Basic Computer Coding:
PHP

2nd Edition

BIBLIOTEX
Digital Library

BASIC COMPUTER CODING: PHP

2nd Edition

BIBLIOTEX
Digital Library

www.bibliotex.com

BASIC COMPUTER CODING: PHP

2ND EDITION

www.bibliotex.com

email: info@bibliotex.com

e-book Edition 2022

ISBN: 978-1-98467-609-2 (e-book)

In Collaboration with 3G E-Learning LLC. Originally Published in printed book format by 3G E-Learning LLC with ISBN 978-1-98465-902-6

EDITORIAL BOARD

Aleksandar Mratinković was born on May 5, 1988 in Arandjelovac, Serbia. He has graduated on Economic high school (2007), The College of Tourism in Belgrade (2013), and also has a master degree of Psychology (Faculty of Philosophy, University of Novi Sad). He has been engaged in different fields of psychology (Developmental Psychology, Clinical Psychology, Educational Psychology and Industrial Psychology) and has published several scientific works.

Dan Piestun (PhD) is currently a startup entrepreneur in Israel working on the interface of Agriculture and Biomedical Sciences and was formerly president-CEO of the National Institute of Agricultural Research (INIA) in Uruguay. Dan is a widely published scientist who has received many honours during his career including being a two-time recipient of the Amit Golda Meir Prize from the Hebrew University of Jerusalem, his areas of expertise includes stem cell molecular biology, plant and animal genetics and bioinformatics. Dan's passion for applied science and technological solutions did not stop him from pursuing a deep connection to the farmer, his family and nature. Among some of his interest and practices counts enjoying working as a beekeeper and onboard fishing.

Hazem Shawky Fouda has a PhD. in Agriculture Sciences, obtained his PhD. From the Faculty of Agriculture, Alexandria University in 2008, He is working in Cotton Arbitration & Testing General Organization (CATGO).

Felecia Killings is the Founder and CEO of LiyahAmore Publishing, a publishing company committed to providing technical and educational services and products to Christian Authors. She operates as the Senior Editor and Writer, the Senior Writing Coach, the Content Marketing Specialist, Editor-in-Chief to the company's quarterly magazine, the Executive and Host of an international virtual network, and the Executive Director of the company's online school for Authors. She is a former high-school English instructor and professional development professor. She possesses a Master of Arts degree in Education and a Bachelor's degree in English and African American studies.

Dr. Sandra El Hajj, Ph.D. in Health Sciences from Nova Southeastern University, Florida, USA is a health professional specialized in Preventive and Global Health. With her 12 years of education obtained from one of the most prominent universities in Beirut, in addition to two leading universities in the State of Florida (USA), Dr. Sandra made sure to incorporate interdisciplinary and multicultural approaches in her work. Her long years of studies helped her create her own miniature world of knowledge linking together the healthcare field with Medical Research, Statistics, Food Technology, Environmental & Occupational Health, Preventive Health and most noteworthy her precious last degree of Global Health. Till today, she is the first and only doctor specialized in Global Health in the Middle East area.

Fozia Parveen has a Dphil in Sustainable Water Engineering from the University of Oxford. Prior to this she has received MS in Environmental Sciences from National University of Science and Technology (NUST), Islamabad Pakistan and BS in Environmental Sciences from Fatima Jinnah Women University (FJWU), Rawalpindi.

Igor Krunic 2003-2007 in the School of Economics. After graduating in 2007, he went on to study at The College of Tourism, at the University of Belgrade where he got his bachelor degree in 2010. He was active as a third-year student representative in the student parliament.Then he went on the Faculty of science, at the University of Novi Sad where he successfully defended his master's thesis in 2013. The crown of his study was the work titled Opportunities for development of cultural tourism in Cacak". Later on, he became part of a multinational company where he got promoted to a deputy director of logistic. Nowadays he is a consultant and writer of academic subjects in the field of tourism.

Dr. Jovan Pehcevski obtained his PhD in Computer Science from RMIT University in Melbourne, Australia in 2007. His research interests include big data, business intelligence and predictive analytics, data and information science, information retrieval, XML, web services and service-oriented architectures, and relational and NoSQL database systems. He has published over 30 journal and conference papers and he also serves as a journal and conference reviewer. He is currently working as a Dean and Associate Professor at European University in Skopje, Macedonia.

Dr. Tanjina Nur finished her PhD in Civil and Environmental Engineering in 2014 from University of Technology Sydney (UTS). Now she is working as Post-Doctoral Researcher in the Centre for Technology in Water and Wastewater (CTWW) and published about eight International journal papers with 80 citations. Her research interest is wastewater treatment technology using adsorption process.

Stephen obtained his PhD from the University of North Carolina at Charlotte in 2013 where his graduate research focused on cancer immunology and the tumor microenvironment. He received postdoctoral training in regenerative and translational medicine, specifically gastrointestinal tissue engineering, at the Wake Forest Institute of Regenerative Medicine. Currently, Stephen is an instructor for anatomy and physiology and biology at Forsyth Technical Community College.

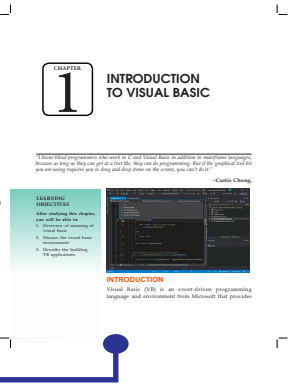
Michelle holds a Masters of Business Administration from the University of Phoenix, with a concentration in Human Resources Management. She is a professional author and has had numerous articles published in the Henry County Times and has written and revised several employee handbooks for various YMCA organizations throughout the United States.

HOW TO USE THE BOOK

This book has been divided into many chapters. Chapter gives the motivation for this book and the use of templates. The text is presented in the simplest language. Each paragraph has been arranged under a suitable heading for easy retention of concept. Keywords are the words that academics use to reveal the internal structure of an author's reasoning. Review questions at the end of each chapter ask students to review or explain the concepts. References provides the reader an additional source through which he/she can obtain more information regarding the topic.

LEARNING OBJECTIVES

See what you are going to cover and what you should already know at the start of each chapter


CHAPTER 1

INTRODUCTION TO VISUAL BASIC

"I know blind programmers who work in C and Visual Basic in addition to mainframe languages, because as long as they can get at a text file, they can do programming. But if the graphical tool kit you are using requires you to drag and drop items on the screen, you can't do it."

–Curtis Chong,

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Overview of meaning of visual basic
2. Discuss the visual basic programming
3. Describe the building VB applications

INTRODUCTION

Visual Basic (VB) is an event-driven programming language and environment from Microsoft that provides

enhancements, including the striking ability of creating web based applications. The extended support for Visual Basic 6.0 was ceased in the month of March in 2008. The basic parts of development environment of Visual Basic 6, however, still run in all the 32-bit Microsoft windows, including Windows 8.1.

After the cessation of mainstream and extended support for Visual Basic 6.0 caused a number of programs to show concern. The community members then created a lobby of users and a petition was signed by them. The basic aim of this petition was to ensure that the product remains alive. However, the petition did not attain its aim effectively.

1.1.2 The Importance of Visual Basic Programming Language

Visual Basic is regarded as the third generation event-drivon programming language. It was released in 1987. Being the first visual development tool from Microsoft, it is considered as one of the most powerful programming languages. As compared to other computer programming languages, such as, C, C++, it is easy to learn and understand, provided that one has determination and dedication to do so.

Visual basic programming language allows programmers to create software interface and codes in an easy to use graphical environment. VB is the combination of different components that are used on forms having specific attributes and actions with the help of those components. On the one hand it allows programmers to develop seldom based applications rapidly; on the other hand, it helps greatly in accessing data bases, using ADO while letting the programmers use ActiveX controls and various objects. While it is intended more to develop applications, it is also useful for games development for particular or limited purposes, unlike C++ that is more suitable for developing games.

As compared to other languages, Visual basic may be slower though, yet it is flexible and it can be rightly said that things that are difficult in other languages are comparatively easier in visual basic programming language. It may also be said that, since it is one of the most popular programming languages, lots of related books and material and other resources are available and can be accessed for developing programming skills at visual basic programming language conveniently.

One of the most important things to be considered with regard to programming in Visual basic is that the structure of VB is designed in a way that allows programmers to create executable code – Exe files. It enables programmers to develop programs that can be used as front end to databases. Besides, it's with the help of visual basic tools, one can change the abstract ideas into programs or into the whole software while it allows revising and modifying the programs fittingly.

ABOUT THIS CHAPTER

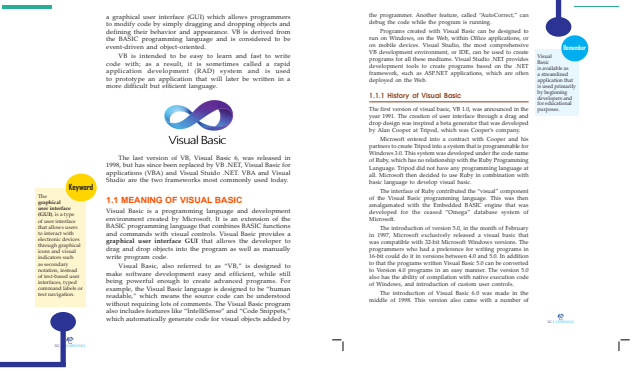
An introduction is a beginning of section which states the purpose and goals of the topics which are discussed in the chapter. It also starts the topics in brief.

REMEMBER

This revitalizes a must read information of the topic.

KEYWORDS

This section contains some important definitions that are discussed in the chapter. A keyword is an index entry that identifies a specific record or document. It also gives the extra information to the reader and an easy way to remember the word definition.


Keyword

The graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to of text-based user interfaces, typed command labels or text navigation.

a graphical user interface (GUI) which allows programmers to modify code by simply dragging and dropping objects and defining their behavior and appearance. VB is derived from the BASIC programming language and is considered to be event-driven and object-oriented.

VB is intended to be easy to learn and fast to write code with; as a result, it is sometimes called a rapid application development (RAD) system and is used to prototype an application that will later be written in a more difficult but efficient language.

Visual Basic

The last version of VB, Visual Basic 6, was released in 1998, but has since been replaced by VB .NET, Visual Basic for applications (VBA) and Visual Studio .NET. VBA and Visual Studio are the two frameworks most commonly used today.

1.1 MEANING OF VISUAL BASIC

Visual Basic is a programming language and development environment created by Microsoft. It is an extension of the BASIC programming language that combines BASIC functions and commands with visual controls. Visual Basic provides a **graphical user interface GUI** that allows the developer to drag and drop objects into the program as well as manually write program code.

Visual Basic, also referred to as "VB," is designed to make software development easy and efficient, while still being powerful enough to create advanced programs. For example, the Visual Basic language is designed to be "human readable," which means the source code can be understood without requiring lots of comments. The Visual Basic program also includes features like "IntelliSense" and "Code Snippets," which automatically generate code for visual objects added by

the programmer. Another feature, called "AutoCorrect," can debug the code while the program is running.

Programs created with Visual Basic can be designed to run on Windows, on the Web, within Office applications, or on mobile devices. Visual Studio, the most comprehensive VB development environment, or IDE, can be used to create programs for all these mediums. Visual Studio .NET provides development tools to create programs based on the .NET framework, such as ASP.NET applications, which are often deployed on the Web.

Remember

Visual Basic is available as a standalone application that is used primarily by beginning developers and for educational purposes.

1.1.1 History of Visual Basic

The first version of visual basic, VB 1.0, was announced in the year 1991. The creation of user interface through a drag and drop design was inspired a beta generator that was developed by Alan Cooper at Tripod, which was Cooper's company.

Microsoft entered into a contract with Cooper and his partners to create Tripod into a system that is programmable for Windows 3.0. This system was developed under the code name of Ruby, which has no relationship with the Ruby Programming Language. Tripod did not have any programming language at all. Microsoft then decided to use Ruby in combination with basic language to develop visual basic.

The interface of Ruby contributed the "visual" component of the Visual Basic programming language. This was then amalgamated with the Embedded BASIC engine that was developed for the ceased "Omega" database system of Microsoft.

The introduction of version 5.0, in the month of February in 1997, Microsoft exclusively released a visual basic that was compatible with 32-bit Microsoft Windows versions. The programmers who had a preference for writing programs in 16-bit could do it in versions between 4.0 and 5.0. In addition to that the programs written Visual Basic 5.0 can be converted to Version 4.0 programs in an easy manner. The version 5.0 also has the ability of compilation with native execution code of Windows, and introduction of custom user controls.

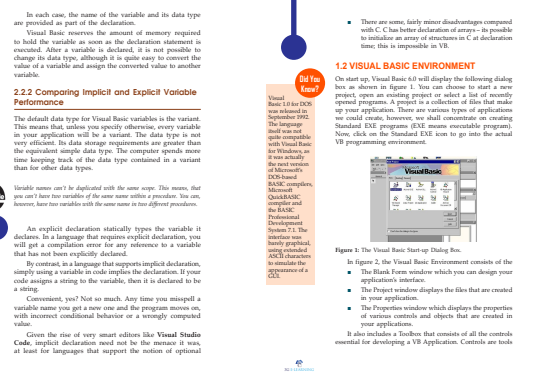
The introduction of Visual Basic 6.0 was made in the middle of 1998. This version also came with a number of

DID YOU KNOW?

This section equip readers the interesting facts and figures of the topic.

EXAMPLE

The book cabinets' examples to illustrate specific ideas in each chapter.



In each case, the name of the variable and its data type are provided as part of the declaration.

Visual Basic reserves the amount of memory required to hold the variable as soon as the declaration statement is executed. After a variable is declared, it is not possible to change its data type, although it is quite easy to convert the value of a variable and assign the converted value to another variable.

2.2.2 Comparing Implicit and Explicit Variable Performance

The default data type for Visual Basic variables is the variant. This means that, unless you specify otherwise, every variable in your application will be a variant. The data type is not very efficient. Its data storage requirements are greater than the equivalent simple data type. The computer spends more time keeping track of the data type contained in a variant than for other data types.

Variable names can't be duplicated with the same scope. This means, that you can't have two variables of the same name within a procedure. You can, however, have two variables with the same name in two different procedures.

An explicit declaration statically types the variable it declares. In a language that requires explicit declaration, you will get a compilation error for any reference to a variable that has not been explicitly declared.

By contrast, in a language that supports implicit declaration, simply using a variable in code implies the declaration. If your code assigns a string to the variable, then it is declared to be a string.

Convenient, yes? Not so much. Any time you misspell a variable name you get a new one and the program moves on, with incorrect conditional behavior or a wrongly computed value.

Given the rise of very smart editors like **Visual Studio Code**, implicit declaration need not be the menace it was, at least for languages that support the notion of optional

- There are some, fairly minor disadvantages compared with C. C has better declaration of arrays – its possible to initialize an array of structures in C at declaration time; this is impossible in VB.

1.2 VISUAL BASIC ENVIRONMENT

Did You Know?

Visual Basic 1.0 for DOS was released in September 1992. The language itself was not quite compatible with Visual Basic for Windows, as it was actually the next version of Microsoft's DOS-based BASIC compilers, Microsoft QuickBASIC compiler and the BASIC Professional Development System 7.1. The interface was barely graphical, using extended ASCII characters to simulate the appearance of a GUI.

On start up, Visual Basic 6.0 will display the following dialog box as shown in figure 1. You can choose to start a new project, open an existing project or select a list of recently opened programs. A project is a collection of files that make up your application. There are various types of applications we could create, however, we shall concentrate on creating Standard EXE programs (EXE means executable program). Now, click on the Standard EXE icon to go into the actual VB programming environment.

Figure 1: The Visual Basic Start-up Dialog Box.

In figure 2, the Visual Basic Environment consists of the

- The Blank Form window which you can design your application's interface.
- The Project window displays the files that are created in your application.
- The Properties window which displays the properties of various controls and objects that are created in your applications.

It also includes a Toolbox that consists of all the controls essential for developing a VB Application. Controls are tools

ROLE MODEL

A biography of someone who has/had acquired remarkable success in their respective field as Role Models are important because they give us the ability to imagine our future selves.

CASE STUDY

This reveals what students need to create and provide an opportunity for the development of key skills such as communication, group working and problem solving.

ROLE MODEL

ALAN COOPER: FATHER OF VISUAL BASIC

Born in San Francisco in 1952 and raised in Marin County, California, Alan Cooper has always taken the path less traveled. A rebellious teenager, he dropped out of high school, but eventually made his way to the College of Marin to pursue his interest in architecture. After an exploratory course in programming, it became clear that his future was in architecture—software architecture. After getting his associate degree and a COBOL programming job, he saw an advertisement for one of the first personal computers and .conceived an idea for a new business venture

In 1976, Cooper founded Structured Systems Group (SSG), a company Fire in the Valley authors Paul Freiberger and Michael Swaine said created "the first serious business software for microcomputers." In four years, Cooper wrote and shipped a dozen application programs. SSG became the archetype for many software startups in the early days of the personal computer revolution.

During the 1980s, after leaving SSG, Cooper invented, wrote, and sold three major software packages to prominent publishers. One of those was the visual programming front-end code named "Ruby," for what became Visual BASIC. Bill Gates purchased it from Cooper in 1988, noting that it would have significant impact across Microsoft's entire product line. Visual BASIC was deemed both a commercial and critical success, earning Cooper the moniker "Father of Visual BASIC." Visual BASIC has influenced integrated development languages ever since.

In 1990 Cooper became fascinated with the challenge of making software products that were easy to use and understand. He and his wife, Susan, founded Cooper Interaction Design (now "Cooper") to assist in what Cooper calls "interaction design." In the design field, Cooper's software development background was unique and, over the next few years, he invented many of the tools and techniques now standard in the user experience industry, including personas and scenarios,

CASE STUDY

FUJITSU FACILITATES SMOOTH MIGRATION TO VB.NET AT AN POST

Fujitsu has an excellent technical team, which works closely with our staff. We have had a good working relationship for many years and Fujitsu has an in-depth knowledge of our mission-critical application gained from several years' development and support work.

Challenge

A Post, one of Ireland's largest companies, is a major commercial organization providing a wide range of postal, communication, retail and financial services. With 9,600 employees throughout its national network of retail, processing and delivery points, the business also provides services to government departments, the National Treasury Management Agency and its own National Lottery Company. A decade ago, A Post implemented a new nationwide time and attendance system to calculate and record staff salary and wages functions. The Staff Remuneration and Administration Management System (STREAMS) is a bespoke, mission-critical application developed by Fujitsu as a reliable, scalable client server system using Microsoft technologies. The STREAMS front-end system gathers information and feeds the data to the company's HR, payroll and financial departments. It primarily creates more efficient processes for A Post to capture data for the weekly payroll run whilst simultaneously minimizing the number of payroll queries by employees. Following deployment, STREAMS improved cost center reporting, significantly lowered the time to record pay details and enhanced the processing of casual staff pay. During this period, Fujitsu provided quality support and maintenance services and application enhancements to increase functionality, ensuring the long-term reliability of STREAMS. For instance, as employee numbers steadily increased to exceed original expectations, Fujitsu boosted system performance by upgrading the infrastructure and optimizing the software. STREAMS originally employed Visual Basic (VB), a third-generation event-driven programming language and integrated development environment (IDE) from Microsoft. IDE provides programmers with comprehensive facilities for software development and comprises a source code editor, a compiler and/or an interpreter, build automation tools and a debugger. However, Microsoft no longer supports VB version 6.0, the edition employed by A Post. Syl Byrne, IT Manager Remuneration Services, A Post, explains: "To ensure that our business-critical application is future-proof, we needed to move to a platform that Microsoft will support for the foreseeable future." A Post therefore decided to migrate STREAMS to the VB.NET platform, an object-orientated programming language. This strategy would protect its investment for the next 10 years by creating a secure, scalable

KNOWLEDGE CHECK

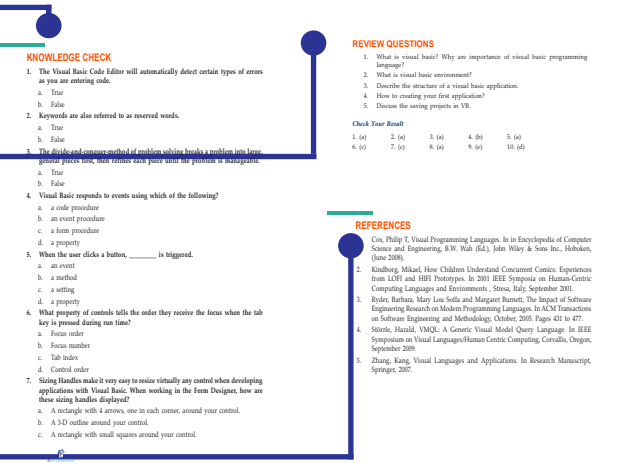
This is given to the students for progress check at the end of each chapter.

REVIEW QUESTIONS

This section is to analyze the knowledge and ability of the reader.

REFERENCES

References refer those books which discuss the topics given in the chapters in almost same manner.



KNOWLEDGE CHECK

1. **The Visual Basic Code Editor will automatically detect certain types of errors as you are entering code.**
   a. True
   b. False
2. **Keywords are also referred to as reserved words.**
   a. True
   b. False
3. **The divide-and-conquer method of problem solving breaks a problem into large general pieces first, then refines each piece until the problem is manageable.**
   a. True
   b. False
4. **Visual Basic responds to events using which of the following?**
   a. a code procedure
   b. an event procedure
   c. a form procedure
   d. a property
5. **When the user clicks a button, _____ is triggered.**
   a. an event
   b. a method
   c. a setting
   d. a property
6. **What property of controls tells the order they receive the focus when the tab key is pressed during run time?**
   a. Focus order
   b. Focus number
   c. Tab index
   d. Control order
7. **Sizing Handles make it very easy to resize virtually any control when developing applications with Visual Basic. When working in the Form Designer, how are these sizing handles displayed?**
   a. A rectangle with 4 arrows, one in each corner, around your control.
   b. A 3-D outline around your control.
   c. A rectangle with small squares around your control.

REVIEW QUESTIONS

1. What is visual basic? Why are importance of visual basic programming language?
2. What is visual basic environment?
3. Describe the structure of a visual basic application.
4. How to creating your first application?
5. Discuss the saving projects in VB.

Check Your Result

1. (a)	2. (a)	3. (a)	4. (b)	5. (a)
6. (c)	7. (c)	8. (a)	9. (e)	10. (d)

REFERENCES

1. Cox, Philip T, Visual Programming Languages. In in Encyclopedia of Computer Science and Engineering, B.W. Wah (Ed.), John Wiley & Sons Inc., Hoboken, (June 2008).
2. Kindborg, Mikael, How Children Understand Concurrent Comics: Experiences from LOFI and HIFI Prototypes. In 2001 IEEE Symposia on Human-Centric Computing Languages and Environments , Stresa, Italy, September 2001.
3. Ryder, Barbara, Mary Lou Soffa and Margaret Burnett, The Impact of Software Engineering Research on Modern Programming Languages. In ACM Transactions on Software Engineering and Methodology, October, 2005. Pages 431 to 477.
4. Störrle, Harald, VMQL: A Generic Visual Model Query Language. In IEEE Symposium on Visual Languages/Human Centric Computing, Corvallis, Oregon, September 2009.
5. Zhang, Kang, Visual Languages and Applications. In Research Manuscript, Springer, 2007.

TABLE OF CONTENTS

PREFACE

PHP is an open source, server-side, HTML embedded scripting language used to create dynamic web pages. However, there are many languages which are used for web development or web programming. But among all of them PHP is the most popular web scripting language. So, let us find out why PHP is widely used for web development. PHP language has its roots in C and C++. PHP syntax is most similar to C and C++ language syntax. So, programmers find it easy to learn and manipulate.

Organization of the Book

The 2nd edition of this book is comprised of nine chapters. The book includes basic through advanced functionality, this book explains how to develop dynamic Web applications, such as guest books, chat rooms, and shopping carts.

Chapter 1 covers essential background on the PHP language. It describes the nature and history of PHP, which platforms it runs on, and how to configure it. This chapter ends by showing you PHP in action, with a quick walkthrough of several PHP programs that illustrate common tasks, such as processing form data, interacting with a database, and creating graphics.

Chapter 2 shows you the data types, the PHP operators, the flow-control statements. PHP allows eight different types of data types. All of them are discussed in this chapter.

Chapter 3 discusses the functions in PHP and explains the CPU usage information. Functions are useful because they contribute to rapid, reliable, error-reducing coding, and increase legibility by tiding up complicated code sequences. A focus on serialization is given.

Chapter 4 focuses on PHP string functions and use of parsing in PHP. A PHP string is a sequence of characters i.e. used to store and manipulate text. String is one of the data types supported by PHP. The string variables can contain alphanumeric characters.

Chapter 5 talks about creating an array, adding and removing elements from an array, and looping over the contents of an array. Because arrays are very common and useful, there are many built-in functions that work with them in PHP.

Chapter 6 is intended to focus on PHP objects and classes. The basic building blocks of object-oriented programming are classes and objects. Understanding object and class and the differences between object and class is very important before we can dive into object-oriented programming approach.

Chapter 7 focuses on understanding the HTTP basics. The HTTP daemon in the destination server machine receives the request and sends back the requested file or files associated with the request. Therefore, all of the components that make up a site are interdependent on one another.

Chapter 8 focuses on database functions. It describes the PHP functions that communicate with various systems, it does not pursue introducing the intricacies of all the systems.

Chapter 9 explores on designing and debugging. Debugging tactics can involve interactive debugging, control flow analysis, unit testing, integration testing, log file analysis, monitoring at the application or system level, memory dumps, and profiling.

CHAPTER

1

INTRODUCTION TO PHP

"The main languages out of which web applications are built - whether it's Perl or Python or PHP or any of the other languages - those are all open source languages. So the infrastructure of the web is open source... the web as we know it is completely dependent on open source."

– Mitch Kapor

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand the PHP (hypertext preprocessor)
2. Learn about PHP installation

INTRODUCTION

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it

is used to develop web applications (an application that executes on the server and generates the dynamic page.).

Some important points need to be noticed about PHP are as followed:

- PHP stands for hypertext preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.

- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language to learn.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. For example - Registration form.

1.1 PHP (HYPERTEXT PREPROCESSOR)

PHP Stands for "Hypertext Preprocessor." (It is a recursive acronym, if you can understand what that means.) PHP is an HTML-embedded Web scripting language. This means PHP code can be inserted into the HTML of a Web page. When a PHP page is accessed, the PHP code is read or "parsed" by the server the page resides on. The output from the PHP functions on the page are typically returned as HTML code, which can be read by the browser. Because the PHP code is transformed into HTML before the page is loaded, users cannot view the PHP code on a page. This make PHP

pages secure enough to access databases and other secure information. A lot of the syntax of PHP is borrowed from other languages such as C, Java and Perl. However, PHP has a number of unique features and specific functions as well. The goal of the language is to allow Web developers to write dynamically generated pages quickly and easily. PHP is also great for creating database-driven Web sites. If you would like to learn more about PHP, the official site is PHP.net.

The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers. PHP, originally derived from *Personal Home Page* Tools, now stands for *PHP: Hypertext Preprocessor*, which the PHP FAQ describes as a "recursive acronym."

PHP executes on the server, while a comparable alternative, JavaScript, executes on the client. PHP is an alternative to Microsoft's Active Server Page (ASP) technology. As with ASP, the PHP script is embedded within a Web page along with its HTML. Before the page is sent to a user that has requested it, the Web server calls PHP to interpret and perform the operations called for in the PHP script.

An HTML page that includes a PHP script is typically given a file name suffix of ".php" ".php7," or ".phtml". Like ASP, PHP can be thought of as "dynamic HTML pages," since content will vary based on the results of interpreting the script.

PHP is an open-source language, used primarily for dynamic web content and server-side applications. PHP is often pointed to as the main competitor with the following:

- Microsoft's C# — Visual Basic.NET — ASP family
- Sun's Java — JSP
- Macromedia's ColdFusion
- CGI — Perl

PHP has many open-source libraries included with the core build, and many more are readily available. Extensions exist to help it **interface** with a number of systems, including IRC, a number of compression formats, and Windows API.

Did You Know?

PHP was developed by Rasmus Lerdorf in 1995, and later it has been developed as open source. The PHP Group now manages the implementation of PHP.

Keyword

Interface is a shared boundary across which two or more separate components of a computer system exchange information.

