **expr3** – is a re-initialization expression that increases or decreases the value in the counter variable

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 14 of 32

for Loop

3-6

- Displaying the double of the given number using for loop

Snippet

```
<?php
$number=6;
for($counter=1; $counter <= 3; $counter++)
{
    echo "$number<br>";
    $number=$number*2;
}
echo "The loop ends because the condition is
satisfied.";
?>
```

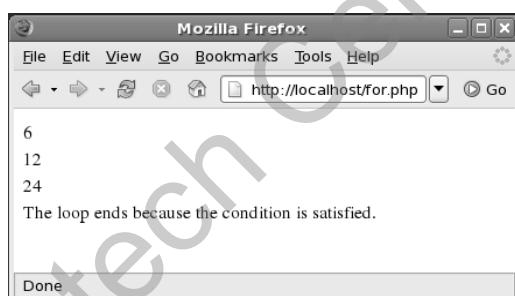
Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 15 of 32

for Loop

4-6

Displays the following output:



The variable, `$number` is initialized with a value of 6.

When the loop starts, 6 is multiplied by 2 And the value is 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.

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 16 of 32

for Loop

5-6

- ◆ Displaying the first five odd numbers in the reverse order using `for` loop

Snippet

```
<?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.";
?>
```

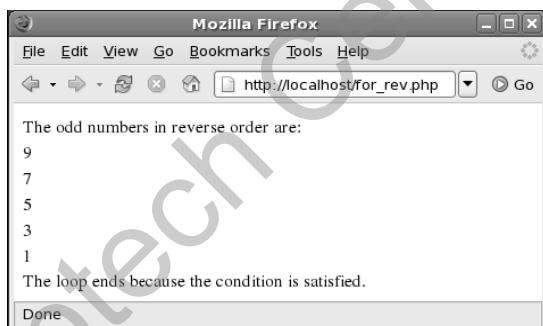
Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 17 of 32

for Loop

6-6

Displays the following output:



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.

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 18 of 32

Using slides 13 and 14, explain the students about the `for` loop and its uses. The `for` loops are the most complex loops in PHP. There are three expressions in the `for` loop structure. The first expression is evaluated once unconditionally at the beginning of the loop. The second expression is evaluated at the beginning of each iteration. If it evaluates to true, the loop continues and the nested statements are executed. If the expression evaluates to false, the execution of the loop ends. At the end of each iteration, the next expression is evaluated. Each expression can be empty or may contain multiple expressions that are separated by commas.

Use slides 15 and 16 and explain them the simple `for` loop and its execution. Using slides 17 and 18, explain the use of `for` loop in displaying the first five odd numbers in reverse order. Show them the output and explain how the values are obtained as resultant values.

Slide 19

Let us understand jump statements.



Jump Statements

- ◆ Control the execution of the loop and conditional statements
- ◆ PHP provides the following jump statements:
 - ◆ break
 - ◆ continue
 - ◆ exit

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 19 of 32

Slide 19 is used to describe the jump statements. Tell them that a jump statement controls the execution of the loop and conditional statements. Explain them each type of jump statements that are listed on slide 19: `break`, `continue`, and `exit`.

In-Class Question:

After you finish with the explanation of the `for` loop and jump statements, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you describe the `for` loop in PHP?

Answer:

The `for` loop:

- Executes block of code repetitively for a fixed number of times.
- Statements in the loop body are executed as long as the condition is satisfied.
- Stops the execution only when the condition is not satisfied.

Slides 20 to 25

Let us understand the `break` statement.

break Statement  1-6

- ◆ 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
- ◆ Can be used with the `if` statement, `switch` statement, `for` loop, `while` loop, and `do-while` loop

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 20 of 32

break Statement  2-6

- ◆ Displaying consecutive numbers from 1 to 5 using `break` statement

Snippet

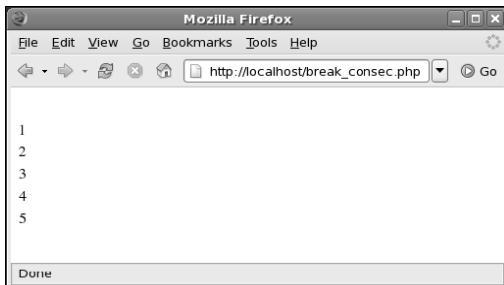
```
<?php
for($i=1;;$i++) {
if($i>5)
{
    break;
}
echo "<br>$i";
?>
```

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 21 of 32

break Statement

3-6

Displays the following output:



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.

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 22 of 32

break Statement

4-6

- ◆ Checking whether the alphabet is a vowel using **switch** statement

Snippet

```
<?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;
case 'i':
```

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 23 of 32

break Statement

5-6

```

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.";
}
?>
```

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 24 of 32

break Statement

6-6

Displays the following output:



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.

Version 1.0 © Aptech Limited.

Flow Control in PHP / Session 9 / Slide 25 of 32

Using slide 20, explain about the `break` statement. The `break` statement is used to end the execution of the current `for`, `while`, `do-while`, or `switch` structure. Using slides 21 and 22, explain the given example that displays consecutive numbers between 1 to 5. Use slides 23 to 25 and explain the use of `break` statement in each line of code where it is present. Use the points given in the slides along with the output to explain the students about `break` statement in brief.

Slides 26 to 28

Let us understand the `continue` statement.

continue Statement

1-3

- ◆ Used within the loop statements
- ◆ Skips the code following the `continue` statement in the loop body and executes the next iteration of the loop
- ◆ Can be used with the `if` statement, `for` loop, `while` loop, and `do-while` loop

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 26 of 32

continue Statement

2-3

- ◆ Displaying the consecutive numbers from 1 to 5 using the `while` loop

Snippet

```
<?php
$counter = 0;
while($counter<5)
{
    $counter++;
    if($counter==3)
    {
        echo "Continues the loop<br>";
        continue;
    }
    echo "$counter<br>";
}
echo "The loop ends here";
?>
```

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 27 of 32

continue Statement 3-3

Displays the following output:

The screenshot shows a Mozilla Firefox browser window displaying the output of a PHP script. The output consists of the numbers 1, 2, 4, and 5, followed by the text "Continues the loop" and "The loop ends here". A "Done" button is visible at the bottom of the browser window.

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`.

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 28 of 32

Explain the students about `continue` statement using slide 26. The `continue` statement is used within looping structures to skip the rest of the current loop iteration and resume execution at the condition evaluation and then, begin the next iteration. It accepts an optional numeric argument that tells it how many levels of enclosing loops it should skip. The default value is 1, thus, skipping to the end of the current loop.

With the example given in slides 27 and 28, explain the practical usage of `continue` statement in brief.

Slides 29 and 30

Let us understand the `exit` statement.

exit Statement 1-2

- ◆ Ends the loop and the control is transferred to the statement following the loop body
- ◆ Following code calculates the HRA using `exit` statement:

Snippet

```
<?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";
}
?>
```

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 29 of 32

exit Statement 2-2

Displays the following output:



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.

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 30 of 32

Use slides 29 and 30 to explain about the `exit` statement. The `exit` statement terminates the execution of the script. Explain the example given along with the output.

Additional Information:

For more information on the `continue` and `exit` statements, visit the following links:

<http://www.w3resource.com/php/statement/continue.php>

<http://php.net/manual/en/function.exit.php>

Slides 31 and 32

Let us summarize the session.

Summary 1-2

- ◆ 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

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 31 of 32

Summary 2-2

- ◆ 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

Version 1.0 © Aptech Limited. Flow Control in PHP / Session 9 / Slide 32 of 32

Use slides 31 and 32 to list and explain the summary of this session.

The summary points are as follows:

- 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 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.

12.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to functions in PHP.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 14 –Functions in PHP

14.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

14.1.1 Objectives

At the end of this session, the learners 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

14.1.2 Teaching Skills

To teach this session, you should be well-versed with the functions in PHP. You should be able to explain the built-in functions in PHP. In addition, you should capable of describing the process of creating a user-defined function, passing arguments to a function, and returning values from a function. You should be able to explain about the use of recursive functions.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 2 of 55

Tell the students that they will be introduced to functions in PHP and learn about built-in functions. They will also be able to create a user-defined function. In addition, they will also understand the process of passing arguments to a function and returning values from a function.

14.2 In-Class Explanations

Slide 3

Let us introduce functions in PHP.

Introduction

- ◆ Functions
 - ❖ Are named section of a program
 - ❖ Are used to perform a specific task
 - ❖ Split program into modules
 - ❖ Are used to enable the developer to reuse the same piece of code
 - ❖ Can be easily modified in a program instead of going through entire code to make changes

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 3 of 55

Using slide 3, introduce briefly the concept of functions in PHP.

Tell them that functions in PHP are similar to other programming languages. A function is defined as a piece of code that may take input in the form of one or more parameters, does some processing, and returns a value. A key benefit of using functions is that they are reusable. If a task is need to be performed a number of times, a function is an ideal solution. Functions also make it easier to debug and maintain a program.

Additional Information:

For more information on functions in PHP, visit the following links:

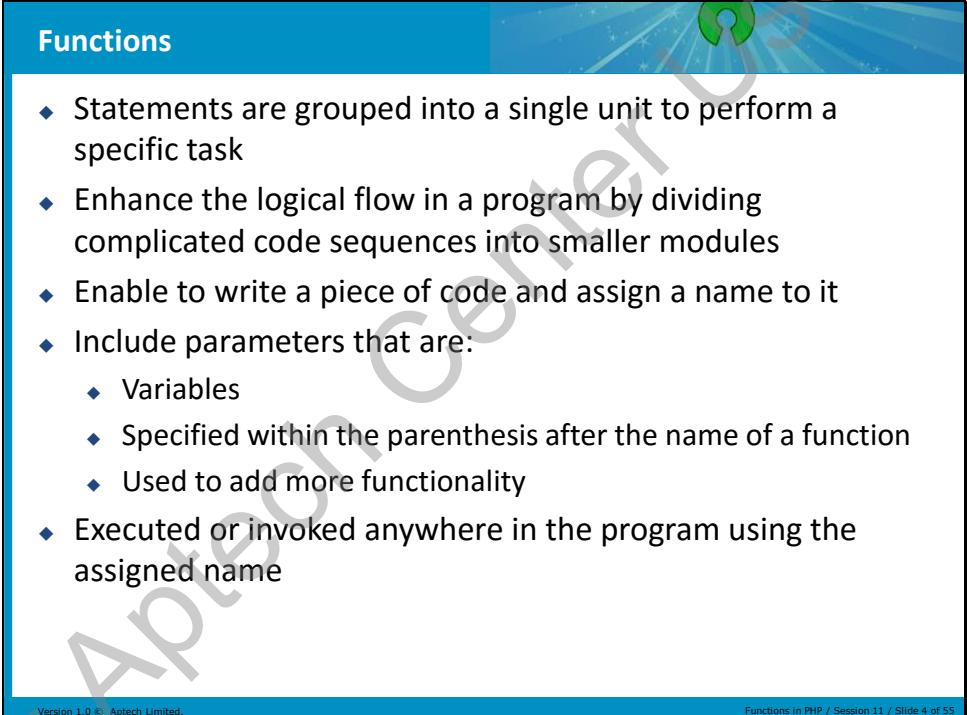
<http://www.sitepoint.com/defining-and-using-functions-in-php/>

<http://www.tizag.com/phpT/phpfunctions.php>

http://www.tutorialspoint.com/php/php_functions.htm

Slide 4

Let us learn about functions.



The screenshot shows a slide with a blue header bar containing the title "Functions". The main content area contains a bulleted list of six points describing the benefits of functions:

- ◆ Statements are grouped into a single unit to perform a specific task
- ◆ Enhance the logical flow in a program by dividing complicated code sequences into smaller modules
- ◆ Enable to write a piece of code and assign a name to it
- ◆ Include parameters that are:
 - ◆ Variables
 - ◆ Specified within the parenthesis after the name of a function
 - ◆ Used to add more functionality
- ◆ Executed or invoked anywhere in the program using the assigned name

At the bottom of the slide, there is a footer bar with the text "Version 1.0 © Aptech Limited." on the left and "Functions in PHP / Session 11 / Slide 4 of 55" on the right.

Tell the students about the definition of functions in PHP. Tell the students to note that the function is a block of code that can be called from any point in a script after it has been declared. It is a compartmentalized PHP script designed to accomplish a single task. Furthermore, code contained within a function is ignored until the function is called from another part in the script.

Using slide 4, explain the students briefly about the functions in PHP.

Slide 5

Let us learn about built-in PHP functions.


Built-in PHP Functions

- ◆ Provides different built-in functions to be included in the PHP script
- ◆ Built-in functions are grouped into following categories:
 - ❖ Mathematical functions
 - ❖ String functions
 - ❖ Date and time functions
 - ❖ Error handling functions
 - ❖ Database functions
 - ❖ Array functions
 - ❖ Mail functions

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 5 of 55

Using slide 5, explain the students about built-in PHP functions. Tell them that PHP provides different readymade functions also called as built-in functions that are categorized into various types. List these types as given on slide 5.

Slides 6 to 8

Let us learn about mathematical functions.


Mathematical Functions
1-3

- ◆ Operate on numerical data

Table 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
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

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 6 of 55

Mathematical Functions

2-3

- ◆ **maths.php** - Use of mathematical functions

Snippet

```
<?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 min(99.9,9.9999,99.99);
echo "<br>";
echo "The square root of 256 is: ";
echo sqrt(256);
echo "<br>";
?>
```

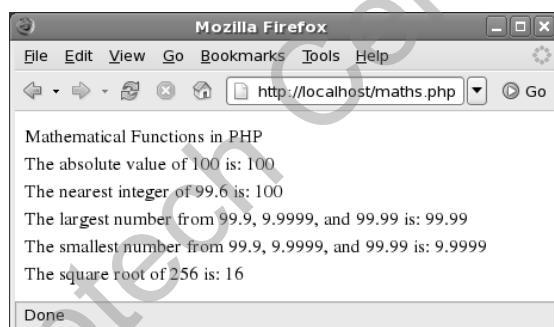
Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 7 of 55

Mathematical Functions

3-3

Displays the following output:



The screenshot shows a Mozilla Firefox browser window displaying the output of a PHP script. The title bar says "Mozilla Firefox". The address bar shows "http://localhost/math.php". The main content area displays the following text:
Mathematical Functions in PHP
The absolute value of 100 is: 100
The nearest integer of 99.6 is: 100
The largest number from 99.9, 9.9999, and 99.99 is: 99.99
The smallest number from 99.9, 9.9999, and 99.99 is: 9.9999
The square root of 256 is: 16
A "Done" button is visible at the bottom of the browser window.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 8 of 55

Slide 6 consists of a tabulated set of mathematical functions along with descriptions. The mathematical functions in PHP will only handle values within the range of the integer and float types. The simplest math functions are those that only require one parameter. This parameter is usually a number, which is accepted either as a variable or as a string. Here, the basic math such as adding, subtracting, multiplying, and dividing can be done using mathematical operators.

Then using slides 7 and 8, explain the students briefly about the math functions involved in the program.

Slides 9 to 12

Let us learn about string functions.

String Functions
1-4

- ◆ Operate on character type of data
- ◆ Table lists some of the string functions in PHP:

Function Name	Syntax	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
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

Version 1.0 © Aptech Limited.
Functions in PHP / Session 11 / Slide 9 of 55

String Functions

2-4

Function Name	Syntax	Description
strrpos()	strrpos(string,find,start)	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strcmp()	strcmp(string1,string2,length)	String comparison of the first n characters

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 10 of 55

String Functions

3-4

- ◆ **string.php** - Use of string functions

Snippet

```
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");
?>
```

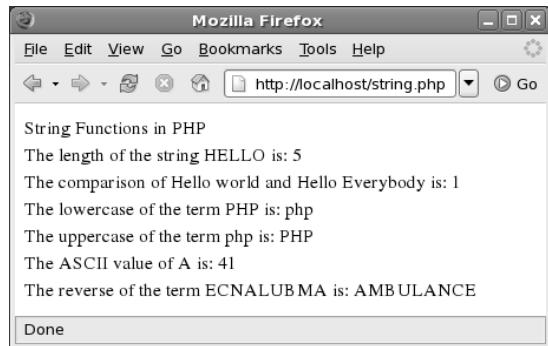
Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 11 of 55

String Functions

4-4

Displays the following output:



Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/string.php
String Functions in PHP
The length of the string HELLO is: 5
The comparison of Hello world and Hello Everybody is: 1
The lowercase of the term PHP is: php
The uppercase of the term php is: PHP
The ASCII value of A is: 41
The reverse of the term ECNALUB MA is: AMB ULANCE
Done

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 12 of 55

Slides 9 and 10 describe the string functions in PHP. The simplest string functions are those that require one parameter - the string itself, which is accepted either as a variable or as a single or double-quoted string. The `strlen()` function, for example, will return the length of the string that is given as a parameter.

Using slides 11 and 12, explain the students about the example code given for string functions along with the output as displayed in the browser window.

In-Class Question:

After you finish with the explanation of functions, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you define what is meant by a function?

Answer:

Functions are:

- named section of a program
- used to perform a specific task

Slides 13 to 16

Let us understand the date and time functions in PHP.

Date and Time Functions

1-4

- ◆ Enables to calculate the date and time on the system
- ◆ Table lists and describes some of the date and time functions :

Function Name	Syntax	Description
checkdate	checkdate(month,day,year)	Returns the value as 1 if the specified date is valid. A valid date contains: <ul style="list-style-type: none"> ◆ Month between 1 and 12 ◆ Day within the range of days for the specified month ◆ Year between 1 and 32767
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

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 13 of 55

Date and Time Functions

2-4

Function Name	Syntax	Description
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 zero s – seconds with leading zeros

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 14 of 55

Date and Time Functions

3-4

- ◆ **date_time.php** - Use of common date and time functions

Snippet

```
<?php
date_default_timezone_set('Asia/Calcutta');

echo "Today is : " .date("l");

$Today_Date=getdate();

$current_month=$Today_Date['month'];

echo "<br>";

echo "Current month is: ";

echo $current_month;

?>
```

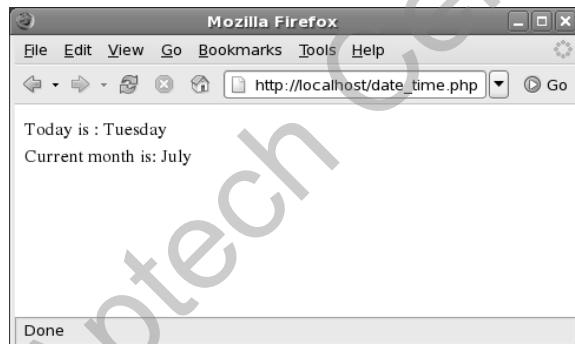
Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 15 of 55

Date and Time Functions

4-4

Displays the following output:



The "1" is the argument string, which corresponds to the textual representation of the day of the week and will return the day.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 16 of 55

Slides 13 and 14 describe about the date and time functions. The date and time functions allow to retrieve the date and time from the server where the PHP script runs. The date and time functions are used to format the date and time in several ways. In general, these functions depend on the locale settings of the server. Additionally, tell the students that PHP date/time functions are part of the PHP core and hence, no installation is required for it.

Using slides 15 and 16, explain the students the given example along with its output. Give them some more examples for better understanding.

Slides 17 to 20

Let us understand error handling functions.

Error Handling Functions 1-4

- ◆ Defines the error handling rules and modify the way the errors are handled
- ◆ Table 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(e rror_ 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
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)

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 17 of 55

Error Handling Functions 2-4

Table 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

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 18 of 55

Error Handling Functions

3-4

- ◆ **user_error.php** - Use of error handling functions

Snippet

```
<?php
$num1=0;

if($num1==0)
{
echo "Dividing by zero";

trigger_error("Cannot divide by zero", E_USER_ERROR);
}

else
{
$B=100/$num1;
}

?>
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 19 of 55

Error Handling Functions

4-4

Displays the following output:

The value of \$num1 variable is tested.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 20 of 55

The default error handling mechanism in PHP is very simple. An error message with filename, line number, and a message describing the error is sent to the browser. When creating scripts and Web applications, error handling is an important part. If the code lacks error-checking code, the program may look very unprofessional and may be open to security risks.

Also, explain the students about the custom error handling functions. Using slides 17 and 18, explain the students the tabulated information of various error handling functions.

The code given in slide 19 describes the use of error handling functions. Using the slide 20 show them the output for the given code.

Slides 21 to 25

Let us understand user-defined functions.

User-defined Functions  **1-5**

- ◆ Function can be defined or created
- ◆ Function definition contains the code to be executed
- ◆ Defining a function is as follows:

Syntax

```
function function_name (arg1, arg2, arg3, ...)
{
    statement list
}
```

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 21 of 55

User-defined Functions  **2-5**

- ◆ The `return expr` statement within the body of the function is used to return a value from a function
- ◆ Snippet accepts an argument, `$x`, and returns its square

Snippet

```
function square ($x)
{
    return $x*$x;
}
```

- ◆ To execute the function, invoke the function as follows:

Syntax

```
fun_name();
```

where,

fun_name – specifies the name of the function

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 22 of 55

User-defined Functions

3-5

- ◆ A PHP code can be included inside a function, other functions, and class definitions
- ◆ Rules to define function names are similar to the rules for defining labels in PHP
- ◆ Functions are not required to be defined before referencing
- ◆ When defining a function in a conditional manner, the script must first process the function definition before invoking the function

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 23 of 55

User-defined Functions

4-5

- ◆ **user_func.php** - Use of user defined functions

Snippet

```
<?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();
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 24 of 55

User-defined Functions**5-5**

```
echo "<br>";  
// A function to display the text  
function Display()  
{  
    echo "LEARNING PHP IS FUN";  
}  
Display();  
?>
```

Displays the following output:

Version 1.0 © Aptech Limited.Functions in PHP / Session 11 / Slide 25 of 55

Using slides 21 to 23, explain the students about user-defined functions in detail. You start explaining the points given in the slides.

The user-defined functions are simply defined as the functions that are written by the user. The syntax of a user-defined function has a unique function name, followed by a number of arguments separated by comma. The user can include any PHP code and even another function.

Functions provide a way to group together related statements into a cohesive block. For reusable code, a function saves duplicating statements and makes maintenance of the code easier.

Using slides 24 and 25, explain the students the given code and the corresponding output.

Slides 26 to 31

Let us understand passing arguments to functions.

Passing Arguments to Functions

1-6

- ◆ 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

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 26 of 55

Passing Arguments to Functions

2-6

- ◆ 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

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 27 of 55

Passing Arguments by Value

3-6

- ◆ When an argument is passed by value it must:
 - ◆ Prefix with the dollar (\$) sign
 - ◆ Be any valid expression which are evaluated and assigned to the corresponding variable

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 28 of 55

Passing Arguments by Value

4-6

- ◆ **arg_value.php** - Passing arguments by value

Snippet

```
<?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
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 29 of 55

Passing Arguments by Value

5-6

```

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);
?>

```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 30 of 55

Passing Arguments by Value

6-6

Displays the following output:



The subtraction() function subtracts the variable **\$B** from **\$A** and stores the resultant value in **\$C**.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 31 of 55

Slides 26 to 28 describes the passing of arguments by value. Explain the students about the arguments that can be passed to a function, using arguments list separated by commas. Arguments are passed by value as default in PHP, and the value is assigned directly in the function definition.

Using slides 29 and 30, explain them the given code that describes the use of passing of arguments by value in detail. Show them slide 31 that displays the output for the given code in the browser window.

In-Class Question:

After you finish with the explanation of user-defined functions, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What are the three different ways of passing arguments to a function?

Answer:

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

Slides 32 to 36

Let us understand the passing argument by reference.

Passing Argument by Reference

1-5

- ◆ When an argument is passed by reference it must:
 - ◊ Be a variable
 - ◊ Prefix the arguments with the ampersand (&) sign to indicate that the value is passed by reference

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 32 of 55

Passing Argument by Reference

2-5

- ◆ **arg_ref.php** - Passing values to the function by reference

Snippet

```
<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
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 33 of 55

Passing Argument by Reference

3-5

- ◆ **arg_ref.php** - Passing values to the function by reference

Snippet

```
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
    //storing it in a variable
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 34 of 55

Passing Argument by Reference

4-5

- ◆ **arg_ref.php** - Passing values to the function by reference

Snippet

```
$C=$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);
?>
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 35 of 55

Passing Argument by Reference

5-5

Displays the following output:



The multiplication () function calculates
the product of two variable \$A and \$B and stores
the value in \$C.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 36 of 55

The variable can be passed by reference to a function so that the function can modify the variable. Explain them the general format and the points given in slide 32. It is to be noted that there is no reference sign present on a function call. The reference symbol & is given only on function definitions. Function definitions alone are enough to correctly pass the argument by reference.

References allow two variables to refer to the same content. In other words, a variable points to its content. Passing by reference allows two variables to point to the same content under different names. The ampersand is placed before the variable to be referenced.

Using slides 33 to 36, explain them the given examples with the output for the corresponding code.

Slides 37 to 40

Let us understand the setting default values.

Setting Default Values 1-4

- ◆ 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

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 37 of 55

Setting Default Values 2-4

- ◆ The use of default parameters

Snippet

```
function increment(&$num, $increment = 1)
{
    $num += $increment;
}
$num = 4;
increment($num);
increment($num, 3);
```

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 38 of 55

Setting Default Values

3-4

- ◆ **travel.php** - Assigning default values for an argument

Snippet

```
<?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();
?>
```

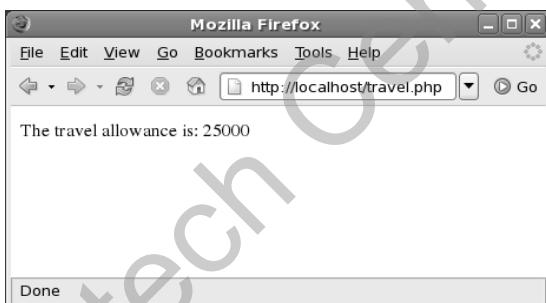
Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 39 of 55

Setting Default Values

4-4

Displays the following output:



The T_ALLOWANCE () function calculates
and displays the traveling allowance.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 40 of 55

Using slide 37, explain about setting default values for arguments. When designing the functions it is often helpful to be able to assign default values for parameters that are not passed. It will save the effort of having to pass in parameters most of the time if they are usually the same. To define the default parameters for a function simply add the constant value likely to be set after the variables.

Using slides 38 to 40, explain the students each line of code with neat explanation and show them the corresponding output.

Slides 41 to 44

Let us understand the returning values from functions.

Returning Values from Functions

1-4

- ◆ The `return` statement in a function returns the value from the function
- ◆ 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
- ◆ The reference operator is required to be used while declaring a function as well as when assigning the returned value to the variable

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 41 of 55

Returning Values from Functions

2-4

- ◆ `hra_func.php` - The use of return keyword to calculate the house rent allowance

Snippet

```
<?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;
}
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 42 of 55

Returning Values from Functions

3-4

```
//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;

?>
```

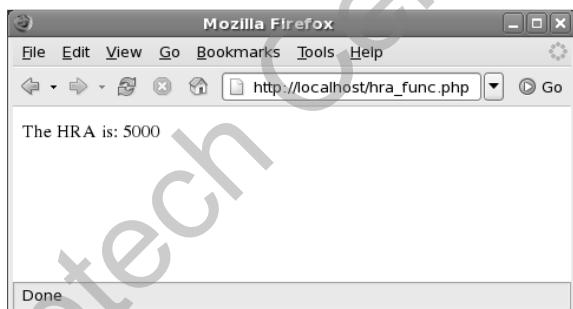
Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 43 of 55

Returning Values from Functions

4-4

Displays the following output



The HRA () function returns the house rent allowance, calculated at the rate of 25% of basic salary.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 44 of 55

Values are returned by using the optional `return` statement. Any type may be returned, including arrays and objects. This causes the function to end its execution immediately and pass control back to the line from which it was called.

The function parameters are specified after the function name, inside the parentheses. Slide 41 describes about the returning values. Using slides 42 to 44, explain students about the example code snippet that is used to calculate the house rent allowance.

Slides 45 to 50

Let us understand the nesting of functions.

Nesting of Functions

1-6

- ◆ One function can be dependent on another function in the program
- ◆ The execution of one function inside another function is called nesting of functions

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 45 of 55

Nesting of Functions

2-6

- ◆ **nested.php** - Use of nested functions

Snippet

```
<?php
//Assigning value to variable
$Basic_Sal=75000;
//Creating a function and passing value by reference
function HRA($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)
{
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 46 of 55

Nesting of Functions**3-6**

```
//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: ";
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 47 of 55

Nesting of Functions**4-6**

```
//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
```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 48 of 55

Nesting of Functions

5-6

```
$Net_Sal=75000+$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>";
```

Version 1.0 © Aptech Limited.

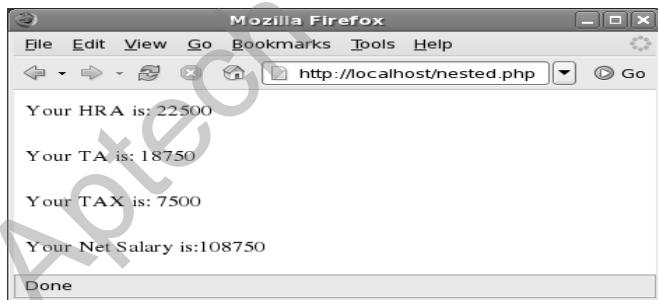
Functions in PHP / Session 11 / Slide 49 of 55

Nesting of Functions

6-6

```
Net_Salary($Basic_Sal);
echo "<br>";
?>
```

Displays the following output:



The Net_Salary() function calls three functions, HRA(), TA(), and TAX().

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 50 of 55

Explain to students the concept of nested functions using slide 45. When defining a function within another function, it will not exist until the parent function has been executed. Once the parent function is executed, the nested function is defined and similar to any other function; it is accessible from anywhere within the current document. If the nested function is present in the code, you can run the outer function only once. If it is called again, the inner functions will try to get declared again, which will result in an error.

Using slide 45, describe the nesting of functions in brief. Use slides 46 to 49 to explain the code that generates and displays calculated HRA, TA, TAX, and Net Salary. Show them the corresponding output displayed in slide 50.

Slides 51 to 54

Let us understand recursion.

Recursion 1-4

- ◆ 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

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 51 of 55

Recursion 2-4

- ◆ **recursion.php** - Use of recursive functions in programs to calculate the factorial

Snippet

```
<?php  
//Assigning Value to a variable  
$A=10;  
//Creating a function to calculate the factorial  
function factorial($A)  
{  
    //Calculating the factorial  
    if($A<=1)  
    {
```

Version 1.0 © Aptech Limited. Functions in PHP / Session 11 / Slide 52 of 55

Recursion

3-4

```

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;
?>

```

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 53 of 55

Recursion

4-4

Displays the following output:



The factorial () function returns 1 if the input is 0 or 1.

Version 1.0 © Aptech Limited.

Functions in PHP / Session 11 / Slide 54 of 55

Use slide 51 to explain about recursion. A recursive function is one that calls itself, either directly or through a cycle of function calls.

Give an analogy for students to understand the process of recursion. Ask them to consider the task of building a wall that is five feet tall; to accomplish this, we can build a four-foot high wall and then add an extra foot of bricks. To build the four-foot high wall, we build a three-foot one and then add one more layer. This way we keep reducing till it becomes the smallest and then go upwards till we reach the goal.

If you relate this to programming, we could say 'build wall' is a function takes a height and if that height is greater than one, first calls itself to build a lower wall, and then adds one a foot of bricks.

Using slides 52 to 54, explain the code snippet and the output produced while executing the given code.

Additional Information:

For more information on recursion, visit the following links:

<http://www.sitepoint.com/understanding-recursion/>

<http://www.lornajane.net/posts/2012/php-recursive-function-example-factorial-numbers>

<http://www.tuxradar.com/practicalphp/4/18/0>

<http://www.phpfresher.com/tag/recursion-function/>

Slide 55

Let us summarize the session.



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

Use slide 55 to list and explain the summary of this session.

The summary points are as follows:

- 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

14.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to working with arrays.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 16 –Working with Arrays

16.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

16.1.1 Objectives

At the end of this session, the learners 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

16.1.2 Teaching Skills

To teach this session, you should be well-versed with the concepts related to arrays in PHP. You should be able to explain the use of arrays. You should be able to describe the process of merging arrays. You should familiarize yourself with the use of single dimensional array, multi-dimensional array and also array-related functions.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 2 of 45

Tell the students that they will be introduced to arrays in PHP and learn the use of arrays. They will also be taught the process of merging arrays. In addition, they will understand use of array related functions.

16.2 In-Class Explanations

Slide 3

Let us introduce arrays in PHP.

Introduction

- ◆ An array is a variable that can store a list of values referred by the same name
- ◆ Types of arrays are as follows:
 - ◆ Single-dimensional
 - ◆ Multi-dimensional

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 3 of 45

With slide 3, give a brief introduction to arrays in PHP. Tell the students that an array is a data structure that stores more than one value under a single name. For example, a list of student names can be stored as an array.

Give the students a real-world analogy to understand arrays. Tell them they can think of it as similar to a shopping list. Just as you use a single paper to record and remember all the items that are needed to be bought, in the same manner, you can create an array to store various items under a single name. Sometimes, you even note down the respective quantities beside each item on a shopping list, likewise, you can create multidimensional arrays to store two or more columns of data.

Additional Information:

For more information on arrays, visit the following links:

http://www.w3schools.com/php/php_arrays.asp

http://www.tutorialspoint.com/php/php_arrays.htm

Slide 4

Let us understand arrays in PHP.

The screenshot shows a presentation slide with a blue header bar containing the word 'Array'. Below the header is a large white content area with a blue border. Inside this area, there is a bulleted list of nine points. At the bottom of the slide, there is a thin blue footer bar with small text. A large watermark reading 'For Author Only' is diagonally across the slide.

- ◆ Is a variable that can store a set of values of the same data type
- ◆ Each element of an array can be referred by an array name and an index
- ◆ Array index
 - ◆ Is used to access an element
 - ◆ Can be a number or a string
- ◆ If index is string, then array is an associative array
- ◆ If index is number, then array is an indexed array
- ◆ By default, the index value in an array starts at zero

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 4 of 45

Using slide 4, explain the students about arrays briefly.

Using arrays, the data can be stored and organized more quickly and efficiently. Arrays are the ordered list of elements. The index position is used to refer the elements in an array. The indexed array is defined as an array with a numeric index while the array that has named positions is called an associative array.

Slide 5

Let us understand initializing an array.

The screenshot shows a slide titled "Initializing an Array". It contains a bulleted list with two items:

- ◆ Two ways of initializing an array 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

At the bottom of the slide, there is a footer bar with the text "Version 1.0 © Aptech Limited." on the left and "Working with Arrays / Session 13 / Slide 5 of 45" on the right.

Slide 5 describes the process of initializing an array. Explain the students the two ways of initializing an array.

Give the following example to illustrate these two approaches.

First Approach:

```
<?php  
$fruits = array(  
    "1" => "apple",  
    "2" => "orange",  
) ;  
...  
...
```

Second Approach:

```
<?php  
$arr[] = 56;  
...  
...
```

In-Class Question:

After you finish explaining about arrays, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you state the definition of an array?

Answer:

- Array is a variable that can store a set of values of the same data type.
- Each element of an array can be referred by an array name and an index.

Slide 6

Let us understand the `array()` function.

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

Syntax

```
$array_name = array([key => ] value, [key => ] value)
```

Where,

- ◆ `array_name` - specifies the array name
- ◆ `key` - specifies the index value of the array element
- ◆ `value` - specifies the value of the element
- ◆ Using the `array()` function, both the indexed and the associative arrays can be initialized

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 6 of 45

Slide 6 shows how to define `array()` function. Tell the students about `array()` function briefly with the points given on slide 6. It is to be noted that this function uses key-value pairs separated by a comma to create an array. Show the syntax for declaring an array with a specified value and explain each of the keywords elaborately.

Slides 7 and 8

Let us understand indexed arrays.

1-2

2-2

- ◆ 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

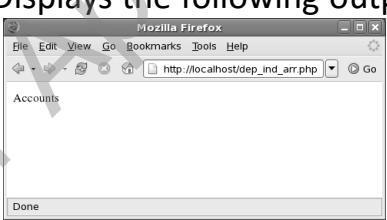
- ◆ The index value can start with any integer, such as 1, 20, or 123

- ◆ Creating an indexed array named **department**

Snippet

```
<?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];
?>
```

Displays the following output:



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**.

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 7 of 45

Using slide 7, introduce indexed arrays. The indexed array is defined as a numeric array with a numeric index. Here, the values are stored and accessed in linear fashion. The strings, numbers, and any other objects can be stored in an array, but their index will be presented as numbers.

Use slide 8 to explain the example code given. Highlight the indexed array that is defined in the code. Show the students the corresponding output that is displayed in the browser window.

Slides 9 and 10

Let us understand associative arrays.

Associative Arrays 1-2

- ◆ Is an array where the index type is a string
- ◆ The index value must be specified within double quotes

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 9 of 45

Associative Arrays 2-2

- ◆ Creating an associative array named **associate**

Snippet

```
<?php
$associate = array("a" => 'Finance', "b" => 'Sales', "c" => 'HR',
                   "d" => 'Purchase');
echo "The value of the associative array is: ";
echo $associate["c"];
?>
```

Displays the following output:



In the code, the array index type is a string.
 The index value starts from **a**. and 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".

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 10 of 45

Slide 9 describes about associative arrays in PHP. The associative array has strings as index values. This type of array stores element values in association with key values rather than in a strict linear index order.

The associative array and numeric array are similar in terms of functionality. The index of an associative array will be a string that establishes a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, the employees names could be used as the keys in an associative array, and the value would be their respective salary. Likewise, use the example given in slide 10 and show them the output obtained for the corresponding code respectively.

Slides 11 to 13

Let us learn about array identifier.

Array Identifier 1-2

- ◆ Enables to initialize the value of a specific element in an array

Syntax

```
$array_name[key] = "element_value";
```

where,

- ◆ **array_name** - specifies the name of the array
- ◆ **key** - Specifies the index value of the array element
- ◆ **element_value** - specifies the value assigned to the element of the array

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 11 of 45

Array Identifier 2-3

- ◆ Using array identifiers to create an array named **department**

Snippet

```
<?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>";
}
?>
```

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 12 of 45

Array Identifier

3-3

Displays the following output:

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.

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 13 of 45

Use slide 11 to explain the students about the array identifier. Another way to create PHP arrays is with array identifier. In general, an array index starts at 0 when no index value is specified, and the index value increases as and when an element is appended in the array. When the value of index is specified, an array with named elements is created.

Using slides 12 and 13, explain the example program that is used to display the set of array elements, explicitly specified using the array identifier.

Slides 14 to 17

Let us learn about merging arrays.

Merging Arrays

1-4

- The `array_merge()` function is used to combine the element values of two or more arrays

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

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 14 of 45

Merging Arrays

2-4

- ◆ Demonstrating the merging of the arrays

Snippet

```
<?php
$ITdept = array(0 => "Testing", 1 => "Training");
$Salesdept = array(0 => "Advertising", 1 => "Marketing");
$num_of_element = count($ITdept);
for ($i=0; $i< $num_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] $rec[value] ";
    echo "<br>";
}
echo "<br>";
echo "$rec[key] $rec[value] ";
```

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 15 of 45

Merging Arrays

3-4

```
$AdminDept = array_merge($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);
for ($i=0; $i< $num2_of_element; $i++)
{
    $rec = each($AdminDept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
?>
```

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 16 of 45

Merging Arrays

Displays the following output:

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.

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 17 of 45

Slide 14 describes about merging arrays in PHP. The `array_merge()` function is used to merge the elements of one or more arrays together, so that the values of one array are appended to the end of the second one. It returns the resulting array. If the string keys are same for the input arrays, then the values will be overwritten and when the array has numeric keys, the values in first array will be appended with the values of the second array.

When the input array has values with numeric keys, they will be renumbered with incrementing keys starting from zero in the result array.

Using slides 15 and 16, explain the example code that is used to demonstrate merging of arrays. Show them the corresponding output, that is given in slide 17.

In-Class Question

After you finish with explaining the merging of arrays, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What is the function used to merge arrays?

Answer:

The `array_merge()` function is used to combine the element values of two or more arrays.

Slides 18 to 22

Let us understand multi-dimensional arrays.

Multi-dimensional Arrays

1-5

- ◆ Contains one array stored within another
- ◆ In a multi-dimensional array, each element is an array
- ◆ Each element requires an array name and multiple set of indices

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

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 18 of 45

Multi-dimensional Arrays

2-5

- ◆ Illustrating the use of multi-dimensional arrays

Snippet

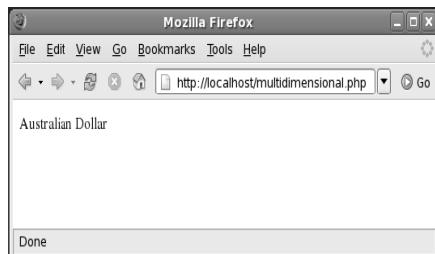
```
<?php
$countrymdlist = array(
    "USA" => array(
        "Capital" => "Washington D.C.",
        "Currency" => "US Dollar",
        "England" => array(
            "Capital" => "London",
            "Currency" => "Pound Sterling"),
        "Australia" => array(
            "Capital" => "Canberra",
            "Currency" => "Australian Dollar"),
        "New Zealand" => array(
            "Capital" => "Wellington",
            "Currency" => "NZ Dollar"));
echo $countrymdlist["Australia"]["Currency"];
?>
```

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 19 of 45

Multidimensional Arrays

3-5

Displays the following output:



In the code, `country_mdlist` is a multidimensional associative array.

It contains key indices such as `USA`, `England`, `Australia`, and `New Zealand`.

Each array element of a multidimensional array includes another array within it containing key indices such as `Capital` and `Currency`.

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 20 of 45

Multidimensional Arrays

4-5

- Creating an array that stores the employee commission details

Snippet

```
<?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];?>
```

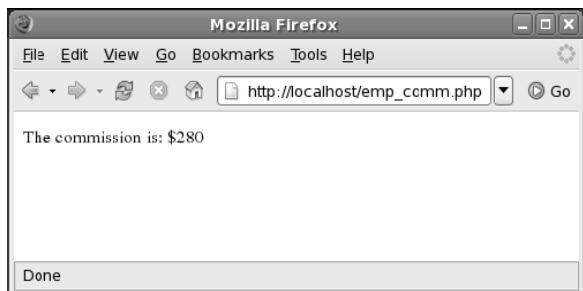
Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 21 of 45

Multi-dimensional Arrays

5-5

Displays the following output:



In the code, both associative and indexed indices are used to create a multi-dimensional array.

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 22 of 45

Using slide 18, explain about multi-dimensional array in PHP. A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

Use slides 19 and 20 to explain the example code that illustrates the use of multi-dimensional array.

Using slides 21 and 22, explain the example code that is used to display the employee details stored in an array.

Slide 23

Let us understand the array-related functions.

Array-Related Functions

- ◆ Some of the array-related functions supported by PHP are as follows:
 - ❖ `sort()` Function
 - ❖ `rsort()` Function
 - ❖ `arsort()` Function

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 23 of 45

Using slide 23, explain the students about the array related functions. In the subsequent slides, each of the function is explained with syntax and examples.

Slides 24 to 26

Let us understand `sort()` function.

sort() Function

1-3

- ◆ Arranges the element values in alphabetical order

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

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 24 of 45

sort() Function

2-3

- Illustrating the sorting of the department array

Snippet

```
<?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>";
}
?>
```

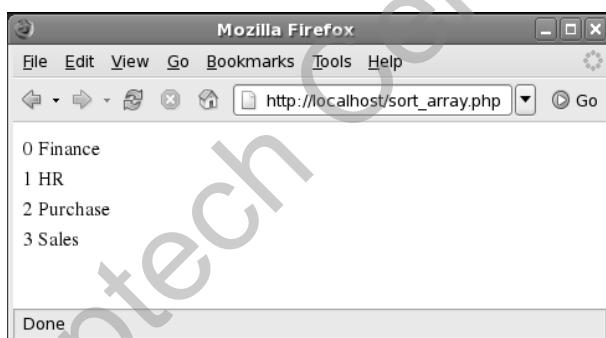
Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 25 of 45

sort() Function

3-3

Displays the following output:



The code 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.

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 26 of 45

Slide 24 describes about `sort()` function. The `sort()` function sorts the elements of an array in alphabetical order.

Using slide 25, explain the example code that illustrates the sorting of an array named `department`. Use slide 26 to show them the output that is displayed in the browser window for the corresponding code.

Slides 27 to 29

Let us understand `rsort()` function.

rsort() Function

1-3

- ◆ Sorts the element values in descending alphabetical order

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

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 27 of 45

rsort() Function

2-3

- ◆ Illustrating the use of the `rsort()` function to display the values of the department array in the descending alphabetical

Snippet

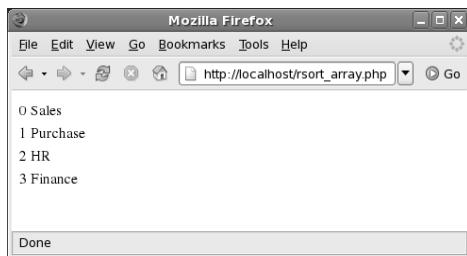
```
<?php
$department[0] = "Finance";
$department[1] = "Sales";
$department[2] = "HR";
$department[3] = "Purchase";
rsort($department);
$no_of_element = count($department);
for ($i=0; $i < $no_of_element; $i++)
{
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 28 of 45

rsort() Function

3-3

Displays the following output:



The code 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.