Other extensions exist to let PHP generate file formats on-the-fly, such as a popular extension that allows it to create Adobe Flash® movies.

Since version 3, PHP has integrated object oriented features. Version 5 built substantially on this limited functionality, and the language now has robust object oriented capabilities, including interfaces, exceptions, destructions, and abstracts.

PHP reached wide-spread popularity with version 4, released in 2000. In 2004, version 5 debuted, and it is now considered one the top languages used for server-side scripting.

No doubt much of its popularity is due to its relative ease to learn, and its notorious looseness. Arrays and variables in PHP are able to hold any type of object, variables need not be declared, and the syntax is remarkably simple.

Unlike many languages, such as C# or Perl, which have primarily a following of more generalist programmers, many PHP programmers know no other language. This occasionally causes it to be dismissed as a lesser language, but its growing popularity and the many robust and efficient sites built using it as a structure seem to dispel this myth.

PHP has occasionally been criticized for what are viewed by some as security flaws, in comparison to languages such as ASP. A lack of easily understandable error messages, a sometimes overly robust configuration file, and an obviously incomplete set of built-in functions are also pointed to as areas which could use marked improvement.

Development continues apace, however, and with each successive build, the language appears to address more and more of the concerns raised by its open-source community.

1.1.1 What Does PHP Do?

PHP can be used in three primary ways:

- Server-side scripting
- Command-line scripting
- Client-side GUI applications

Server-Side Scripting

PHP was originally designed to create dynamic web content, and it is still best suited for that task. To generate HTML, you need the PHP parser and a web server through which to send the coded documents. PHP has also become popular for generating XML documents, graphics, Flash animations, PDF files, and so much more.

Command-Line Scripting

PHP can run scripts from the command line, much like Perl, awk, or the Unix shell. You might use the command-line scripts for system administration tasks, such as backup and log parsing; even some CRON job type scripts can be done this way (nonvisual PHP tasks).

Client-Side GUI Applications

Using PHP-GTK, you can write full-blown, cross-platform GUI applications in PHP.

However, we concentrate on the first item: using PHP to develop dynamic web content.

PHP runs on all major operating systems, from Unix variants including Linux, FreeBSD, Ubuntu, Debian, and Solaris to Windows and Mac OS X. It can be used with all leading web servers, including Apache, Microsoft IIS, and the Netscape/iPlanet servers.

The language itself is extremely flexible.

You are not limited to outputting just HTML or other text files—any document format can be generated. PHP has built-in support for generating PDF files, GIF, JPEG, and PNG images, and Flash movies.

One of PHP's most significant features is its wide-ranging support for databases. PHP supports all major databases (including MySQL, PostgreSQL, Oracle, Sybase, MS-SQL, DB2, and ODBC-compliant databases), and even many obscure ones.

Even the more recent NoSQL-style databases like SQLite and MongoDB are also supported. With PHP, creating web pages with dynamic content from a database is remarkably simple.

Finally, PHP provides a library of PHP code to perform common tasks, such as database abstraction, error handling, and so on, with the PHP Extension and Application Repository (PEAR). PEAR is a framework and distribution system for reusable PHP components. You can find out more about it here.

1.1.2 Common Uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

1.1.3 Characteristics of PHP

Five important characteristics make PHP's practical nature possible –

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

1.1.4 "Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this –

```
<html>

   <head>
      <title>Hello World</title>
   </head>

   <body>
      <?php echo "Hello, World!";?>
   </body>
```

</html>

It will produce following result –

Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

<?php PHP code goes here ?>

<? PHP code goes here ?>

<script language = "php"> PHP code goes here </script>

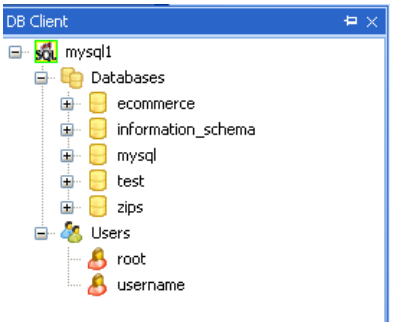
A most common tag is the <?php...?>

We will start with PHP Environment Setup on your machine and then we will dig out almost all concepts related to PHP to make you comfortable with the PHP language.

1.1.5 The History of PHP

PHP is an "HTML-embedded scripting language" primarily used for dynamic Web applications. The first part of this definition means that PHP code can be interspersed with HTML, making it simple to generate dynamic pieces of Web pages on the fly. As a scripting language, PHP code requires the presence of the PHP processor. PHP code is normally run in plain-text scripts that will only run on PHP-enabled computers (conversely programming languages can create standalone binary executable files, a.k.a. programs). PHP takes most of its syntax from C, Java, and Perl. It is an open source technology and runs on most operating systems and with most Web servers. PHP was written in the C programming language by Rasmus Lerdorf in 1994 for use in monitoring his online resume and related personal information. For this reason, PHP originally stood for "Personal Home Page". Lerdorf combined PHP with his own Form **Interpreter**, releasing the combination publicly as PHP/FI (generally referred to as PHP 2.0) on June 8, 1995. Two programmers, Zeev Suraski and Andi Gutmans, rebuilt PHP's core, releasing the updated result as PHP/FI 2 in 1997. The acronym was formally changed to PHP: HyperText Preprocessor, at this time. (This is an example of a recursive acronym: where the acronym itself is in its own definition.) In 1998, PHP 3 was released, which was the first widely used version. PHP 4 was released in May 2000, with a new core, known as the Zend Engine 1.0. PHP 4 featured improved

speed and reliability over PHP 3. In terms of features, PHP 4 added references, the Boolean type, COM support on Windows, output buffering, many new array functions, expanded object-oriented programming, inclusion of the PCRE library, and more. Maintenance releases of PHP 4 are still available, primarily for security updates.



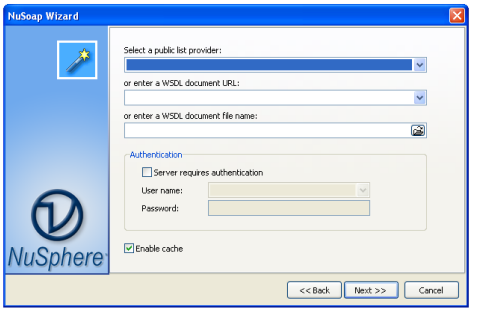
Keyword

Interpreter is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program.

PHP 5 was released in July 2004, with the updated Zend Engine 2.0. Among the many new features in PHP 5 are:

- improved object-oriented programming
- embedded SQLite
- support for new MySQL features (see the image at right)
- exception handling using a try..catch structure
- integrated SOAP support (see the image at right)
- the Filter library (in PHP 5.1)
- better XML tools
- iterators

and much, much more.



PHP 6 has been in development since October of 2006. The most significant change will be native support for Unicode. Unpopular, deprecated features such as Magic Quotes, register_globals, safe_mode, and the HTTP_*_VARS variables will disappear in PHP 6. Although PHP is still primarily used for server-side generation of Web pages, it can also be used to perform command-line scripting or to create graphical applications with the help of GTK+.

1.2 PHP INSTALLATION

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

- ***Web Server*** – PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.
- ***Database*** – PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
- ***PHP Parser*** – In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the **Web Browser**.

Keyword

Web browser is a software application for accessing information on the World Wide Web.

1.2.1 PHP Parser Installation

Before you proceed it is important to make sure that you have proper environment setup on your machine to develop your web programs using PHP.

Type the following address into your browser's address box.

http://127.0.0.1/info.php

If this displays a page showing your PHP installation related information then it means you have PHP and Webserver installed properly. Otherwise you have to follow given procedure to install PHP on your computer.

We will guide you to install and configure PHP over the following four platforms –

- PHP Installation on Linux or Unix with Apache

- PHP Installation on Mac OS X with Apache
- PHP Installation on Windows NT/2000/XP with IIS
- PHP Installation on Windows NT/2000/XP with Apache

PHP Installation on Linux or Unix with Apache

If you plan to install PHP on Linux or any other variant of Unix, then here is the list of prerequisites −

- The PHP source distribution http://www.php.net/downloads.php
- The latest Apache source distribution https://httpd.apache.org/download.cgi
- A working PHP-supported database, if you plan to use one (*For example* MySQL, Oracle etc.)
- Any other supported **software** to which PHP must connect (mail server, BCMath package, JDK, and so forth)
- An ANSI C compiler
- Gnu make utility − you can freely download it at https://www.gnu.org/software/make

Keyword

Software is a generic term that refers to a collection of data or computer instructions that tell the computer how to work, in contrast to the physical hardware from which the system is built, that actually performs the work.

Now here are the steps to install Apache and PHP5 on your Linux or Unix machine. If your PHP or Apache versions are different then please take care accordingly.

- If you haven't already done so, unzip and untar your Apache source distribution. Unless you have a reason to do otherwise, /usr/local is the standard place.

```
gunzip -c apache_1.3.x.tar.gz
tar -xvf apache_1.3.x.tar
```

- Build the apache Server as follows

```
cd apache_1.3.x
./configure --prefix=/usr/local/apache --enable-so
make
make install
```

- Unzip and untar your PHP source distribution. Unless you have a reason to do otherwise, /usr/local is the standard place.

gunzip -c php-5.x.tar.gz

tar -xvf php-5.x.tar

cd php-5.x

- Configure and Build your PHP, assuming you are using MySQL database.

./configure --with-apxs=/usr/sbin/apxs \

--with-mysql=/usr/bin/mysql

make

make install

- Install the php.ini file. Edit this file to get configuration directives –

cd ../../php-5.x

cp php.ini-dist /usr/local/lib/php.ini

- Tell your Apache server where you want to serve files from, and what extension(s) you want to identify PHP files A .php is the standard, but you can use .html, .phtml, or whatever you want.
  - Go to your HTTP configuration files (/usr/local/apache/conf or whatever your path is)
  - Open httpd.conf with a text editor.
  - Search for the word DocumentRoot (which should appear twice), and change both paths to the directory you want to serve files out of (in our case, /home/httpd). We recommend a home directory rather than the default /usr/local/apache/htdocs because it is more secure, but it doesn.t have to be in a home directory. You will keep all your PHP files in this directory.
- Add at least one PHP extension directive, as shown in the first line of code that follows. In the second line, we.ve also added a second handler to have all HTML files parsed as PHP.

AddType application/x-httpd-php .php

AddType application/x-httpd-php .html

- Restart your server. Every time you change your HTTP configuration or php. ini files, you must stop and start your server again.

cd ../bin

./apachectl start

- Set the document root directory permissions to world-executable. The actual PHP files in the directory need only be world-readable (644). If necessary, replace /home/httpd with your document root below –

chmod 755 /home/httpd/html/php

- Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php.
- Start any Web browser and browse the file.you must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly

You should see a long table of information about your new PHP installation message Congratulations!

PHP Installation on Mac OS X with Apache

Mac users have the choice of either a binary or a source installation. In fact, your OS X probably came with Apache and PHP preinstalled. This is likely to be quite an old build, and it probably lacks many of the less common extensions.

However, if all you want is a quick Apache + PHP + MySQL/PostgreSQL setup on your laptop, this is certainly the easiest way to fly. All you need to do is edit your Apache configuration file and turn on the Web server.

So just follow the following steps –

- Open the Apache config file in a text editor as root.

sudo open -a TextEdit /etc/httpd/httpd.conf

- Edit the file. Uncomment the following lines –

Load Module php5_module

AddModule mod_php5.c

AddType application/x-httpd-php .php

You may also want to uncomment the <Directory /home/*/Sites> block or otherwise tell Apache which directory to serve out of.

- Restart the Web server

sudo apachectl graceful

- Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php.
- Start any Web browser and browse the file.you must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or

http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly

You should see a long table of information about your new PHP installation message Congratulations!

PHP Installation on Windows with IIS

The Windows server installation of PHP running IIS is much simpler than on Unix, since it involves a precompiled binary .rather than a source build

If you plan to install PHP over Windows, then here is – the list of prerequisites

- A working PHP-supported Web server. Under previous versions of PHP, IIS/PWS was the easiest choice because a module version of PHP was available for it; but PHP now has added a much wider selection of modules for Windows.
- A correctly installed PHP-supported **database** like MySQL or Oracle etc. (if you plan to use one)
- The PHP Windows binary distribution (download it at www.php.net/downloads.php)
- A utility to unzip files (search http://download.cnet.com for PC file compression utilities)

Now here are the steps to install Apache and PHP5 on your Windows machine. If your PHP version is different then please take care accordingly.

- Extract the binary archive using your unzip utility; C:\PHP is a common location.
- Copy some .dll files from your PHP directory to your systems directory (usually C:\Winnt\System32). You need php5ts.dll for every case. You will also probably need to copy the file corresponding to your Web server module - C:\PHP\Sapi\php5isapi.dll. It's possible you will also need others from the dlls subfolder - but start with the two mentioned above and add more if you need them.
- Copy either php.ini-dist or php.ini-recommended (preferably the latter) to your Windows directory (C:\Winnt or C:\Winnt40), and rename it php.ini. Open

Keyword

Database is an organized collection of data, stored and accessed electronically.

this file in a text editor (*for example*, Notepad). Edit this file to get configuration directives; We highly recommend new users set error reporting to E_ALL on their development machines at this point. For now, the most important thing is the doc_root directive under the Paths and Directories section.make sure this matches your IIS Inetpub folder (or wherever you plan to serve out of).

- Stop and restart the WWW service. Go to the **Start menu → Settings → Control Panel ⚜ Services.** Scroll down the list to IIS Admin Service. Select it and click Stop. After it stops, select World Wide Web Publishing Service and click Start. Stopping and restarting the service from within Internet Service Manager will not suffice. Since this is Windows, you may also wish to reboot.
- Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php.
- Start any Web browser and browse the file.you must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly

You should see a long table of information about your new PHP installation message Congratulations!

PHP Installation on Windows with Apache

To install Apache with PHP 5 on Windows follow the following steps. If your PHP and Apache versions are different then please take care accordingly.

- Download Apache server from www.apache.org/dist/httpd/binaries/win32. You want the current stable release version with the no_src.msi extension. Double-click the installer file to install; C:\Program Files is a common location. The installer will also ask you whether you want to run Apache as a service or from the command line or DOS prompt. We recommend you do not install as a service, as this may cause problems with startup.
- Extract the PHP binary archive using your unzip utility; C:\PHP is a common location.
- Copy some .dll files from your PHP directory to your system directory (usually C:\Windows). You need php5ts.dll for every case. You will also probably need to copy the file corresponding to your Web server module - C:\PHP\Sapi\ php5apache.dll. to your Apache modules directory. It's possible that you will also need others from the dlls subfolder.but start with the two mentioned previously and add more if you need them.
- Copy either php.ini-dist or php.ini-recommended (preferably the latter) to your Windows directory, and rename it php.ini. Open this file in a text editor (*for example*, Notepad). Edit this file to get configuration directives; At this

point, we highly recommend that new users set error reporting to E_ALL on their development machines.

- Tell your Apache server where you want to serve files from and what extension(s) you want to identify PHP files (.php is the standard, but you can use .html, .phtml, or whatever you want). Go to your HTTP configuration files (C:\Program Files\Apache Group\Apache\conf or whatever your path is), and open httpd.conf with a text editor. Search for the word DocumentRoot (which should appear twice) and change both paths to the directory you want to serve files out of. (The default is C:\Program Files\Apache Group\Apache\htdocs.). Add at least one PHP extension directive as shown in the first line of the following code –

LoadModule php5_module modules/php5apache.dll

AddType application/x-httpd-php .php .phtml

- You may also need to add the following line –

AddModule mod_php5.c

- Stop and restart the WWW service. Go to the **Start menu → Settings → Control Panel → Services.** Scroll down the list to IIS Admin Service. Select it and click Stop. After it stops, select World Wide Web Publishing Service and click Start. Stopping and restarting the service from within Internet Service Manager will not suffice. Since this is Windows, you may also wish to reboot.
- Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php.
- Start any Web browser and browse the file.you must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly

You should see a long table of information about your new PHP installation message Congratulations!

1.2.2 Apache Configuration for PHP

Apache uses httpd.conf file for global settings, and the .htaccess file for per-directory access settings. Older versions of Apache split up httpd.conf into three files (access.conf, httpd.conf, and srm.conf), and some users still prefer this arrangement.

Apache server has a very powerful, but slightly complex, configuration system of its own. Learn more about it at the Apache Web site – www.apache.org

We describe settings in httpd.conf that affect PHP directly and cannot be set elsewhere. If you have standard installation then httpd.conf will be found at /etc/httpd/conf:

Timeout

This value sets the default number of seconds before any HTTP request will time out. If you set PHP's max_execution_time to longer than this value, PHP will keep grinding away but the user may see a 404 error. In safe mode, this value will be ignored; you must use the timeout value in php.ini instead

DocumentRoot

Remember

You can associate any file extension with PHP like .php3, .php5 or .htm.

DocumentRoot designates the root directory for all HTTP processes on that server. It looks something like this on Unix –

DocumentRoot ./usr/local/apache_1.3.6/htdocs.

You can choose any directory as document root.

AddType

The PHP MIME type needs to be set here for PHP files to be parsed.

AddType application/x-httpd-php .php

AddType application/x-httpd-phps .phps

AddType application/x-httpd-php3 .php3 .phtml

AddType application/x-httpd-php .html

Action

You must uncomment this line for the Windows apxs module version of Apache with shared object support –

LoadModule php4_module modules/php4apache.dll

or on Unix flavors –

LoadModule php4_module modules/mod_php.so

AddModule

You must uncomment this line for the static module version of Apache.

AddModule mod_php4.c

1.2.3 PHP.INI File Configuration

The PHP configuration file, php.ini, is the final and most immediate way to affect PHP's functionality. The php.ini file is read each time PHP is initialized.in other words, whenever httpd is restarted for the module version or with each script execution for the CGI version. If your change isn.t showing up, remember to stop and restart httpd. If it still isn.t showing up, use phpinfo() to check the path to php.ini.

The configuration file is well commented and thorough. Keys are case sensitive, keyword values are not; whitespace, and lines beginning with semicolons are ignored. Booleans can be represented by 1/0, Yes/No, On/Off, or True/False. The default values in php.ini-dist will result in a reasonable PHP installation that can be tweaked later.

Here we are explaining the important settings in php.ini which you may need for your PHP Parser.

short_open_tag = Off

Short open tags look like this: <? ?>. This option must be set to Off if you want to .use XML functions

safe_mode = Off

If this is set to On, you probably compiled PHP with the --enable-safe-mode flag. Safe mode is most relevant to CGI use.

safe_mode_exec_dir = [DIR]

This option is relevant only if safe mode is on; it can also be set with the --with-exec-dir flag during the Unix build process. PHP in safe mode only executes external binaries out of this directory. The default is /usr/local/bin. This has nothing to do with serving up a normal PHP/HTML Web page.

safe_mode_allowed_env_vars = [PHP_]

This option sets which environment variables users can change in safe mode. The default is only those variables prepended with "PHP_". If this directive is empty, most variables are alterable.

safe_mode_protected_env_vars = [LD_LIBRARY_PATH]

This option sets which environment variables users can't change in safe mode, even if safe_mode_allowed_env_vars is set permissively

disable_functions = [function1, function2...]

A welcome addition to PHP4 configuration and one perpetuated in PHP5 is the ability to disable selected functions for security reasons. Previously, this necessitated hand-editing the C code from which PHP was made. Filesystem, system, and network functions should probably be the first to go because allowing the capability to write files and alter the system over HTTP is never such a safe idea.

max_execution_time = 30

The function set_time_limit() won.t work in safe mode, so this is the main way to make a script time out in safe mode. In Windows, you have to abort based on maximum memory consumed rather than time. You can also use the Apache timeout setting to timeout if you use Apache, but that will apply to non-PHP files on the site too.

error_reporting = E_ALL & ~E_NOTICE

The default value is E_ALL & ~E_NOTICE, all errors except notices. Development servers should be set to at least the default; only production servers should even consider a lesser value

error_prepend_string = [“”]

With its bookend, error_append_string, this setting allows you to make error messages a different color than other text, or what have you.

warn_plus_overloading = Off

This setting issues a warning if the + operator is used with strings, as in a form value.

variables_order = EGPCS

This configuration setting supersedes gpc_order. Both are now deprecated along with register_globals. It sets the order of the different variables: Environment, GET, POST, COOKIE, and SERVER (aka Built-in).You can change this order around. Variables will be overwritten successively in left-to-right order, with the rightmost one winning the hand every time. This means if you left the default setting and happened to use the same name for an environment variable, a POST variable, and a COOKIE variable, the COOKIE variable would own that name at the end of the process. In real life, this does not happen much.

register_globals = Off

This setting allows you to decide whether you wish to register EGPCS variables as global. This is now deprecated, and as of PHP4.2, this flag is set to Off by default. Use superglobal arrays instead. All the major code listings in this book use superglobal arrays.

gpc_order = GPC

This setting has been GPC Deprecated.

magic_quotes_gpc = On

This setting escapes quotes in incoming GET/POST/COOKIE data. If you use a lot of forms which possibly submit to themselves or other forms and display form values, you may need to set this directive to On or prepare to use addslashes() on string-type data.

magic_quotes_runtime = Off

This setting escapes quotes in incoming database and text strings.

If this setting is Off, you will need to use stripslashes() when outputting any type of string data from a SQL database. If magic_quotes_sybase is set to On, this must be Off.

Remember

SQL adds slashes to single quotes and apostrophes when storing strings and does not strip them off when returning them.

magic_quotes_sybase = Off

This setting escapes single quotes in incoming database and text strings with Sybase-style single quotes rather than backslashes. If magic_quotes_runtime is set to On, this must be Off.

auto-prepend-file = [path/to/file]

If a path is specified here, PHP must automatically include() it at the beginning of every PHP file. Include path restrictions do apply.

auto-append-file = [path/to/file]

If a path is specified here, PHP must automatically include() it at the end of every PHP file.unless you escape by using the exit() function. Include path restrictions do apply.

include_path = [DIR]

If you set this value, you will only be allowed to include or require files from these directories. The include directory is generally under your document root; this is mandatory if you.re running in safe mode. Set this to . in order to include files from the same directory your script is in. Multiple directories are separated by colons: .:/usr/local/apache/htdocs:/usr/local/lib.

doc_root = [DIR]

If you are using Apache, you.ve already set a document root for this server or virtual host in httpd.conf. Set this value here if you.re using safe mode or if you want to enable PHP only on a portion of your site (*for example*, only in one subdirectory of your Web root).

file_uploads = [on/off]

Turn on this flag if you will upload files using PHP script.

upload_tmp_dir = [DIR]

Do not uncomment this line unless you understand the implications of HTTP uploads!

session.save-handler = files

Except in rare circumstances, you will not want to change this setting. So don't touch it.

ignore_user_abort = [On/Off]

This setting controls what happens if a site visitor clicks the browser.s Stop button. The default is On, which means that the script continues to run to completion or timeout. If the setting is changed to Off, the script will abort. This setting only works in module mode, not CGI.

mysql.default_host = hostname

The default server host to use when connecting to the database server if no other host is specified.

mysql.default_user = username

The default user name to use when connecting to the database server if no other name is specified.

mysql.default_password = password

The default password to use when connecting to the database server if no other password is specified.

1.2.4 Windows IIS Configuration

To configure IIS on your Windows machine you can refer your IIS Reference Manual shipped along with IIS.

ROLE MODEL

RASMUS LERDORF

Rasmus Lerdorf (born 22 November 1968) is a Danish-Canadian programmer. He created the PHP scripting language, authoring the first two versions of the language and participated in the development of later versions led by a group of developers including Jim Winstead (who later created blo.gs), Stig Bakken, Shane Caraveo, Andi Gutmans, and Zeev Suraski. He continues to contribute to the project.

Early Life and Education

Lerdorf was born in Disko Island on Greenland and moved to Denmark in his early years. Lerdorf's family moved to Canada from Denmark in 1980, and later moved to King City, Ontario in 1983. He graduated from King City Secondary School in 1988, and in 1993 he graduated from the University of Waterloo with a Bachelor of Applied Science in Systems Design Engineering. He contributed to the Apache HTTP Server and he added the LIMIT clause to the mSQL DBMS. A variant of this LIMIT clause had already been around for a decade in mainframe relational database management systems (like Oracle Rdb running on VAX/VMS, formerly from Digital Equipment Corporation), but apparently it had not yet been picked up by the emerging PC-based databases. It was later adapted by several other SQL-compatible DBMS. He released the first version of PHP in 1995.

Career

From September 2002 to November 2009 Lerdorf was employed by Yahoo! Inc. as an Infrastructure Architecture Engineer. In 2010, he joined WePay in order to develop their application programming interface. Throughout 2011 he was a roving consultant for startups. On 22 February 2012 he announced on Twitter that he had joined Etsy. In July, 2013 Rasmus joined Jelastic as a senior advisor to help them with the creation of new technology.

Lerdorf is a frequent speaker at Open Source conferences around the world. During his keynote presentation at OSCMS 2007, he presented a security vulnerability in each of the projects represented at the conference that year.

Lerdorf also made an appearance at the WeAreDevelopers Conference 2017, making a speech on the history of PHP and the new PHP 7 release in 2017.

Awards

In 2003, Lerdorf was named in the MIT Technology Review TR100 as one of the top 100 innovators in the world under the age of 35.

SUMMARY

- PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development.
- PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.
- The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.
- PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers.
- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- In order to develop and run PHP Web pages three vital components need to be installed on your computer system.
- The Windows server installation of PHP running IIS is much simpler than on UNIX, since it involves a precompiled binary rather than a source build.

KNOWLEDGE CHECK

1. **PHP Stands for**
   a. Php Hypertext Processor
   b. Php Hypertext Preprocessor
   c. Php Hypermarkup Preprocessor
   d. Php Hypermarkup Processor
2. **PHP is ..... scripting language.**
   a. Server-side
   b. Clint-side
   c. Middle-side
   d. Out-side
3. **PHP scripts are executed on .....**
   a. ISP Computer
   b. Client Computer
   c. Server Computer
   d. It depends on PHP scripts
4. **PHP Scripts starts with .....**
   a. <php> ... </php>
   b. <?php .... ?>
   c. ?php ... ?php
   d. ...
5. **Which of the following is correct about PHP?**
   a. PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
   b. PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
   c. You add, delete, modify elements within your database thru PHP.
   d. All of the above.
6. **Which of the following type of variables are special variables that hold references to resources external to PHP (such as database connections)?**
   a. Strings
   b. Arrays

c. Objects

d. Resources

7. Which of the following is correct about constants vs variables in PHP?

a. Constants may be defined and accessed anywhere without regard to variable scoping rules.

b. Once the constants have been set, may not be redefined or undefined.

c. Both (a) and (b).

d. None of the above.

REVIEW QUESTIONS

1. What is PHP?
2. What is the common usage of PHP?
3. Describe the characteristics of PHP.
4. Discuss about the history of PHP.
5. How many components are needed to be installed PHP web pages on your computer system?

Check Your Result

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

REFERENCES

1. A. Fayyaz and M. Madiha, "Performance Evaluation of PHP Frameworks (CakePHP and CodeIgniter) in relation to the Object-Relational Mapping, with respect to Load Testing," pp. 1–54, 2013.
2. Asghar MZ, Ahmad H, Ahmad S, Saqib SM, Ahmad B, Asghar MJ. Simplified Neural Network Design for Hand Written Digit Recognition. International Journal of Computer Science and Information Security. 2011 Jun 1;9(6):319.
3. Asghar MZ, Ahmad S, Marwat A, Kundi FM. Sentiment Analysis on YouTube: A Brief Survey. arXiv preprint arXiv:1511.09142. 2015 Nov 30.
4. Kundi FM, Asghar D, Zubair M. Lexicon-Based Sentiment Analysis in the Social Web. Journal of Basic and Applied Scientific Research. 2014;4(6):238-48.
5. N. Prokofyeva and V. Boltunova, "Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems," Procedia Comput. Sci., vol. 104, no. December 2016, pp. 51–56, 2016.
6. R. Das and D. Prasad Saikia, "Comparison of Procedural PHP with Codeigniter and Laravel Framework," Int. J. Curr. Trends Eng. Res. Sci. J. Impact Factor, vol. 2, no. 6, pp. 42–48, 2016.
7. R. F. Olanrewaju, T. Islam, and N. Ali, "An empirical study of the evolution of PHP MVC framework," in Lecture Notes in Electrical Engineering, 2015, vol. 315, pp. 399–410.
8. R. Semeteys, Technology innovation management review "Method for Qualification and Selection of Open Source Software," no. May 2008. Talent First Network, 2008.
9. V. Petrosyan, "A Guide to Popular PHP Frameworks for Beginners | SEJ," 2016.