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 29 of 45

Using slide 27, explain the `rsort()` function in PHP. The `rsort()` function is used to sort the array in an reverse order from highest to lowest. It is to be noted that this function assigns new keys for the elements in array. It will remove any existing keys that are assigned, rather than just reordering the keys. It is said that this function behaves opposite to the `sort` function.

Slides 28 and 29 consist of example program for `rsort()` function. Describe each line of code elaborately and show them the corresponding output that is displayed in the browser window.

Slides 30 to 32

Let us understand `arsort()` function.

arsort() Function

1-3

- ◆ Similar to `rsort()` function
- ◆ The only difference between `rsort()` and `arsort()` function is that the `arsort()` function can sort both associative and indexed arrays

Syntax

```
arsort (ArrayName)
```

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 30 of 45

arsort() Function

2-3

- ◆ Demonstrating the use of the arsort () function on the array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
arsort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

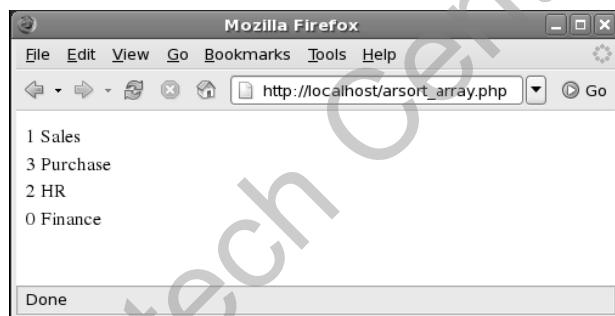
Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 31 of 45

arsort() Function

3-3

Displays the following output:



Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 32 of 45

The `arsort()` function in PHP sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant. Use slide 30 to explain the students about `arsort()` function.

The `arsort()` function can be explained more clearly with the given example that displays the set of sorted values. The example code that describes the `arsort()` function is given in slide 31. Show them the corresponding output displayed in slide 32.

Slides 33 to 43

Let us see other array-related functions.

Other Array-Related Functions

1-11

Table lists different 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
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

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 33 of 45

Other Array-Related Functions

2-11

- ◆ To flip the element values to the index values and the index values to the element values of the department array

Snippet

```

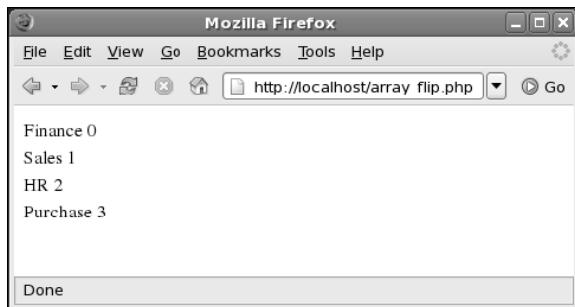
<?php
$department[0] = "Finance";
$department[1] = "Sales";
$department[2] = "HR";
$department[3] = "Purchase";
$dept = array_flip($department);
$no_of_element = count($department);
for ($i=0; $i < $no_of_element; $i++)
{
    $rec = each($dept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>

```

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 34 of 45

Other Array-Related Functions**3-11**

Displays the following output:

**Other Array-Related Functions****4-11**

- Reversing the order of elements of the department array

Snippet

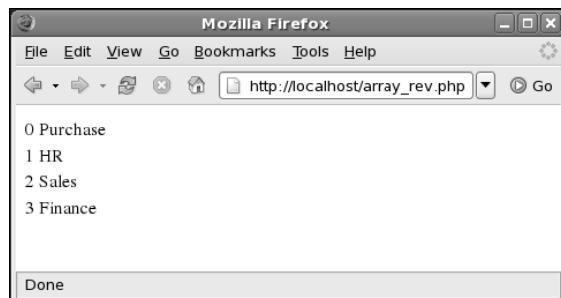
```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_reverse($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($dept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 36 of 45

Other Array-Related Functions**5-11**

Displays the following output:

**Other Array-Related Functions****6-11**

- Viewing all the key values of the department array

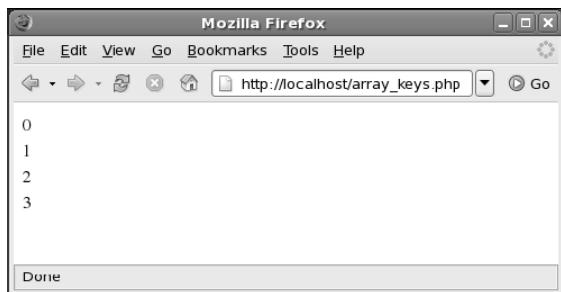
Snippet

```
<?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>";
}
?>
```

Other Array-Related Functions

7-11

Displays the following output:



Other Array-Related Functions

8-11

- Removing an element value from the department array

Snippet

```
<?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>";
}
?>
```

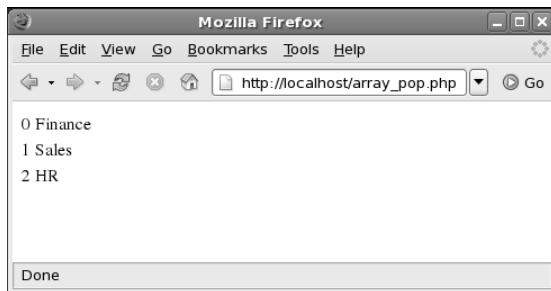
Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 40 of 45

Other Array-Related Functions

9-11

Displays the following output:



Other Array-Related Functions

10-11

- Adding an element value to the department array

Snippet

```
<?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>";
}
?>
```

Version 1.0 © Aptech Limited.

Working with Arrays / Session 13 / Slide 42 of 45

Other Array-Related Functions

11-11

Displays the following output:

0 Finance
1 Sales
2 HR
3 Purchase
4 Marketing

Done

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 43 of 45

PHP provides many array-related functions. Some of the array-related functions are tabulated in slide 33. Start describing each of the function using the given description for it. Then, continue explaining each of the code snippet that is given as examples. Slides 34 and 35 consist of the example code that is used to flip the element values to the index values and index values to the element values of the department array. With that, tell them the exact use of `array_flip()` function and the general syntax for specifying it to the code.

Using slides 36 and 37, teach them how to write the code for reversing the order of elements of the department array.

To display only the key elements of an array, the `array_keys()` function is used. Use slides 38 and 39 to teach them the exact use of `array_keys()` function.

Removing an element value from department array is done using `array_pop()` function. It is to be noted that the `array_pop()` function is used to pop the element off the end of array. This is explained briefly using the code given in slides 40 and 41.

The `array_push()` is the function that behaves opposite to the `array_pop()` function. It is used to push one or more elements onto the end of array. Use slide 42 and explain the given example code for adding an element value to the department array. Show them the corresponding output for the code using slide 43.

Additional Information:

For more information on multi-dimensional array and array functions in PHP, visit the following links:
http://www.w3schools.com/php/php_arrays_multi.asp

http://webcheatsheet.com/php/multidimensional_arrays.php

http://www.w3schools.com/php/php_ref_array.asp

http://www.tutorialspoint.com/php/php_array_functions.htm

Slides 44 and 45

Let us summarize the session.

Summary

1-2

- ◆ 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

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 44 of 45

Summary

2-2

- ◆ 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

Version 1.0 © Aptech Limited. Working with Arrays / Session 13 / Slide 45 of 45

Use slides 44 and 45 to list and explain the summary of this session.

The summary points are as follows:

- 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

16.3 Post-Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to handling databases with PHP.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 18 –Handling Databases with PHP

18.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

18.1.1 Objectives

At the end of this session, the learners 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

18.1.2 Teaching Skills

To teach this session, you should be well versed with the concepts related to database APIs in PHP. You should be able to explain the process of connecting to a database. You should be able to describe the use of data access functions. You should familiarize yourself with the topics related to SQL queries and HTML tables.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 2 of 50

Tell them that they will be introduced to database APIs in PHP. They will also learn the process of connecting to a database. They will be taught about the use of data access functions. In addition, they will also understand about SQL queries and HTML tables using SQL queries.

18.2 In-Class Explanations

Slide 3

Let us introduce the concept of RDBMS in PHP.

Introduction

- ◆ Relational Database Management System (RDBMS):
 - ◆ Stores data in tables that are linked with common fields, known as keys
 - ◆ MS Access, Oracle, and MySQL are the different types of RDBMS
- ◆ In the Linux operating system, MySQL is mostly preferred because it is an open source software, and is easy to use

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 3 of 50

Using slide 3, define RDBMS with respect to PHP.

Relational Database Management System (RDBMS) is used as a data manager, that is, it helps to create, manage, and manipulate data. The data is represented in rows and columns, in a table format in an RDBMS. MySQL is a popular open source RDBMS.

Structured Query Language (SQL) is used in PHP to communicate with the relational database systems such as MySQL and Oracle. The SQL commands are categorized as Data Manipulation Language (DML) and Data Definition Language (DDL).

The syntax for DML is standardized while DDL has no standardized syntax. Explain the students about DML and DDL in brief.

Additional Information:

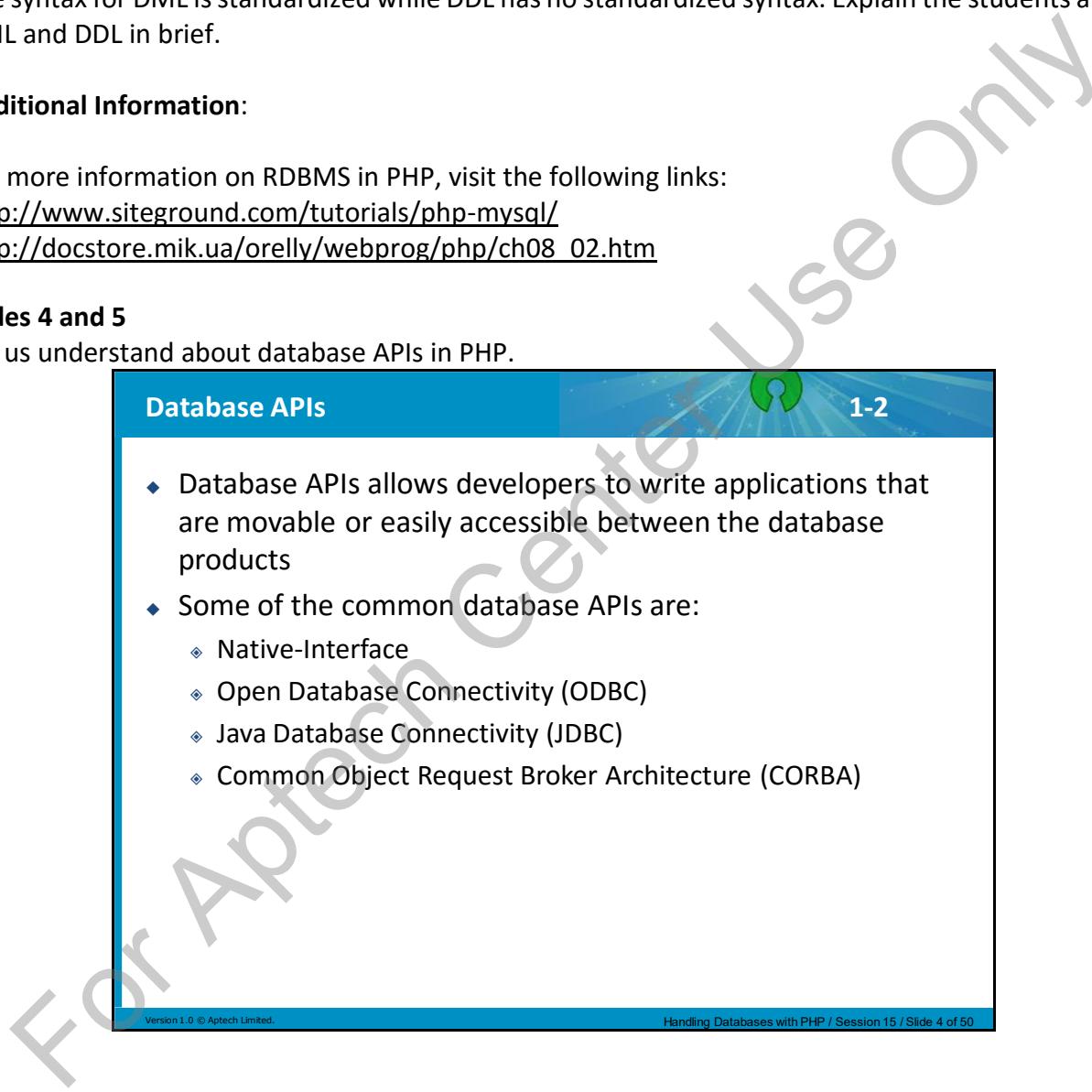
For more information on RDBMS in PHP, visit the following links:

<http://www.siteground.com/tutorials/php-mysql/>

http://docstore.mik.ua/orelly/webprog/php/ch08_02.htm

Slides 4 and 5

Let us understand about database APIs in PHP.



Database APIs 1-2

- ◆ 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)
 - ❖ Common Object Request Broker Architecture (CORBA)

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 4 of 50

Database APIs

2-2

- ◆ 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
 - ❖ Steps for connecting to a database are as follows:
 - Open the database connection
 - Work with the database
 - Close the database connection

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 5 of 50

The set of classes, methods, functions and variables that an application needs are defined in an Application Programming Interface (API). The PHP extensions are used to expose APIs in PHP applications that needs to communicate with the databases.

The API and steps for connecting to a database is explained using slides 4 and 5.

Slides 6 to 8

Let us understand about connecting to the MySQL server.

Connecting to the MySQL Server

1-3

- ◆ PHP and MySQL are automatically installed while customizing the installation of Linux operating system
- ◆ A connection needs to be establish to the MySQL server and PHP with the help of `mysql_connect()` function

This function takes three arguments:

 - ❖ Name of the machine on which the database is running
 - ❖ Database username
 - ❖ Database user password

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 6 of 50

Connecting to the MySQL Server

2-3

- ◆ Connecting to the MySQL server is as follows:

Syntax

```
$link_id = mysql_connect("host_name", "user_name", "password");
```

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 username
- password** - specifies the password to connect to the database
- link_id** - stores the return value of the connection

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 7 of 50

Connecting to the MySQL Server

3-3

- ◆ **mysql.php** - Connecting to the MySQL server

Snippet

```
<?php  
$link_id  
mysql_connect("localhost", "root", "abc123");  
?>
```

In the code,
root is the username.
localhost is the server name.
abc123 is the password.

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 8 of 50

Slides 6 and 7 describe the process of connecting to the MySQL server. Explain the students how to establish a connection to MySQL from inside a PHP script. It is simple and easy to perform basic queries in MySQL. The `mysql_connect` function that returns a resource called pointer to the database is used to connect to MySQL.

Show the students the snippet given in slide 8 and explain how the connection is established.

In-Class Question:

After you finish explaining about RDBMS and MySQL, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you describe RDBMS?

Answer:

Relational Database Management System (RDBMS):

- Stores data in tables that are linked with common fields, known as keys
- MS Access, Oracle, and MySQL are the different types of RDBMS

Slides 9 to 19

Let us understand the working with the database.

Working with the Database
1-11

- ◆ Before starting work with the database, a connection needs to be established with the MySQL server
- ◆ PHP provides the following functions to work with the MySQL database:
 - ❖ **`mysql_list_dbs()`** - This function displays all the databases available on the server

The `mysql_list_dbs()` function is as follows:

Syntax

```
mysql_list_dbs($link_id);
```

where,

`link_id` - specifies the return value of the connection

Version 1.0 © Aptech Limited.
Handling Databases with PHP / Session 15 / Slide 9 of 50

Working with the Database

2-11

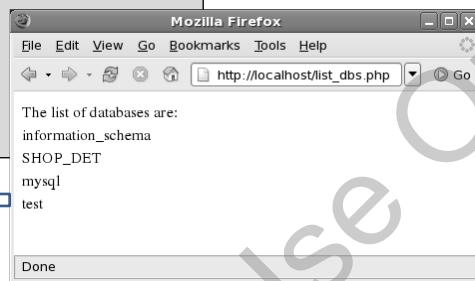
- ◆ **list_dbs.php** – Displaying all the databases present on the server

Snippet

```
<?php
$connect = mysql_connect('localhost', 'root', '');
$db_list = mysql_list_dbs($connect);
echo "The list of databases are:<br>";
while ($row = mysql_fetch_object($db_list))
{
echo $row->Database . "<br>";
}
?>
```

Displays the list of the databases that are present in the instance of MySQL.

Displays the output:



Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 10 of 50

Working with the Database

3-11

- ◆ **mysql_select_db()** - This function defines the database that will be used for the connection

The `mysql_select_db()` function is as follows:

Syntax

```
mysql_select_db("database name", $link id);
```

Where,

database_name - specifies the database name

link_id - specifies the return value of the connection

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 11 of 50

Working with the Database

4-11

- ❖ **mysql_select_db.php** - Connecting to the MySQL database

Snippet

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username, $password);

$mysql_db = mysql_select_db("mysql", $connect_mysql);
if (!$mysql_db)
{
    die("Connection failed");
}
else
{
    echo "Current Database is selected";
}
?>
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 12 of 50

Working with the Database

5-11

- ❖ **mysql_list_tables()** - This function displays a list of all the tables available in the specified database

The `mysql_list_tables()` function is as follows:

Syntax

```
mysql_list_tables("database_name", $link_id);
```

Where,

database_name - specifies the database name

link_id - specifies the return value of the connection

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 13 of 50

Working with the Database

6-11

- ❖ **listables.php** - To list all the tables of the MySQL database

Snippet

```
<?php
$dbname = 'mysql';

if (!mysql_connect('127.0.0.1', 'root', ''))
{
    echo 'Could not connect to mysql';
    exit;
}

$sql = "SHOW TABLES FROM $dbname";
$result = mysql_query($sql);

echo " The tables from the database are: <br><br>";
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 14 of 50

Working with the Database

7-11

```
if (!$result)
{
    $result = mysql_query($sql);

    echo " The tables from the database are: <br><br>";

    if (!$result)
    {
        echo "DB Error, Unable to list tables<br>";
        echo 'MySQL Error: ' . mysql_error();
        exit;
    }

    while ($row = mysql_fetch_row($result))
    {
        echo "Table: {$row[0]}<br>";
    }
}
?>
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 15 of 50

Working with the Database

8-11

Displays the following output:

```

Mozilla Firefox
File Edit View Go Bookmarks Tools Help
< > Stop Refresh http://localhost/listables.php Go
The tables from the database are:
Table: columns_priv
Table: db
Table: event
Table: func
Table: general_log
Table: help_category
Table: help_keyword
Table: help_relation
Table: help_topic
Table: host
Table: ndb_binlog_index
Done

```

All the tables available in the mysql database are listed and stored in the \$result variable.

Working with the Database

9-11

- ❖ **`mysql_num_rows()`** - This function displays the number of rows present in the specified table

The `mysql_num_rows()` function is as follows:

Syntax