CHAPTER

2

PHP LANGUAGE BASIC

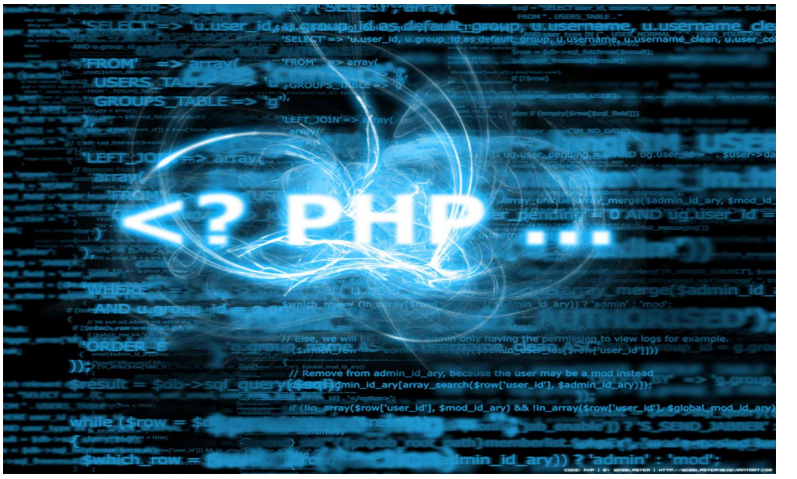
"PHP is a minor evil perpetrated and created by incompetent amateurs, whereas Perl is a great and insidious evil, perpetrated by skilled but perverted professionals."

– Jon Ribbens

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Discuss the data types
2. Describe the PHP operators
3. Explain the flow-control statements



INTRODUCTION

The PHP programming language is one of the most popular programming languages in the world. PHP is ranked as

the fifth most-searched language on Google. With this in mind, it is clear just how widespread the PHP programming language is around the web.

PHP is the most popular server-side scripting language. It is designed for web development and general purpose programming in 1994 by Rasmus Lerdorf. With over two decades of development, PHP has seen both the highs and lows. Now, PHP is managed by The PHP Group and is in continuous development. PHP stands for Hypertext Preprocessor which was changed from an initial name of "Personal Home Page".

PHP is mainly used in conjunction with HTML code, web content management system, web template systems, and other popular web frameworks. The PHP language is processed by the server or the Common Gateway Interface (CGI). Either way, it can be used to create an amazing web application where the PHP code is always executed on the server side. You can also execute PHP code with the help of command-line interface (CLI). Furthermore, it can also be used to implement a graphical application that is standalone in nature.

The great thing about the PHP programming language is that, in comparison to other languages, it is easy to learn. The language uses a simple syntax that is readable even for people without much technical experience. And, as a result, you should have no trouble getting started.

PHP, which stands for Hypertext Preprocessor, is a server-side scripting language that allows you to develop full stack web applications.

Because PHP is a server-side technology, it runs on the back end of a website. This is the part of a website that a user does not see. This means that PHP is often used to execute server-side scripts, such as gathering and processing data and working with databases.

The PHP language has a wide range of potential use cases.

These include:

- PHP can add content to a web page. This allows you to load data onto a website based on a database or another source of content.
- PHP can work with files on a server
- PHP can collect and process form data
- PHP allows you to implement login and registration pages on a website

2.1 DATA TYPES

Data Types defines the type of data a variable can store. PHP allows eight different types of data types. All of them are discussed below. The first five are called simple data types and the last three are compound data types:

Integer

Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or **decimal point**. They can be decimal (base 10), octal (base 8) or hexadecimal (base 16). The default base is decimal (base 10). The octal integers can be declared with leading 0 and the hexadecimal can be declared with leading 0x. The range of integers must lie between -2^31 to 2^31.

Keyword

Decimal Point- A point or dot used to separate the whole number part from the fractional part of a number.

Example:

```
<?php

// decimal base integers
$deci1 = 50;
$deci2 = 654;

// octal base integers
$octal1 = 07;

// hexadecimal base integers
$octal = 0x45;

$sum = $deci1 + $deci2;
echo $sum;

?>
```

Output:

704

Double

Can hold numbers containing fractional or decimal part including positive and negative numbers. By default, the variables add a minimum number of decimal places.

Example:

```
<?php

$val1 = 50.85;
$val2 = 654.26;

$sum = $val1 + $val2;

echo $sum;

?>
```

String

Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes but it will be treated differently while printing variables.

Example:

```
<?php

$name = "Jone";
echo "The name of the Geek is $name \n";
echo 'The name of the geek is $name';

?>
```

Output

```
The name of the Geek is Jone
The name of the geek is $name
```

NULL

These are special types of variables that can hold only one value i.e., NULL. We follow the convention of writing it in capital form, but its case sensitive.

Example:

```
<?php
```

```
$nm = NULL;
echo $nm;    // This will give no output

?>
```

Boolean

Hold only two values, either TRUE or FALSE. Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean. Apart from NULL, 0 is also consider as false in boolean. If a string is empty then it is also considered as false in **boolean data type**.

Example:

```
<?php

if(TRUE)
    echo "This condition is TRUE";
if(FALSE)
    echo "This condition is not TRUE";
?>
```

Keyword

The **Boolean data type** is a data type that has one of two possible values (usually denoted true and false), intended to represent the two truth values of logic and Boolean algebra.

Output:

This condition is TRUE
This condition is not TRUE

Arrays

Array is a compound data-type which can store multiple values of same data type. There is an example of array of integers.

```
<?php

$intArray = array( 10, 20 , 30);

echo "First Element: $intArray[0]\n";
echo "Second Element: $intArray[1]\n";
```

```
;"echo "Third Element: $intArray[2]\n

<?
```

Output:

```
First Element: 10
Second Element: 20
    Third Element: 30
```

Objects: Objects are defined as instances of user defined classes that can hold both values and functions.

Resources: Resources in PHP are not an exact data type. These are basically used to store references to some function call or to external PHP resources. For example, consider a database call. This is an external resource.

Did You Know?

HipHop – developed at Facebook and available as open source, it transforms the PHP scripts into C++ code and then compiles the resulting code, reducing the server load up to 50%. In early 2013, Facebook deprecated it in favor of HHVM due to multiple reasons, including deployment difficulties and lack of support for the whole PHP language, including the create_function() and eval() constructs.

2.2 PHP OPERATORS

An operator takes one or more values, which are known as operands, and performs operation on them such as adding them together. The following are the most common used operators in PHP:

- Arithmetic Operators
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Increment/Decrement Operators
- Logical Operators
- Concatenating Operators

2.2.1 Arithmetic Operators

The arithmetic operators require numeric values. And non-numeric values are converted automatically to numeric values. The following are the list of arithmetic operators:

- Addition (+) returns the sum of two operands
- Subtraction (-) returns the difference between two operands

- Multiplication (*) return the product of two operands
- Division (/) returns the quotient of two operands. The result of division of two integers may produce an integer (4/2 = 2) or a floating-point number (3/2 = 1.5)
- Modulus (%) returns the remainder of the division of the first operand by the second operand. Both operands are integers.

The following example demonstrates PHP arithmetic operators:

```
<?php

$x = 20;
$y = 10;

// add, subtract, and multiplication operators demo
echo $x + $y . '<br/>';  // 30
echo $x - $y . '<br/>';  // 10
echo $x * $y . '<br/>';  // 200

// division operator demo
$z = $x / $y;
echo gettype($z)  . '<br/>'; //  integer

$z = $y / $x;
echo gettype($z)  . '<br/>'; // double

// modulus demo

$y = 15;
echo $x % $y . '<br/>'; // 5
```

2.2.2 Assignment Operators

The main function of the assignment operators is to assign data value to the **variable**. The simplest assignment operator is equal sign (=) that just assigns some value. While the other assignment operators are shortcut assignment operators that make other operations before assignment operators are addition assignment (+=), multiplication

assignment (*=), division assignment (/=) and concatenation assignment (.=) operators.

```
php?>
    $num=10;
    $new_num=$num;
    echo "Simple Assignment Operator $new_num <br/>";
    $num+=5;
    echo "Addition Assignment Operator $num <br/>";
    $num*=2;
    echo "Multiplication Assignment Operator $num <br/>";
    $num/=3;
    echo "Division Assignment Operator $num <br/>";
    $num.=5;
    echo "Concatenation Assignment Operator $num <br/>";
    ?>
```

Keyword

A **variable** is a value that can change, depending on conditions or on information passed to the program.

:Output

```
Simple Assignment Operator 10
Addition Assignment Operator 15
Multiplication Assignment Operator 30
Division Assignment Operator 10
Concatenation Assignment Operator 105
```

2.2.3 String Operators

String operators in PHP provides two operators for concatenating the strings. Since, the first operator (.) called concatenation operator that concatenates two strings together. Whereas, the second operator (.=) called concatenation assignment operator that assigns the value after concatenating.

Here is an example showing the implementation of concatenation operator in order to concatenate two stings in PHP.

```
<?php
```

```
$lang1=”JavaScript”;

$lang2=”PHP”;

$lang=$lang1.” & “.$lang2;

echo $lang;

$out.=”are programming languages”;

echo $out;

?>
```

Output:

JavaScript & PHP


```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo “Hi, I’m a PHP script!”;
        ?>
    </body>
</html>
```

2.2.4 Bitwise Operators

Unlike to the other types of PHP operators, Bitwise operators examine and manipulate integer values in the level of individual

bits that make up the integer values. Since these types of operators converts integer numbers to binary numbers and performs logical operations then finally display them in integer numbers, thus the name bitwise operators. AND(&), OR(|), XOR(^), NOT(~), Shift left(<<) and Shift right(>>) are the most commonly used bitwise operators.

Here is an example that shows the uses of PHP bitwise AND(&) operator which displays the output of bitwise AND operation of two variables $a and $b.

```
<?php

$a=66;

$b=88;

$c=$a&$b;

echo "Bitwise AND on a and b is $c";

?>
```

Output:

Bitwise AND on a and b is 64

2.2.5 Comparison Operators

Since, logical operator provides the way to make decisions, comparison operator provides a method to direct program flow. Comparison operator compares the two or more values of the variables. Some of the most commonly used comparison operators are less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=) and ternary operator (? :).

Here is an example showing how comparison operators are used in PHP.

- Equality (==) returns true if both operands are equal, otherwise returns false.
- Identity(===) return true if both operands have the same data type and equal, otherwise return false.
- Inequality(!= or <>) returns true if both operands are not equal, otherwise return false.
- Not identical (!===) returns true if both operands are not equal or not have the same data type, otherwise returns false.

- Greater than (>) returns true if the operand on the left is greater than operand on the right, otherwise returns false.
- Greater than or equal (>=) returns true if the operand on the left is greater than or equal to operand on the right, otherwise returns false.
- Less than (<) returns true if the operand on the left is less than operand on the right, otherwise returns false.
- Less than or equal (<=) returns true if the operand on the left is less than or equal to operand on the right, otherwise returns false.

2.2.6 Increment and Decrement Operators

There are another sets of operators that can increase or decrease the value of the variable called increment (++) and decrement operators (–). The increment operator(++) increments the value by 1 whereas the decrement(–) operator decrements the value by 1.

In the following example a number is assigned to an integer variable and incremented the value after assigning to the first variable and the value of the variable assigned to the second variable after decreasing the value of an integer.

```
<?php
$int=45; // Assigns the integer value 45 to \$int

$int1=++$int;

$int2=$int–;

echo “The value of first integer is $int1<br/>”;

echo “The value of second integer is $int2<br/>”;

?>
```

Remember

In PHP, you can place these increment and decrement operators either side of the variables. It will provides slightly different effect while assigning the value with increment or decrement operators.

Output

The value of first integer is 46

The value of second integer is 46

2.2.7 Logical Operators

Since the logical operators provides the way to make decisions based on the values of multiple variables. These operators are most useful while crating PHP applications. So, it is possible to direct the flow of program with logical operators. Logical operators are mostly used with control structures such as if condition, while and for loop. Following are the list of logical operators in PHP and their uses.

- AND, && → And operators → True if both of the variables are True.
- OR, || → OR operators → True if at least one variable is True.
- NOT, ! → Not operator → True if the variable is not True.
- XOR → Exclusive OR → True if only one of the variable is True.

Following is an example that shows the uses of different logical operators used in PHP programming.

```
<?php
$a=5;

$b=6;

$c=7;

if (($a<$b) && ($b<$c)) {

echo "$c is the greatest number<br/>";

}

if ($a=5 || $b=5) {

echo "Either a of b is equal to 5<br/>";

}
```

```
if ($a!==$b) {

echo "a and b are not equal<br/>";

}

?>
```

Output

7 is the greatest number
Either a or b is equal to 5
a and b are not equal

2.2.8 Concatenating Operator

Concatenating operator (.) allows you to combine two strings into one. It appends the second string to the first string and returns the combined string.
Let's take a look at the following example:

```
<?php

$str = 'This is ' . ' the string #' . 1;
;echo $str
```

2.3 FLOW-CONTROL STATEMENTS

PHP supports a number of traditional programming constructs for controlling the flow of execution of a program.

Conditional statements, such as if/else and switch, allow a program to execute different pieces of code, or none at all, depending on some condition. Loops, such as while and for, support the repeated execution of particular code.

2.3.1 PHP if Statement

The if statement has the following general form:

(if (expression

statement

The if keyword is used to check if an expression is true. If it is true, a statement is then executed. The statement can be a single statement or a **compound statement**. A compound statement consists of multiple statements enclosed by curly brackets.

Keyword

Compound statements are statements using two or more logic operations. Creating a truth table is a systematic way of determining when a compound statement is true and when it is false.

ifstatement.php

```
<?php

$num = 31;

if ($num > 0)
    echo "\$num variable is positive\n";

?>
```

We have a $num variable. It is assigned value 31. The if keyword checks for a boolean expression. The expression is put between square brackets. The expression 31 > 0 is true, so the next statement is executed. Curly brackets are optional if there is only one statement to execute.

```
$ php ifstatement.php
$num variable is positive
```

This is the output of the example.

ifstatement2.php

```
<?php

$num = 31;

if ($num > 0) {
    echo "\$num variable is positive\n";
    echo "\$num variable equals to $num\n";
```

```
}

?>
```

If we intend to execute more than one statement, we have to put them inside square brackets. If we did not use them, only the first statement would be executed. Curly brackets form the *body* of the ifstatement.

We can use the else keyword to create a simple branch. If the expression inside the square brackets following the if keyword evaluates to false, the statement inside the else body is automatically executed.

boyorgirl.php

```
<?php

$sex = "female";

if ($sex == "male") {
   echo "It is a boy\n";
} else {
   echo "It is a girl\n";
}

?>
```

We have a $sex variable. It has "female" string. The boolean expression evaluates to false and we get "It is a girl" in the console.

```
$ php boyorgirl.php
It is a girl
```

This is the output of the example.

We can create multiple branches using the elseif keyword. The elseif keyword tests for another condition if and only if the previous condition was not met. Note that we can use multiple elseif keywords in our tests.

ifelsestm.php

```
<?php
```

```
echo "Enter a number: ";

$a = intval(fgets(STDIN));

if ($a < 0) {
    printf("%d is a negative number\n", $a);
} elseif ($a == 0) {
    printf("%d is a zero\n", $a);
} elseif ($a > 0) {
    printf("%d is a positive number\n", $a);
}

?>
```

We read a value from the user using the fgets() function. The value is tested if it is a negative number or positive or if it equals to zero.

```
$ php ifelsestm.php
Enter a number: 4
4 is a positive number
$ php ifelsestm.php
Enter a number: -3
-3 is a negative number
```

Sample output of the program.

2.3.2 PHP Switch Statement

The switch statement is a selection control flow statement. It allows the value of a variable or expression to control the flow of program execution via a multiway branch. It creates multiple branches in a simpler way than using the if, elseif statements. The switch statement works with two other keywords: case and break. The case keyword is used to test a label against a value from the round brackets. If the label equals to the value, the statement following the case is executed. The break keyword is used to jump out of the **switch statement**. There is an optional default statement. If none of the labels equals the value, the default statement is executed.

domains.php
php?>

```
$domain = 'sk';

switch ($domain) {

    case 'us':
        echo "United States\n";
    break;
    case 'de':
        echo "Germany\n";
    break;
    case 'sk':
        echo "Slovakia\n";
    break;
    case 'hu':
        echo "Hungary\n";
    break;
    default:
        echo "Unknown\n";
    break;
}

?>
```

Keyword

A **switch statement** is a type of selection control mechanism used to allow the value of a variable or expression to change the control flow of program execution via a multiway branch.

2.3.3 PHP while loop

The while is a control flow statement that allows code to be executed repeatedly based on a given boolean condition.

This is the general form of the while loop:

```
while (expression):
```

statement

The while loop executes the statement when the expression is evaluated to true. The statement is a simple statement terminated by a semicolon or a compound statement enclosed in curly brackets.

whilestm.php

```
<?php

$i = 0;

while ($i < 5) {
    echo "PHP language\n";
    $i++;
}

?>
```

In the code example, we repeatedly print "PHP language" string to the console.

The while loop has three parts: initialization, testing, and updating. Each execution of the statement is called a cycle.

```
$i = 0;
```

We initiate the $i variable. It is used as a counter in our script.

```
while ($i < 5) {
    ...
}
```

The expression inside the square brackets is the second phase, the testing. The while loop executes the statements in the body until the expression is evaluated to false.

```
$i++;
```

The last, third phase of the while loop is the updating; a counter is incremented. Note that improper handling of the while loop may lead to endless cycles.

```
$ php whilestm.php
PHP language
PHP language
PHP language
PHP language
PHP language
```

The program prints a message five times to the console.

The do while loop is a version of the while loop. The difference is that this version is guaranteed to run at least once.

dowhile.php

```
<?php

$count = 0;

do {
    echo "$count\n";
} while ($count != 0)

?>
```

First the iteration is executed and then the truth expression is evaluated.

The while loop is often used with the list() and each() functions.

seasons.php

```
<?php

$seasons = ["Spring", "Summer", "Autumn", "Winter"];

while (list($idx , $val) = each($seasons)) {
    echo "$val\n";
}
```

```
?>
```

We have four seasons in a $seasons array. We go through all the values and print them to the console. The each() function returns the current key and value pair from an array and advances the array cursor. When the function reaches the end of the array, it returns false and the loop is terminated. The each() function returns an array. There must be an array on the left side of the assignment too. We use the list() function to create an array from two variables.

```
$ php seasons.php
Spring
Summer
Autumn
Winter
```

This is the output of the seasons.php script.

2.3.4 PHP for Keyword

The for loop does the same thing as the while loop. Only it puts all three phases, initialization, testing and updating into one place, between the round brackets. It is mainly used when the number of iteration is know before entering the loop.

Let's have an example with the for loop.

forloop.php

```
<?php

$days = [ "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
          "Saturday", "Sunday" ];

$len = count($days);

for ($i = 0; $i < $len; $i++) {
    echo $days[$i], "\n";
}

?>
```

We have an array of days of a week. We want to print all these days from this array.

```
$len = count($days);
```

Or we can programmatically figure out the number of items in an array.

```
for ($i = 0; $i < $len; $i++) {
   echo $days[$i], "\n";
}
```

Here we have the for loop construct. The three phases are divided by semicolons. First, the $i counter is initiated. The initiation part takes place only once. Next, the test is conducted. If the result of the test is true, the statement is executed. Finally, the counter is incremented. This is one cycle. The for loop iterates until the test expression is false.

```
$ php forloop.php
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

This is the output of the forloop.php script.

2.3.5 PHP foreach Statement

The foreach construct simplifies traversing over collections of data. It has no explicit counter. The foreach statement goes through the array one by one and the current value is copied to a variable defined in the construct. In PHP, we can use it to traverse over an array.

foreachstm.php

```
<?php

$planets = [ "Mercury", "Venus", "Earth", "Mars", "Jupiter",
```

```
                                    "Saturn", "Uranus", "Neptune" ];

    foreach ($planets as $item) {
        echo "$item ";
    }

    echo "\n";

    ?>
```

Keyword

The **foreach statement** executes a statement or a block of statements for each element in an instance of the type that implements the System

In this example, we use the foreach statement to go through an array of planets.

```
foreach ($planets as $item) {
    echo "$item ";
}
```

The usage of the foreach statement is straightforward. The $planets is the array that we iterate through. The $item is the temporary variable that has the current value from the array. The **foreach statement** goes through all the planets and prints them to the console.

Remember

Keep always in mind that client-side validation is never enough. Every validation we do in the client-side, it should be done also in the server - remember that here is the person in the client-side who has the control, not you.

```
$ php foreachstm.php
Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune
```

Running the above PHP script gives this output.

There is another syntax of the foreach statement. It is used with maps.

foreachstm2.php

```
<?php

$benelux =  [ 'be' => 'Belgium',
              'lu' => 'Luxembourgh',
              'nl' => 'Netherlands' ];
```

```
foreach ($benelux as $key => $value) {
    echo "$key is $value\n";
}

?>
```

In our script, we have a $benelux map. It contains domain names mapped to the benelux states. We traverse the array and print both keys and their values to the console.

```
$ php foreachstm2.php
be is Belgium
lu is Luxembourgh
nl is Netherlands
```

This is the outcome of the script.

2.3.6 PHP Break, Continue Statements

The break statement is used to terminate the loop. The continue statement is used to skip a part of the loop and continue with the next iteration of the loop.

```
testbreak.php
php?>

} (while (true

;(val = rand(1, 30$
;" " ,echo $val
;if ($val == 22) break
{

;"echo "\n

<?
```

We define an endless while loop. There is only one way to jump out of a such

loop—using the break statement. We choose a random value from 1 to 30 and print .it. If the value equals to 22, we finish the endless while loop

```
$ php testbreak.php
6 11 13 5 5 21 9 1 21 22
```

We might get something like this.

In the following example, we print a list of numbers that cannot be divided by 2 without a remainder.

testcontinue.php

```
<?php

$num = 0;

while ($num < 1000) {

    $num++;
    if (($num % 2) == 0) continue;

    echo "$num ";

}

echo "\n";

?>
```

We iterate through numbers 1..999 with the while loop.

```
if (($num % 2) == 0) continue;
```

If the expression $num % 2 returns 0, the number in question can be divided by 2. The continue statement is executed and the rest of the cycle is skipped. In our case, the last statement of the loop is skipped and the number is not printed to the console. The next iteration is started.

SUMMARY

- The PHP programming language is one of the most popular programming languages in the world. PHP is ranked as the fifth most-searched language on Google.
- PHP is mainly used in conjunction with HTML code, web content management system, web template systems, and other popular web frameworks.
- The great thing about the PHP programming language is that, in comparison to other languages, it is easy to learn.
- A scripting language is a form of programming language that is usually interpreted rather than compiled.
- An operator takes one or more values, which are known as operands, and performs operation on them such as adding them together.
- PHP supports a number of traditional programming constructs for controlling the flow of execution of a program.
- The switch statement is a selection control flow statement. It allows the value of a variable or expression to control the flow of program execution via a multiway branch.
- The while is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition.
- The break statement is used to terminate the loop. The continue statement is used to skip a part of the loop and continue with the next iteration of the loop.

KNOWLEDGE CHECK

1. PHP is an example of …….. scripting language.

a. Server-side
b. Client-side
c. Browser-side
d. In-side

2. Which logical operator has higher precedence from following?

a. &&
b. !
c. ||
d. xor

3. Which operators have higher precedence?

a. Arithmetic operators
b. Comparison operators
c. Logical operators
d. Boolean operators

4. Decrement operator is denoted by

a. ++
b. --
c. %%
d. //

5. Comparison operators have higher precedence than

a. Arithmetic operators
b. Assignment operators
c. Boolean operators
d. All of them

6. Which of the following is the default file extension of PHP?

.php
.hphp
.xml
.html

7. Which of the following is not a variable scope in PHP?

Local

Extern

Static

Global

REVIEW QUESTIONS

1. What is data type?
2. How to use arithmetic operators in php?
3. What is the function of a string operator? Discuss.
4. Explain the logical operators.
5. Discuss about the flow-control statements.

Check Your Result

1. (a) 2. (b) 3. (a) 4. (b) 5. (b)
6. (a) 7. (b)

REFERENCES

1. D. Letarte and E. Merlo, "Extraction of inter-procedural simple role privilege models from php code," IEEE Computer Society, p. 187–191, 2009. https://doi.org/10.1109/ wcre.2009.32
2. D. Letarte, F. Gauthier and E. Merlo, "Security Model Evolution of PHP Web Applications," Fourth IEEE International Conference on Software Testing, Verification and Validation, pp. 290-298, 2011. https://doi.org/10.1109/ICST.2011.36
3. D. S. Team, "Security advisories," 8 November 2016. [Online]. Available: https://www.drupal.org/SA-CORE-2014-005.
4. E. Merlo, D. Letarte and G. Antoniol, "SQL-Injection Security Evolution Analysis in PHP," IEEE, pp. 45-49, 2007. https://doi.org/10.1109/wse.2007.4380243.
5. I. Herraiz, D. Rodriguez, G. Robles and J. M. Gonzalez-Barahona, "The Evolution of the Laws of Software Evolution: A Discussion Based on a Systematic Literature Review," ACM, pp. 28:1-28:28, 2013.
6. J. Tulach, "Practical API Design: Confessions of a Java Framework," Apress, 2008.
7. J. Walden, M. Doyle, R. Lenhof and J. Murray, "Java vs. PHP: Security Implications of Language Choice for Web Applications," in Engineering Secure Software and Systems, Berlin , Springer Heidelberg, 2010, pp. 61-69
8. Kellanved, "3.0.6 CAPTCHA plugins and you," 7 November 2016. [Online]. Available: https://blog.phpbb.com/2009/06/27/3-0-6-captcha-plugins-and-you/.
9. L. Eshkevar, F. Dos Santos, J. R. Cordy and G. Antoniol, "Are PHP Applications Ready for Hack?," IEEE, pp. 63-72, 2015. https://doi.org/10.1109/saner.2015.7081816.
10. M. Hills and P. Klint, "PHP AiR: Analyzing PHP Systems with Rascal," IEEE, pp. 454- 457, 2014.
11. P. Kyriakakis and A. Chatzigeorgiou, "Maintenance Patterns of large-scale PHP Web Applications," IEEE International Conference on Software Maintenance and Evolution, pp. 381-390, 2014.
12. S. Bergmann and S. Priebsch, Real-World Solutions for Developing High-Quality PHP Frameworks and Applications, Wiley, 2011.

CHAPTER

3

PHP FUNCTION

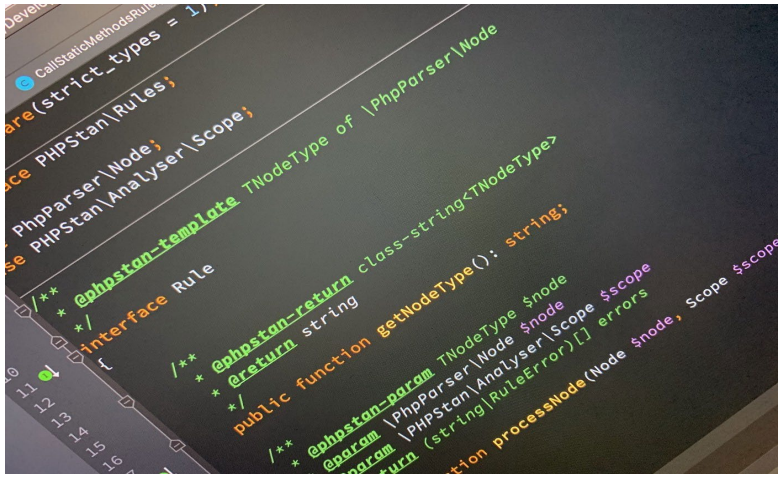
"Social media websites are no longer performing an envisaged function of creating a positive communication link among friends, family and professionals. It is a veritable battleground, where insults fly from the human quiver, damaging lives, destroying self-esteem and a person's sense of self-worth"

– Anthony Carmona

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Discuss the functions in PHP
2. Define the CPU usage information
3. Understand the concept of serialization



INTRODUCTION

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value.

There are thousands of built-in functions in PHP. PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

In addition to the built-in functions, PHP also allows you to define your own functions. It is a way to create reusable code packages that perform specific tasks and can be kept and maintained separately form main program.

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it. Among other things, this can be used to implement callbacks, function tables, and so forth.

Variable functions will not work with language constructs such as echo, print, unset(), isset(), empty(), include, require and the like. Utilize wrapper functions to make use of any of these constructs as variable functions

3.1 FUNCTIONS IN PHP

A function is a piece of code in a larger program. The function performs a specific task. The advantages of using functions are:

- Reducing duplication of code
- Decomposing complex problems into simpler pieces
- Improving clarity of the code
- Reuse of code
- Information hiding

There are two basic types of functions. Built-in functions and user defined ones. The built-in functions are part of the PHP language. Examples are: phpinfo(), round() or abs(). The user defined functions are created by application programmers to cover their needs. They are created with the function keyword.

3.1.1 PHP defining functions

A function is created with the function keyword.

simple.php

```
<?php

function displayVersion() {

    echo “this is PHP “ . phpversion();
```

```
    echo "\n";
}

displayVersion();

?>
```

The function keyword is followed by the function name with round brackets. The body of the function lies between the curly brackets. We say that we *call* a function. If we call a function, the statements inside the function body are executed.

displayVersion();

This line of the code calls the function.