```
mysql_num_rows("table_name");
```

Where,

table_name - specifies the name of the table for displaying the number of rows

Working with the Database 10-11

- ❖ **list_rows.php** - Listing the number of rows from a table in a database

Snippet

```
<?php
$connect = mysql_connect("localhost", "root", "");
mysql_select_db("mysql", $connect);
$result = mysql_query("SELECT * FROM user", $connect);

$rows = mysql_num_rows($result);
echo "The table contains $rows rows.<br>";
?>
```

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 18 of 50

Working with the Database 11-11

Displays the following output:

The number of rows from the table are listed.

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 19 of 50

Slide 9 describes the functions to work with in PHP. Using slide 10, explain the students about the `list_dbs` function. It lists the databases available on a MySQL server. It consists of only one parameter namely `connection` that specifies the MySQL connection.

The `mysql_select_db()` function is used to set the existing database on server. It returns TRUE or FALSE values on success or failure.

Slides 11 and 12 consist of the example code snippet that describe about `mysql_select_db()` function.

Explain the listtables function briefly using slides 13 to 16. This function is used to list tables in a MySQL database. The `mysql_num_rows()` function is explained with an appropriate example code that is given in slides 17 and 18. The `mysql_num_rows()` function returns the number of rows in a result.

Using slide 19, show them the corresponding output for the given code.

Slides 20 to 22

Let us understand about closing the connection.

Closing the Connection 1-3

- ◆ The connection with MySQL server can be closed with the help of the `mysql_close()` function
- ◆ The `mysql_close()` function is as follows:

Syntax

```
mysql_close($link_id);
```

Where,

link_id - specifies the return value of the connection

Closing the Connection 2-3

- ◆ **conn_close.php** - Closing the connection to the MySQL database

Snippet

```
<?php
$connect = mysql_connect("localhost", "root", "");
mysql_select_db("mysql", $connect);
$result = mysql_query("SELECT * FROM user", $connect);
$rows = mysql_num_rows($result);
echo "The table contains $rows rows.<br><br>";
mysql_close($connect);
echo "The connection to the database has been closed.";
?>
```

Closing the Connection 3-3

Displays the following output:



Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 22 of 50

The `mysql_close()` function closes a previously opened database connection. It is described in slide 20. It consists of connection parameter that requires the specification of MySQL connection that is to be closed. It returns TRUE value on success and FALSE on failure. Explain the code and the output given on slides 21 and 22.

Slides 23 to 26

Let us understand about data access functions.

Data Access Functions 1-4

- ◆ PHP provides the following functions for accessing data from the database:
 - ❖ `mysql_query()` - Executes MySQL query for retrieving data from tables and commands such as SELECT, SHOW, EXPLAIN, and DESCRIBE can be used with this function

The `mysql_query()` function is as follows:

Syntax

```
mysql_query(query, link_id);
```

Where,

query - specifies the MySQL query

link_id - specifies the return value of the connection

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 23 of 50

Data Access Functions 2-4

- ❖ **mysql_fetch_array()** - Retrieves the rows of the table and saves it as an array. It is an extended version of mysql_fetch_row() function

The mysql_fetch_array() function is as follows:

Syntax

```
mysql_fetch_array("table_name");
```

Where,

table_name - specifies the name of the selected table

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 24 of 50

Data Access Functions

3-4

- ❖ **mysql_field_len()** - Displays the length of the specified field

The mysql_field_len() function is as follows:

Syntax

```
mysql_field_len("table_name", "field_name");
```

Where,

table_name - specifies the table name

field_name - specifies the field name for which the length needs to be displayed

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 25 of 50

Data Access Functions

4-4

- ❖ **mysql_num_fields()** - Displays the number of fields in the specified table

The `mysql_num_fields()` function is as follows:

Syntax

```
mysql_num_fields("table_name");
```

Where,

table_name - specifies the table name

Version 1.0 © Aptech Limited.Handling Databases with PHP / Session 15 / Slide 26 of 50

Slide 23 describes about data access functions in PHP. Explain each of the functions in detail.

The `mysql_query()` function is used to send a MySQL query from within a PHP script. This function sends a unique query to the existing active database on the server. It takes two parameters namely, `query` and the `link_identifier`. The resultant array that is fetched by the `mysql_fetch_array()` function may be an associative array, a numeric array, or both. Explain to the students about `result` and `result_type` parameters using slide 24.

Using slide 25, explain the students about the `mysql_field_len()` that returns the length of the specified field. Explain about the `mysql_num_fields` function that is used to get number of fields in result. Use slide 26 to describe about `mysql_num_fields` function.

Slides 27 to 44

Let us learn about executing SQL queries in PHP.

Executing SQL Queries in PHP

1-18

- ◆ Before executing the SQL queries in PHP, a database connection must be established
- ◆ Create a table named `USER_CONTACT` in the `USER` database with the fields as shown in table

Field Name	Data Type	Constraint
<code>USER_ID</code>	INT	NOT NULL PRIMARY KEY
<code>USER_NAME</code>	CHAR(25)	NOT NULL
<code>USER_EMAIL_ID</code>	CHAR(25)	

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 27 of 50

Executing SQL Queries in PHP

2-18

- ◆ `mysqltable.php` - To create a table using SQL commands in PHP

Snippet

```

<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username, $password);
if($connect_mysql)
{
echo "Connection established<BR>";
}
else
{
die("Unable to connect to the database<BR>");
}
$sql_table = "CREATE TABLE USER_CONTACT(`USER_ID` INT NOT NULL PRIMARY KEY,
`USER_NAME` CHAR(25) NOT NULL, `USER_EMAIL_ID` CHAR(25).)";
if(mysql_query($sql_table))
{
echo "Table is created<BR>";
}

```

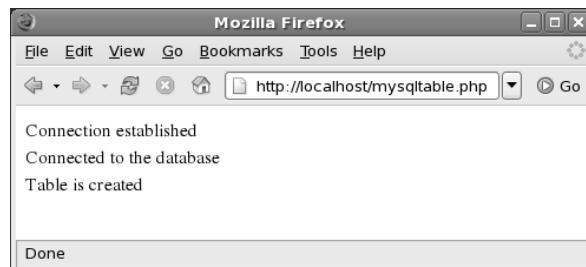
Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 28 of 50

Executing SQL Queries in PHP

3-18

```
else
{
    die("Unable to create the table<BR>");
}
?>
```

Displays the following output:



Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 29 of 50

Executing SQL Queries in PHP

4-18

- ◆ In the code, the `USER_CONTACT` table is created in the `USER` database
- ◆ The table is created using the `CREATE` command in MySQL
- ◆ The records in the table are inserted with the HTML FORM method

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 30 of 50

Executing SQL Queries in PHP

5-18

- ◆ **usercontact.php** - Inserting records in the USER_CONTACT table

Snippet

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username,
$password);
if($connect_mysql)
{
    echo "Connection established.";
}
else
{
    die("Unable to connect");
}
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 31 of 50

Executing SQL Queries in PHP

6-18

```
}
$db = "user";
$mysql_db = mysql_select_db($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')";
$result = mysql_query($sql_insert);
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 32 of 50

Executing SQL Queries in PHP

7-18

```
if($result)
{
echo "<BR><BR>The records have been added to the table.";
}
else
{
echo "Unable to insert records.";
mysql_error();
}
?>
```

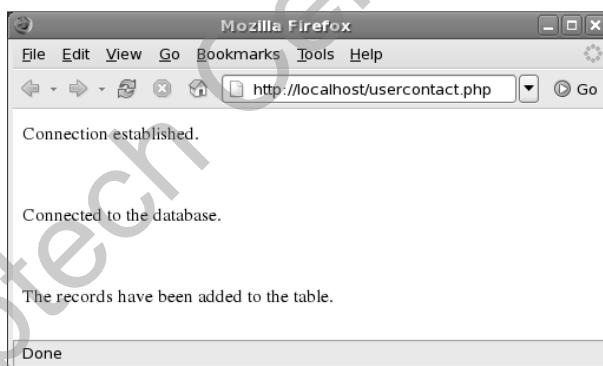
Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 33 of 50

Executing SQL Queries in PHP

8-18

Displays the following output:



Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 34 of 50

Executing SQL Queries in PHP

9-18

- Using the SELECT command, a data can be accessed from the tables
- displaytable.php** - Displaying the records of the USER_CONTACT table from the USER database

Snippet

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username,
$password);
if($connect_mysql)
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 35 of 50

Executing SQL Queries in PHP

10-18

```
{
echo "Connection established<br>";
}
else
{
die("Unable to connect<br>");
}
$mysql_db = mysql_select_db("USER");
if($mysql_db)
{
echo "Connected to the database<br>";
}
else
{
die("Unable to connect to the database<br>");
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 36 of 50

Executing SQL Queries in PHP

11-18

```

}
$sql_disp="SELECT * FROM USER_CONTACT;";
echo "<br>Displaying Records from the USER_CONTACT table:<br>";
$result = mysql_query($sql_disp);
while ($row = mysql_fetch_array($result))
{
    echo "$row[USER_ID] ";
    echo "$row[USER_EMAIL_ID] <br>";
}
?>

```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 37 of 50

Executing SQL Queries in PHP

12-18

Displays the following output:

Mozilla Firefox

File Edit View Go Bookmarks Tools Help

Connection established
Connected to the database

Displaying Records from the USER_CONTACT table :

101 ADAMS adams@abc.com
102 JERRY jerry@abc.com
103 DEXTER dexter@abc.com

Done

The records such as **USER_ID**, **USER_NAME**, and **USER_EMAIL_ID** from the **USER_CONTACT** table are displayed

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 38 of 50

Executing SQL Queries in PHP

13-18

- ◆ The DELETE and UPDATE commands enable to modify the contents of the table
- ◆ **delete_record.php** - To delete a record from the table

Snippet

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username, $password);
if($connect_mysql)
{
    echo "Connection established<br>";
}
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 39 of 50

Executing SQL Queries in PHP

14-18

```
else
{
    die("Unable to connect<br>");
}
$mysql_db = mysql_select_db("USER");
if($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'");
$result = mysql_query($sql_delete);
if($result)
{
    echo "Records Deleted: $result<br>";
}
```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 40 of 50

Executing SQL Queries in PHP

15-18

```

else
{
    echo "RECORDS NOT FOUND IN THE TABLE<br>";
    mysql_error();
}
?>

```

Displays the following output:



Executing SQL Queries in PHP

16-18

- ◆ **update_record.php** - Updating a record in the table

Snippet

```

<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username, $password);
if($connect_mysql)
{
    echo "Connection established<br>";
}
else
{
    die("Unable to connect<br>");
}
$mysql_db = mysql_select_db("USER");
if($mysql_db)

```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 42 of 50

Executing SQL Queries in PHP 17-18

```
{
    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'");
$result=mysql_query($sql_update);
if($result)
{
    echo "RECORDS UPDATED: $result<br>";
}
else
{
    echo "UNABLE TO UPDATE RECORDS<br>";
mysql_error();
}
?>
```

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 43 of 50

Executing SQL Queries in PHP 18-18

Displays the following output:

Connection established
Connected to the database
RECORDS UPDATED: 1

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 44 of 50

Explain the students the set of example code that is given in each slide. Tell them how the connection is established and closed, how the records are updated, inserted and deleted. Slide 27 describes the attributes and the values for the table that is created. Use slide 28 and 29 to explain the connection establishment to the database. Also, tell them about creating a table for user contact with the given attributes. Slide 30 describes the next set of code content. The corresponding data value are added to the table created. The code for inserting records are given in slides 31 to 33. Explain them each line of code briefly. Show them the corresponding code using slide 34. The code for displaying the records are explained using slides 35 to 38.

Likewise, explain each line of code that is used to update and delete records from the table. Use slides 39 to 43 to explain the set of code to perform update, delete, and closing connection. Show them the final output window with slide 44.

Slides 45 to 48

Let us learn about building HTML tables using SQL queries.

Building HTML Tables Using SQL Queries 1-4

- ◆ 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

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 45 of 50

Building HTML Tables Using SQL Queries 2-4

- ◆ **display_records.php** - Displaying all the records of the user_contact

Snippet

```
<HTML>
<BODY>
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username, $password);
if($connect_mysql)
echo "Connection established";
$mysql_db = mysql_select_db("USER");
if($mysql_db)
echo "<BR><BR>Connected to the database<BR><BR>";
echo "<TABLE BORDER BGCOLOR='WHITE'>";
```

Version 1.0 © Aptech Limited. Handling Databases with PHP / Session 15 / Slide 46 of 50

Building HTML Tables Using SQL Queries

3-4

```

echo "<TR><TH>USER_ID<TH>USER_NAME<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>

```

Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 47 of 50

Building HTML Tables Using SQL Queries

4-4

Displays the following output:



Version 1.0 © Aptech Limited.

Handling Databases with PHP / Session 15 / Slide 48 of 50

Slide 45 describes about building HTML tables using SQL queries. Tell them how the HTML tags are combined into the code and the data stored in the database are retrieved. Explain the students the code given in slides 46 and 47. Show them the corresponding output given in slide 48.

In-Class Question

After you finish with explaining the data access functions, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What is the use of `mysql_fetch_array()`?

Answer:

The `mysql_fetch_array()` is used to retrieve the rows of the table and saves it as an array. It is an extended version of `mysql_fetch_row()` function.

Additional Information:

For more information on SQL queries and related functions in PHP, visit the following links:

http://php.about.com/od/phpwithmysql/ss/mysql_php_3.htm

<http://www.tutorialspoint.com/mysql/mysql-select-query.htm>

http://www.w3schools.com/php/php_mysql_intro.asp

http://www.w3schools.com/php/php_mysql_select.asp

Slides 49 and 50

Let us summarize the session.

Summary
1-2

- ◆ Database APIs enable developers to write applications that are movable or easily accessible between the database products
- ◆ The common database 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 user password
- ◆ The connection with the server is established with the help of `mysql_connect()` function
- ◆ The basic PHP functions that are used with respect to the database are: `mysql_list_dbs()`, `mysql_select_db()`, `mysql_list_tables()`, and `mysql_num_rows()`

Version 1.0 © Aptech Limited.
Handling Databases with PHP / Session 15 / Slide 49 of 50

Summary

2-2

- ◆ The `mysql_close()` function closes the connection with the MySQL server
- ◆ The data access functions used in PHP are: `mysql_query()`, `mysql_fetch_array()`, `mysql_fetch_row()`, `mysql_fetch_field()`, `mysql_field_len()`, and `mysql_num_fields()`

Use slides 49 and 50 to list and explain the summary of this session.

The summary points are as follows:

- Database APIs enable developers to write applications that are movable or easily accessible between the database products
- The common database 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 user password
- The connection with the server is established with the help of `mysql_connect()` function
- The basic PHP functions that are used with respect to the database are: `mysql_list_dbs()`, `mysql_select_db()`, `mysql_list_tables()`, and `mysql_num_rows()`
- The `mysql_close()` function closes the connection with the MySQL server
- The data access functions used in PHP are: `mysql_query()`, `mysql_fetch_array()`, `mysql_fetch_row()`, `mysql_fetch_field()`, `mysql_field_len()`, and `mysql_num_fields()`

18.3 Post-Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to handling working with cookies.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 19 – Working with Cookies

19.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session.. Prepare a question or two that will be a key point to relate the current session objectives.

19.1.1 Objectives

At the end of this session, the learners 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

19.1.2 Teaching Skills

To teach this session, you should be well-versed with the concepts related to cookies in PHP. You should be able to explain the process of setting a cookie, retrieving a cookie and to delete a cookie. You should be able to describe the drawbacks associated with cookies.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 2 of 29

Tell the students that they will be introduced to cookies in PHP. They will learn the process of setting up a cookie, retrieving a cookie, and deleting a cookie. They will also be taught the drawbacks associated with cookies.

19.2 In-Class Explanations

Slide 3

Let us introduce cookies in PHP.

Introduction

- ◆ Web sites store user information in databases to maintain a track of their visits
- ◆ Cookies enable Web sites to store user information
- ◆ PHP supports Hyper Text Transfer Protocol (HTTP) cookies

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 3 of 29

Using slide 3, introduce the concept of cookies in PHP. Tell students that the Web sites store user information to maintain a track of their visits. Then, explain them about cookies in brief. Tell them that a cookie is a small text file stored in a temporary folder on the local system of the user. The cookie is a file in which a Web site can store different information. Most cookies delete themselves after a predetermined period. It is also possible for a user to identify and delete cookies on his/her computer. There are also cookies that remain persistent on the system. They are called persistent cookies.

Give the students a real life example to illustrate the concept of a cookie. For example, if you visit an online shopping site such as Amazon, Apple Online Store, or eBay, it will store your login details in cookies on your machine for the duration of your online session. Cookies are a safe way to personalize your online shopping experience. Cookies can store information that identifies you, such as your name and email address. These details will be used when the checkout process begins and the user information can be retrieved easily from the cookie.

Most browsers, such as Microsoft Internet Explorer, Mozilla Firefox, and Google Chrome, can be configured to let the user choose whether he/she will accept a cookie.

Using slide 3, also explain students about Hyper Text Transfer Protocol (HTTP).

Additional Information:

For more information on cookies in PHP, visit the following links:

http://www.tutorialspoint.com/php/php_cookies.htm

<http://www.tizag.com/phpT/phpcookies.php>

http://php.about.com/od/advancedphp/qt/php_cookie.htm

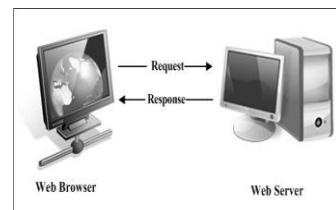
<http://www.phpnerds.com/article/using-cookies-in-php>

Slides 4 and 5

Let us understand Web pages.

Web Pages 1-2

- ◆ Uses HTTP protocol for sending information to the server
 - ◆ HTTP is a stateless protocol, because the execution of the current command is completed without the knowledge of commands that came before it
- ◆ Are of two types:
 - ◆ Static Web pages
 - ◆ Dynamic Web pages
- ◆ Static Web pages
 - ◆ Web browser requests for a page and the server completes the request by sending the required file
 - ◆ Does not involve any interaction with the user



The diagram illustrates the interaction between a Web Browser and a Web Server. A computer monitor representing the Web Browser is on the left, and a server tower representing the Web Server is on the right. Two arrows connect them: one pointing from the browser to the server labeled 'Request', and another pointing from the server back to the browser labeled 'Response'.

Version 1.0 © Aptech Limited.

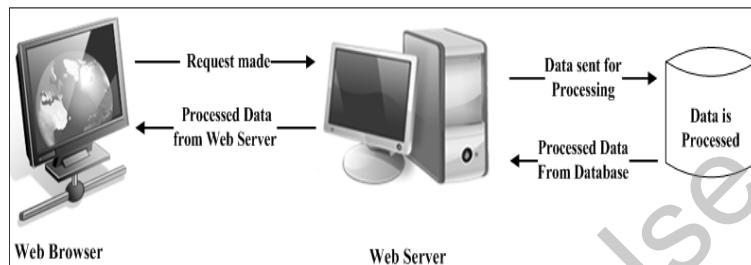
Working with Cookies / Session 16 / Slide 4 of 29

Web Pages

2-2

◆ Dynamic Web pages

- ❖ Require user interaction, so scripting languages such as JavaScript, PHP, and ASP are used
- ❖ Accept information from the user and record it for further processing



Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 5 of 29

Use slides 4 and 5 to explain the students about Web pages in detail.

Web pages can be either static or dynamic. Tell them that static pages remains constant while the dynamic means changing. It is to be noted that the static HTML pages are static Web pages. The content of the Web page changes only when the Web developer updates and publishes the file.

You can tell the students that they can determine if a page is static or dynamic simply by looking at the page's file extension in the URL, located in the address field of the Web browser. If it is '.htm' or '.html', the page is static. If the extension is '.php', '.asp', or '.jsp', the page is most likely dynamic.

Slides 6 and 7

Let us introduce cookies.

Introducing a Cookie

1-2

- ◆ Data stored by Web sites are as follows:
 - ❖ 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
- ◆ Types of cookies 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

Version 1.0 © Aptech Limited. Working with Cookies / Session 16 / Slide 6 of 29

Introducing a Cookie

2-2

- ◆ 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
 - ❖ 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

Version 1.0 © Aptech Limited. Working with Cookies / Session 16 / Slide 7 of 29

Slides 6 and 7 describe about cookies in detail. One important consideration when using cookies in a script is the timing. If there is a need to send a cookie to the user's system, then it should be sent before sending anything else (before any part of the page itself is sent, even before a blank line). When cookies are created, they are set by default to be deleted when the user closes their browser. Using slide 7, explain the uses of cookies.

In-Class Question:

After you finish explaining about cookies in PHP, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you describe a few characteristics of static Web pages?

Answer:

- In a static Web page, Web browser requests for a page and the server completes the request by sending the required file
- It does not involve any interaction with the user

Slides 8 to 11

Let us understand the working of a cookie.

Working of a Cookie

1-4

- 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
- Browser stores the cookie and sends it back to the Web site in subsequent requests
- Life of a cookie depends on the expiration time and date
- Cookie is stored on the hard disk of the user's computer which enables the Web site to keep a track on the user visiting the Web site
- Web servers and Web browsers send cookies to each other in HTTP headers
- Web server sends the cookie to the browser in the `setcookie` header field which is part of the HTTP response
- Web browser stores the cookie and uses the same in subsequent requests to the same Web server

Version 1.0 © Aptech Limited

Working with Cookies / Session 16 / Slide 8 of 29

Working of a Cookie

2-4

- ◆ Consider the following HTTP response header:

Snippet

```
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
```

Working of a Cookie

3-4

- ◆ 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

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 10 of 29

Working of a Cookie

4-4

- ◆ Consider the following HTTP response header:

Snippet