```
$ php simple.php
this is PHP 5.5.9-1ubuntu4.14
```

Here we see the outcome of the script.

3.1.2 PHP Return keyword

The return keyword is used to return a value from the function. A function may or may not return a value.

returning.php

```
<?php

function maximum($x, $y) {

    if ($x > $y) {

        return $x;
    } else {

        return $y;
    }
}

$a = 23;
```

```
$b = 32;

$val = maximum($a, $b);
echo "The max of $a and $b is $val \n";

?>
```

We have a maximum () function. It returns a max for two numbers. We could not name it max, because there is already a built-in max () function. This example was created for learning purposes; we would always prefer the built-in functions in our real-world programs.

```
} (if ($x > $y

;return $x
} else {

;return $y
{
```

.If the $x variable is greater than $y, we return $x. Otherwise we return $y

```
;(val = maximum ($a, $b$
```

.The value returned by the maximum () function is assigned to the $val variable

```
;"echo "The max of $a and $b is $val \n
```

.We print the max value of the two numbers to the console

3.1.3 PHP function arguments

Most functions accept arguments. Arguments are values that are sent to the function. The functions process the values and possibly return some outcome.

fahrenheit.php

```
<?php

function FTC($c) {

    return $c * 9/5 + 32;
}
```

```
echo FTC(100);
echo "\n";
echo FTC(0);
echo "\n";
echo FTC(30);
echo "\n";

?>
```

In our example, we convert Fahrenheit temperature to Celsius. The FTC () function accepts one argument $c, which is the Celsius temperature.

```
$ php fahrenheit.php
212
32
86
```

3.1.4 PHP implicit values

The arguments in PHP functions may have implicit values. An implicit value is used if no value is provided.

implicit_value.php

```
<?php

function power($a, $b=2) {

    if ($b == 2) {

        return $a * $a;
    }

    $value = 1;

    for ($i = 0; $i < $b; $i++) {

        $value *= $a;
```

```
{

;return $value
{

    $v1 = power(5);
    $v2 = power(5, 4);

    echo "5^2 is $v1 \n";
    echo "5^4 is $v2 \n";

    ?>
```

Here we created a power function. The function has one argument with an implicit value. We can call the function with one or two arguments.

```
$ php implicit_value.php
5^2 is 25
5^4 is 625
```

3.1.5 PHP variable number of arguments

A function may accept variable number of arguments. In other words, sometimes we do not know how many arguments will be passed to the function. The func_get_args() function returns an array comprising a function's argument list.

Since PHP 5.6 there is the ... operator available to create variadic functions.

variable_arguments1.php

```
<?php

function sum(...$nums) {

    $sum = 0;

    foreach ($nums as $n) {
        $sum += $n;
    }
```

```
        return $sum;
    }

;“echo sum (1, 2, 3) . “\n
;“echo sum (1, 2, 3, 4) . “\n
;“echo sum (1, 2, 3, 4, 5) . “\n

<?
```

We create a sum () method which can take variable number of arguments. The .method calculates the sum of values passed to the method

```
;sum = 0$
} (foreach ($args as $n
;sum += $n$
{

;return $sum
```

.We calculate and return the sum of the values

```
;“echo sum (1, 2, 3) . “\n
;“echo sum (1, 2, 3, 4) . “\n
;“echo sum (1, 2, 3, 4, 5) . “\n
```

.We pass three, four, and five values to the sum() function

```
php variable_arguments1.php $
6
10
15
```

.This is the output

```
variable_arguments2.php
php?>

} ()function sum

;()args = func_get_args$
```

```
;sum = 0$
} (foreach ($args as $n
;sum += $n$
{

        return $sum;
    }

    echo sum(1, 2, 3) . "\n";
    echo sum(1, 2, 3, 4) . "\n";
    echo sum(1, 2, 3, 4, 5) . "\n";

    ?>
```

Now the same example is created with the func_get_args() function.

3.1.6 PHP static variables

A ***static variable*** is a variable that has been allocated statically, whose lifetime extends across the entire run of the program. The default local variables do not retain their value within consecutive calls of the function.

Keyword

Static Variable is a variable that has been allocated "statically", meaning that its lifetime (or "extent") is the entire run of the program.

```
non_static.php
<?php

function nonstatic() {

    $value = 0;
    $value += 1;

    return $value;
}
```

```
nonstatic();
nonstatic();
nonstatic();
nonstatic();

echo nonstatic(), "\n";

?>
```

In the above example, we have a normal, non-static variable. We increment the variable each time the function is called. We call the function 5 times. However, non-static variables are initiated for each call of the function. After 5 function calls the $value equals to 2.

The static variables are initiated only once, when the function is first called. They retain their value afterwards.

static.php

```
<?php

function staticfun() {

    static $value = 0;
    $value += 1;

    return $value;
}

staticfun();
staticfun();
staticfun();
staticfun();

echo staticfun(), "\n";
```

Remember

Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols.

```
?>
```

After 5 consecutive calls, the $value is equal to 5.

```
$ php nonstatic.php
2
$ php static.php
6
```

3.1.7 PHP anonymous functions

Anonymous functions do not have a name.

anonymous.php

```
<?php

$var = function() {

    echo "This is anonymous function\n";
};

$var();

?>
```

We assign a function body to a variable. It is possible to call the function only via this variable.

```
$var = function () {

    echo "This is anonymous function\n";
};
```

Notice the semicolon after the closing curly bracket. It is required because the construct is an assignment.

```
$ php anonymous.php
This is anonymous function
```

This is the output of the example.

Remember

A function name must start with a letter or underscore character not with a number, optionally followed by the more letters, numbers, or underscore characters. Function names are case-insensitive.

3.1.8 Passing arguments by value and by reference

PHP supports two ways of passing arguments to functions. The default way is passing arguments by value. When we pass arguments by value, the function works only with the copies of the values. This may lead to performance overheads when we work with large amounts of data.

When we pass values by reference, the function receives a reference to the actual value. The original values are affected when modified. This way of passing values is more time and space efficient. On the other hand, it is more error prone.

Which way of passing arguments should we use? It depends on the situation. Say we have a set of data, salaries of employees. If we want to compute some statistics of the data, we do not need to modify them. We pass by values. If we work with large amounts of data and the speed of computation is critical, we pass by reference. If we want to modify the data, e.g. do some reductions or raises to the salaries, we might pass by reference.

The following two examples cover both concepts.

swap1.php

```
<?php

function swap($a, $b) {

    $temp = $a;
    $a = $b;
    $b = $temp;
    echo "inside swap function:\n";
    echo "\$a is $a \n";
    echo "\$b is $b \n";
}

$a = 4;
$b = 7;
```

Did You Know?

In 1993, the National Center for Supercomputing Applications (NCSA), a unit of the University of Illinois at Urbana-Champaign, released NCSA Mosaic, the first popular graphical Web browser, which played an important part in expanding the growth of the nascent World Wide Web.

```
echo "outside swap function:\n";
echo "\$a is $a \n";
echo "\$b is $b \n";

swap($a, $b);

echo "outside swap function:\n";
echo "\$a is $a \n";
echo "\$b is $b \n";

?>
```

The swap function swaps the numbers between the $a and $b **variables**. The original variables are not affected.

```
$a = 4;
$b = 7;
```

At the beginning, these two variables are initiated.

```
swap($a, $b);
```

We call the swap() function. The function takes $a and $b variables as parameters.

```
$temp = $a;
$a = $b;
$b = $temp;
```

Inside the swap() function, we change the values. Note that the $a and $b variables are defined locally. They are valid only inside the swap() function.

```
$ php swap1.php
outside swap function:
$a is 4
$b is 7
inside swap function:
$a is 7
$b is 4
outside swap function:
$a is 4
```

Keyword

Variable is a storage location (identified by a memory address) paired with an associated symbolic name (an identifier), which contains some known or unknown quantity of information referred to as a value.

```
$b is 7
```

The output shows that the original variables were not affected.

The next code example passes values to the function by reference. The original variables are changed inside the swap() function.

swap2.php

```
<?php

function swap(&$a, &$b) {

    $temp = $a;
    $a = $b;
    $b = $temp;
    echo "Inside the swap function:\n";
    echo "\$a is $a \n";
    echo "\$b is $b \n";
}

$a = 4;
$b = 7;

echo "Outside the swap function:\n";
echo "\$a is $a \n";
echo "\$b is $b \n";

swap($a, $b);

echo "Outside the swap function:\n";
echo "\$a is $a \n";
echo "\$b is $b \n";

?>
```

We use the & character to pass values by reference.

```
function swap(&$a, &$b) {
```

```
...
}
```

The example is almost identical to the previous one. Except for the ampersand characters.

```
$ php swap2.php
Outside the swap function:
$a is 4
$b is 7
Inside the swap function:
$a is 7
$b is 4
Outside the swap function:
$a is 7
$b is 4
```

Here we see that the swap() function really changed the values of the variables.

3.1.9 PHP function recursion

Keyword

Factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n.

Recursion, in mathematics and computer science, is a method of defining functions in which the function being defined is applied within its own definition. In other words, a recursive function calls itself to do its job. Recursion is an alternative to iteration. Recursion is the main approach in functional languages.

A typical example is the calculation of the **factorial**.

recursion.php

```
<?php

function factorial($n) {

    if ($n==0) {

        return 1;
    } else {
```

```
        return $n * factorial ($n - 1);
    }
}
echo factorial (4), "\n";
echo factorial (10), "\n";

?>
```

In this code example, we calculate the factorial of two numbers.

return $n * factorial ($n - 1);

Inside the body of the factorial function, we call the factorial function with a modified argument. The function calls itself.

```
$ php recursion.php
24
3628800
```

These are the results.

So what does the syntax for defining a function look like? Here is a simple example:


```
<?php
function addNumbers($num1, $num2) {
    $result = $num1 + $num2;
    return $result;
}
```

3.1.10 PHP global and local variables

Next we will talk about the scope of the variables in PHP. A *scope* is the range in which a variable can be referenced. When we work with functions, there are two basic scopes: the global and the local scope. The local scope is also called a function scope.

scope1.php

```
<?php

$value = 1;

function simple() {

    var_dump($value);
}

simple();

?>
```

A variable defined outside a function body cannot be referenced within a function.

```
$ php scope1.php
PHP Notice:  Undefined variable: value in /home/janbodnar/prog/php/functions/
scope1.php on line 7
NULL
```

The $value variable is NULL in the simple() function.

scope2.php

```
<?php

$value = 4;

function simple() {
    $value = 3;
    echo $value, "\n";
}

simple();
echo $value, "\n";

?>
```

In this example, we have two variables with the same name. They do not collide because they exist in different scopes.

```
$ php scope2.php
3
4
```

The value is 3 inside the function and 4 outside the function.

In the next example, we will modify a value inside the function.

scope3.php

```
<?php

$value = 1;
function simple() {

    global $value;
    $value = 2;
}

echo $value, "\n";
simple();
echo $value, "\n";

?>
```

We use the global keyword to reference a variable defined outside the body of the function.

```
$ php scope3.php
1
2
```

The $value was successfully modified inside the simple () function.

In this part of the PHP tutorial, we covered PHP functions.

3.2 CPU USAGE INFORMATION

We are going to utilize the getrusage() function. Keep in mind that this is not available on Windows platforms.

```
print_r(getrusage());
/* prints
Array
(
    [ru_oublock] => 0
    [ru_inblock] => 0
    [ru_msgsnd] => 2
    [ru_msgrcv] => 3
    [ru_maxrss] => 12692
    [ru_ixrss] => 764
    [ru_idrss] => 3864
    [ru_minflt] => 94
    [ru_majflt] => 0
    [ru_nsignals] => 1
    [ru_nvcsw] => 67
    [ru_nivcsw] => 4
    [ru_nswap] => 0
    [ru_utime.tv_usec] => 0
    [ru_utime.tv_sec] => 0
    [ru_stime.tv_usec] => 6269
    [ru_stime.tv_sec] => 0
)

*/
```

That may look a bit cryptic unless you already have a system administration background. Here is the explanation of each value (you don't need to memorize these):

- ru_oublock: block output operations
- ru_inblock: block input operations
- ru_msgsnd: messages sent
- ru_msgrcv: messages received
- ru_maxrss: maximum resident set size
- ru_ixrss: integral shared memory size

- ru_idrss: integral unshared data size
- ru_minflt: page reclaims
- ru_majflt: page faults
- ru_nsignals: signals received
- ru_nvcsw: voluntary context switches
- ru_nivcsw: involuntary context switches
- ru_nswap: swaps
- ru_utime.tv_usec: user time used (microseconds)
- ru_utime.tv_sec: user time used (seconds)
- ru_stime.tv_usec: system time used (microseconds)
- ru_stime.tv_sec: system time used (seconds)

To see how much CPU power the script has consumed, we need to look at the 'user time' and 'system time' values. The seconds and **microseconds** portions are provided separately by default. You can divide the microseconds value by 1 million, and add it to the seconds value, to get the total seconds as a decimal number.

Keyword

Microsecond is an SI unit of time equal to one millionth (0.000001 or 10–6 or 1⁄1,000,000) of a second.

Let's see an example:

```
// sleep for 3 seconds (non-busy)
sleep(3);

$data = getrusage();
echo "User time: ".
    ($data['ru_utime.tv_sec'] +
    $data['ru_utime.tv_usec'] / 1000000);
echo "System time: ".
    ($data['ru_stime.tv_sec'] +
    $data['ru_stime.tv_usec'] / 1000000);

/* prints
User time: 0.011552
System time: 0
*/
```

Even though the script took about 3 seconds to run, the CPU usage was very very low. Because during the sleep operation, the script actually does not consume CPU resources. There are many other tasks that may take real time, but may not use CPU time, like waiting for disk operations. So as you see, the CPU usage and the actual length of the runtime are not always the same.

Here is another example:

```
// loop 10 million times (busy)
for($i=0;$i<10000000;$i++) {

}

$data = getrusage();
echo "User time: ".
    ($data['ru_utime.tv_sec'] +
    $data['ru_utime.tv_usec'] / 1000000);
echo "System time: ".
    ($data['ru_stime.tv_sec'] +
    $data['ru_stime.tv_usec'] / 1000000);

/* prints
User time: 1.424592
System time: 0.004204
*/
```

That took about 1.4 seconds of CPU time, almost all of which was user time, since there were no system calls.

System Time is the amount of time the CPU spends performing system calls for the kernel on the program's behalf. Here is an example of that:

```
$start = microtime(true);
// keep calling microtime for about 3 seconds
while(microtime(true) - $start < 3) {

}
```

```
$data = getrusage();
echo "User time: ".
    ($data['ru_utime.tv_sec'] +
    $data['ru_utime.tv_usec'] / 1000000);
echo "System time: ".
    ($data['ru_stime.tv_sec'] +
    $data['ru_stime.tv_usec'] / 1000000);

/* prints
User time: 1.088171
System time: 1.675315
*/
```

Now we have quite a bit of system time usage. This is because the script calls the microtime() function many times, which performs a request through the operating system to fetch the time.

Also you may notice the numbers do not quite add up to 3 seconds. This is because there were probably other processes on the server as well, and the script was not using 100% CPU for the whole duration of the 3 seconds.

3.3 SERIALIZATION

Have you ever needed to store a complex variable in a database or a text file? You do not have to come up with a fancy solution to convert your arrays or objects into formatted strings, as PHP already has functions for this purpose.

There are two popular methods of serializing variables. Here is an example that uses the serialize () and unserialize():

```
// a complex array
$myvar = array(
    'hello',
    42,
    array(1,'two'),
    'apple'
);
```

```
convert to a string //
    $string = serialize($myvar);

    echo $string;
    /* prints
    a:4:{i:0;s:5:"hello";i:1;i:42;i:2;a:2:{i:0;i:1;i:1;s:3:"two";}i:3;s:5:"apple";}
    */

    // you can reproduce the original variable
    $newvar = unserialize($string);

    print_r($newvar);
    /* prints
    Array
    (
        [0] => hello
        [1] => 42
        [2] => Array
            (
                [0] => 1
                [1] => two
            )

        [3] => apple
    )
    */
```

This was the native PHP serialization method. However, since JSON has become so popular in recent years, they decided to add support for it in PHP 5.2. Now you can use the json_encode() and json_decode() functions as well:

```
    // a complex array
    $myvar = array(
        'hello',
        42,
```

```
    array(1,'two'),
    'apple'
);

// convert to a string
$string = json_encode($myvar);

echo $string;
/* prints
["hello",42,[1,"two"],"apple"]
*/

// you can reproduce the original variable
$newvar = json_decode($string);

print_r($newvar);
/* prints
Array
(
    [0] => hello
    [1] => 42
    [2] => Array
        (
            [0] => 1
            [1] => two
        )

    [3] => apple
)
*/
```

It is more compact, and best of all, compatible with **javascript** and many other languages. However, for complex objects, some information may be lost.

Keyword

JavaScript often abbreviated as JS, is a high-level, interpreted programming language.

ROLE MODEL

ANDI GUTMANS

Andi Gutmans is an Israeli programmer and entrepreneur. He helped co-create PHP and co-founded Zend Technologies and is a General Manager at Amazon Web Services. A graduate of the Technion, the Israel Institute of Technology in Haifa, Gutmans and fellow student Zeev Suraski created PHP 3 in 1997. In 1999 they wrote the Zend Engine, the core of PHP 4, and founded Zend Technologies, which has since overseen PHP advances, including the PHP 5 and most recent PHP 7 releases. The name Zend is a portmanteau of their forenames, Zeev and Andi.

Gutmans served as CEO of Zend Technologies until October 2015 when Zend was acquired by Rogue Wave Software. Before being appointed CEO in February 2009, he led Zend's R&D including development of all Zend products and Zend's contributions to the open-source Zend Framework and PHP Development Tools projects. He has participated at Zend in its corporate financing and has also led alliances with vendors like Adobe, IBM, Microsoft, and Oracle.

Gutmans served on the board of the Eclipse Foundation (October 2005 - October 2008), is an emeritus member of the Apache Software Foundation, and was nominated for the FSF Award for the Advancement of Free Software in 1999.

In 2004 he wrote a book called "PHP 5 Power Programming" together with Stig Bakken and Derick Rethans.

Gutmans was recognized by ComputerWorld magazine in July 2007 in their article "40 Under 40: 40 Innovative IT People to Watch, Under the Age of 40."

In March 2016, Gutmans left Rogue Wave to join Amazon Web Services. Explaining his motivations, Gutmans cited "Cloud infrastructure adoption is at a tipping point" and "the data 'center of gravity' is moving to the cloud", where Amazon "appears to effectively balance innovation and invention: a focus on customer value with a bias to action". In his role at Amazon Web Services, Gutmans manages Amazon Elasticsearch Service, Amazon CloudSearch, Amazon ElastiCache and Amazon Neptune.

SUMMARY

- PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.
- PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- There are two basic types of functions. Built-in functions and user defined ones.
- Recursion is an alternative to iteration. Recursion is the main approach in functional languages.
- To see how much CPU power the script has consumed, we need to look at the 'user time' and 'system time' values.
- Inside the body of the factorial function, we call the factorial function with a modified argument. The function calls itself.
- The static variables are initiated only once, when the function is first called.

KNOWLEDGE CHECK

1. **Which one of the following is the right way of defining a function in PHP?**
   a. function {function body}
   b. data type functionName(parameters) {function body}
   c. functionName(parameters) {function body}
   d. function fumctionName(parameters) {function body}
2. **Type Hinting was introduced in which version of PHP?**
   a. PHP 4
   b. PHP 5
   c. PHP 5.3
   d. PHP 6
3. **What will happen in this function call?**

```
<?php
function calc($price, $tax)
{
    $total = $price + $tax;
}
$pricetag = 15;
$taxtag = 3;
calc($pricetag, $taxtag);
?>
```

   a. Call By Value
   b. Call By Reference
   c. Default Argument Value
   d. Type Hinting
4. **What will be the output of the following PHP code?**

```
<?php
function calc($price, $tax="")
{
    $total = $price + ($price * $tax);
    echo "$total";
}
calc(42);
```

?>

a. Error
b. 0
c. 42
d. 84

5. Which of the following are valid function names?

i) function ()
ii) €()
iii). function ()
iv) $function ()

a. Only ii)
b. None of the mentioned.
c. All of the mentioned.
d. iii) and iv)

6. How to define a function in PHP?

a. function {function body}
b. data type functionName(parameters) {function body}
c. function functionName(parameters) {function body}
d. functionName(parameters) {function body}

7. Why should we use functions?

a. Reusability
b. Easier error detection
c. Easily maintained
d. All of the above

REVIEW QUESTIONS

1. Write a program create a user defined function in PHP.
2. Write a program passing arrays to functions.
3. Write a program passing function parameters by reference.
4. Write a program use of default parameters in functions.
5. Write a program using non-scalar types as default values.

Check Your Result

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

REFERENCES

1. Artzi et al. Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking. IEEE Trans. on Soft. Eng., 36(4), 2010.
2. D. Balzarotti et al. Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications. S&P'2008, 2008.
3. D. Dhurjati, M. Das, and Y. Yang. Path-sensitive dataflow analysis with iterative refinement. In Static Analysis, LNCS. Springer, 2006.
4. F. Yu, M. Alkhalaf, and T. Bultan. Stranger: An automata-based string analysis tool for PHP. TACAS'10, 2010.
5. G. Balakrishnan, S. Sankaranarayanan, F. Ivancic, O. Wei, and A. Gupta. Slr: Path-sensitive analysis through infeasible-path detection and syntactic language refinement. In Static Analysis, LNCS. Springer, 2008.
6. G. Snelting, T. Robschink, and J. Krinke. Efficient path conditions in dependence graphs for software safety analysis. ACM Trans. Softw. Eng. Methodol., 2006.
7. G. Wassermann and Z. Su. Sound and precise analysis of web applications for injection vulnerabilities. PLDI '07. ACM, 2007.
8. G. Wassermann and Z. Su. Static detection of cross-site scripting vulnerabilities. ICSE '08, 2008.
9. I. Dillig, T. Dillig, and A. Aiken. Sound, complete and scalable pathsensitive analysis. In PLDI '08. ACM, 2008.
10. J. Fonseca, M. Vieira, and H. Madeira. Testing and comparing web vulnerability scanning tools for sql injection and xss attacks. In PRDC'07. IEEE CS, 2007.
11. M. Das, S. Lerner, and M. Seigle. Esp: path-sensitive program verification in polynomial time. In PLDI '02. ACM, 2002.
12. M. Taghdiri, G. Snelting, and C. Sinz. Information flow analysis via path condition refinement. In FAST 2010. Springer-Verlag, 2010.
13. M. Vieira, N. Antunes, and H. Madeira. Using web security scanners to detect vulnerabilities in web services. In DSN '09, 2009.
14. N. Jovanovic, C. Kruegel, and E. Kirda. Pixy: a static analysis tool for detecting Web application vulnerabilities. In S&P'06. IEEE, 2006.
15. P. Biggar and D. Gregg. Static analysis of dynamic scripting languages, 2009.
16. Y. Minamide. Static approximation of dynamically generated web pages. In WWW '05. ACM, 2005.
17. Y. Xie and A. Aiken. Static detection of security vulnerabilities I scripting languages. In 15th USENIX Security Symposium, 2006.
18. Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo. Securing web application code by static analysis and runtime protection. In WWW '04, 2004.

CHAPTER

4

PHP: STRING

"String Theory describes energy and matter as being composed of tiny, wiggling strands of energy that look like strings. And the pitch of a string's vibration determines the nature of its effect."

– Roy H. Williams

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Define the basics of PHP string functions
2. Know the use of parsing in PHP

INTRODUCTION

Just like any other programming language, Strings in PHP are also a collection of alphanumeric characters, enclosed in single quotes for simple string data and double quotes for complex string data.

PHP string is a sequence of characters i.e., used to store and manipulate text. PHP supports only 256-character set and so that it does not offer native Unicode support. There are 4 ways to specify a string literal in PHP.

We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP. For specifying a literal single quote, escape it with a backslash (\) and to specify a literal backslash (\) use double backslash (\\). All the other instances with backslash such as \r or \n, will be output same as they specified instead of having any special meaning.

4.1 BASIC PHP STRING FUNCTIONS

A PHP string is a sequence of characters i.e. used to store and manipulate text. A string is a collection of characters. String is one of the data types supported by PHP. The string variables can contain alphanumeric characters. Strings are created when;

- You declare variable and assign string characters to it
- You can directly use them with echo statement.
- String are language construct, it helps capture words.
- Learning how strings work in PHP and how to manipulate them will make you a very effective and productive developer.

They are sequences of characters, like “PHP supports string operations”.

Built-in string functions is given in function reference PHP String Functions

Following are valid examples of string

$string_1 = “This is a string in double quotes”;

$string_2 = “This is a somewhat longer, singly quoted string”;

$string_39 = “This string has thirty-nine characters”;

$string_0 = “”; // a string with zero characters

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php
   $variable = “name”;
   $literally = ‘My $variable will not print!\\n’;

   print($literally);
   print “<br />”;
```

```
$literally = "My $variable will print!\\n";

print($literally);
?>
```

This will produce the following result –

My $variable will not print!\n

My name will print!\n

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP –

Certain character sequences beginning with backslash (\) are replaced with special characters

Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are –

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

A string literal can be specified in four different ways:

- Single quoted
- Double quoted
- Heredoc syntax
- Nowdoc syntax (since PHP 5.3.0)

4.1.1 Single Quoted

The simplest way to specify a string is to enclose it in single quotes (the character '). To specify a literal single quote, escape it with a backslash (\). To specify a literal backslash, double it (\\). All other instances of backslash will be treated as a literal backslash: this means that the other escape sequences you might be used to, such as \r or \n, will be output literally as specified rather than having any special meaning.

Unlike the double-quoted and heredoc syntaxes, variables and escape sequences for special characters will *not* be expanded when they occur in single quoted strings.

```
<?php
echo ‹this is a simple string›;

echo ‹You can also have embedded newlines in
strings this way as it is
okay to do›;

// Outputs: Arnold once said: “I’ll be back”
echo ‹Arnold once said: «I\›ll be back»›;

// Outputs: You deleted C:\*.*?
echo ‹You deleted C:\\*.*?›;

// Outputs: You deleted C:\*.*?
echo ‹You deleted C:\*.*?›;

// Outputs: This will not expand: \n a newline
echo ‹This will not expand: \n a newline›;

// Outputs: Variables do not $expand $either
echo ‹Variables do not $expand $either›;
?>
```

4.1.2 Double quoted

If the string is enclosed in double-quotes (“), PHP will interpret the following escape sequences for special characters:

Escaped characters	
Sequence	**Meaning**
\n	linefeed (LF or 0x0A (10) in ASCII)
\r	carriage return (CR or 0x0D (13) in ASCII)

\t	horizontal tab (HT or 0x09 (9) in ASCII)
\v	vertical tab (VT or 0x0B (11) in ASCII) (since PHP 5.2.5)
\e	escape (ESC or 0x1B (27) in ASCII) (since PHP 5.4.4)
\f	form feed (FF or 0x0C (12) in ASCII) (since PHP 5.2.5)
\\	backslash
\$	dollar sign
\"	double-quote
\[0-7]{1,3}	the sequence of characters matching the regular expression is a character in octal notation, which silently overflows to fit in a byte (e.g. "\400" === "\000")
\x[0-9A-Fa-f]{1,2}	the sequence of characters matching the regular expression is a character in hexadecimal notation
\u{[0-9A-Fa-f]+}	the sequence of characters matching the regular expression is a Unicode codepoint, which will be output to the string as that codepoint's UTF-8 representation (added in PHP 7.0.0)

As in single quoted strings, escaping any other character will result in the backslash being printed too. Before PHP 5.1.1, the backslash in *\{$var}* had not been printed.

The most important feature of double-quoted strings is .the fact that variable names will be expanded

4.1.3 Heredoc

A third way to delimit strings is the heredoc syntax: <<<. After this operator, an **identifier** is provided, then a newline. The string itself follows, and then the same identifier again to close the quotation.

The closing identifier *must* begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

It is very important to note that the line with the closing identifier must contain no other characters, except a semicolon (;). That means especially that the identifier *may not be indented*,

Keyword

Identifier is a name that identifies (that is, labels the identity of) either a unique object or a unique class of objects, where the "object" or class may be an idea, physical [countable] object (or class thereof), or physical [noncountable] substance (or class thereof).

and there may not be any spaces or tabs before or after the semicolon. It's also important to realize that the first character before the closing identifier must be a newline as defined by the local operating system. This is \n on UNIX systems, including macOS. The closing delimiter must also be followed by a newline.

If this rule is broken and the closing identifier is not "clean", it will not be considered a closing identifier, and PHP will continue looking for one. If a proper closing identifier is not found before the end of the current file, a parse error will result at the last line.

Example #1. Invalid example

```
<?php
class foo {
    public $bar = <<<EOT
bar
;EOT
}
// Identifier must not be indented
?>
```

Example #2. Valid example

```
<?php
class foo {
    public $bar = <<<EOT
bar
EOT;
}
?>
```

Heredoc text behaves just like a double-quoted string, without the double quotes. This means that quotes in a heredoc do not need to be escaped, but the escape codes listed above can still be used. Variables are expanded, but the same care must be taken when expressing complex variables inside a heredoc as with strings.

Example #3. Heredoc string quoting example

```
<?php
```

Remember

Heredocs can not be used for initializing class properties. Since PHP 5.3, this limitation is valid only for heredocs containing variables.

```
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo
{
    var $foo;
    var $bar;

    function __construct()
    {
        $this->foo = ‹Foo›;
        $this->bar = array(‘Bar1’, ‘Bar2’, ‘Bar3’);
    }
}

$foo = new foo();
$name = ‘MyName’;

echo <<<EOT
My name is “$name”. I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital ‘A’: \x41
EOT;
?>
```

The above example will output:

```
My name is “MyName”. I am printing some Foo.
Now, I am printing some Bar2.
This should print a capital ‘A’: A
```

It is also possible to use the Heredoc syntax to pass data to function arguments:

Example #4. Heredoc in arguments example

```
<?php
var_dump(array(<<<EOD
foobar!
EOD
));
?>
```

As of PHP 5.3.0, it's possible to initialize static variables and class properties/constants using the Heredoc syntax:

Example #5. Using Heredoc to initialize static values

```
<?php
// Static variables
function foo()
{
    static $bar = <<<LABEL
Nothing in here...
LABEL;
}

// Class properties/constants
class foo
{
    const BAR = <<<FOOBAR
Constant example
FOOBAR;

    public $baz = <<<FOOBAR
Property example
FOOBAR;
}
?>
```

Starting with PHP 5.3.0, the opening Heredoc identifier may optionally be enclosed in double quotes:

Example #6 Using double quotes in Heredoc

```
<?php
echo <<<»FOOBAR»
Hello World!
FOOBAR;
?>
```

4.1.4 Nowdoc

Nowdocs are to single-quoted strings what heredocs are to double-quoted strings. A nowdoc is specified similarly to a heredoc, but *no parsing is done* inside a nowdoc. The construct is ideal for embedding PHP code or other large blocks of text without the need for escaping. It shares some features in common with the SGML *<![CDATA[ ]]>*construct, in that it declares a block of text which is not for parsing.

A nowdoc is identified with the same <<< sequence used for heredocs, but the identifier which follows is enclosed in single quotes, e.g. *<<<'EOT'*. All the rules for heredoc identifiers also apply to nowdoc identifiers, especially those regarding the appearance of the closing identifier.

Example #7. Nowdoc string quoting example

```
<?php
$str = <<<'EOD'
Example of string
spanning multiple lines
using nowdoc syntax.
EOD;

/* More complex example, with variables. */
class foo
{
    public $foo;
    public $bar;

    function __construct()
    {
        $this->foo = 'Foo';
```

```
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$name = 'MyName';

echo <<<'EOT'
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should not print a capital 'A': \x41
EOT;
?>
```

The above example will output:

```
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should not print a capital 'A': \x41
```

Example #8. Static data example

```
<?php
class foo {
    public $bar = <<<'EOT'
bar
EOT;
}
?>
```

4.2 PARSING IN PHP

To parse, in computer science, is where a string of commands – usually a program – is separated into more easily processed components, which are analyzed for correct syntax and then attached to tags that define each component. The computer can then process each program chunk and transform it into machine language.

To parse is to break up a sentence or group of words into separate components, including the definition of each part's function or form. The technical definition implies the same concept.

Parsing is used in all **high-level programming languages**. Languages like C++ and Java are parsed by their respective compilers before being transformed into executable machine code. Scripting languages, like PHP and Perl, are parsed by a .web server, allowing the correct HTML to be sent to a browser

4.2.1 Variable parsing

When a string is specified in double quotes or with heredoc, variables are parsed within it.

There are two types of syntax: a simple one and a complex one. The simple syntax is the most common and convenient. It provides a way to embed a variable, an array value, or an object property in a string with a minimum of effort.

The complex syntax can be recognized by the curly braces surrounding the expression.

If a dollar sign ($) is encountered, the parser will greedily take as many tokens as possible to form a valid variable name. Enclose the variable name in curly braces to explicitly specify the end of the name.

```
<?php
$juice = “apple”;


echo “He drank some $juice juice.”.PHP_EOL;
// Invalid. “s” is a valid character for a variable name, but the variable is $juice.
echo “He drank some juice made of $juices.”;
// Valid. Explicitly specify the end of the variable name by enclosing it in braces:
echo “He drank some juice made of ${juice}s.”;
?>
```

The above example will output:

He drank some apple juice.

He drank some juice made of .

He drank some juice made of apples.

Similarly, an array index or an object property can be parsed. With array indices, the closing square bracket (*]*) marks

Keyword

High-level programming language is a programming language with strong abstraction from the details of the computer. In contrast to low-level programming languages, it may use natural language elements, be easier to use, or may automate (or even hide entirely) significant areas of computing systems (e.g. memory management), making the process of developing a program simpler and more understandable than when using a lower-level language.

the end of the index. The same rules apply to object properties as to simple variables.

Example #9. Simple syntax example

```
<?php
$juices = array("apple", "orange", "koolaid1" => "purple");

echo "He drank some $juices[0] juice.".PHP_EOL;
echo "He drank some $juices[1] juice.".PHP_EOL;
echo "He drank some $juices[koolaid1] juice.".PHP_EOL;

class people {
    public $john = "John Smith";
    public $jane = "Jane Smith";
    public $robert = "Robert Paulsen";

    public $smith = "Smith";
}

$people = new people();

echo "$people->john drank some $juices[0] juice.".PHP_EOL;
echo "$people->john then said hello to $people->jane.".PHP_EOL;
echo "$people->john's wife greeted $people->robert.".PHP_EOL;
echo "$people->robert greeted the two $people->smiths."; // Won't work
?>
```

The above example will output:

```
He drank some apple juice.
He drank some orange juice.
He drank some purple juice.
John Smith drank some apple juice.
John Smith then said hello to Jane Smith.
John Smith's wife greeted Robert Paulsen.
Robert Paulsen greeted the two.
```

As of PHP 7.1.0 also *negative* numeric indices are supported.

Example #10. Negative numeric indices

```
<?php
$string = 'string';
echo "The character at index -2 is $string[-2].", PHP_EOL;
$string[-3] = 'o';
echo "Changing the character at index -3 to o gives $string.", PHP_EOL;
?>
```

The above example will output:

The character at index -2 is n.

Changing the character at index -3 to o gives strong.

For anything more complex, you should use the complex syntax.

4.2.2 Complex (curly) Syntax

This isn't called complex because the syntax is complex, but because it allows for the use of complex expressions.

Any scalar variable, array element or object property with a string representation can be included via this syntax. Simply write the expression the same way as it would appear outside the string, and then wrap it in {and}. Since {cannot be escaped, this syntax will only be recognized when the $ immediately follows the {Use {\$ to get a literal {$. Some examples to make it clear:

```
<?php
// Show all errors
error_reporting(E_ALL);

$great = 'fantastic';

// Won't work, outputs: This is { fantastic}
echo "This is { $great}";

// Works, outputs: This is fantastic
echo "This is {$great}";

// Works
echo "This square is {$square->width}00 centimeters broad.";
```

```
// Works, quoted keys only work using the curly brace syntax
echo "This works: {$arr['key']}";


// Works
echo "This works: {$arr[4][3]}";

// This is wrong for the same reason as $foo[bar] is wrong  outside a string.
// In other words, it will still work, but only because PHP first looks for a
// constant named foo; an error of level E_NOTICE (undefined constant) will be
// thrown.
echo "This is wrong: {$arr[foo][3]}";

// Works. When using multi-dimensional arrays, always use braces around arrays
// when inside of strings
echo "This works: {$arr['foo'][3]}";

// Works.
echo "This works: " . $arr['foo'][3];

echo "This works too: {$obj->values[3]->name}";

echo "This is the value of the var named $name: {${$name}}";

echo "This is the value of the var named by the return value of getName(): {${getName()}}";

echo "This is the value of the var named by the return value of \$object->getName(): {${$object->getName()}}";

// Won't work, outputs: This is the return value of getName(): {getName()}
echo "This is the return value of getName(): {getName()}";
```

```
?>
```

It is also possible to access class properties using variables within strings using this syntax.

```
<?php
class foo {
    var $bar = 'I am bar.';
}

$foo = new foo();
$bar = 'bar';
$baz = array('foo', 'bar', 'baz', 'quux');
echo "{$foo->$bar}\n";
echo "{$foo->{$baz[1]}}\n";
?>
```

The above example will output:

```
I am bar.
I am bar.
```

Using single curly braces (*{}*) will not work for accessing the return values of functions or methods or the values of class constants or static class variables.

```
<?php
// Show all errors.
error_reporting(E_ALL);

class beers {
    const softdrink = 'rootbeer';
    public static $ale = 'ipa';
}

$rootbeer = 'A & W';
$ipa = 'Alexander Keith\'s';

// This works; outputs: I'd like an A & W
echo "I'd like an {${beers::softdrink}}\n";
```

Remember

Functions, method calls, static class variables, and class constants inside {$} work since PHP 5. However, the value accessed will be interpreted as the name of a variable in the scope in which the string is defined.

```
// This works too; outputs: I'd like an Alexander Keith's
echo "I'd like an {${beers::$ale}}\n";
?>
```

4.2.3 String Access and Modification by Character

Characters within strings may be accessed and modified by specifying the zero-based offset of the desired character after the string using square array brackets, as in *$str*. Think of a string as an array of characters for this purpose. The functions substr() and substr_replace() can be used when you want to extract or replace more than 1 character.

Keyword

Integer is a whole number (not a fraction) that can be positive, negative, or zero. Therefore, the numbers 10, 0, -25, and 5,148 are all integers. Unlike floating point numbers, integers cannot have decimal places. Integers are a commonly used data type in computer programming.

As of PHP 7.1.0, negative string offsets are also supported. These specify the offset from the end of the string. Formerly, negative offsets emitted E_NOTICE for reading (yielding an empty string) and E_WARNING for writing (leaving the string untouched).

Writing to an out of range offset pads the string with spaces. Non-integer types are converted to **integer**. Illegal offset type emits E_NOTICE. Only the first character of an assigned string is used. As of PHP 7.1.0, assigning an empty string throws a fatal error. Formerly, it assigned a NULL byte.

Internally, PHP strings are byte arrays. As a result, accessing or modifying a string using array brackets is not multi-byte safe, and should only be done with strings that are in a single-byte encoding such as ISO-8859-1.

Example #11. Some string examples

```
<?php
// Get the first character of a string
$str = 'This is a test.';
$first = $str[0];

// Get the third character of a string
$third = $str[2];
```

```
// Get the last character of a string.
$str = 'This is still a test.';
$last = $str[strlen($str)-1];

// Modify the last character of a string
$str = 'Look at the sea';
$str[strlen($str)-1] = 'e';

?>
```

As of PHP 5.4 string offsets have to either be integers or integer-like strings, otherwise a warning will be thrown. Previously an offset like "foo" was silently cast to 0.

Example #12. Differences between PHP 5.3 and PHP 5.4

```
<?php
$str = 'abc';

var_dump($str['1']);
var_dump(isset($str['1']));

var_dump($str['1.0']);
var_dump(isset($str['1.0']));

var_dump($str['x']);
var_dump(isset($str['x']));

var_dump($str['1x']);
var_dump(isset($str['1x']));
?>
```

Output of the above example in PHP 5.3:

```
string(1) "b"
bool(true)
string(1) "b"
bool(true)
string(1) "a"
```

bool(true)
string(1) "b"
bool(true)
Output of the above example in PHP 5.4:
string(1) "b"
bool(true)

Warning: Illegal string offset '1.0' in /tmp/t.php on line 7
string(1) "b"
bool(false)

Warning: Illegal string offset 'x' in /tmp/t.php on line 9
string(1) "a"
bool(false)
string(1) "b"
(bool(false

4.2.4 Converting to String

A value can be converted to a string using the *(string)* cast or the strval() function. String conversion is automatically done in the scope of an expression where a string is needed. This happens when using the echo or print functions, or when a variable is compared to a string.

A boolean TRUE value is converted to the string *"1"*. Boolean FALSE is converted to "" (the empty string). This allows conversion back and forth between boolean and string values.

An integer or float is converted to a string representing the number textually (including the exponent part for floats). Floating point numbers can be converted using exponential notation (*4.1E+6*).

The decimal point character is defined in the script's locale (category LC_NUMERIC).

Arrays are always converted to the string *"Array"*; because of this, echo and print can not by themselves show the contents of an array. To view a single element, use a construction such as *echo $arr['foo']*.

In order to convert objects to string magic method __toString must be used.

Resources are always converted to strings with the structure *"Resource id #1"*, where *1* is the resource number assigned to the resource by PHP at runtime. While

the exact structure of this string should not be relied on and is subject to change, it will always be unique for a given resource within the lifetime of a script being executed (ie a Web request or CLI process) and won't be reused. To get a resource's type, use the get_resource_type() function.

NULL is always converted to an empty string.

As stated above, directly converting an array, object, or resource to a string does not provide any useful information about the value beyond its type. See the functions print_r() and var_dump() for more effective means of inspecting the contents of these types. Most PHP values can also be converted to strings for permanent storage. This method is called serialization, and is performed by the serialize() function. If the PHP engine was built with WDDX support, PHP values can also be serialized as well-formed XML text.

Did You Know?

On July 14, 2004, PHP 5 was released, powered by the new Zend Engine II. PHP 5 included new features such as improved support for object-oriented programming, the PHP Data Objects (PDO) extension (which defines a lightweight and consistent interface for accessing databases), and numerous performance enhancements.

4.2.5 String Conversion to Numbers

When a string is evaluated in a numeric context, the resulting value and type are determined as follows.

If the string does not contain any of the characters '.', 'e', or 'E' and the **numeric value** fits into integer type limits (as defined by PHP_INT_MAX), the string will be evaluated as an integer. In all other cases it will be evaluated as afloat.

The value is given by the initial portion of the string. If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be 0 (zero). Valid numeric data is an optional sign, followed by one or more digits (optionally containing a decimal point), followed by an optional exponent. The exponent is an 'e' or 'E' followed by one or more digits.

```
<?php
$foo = 1 + "10.5";                // $foo is float (11.5)
$foo = 1 + "-1.3e3";              // $foo is float (-1299)
$foo = 1 + "bob-1.3e3";           // $foo is integer (1)
$foo = 1 + "bob3";                // $foo is integer (1)
$foo = 1 + "10 Small Pigs";       // $foo is integer (11)
$foo = 4 + "10.2 Little Piggies"; // $foo is float (14.2)
$foo = "10.0 pigs " + 1;          // $foo is float (11)
```

Keyword

Numerical value is - a quantitative value assigned to a letter of the alphabet.

```
$foo = "10.0 pigs " + 1.0;          // $foo is float (11)
?>
```

To test any of the examples, cut and paste the examples and insert the following line to see what's going on:

```
<?php
echo "\$foo==$foo; type is " . gettype ($foo) . "<br />\n";
?>
```

Do not expect to get the code of one character by converting it to integer, as is done in C. Use the ord() and chr()functions to convert between ASCII codes and characters.

4.2.6 Details of the String Type

The string in PHP is implemented as an array of bytes and an integer indicating the length of the buffer. It has no information about how those bytes translate to characters, leaving that task to the programmer. There are no limitations on the values the string can be composed of; in particular, bytes with value *0* ("NUL bytes") are allowed anywhere in the string (however, a few functions, said in this manual not to be "binary safe", may hand off the strings to libraries that ignore data after a NUL byte.)

This nature of the string type explains why there is no separate "byte" type in PHP – strings take this role. Functions that return no textual data – for instance, arbitrary data read from a network socket – will still return strings.

Given that PHP does not dictate a specific encoding for strings, one might wonder how string literals are encoded. For instance, is the string *"á"* equivalent to *"\xE1"* (ISO-8859-1), *"\xC3\xA1"* (UTF-8, C form), *"\x61\xCC\x81"* (UTF-8, D form) or any other possible representation? The answer is that string will be encoded in whatever fashion it is encoded in the script file.

Thus, if the script is written in ISO-8859-1, the string will be encoded in ISO-8859-1 and so on. However, this does not apply if Zend Multibyte is enabled; in that case, the script may be written in an arbitrary encoding (which is explicity declared or is detected) and then converted to a certain internal encoding, which is then the encoding that will be used for the string literals. Note that there are some constraints on the encoding of the script (or on the internal encoding, should Zend Multibyte be enabled) – this almost always means that this encoding should be a compatible superset of ASCII, such as UTF-8 or ISO-8859-1. Note, however, that state-dependent encodings where the same byte values can be used in initial and non-initial shift states may be problematic.Of course, in order to be useful, functions that operate on text may have to make some assumptions about how the string is encoded. Unfortunately, there is much variation on this matter throughout PHP's functions:

- Some functions assume that the string is encoded in some (any) single-byte encoding, but they do not need to interpret those bytes as specific characters. This is case of, for instance, substr(), strpos(), strlen() or strcmp(). Another way to think of these functions is that operate on memory buffers, i.e., they work with bytes and byte offsets.
- Other functions are passed the encoding of the string, possibly they also assume a default if no such information is given. This is the case of htmlentities() and the majority of the functions in the mbstringextension.
- Others use the current locale (see setlocale()), but operate byte-by-byte. This is the case of strcasecmp(),strtoupper() and ucfirst(). This means they can be used only with single-byte encodings, as long as the encoding is matched by the locale. For instance *strtoupper("á")* may return *"Á"* if the locale is correctly set and *á* is encoded with a single byte. If it is encoded in UTF-8, the correct result will not be returned and the resulting string may or may not be returned corrupted, depending on the current locale.
- Finally, they may just assume the string is using a specific encoding, usually UTF-8. This is the case of most functions in the intl extension and in the PCRE extension (in the last case, only when the *u* modifier is used). Although this is due to their special purpose, the function utf8_decode() assumes a UTF-8 encoding and the function utf8_encode() assumes an ISO-8859-1 encoding.

Ultimately, this means writing correct programs using Unicode depends on carefully avoiding functions that will not work and that most likely will corrupt the data and using instead the functions that do behave correctly, generally from the intl and mbstring extensions. However, using functions that can handle Unicode encodings is just the beginning. No matter the functions the language provides, it is essential to know the Unicode specification. For instance, a program that assumes there is only uppercase and lowercase is making a wrong assumption.

SUMMARY

- PHP string is a sequence of characters i.e., used to store and manipulate text.
- PHP supports only 256-character set and so that it does not offer native Unicode support.
- Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.
- Nowdocs are to single-quoted strings what heredocs are to double-quoted strings. A nowdoc is specified similarly to a heredoc, but no parsing is done inside a nowdoc.
- Parsing is used in all high-level programming languages. Languages like C++ and Java are parsed by their respective compilers before being transformed into executable machine code.
- When a string is specified in double quotes or with heredoc, variables are parsed within it.
- String conversion is automatically done in the scope of an expression where a string is needed.
- The string in PHP is implemented as an array of bytes and an integer indicating the length of the buffer.
- An integer or float is converted to a string representing the number textually (including the exponent part for floats).

KNOWLEDGE CHECK

1. **Returns a string arguments with trilling blank space removed, is a behavior of**
   a. starts ( ) function
   b. chop ( ) function
   c. rtrim ( ) function
   d. Both B and C
2. **A sequence of characters that is treated as a unit is called**
   a. Arrays
   b. Functions
   c. Methods
   d. Strings
3. **Ltrim ( ) returns its string argument with**
   a. Leading blankspace removed
   b. Trailing blankspace removed
   c. Numerical values
   d. None of them
4. **When two strings are exactly equivalent Strcmp ( ) returns what?**
   a. Returns a string
   b. Returns 0
   c. Returns 1
   d. Returns nothing
5. **For finding out one string is equal to another string which function we can use?**
   a. strpid ( )
   b. strpos ( )
   c. str ( )
   d. All of them
6. **In how many ways we can create strings in PHP?**
   a. 1
   b. 2
   c. 3
   d. 4

7. Which type of string can processes special characters inside quotes?

a. single quote string
b. double quote string
c. Both A and B
d. None of the above

REVIEW QUESTIONS

1. Discuss on the single quoted string.
2. Write a program on double quoted string technique.
3. What do understand by the variable parsing?
4. Explain the procedure of complex (curly) syntax.
5. How to access string and modify by character?

Check Your Result

1. (d) 2. (d) 3. (a) 4. (b) 5. (b)
6. (b) 7. (d)

REFERENCES

1. S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. M. Paradkar, ˙ and M. D. Ernst, "Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking," IEEE TSE, vol. 36, no. 4, pp. 474–494, 2010.
2. H. V. Nguyen, H. A. Nguyen, T. T. Nguyen, and T. N. Nguyen, "Auto-Locating and Fix-Propagating for HTML Validation Errors to PHP Server-Side Code," in ASE, 2011, pp. 13–22.
3. E. Torlak, "A constraint solver for software engineering: Finding models and cores of large relational specifications," Ph.D. dissertation, MIT, 2009.
4. V. Ganesh, A. Kiezun, S. Artzi, P. J. Guo, P. Hooimeijer, and ˙ M. D. Ernst, "HAMPI: A String Solver for Testing, Analysis and Vulnerability Detection," in CAV, 2011, pp. 1–19.
5. P. Saxena, D. Akhawe, S. Hanna, F. Mao, S. McCamant, and D. Song, "A Symbolic Execution Framework for JavaScript," in IEEE Symp. on Security and Privacy, 2010, pp. 513–528.
6. Y. Minamide, "Static Approximation of Dynamically Generated Web Pages," in WWW, 2005, pp. 432–441.
7. A. Møller and M. Schwarz, "HTML Validation of ContextFree Languages," in FOSSACS, 2011, pp. 426–440.
8. Y. Minamide and A. Tozawa, "XML Validation for ContextFree Grammars," in APLAS, 2006, pp. 357–373.
9. G. Wassermann, C. Gould, Z. Su, and P. Devanbu, "Static Checking of Dynamically Generated Queries in Database Applications," ACM TOSEM, vol. 16, September 2007.
10. G. Wassermann and Z. Su, "Static Detection of Cross-Site Scripting Vulnerabilities," in ICSE, 2008, pp. 171–180.
11. F. Yu, M. Alkhalaf, and T. Bultan, "Patching Vulnerabilities with Sanitization Synthesis," in ICSE, 2011, pp. 251–260.
12. A. Solar-Lezama, L. Tancau, R. Bod´ık, S. Seshia, and V. Saraswat, "Combinatorial Sketching for Finite Programs," in ASPLOS, 2006, pp. 404–415.
13. A. Solar-Lezama, C. G. Jones, and R. Bod´ık, "Sketching Concurrent Data Structures," in PLDI, 2008, pp. 136–148.
14. S. Gulwani, "Automating String Processing in Spreadsheets Using Input-Output Examples," in POPL, 2011, pp. 317–330.
15. S. Chandra, E. Torlak, S. Barman, and R. Bod´ık, "Angelic Debugging," in ICSE, 2011, pp. 121–130.

16. F. Tip, R. M. Fuhrer, A. Kiezun, M. D. Ernst, I. Balaban, ˙ and B. D. Sutter, "Refactoring Using Type Constraints," ACM TOPLAS, vol. 33, no. 3, 2011.

17. A. Donovan, A. Kiezun, M. S. Tschantz, and M. D. Ernst, ˙ "Converting Java Programs to Use Generic Libraries," in OOPSLA, 2004, pp. 15–34.

18. F. Steimann and A. Thies, "From Public to Private to Absent: Refactoring Java Programs under Constrained Accessibility," in ECOOP, 2009, pp. 419–443.

CHAPTER

5

PHP ARRAYS

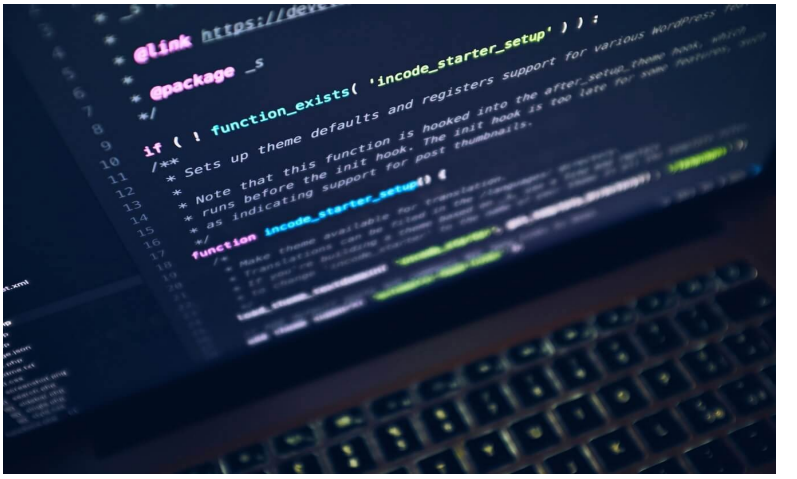
"As Bromberger observed, rules are understood to be elements of the computational systems that determine the sound and meaning of the infinite array of expressions of a language; the information so derived is accessed by other systems in language use."

– Noam Chomsky

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Discuss about array functions
2. Understand the arrays



INTRODUCTION

An array in PHP is actually an ordered map. A map is a type that associates values to keys. This type is optimized

for several different uses; it can be treated as an array, list (vector), hash table (an implementation of a map), dictionary, collection, stack, queue, and probably more. As array values can be other arrays, trees and multidimensional arrays are also possible.

Arrays in PHP is a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key. Suppose we want to store five names and print them accordingly. This can be easily done by the use of five different string variables. But if instead of five, the number rises to a hundred, then it would be really difficult for the user or developer to create so many different variables. Here array comes into play and helps us to store every element within a single variable and also allows easy access using an index or a key. An array is created using an array() function in PHP.

5.1 ARRAY FUNCTIONS

These functions allow you to interact with and manipulate arrays in various ways. Arrays are essential for storing, managing, and operating on sets of variables.

5.1.1 Installation

There is no installation needed to use these functions; they are part of the PHP core.

5.1.2 Runtime Configuration

This extension has no **configuration** directives defined in php.ini.