```
GET /usa/florida.php HTTP/2.0
Connection: Keep-Alive
Cookie: city=east-coast-usa
Host: www.Webworldmaps.com
Referer: http://www.Webworldmaps.com/
```

Cookie can be defined using the `setcookie` function.

Code displays a subsequent request that the Web browser sends to the Web server.

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 11 of 29

Slide 8 describes about working of a cookie. When the browser connects for the first time, there will be no cookies in it. A call to the `setcookie()` function is made when the request to the script is sent. The set-cookie header value is stored as a local cookie, when the response is received by the browser. A cookie header containing name and value is included to the browser, when multiple requests are found. Explain working of cookie, in brief using slides 9 to 11.

Slides 12 to 17

Let us understand setting a cookie.

Setting a Cookie

1-6

- ◆ Setting a cookie is sending the cookie to the browser
- ◆ PHP uses two functions, `setcookie()` and `setrawcookie()` to set a cookie
- ◆ `setrawcookie()` function sends a cookie without encoding the cookie value
- ◆ `setcookie()` function generates the cookie header field that is sent along with the rest of the header information

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 12 of 29

Setting a Cookie

2-6

- The `setcookie()` function is as follows:

Syntax

```
setcookie(name, value, expiry date, path, domain, secure)
```

Where,

name - defines the name of the cookie

value - defines the value of the cookie that is stored on the client system

expiry date - defines the date and time (UNIX timestamp) when the cookie will expire

path - defines the location on the server where the cookie will be stored.

domain - defines the domain name where the cookie is made available

secure - defines the type of HTTP connection that the cookies will pass through

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 13 of 29

Setting a Cookie

3-6

- 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
- Cookies are a part of the HTTP header and 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`
- If the `setcookie()` function runs successfully, the function returns `True`

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 14 of 29

Setting a Cookie

4-6

- Setting a cookie that expires in one day in a Web site that displays country maps when a user enters a country name in the search feature of the Web site are as follows:

Snippet

```
$mapname = $_GET['fmapname'];
setcookie("mycookie", $mapname, time() + 86400,
"/Webmap/", ".Webworldmaps.com");
```

Setting a Cookie

5-6

- 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

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 16 of 29

Setting a Cookie

6-6

- Creating a cookie that expires when the Web browser window is closed are as follows:

Snippet

```
$val = $_GET['uname'];
setcookie("uname",$val);
```

In 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`.

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 17 of 29

Using slide 12 to 14, explain students the procedure for setting up a cookie. The function `setcookie()` is used to set a value and the optional expiration date of a cookie. Explain them the meaning of each parameter and explain whether it is required or optional. Use slides 15to 17 to explain about the `setcookie()` function with the given example.

Slides 18 to 23

Let us understand retrieving cookies in PHP.

Retrieving Cookies in PHP

1-6

- Cookies are useful only when the Web server can retrieve the information from it
- 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
- Document that created the cookie as well as that are present in the same directory can access it
- Documents outside the directory need to include the path or the domain name of the cookie to access the cookie

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 18 of 29

Retrieving Cookies in PHP

 2-6

- ◆ PHP provides three ways of retrieving a cookie value and they are as follows:

- ◆ Passing a variable as the cookie name - retrieve the cookie value, use the variable as the cookie name. The following code snippet displays a cookie value:

Snippet

```
echo $cookie_name;
```

- ◆ This method of retrieving the cookie value is not recommended as PHP will start searching all the variables present in the client system
- ◆ The `register_globals` option must be enabled in the configuration file

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 19 of 29

Retrieving Cookies in PHP

 3-6

- ◆ 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
 - ◆ An example of this is shown as follows:

Snippet

```
echo $_COOKIE[$cookie_name];
```

- ◆ This is more reliable and faster than retrieving the cookie value through a variable

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 20 of 29

Retrieving Cookies in PHP

4-6

Snippet

```
<?php  
  
$cookieval = $_COOKIE ['uname']; ?>  
  
<HTML>  
<BODY>  
  
<?php  
  
if (isset($cookieval))  
  
{  
  
echo "Welcome $cookieval";  
}
```

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 21 of 29

Retrieving Cookies in PHP

5-6

- ◆ **retrieve_cookie.php** - Retrieving a cookie value using the `$_COOKIE` global variable

Snippet

```
else  
{  
echo "You need to log in";  
}  
?>  
</BODY>  
</HTML>
```

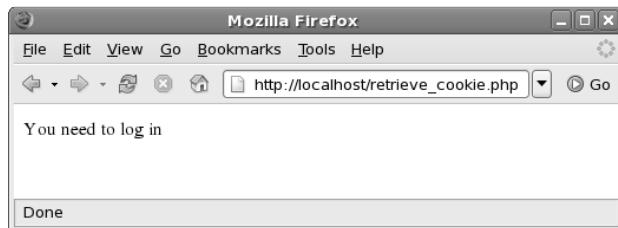
Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 22 of 29

Retrieving Cookies in PHP

6-6

The output of the script is as follows:



`$cookieeval` stores the cookie value.

The `isset()` function checks whether the cookie is set

Version 1.0 © Aptech Limited.

Working with Cookies / Session 16 / Slide 23 of 29

Slide 18 describes about retrieving cookies in PHP. Slides 19 and 20 consist of a code snippet that describes three ways of retrieving a value in PHP. After the cookie is set, the information needs to be retrieved. The name of each cookie sent by the server is accessed with the superglobal array `$_COOKIE`. In slides 21 and 22, an example code is given that describes the process of retrieving a cookie. Explain the students each line of code and show them the corresponding output that is displayed in slide 23.

Additional Information:

For more information on retrieving value from a cookie in PHP, visit the following links:

<http://php.net/manual/en/function.setcookie.php>

<http://Webcheatsheet.com/php/cookies.php>

<http://www.sitepoint.com/baking-cookies-in-php/>

Slides 24 and 25

Let us learn about deleting a cookie.

Deleting a Cookie 1-2

- ◆ 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
- ◆ To delete a cookie with a date in the past, enter the code as shown in the Snippet in a PHP script

Snippet

```
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

Version 1.0 © Aptech Limited. Working with Cookies / Session 16 / Slide 24 of 29

Deleting a Cookie 2-2

- ◆ This process is called as deconstructing the variable
- ◆ Use the following syntax to delete a cookie through deconstruction:

Snippet

```
setcookie($cookie_name);
```

- ◆ To delete the cookie named `uname`

Snippet

```
setcookie($uname);
```

Version 1.0 © Aptech Limited. Working with Cookies / Session 16 / Slide 25 of 29

Explain the students the process of deleting a cookie using slides 24 and 25. By default, the cookies will be deleted when the browser is closed. That default setting can be overridden by setting a time for the cookie's expiration. However, there may be occasions when there is a need to delete a cookie before the user closes his or her browser, and before its expiration time arrives. To do so, you must set an expiration date that is in the past.

Slides 26 and 27

Let us understand the problems with cookies.

Problems with Cookies  1-2

- ◆ 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
- ◆ Following are some of the drawbacks of cookies:
 - ❖ Cookies cannot contain more than a certain amount of information
 - ❖ Only a maximum of 29 cookies of a domain can be maintained
 - ❖ A browser can maintain maximum of 300 cookies
 - ❖ Storing large number of cookie files slows down the system
 - ❖ Some users disable cookies while accessing Web sites as a result Web sites that depend on cookies lose information of such users

Version 1.0 © Aptech Limited. Working with Cookies / Session 16 / Slide 26 of 29

Problems with Cookies  2-2

- ◆ 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 can contain large amount of information and retrieving larger amount of information on each page requires repetitive coding across the pages

Version 1.0 © Aptech Limited. Working with Cookies / Session 16 / Slide 27 of 29

Using slide 26 and 27, explain the students the problems with cookies. Tell the students that as explained in earlier slides, cookies are useful and beneficial in many ways, however, they do have some drawbacks. Discuss the points given in slides 26 and 27.

In-Class Question:

After you finish with explaining about the problems associated with cookies, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What are the two ways to delete a cookie?

Answer:

The two ways to delete a cookie 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

Slides 28 and 29

Let us summarize the session.

The screenshot shows a presentation slide with a blue header bar. On the left of the bar is the word "Summary". On the right is a small green circular icon with a white keyhole-like shape in the center, followed by the text "1-2". The main content area contains a bulleted list of eight points about cookies. At the bottom of the slide, there is a thin blue footer bar with the text "Version 1.0 © Aptech Limited." on the left and "Working with Cookies / Session 16 / Slide 28 of 29" on the right.

- ◆ 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 records 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

Summary

2-2

- ◆ 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 by 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 maximum of 300 cookies

Use slides 28 and 29 to list and explain the summary of this session.

The summary points are as follows:

- 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 records 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 by 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 maximum of 300 cookies

19.3 Post-Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to session management in PHP.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

For Aptech Center Use Only

Session 21 – Session Management in PHP

21.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

21.1.1 Objectives

At the end of this session, the learners 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

21.1.2 Teaching Skills

To teach this session, you should be well-versed with the concepts related to sessions in PHP. You should be able to explain the procedure to work with a session. You should be able to describe the methods to start, register, and end a session. You should be able to point out the use of the php.ini file.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *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*

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 2 of 42

Tell students that they will be introduced to sessions in PHP along with the procedure to work with a session. They will also understand the various methods used to start, register, and end a session. In addition, they will also learn the use of the php.ini file.

21.2 In-Class Explanations

Slides 3 to 5

Let us introduce the concept of sessions in PHP.

Sessions

1-3

- ◆ Sessions are similar to cookies and enable the functionality of storing temporary user information
- ◆ The difference between cookies and sessions is as follows:
 - ◆ Pertinent cookies store information on the local computer
 - ◆ Session enables PHP to store information on the Web server

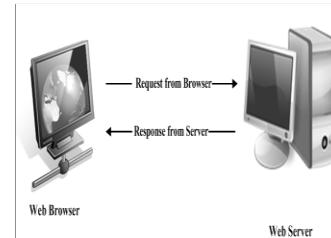
Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 3 of 42

Sessions

2-3

- ◆ Web browsers and Web servers have a stateless interaction and do not maintain track of user sessions
- ◆ HTTP protocol
 - ❖ Enables Web browsers to communicate with Web servers
 - ❖ Has no methods or functions to maintain the status of a particular user



Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 4 of 42

Sessions

3-3

- ◆ Web sites that cannot depend on HTTP or Web servers for complex user interaction need session tracking
- ◆ Refers to the total time the user accesses information on a particular Web site before exiting the Web site
- ◆ Manages data for a particular user in a specific session
- ◆ Enable distinguishing user specific information for the entire duration of the session

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 5 of 42

Using slides 3 to 5, introduce the concept of sessions in PHP. Tell the students that sessions allow a PHP script to store data on a Web server for later use. This use can even include requests to different PHP pages. A session ends when the user closes the browser, or when the server deletes the session information, or when the developer explicitly destroys the session. Sessions are useful to protect the data that cannot be written or read easily, such as in cookies.

The PHP session support provides a way to preserve data across consequent accessing. A session id is a unique id that is assigned to the visitors accessing the Web site. The session id is stored in a

cookie or propagated in the URL. Using slide 3, state the differences between session and cookie. Through slides 4 and 5, elaborate more on sessions.

Additional Information:

For more information on sessions in PHP, visit the following links:

<http://php.net/manual/en/intro.session.php>

<http://php.net/manual/en/reserved.variables.session.php>

Slides 6 to 12

Let us understand importance of a session.

The slide has a blue header bar with the title 'Importance of a Session' on the left and a green circular icon with a white keyhole symbol on the right. To the right of the icon is the text '1-7'. Below the header is a white content area containing a bulleted list of points. At the bottom of the slide is a thin blue footer bar with small text.

Importance of a Session

1-7

- ◆ 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:

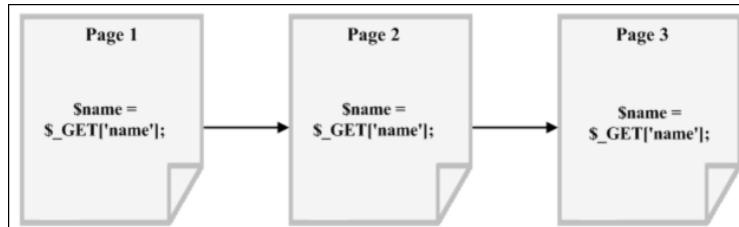
```
$name = $_GET['name'];
```
- ◆ In the code, the variable \$name stores the value that the script retrieves from the HTML form
- ◆ This process of transferring user information is time consuming and not essential for large Web sites

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 6 of 42

Importance of a Session

2-7

- The code to be used across all the Web pages of the Web site is as follows:



Version 1.0 © Aptech Limited.

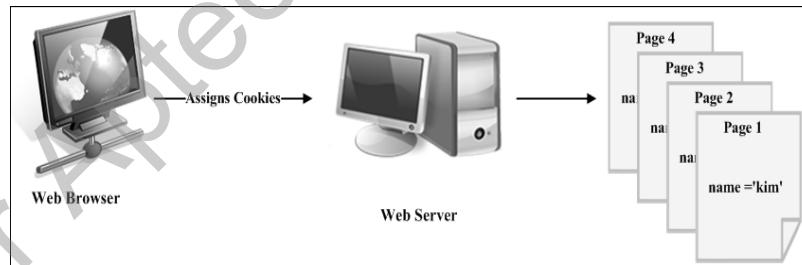
Session Management in PHP / Session 18 / 7 of 42

Importance of a Session

3-7

- Consider a scenario, where it is required 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 displays the assignment of cookies by the Web server to the browser



Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 8 of 42

Importance of a Session

4-7

- ◆ 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

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 9 of 42

Importance of a Session

5-7

◆ 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 and 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 as a result the performance of computers reduces
- ◆ To improve the performance of such computers, users disable cookies making the assignment of cookies pointless

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 10 of 42

Importance of a Session

6-7

- ◆ Sessions play an important role in such situations
- ◆ 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

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 11 of 42

Importance of a Session

7-7

Table 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

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 12 of 42

Begin the explanation by first illustrating an example scenario and the various approaches (such as GET/POST methods, cookies, and sessions) that can be used to solve the scenario.

Using slides 6 to 8, explain the students the given example and tell them the first approach of solving this - through GET and POST methods. Show them the image in slide 7, which displays the code to be given. Slide 8 shows the scenario that explains the working architecture. Tell the students the disadvantages of using the GET and POST methods in case of large number of users and pages. Bring out the need for cookies here.

Using slides 9 and 10, explain them the process of storing and retrieving the sessions. The two attributes that are to be considered are the size of the cookies and cookie disabled, as described in slide 10. Now, tell the students the drawbacks and risks of using cookies. Explain that it is here, that sessions play an important role. When the need is such that you cannot use GET/POST methods or cookies, it is sessions that prove most useful.

To explain the difference between cookies and sessions, use the tabulated information given in slide 12.

Slides 13 to 16

Let us understand working with sessions.

Working with Sessions

1-4

- ◆ 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
- ◆ 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

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 13 of 42

Working with Sessions

2-4

◆ Cookies disabled:

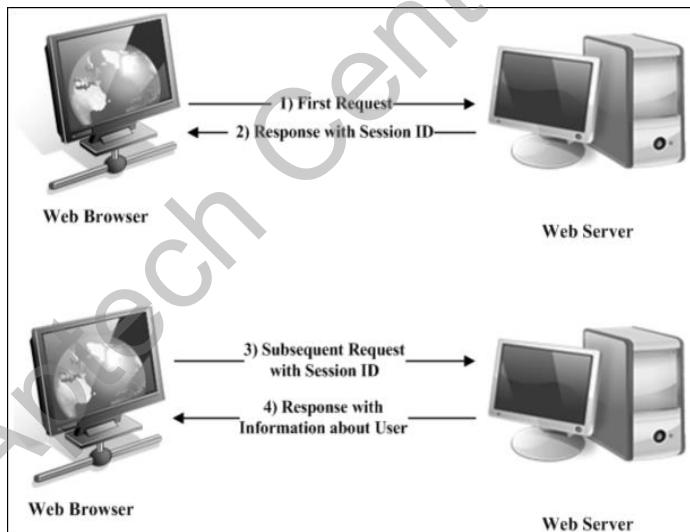
- ◆ Web server allots a session ID to the browser using the Uniform Resource Locator (URL)
- ◆ The URL transfers user information from the browser to the server
- ◆ PHP stores the session variables in a file and names the file based on the session ID
- ◆ While using a session, PHP stores all the user information in a file on the Web server
- ◆ The file includes a session ID that is related to the user's session variable
- ◆ Each session ID identifies a different user and relates to a file that belongs to that user
- ◆ PHP destroys the session file once the user exits the Web site

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 14 of 42

Working with Sessions

3-4



Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 15 of 42

Working with Sessions

4-4

- ◆ PHP works with sessions in the following sequence:
 - ❖ User accesses a session-enabled Web site which checks the user identity
 - ❖ 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 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
 - ❖ The session file is destroyed from the Web server, when the user exists from the Web site

Using the points given in slides 13 and 14, explain the students about working with sessions. It is to be noted that there are two situations upon which a session id depends. They are: a) when cookies are enabled and b) when cookies are disabled. Using the image given in slide 15, explain the process carried out between the Web browser and the Web server. Use slide 16 to explain the sequence in which the PHP works with sessions. Explain them about the stored session id and the Web browser record that holds the session ID.

In order to make session information available across pages, PHP performs the following tasks:

- **PHP assigns a session ID number**
The number is a long random number that is unique for the user. The session ID is stored in a PHP system variable, `PHPSESSID`.
- **PHP stores the variables that you want to save for the session, in a file on the server**
The file is given a name with the session ID number. It is stored in a directory specified by 'session.save_path' in the `php.ini` file. The session directory must be existing prior to saving.
- **PHP passes the session ID number to every page**
If the user has enabled cookies, PHP passes the session ID in cookies. If the user has disabled cookies, PHP behavior depends on whether 'trans-sid' is turned on in the `php.ini` file.
- **Finally, PHP retrieves the variables from the session file for each new session page**

Whenever a user opens a new page that is a part of the session, PHP retrieves the variables available in the `$_SESSION` array from the file, using the session ID number that was passed from the previous page.

Slide 17

Let us see the life cycle of a session.



Life cycle of a Session

- ◆ There are three stages in the life cycle 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

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 17 of 42

Slide 17 describes the life cycle of a session. Tell them there are three stages in the life cycle: starting the session, registering the session variable, and ending the session. These stages describe the communication between the Web browser and Web server.

In-Class Question:

After you finish explaining about life cycle of sessions in PHP, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you explain the difference between cookies and sessions?

Answer:

- Pertinent cookies store information on the local computer.
- Session enables PHP to store information on the Web server.

Slides 18 to 21

Let us learn about the process of starting a session.

Starting the Session 1-4

- ◆ A session starts when a user logs on to the Web site
- ◆ The `session_start()` function enables to start a session
- ◆ The process of starting a session is also called as initializing a session
- ◆ 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

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 18 of 42

Starting the Session 2-4

- ◆ 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
- ◆ The Web server passes the session ID as a response to the browser
- ◆ 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
- ◆ If the session is valid and existing, it activates the frozen variables of the session
- ◆ If the session is invalid or non existing, it creates a session ID for the new session

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 19 of 42

Starting the Session

3-4

- The `session_start()` function is as follows:

Syntax

```
session_start();
```

- To start a session, perform the following steps:

- Open a new file in **gedit** text editor

- Enter the following code:

Snippet

```
<?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>";
?>
```

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 20 of 42

Starting the Session

4-4

- Save the script as `session1.php` in the `/usr/local/apache2/htdocs/` directory.
- Open the Mozilla Firefox Web browser.
- Enter `http://localhost/session1.php` in the Address bar and press Enter.

The following output is displayed:



Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 21 of 42

Slides 18 and 19 describe how to start the session. Explain the students about the `session_start()` function and how a session is created. Explain that this function first checks for an existing session ID number. If it finds one, it sets up the session variables; otherwise, it starts a new session by creating a new session ID number. `session_start` is subject to the same limitation as cookies. This means that, to avoid an error, the `session_start` function must be called before any output is sent.

Tell students that they can indicate to PHP that every page on a site should automatically start with a `session_start` statement. They can do this with a setting in the configuration file `php.ini`.

They should search for the variable session.auto_start, set its value to 1, and restart the Web server for this setting to take effect. With auto_start turned on, it is not required to add a session_start at the beginning of each page.

To give a brief explanation about the syntax and the code for implementing the session_start() function, use the code snippet given in slide 20. The output for the corresponding code is explained in slide 21.