5.1.3 PHP Array Constants

Sr.No	Constant & Description
1	**CASE_LOWER** Used with array_change_key_case() to convert array keys to lower case
2	**CASE_UPPER** Used with array_change_key_case() to convert array keys to upper case
3	**SORT_ASC** Used with array_multisort() to sort in ascending order

4	**SORT_DESC** Used with array_multisort() to sort in descending order
5	**SORT_REGULAR** Used to compare items normally
6	**SORT_NUMERIC** Used to compare items numerically
7	**SORT_STRING** Used to compare items as strings
8	**SORT_LOCALE_STRING** Used to compare items as strings, based on the current locale
9	**COUNT_NORMAL**
10	**COUNT_RECURSIVE**
11	**EXTR_OVERWRITE**
12	**EXTR_SKIP**
13	**EXTR_PREFIX_SAME**
14	**EXTR_PREFIX_ALL**
15	**EXTR_PREFIX_INVALID**
16	**EXTR_PREFIX_IF_EXISTS**
17	**EXTR_IF_EXISTS**
18	**EXTR_REFS**

5.1.4 List of Functions

PHP – indicates the earliest version of PHP that supports the function.

Sr.No	Function & Description	PHP
1	**array()** Create an array	3
2	**array_change_key_case()** Returns an array with all keys in lowercase or uppercase	4
3	**array_chunk()** Splits an array into chunks of arrays	4

4	**array_combine()** Creates an array by using one array for keys and another for its values	5
5	**array_count_values()** Returns an array with the number of occurrences for each value	4
6	**array_diff()** Compares array values, and returns the differences	4
7	**array_diff_assoc()** Compares array keys and values, and returns the differences	4
8	**array_diff_key()** Compares array keys, and returns the differences	5
9	**array_diff_uassoc()** Compares array keys and values, with an additional user-made function check, and returns the differences	5
10	**array_diff_ukey()** Compares array keys, with an additional user-made function check, and returns the differences	5
11	**array_fill()** Fills an array with values	4
12	**array_fill_keys()** Fill an array with values, specifying keys	5
13	**array_filter()** Filters elements of an array using a user-made function	4
14	**array_flip()** Exchanges all keys with their associated values in an array	4

15	**array_intersect()** Compares array values, and returns the matches	4
16	**array_intersect_assoc()** Compares array keys and values, and returns the matches	4
17	**array_intersect_key()** Compares array keys, and returns the matches	5
18	**array_intersect_uassoc()** Compares array keys and values, with an additional user-made function check, and returns the matches	5
19	**array_intersect_ukey()** Compares array keys, with an additional user-made function check, and returns the matches	5
20	**array_key_exists()** Checks if the specified key exists in the array	4
21	**array_keys()** Returns all the keys of an array	4
22	**array_map()** Sends each value of an array to a user-made function, which returns new values	4
23	**array_merge()** Merges one or more arrays into one array	4
24	**array_merge_recursive()** Merges one or more arrays into one array	4
25	**array_multisort()** Sorts multiple or multi-dimensional arrays	4
26	**array_pad()** Inserts a specified number of items, with a specified value, to an array	4

27	**array_pop()** Deletes the last element of an array	4
28	**array_product()** Calculates the product of the values in an array	5
29	**array_push()** Inserts one or more elements to the end of an array	4
30	**array_rand()** Returns one or more random keys from an array	4
31	**array_reduce()** Returns an array as a string, using a user-defined function	4
32	**array_reverse()** Returns an array in the reverse order	4
33	**array_search()** Searches an array for a given value and returns the key	4
34	**array_shift()** Removes the first element from an array, and returns the value of the removed element	4
35	**array_slice()** Returns selected parts of an array	4
36	**array_splice()** Removes and replaces specified elements of an array	4
37	**array_sum()** Returns the sum of the values in an array	4
38	**array_udiff()** Compares array values in a user-made function and returns an array	5

39	**array_udiff_assoc()** Compares array keys, and compares array values in a user-made function, and returns an array	5
40	**array_udiff_uassoc()** Compares array keys and array values in user-made functions, and returns an array	5
41	**array_uintersect()** Compares array values in a user-made function and returns an array	5
42	**array_uintersect_assoc()** Compares array keys, and compares array values in a user-made function, and returns an array	5
43	**array_uintersect_uassoc()** Compares array keys and array values in user-made functions, and returns an array	5
44	**array_unique()** Removes duplicate values from an array	4
45	**array_unshift()** Adds one or more elements to the beginning of an array	4
46	**array_values()** Returns all the values of an array	4
47	**array_walk()** Applies a user function to every member of an array	3
48	**array_walk_recursive()** Applies a user function recursively to every member of an array	5
49	**arsort()** Sorts an array in reverse order and maintain index association	3

50	**asort()** Sorts an array and maintain index association	3
51	**compact()** Create array containing variables and their values	4
52	**count()** Counts elements in an array, or properties in an object	3
53	**current()** Returns the current element in an array	3
54	**each()** Returns the current key and value pair from an array	3
55	**end()** Sets the internal pointer of an array to its last element	3
56	**extract()** Imports variables into the current symbol table from an array	3
57	**in_array()** Checks if a specified value exists in an array	4
58	**key()** Fetches a key from an array	3
59	**krsort()** Sorts an array by key in reverse order	3
60	**ksort()** Sorts an array by key	3
61	**list()** Assigns variables as if they were an array	3
62	**natcasesort()** Sorts an array using a case insensitive "natural order" algorithm	4

63	**natsort()** Sorts an array using a "natural order" algorithm	4
64	**next()** Advance the internal array pointer of an array	3
65	**pos()** Alias of current()	3
66	**prev()** Rewinds the internal array pointer	3
67	**range()** Creates an array containing a range of elements	3
68	**reset()** Sets the internal pointer of an array to its first element	3
69	**rsort()** Sorts an array in reverse order	3
70	**shuffle()** Shuffles an array	3
71	**sizeof()** Alias of count()	3
72	**sort()** Sorts an array	3
73	**uasort()** Sorts an array with a user-defined function and maintain index association	3
74	**uksort()** Sorts an array by keys using a user-defined function	3
75	**usort()** Sorts an array by values using a user-defined function	3

5.2 ARRAYS

An array is a data structure that stores one or more similar type of values in a single value.

If you want to store 100 numbers then instead of defining 100 variables it is easy to define an array of 100 length.

5.2.1 Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create **array**. This function is explained in function reference.

```
<html>
   <body>

      <?php
         /* First method to create array. */
         $numbers = array( 1, 2, 3, 4, 5);

         foreach( $numbers as $value ) {
            echo "Value is $value <br />";
         }

         /* Second method to create array. */
         $numbers[0] = "one";
         $numbers[1] = "two";
         $numbers[2] = "three";
```

```
        $numbers[3] = "four";
        $numbers[4] = "five";

        foreach( $numbers as $value ) {
            echo "Value is $value <br />";
        }
    ?>

  </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five
```

5.2.2 Indexed Versus Associative Arrays

There are two kinds of arrays in PHP: indexed and associative. The keys of an indexed array are integers, beginning at 0. Indexed arrays are used when you identify things by their position. Associative arrays have strings as keys and behave more like two-column tables. The first column is the key, which is used to access the value.

PHP internally stores all arrays as associative arrays; the only difference between associative and indexed arrays is what the keys happen to be. Some array features are provided mainly for use with indexed arrays because they assume that you have or want keys that are consecutive integers beginning at 0. In both cases, the keys are unique. In other words, you cannot have two elements with the same key, regardless of whether the key is a string or an integer.

PHP arrays have an internal order to their elements that is independent of the keys and values, and there are functions that you can use to traverse the arrays based

on this internal order. The order is normally that in which values were inserted into the array.

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish 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, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

Remember

Do not keep associative array inside double quote while printing otherwise it would not return any value.

Example

```
<html>
   <body>

      <?php
         /* First method to associate create array. */
          $salaries = array("mohammad" => 2000, "qadir"
=> 1000, "zara" => 500);

        echo "Salary of mohammad is ". $salaries['mohammad']
. "<br />";
          echo "Salary of qadir is ".  $salaries['qadir']. "<br
/>";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";

         /* Second method to create array. */
         $salaries['mohammad'] = "high";
         $salaries['qadir'] = "medium";
         $salaries['zara'] = "low";

        echo "Salary of mohammad is ". $salaries['mohammad']
. "<br />";
```

```
            echo "Salary of qadir is ".  $salaries['qadir']. "<br
/>";
            echo "Salary of zara is ".  $salaries['zara']. "<br />";
          ?>

      </body>
    </html>
```

This will produce the following result –

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

5.2.3 Identifying Elements of an Array

Before we look at creating an array, let us look at the structure of an existing array. You can access specific values from an existing array using the array variable's name, followed by the element's key, or index, within square brackets:

```
$age['fred']
$shows[2]
```

The key can be either a string or an integer. String values that are equivalent to integer numbers (without leading zeros) are treated as integers. Thus, $array[3] and $array['3'] reference the same element, but $array['03'] references a different element. Negative numbers are valid keys, but they do not specify positions from the end of the array as they do in Perl.

You do not have to quote single-word strings. *For instance*, $age['fred'] is the same as $age[fred]. However, it is considered good PHP style to always use quotes, because quoteless keys are indistinguishable from constants. When you use a constant as an unquoted index, PHP uses the value of the **constant** as the index and emits a warning:

```
define('index', 5);
echo $array[index];              // retrieves $array[5], not $array['index'];
```

Keyword

Constant is a value that cannot be altered by the program during normal execution, i.e., the value is constant.

You must use quotes if you are using interpolation to build the array index:

```
$age["Clone{$number}"]
```

Although sometimes optional, you should also quote the key if you are interpolating an array lookup to ensure that you get the value you expect:

```
// these are wrong
print "Hello, {$person['name']}";
print "Hello, {$person["name"]}";
```

5.2.4 Storing Data in Arrays

Storing a value in an array will create the array if it did not already exist, but trying to retrieve a value from an array that has not been defined would not create the array. *For example*:

```
// $addresses not defined before this point
echo $addresses[0];                  // prints nothing
echo $addresses;                     // prints nothing

$addresses[0] = "spam@cyberpromo.net";
echo $addresses;                        // prints "Array"
```

Using simple assignment to initialize an array in your program can lead to code like this:

```
$addresses[0] = "spam@cyberpromo.net";
$addresses[1] = "abuse@example.com";
$addresses[2] = "root@example.com";
```

That's an indexed array, with integer indices beginning at 0. Here's an associative array:

```
$price['gasket'] = 15.29;
$price['wheel'] = 75.25;
$price['tire'] = 50.00;
```

An easier way to initialize an array is to use the array() construct, which builds an array from its arguments. This builds an indexed array, and the index values (starting at 0) are created automatically:

```
$addresses = array("spam@cyberpromo.net", "abuse@example.com", "root@example.com");
```

To create an **associative array** with array(), use the => symbol to separate indices (keys) from values:

```
$price = array(
  'gasket' => 15.29,
  'wheel'  => 75.25,
  'tire'   => 50.00
);
```

Notice the use of whitespace and alignment. We could have bunched up the code, but it would not have been as easy to read (this is equivalent to the previous code sample), or as easy to add or remove values:

;(price = array('gasket' => 15.29, 'wheel' => 75.25, 'tire' => 50.00$

You can also specify an array using a shorter, alternate syntax:

$days = [‘gasket’ => 15.29, ‘wheel’ => 75.25, ‘tire’ => 50.0];

To construct an empty array, pass no arguments to array():

$addresses = array();

You can specify an initial key with => and then a list of values. The values are inserted into the array starting with that key, with subsequent values having sequential keys:

$days = array(1 => “Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”, “Sun”);

// 2 is Tue, 3 is Wed, etc.

If the initial index is a nonnumeric string, subsequent indices are integers beginning at 0. Thus, the following code is probably a mistake:

```
$whoops = array('Fri' => "Black", "Brown", "Green");

// same as
$whoops = array('Fri' => "Black", 0 => "Brown", 1 => "Green");
```

Keyword

Associative array is an abstract data type composed of a collection of (key, value) pairs, such that each possible key appears at most once in the collection.

Adding Values to the End of an Array

To insert more values into the end of an existing indexed array, use the [] syntax:

$family = array(“Fred”, “Wilma”);

$family[] = “Pebbles”; // $family[2] is “Pebbles”

This construct assumes the array’s indices are numbers and assigns elements into the next available numeric index, starting from 0. Attempting to append to an associative array without appropriate keys is almost always a programmer mistake, but PHP will give the new elements numeric indices without issuing a warning:

```
$person = array('name' => "Fred");
$person[] = "Wilma"; // $person[0] is now "Wilma"
```

Assigning a Range of Values

The range() function creates an array of consecutive integer or character values between and including the two values you pass to it as arguments. *For example*:

```
$numbers = range(2, 5);              // $numbers = array(2, 3, 4, 5);
$letters = range('a', 'z');          // $letters holds the alphabet
$reversedNumbers = range(5, 2);      // $reversedNumbers = array(5, 4, 3, 2);
```

Only the first letter of a string argument is used to build the range:

```
range("aaa", "zzz");                 // same as range('a','z')
```

Getting the Size of an Array

The count() and sizeof() functions are identical in use and effect. They return the number of elements in the array. There is no stylistic preference about which function you use. Here's an example:

```
$family = array("Fred", "Wilma", "Pebbles");
$size = count($family);              // $size is 3
```

This function counts only array values that are actually set:

```
$confusion = array( 10 => "ten", 11 => "eleven", 12 => "twelve");
$size = count($confusion);      // $size is 3
```

Padding an Array

To create an array with values initialized to the same content, use array_pad(). The first argument to array_pad() is the array, the second argument is the minimum number of elements you want the array to have, and the third argument is the value to give any elements that are created. The array_pad() function returns a new padded array, leaving its argument (source) array alone.

Here's array_pad() in action:

```
$scores = array(5, 10);
$padded = array_pad($scores, 5, 0);    // $padded is now array(5, 10, 0, 0, 0)
```

Notice how the new values are appended to the end of the array. If you want the new values added to the start of the array, use a negative second argument:

```
$padded = array_pad($scores, -5, 0);   // $padded is now array(0, 0, 0, 5, 10);
```

If you pad an associative array, existing keys will be preserved. New elements will have numeric keys starting at 0.

5.2.5 Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

The values in an array can themselves be arrays. This lets :you easily create multidimensional arrays

```
$row0 = array(1, 2, 3);
$row1 = array(4, 5, 6);
$row2 = array(7, 8, 9);
$multi = array($row0, $row1, $row2);
```

You can refer to elements of multidimensional arrays by appending more []s:

```
$value = $multi[2][0];              // row 2, column 0. $value = 7
```

To interpolate a lookup of a multidimensional array, you must enclose the entire array lookup in curly braces:

```
echo("The value at row 2, column 0 is {$multi[2][0]}\n");
```

:Failing to use the curly braces results in output like this

```
The value at row 2, column 0 is Array[0]
```

Example

In this example we create a two dimensional array to store – marks of three students in three subjects

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
   <body>

      <?php
         $marks = array(
            "mohammad" => array (
               "physics" => 35,
               "maths" => 30,
               "chemistry" => 39
            ),

            "qadir" => array (
               "physics" => 30,
               "maths" => 32,
               "chemistry" => 29
```

Did You Know?

Heinz Rutishauser's programming language Superplan (1949–1951) included multi-dimensional arrays. Rutishauser however although describing how a compiler for his language should be built, did not implement one. Assembly languages and low-level languages like BCPL generally have no syntactic support for arrays. Because of the importance of array structures for efficient computation, the earliest high-level programming languages, including FORTRAN (1957), COBOL(1960), and Algol 60 (1960), provided support for multi-dimensional arrays.

```
        ),

        "zara" => array (
            "physics" => 31,
            "maths" => 22,
            "chemistry" => 39
        )
    );

    /* Accessing multi-dimensional array values */
    echo "Marks for mohammad in physics : " ;
    echo $marks['mohammad']['physics'] . "<br />";

    echo "Marks for qadir in maths : ";
    echo $marks['qadir']['maths'] . "<br />";

    echo "Marks for zara in chemistry : " ;
    echo $marks['zara']['chemistry'] . "<br />";
?>

  </body>
</html>
```

This will produce the following result –

```
Marks for mohammad in physics : 35
Marks for qadir in maths : 32
Marks for zara in chemistry : 39
```

5.2.6 Extracting Multiple Values

To copy all of an array's values into variables, use the list() construct:

```
list ($variable, ...) = $array;
```

The array's values are copied into the listed variables in the array's internal order. By default that's the order in which they were inserted. Here's an example:

```
$person = array("Fred", 35, "Betty");
list($name, $age, $wife) = $person;
// $name is "Fred", $age is 35, $wife is "Betty"
```

If you have more values in the array than in the list(), the extra values are ignored:

```
$person = array("Fred", 35, "Betty");
list($name, $age) = $person;          // $name is "Fred", $age is 35
```

If you have more values in the list() than in the array, the extra values are set to NULL:

```
$values = array("hello", "world");
list($a, $b, $c) = $values;          // $a is "hello", $b is "world", $c is NULL
```

Two or more consecutive commas in the list() skip values in the array:

```
$values = range('a', 'e');              // use range to populate the array
list($m, , $n, , $o) = $values;         // $m is "a", $n is "c", $o is "e"
```

Slicing an Array

To extract only a subset of the array, use the array_slice() function:

$subset = array_slice(array, offset, length);

The array_slice() function returns a new array consisting of a consecutive series of values from the original array. The offset parameter identifies the initial element to copy (0 represents the first element in the array), and the length parameter identifies the number of values to copy. The new array has consecutive numeric keys starting at 0. *For example*:

```
$people = array("Tom", "Dick", "Harriet", "Brenda", "Jo");
$middle = array_slice($people, 2, 2); // $middle is array("Harriet", "Brenda")
```

It is generally only meaningful to use array_slice() on indexed arrays (i.e., those with consecutive integer indices starting at 0):

```
// this use of array_slice() makes no sense
$person = array('name' => "Fred", 'age' => 35, 'wife' => "Betty");
$subset = array_slice($person, 1, 2); // $subset is array(0 => 35, 1 => "Betty")
```

Combine array_slice() with list() to extract only some values to variables:

```
$order = array("Tom", "Dick", "Harriet", "Brenda", "Jo");
list($second, $third) = array_slice($order, 1, 2);
// $second is "Dick", $third is "Harriet"
```

Remember

The use of the list function is a common practice for picking up values from a database selection where only one row is returned. This would automatically load the data from the simple query into a series of local variables. Here is an example of selecting two opposing teams from a sports scheduling database:

$sql = "SELECT HomeTeam, AwayTeam FROM schedule WHERE Ident = 7";

$result = mysql_query($sql);

list($hometeam, $awayteam) = mysql_fetch_assoc($result);

Splitting an Array into Chunks

To divide an array into smaller, evenly sized arrays, use the array_chunk() function:

;([chunks = array_chunk(array, size [, preserve_keys$

The function returns an array of the smaller arrays. The third argument, pre serve_keys, is a **Boolean value** that determines whether the elements of the new arrays have the same keys as in the original (useful for associative arrays) or new numeric keys starting from 0 (useful for indexed arrays). The default is to assign new keys, as shown here:

Keyword

Boolean value is one with two choices: true or false, yes or no, 1 or 0.

```
$nums = range(1, 7);
$rows = array_chunk($nums, 3);
print_r($rows);

Array (
  [0] => Array (
    [0] => 1
    [1] => 2
    [2] => 3
  )
  [1] => Array (
    [0] => 4
    [1] => 5
    [2] => 6
  )
  [2] => Array (
    [0] => 7
  )
)
```

Keys and Values

The array_keys() function returns an array consisting of only the keys in the array in internal order:

$arrayOfKeys = array_keys(array);

Here's an example:

```
$person = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
$keys = array_keys($person); // $keys is array("name", "age", "wife")
```

PHP also provides a (generally less useful) function to retrieve an array of just the values in an array, array_values():

$arrayOfValues = array_values(array);

As with array_keys(), the values are returned in the array's internal order:

```
$values = array_values($person); // $values is array("Fred", 35, "Wilma");
```

Checking Whether an Element Exists

To see if an element exists in the array, use the array_key_exists() function:

if (array_key_exists(key, array)) { ... }

The function returns a Boolean value that indicates whether the first argument is a valid key in the array given as the second argument.

It is not sufficient to simply say:

```
if ($person['name']) { ... }          // this can be misleading
```

Even if there is an element in the array with the key name, its corresponding **value** might be false (i.e., 0, NULL, or the empty string). Instead, use array_key_exists(), as follows:

```
$person['age'] = 0; // unborn?

if ($person['age']) {
  echo "true!\n";
}

if (array_key_exists('age', $person)) {
 echo "exists!\n";
}
```

```
exists!
```

Many people use the isset() function instead, which returns true if the element exists and is not NULL:

```
$a = array(0, NULL, '');

function tf($v)
{
    return $v ? 'T' : 'F';
}

for ($i=0; $i < 4; $i++) {
  printf("%d: %s %s\n", $i, tf(isset($a[$i])), tf(array_key_exists($i, $a)));
}
```

```
0: T T
1: F T
2: T T
3: F F
```

Removing and Inserting Elements in an Array

The array_splice() function can remove or insert elements in an array and optionally create another array from the removed elements:

$removed = array_splice(array, start [, length [, replacement]]);

We will look at array_splice() using this array:

Keyword

Value is the representation of some entity that can be manipulated by a program.

```
$subjects = array("physics", "chem", "math", "bio", "cs", "drama", "classics");
```

We can remove the "math", "bio", and "cs" elements by telling array_splice() to start at position 2 and remove 3 elements:

```
$removed = array_splice($subjects, 2, 3);
// $removed is array("math", "bio", "cs")
// $subjects is array("physics", "chem", "drama", "classics")
```

If you omit the length, array_splice() removes to the end of the array:

```
$removed = array_splice($subjects, 2);
// $removed is array("math", "bio", "cs", "drama", "classics")
// $subjects is array("physics", "chem")
```

If you simply want to delete elements from the source array and you do not care about retaining their values, you do not need to store the results of array_splice():

```
array_splice($subjects, 2);
// $subjects is array("physics", "chem");
```

To insert elements where others were removed, use the fourth argument:

```
$new = array("law", "business", "IS");
array_splice($subjects, 4, 3, $new);
// $subjects is array("physics", "chem", "math", "bio", "law", "business", "IS")
```

The size of the replacement array does not have to be the same as the number of elements you delete. The array grows or shrinks as needed:

```
$new = array("law", "business", "IS");
array_splice($subjects, 3, 4, $new);
// $subjects is array("physics", "chem", "math", "law", "business", "IS")
```

To insert new elements into the array while pushing existing elements to the right, delete zero elements:

```
$subjects = array("physics", "chem", "math');
$new = array("law", "business");
array_splice($subjects, 2, 0, $new);
// $subjects is array("physics", "chem", "law", "business", "math")
```

Although the examples so far have used an indexed array, array_splice() also works on associative arrays:

```
$capitals = array(
  'USA'           => "Washington",
  'Great Britain' => "London",
  'New Zealand'   => "Wellington",
  'Australia'     => "Canberra",
  'Italy'         => "Rome"
  'Canada'        => "Ottawa"
);

$downUnder = array_splice($capitals, 2, 2); // remove New Zealand and Australia
$france = array('France' => "Paris");

array_splice($capitals, 1, 0, $france);      // insert France between USA and GB
```

SUMMARY

- Arrays in PHP is a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data.
- The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key.
- PHP internally stores all arrays as associative arrays; the only difference between associative and indexed arrays is what the keys happen to be.
- The array's values are copied into the listed variables in the array's internal order.
- The array_slice() function returns a new array consisting of a consecutive series of values from the original array.
- The function returns a Boolean value that indicates whether the first argument is a valid key in the array given as the second argument.

KNOWLEDGE CHECK

1. **Which of the following type of variables are named and indexed collections of other values?**
   a. Strings
   b. Arrays
   c. Objects
   d. Resources
2. **Which of the following array represents an array containing one or more arrays?**
   a. Numeric Array
   b. Associative Array
   c. Multidimentional Array
   d. None of the above.
3. **Which of the following type of variables are sequences of characters, like 'PHP supports string operations.'?**
   a. Strings
   b. Arrays
   c. Objects
   d. Resources
4. **Which of the following function returns the sum of the values in an array?**
   a. array_sum()
   b. array_splice()
   c. array_udiff()
   d. array_udiff_assoc()
5. **Which of the following is correct about determine the "truth" of any value not already of the Boolean type?**
   a. If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
   b. Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
   c. Don't use double as Booleans.
   d. All of the above.

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

a. array_unshift()
b. into_array()
c. inend_array()
d. array_push()

7. Which of the following function is used to get the value of the previous element in an array?

a. last()
b. before()
c. prev()
d. previous()

REVIEW QUESTIONS

1. Explain about array functions.
2. Discuss about PHP array constants.
3. What is numeric array?
4. What do you understand by associate array? Describe.
5. What is multidimensional array?

Check Your Result

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

REFERENCES

1. A. Gotlieb, "Euclide: A Constraint-Based Testing Framework for Critical C Programs," in ICST. IEEE Computer Society, 2009, pp. 151–160.
2. A. Gotlieb, B. Botella, and M. Rueher, "A CLP Framework for Computing Structural Test Data," in Computational Logic, ser. Lecture Notes in Computer Science, J. W. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, Eds., vol. 1861. Springer, 2000, pp. 399–413.
3. B. K. Aichernig, "Contract-based testing," in Formal Methods at the Crossroads: From Panacea to Foundational Support, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2757, pp. 34–48.
4. I. Enderlin, F. Dadeau, A. Giorgetti, and A. B. Othman, "Praspel: A specification language for contract-based testing in php," in ICTSS, ser. Lecture Notes in Computer Science, B. Wolff and F. Za¨ıdi, Eds., vol. 7019. Springer, 2011, pp. 64–79.
5. I. Enderlin, F. Dadeau, A. Giorgetti, and F. Bouquet, "Grammar-Based Testing Using Realistic Domains in PHP," in ICST, G. Antoniol, A. Bertolino, and Y. Labiche, Eds. IEEE, 2012, pp. 509–518.
6. M. Barnett, K. Leino, and W. Schulte, "The Spec# Programming System: An Overview," in Proceedings of the International Workshop on Construction and Analysis of Safe, Secure and Interoperable Smart devices (CASSIS'04), ser. LNCS, vol. 3362. Marseille, France: SpringerVerlag, March 2004, pp. 49–69.
7. N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, "Minizinc: Towards a standard cp modelling language," in CP, ser. Lecture Notes in Computer Science, C. Bessiere, Ed., vol. 4741. Springer, 2007, pp. 529–543.
8. P. Baudin, J.-C. Filliatre, T. Hubert, C. March ˆ e, B. Monate, Y. Moy, and ´ V. Prevosto, ACSL: ANSI C Specification Language (preliminary design V1.2), 2008.
9. P. Madsen, "Unit Testing using Design by Contract and Equivalence Partitions," in XP'03: Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering. Berlin, Heidelberg: Springer, 2003, pp. 425–426.
10. S. Bardin and A. Gotlieb, "FDCC: A Combined Approach for Solving Constraints over Finite Domains and Arrays," in CPAIOR, ser. Lecture Notes in Computer Science, N. Beldiceanu, N. Jussien, and E. Pinson, Eds., vol. 7298. Springer, 2012, pp. 17–33.
11. V. Ganesh, A. Kiezun, S. Artzi, P. J. Guo, P. Hooimeijer, and M. D. Ernst, "Hampi: A string solver for testing, analysis and vulnerability detection," in CAV, ser. Lecture Notes in Computer Science, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 1–19.

12. Y. Cheon and G. T. Leavens, "A Simple and Practical Approach to Unit Testing: The JML and JUnit Way," in ECOOP 2002 — Object-Oriented Programming, 16th European Conference, ser. LNCS, B. Magnusson, Ed., vol. 2374. Berlin: Springer, Jun. 2002, pp. 231–255.

CHAPTER

6

OBJECTS

"It's hard enough to find an error in your code when you're looking for it; it's even harder when you've assumed your code is error-free."

– Steve McConnell

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Explain PHP objects and classes
2. Discuss how to create an object
3. Describe how to declare a class

INTRODUCTION

In PHP, Object is a compound data type (along with arrays). Values of more than one types can be stored together in

a single variable. Object is an instance of either a built-in or user defined class. In addition to properties, class defines functionality associated with data.

PHP is an object oriented language, although it does not have to be used as one, since most PHP functions are not object oriented. In object oriented programming, a class is a definition of an object, whereas an object is an instance of an object, meaning that from one class you can create many objects.

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values. Objects of a class is created using the new keyword.

6.1 PHP OBJECTS AND CLASSES

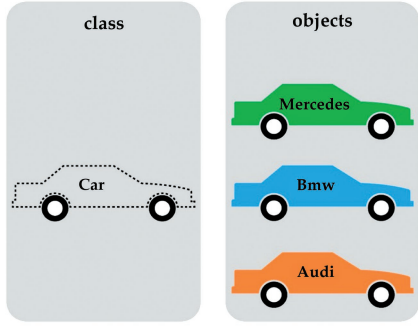
The basic building blocks of object-oriented programming are classes and objects. Understanding object and class and the differences between object and class is very important before we can dive into object-oriented programming approach.

6.1.1 PHP Objects vs. Classes

Remember

Notice that the getter and setter names, match the associated property name.

In the real world, an object has its own characteristics and behaviors. We can group similar objects with the same characteristics and behaviors in into one class. So we can say that a class is a blueprint of the objects. See the image below:



From the same Car class, we created three individual objects with the name of: Mercedes, Bmw, and Audi.

Although all of the objects were created from the same class and thus have the class's methods and properties, they are still different. This is not only, because they have different names, but also because they may have different values assigned to their properties. For example, in the image above, they differ by the color property - the Mercedes is green while the Bmw is blue and the Audi is orange.

6.1.2 Defining Classes and Objects

Before creating any new object, you need a class or blueprint of the object. It is pretty easy to define a new class in PHP. To define a new class in PHP you use the class keyword as the following example:

```php
<?php
class BankAccount{

}
```

We've defined a new empty class named BankAccount. From the BankAccount class, we can create new bank account object using the new keyword as follows:

```php
<?php
class BankAccount{

}

// create a new bank account object
$account = new BankAccount();

var_dump($account);
```

Notice that we use var_dump() function to see the content of the bank account.

6.1.3 Properties

In object-oriented terminology, characteristics of an object are called *properties*. For example, the bank account has account number and total balance. So let's add those properties to the BankAccountclass:

```php
<?php
class BankAccount{
 private $accountNumber;
 private $totalBalance;
}
```

You see that we used the private keyword in front of the properties. This is called property visibility. Each property can have one of three visibility levels, known as private, protected and public.

- private: private properties of a class only can be accessible by the *methods inside the class*. The methods of the class will be introduced a little later.
- protected: protected properties are similar to the private properties except that protected properties can be accessible by the subclasses. You will learn about the subclasses and inheritance later.
- public: public properties can be accessible not only by the methods inside but also by the code outside of the class.

Remember

The data/ variables inside a class (ex: var $name;) are called 'properties'.

In the BankAccount class, we can only access those properties of the bank account inside the class.

6.1.4 Methods

Behaviors of the object or class are called methods. Similar to the class properties, the methods can have three different visibility levels: private, protected, and public.

With a bank account, we can deposit money, withdraw money and get the balance. In addition, we need methods to set the bank account number and get the bank account number. The bank account number is used to distinguish from one bank account to the other.

Keyword

Function was originally the idealization of how a varying quantity depends on another quantity.

To create a method for a class you use the function keyword followed by the name and parentheses. A method is similar to a function except that a method is associated with a class and has a visibility level. Like a **function**, a method can have one or more parameter and can return a value.

Let's add some methods to the BankAccount class:

```
<?php
class BankAccount{
 /**
 * bank account number
 * @var string bank account number
 */
 private $accountNumber;
 /**
 * total balance
```

```
* @var float total balance
*/
private $totalBalance;

/**
* deposit money to the bank account
* @param float $amount amount to deposit
*/
public function deposit($amount){
$this->totalBalance += $amount;
}

/**
* withdraw money from the bank account
* @param double $amount
*/
public function withdraw($amount){
if($amount > $this->totalBalance)
die('Not enough money to withdraw');

$this->totalBalance -= $amount;
}

/**
* returns total balance
* @return float total balance
*/
public function getBalance(){
return $this->totalBalance;
}

/**
* return bacnk account number
```

```
*/
public function getAccountNumber(){
return $this->accountNumber;
}

/**
* set bank account number
* @param string $accountNumber
*/
public function setAccountNumber($accountNumber){
$this->accountNumber = $accountNumber;
 }
}
```

Notice that we used a special $this variable to access a property of an object. PHP uses the $this variable to refer to the current object.

To access a property you use the arrow (->) operator as follows:

```
object->property
```

Let's examine the BankAccount's methods in greater detail:

- The deposit() method increases the total balance of the account.
- The withdraw() method checks the available balance. If the total balance is less than the amount to be withdrew, it issues an error message, otherwise it decreases the total balance.
- The getBalance() method returns the total balance of the account.
- The setAccountNumber() and getAccountNumber() methods are called *setter*/*getter*methods.

Keyword

Parenthesis is a tall, curvy punctuation mark used to set off material that isn't fundamental to the main topic, like an afterthought or an aside.

Calling Methods

To call a method of an object, you use the (->) operator followed by the method name and **parentheses**. Let's call the methods of the bank account class:

```
// create a new bank account object
$account = new BankAccount();

$account->setAccountNumber('1243845355');
echo sprintf("Bank account # %s<br/>",$account->getAccountNumber());

echo sprintf("Deposit $2000 to the bank account.<br/>");
$account->deposit(2000);
echo sprintf("Total balance %0.2f<br/>", $account->getBalance());

echo sprintf("Withdraw $100 from the bank account.<br/>");
$account->withdraw(100);
echo sprintf("Total balance %0.2f<br/>", $account->getBalance());

echo sprintf("Withdraw $2000 from the bank account.<br/>");
$account->withdraw(2000);
```

you have learned the basic PHP object-oriented programming (OOP). You also learned the two important building blocks of PHP OOP, objects, and classes.

6.2 CREATING AN OBJECT

It is much easier to create objects and use them than it is to define object classes, so before we discuss how to define classes, let's look at creating objects. To create an object of a given class, use the new keyword:

*$object = **new** Class;*

Assuming that a Person class has been defined, here's how to create a Person object:

```
$rasmus = new Person;
```

Do not quote the class name, or you'll get a compilation error:

```
$rasmus = new 'Person';                    // does not work
```

Some classes permit you to pass arguments to the new call. The class's documentation should say whether it accepts arguments. If it does, you'll create objects like this:

```
;(object = new Person('Fred', 35$
```

The class name does not have to be hardcoded into your program. You can supply :the class name through a variable

```
$class = 'Person';
$object = new $class;
// is equivalent to
$object = new Person;
```

Specifying a class that doesn't exist causes a runtime error.

Variables containing object references are just normal variables—they can be used in the same ways as other **variables**. Of particular note is that variable variables work with objects, as shown here:

```
$account = new Account;
$object = 'account'
${$object}->init(50000, 1.10);  // same as $account->init
```

Remember

You enclose a class using curly braces ({ }) … just like you do with functions.

6.2.1 Accessing Properties and Methods

Once you have an object, you can use the -> notation to access methods and properties of the object:

object->propertyname$

([... ,object->methodname([arg$

example

```
printf("Rasmus is %d years old.\n", $rasmus->age);  // property access
$rasmus->birthday();                                 // method call
$rasmus->set_age(21);                                // method call with arguments
```

Methods are functions, so they can take arguments and return a value:

```
$clan = $rasmus->family('extended');
```

Keyword

Variables are used to store information to be referenced and used by programs.

PHP does not have the concept of private and public methods or properties. That is, there's no way to specify that only the code in the class should be able to directly access a particular property or method. Encapsulation is achieved by convention—only an object's code should directly access its properties—rather than being enforced by the language itself.

You can use variable variables with property names:

```
$prop = 'age';
echo $rasmus->$prop;
```

A static method is one that is called on a class, not on an object. Such methods cannot access properties. The name of a static method is the class name, followed by two colons and the function name. For instance, this calls the p( ) method in the HTML class:

HTML::p("Hello, world");

A class's documentation tells you which methods are static.

Assignment creates a copy of an object with identical properties. Changing the copy does not change the original:

```
$f = new Person('Fred', 35);
$b = $f;                              // make a copy
$b->set_name('Barney');               // change the copy
printf("%s and %s are best friends.\n", $b->get_name(),
$f->get_name( ));
```

Barney and Fred are best friends.

Class names are case-insensitive and must conform to the rules for PHP identifiers.

6.3 DECLARING A CLASS

To design your program or code library in an object-oriented fashion, you will need to define your own classes, using the class keyword. A class definition includes the class name and the properties and methods of the class. The class name stdClass is reserved. Here's the syntax for a class definition:

```
class classname [ extends baseclass ]
{
    [ var $property [ = value ]; ... ]

    [ function functionname (args) {
          // code
      }
      ...
    ]
}
```

6.3.1 Declaring Methods

A method is a function defined inside a class. Although PHP imposes no special restrictions, most methods act only on data within the object in which the method resides. Method names beginning with two underscores (__) may be used in the future by PHP (and are currently used for the object serialization methods _ _sleep( ) and _ _wakeup( )), so it is recommended that you do not begin your method names with this sequence.

Within a method, the $this variable contains a reference to the object on which the method was called. For instance, if you call $rasmus->birthday( ), inside the birthday( ) method, $this holds the same value as $rasmus. Methods use the $this variable to access the properties of the current object and to call other methods on that object.

Here's a simple class definition of the Person class that shows the $this variable in action:

```
class Person {
    var $name;

    function get_name ( ) {
        return $this->name;
    }

    function set_name ($new_name) {
        $this->name = $new_name;
    }
}
```

Keyword

Static method is a method that belongs to a class rather than an instance of a class.

As you can see, the get_name( ) and set_name( ) methods use $this to access and set the $name property of the current object.

There are no keywords or special syntax for declaring a **static method**. A static method simply doesn't use $this, because the method is called on a class and not on an object. For example:

```
class HTML_Stuff {
  function start_table( ) {
    echo "<table border='1'>\n";
  }
  function end_table ( ) {
    echo "</table>\n";
  }
}
HTML_Stuff->start_table( );
// print HTML table rows and columns
HTML_Stuff->end_table( );
```

6.3.2 Declaring Properties

In the definition of the Person class, we explicitly declared the $name property. Property declarations are optional and are simply a courtesy to whoever maintains your program. It's good PHP style to declare your properties, but you can add new properties at any time.

Here's a version of the Person class that has an undeclared $name property:

```
class Person {
    function get_name ( )
    {
        return $this->name;    }

    function set_name ($new_name) {
        $this->name = $new_name;
    }
}
```

You can assign default values to properties, but those default values must be simple constants:

```
var $name = 'J Doe';        // works
var $age  = 0;              // works
var $day  = 60*60*24;       // doesn't work
```

Did You Know?

PHP provides a set of special predefined constants and magic methods to your programs. Unlike the constants you would set using define(), the value of the constants depend on where they are used in your code and are used to access read-only information about your code and PHP.

6.3.3 Inheritance

To inherit the properties and methods from another class, use the extends keyword in the class definition, followed by the name of the base class:

```
class Person {
  var $name, $address, $age;
}

class Employee extends Person {
  var $position, $salary;
}
```

The Employee class contains the $position and $salary properties, as well as the $name, $address, and $age properties inherited from the Personclass.

I f a derived class has a property or method with the same name as one in its parent class, the property or method in the derived class takes precedence over, or *overrides*, the property or method in the parent class. Referencing the property returns the value of the property on the child, while referencing the method calls the method on the child.

Keyword

Clause consists of a subject and a verb and is the smallest grammatical unit that expresses a thought.

To access an overridden method, use the parent::*method*( ) notation:

```
parent::birthday();          // call parent class's birthday( ) method
```

A common mistake is to hardcode the name of the parent class into calls to overridden methods:

```
Creature::birthday( );       // when Creature is the parent class
```

This is a mistake because it distributes knowledge of the parent class's name all over the derived class. Using parent:: centralizes the knowledge of the parent class in the extends **clause**.

6.3.4 Constructors

You may also provide a list of arguments following the class name when instantiating an object:

```
$person = new Person('Fred', 35);
```

These arguments are passed to the class's *constructor*, a special function that initializes the properties of the class.

A constructor is a function with the same name as the class in which it is defined. Here's a constructor for the Person class:

```
class Person {
    function Person ($name, $age) {
        $this->name = $name;
        $this->age  = $age;
    }
}
```

PHP does not provide for an automatic chain of constructors; that is, if you instantiate an object of a derived class, only the constructor in the derived class is automatically called. For the constructor of the parent class to be called, the constructor in the derived class must explicitly call the constructor. In this example, the Employee class constructor calls the Person constructor:

```
class Person {
  var $name, $address, $age;

  function Person($name, $address, $age) {
    $this->name = $name;
    $this->address = $address;
    $this->age = $age;
  }
}

class Employee extends Person {
  var $position, $salary;

   function Employee($name, $address, $age, $position,
$salary) {
    $this->Person($name, $address, $age);
    $this->position = $position;
```

Remember

Don't forget that in a class, variables are called 'properties' and functions are called 'methods'.

```
        $this->salary = $salary;
      }
    }
```

6.3.5 References

When you assign an object to another variable, you create a copy:

```
$fred = new Person;
$copy = $fred;
$fred->name("Fred");
print $copy->name();       // does not print "Fred"
```

You now have two Person objects, $fred and $copy, with independent property values. This is also the case when you assign the results of a call to a constructor, as shown here:

```
$fred = new Person;
```

The object created by the Person constructor is copied, and the copy is stored in $fred. This means that $this in the constructor and $fred actually refer to two different objects. If the constructor creates an alias to $this through a reference, it won't create an alias to $fred. For example:

```
$people = array();
class Person {
    function Person () {
      global $people;
      $people[] =& $this;
    }
}
$fred = new Person;
$fred->name = "Fred";
$barney =& new Person;
$barney->name = "Barney";
var_dump($people);
array(2) {
   [0]=>
   &object(person)(0) {
{
```

```
    [1]=>
    &object(person)(1) {
      ["name"]=>
      string(6) "Barney"
    }
}
```

$fred is a copy of the object that the constructor stored in $people[0], while $barney is an alias for the object that the constructor stored in $people[1]. When we change the properties of $fred, we're not changing the object that is in $people[0]. However, when we change the properties of $barney, we are changing the object in $people[1].

To prevent copying on assignment, assign by reference:

```
$obj =& new Class;
```

This code makes $obj an alias for the new object, which was $this in the constructor. If the constructor stores a reference to $this, it keeps a reference to $obj.

The documentation for a class should say whether you need to use =& with its constructor. In most cases, this isn't necessary.

CASE STUDY

CUBE EVOLUTION

Cube Evolution is a UK-based software company that produces custom software and web designs. One of its products is a Web Content Management System, which allows individuals and companies to easily maintain their own websites.

As originally designed, however,...

Cube Evolution is a UK-based software company that produces custom software and web designs. One of its products is a Web Content Management System, which allows individuals and companies to easily maintain their own websites.

As originally designed, however, the CMS software was not secure. Any customer who purchased the system could easily see the source code and take any of the programming language, leaving the company vulnerable to software theft. Customers could also make changes to the code that could potentially cause the product to malfunction. This could have resulted in the company's support department becoming overwhelmed with calls from users who had altered the program.

SourceGuardian provided Cube Evolution with a solution that protects its CMS software from theft and alteration, while allowing customers to continue to license the product for easy use. In addition, SourceGuardian allows Cube Evolution to produce a demonstration version of its CMS software that will work for only a specified time, offering the company a way to promote its product without giving it away or exposing its programming.

SUMMARY

- In PHP, Object is a compound data type (along with arrays). Values of more than one types can be stored together in a single variable.
- In object oriented programming, a class is a definition of an object, whereas an object is an instance of an object, meaning that from one class you can create many objects.
- In object-oriented terminology, characteristics of an object are called properties.
- To create a method for a class you use the function keyword followed by the name and parentheses.
- To design your program or code library in an object-oriented fashion, you will need to define your own classes, using the class keyword.
- A constructor is a function with the same name as the class in which it is defined.
- Property declarations are optional and are simply a courtesy to whoever maintains your program.
- A class definition includes the class name and the properties and methods of the class.

KNOWLEDGE CHECK

1. **Which version of PHP introduced the static keyword?**
   a. PHP 4
   b. PHP 5
   c. PHP 5.2
   d. PHP 5.3
2. **Which keyword is used to access a static method or property from within the same class(rather than from child)?**
   a. static
   b. strat
   c. self
   d. set
3. **What does PDO stand for?**
   a. PHP Data Orientation
   b. PHP Database Object
   c. PHP Database Orientation
   d. PHP Data Object
4. **Which version of PHP allows you to define constant properties within a class?**
   a. PHP 4
   b. PHP 4.1
   c. PHP 4.3
   d. PHP 5
5. **Which one of the following is a constant variable?**
   a. const $name
   b. const $NAME
   c. constant NAME
   d. const NAME
6. **Which one of the following functions is used to determine object type?**
   a. obj_type()
   b. type()
   c. is_a()
   d. is_obj()

7. Which one of the following can be used to instantiate an object in PHP assuming class name to be LFC?

a. $obj = new $LFC;

b. $obj = new LFC ();

c. $obj = new LFC;

d. obj = new LFC ();

REVIEW QUESTIONS

1. What is an object? Explain with example.
2. What are constructor and destructor in PHP?
3. Write a Program for finding the smallest number in an array.
4. Write a program to find table of a number.
5. Write a Program for finding the biggest number in an array without using any array functions.

Check Your Result

1. (b) 2. (c) 3. (d) 4. (d) 5. (d)
6. (c) 7. (b)

REFERENCES

1. B. E. Aydemir, A. Bohannon, M. Fairbairn, J. N. Foster, B. C. Pierce, P. Sewell, D. Vytiniotis, G. W. S. Weirich, and S. Zdancewic. Mechanized metatheory for the masses: The POPLmark challenge. May 2005.
2. C. Anderson and S. Drossopoulou. BabyJ - from object based to class based programming via types. In WOOD '03, volume 82. Elsevier, 2003.
3. C. Chambers and the Cecil Group. The Cecil language: Specification and rationale. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, 2004.
4. D. Jamwal, "Analysis of Software Quality Models for Organizations," 2010.
5. D. Rosmala, M. Ichwan, and M. I. Gandalisha, "Komparasi Framework MVC (Codeigniter, dan CakePHP) pada Aplikasi Berbasis Web (Studi Kasus: Sistem Informasi Perwalian di Jurusan Informatika Institut Teknologi Nasional)," J. Inform., vol. 2, no. 2, pp. 22–30, 2011.
6. Muktamyee Sarker, "An overview of Object Oriented Design Metrics Master Thesis," Computer (Long. Beach. Calif)., 2005.
7. R. A. Khan, K. Mustafa, and S. I. Ahson, "An Empirical Validation of Object Oriented Design Quality Metrics," J. King Saud Univ. - Comput. Inf. Sci., vol. 19, pp. 1–16, 2013.
8. R. Anggrainingsih, B. O. P. Johannanda, A. P. Kuswara, D. Wahyuningsih, and T. Rejekiningsih, "Comparison of maintainability and flexibility on open source LMS," in Proceedings - 2016 International Seminar on Application of Technology for Information and Communication, ISEMANTIC 2016, 2017.
9. R. B. Findler and M. Felleisen. Contracts for higher-order functions. In ACM International Conference on Functional Programming, October 2002.
10. R. B. Findler, M. Flatt, and M. Felleisen. Semantic casts: Contracts and structural subtyping in a nominal world. In European Conference on Object-Oriented Programming, 2004.
11. S. M. Jamali, "Object Oriented Metrics (A Survey Approach) Seyyed Mohsen Jamali Department of Computer Engineering Sharif University of Technology January 2006 Tehran Iran," Design, 2006.
12. S. Misra, "Evaluation criteria for object-oriented metrics," Acta Polytech. Hungarica, vol. 8, no. 5, pp. 109–136, 2011.
13. T. L. Saaty, "Decision making — the Analytic Hierarchy and Network Processes (AHP/ANP)," J. Syst. Sci. Syst. Eng., 2004.
14. Y. Bres, B. P. Serpette, and M. Serrano. Compiling scheme programs to .NET common intermediate language. In 2nd International Workshop on .NET Technologies, Pilzen, Czech Republic, May 2004.

CHAPTER

7

WEB TECHNIQUES

"Technology is nothing. What's important is that you have a faith in people, that they're basically good and smart, and if you give them tools, they'll do wonderful things with them"

– Steve Jobs

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understanding the HTTP basics
2. Explain the maintaining state

INTRODUCTION

PHP is one of the most widely used server side scripting language for web development. Popular websites like

Facebook, Yahoo, Wikipedia etc., are developed using PHP. PHP is so popular because it is very simple to learn, code and deploy on server, hence it has been the first choice for beginners since decades.

PHP stands for Hypertext Pre-Processor. PHP is a scripting language used to develop static and dynamic webpages and web applications.

Modern businesses are not confined to any boundary thanks to websites that they use to reach any part of the world. Having a website has become important for any business to ensure its growth and to stay competitive. Choosing the right technology partner is the first step in making a successful website. Lots of technologies and methods have been devised in the past few years to develop websites but PHP has emerged as the topmost technology for this.

It is a premier choice for developing websites and web apps. Indeed it rules the web world with almost 79% of websites that utilize a server are using PHP. And, it is the prime underlying technology in lots of content management systems in the market today. PHP website development is a popular choice for lots of businesses.

PHP which is an acronym of Hypertext Preprocessor is an open-source and free server-side language. It is a general-purpose language that is developed and maintained by a huge community. PHP is all about handing the back-end side or the server-side of a website or web application architecture. Consequently, PHP code gets executed on the server and the client-side only gets the plain HTML code that is displayed to the users by a web browser.

PHP can load data from a server and put it into the HTML DOM but it doesn't have much control over HTML like JavaScript. It can be used to connect to different databases, apply scripts to HTML, start sessions, and perform many other tasks.

You can do a lot of things with PHP and create compelling web applications. And, this is the prime reason for the popularity of this server-side scripting language. PHP is widely used for:

- Add/modify database information.
- Create dynamic content.
- Create, write, open, delete, read, and close files on a server.
- Send or receive cookies.
- Collect data from forms.
- Perform data encryption
- PHP not only generates HTML output but it can also generate output as PDF, XML, Flash, and other files.

7.1 HTTP Basics

HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet).

HTTP concepts include (as the Hypertext part of the name implies) the idea that files can contain references to other files whose selection will elicit additional transfer requests. Any Web server machine contains, in addition to the Web page files it can serve, an HTTP daemon, a program that is designed to wait for HTTP requests and handle them when they arrive. Your Web browser is an HTTP client, sending requests to server machines. When the browser user enters file requests by either "opening" a Web file (typing in a Uniform Resource Locator or URL) or clicking on a hypertext link, the browser builds an HTTP request and sends it to the Internet Protocol address (IP address) indicated by the URL. The HTTP daemon in the destination server machine receives the request and sends back the requested file or files associated with the request.

Did You Know?

The term hypertext was coined by Ted Nelson in 1965 in the Xanadu Project, which was in turn inspired by Vannevar Bush's 1930s vision of the microfilm-based information retrieval and management "memex" system described in his 1945 essay "As We May Think".

7.1.1 File Upload

With PHP, it is easy to upload files to the server.

However, with ease comes danger, so always be careful when allowing file uploads!

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the file_uploads directive, and set it to On:

file_uploads = On

Create The HTML Form

Next, create an HTML form that allow users to choose the image file they want to upload:

```
<!DOCTYPE html>
<html>
<body>
<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>
</body>
</html>
```

Some rules to follow for the HTML form above:

Make sure that the form uses method="post"

The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form

Without the requirements above, the file upload will not work.

Other things to notice:

The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

The form above sends data to a file called "upload.php", which we will create next.

Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
```

```
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

PHP script explained:

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file (in lower case)
- Next, check if the image file is an actual image or a fake image

Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

Check if File Already Exists

Now we can add some restrictions.

First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

Limit File Size

The file input field in our HTML form above is named "fileToUpload".

Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:

```
// Check file size
```

```
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

Limit File Type

The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

Complete Upload File PHP Script

The complete "upload.php" file now looks like this:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
```

```
}
// Check if file already exists
if (file_exists($target_file)) {
    echo “Sorry, file already exists.”;
    $uploadOk = 0;
}
// Check file size
if ($_FILES[“fileToUpload”][“size”] > 500000) {
    echo “Sorry, your file is too large.”;
    $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != “jpg” && $imageFileType != “png” && $imageFileType != “jpeg”
&& $imageFileType != “gif” ) {
    echo “Sorry, only JPG, JPEG, PNG & GIF files are allowed.”;
    $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo “Sorry, your file was not uploaded.”;
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES[“fileToUpload”][“tmp_name”], $target_file)) {
        echo “The file “. basename( $_FILES[“fileToUpload”][“name”]). “ has been uploaded.”;
    } else {
        echo “Sorry, there was an error uploading your file.”;
    }
}
?>
```

7.1.2 Form Validation

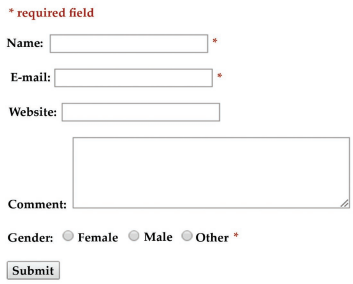
We have already explained about form validation using **JavaScript** and jQuery, but this time we will show you how to validate your form using PHP.

Very first we have to create a form in html setting action "#" and method "POST" with some fields, when a user clicks on submit button all the data starts travel in URL but it will be hidden, as we set method = "POST".

Now, on page refresh, PHP code runs, all values fetches from URL into PHP variables and aplying "preg_match ( )" function on them.

Keyword

JavaScript is a high-level, interpreted programming language.



PHP Code segment from validation.php

```
<?php
    // Initialize variables to null.
    $nameError ="";
    $emailError ="";
    $genderError ="";
    $websiteError ="";
    // On submitting form below function will execute.
    if(isset($_POST['submit'])){
    if (empty($_POST["name"])) {
    $nameError = "Name is required";
    } else {
    $name = test_input($_POST["name"]);
    // check name only contains letters and whitespace
```

```
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
$nameError = "Only letters and white space allowed";
}
}
if (empty($_POST["email"])) {
$emailError = "Email is required";
} else {
$email = test_input($_POST["email"]);
// check if e-mail address syntax is valid or not
if (!preg_match("/([w-]+@[w-]+.[w-]+)/",$email)) {
$emailError = "Invalid email format";
}
}
if (empty($_POST["website"])) {
$website = "";
} else {
$website = test_input($_POST["website"]);
// check address syntax is valid or not(this regular expression also allows dashes in the URL)
if (! preg_match("/b(?:(?:https?|ftp)://|www.)[-a-z0-9+&@#/%?=~_|!:,.;]*[-a-z0-9+&@#/%=~_|]/i",$website)) {
$websiteError = "Invalid URL";
}
}
if (empty($_POST["comment"])) {
$comment = "";
} else {
$comment = test_input($_POST["comment"]);
}
if (empty($_POST["gender"])) {
$genderError = "Gender is required";
} else {
$gender = test_input($_POST["gender"]);
}
```

```
}
function test_input($data) {
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}
//php code ends here
?>
```


```
<?php
$yourname = check_input($_POST['yourname']);
$email    = check_input($_POST['email']);
$likeit   = check_input($_POST['likeit']);
$comments = check_input($_POST['comments']);
?>
<html>
<body>
Your name is: <?php echo $yourname; ?><br />
Your e-mail: <?php echo $email; ?><br />
<br />
Do you like this website? <?php echo $likeit; ?><br />
<br />
Comments:<br />
<?php echo $comments; ?>
</body>
</html>
```

```
<?php
function check_input($data)
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
```

HTML Code segment from "validation.php"

```
<!DOCTYPE html>
    <html>
    <head>
    <title>Form Validation with PHP - Demo Preview</title>
    <meta content="noindex, nofollow" name="robots">
    <link href="style.css" rel="stylesheet">
    </head>
    <body>
    <div class="maindiv">
    <div class="form_div">
    <div class="title">
    <h2>Form Validation with PHP.</h2>
    </div>
    <form action="validation.php" method="post">
    <h2>Form</h2>
    <span class="error">* required field.</span>
    Name:
    <input class="input" name="name" type="text" value="">
    <span class="error">* <?php echo $nameError;?></span>
    E-mail:
    <input class="input" name="email" type="text" value="">
    <span class="error">* <?php echo $emailError;?></span>
    Gender:
    <input class="radio" name="gender" type="radio" value="female">Female
    <input class="radio" name="gender" type="radio" value="male">Male
    <span class="error">*<?php echo $genderError;?></span>
    Website:
    <input class="input" name="website" type="text" value="">
    <span class="error"><?php echo $websiteError;?></span>
    Comment:
    <textarea cols="40" name="comment" rows="5">
    </textarea>
```

```
<input class="submit" name="submit" type="submit" value="Submit">
</form>
</div>
</body>
</html>
```

CSS File: style.css

```
@import "http://fonts.googleapis.com/css?family=Droid+Serif";
    /* Above line is to import Google font style */
    .maindiv{
    margin:0 auto;
    width:980px;
    height:500px;
    background:#fff;
    padding-top:20px;
    font-size:14px
    }
    .title{
    width:500px;
    height:70px;
    text-shadow:2px 2px 2px #cfcfcf;
    font-size:16px;
    text-align:center;
    font-family:'Droid Serif',serif
    }
    .form_div{
    width:70%;
    float:left
    }
    form{
    width:440px;
    border:1px dashed #ccc;
    padding:10px 30px 40px;
```

```
background-color:#f0f8ff;
font-family:'Droid Serif',serif
}
form h2{
text-align:center;
text-shadow:2px 2px 2px #cfcfcf
}
textarea{
width:250px;
height:60px;
border-radius:1px;
box-shadow:0 0 1px 2px #123456;
margin-top:10px;
padding:5px;
border:none
}
.input{
width:250px;
height:15px;
border-radius:1px;
box-shadow:0 0 1px 2px #123456;
margin-top:10px;
padding:5px;
border:none;
margin-bottom:20px
}
.submit{
color:#fff;
border-radius:3px;
background:#1F8DD6;
padding:5px;
margin-top:40px;
border:none;
```

```
width:100px;
height:30px;
box-shadow:0 0 1px 2px #123456;
font-size:16px
}
.error{
color:red
}
.radio{
width:15px;
height:15px;
border-radius:1px;
margin-top:10px;
padding:5px;
border:none;
margin-bottom:20px
}
.formget{
float:right;
margin-top:85px
}
```

Required and optional fields

So far we only worked with optional fields - in all previous examples the scripts worked fine if you didn't enter any data. However, many times you want to make input fields required.

This is an easy task, let's edit the check_input function from the previous page to read:

```
function check_input($data, $problem='')
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    if ($problem && strlen($data) == 0)
```

```
    {
        die($problem);
    }
    return $data;
}
```

We've added an extra parameter to the form: $problem. By default $problem is empty, but if you pass a value for $problem to the function and the length of entered data is 0 the script will stop executing (die) displaying the text passed as $problem parameter.

It's actually easier than it sounds. To validate data from field "yourname" we used this so far:

```
$yourname = check_input($_POST['yourname']);
```

It still works. But if you want to make "yourname" required you now simply add ,"Error message" to the function call, like this:

```
$yourname = check_input($_POST['yourname'],"Enter your name!");
```

Now if the "yourname" fields is empty when the form is submitted, the script will stop and display "Enter your name!" text.As easy as that, for optional fields use

```
$test = check_input($_POST['test']);
```

and for required fields add a comma and an error message before):

```
$test = check_input($_POST['test'], "My error message");
```

Here is the final code of our myform.php script where "Your name", and "Comments" fields are required, but "Your **e-mail**" field is not. If the name is missing you will get an error saying "Enter your name" and if the comments are .missing you will get "Write your comments" error message

```
<?php
$yourname = check_input($_POST['yourname'], "Enter your name");
$email    = check_input($_POST['email']);
$likeit   = check_input($_POST['likeit']);
$comments = check_input($_POST['comments'], "Write your comments");
```

Keyword

E-mail is a method of exchanging messages ("mail") between people using electronic devices.