Additional Example:

You may also use the following additional example to explain to the students:

```
<?php
/* Program name: login.php
* Description: Helps user to log in.
*/
session_start();
if(isset($_POST['sent']) && $_POST['sent'] == "yes")
{
/* check each field for blank fields */
foreach($_POST as $field => $value)
{
if($value == "")
{
$blank_array[$field] = $value;
}
else
{
$good_data[$field]=strip_tags(trim($value));
}
} // end of foreach loop for $_POST
if(@sizeof($blank_array) > 0) // blank fields found
{
$message = "<p style='color: red; margin-bottom: 0;
font-weight: bold'>
You must enter both a user id and a password.</p>";
/* redisplay form */
extract($blank_array);
extract($good_data);
include("form_log.inc");
exit();
} // end if blanks found
include("dbstuff.inc");
$cxn = mysqli_connect($host,$user,$password,$database)
or die ("could not connect to server");
$query = "SELECT first_name FROM Customer
WHERE user_name='".$_POST[user_name]'
AND password=md5('$_POST[password']");
$result = mysqli_query($cxn,$query)
or die ("Could not execute query.");
$n_row = mysqli_num_rows($result);
if($n_row < 1) // if login unsuccessful
{
$message = "<p style='color: red; margin-bottom: 0;
font-weight: bold'>
```

```
User ID and Password not found.</p>";
extract($_POST);
include("form_log.inc");
exit();
}
else //if login successful
{
$row = mysqli_fetch_assoc($result);
$_SESSION['first_name'] = $row['first_name'];
$_SESSION['auth'] = "yes"; →52
header("Location: secret_page_session.php"); →53
}
} // end if submitted
else // first time script is run
{
$user_name = "";
$password = "";
include("form_log.inc");
}
?>
```

The code is briefly explained as follows:

The 'if' statement checks for the hidden field. If the hidden field exists, the form was submitted by a user. If the hidden field does not exist, the script is running for the first time, not started by a form submitted by a user, and the script jumps to the else statement given much later. The foreach statement loops through the \$_POST array. Both the user_name and password fields from the form are present in the array. The foreach statement checks whether a field is blank. If it is blank, its value is added to \$blank_array; if it is not blank, its value is cleaned and added to \$good_data. Then, there is an 'if' statement that executes if any blank fields are found. An error message is created, the form is redisplayed with the information that the user entered, and the script exits. If no blank fields are found, the 'if' statement does not execute, and the script continues.

Then, the code searches the database for the user_name and password that the user typed in the form. It stores the number of matches found in \$n_rows. Then, the code for if statement is given that executes if the login is unsuccessful. The 'if' block creates an error message and redisperses the form. The 'else' statement executes if the login is successful. A new page is downloaded, taking the user to the first page of the Web site. The user's first name is added to the end of the URL, to pass the name to the next Web page. The first_name information added to the URL has the format first_name=\$row[first_name].

The else statement that executes the first time the script is run, before the user submits the form. The blank form is displayed. Without an exit statement, the script might continue on to the next statements after the form is displayed. When the user successfully logs in, the user continues to the first Web page in the restricted site.

The script that retrieves information from the URL is given as follows:

```
<?php
/* Program name: secret_page_url.php
 * Description: Displays a welcome page.
 */
```

```

?>
<html>
<head><title>Secret Page with GET</title></head>
<body>
<?php
echo "<p style='text-align: center; margin: .5in'>
Hello, {"$_GET['first_name']}<br />
Welcome to the secret page</p>";
?>
</body></html>

```

This script displays a Web page to the user who logs in. The message is personalized by adding the first name of the user, which was passed in the URL.

Slides 22 to 26

Let us understand the process of registering the session variable.

Registering the Session Variable

1-5

- ◆ Variables in a session file contain user specific information
- ◆ Session library enables creation, serialization, and storage of session data
- ◆ There are three methods to set a session variable, which 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

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 22 of 42

Registering the Session Variable

2-5

- ◆ Steps to register the value of a session variable are as follows:

1. Open a new file in **gedit** text editor
2. Enter the following code:

Snippet

```
<?php
session_start();
$_SESSION['myname'] = "Jessica";
?>
<HTML>
<HEAD> <TITLE> Session </TITLE></HEAD>
<BODY>
<A HREF = "mypage.php"> Homepage of MyPage.com </A>
</BODY>
</HTML>
```

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 23 of 42

Registering the Session Variable

3-5

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 following code:

Snippet

```
<?php
session_start();
$myname = $_SESSION['myname'];
?>
<HTML>
<HEAD> <TITLE> Homepage </TITLE></HEAD>
<BODY>
Welcome <?php echo $myname ?> to MyPage.com <br>
</BODY>
</HTML>
```

Version 1.0 © Aptech Limited.

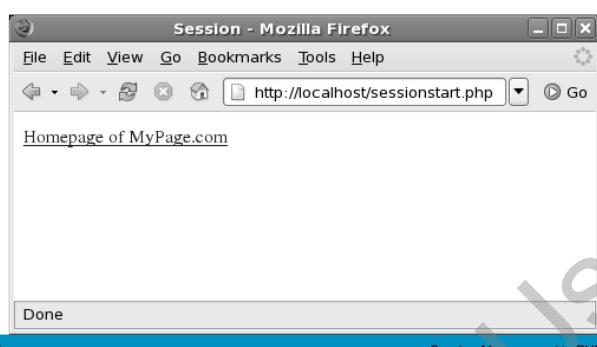
Session Management in PHP / Session 18 / 24 of 42

Registering the Session Variable

4-5

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

The following output is displayed:



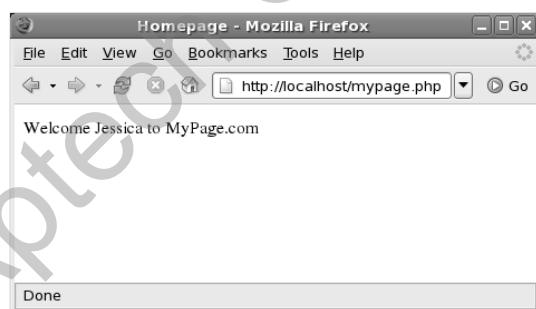
Registering the Session Variable

5-5

6. Click **Homepage of MyPage.com**

The Homepage page appears with the message,

Welcome Jessica to MyPage.com



Using slide 22, explain the students about variables in a session file. Explain the three methods to set the session variable. Explain the serializing concepts with examples. Using slides 23 to 25 describe the steps to register the value of a session variable. Each line of code given in slide 23 has its own part in working with session variables. Explain the students each line of code briefly. In the given example 'myname' is a variable declared within the `$_SESSION()` method. This variable has the reference for the session 'mypage.com'. The output for the corresponding code is given in slide 26.

Slides 27 to 32

Let us understand the process of ending a session.

Ending the Session

1-6

- ◆ When the user logs out of the Web site, the PHP script executes the `session_destroy()` function
- ◆ When the session file is deleted, the `$PHPSESSID` cookie is not removed from the Web browser
- ◆ The syntax for the `session_destroy()` function is as follows:

Syntax

```
session_destroy();
```

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 27 of 42

Ending the Session

2-6

- ◆ PHP uses the following configuration directives when a session ends:
 - ◆ `gc_maxlifetime()`:
 - ◆ Enables PHP to determine the time to wait before ending a session and the process of cleaning up is called as garbage collection
 - ◆ `gc_probability()`:
 - ◆ Enables PHP to determine with what probability the garbage collection routine must be invoked

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 28 of 42

Ending the Session

3-6

- To destroy a session, perform the following steps:

- Open a new file in **gedit** text editor

- Enter the following code:

Snippet

```
<?php
session_start();
$myname = $_SESSION['myname'];
// The session_unset() function unregisters a session
// variable.
session_unset();
session_destroy();
echo "Session destroyed!";
```

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 29 of 42

Ending the Session

4-6

```
?>
<HTML>
<HEAD> <TITLE> Session </TITLE></HEAD>
<BODY>
<br>
<A HREF = "mypage.php"> Homepage of MyPage.com </A>
</BODY>
</HTML>
```

- Save the file as **sessiondestroy.php** in the `/usr/local/apache2/htdocs/` directory
- Open the Mozilla Firefox Web browser

Version 1.0 © Aptech Limited.

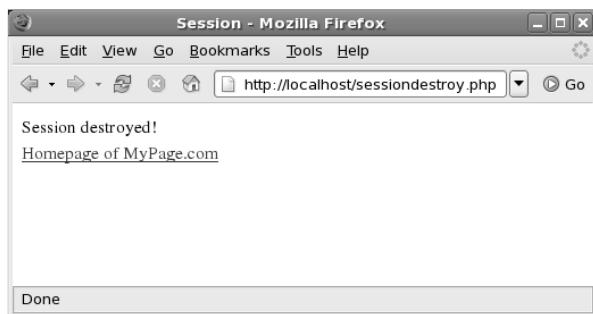
Session Management in PHP / Session 18 / 30 of 42

Ending the Session

5-6

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

The following output is displayed:

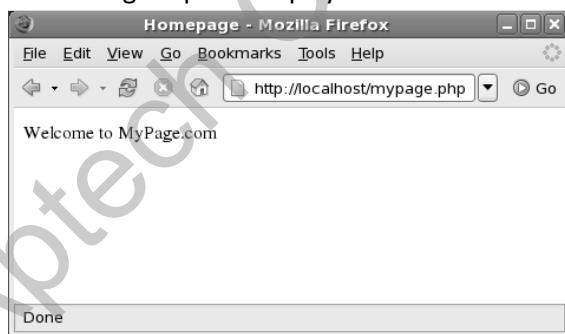


Ending the Session

6-6

6. To view the execution of the `session_destroy()` function, click the [Homepage of MyPage.com](#) hyperlink

The following output is displayed:



The name **Jessica** does not appear in the message because the variable has been cleared.

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 32 of 42

The `session_destroy()` function is used to end the session whenever the user logs out of the Web site. Use the points given in slide 27, to explain the students about the `session_destroy()` function. Explain the students about `gc_maxlifetime()` and `gc_probability()` functions and tell them how they work using slide 28. Slides 29 to 32 describe the steps for destroying the session. The `session_destroy()` function defined in the code, is used to delete the variable that is created.

Slides 33 to 41

Let us understand working with the `php.ini` file.

Working with the `php.ini` File

1-9

- ◆ A Web server can contain multiple `php.ini` files
- ◆ 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

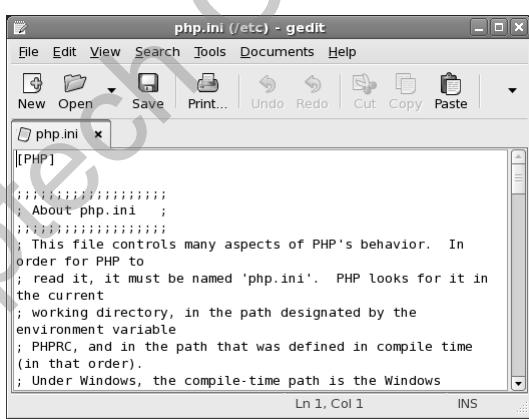
Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 33 of 42

Working with the `php.ini` File

2-9

- ◆ Figure displays the PHP configuration file



The `php.ini` file contains directive listed in the directive = value format.
You can use a semicolon to add a comment to the file.

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 34 of 42

Working with the php.ini File

3-9

Table lists the categories 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
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
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
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

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 35 of 42

Working with the php.ini File

4-9

Categories	Options
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
Magic Quotes	Set magic quotes for incoming Get, Post, Cookie data
	Use Sybase style magic quotes
	Automatically adds file before or after any PHP document
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

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 36 of 42

Working with the php.ini File

5-9

- ◆ Session category of the `php.ini` file include options, which are as follows:
 - ❖ `session.save_handler` - specifies how PHP stores and retrieves session variables
 - ❖ Either of the following values can be used for this option:
 - ❖ `files`: indicates the use of the session files
 - ❖ `mm`: stores and retrieves data from a shared memory
 - ❖ `user`: stores and retrieves variables with custom defined handlers

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 37 of 42

Working with the php.ini File

6-9

- ❖ `session.save_path` - specifies the name of the directory where the session files will be stored
- ❖ `session.use_cookies` - indicates whether PHP must send a session ID to the Web browser through a cookie. 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 and the value is specified in seconds

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 38 of 42

Working with the php.ini File

7-9

- ❖ `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
- ❖ `session.cookie_secure` - specifies whether the cookies must be sent over secured connections. By default, the cookies are not sent through secured connections

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 39 of 42

Working with the php.ini File

8-9

- ❖ Other settings can also be modified in the `php.ini` file, such as:

- ❖ `register_globals` - controls the functioning of server, forms, and environment variables

If this option is disabled, variables can be retrieved using the GET or POST methods as follows:

```
$_POST['$variable_name'];
$_GET['$variable_name'];
```

If `register_globals` is enabled, the variable can be directly accessed using the variable name as follows:

```
$storeValue = $variable_name;
```

`$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`

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 40 of 42

Working with the php.ini File

9-9

- ❖ `upload_tmp_dir` - sets the location of the temporary file that is uploaded with the HTML form
- ❖ `display_errors` and `display_startup_errors` - enables PHP to display errors on the Web browser
- ❖ `log_errors` and `error_log` - enables PHP to display error logs

Version 1.0 © Aptech Limited.

Session Management in PHP / Session 18 / 41 of 42

Slide 33 describes briefly about the php.ini file. Show them the PHP configuration file with the editor window, which is displayed in slide 34. The categories of php.ini file is tabulated and shown in slides 35 and 36. Each of the categories consists of several options. The session category of the php.ini file can be listed as `session.save_handler`, `session.save_path`, `session.use_cookies`, `session.use_only_cookies`, `session.cookie_lifetime`, `session.name`, `session.auto_start`, and `session.cookie_secure`. Describe each category and tell them its uses, using slides 37 to 39. The other settings that can be modified is given in slides 40 and 41.

In-Class Question:

After you finish with explaining about the php.ini file, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you describe briefly about the php.ini file?

Answer:

- A Web server can contain multiple php.ini files
- 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

Additional Information:

For more information on retrieving value from a cookie in PHP, visit the following links:

http://www.tutorialspoint.com/php/php_sessions.htm

<http://php.net/manual/en/ref.session.php>

<http://php.net/manual/en/features.sessions.php>

<http://www.tizag.com/phpT/phpsessions.php>

Slide 42

Let us summarize the session.

The screenshot shows a slide with a blue header bar. The header bar contains the word "Summary" on the left and a green circular logo with a white icon on the right. Below the header is a white content area with a bulleted list of seven points. At the bottom of the slide is a blue footer bar with small text. A large, semi-transparent watermark reading "For Aptech Center Use Only" is overlaid across the slide content.

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 life cycle of a session are: starting a session, registering a session variable, and ending a session

Version 1.0 © Aptech Limited. Session Management in PHP / Session 18 / 42 of 42

Use slide 42 to list and explain the summary of this session.

The summary points are as follows:

- 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 life cycle of a session are: starting a session, registering a session variable, and ending a session

21.3 Post-Class Activities for Faculty

You should familiarize yourself with the topics of the next session. You should know about the topics related to handling e-mail with PHP.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

For Aptech Center Use Only

Session 22 – Handling E-mail with PHP

22.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two which will be a key point to relate the current session objectives.

22.1.1 Objectives

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

- Describe the process of sending an e-mail using PHP
- Explain the process of attaching files with an e-mail using PHP

22.1.2 Teaching Skills

To teach this session, you should be well-versed with the process of sending an e-mail using PHP. You should also be able to explain the process of attaching files with an e-mail using PHP.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *Describe the process of sending an e-mail using PHP*
- ◆ *Explain the process of attaching files with an e-mail using PHP*

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 2 of 26

Tell the students that in this session they will learn the process of sending an e-mail using PHP. They will learn about the mail() function. They will also learn the process of attaching files with an e-mail using PHP.

22.2 In-Class Explanations

Slide 3

Let us introduce the concept of e-mail in PHP.

Introduction

- ◆ E-mail are:
 - ◆ 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

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 3 of 26

Using slide 3, give a brief introduction to e-mail in PHP.

The e-mail acts as a communication gateway to people across the world. Consider a scenario wherein Martina Jones having e-mail address `martinajones@aptech.co.au` from Australia sends an e-mail to Kim Smith (`ksmith@aptech.co.uk`) who is present in Britain. This is done with the help of an e-mail client program. Today there are several popular e-mail clients used all over the world. Technologies such as .NET and Java also provide support for incorporating e-mail functionality in applications. Likewise, PHP also provides support to send and receive e-mails from within PHP applications.

Slides 4 to 16

Let us learn about the process of sending an e-mail.

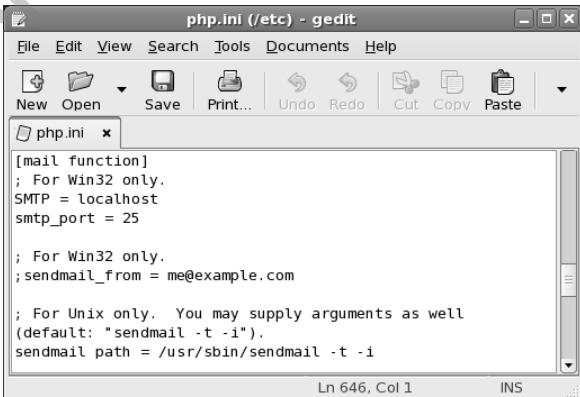
Sending an E-mail 1-13

- ◆ In PHP, you can send an e-mail using the `mail()` function
- ◆ Location of the current local mail server must be specified 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

Version 1.0 © Aptech Limited. Handling E-mail with PHP / Session 19 / Slide 4 of 26

Sending an E-mail 2-13

The contents of the `php.ini` configuration are displayed as follows:



```
[mail function]
; For Win32 only.
SMTP = localhost
smtp_port = 25

; For Win32 only.
;sendmail_from = me@example.com

; For Unix only. You may supply arguments as well
(default: "sendmail -t -i").
sendmail_path = /usr/sbin/sendmail -t -i
```

Version 1.0 © Aptech Limited. Handling E-mail with PHP / Session 19 / Slide 5 of 26

Sending an E-mail

3-13

- 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

Sending an E-mail

4-13

- mail.php:** Sending an e-mail using the `mail()` function

Snippet

```
<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!!!";
}
```

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 7 of 26

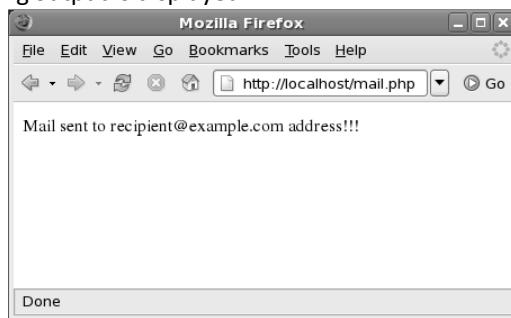
Sending an E-mail

5-13

```

else
{
    echo "Mail could not be sent to $to address!!!";
}
?>
</BODY>
</HTML>
```

The following output is displayed:



Sending an E-mail

6-13

- ◆ 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
- ◆ **`multi_rec.php`** - Sending an e-mail to multiple recipients

Snippet

```

<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]);
```

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 9 of 26

Sending an E-mail

7-13

```
$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>
```

Sending an E-mail

8-13

The following output is displayed:



The `mail()` function is used to send an e-mail.

The `trim()` function is used to remove blank spaces.

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 11 of 26

Sending an E-mail

9-13

- ◆ PHP also enables to read the e-mail addresses of the recipients from the text file and send an e-mail to them
- ◆ Steps for sending e-mail to multiple recipients whose addresses are stored in a text file are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the following text:

```
recipient1@example.com
recipient2@example.com
recipient2@example.com
```

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 12 of 26

Sending an E-mail

10-13

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

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 13 of 26

Sending an E-mail

11-13

6. Enter the code as shown in the Snippet:

Snippet

```
<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!!!";
}
```

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 14 of 26

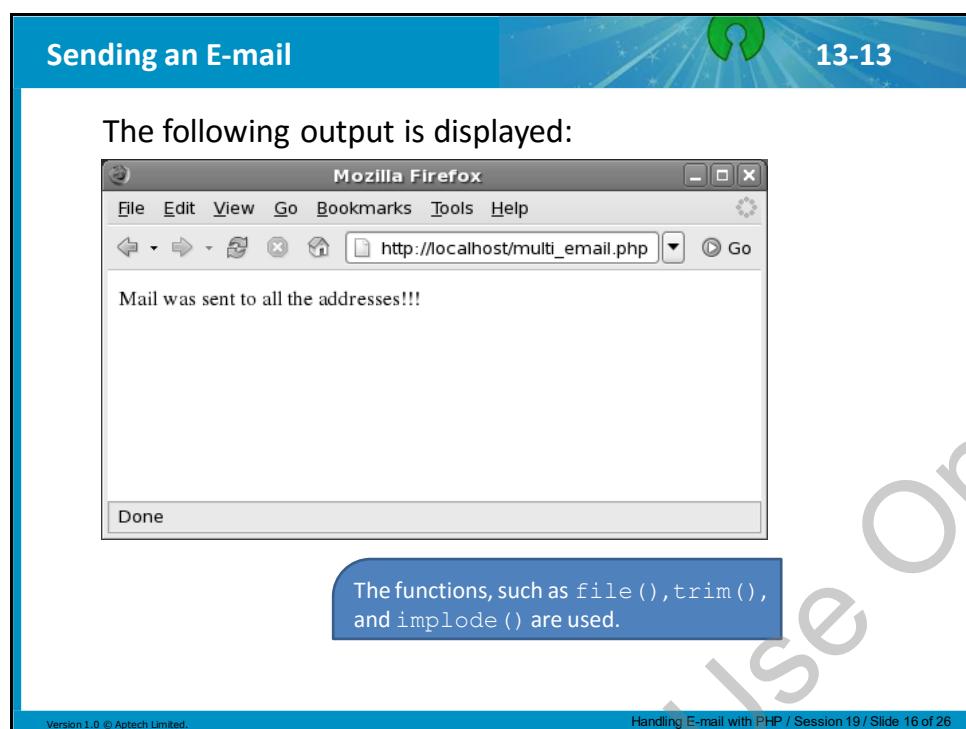
Sending an E-mail

12-13

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**

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 15 of 26



Tell the students the need for using `mail()` function. The `mail()` function in PHP is used to send an e-mail. Give an introduction to `mail()` function using slide 4. The contents that are to be present in the `php.ini` configuration file are displayed in slide 5. Use slide 6, explain to the students the syntax for the `mail()` function along with its parameters.

Use the example code snippet given in slides 7 and 8 to explain the students about the `mail()` function more elaborately. Describe the use of each line of code snippet given and show them the corresponding output.

Often, there may be a case where one needs to send the same e-mail message to several people. Instead of writing the e-mail several times, it is possible to specify multiple recipients at the same time for an e-mail. The effort required for this is considerably lesser than writing multiple e-mails. Now, continue to explain the code snippet given in slides 9 and 10, which is used to send e-mail to more than one recipient. The corresponding output for the code snippet is shown in slide 11.

PHP enables to read the e-mail addresses of the recipients from the text file and send e-mail to them. Using slides 12 to 15, explain the steps for sending e-mail to multiple recipients. Explain the use of `trim()`, `implode()`, and `uniqid()` functions that are used in the code. The output for the code snippet displayed as an acknowledgement, is shown in slide 16.

Additional Information:

For more information on e-mail in PHP, visit the following links:

http://www.w3schools.com/php/func_mail_mail.asp

<http://php.net/manual/en/function.mail.php>

http://www.tutorialspoint.com/php/php_sending_emails.htm

Slides 17 to 24

Let us see the process to attach files in an e-mail.

Attaching Files in an E-mail

1-8

- ◆ An attachment is
 - ❖ Any file that you want to send along with the e-mail
 - ❖ Attached with the e-mail by setting up the header

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 17 of 26

Attaching Files in an E-mail

2-8

- ◆ 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
 5. Open a new file in the **gedit** text editor
 6. Enter the code as shown in the Snippet

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 18 of 26

Attaching Files in an E-mail

3-8

Snippet

```
<?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";
```

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 19 of 26

Attaching Files in an E-mail

4-8

```
// 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-9959-1\"\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";
```

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 20 of 26

Attaching Files in an E-mail

5-8

```
// 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>";
}
?>
```

Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 21 of 26

Attaching Files in an E-mail

6-8

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**

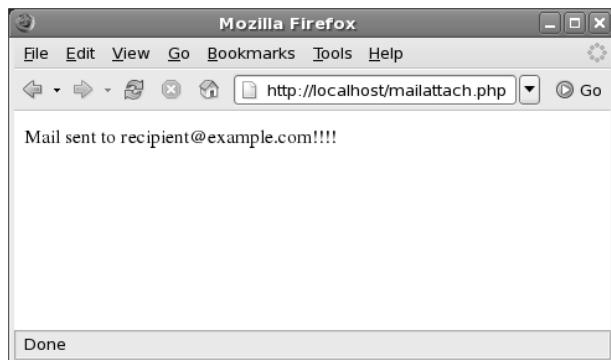
Version 1.0 © Aptech Limited.

Handling E-mail with PHP / Session 19 / Slide 22 of 26

Attaching Files in an E-mail

7-8

The following output is displayed:



A boundary is created between the actual e-mail and the attachments.

The `fopen()` function opens the file that is attached with the e-mail in the read mode.

Attaching Files in an E-mail

8-8

- ◆ The two built-in PHP function are as follows:
 - ◆ `base64_encode()` - used for encoding the specified string
 - ◆ `chunk_split()` - used for splitting the data into a series of smaller parts

Using slide 17, describe about file attachments. While sending an e-mail, it may often be necessary to send a file (such as an image, document, or some other type) as an attachment. The steps given in the slides are sufficient to explain the students the complete process of attaching files in an e-mail. Use slide 18 and tell the students to create a .txt file with a text editor to specify the code that is needed. Explain the each line of code snippets given in the slides 19 to 21. After specifying the code, the file is then saved. To see the output for the code, follow the steps given in slide 22. Tell the students that the output window is displayed as given in slide 23. Also, explain the students about the built-in PHP functions `base64_encode()` and `chunk_split()` using slide 24.

Additional Information:

Explain to the students that PHP by itself does not have any built-in mail-sending features. It relies on the existing mail server (usually Sendmail or Qmail) to perform the actual work. Tell the students that the mail server information must be specified in the php.ini file. This may include settings of mail servers including the Simple Mail Transfer Protocol (SMTP) details and so on.

Following are some of the configuration directives for PHP mail:

- **SMTP:** The default is 'localhost'. You can change this setting in your php.ini file.
- **SMTP_port:** The default is 25. You can change this setting in php.ini file. This modification is only necessary on Windows servers. Servers running Mac, Linux, or Unix will automatically detect the correct port number.
- **sendmail_from:** The default is NULL. This is the string to insert into the 'From:' and 'Return-Path:' strings in the header when sending mail from a Windows computer.
- **sendmail_path:** The default is NULL. PHP makes an attempt to find Sendmail for you, but if you have Sendmail installed in a non standard location or if you are using another mail server, you should modify the sendmail_path setting. Replace the default value with the path to Sendmail or another mail server.

For example, the php.ini file may have following settings:

```
[mail function]
; For Win32 only.
SMTP =

; For win32 only
sendmail_from =

; For Unix only
sendmail_path = /usr/sbin/sendmail -t -i
```

In-Class Question:

After you finish explaining about e-mail in PHP, you will ask the students a few In-Class questions. This will help you in reviewing their understanding of the topic.



Can you identify the function used in PHP to send e-mail?

Answer:

The mail() function is used in PHP to send e-mail. This function requires three mandatory arguments that specify the recipient's email address, the subject of the message, and the actual message. There may be additional optional parameters.



Which file needs to be modified to configure e-mail settings and directives?

Answer:

The php.ini file needs to be modified to configure e-mail settings and directives.

Slides 25 and 26

Let us summarize the session.

Summary 1-2

- ◆ 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

Version 1.0 © Aptech Limited. Handling E-mail with PHP / Session 19 / Slide 25 of 26

Summary 2-2

- ◆ 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

Version 1.0 © Aptech Limited. Handling E-mail with PHP / Session 19 / Slide 26 of 26

Use slides 25 and 26 to list and explain the summary of this session.

The summary points are as follows:

- 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.

22.3 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 24 – OOP Concepts

24.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

24.1.1 Objectives

At the end of this session, the learners 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

24.1.2 Teaching Skills

To teach this session, you should be well-versed with the use of the OOP concepts in PHP. You should be able to explain about inheritance in PHP. Also, tell them about the use of classes and constructors. Describe them about the use of objects.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students the overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives

- ◆ *Explain the OOP Concepts*
- ◆ *Explain inheritance*
- ◆ *Explain the use of classes*
- ◆ *Describe the use of constructors*
- ◆ *Explain the use of objects*

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 2 of 41

Tell the students that in this session they will learn OOP concepts in PHP. They will also learn about inheritance concept in brief. They will be able to describe about the use of the classes, constructors, and objects.

24.2 In-Class Explanations

Slide 3

Let us introduce OOP concepts in PHP.

Introduction

- ◆ Object-Oriented Programming (OOP) is:
 - ❖ Term used to characterize a programming language
 - ❖ Widely accepted paradigm for programming languages
- ◆ OOP uses three basic concepts are as follows:
 - ❖ Classes
 - ❖ Objects
 - ❖ Methods
- ◆ Additional concepts in object-oriented languages are as follows:
 - ❖ Inheritance
 - ❖ Abstraction
 - ❖ Polymorphism
 - ❖ Event Handling
 - ❖ Encapsulation

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 3 of 41

Using slide 3, introduce Object Oriented Programming (OOP) concepts in PHP.

OOP has gained immense popularity in the recent years since its inception in the 1960s. From software developers to students, everyone is striving to experience object-oriented programming. OOP has revolutionized the entire software industry and has emerged as a totally new concept unlike anything that has come up earlier in programming.

Until today, you might have used procedural programming languages such as C, which is considered as a traditional programming methodology. Procedural programming is relatively simple as it allows code reuse and needs less memory for execution. It uses procedures, also known as methods or functions that contain a series of instructions to be executed. However, one of the drawbacks of procedural programming is that data is not secure.

It is exposed to the entire program and easily accessible from anywhere within or outside the program. In procedural programming, it is difficult to derive new data types, which reduces extensibility. Also, it gives more importance to the processing of data rather than data itself.

'Object', as a concept, was introduced in the 1960s in a programming language called Simula 67. The term Object-oriented programming was introduced in 1970s with the invention of a language called Smalltalk by Alan Kay. OOP was coined to represent the use of messages and objects for programming. The popularity of OOP really caught momentum in the 1980s.

What is an object? An object is any person or a thing, living or non-living that has some characteristics or attributes which help to describe it.

Suppose that a person is holding a cup of tea. The teacup is an object, the tea inside the cup is also an object, and the tray on which the cup is placed is an object too. You can differentiate between tea and coffee, even though both are drinks, as both have different characteristics in terms of look and taste. Thus, one can characterize both tea and coffee as a type of drink that is their common characteristic. However, the color and taste of each can be used to differentiate them. Also, one can state that tea and coffee are objects belonging to category or class, drink. The color and taste attributes are applicable to any drink and thus, common to both tea as well as coffee.

OOP recognizes the need of building an application that represents the real world. Tell the students the various concepts of OOP available in PHP that are listed in slide 3.

Additional Information:

For more information on OOP concepts, visit the following links:

<https://www.udemy.com/blog/php-oop-tutorial/>

<http://www.phpro.org/tutorials/Object-Oriented-Programming-with-PHP.html#1>

Slides 4 to 8

Let us learn more about OOP.

Object-Oriented Programming

1-5

- ◆ Defines
 - ❖ Data type of the data structure
 - ❖ Type of functions to be performed on the data structure
- ◆ Unit of execution in an object-oriented system are objects
- ◆ Combines data and functions in a single unit

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 4 of 41

Object-Oriented Programming

2-5

- ◆ 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

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 5 of 41

Object-Oriented Programming

3-5

❖ **Class:**

- ❖ Contains a collection of similar types of objects
- ❖ 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

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 6 of 41

Object-Oriented Programming

4-5

❖ **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

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 7 of 41

❖ **Inheritance:**

- ❖ Specifies the process of creating a new class from the existing one
- ❖ The new class that is formed is called the derived class
- ❖ The existing class is called the base class

Using slide 4, describe OOP. Describe the pointers given on slide 4. Then, using slide 5, tell the students about objects and how they are used in programming. An object is the core element of OOP. Give some real world examples to illustrate the concept of objects. Any real world entity can be mapped to an object. For example, an employee or a particular model of a car or a specific book, and so forth.

Using slides 6 to 8, explain the students in brief about a class and features of OOP such as abstraction, encapsulation, polymorphism, and inheritance. Abstraction is defined as the process of hiding the unnecessary details and revealing only the essential features of an object to the user. It is implemented using a class.

Abstraction is a mechanism of showing only the relevant details of a process or artifact and hiding the irrelevant details pertaining to an object. For example, you open a book on human anatomy to study the mechanism involved in the walking movement. The first level of anatomy structure will show only the outer skin of the human body. The next level will show the body as composed of bones and muscles. Further, one might check the structure of muscle that contains tissues. The tissue is made of cells, cells again are made of molecules, and molecules are made up of atoms. This level of detail might not be necessary and may not be easy to understand.

Thus, any explanation must be given at the right level of detail and the rest should be abstracted. To know how a person can walk by attempting to study the mechanism at the atomic level is not only difficult but also irrelevant. It makes it complicated for an average person to understand.

Encapsulation helps to hide unwanted details of an entity from other entities such as other classes. It is implemented using objects and methods. In general, 'encapsulate' means 'to enclose

something'. In OOP, encapsulation is used to consciously hide information in a small part of a program. It is a feature used to restrict access to some of the data members by objects. In other words, it provides bundling of data members and methods into an enclosed structure.

Consider a real world situation where a person, Robin wishes to put his money and other valuables in such a place where no one except him can access it. Robin goes to a bank where he is provided with a safe whose key only he can possess. Robin puts his belongings into the safe, takes the key, and goes away. Another person named Chris comes to the bank with the same purpose. However, Chris is allocated another safe into which he can place his valuables. Both Robin's and Chris' safes are isolated from each other and neither can access the other person's safe. Thus, both Robin and Chris accessed the same service of the bank, yet they were only allowed to access items for which they were authorized. The bank safe worked as an enclosing entity, which protected access to Robin's valuables from Chris and vice versa. In a similar manner, OOP achieves encapsulation by creating a class, which encloses the data members and methods and prevents their unauthorized access.

Inheritance allows to reuse full or part of the functionality of a class in another class by inheriting it. 'Inheritance', in real life, means to pass on characteristics, property, titles, and rights of an Individual to his/her successors. In day to day life, one may often find that a person looking similar to his/her father or mother in facial features, height, skin color, eyes, and mannerism. In addition to physical looks, behavior can also be inherited. The way of walking, talking, temperament, and other characteristics are inherited which relates an individual with his/her ancestors. This gives rise to a thought that nature reuses attributes and characteristics of people of one generation to pass on to another generation of the same lineage. However, the descendants of a generation will also have characteristics of their own, apart from those inherited, which helps to differentiate them from their ancestors.

In OOP, inheritance helps to define hierarchical relationships among classes at different levels and gives code reusability. A class can pass on its state and behavior to its child classes. A child class is one, which inherits from another class called parent class or super class. There are different types of inheritance namely, single, multiple, multilevel, hierarchical, and hybrid inheritance.

'Poly' means 'many' and 'morphos' means 'forms'. The term polymorphism is a Greek word which means 'many forms'. A polymorph is one that can appear in more than one form. For example, a chameleon changes its color each time to adjust it according to the surroundings. This helps it to hide from its predators. Another example is an Amoeba that keeps changing its shape to facilitate moving around.

Give a real world example of a man/woman in different roles. He/she can be considered a polymorph, as he/she behaves differently under different circumstances. Apart from being a human being, he/she can be an employee when at work. At home, the same person may be a spouse and a parent whereby roles and responsibilities change and the duty of the person is to cater to the needs

of the family members. Polymorphism in OOP is the ability of an object to respond to the same message in different ways.

Thus, Encapsulation, Abstraction, Inheritance, and Polymorphism are the four pillars of OOP that make it a robust and flexible tool for designing computer applications.

Additional Information:

For more information on the features of OOP, visit the following links:

<https://www.clear.rice.edu/mec517/Books/oop3.pdf>

<http://www.jamesbooth.com/OOPBasics.htm>

Slides 9 to 12

Let us learn in detail about inheritance.

Inheritance 1-4

- ◆ Inheritance means creating a new class with the help of a base class
- ◆ 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
- ◆ The `extends` keyword needs to be used for inheriting a new class from the base class
- ◆ A derived class can also have its own variables and functions

Version 1.0 © Aptech Limited. OOP Concepts/ Session 21 / Slide 9 of 41

Inheritance

2-4

- ◆ The different types of inheritances are as follows:

- ◆ **Single Inheritance:**

- ◆ Contains only one base class and 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 and inherits the properties of all base classes
 - ◆ A class derived from this inheritance works with the derived properties along with its own properties

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 10 of 41

Inheritance

3-4

- ◆ **Hierarchical Inheritance:**

- ◆ Contains one base class and the properties of a single base class can be used multiple times in the sub-multiples
 - ◆ The derived class contains its own properties and also uses the derived properties of all the base classes

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 11 of 41

Inheritance

4-4

❖ **Multilevel Inheritance:**

- ❖ Contains one base class and 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

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 12 of 41

Using slide 9, describe about the concept of inheritance. First, tell the students what inheritance means. With that, explain the students about base class and derived class. It is to be noted that 'extends' keyword needs to be used in PHP for inheriting a new class from the base class. There are several types of inheritance. Using slides 10 to 12, explain each type of inheritance in brief.

Additional Information:

For more information on inheritance, visit the following links:

<https://www.clear.rice.edu/mec517/Books/oop3.pdf>

<http://www.jamesbooth.com/OOPBasics.htm>

<http://jaskokojn.com/2013/05/29/inheritance-php-oop-tutorial/>

<http://php.net/manual/en/language.oop5.inheritance.php>

In-Class Question:

After you finish explaining about inheritance, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Can you explain in brief what is meant by an object?

Answer:

An 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.

Slides 13 to 26

Let us understand the process of creating a class.

Creating a Class

1-14

- ◆ 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 new derived class uses all the properties of the base class including its own properties

Version 1.0 © Aptech Limited. OOP Concepts/ Session 21 / Slide 13 of 41

Creating a Class

2-14

- ◆ Declaring a class

Syntax

```

class class_name
{
    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
var1, var2 - specifies the variable name
function_name() - specifies the function name
return - returns the value after performing the specified mathematical function

Version 1.0 © Aptech Limited. OOP Concepts/ Session 21 / Slide 14 of 41

Creating a Class

3-14

- ◆ In the class syntax:
 - ❖ Definition for a class is included within curly braces
 - ❖ Variables defined in the class are local to the class
 - ❖ Variable inside a class is declared with the keyword `var`
 - ❖ Class can also use global variables
 - ❖ Functions inside a class may use its own local variables or may use the class variables

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 15 of 41

Creating a Class

4-14

- ◆ Following is the code for a class named `empdetail`

Snippet

```
<?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;
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 16 of 41

Creating a Class**5-14**

```

}
function enterdet($dept, $design)
{
    $this->empdept=dept;
    $this->empdesign=design;
}
?>
```

- ◆ This file need to be saved 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

Creating a Class**6-14**

- ◆ The `enteremp()` function accepts the employee id, employee name, and city
- ◆ The `enterdet()` function accepts the employee department and the designation
- ◆ The `filename.inc` file is included in the `filename.php` file with the help of the `include` keyword
- ◆ The `include` keyword enables to incorporate any type of file with the main file

Creating a Class

7-14

- The syntax to include a file in the program is as follows:

Syntax

```
include "filename.ext";
```

- Assuming that the code empdetail.inc is saved, it will be used in the next code

Snippet

```
<?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;
?>
```

Version 1.0 © Aptech Limited.

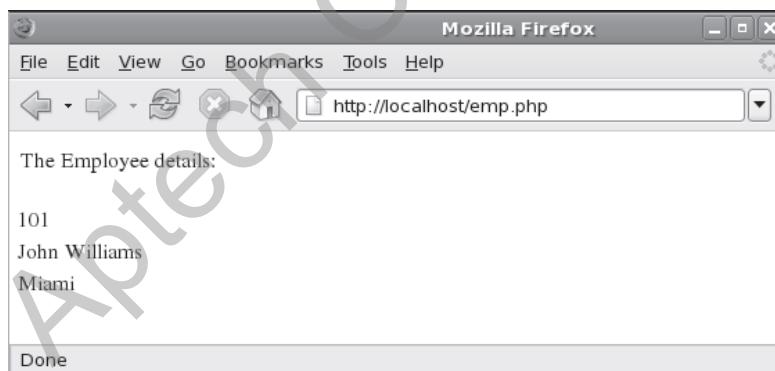
OOP Concepts/ Session 21 / Slide 19 of 41

Creating a Class

8-14

- Save this file as **emp . php** and open it in Mozilla Firefox Web Browser

The following output is displayed:



Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 20 of 41

Creating a Class

9-14

- ◆ 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

Creating a Class

10-14

- ◆ **salary.inc** - Deriving the `net_salary` class from the `salary` class in PHP

Snippet

```
<?php
class salary
{
    public $hra;
    public $ta;
    public $tax;
    public function hra_calc($basic)
    {
        $hra = $basic * 0.25;
        return $hra;
    }
}
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 22 of 41

Creating a Class**11-14**

```
public function travelallow_calc($basic)
{
    $ta = $basic * 0.08;
    return $ta;
}

public function tax_calc($basic)
{
    $tax = $basic * 0.05;
    return $tax;
}
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 23 of 41

Creating a Class**12-14**

```
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;
}
}
?>
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 24 of 41

Creating a Class

13-14

- ◆ 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

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 25 of 41

Creating a Class

14-14

- ◆ This class is used in a file named `salarydemo.php`

Snippet

```
<?php
include "salary.inc";
echo "The Salary details of the employee: <BR><BR>";
$sal = new net_salary();
echo $sal->net(372500);
?>
```

The following output is displayed:

The screenshot shows a Mozilla Firefox window with the URL `http://localhost/salarydemo.php`. The page content is as follows:

```
The Salary details of the employee:
HRA: 93125
TA: 29800
TAX: 18625
NET SALARY: 476800
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 26 of 41

Use slide 13 to describe about a class in brief. Classes are the templates that are used to define objects. For example, you can create a class named `Car` that defines the characteristics and behavior (actions) of the car. All the objects of type `Car` will have these common characteristics though their values may be different. For example, a car of make Toyota and a car of make Ford will have mileage but the actual values will differ.

Explain the students the syntax for creating a class. Use an example to explain them more elaborately. Show them the syntax that is given in slide 14. Also, while explaining the syntax, tell them how a class name, function name, and variables are differentiated to work in their own way.

Using slide 15, explain them about class declaration. Then, explain the example of code snippet given in slides 16 and 17 that creates a new class called empdetail. Explain the code using slides 17 and 18. Tell the students about the enterdet() and enteremp() functions and the significance of the include statement.

Also, tell the students, the syntax to include a file in the program. Use slide 19 to explain the given code snippet. Show them the corresponding output that is displayed in slide 20.

It is important to explain the students about inheriting a class. Explain each line of code in brief and tell them how it is executed using slides 21 to 24. Explain the given points in slides 25 and 26, to explain the given code snippet and its output.

Additional Information:

For more information on classes in PHP, visit the following links:

<http://www.htmlgoodies.com/beyond/php/object-orientated-programming-in-php-class-1-principles.html>

<http://www.techflirt.com/tutorials/oop-in-php/classes-and-objects-php.html>

Slides 27 to 34

Let us understand about constructors.

The screenshot shows a presentation slide with a blue header bar. The title 'Creating a Constructor' is centered in white text. In the top right corner of the header, there is a small green circular icon with a white symbol and the number '1-8'. The main content area has a white background. It contains a single bullet point under the heading 'Constructor':

- ◆ Constructor
 - ◆ Is a special function that is a member of the class
 - ◆ Is called in the main program by using the new operator
 - ◆ Is invoked whenever an object of the class is created
 - ◆ Is declared, it provides a value to all the objects that are created in the class and this process is known as initialization
 - ◆ Has the same name as that of its class name

At the bottom of the slide, there is a thin blue footer bar with the text 'Version 1.0 © Aptech Limited.' on the left and 'OOP Concepts/ Session 21 / Slide 27 of 41' on the right.

Creating a Constructor

2-8

Syntax

```
class class_name
{
    var class_variable;
    Function constructor_name()
}
```

Where,
constructor_name - specifies the name of
the constructor

- ◆ **emp_constructor.inc** - Creating a constructor for the class named empdetail

Snippet

```
class empdetail
{
    var $empid;
    var $empname;
    var $empcity;
    var $empdept;
    function empdetail($id, $name, $city, $dept)
    {
        $this->empid=$id;
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 28 of 41

Creating a Constructor

3-8

```
$this->empname=$name;
$this->empcity=$city;
$this->empdept=$dept;
}
?
?>
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 29 of 41

Creating a Constructor

4-8

- 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
- emp_constructor.php** - Calling the constructor of the **empdetail** class

Snippet

```
<?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;
?>
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 30 of 41

Creating a Constructor

5-8

- The **new** operator creates a new object of the **empdetail** class
- It calls the functions defined in the **empdetail** class

The following output is displayed:



Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 31 of 41

Creating a Constructor

6-8

- ◆ **stringtest.php** - Calling a constructor from the derived class

Snippet

```
<?php
class string
{
function string()
{
echo "This function is a constructor.";
}
function stringdisp()
{
echo "This is function.";
}
}
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 32 of 41

Creating a Constructor

7-8

```
class display extends string
{
function display()
{
echo "This is a constructor of a new class.";
}
}
$displ = new string;
echo "<BR><BR>";
$disp = new display;
?>
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 33 of 41

Creating a Constructor

8-8

The following output is displayed:



Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 34 of 41

A constructor is defined as a special function that is a member of the class. Explain the points given in slide 27, to understand the use of constructor. It is to be noted that the constructor has same name as that of the class name. Use slides 28 and 29 to explain the syntax for creating constructor and the code snippet. Explain the constructor calling method with the points given in slide 30. Also, explain the code that describes the call to the constructor of main class called `empdetail`. Tell them about the new operator and its uses. Show them the corresponding output that is displayed in slide 31. Then, using slides 32 and 33, explain the call to the constructor of derived class. The output for the given code snippet is shown in slide 34.

Slides 35 to 39

Let us learn about creating and using objects.

Creating and Using Objects

1-5

- ◆ An object is a self-contained entity that includes both data and procedures to manipulate the data
- ◆ 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

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 35 of 41

Creating and Using Objects

2-5

- ◆ 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

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 36 of 41

Creating and Using Objects

3-5

- ◆ **usermail.php** - Create an object for the class, **usermail**

Snippet

```
<?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";
```

Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 37 of 41

Creating and Using Objects

4-5

```
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();
?>
```

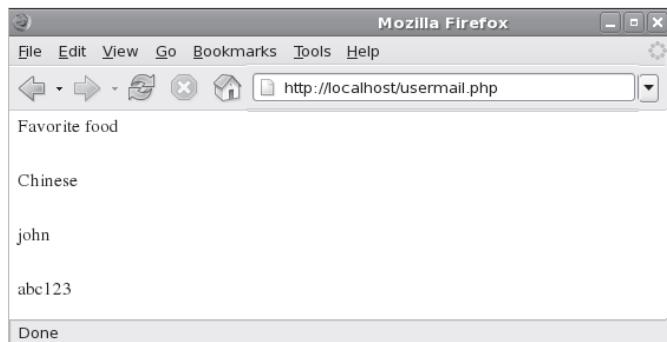
Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 38 of 41

Creating and Using Objects

5-5

The following output is displayed:



Version 1.0 © Aptech Limited.

OOP Concepts/ Session 21 / Slide 39 of 41

Use slide 35 to describe about objects. List the points given on slide 35. Explain the syntax for creating an object for a class as given in slide 36. Mention that the object created is later used to call the corresponding methods of that class. Using slides 37 and 38, explain each line of given code snippet. Show the corresponding output that is displayed in slide 39.

In-Class Question:

After you finish with the explanation of creating and using objects, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



What is meant by a constructor?

Answer:

A constructor is a special function that is a member of the class and is called in the main program by using the new operator. It is invoked whenever an object of the class is created. It provides a value to all the objects that are created in the class and this process is known as initialization. It has the same name as that of its class name.

Additional Information:

For more information on constructors in PHP, visit the following links:

<http://phpduck.com/oop-basics-in-php-5/>

<http://www.c-sharpcorner.com/UploadFile/736ca4/oops-concepts-in-php/>

<http://www.techflirt.com/tutorials/oop-in-php/index.html>

Slides 40 and 41

Let us summarize the session.

Summary 1-2

- ◆ 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.

Version 1.0 © Aptech Limited. OOP Concepts/ Session 21 / Slide 40 of 41

Summary 2-2

- ◆ 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

Version 1.0 © Aptech Limited. OOP Concepts/ Session 21 / Slide 41 of 41

Use slides 40 and 41 to list and explain the summary of this session.

The summary points are follows:

- 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.

24.3 Post Class Activities for Faculty

Tips:

You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 26 – Generator Delegation and Throwable Interface

26.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

26.1.1 Objectives

By the end of this session, learners 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.

26.1.2 Teaching Skills

To teach this session, you should be well versed with the 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.

You should teach the concepts in the class using the code snippet provided. For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities

Follow the order given here during In-Class activities.

Overview of the Session

Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Objectives



- ◆ 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.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 2 of 32

26.2 In-Class Explanations

Slides 3 to 5

Let us understand the use of generators.

Generators



1-3

- ◆ Returns/yields values on demand
- ◆ Return multiple values from a function
- ◆ Use the `yield` keyword to return data

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 3 of 32

Generators 2-3

- Demonstrating the use of a generator that prints a range of values

Snippet

```
<?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;
}
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 4 of 32

Generators 3-3

- Displays the following output:

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 5 of 32

Using slide 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.

Using slide 4, display the code that uses a generator to print a range of values. Use slide 5 to display the range of values.

Add that 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.

Slides 6 to 8

Let us now understand the usage of generator return expressions.

Generator Return Expressions  **1-3**

- ◆ Demonstrating the use of range() function

Snippet

```
<?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 $x;
}
}
else {
if ($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 ";
}
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 6 of 32

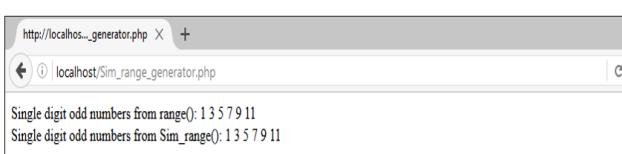
Generator Return Expressions  **2-3**

```
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 ";
}
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 7 of 32

Generator Return Expressions  **3-3**

- ◆ Displays the following output:



http://localhost/Sim_range_generator.php X +
 ↶ | localhost/Sim_range_generator.php | C
Single digit odd numbers from range():1357911
Single digit odd numbers from Sim_range():1357911

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 8 of 32

Tell them that 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.

Using slides 6 and 7, display the code that uses the `range()` function. The `range()` function creates an array consists of set of values. This function returns a set of values from low to high. If the value of low parameter is greater than the value of high parameter, then the range will return values from high to low.

Explain that the result of the code will display 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.

Using slide 8, display the output of the code.

Slides 9 to 12

Let us now understand generator delegation.

Explain that generator delegation helps programmers to write a generator that can yield other generators, arrays, and traversable objects.

Using slide 9, display the syntax that is used to yield values.

Generator Delegation  1-4

- ◆ Enables to yield values from other generator, arrays, and traversable objects

Syntax

```
yield from <thing_to_iterate>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 9 of 32

Generator Delegation  2-4

- ◆ Demonstrating the use of generator delegation

Snippet

```
<?php
    function DisplayHello() {
        yield "Hello";
        yield " ";
        yield "World!";
        yield from DisplayGoodbye();
    }
    function DisplayGoodbye() {
        yield "Goodbye";
        yield " ";
        yield "Mars!";
    }
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 10 of 32

Generator Delegation

3-4

```
$gen = DisplayHello();
foreach ($gen as $value) {
    echo $value;
    echo '</br>';
}
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 11 of 32

Generator Delegation

4-4

- ◆ Displays the following output:

http://localhost/delegation_generator.php

Hello
World!
Goodbye

Mars!

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 12 of 32

Using slides 10 and 11, display another code that display a function that yields another generator function.

Explain that in the code there are 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()`. Display slide 12 to show the output of the code.

In-Class Question:

After you finish explaining the various type comparisons, you will ask the students an In-Class question. This will help you in reviewing their understanding of the topic.



Explain the difference between a normal function and a generator function.

Answer:

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.

Slides 13 to 17

Let us now understand about exceptions.

Exceptions 1-5

- ◆ Are a mechanism for error handling
- ◆ An exception handling codes include:

Code	Description
try block	If exceptions do not occur, then the code executes and exists successfully. OR If exceptions occur, then the exception is caught by the catch block and action is taken to handle to exception.
catch block	If exceptions occur, then action statements are included in the block to handle the exception.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 13 of 32

Exceptions 2-5

Code	Description
throw	A method to manually trigger an exception. However, each throw should be associated with an corresponding catch.
finally	A block usually executed after the try and catch blocks. The code within this block is executed irrespective of whether an exception has been thrown or before normal execution continues.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 14 of 32

Exceptions

3-5

- ◆ Demonstrating the use of try-catch-finally statements

Snippet

```
<?php
function computeDiv($num) {
    if (!$num) {
        throw new Exception('Division by zero.');
    }
    return 1/$num;
}
```

Exceptions

4-5

```
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>";
?>
```

Exceptions

5-5

- ◆ Displays the following output:

```
Caught exception: Division by zero.
Now in finally block.
Program execution continues
```

Explain that in PHP 5 and later versions support exceptions, which are a mechanism for error handling. Exceptions are supported by several object-oriented programming languages.

Display slide 13 and explain the various exception handling code.

Using slides 15 and 16, demonstrate an example that uses try-catch-finally statements.

Explain that 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.

Display slide 17 to show the output of the code.

Slides 18 to 20

Let us now learn more about how to catch exceptions and errors.

E_ERROR 1-3

- A fatal runtime error.
- A non recoverable error.
- The script or program terminates prematurely when this error occurs.
- The error enables backward compatibility.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 18 of 32

E_ERROR 2-3

- ◆ Demonstrating the use of E_ERROR error

Snippet

```
<?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();
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 19 of 32

E_ERROR 3-3

- ◆ Displays the following output:

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 20 of 32

Explain that 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.

Let us see how to handle the `E_ERROR`. Use slide 18 to explain about the error.

Explain that, 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.

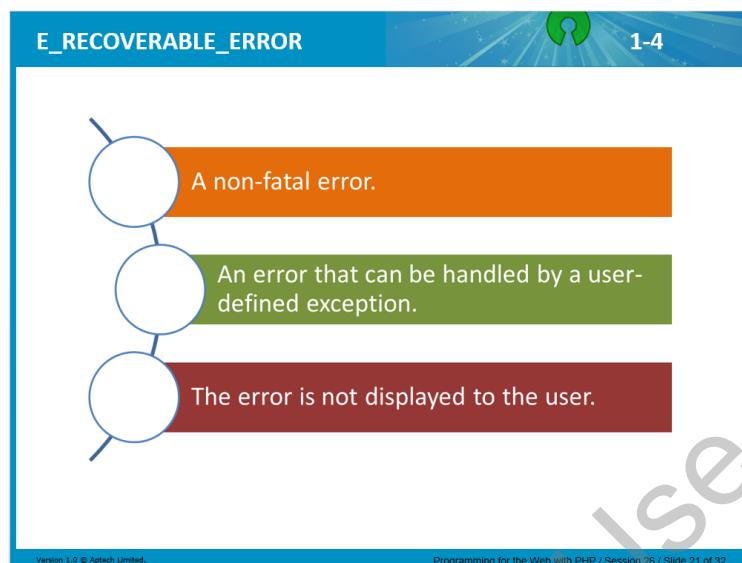
Let us see an example to handle `E_ERROR`.

Explain that in the code `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` and results in an error.

Display slide 20 to show the error.

Slides 21 to 24

Let us now understand about the `E_RECOVERABLE_ERROR`.



E_RECOVERABLE_ERROR

2-4

- ◆ Demonstrating the use of `E_RECOVERABLE_ERROR`

Snippet

```
<?php
error_reporting(E_RECOVERABLE_ERROR);
function handle_error($errno,
$errorstr,$error_file,$error_line) {
echo "<b>Error:</b> [".$errno."] $errorstr -
$error_file:$error_line";
echo "</br>";
echo "Terminating PHP Script";
die();
}

//set error handler
set_error_handler("handleError", E_RECOVERABLE_ERROR);

//trigger error
myFunction();
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 22 of 32

E_RECOVERABLE_ERROR

3-4

- ◆ Displays the following output:

http://localhost/E_RECOVERABLE_ERROR_demo.php

localhost/E_RECOVERABLE_ERROR_demo.php

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 23 of 32

The slide has a blue header bar with the title 'E_RECOVERABLE_ERROR' on the left and a small green circular icon with a white question mark on the right. The number '4-4' is in the top right corner. The main content area contains a bulleted list: '◆ Benefits of handling fatal errors as exceptions are:' followed by three items in colored boxes: 'Enables to handle fatal errors using the try-catch block' (green), 'Helps handle catchable errors easily' (purple), and 'Helps easy debugging' (orange). At the bottom of the slide, there is a thin blue footer bar with the text 'Version 1.0 © Aptech Limited.' on the left and 'Programming for the Web with PHP / Session 26 / Slide 24 of 32' on the right.

- ◆ Benefits of handling fatal errors as exceptions are:

Enables to handle fatal errors using the try-catch block

Helps handle catchable errors easily

Helps easy debugging

Display slide 21 and explain that 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 see an example to handle `E_RECOVERABLE_ERROR`. Display slide 22 to show the code to handle the error.

Explain that the code is 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.

Display slide 23 to display the output.

Add further to explain that 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.

Display slide 24 to show the benefits of handling fatal errors.

Slides 25 to 30

Let us now understand about the Throwable interface.

Throwable Interface  1-6

- ◆ Exception and Error classes implement the Throwable interface.
- ◆ The root of the hierarchy was \Exception in the previous versions of PHP.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 25 / Slide 25 of 32

Throwable Interface  2-6

- ◆ Demonstrating the hierarchy of classes and interfaces.

➤ Interface Throwable

- Exception implements Throwable
 - LogicException (extends Exception)
 - InvalidArgumentException (extends LogicException)
 - ...
- Error implements Throwable (Replaces EngineException)
 - TypeError extends Error
 - ParseError extends Error
 - ...

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 26 of 32

Throwable Interface  3-6

- ◆ Demonstrating a fatal error where exception is not caught and error is thrown.

Snippet

```
<?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";
}
?>
```

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 27 of 32

Throwable Interface

4-6

- ◆ Displays the following output:

The screenshot shows a browser window with the URL `http://localhost/exception_1.php`. The page displays two error messages in red boxes:

- (!)- Fatal error: Uncaught TypeError: Argument 1 passed to add() must be of the type integer, string given, called in C:\wamp64\www\exception_1.php on line 8 and defined in C:\wamp64\www\exception_1.php on line 3
- (!)- TypeError: Argument 1 passed to add() must be of the type integer, string given, called in C:\wamp64\www\exception_1.php on line 8 in C:\wamp64\www\exception_1.php on line 3

Below the errors is a "Call Stack" table:

#	Time	Memory	Function	Location
1	0.0003	361200	[main]()	...exception_1.php:0
2	0.0003	361264	add()	...exception_1.php:8

Throwable Interface

5-6

- ◆ Demonstrating to catch an Exception and Error

Snippet

```
<?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";
}
?>
```

Throwable Interface

6-6

- ◆ Displays the following output:

The screenshot shows a browser window with the URL `http://localhost/n_demo1.php`. The page displays the text "Error occured in the program".

Display slide 25 and explain about the interface. Display slide 26 and explain the hierarchy of classes and interfaces. Now, let us look at an example where the exception is not caught and an

error is thrown. Display slide 27 to show the code. Explain that in the code 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 is thrown.

Display slide 28 to show the error.

Add further that the code 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.

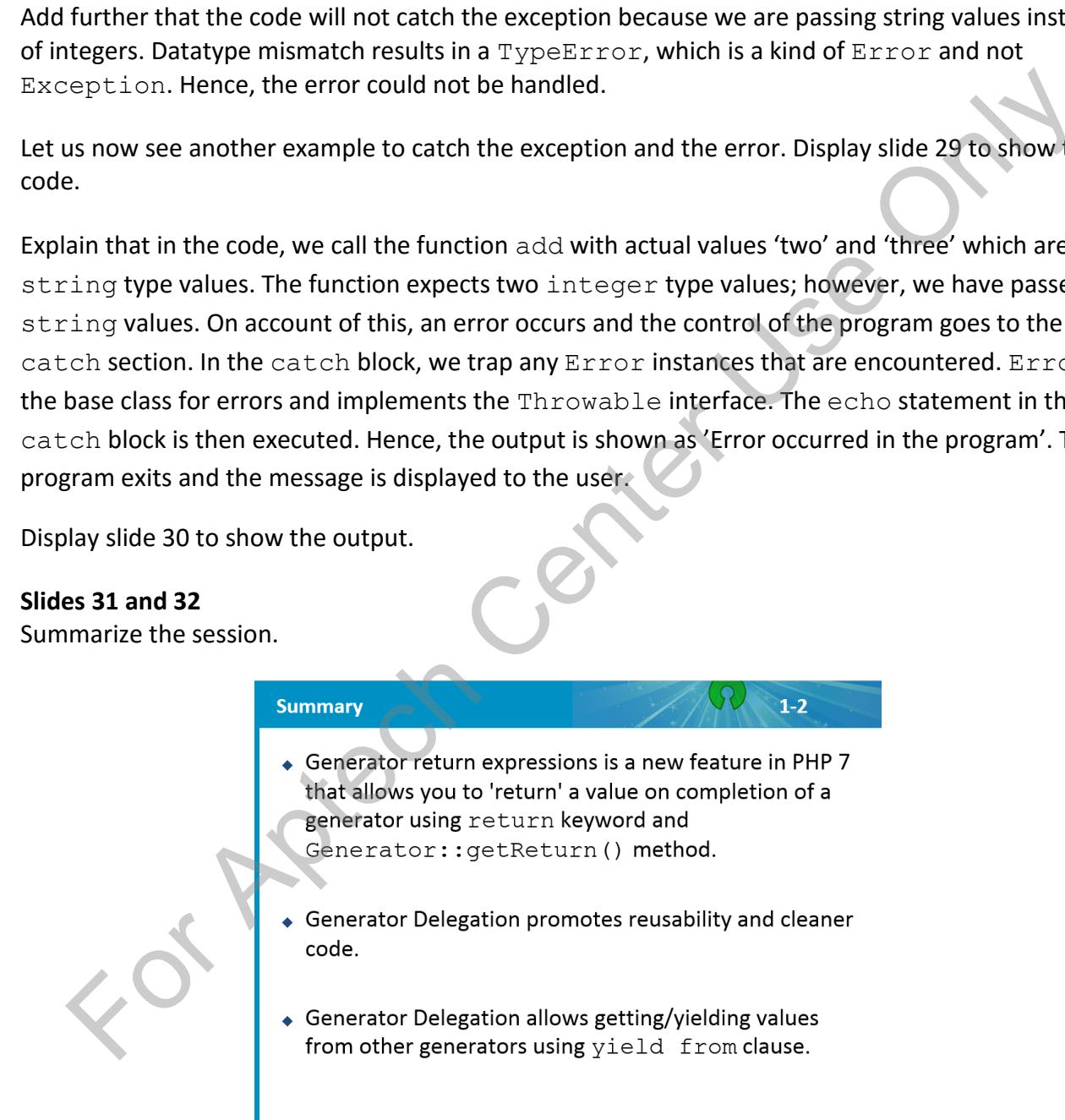
Let us now see another example to catch the exception and the error. Display slide 29 to show the code.

Explain that in the code, 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.

Display slide 30 to show the output.

Slides 31 and 32

Summarize the session.



The slide is titled 'Summary' and includes a navigation icon and the number '1-2'. It contains three bullet points about Generator Delegation:

- ◆ 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.

Version 1.0 © Aptech Limited. Programming for the Web with PHP / Session 26 / Slide 31 of 32

- ◆ 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.

Using slides 31 and 32, explain each of the following points in brief:

- A generator would return/yield the values on demand. 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.
- Generator return expressions allow you to 'return' a value upon successful completion of a generator using the `return` keyword and the `Generator::getReturn()` method.
- The `range()` function creates an array consists of set of values. This function returns a set of values from low to high.
- Exceptions are a mechanism of error handling. The various error handling codes are `try` block, `catch` block, `throw`, and `finally` statements.
- The `E_ERROR` a runtime error, which is fatal and the error is not recoverable.
- The `E_RECOVERABLE_ERROR` is a non-fatal error and can be handled by a user-defined exception. This error is not displayed to the user.

26.3 Post Class Activities for Faculty

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the session.