```
?>
<html>
<body>
Your name is: <?php echo $yourname; ?><br />
Your e-mail: <?php echo $email; ?><br />
<br />
Do you like this website? <?php echo $likeit; ?><br />
<br />
Comments:<br />
<?php echo $comments; ?>
</body>
</html>
<?php
function check_input($data, $problem='')
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    if ($problem && strlen($data) == 0)
    {
        die($problem);
    }
    return $data;
}
?>
```

The die() PHP function just displays the error text. If you want a fancier error page that fits your website design we can add a custom error reporting function. In this example we will name it show_error:

```
<?php
$yourname = check_input($_POST['yourname'], "Enter your name");
$email    = check_input($_POST['email']);
$likeit   = check_input($_POST['likeit']);
$comments = check_input($_POST['comments'], "Write your comments");
?>
```

```
<html>
<body>
Your name is: <?php echo $yourname; ?><br />
Your e-mail: <?php echo $email; ?><br />
<br />
Do you like this website? <?php echo $likeit; ?><br />
<br />
Comments:<br />
<?php echo $comments; ?>
</body>
</html>
<?php
function check_input($data, $problem='')
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    if ($problem && strlen($data) == 0)
    {
        show_error($problem);
    }
    return $data;
}
function show_error($myError)
{
?>
    <html>
    <body>
    <b>Please correct the following error:</b><br />
    <?php echo $myError; ?>
    </body>
    </html>
<?php
```

```
exit();
}
?>
```

Now you can simply edit the HTML code inside show_error function as much as you like and place this PHP code where you want the error message to appear:

```
<?php echo $myError; ?>
```

Note that we printed $myError directly without passing it through check_input function. Why? Because this variable was declared inside the script so we can control exactly what it is set to and it is not possible to change it from outside the script (using either POST or GET parameters) and insert any malicious code.

We end the show_error function with exit (); which tells PHP to stop executing code after displaying the error.

7.2 MAINTAINING STATE

HTTP is a stateless protocol, which means that once a web server completes a client's request for a web page, the connection between the two goes away. In other words, there is no way for a server to recognize that a sequence of requests all originate from the same client.State is useful, though. You can't build a shopping-cart application, for example, if you can't keep track of a sequence of requests from a single user. You need to know when a user puts an item in his cart, when he adds items, when he removes them, and what's in the cart when he decides to check out.

To get around the Web's lack of state, programmers have come up with many tricks to keep track of state information between requests (also known as session tracking). One such technique is to use hidden form fields to pass around information. PHP treats hidden form fields just like normal form fields, so the values are available in the $_GET and $_POST arrays. Using hidden form fields, you can pass around the entire contents of a shopping cart. However, a more common technique is to assign each user a unique identifier and pass the ID around using a single hidden form field. While hidden form fields work in all browsers, they work only for a sequence of dynamically generated forms, so they aren't as generally useful as some other techniques.

Another technique is **URL** rewriting, where every local URL on which the user might click is dynamically modified to include extra information. This extra information is often specified as a parameter in the URL. For example, if you assign every user a unique ID, you might include that ID in all

If you make sure to dynamically modify all local links to include a user ID, you can now keep track of individual users in your application. URL rewriting works for all dynamically generated documents, not just forms, but actually performing the rewriting can be tedious.

The third and most widespread technique for maintaining state is to use cookies. A cookie is a bit of information that the server can give to a client. On every subsequent request the client will give that information back to the server, thus identifying itself. Cookies are useful for retaining information through repeated visits by a browser, but they're not without their own problems. The main problem is that most browsers allow users to disable cookies. So any application that uses cookies for state maintenance needs to use another technique as a fallback mechanism.

The best way to maintain state with PHP is to use the built-in session-tracking system. This system lets you create persistent variables that are accessible from different pages of your application, as well as in different visits to the site by the same user. Behind the scenes, PHP's session-tracking mechanism uses cookies (or URLs) to elegantly solve most problems that require state, taking care of all the details for you.

Remember

URL is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

7.2.1 Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users –

- Server script sends a set of cookies to the browser. For example, name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

The Anatomy of a Cookie

Cookies are usually set in an **HTTP** header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this –

HTTP/1.1 200 OK

Date: Fri, 04 Feb 2000 21:03:38 GMT

Server: Apache/1.3.9 (UNIX) PHP/4.0b3

Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;

path=/; domain=tutorialspoint.com

Connection: close

Content-Type: text/html

As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded. The expires field is an instruction to the browser to "forget" the cookie after the given time and date.

Keyword

HTTP is an application protocol for distributed, collaborative, and hypermedia information systems.

If the browser is configured to store cookies, it will then keep this information until the expiry date. If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server. The browser's headers might look something like this −

GET / HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)

Host: zink.demon.co.uk:1126

Accept: image/gif, */*

Accept-Encoding: gzip

Accept-Language: en

Accept-Charset: iso-8859-1, *, utf-8

Cookie: name=xyz

A PHP script will then have access to the cookie in the environmental variables $_COOKIE or $HTTP_COOKIE_VARS [] which holds all cookie names and values. Above cookie can be accessed using $HTTP_COOKIE_VARS["name"].

Setting Cookies with PHP

PHP provided setcookie() function to set a cookie. This function requires up to six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

setcookie(name, value, expire, path, domain, security);

Here is the detail of all the arguments –

- **Name** – This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value** – This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** – This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set, then cookie will automatically expire when the Web Browser is closed.
- **Path** – This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain** – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** – This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Following example will create two cookies **name** and **age** these cookies will be expired after one hour.

```
<?php
   setcookie("name", "John Watkin", time()+3600, "/","", 0);
   setcookie("age", "36", time()+3600, "/", "",  0);
?>
<html>

   <head>
      <title>Setting Cookies with PHP</title>
   </head>

   <body>
      <?php echo "Set Cookies"?>
   </body>

</html>
```

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables. Following example will access all the cookies set in above example.

```
<html>

   <head>
      <title>Accessing Cookies with PHP</title>
   </head>

   <body>

      <?php
         echo $_COOKIE[“name”]. “<br />”;

         /* is equivalent to */
         echo $HTTP_COOKIE_VARS[“name”]. “<br />”;

         echo $_COOKIE[“age”] . “<br />”;

         /* is equivalent to */
         echo $HTTP_COOKIE_VARS[“age”] . “<br />”;
      ?>

   </body>
</html>
```

You can use isset() function to check if a cookie is set or not.

```
<html>

   <head>
      <title>Accessing Cookies with PHP</title>
```

```
    </head>

    <body>

        <?php
            if( isset($_COOKIE[“name”]))
                echo “Welcome “ . $_COOKIE[“name”] . “<br />”;

            else
                echo “Sorry... Not recognized” . “<br />”;
        ?>

    </body>
</html>
```

Deleting Cookie with PHP

Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired

```
<?php
    setcookie( “name”, “”, time()- 60, “/”,””, 0);
    setcookie( “age”, “”, time()- 60, “/”,””, 0);
?>
<html>

    <head>
        <title>Deleting Cookies with PHP</title>
    </head>

    <body>
        <?php echo “Deleted Cookies” ?>
    </body>\
</html>
```

7.2.2 Sessions

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**. Before using any session variable make sure you have setup this path.

When a session is started following things happen –

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called **PHPSESSID** is automatically sent to the user›s computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c 7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes' duration.

Remember

This must occur before any HTML code is sent to the Web browser. Even a blank line before the <?php will qualify as sending information to the browser, and it will then be too late to use session_start.

Starting a PHP Session

A PHP session is easily started by making a call to the **session_start()** function. This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session_start()** at the beginning of the page.

Session variables are stored in associative array called **$_SESSION[]**. These variables can be accessed during lifetime of a

session. The following example starts a session then register a variable called **counter** that is incremented each time the page is visited during the session.

Make use of **isset()** function to check if session variable is already set or not.

Put this code in a test.php file and load this file many times to see the result –

```
<?php
   session_start();

   if( isset( $_SESSION['counter'] ) ) {
      $_SESSION['counter'] += 1;
   }else {
      $_SESSION['counter'] = 1;
   }

   $msg = "You have visited this page ".  $_SESSION['counter'];
   $msg .= "in this session.";
?>
<html>

   <head>
      <title>Setting up a PHP session</title>
   </head>

   <body>
      <?php  echo ( $msg ); ?>
   </body>
   </html>
```

It will produce the following result –

You have visited this page 1in this session.

Destroying a PHP Session

A PHP session can be destroyed by **session_destroy()** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use **unset ()** function to unset a session variable.

Here is the example to unset a single variable –

```
<?php
   unset($_SESSION['counter']);
?>
```

Here is the call which will destroy all the session variables –

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

Turning on Auto Session

You don't need to call start_session() function to start a session when a user visits your site if you can set **session.auto_start**variable to 1 in **php.ini** file.

Remember

Every developer should be familiar with good security practices, and apps should be designed with these practices in mind. A fundamental rule is to never trust data you receive from somewhere else. Another rule is to escape data before you send it somewhere else. Combined, these rules can be simplified to make up a basic tenet of security: filter input, escape output (FIEO).

Sessions without cookies

There may be a case when a user does not allow to store cookies on their machine. So there is another method to send session ID to the browser.

Alternatively, you can use the constant SID which is defined if the session started. If the client did not send an appropriate session cookie, it has the form session_name=session_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

```
<?php
   session_start();

   if (isset($_SESSION['counter'])) {
      $_SESSION['counter'] = 1;
   }else {
      $_SESSION['counter']++;
   }
```

```
    $msg = "You have visited this page ".  $_SESSION['counter'];
    $msg .= "in this session.";

    echo ( $msg );
?>

<p>
    To continue click following link <br />

    <a  href = "nextpage.php?<?php echo htmlspecialchars(SID); ?>">
</p>
```

It will produce the following result –

You have visited this page 1in this session.

To continue click following link

The **htmlspecialchars()** may be used when printing the SID in order to prevent XSS related attacks.

ROLE MODEL

VANNEVAR BUSH

Vannevar Bush was born on March 11, 1890, in Everett, Massachusetts, in a family of the Universalist minister—the Reverend Richard Perry Bush of Provincetown, MA, and Emma Linwood (Paine) Bush. Vannevar had two sisters. As a child, he was sickly and was occasionally bedridden for long periods of time. Still, he was self-confident and sometimes got into fights with other boys. He once said, "all of my recent ancestors [before my father] were sea captains, and they have a way of running things without any doubt. So it may have been partly that, and partly my association with my grandfather, who was a whaling skipper. That left me with some inclination to run a show once I was in it."

Bush did well in the primary school, where he showed an aptitude for math and a love to tinker. Later his father became a pastor at Chelsea, MA, where Vannevar attended the high school. At he same time he has a versatile workshop in his home. After graduation, he went off to *Tufts College* to study engineering. Half of his expenses were paid by a scholarship, to pay the other half, he worked as a tutor and aid in the math department. Vannevar studied earnestly and earned a master›s degree in the time it usually takes to earn a bachelor›s degree. In 1913 he graduated the college as M. S. and went to work for *General Electric*, testing electrical equipment. In 1914-1915 Bush went to taught mathematics at Jackson College.

In 1915 he entered the Massachusetts Institute of Technology (MIT) electrical engineering program. Spurred by the need for enough financial security to marry, Bush finished his thesis in less than a year. On 5th of September, 1916, he married at Chelsea, to Phoebe Davis, a Chelsea girl, whom he had known since Tufts. The family had two sons: Richard Davis Bush and John Hathaway Bush. At the end of the same 1916 he had earned a Doctor if Engineering degree (from MIT and Harvard University, jointly) and became an assistant professor of electrical engineering at Tufts.

While at Tufts Bush enjoyed his first experience as an inventor, securing a patent for a device for land surveying,

which he called *the profile tracer*. It looked something like a lawnmower with two bicycle wheels and a pendulum, and as was pushed over land, integrating and recording horizontal and vertical measurements. Vannevar thought it would be commercially successful, but it never caught on. The young man learned many things from this failure—to become a real engineer he needed to learn more than math and physics. He needed to learn how to effectively deal with people.

The same difficulties faced Bush during the WWI, when he was working for US Navy, to which he proposed his idea for a device, that would use magnetic fields to detect submarines. His device proved to be successful in testing, but Navy officials, did not deploy the device correctly and it proved virtually useless in combat. Bush again learned that a successful engineer also had to be a good politician.

In 1919, Bush left Tufts and went to MIT's electrical engineering department as a professor of electrical power transmission. Starting from 1925, Bush began construction of mechanical devices for performing integration. First was the *Product Intergraph*, next was the *Differential Analyzer*, an analog computer that could solve differential equations with as many as 18 independent variables. It actually used large gears and other mechanical parts to solve equations. In 1931, he completed the first differential analyzer (see the lower photo). At the same tine, Bush and team at MIT develop the "network analyzer", a system for setting up miniature versions of large and important electrical networks. Later Bush constructed an improved *Differential Analyzer*, built by electronic tubes, which was used intensively during the WWII.

Bush and his *Differential Analyzer*

In 1922, Bush, his college roommate, Laurence Marshall, and a colleague of Bush from the company Amrad (a small company for radio devices, where Vannevar served as a consultant), Charles G. Smith, set up the *Metals and Controls Corporation*, for producing thermostats and thermionic tubes. Later the company managed to market successfully a device, called the *S-tube*. This was a gaseous rectifier, invented by Charles Smith (the patent was purchased by Amrad), that greatly improved the efficiency of radios, eliminating the need for radio batteries. Bush made much money from the venture. The company, renamed *Raytheon* in 1925, became a large electronics company and defense contractor, and still exist now as a major American defense contractor and industrial corporation.

Bush also worked on developing machines that would automate human thinking. Specialization in just about every field of science was creating a glut of information. Something was needed to help sort through the growing store of accumulated knowledge. In the 1930s microfilm, which had been around for some decades, was growing in popularity as a storage device, especially among librarians. Bush, who was a photography enthusiast, was quite interested in this resurgent technology. He proposed to build a machine for the FBI that could review 1000 fingerprints a minute. They however turned him down. In 1938 Bush and John Howard built and patented *the rapid selector*, a machine designed for high-speed referencing of information, stored on microfilm. In

1945 Bush wrote an article, describing a device (called *memex*), the prototype of the modern hypertext systems.

Bush became vice-president and dean of engineering at MIT from 1932 until 1938. In 1938 he accepted a prestigious appointment as President of the Carnegie Institution of Washington. The institution spent $1.5 million annually on research. The presidency of the institution came with a lot of prestige. The president influenced the direction of research in the U.S. and informally advised the government on scientific matters.

In 1940 Bush was appointed Chairman of the President's *National Defense Research Committee*, and in 1941 Director of the *Office of Scientific Research and Development*. These two appointments made Bush a central figure in the development of nuclear fission and the *Manhattan Project* (the atomic bomb).

In 1944 President Roosevelt asks Bush for recommendations on the application of "lessons learned" from WWII to civilian, peace-time activities. Bush submits the article Science, the Endless Frontier in response to Roosevelt›s request. He outlined the importance of continued support for research. He called for a *National Research Foundation* that «should develop and promote a national policy for scientific research and scientific education, should support basic research in nonprofit organizations, should develop scientific talent in American youth by means of scholarships and fellowships, and should by contract and otherwise support long-range research on military matters". It was this proposal which set off events leading to the development of the National Science Foundation in 1950.

During the next decades Bush was appointed a Chairman or Director of many institutions of critical importance for USA: in 1946—Chairman of the Joint Research and Development Board of the War and Navy Departments; in 1947—Chairman of the Development Board of National Military Establishment; in 1947—director of AT&T; in 1957—Chairman of the MIT Corporation, from 1957 to 1962 he was chairman of the large pharmaceutical corporation Merck & Co., etc.

Vannevar Bush died from pneumonia after suffering a stroke on June 28, 1974, in Belmont, Massachusetts.

SUMMARY

- PHP is one of the most widely used server side scripting language for web development.
- Popular websites like Facebook, Yahoo, Wikipedia etc., are developed using PHP.
- PHP is so popular because it is very simple to learn, code and deploy on server, hence it has been the first choice for beginners since decades.
- HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet).
- The HTTP daemon in the destination server machine receives the request and sends back the requested file or files associated with the request.
- To get around the Web's lack of state, programmers have come up with many tricks to keep track of state information between requests (also known as session tracking).
- The best way to maintain state with PHP is to use the built-in session-tracking system.
- Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.
- PHP provided setcookie() function to set a cookie. This function requires up to six arguments and should be called before <html> tag. For each cookie this function has to be called separately.
- The location of the temporary file is determined by a setting in the php.ini file called session.save_path.

KNOWLEDGE CHECK

1. **For finding out one string is equal to another string which function we can use?**
   a. strpid ( )
   b. strpos ( )
   c. str ( )
   d. All of them
2. **Communication between browser and Web server takes place via**
   a. GUI
   b. HTTP
   c. ASP
   d. JSP
3. **Parsed HTML code of in-memory tree structure is defined by a standard, called**
   a. Dynamic Obtained Model
   b. Document Obtained Model
   c. Dynamic Object Model
   d. Document Object Model
4. **Several technologies that are used by JavaScript to create dynamic web pages, are known as**
   a. DOM
   b. Ajax
   c. MVC
   d. MAVC
5. **Client and Web server have a connection of type**
   a. Continuous
   b. Dependent
   c. Permanent
   d. Not Continuous
6. **The latest HTML standard is**
   a. XML
   b. SGML
   c. HTML 4.0
   d. HTML 5.0

7. While working on a JavaScript project, in your JavaScript application, which function would you use to send messages to users requesting for text input?

a. Display()

b. Prompt()

c. Alert()

d. GetInput()

REVIEW QUESTIONS

1. Describe the process to configure The "php.ini" File.
2. How to create the HTML form?
3. Explain the concept of "form validation".
4. What is cookie?
5. What is a session?

Check Your Result

1. (b) 2. (b) 3. (d) 4. (b) 5. (d)
6. (d) 7. (b)

REFERENCES

1. B.Vasavi, Y.V.Sreevani, & Priya, G. (2011, June). Hibernate Technology for an Efficient Business Application Extension. Journal of Global Research in Computer Science, 2(6), 118- 125. Retrieved from www.jgrcs.info
2. Dashrath Mane, K. C. (2013, july). The Spring Framework: An Open Source Java Platform for Developing Robust Java Applications. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 3(2), 138-143.
3. Dimitris K. Iakovidis, D. D. (2014, July). Open-Access Framework for Efficient Object-Oriented Development of Video Analysis Software. Journal of Software Engineering and Applications, 730-743. doi:10.4236/jsea.2014.78068.
4. Furtună, C. F. (2006). Enterprise Application Integration Using Java Technologies. Revista Informatica Economică, 9-17.
5. Jing Ni, L. Z. (2007, September). A Study of Document Management System Based on J2EE. Modern Applied Science, 1(3), 2-5. Retrieved from www.ccsenet.org/mas.html
6. Long, B. (2009, October). A Framework for Model Checking Concurrent Java Components. JOURNAL OF SOFTWARE, 4(8), 867-874. doi:10.4304/jsw.4.8.867-874
7. Marko Jurišić, D. K. (2014, September 17-19). Application Framework Development and Design Patterns: Current State and Prospects. Central European Conference on Information and Intelligent Systems, 306- 344.
8. Neha Munsi, N. S. (2014, October). Integration of Struts, Spring and Hibernate for an E-Commerce System. International Journal of Computer Science and Mobile Computing, 3(10), 853 – 859. Retrieved from www.ijcsmc.com
9. Oak, H. (2003, September 16). Choose the best Java framework for your application. Retrieved from http://www.techrepublic.com/article/choose-the-best-java-framework-for-your-application/.
10. R´emi Forax, E. D. (2005). A Reflective Implementation of Java Multi-Methods. IEEE Transactions on Software Engineering, 1-14.
11. Schwabe, D., Rossi, G., & Esmeraldo, L. (2001). Web Design Frameworks: An approach to improve reuse in Web applications. (Y. D. San Murugesan, Ed.) Web Engineering (Managing Diversity and Complexity of Web Application Development), 2016, 335-352. doi:10.1007/3-540-45144-7_32.

CHAPTER

8

DATABASE FUNCTIONS

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

-----Martin Fowler

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand dBase database file
2. Define DBM-style database abstraction
3. Discuss about relational database known as filePro
4. Explain the concept of Informix
5. Know InterBase database
6. Define Mini SQL and MySQL
7. Describe the term open database connectivity (ODBC)
8. Explain oracle database

INTRODUCTION

PHP offers support for many databases. Open source relational databases are well represented, as are many

commercial products. If native support for a database does not exist, it's likely you may use ODBC with an appropriate driver.

Most of the functions rely on an extension module. These may be loaded either in the php.ini file or the dl function, but most likely are compiled into PHP. Typically Windows requires use of the first method and other operating systems require the second.

We describe the PHP functions that communicate with various systems, it does not pursue introducing the intricacies of all the systems.

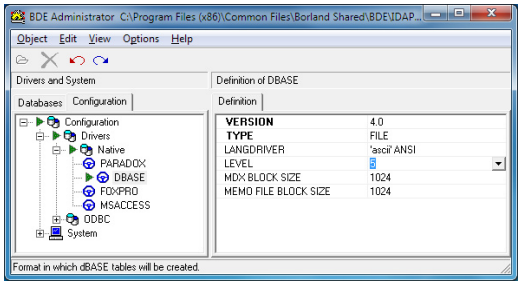
8.1 DBASE

The following functions work on dBase files, which typically end with a .dbf extension. If you are using the precompiled version for Windows, you will need to load the dBase extension by editing php.ini or using the dl function. The extension is likely called php_dbase.dll but was unavailable at the time of this writing. On other operating systems it's easy to compile dBase support into PHP.

Keyword

dBase was one of the first database management systems for microcomputers and the most successful in its day

The **dBase** functionality in PHP is somewhat limited. Index and memo fields are not supported. Neither is any kind of locking. The dBase functionality in PHP is meant to be a means of importing data from what has become somewhat of a lowest common denominator in data exchange.



Jim Winstead added dBase support to PHP.

boolean dbase_add_record(integer database, array record)

The dbase_add_record function adds a record to the database specified by a database identifier returned by dbase_open. The

record array contains an element for each field in the database, in order and starting at zero. If the correct number of fields is not supplied, FALSE is returned.

```
<?
  //open connection to database
  $db = dbase_open("customer.dbf", 2);

  //create record to be added
  $newRecord = array("John Smith", 100.00, "19980901","Y");

  //add record
  dbase_add_record($db, $newRecord);

  //close connection
  dbase_close($db);
?>
```

boolean dbase_close(integer database)

The dbase_close function closes a database.

integer dbase_create(string filename, array fields)

The dbase_create function creates a dBase database. The fields argument is an array of arrays that describe fields. Each array may have up to four elements. In order, they are name, type, length, and precision. Type is a single character. Some types require length, and precision; others do not.

Table 1. dBase Field Types

Type	Code	Description
Boolean	L	This type does not have a length or a precision.
Date	D	Dates are stored in YYYYMMDD format and do not use length or precision.
Number	N	The length property refers to the total number of digits in the number and the precision refers to the number of digits after the decimal point.
String	S	The length property specifies how many characters are stored in the field. The precision property is not used.

If a database is successfully created, a database identifier is returned; otherwise FALSE is returned.

```
<?
  // create field definition
  $fields = array(
    array("Name", "C", 32),
    array("Balance", "N", 8, 2),
    array("Birthday", "D"),
    array("Commercial", L));

  $db = dbase_create("customer.dbf", $fields);

  dbase_close($db);
?>
```

boolean dbase_delete_record(integer database, integer record)

The dbase_delete_record function marks a record for deletion. The record will remain in the database until dbase_pack is called.

```
<?
  //open connection to database
  $db = dbase_open("customer.dbf", 2);

  //mark record for deletion
  dbase_delete_record($db, 2);

  //close connection
  dbase_close($db);
?>
```

array dbase_get_record(integer database, integer record)

The dbase_get_record function returns the fields of a record in an array. The first field will be numbered zero. In addition, an element indexed by deleted will contain 1 if the row was marked for deletion. Records are numbered from one.

```
<?
  //connect to database
  $db = dbase_open("customer.dbf", 2);

  //get some information about database
  $numRecords = dbase_numrecords($db);
  $numFields = dbase_numfields($db);

  // get every record
  for($index = 1; $index = $numRecords; $index++)
  {
     //get a record
     $record = dbase_get_record($db, $index);
```

```
  {
     //get a record
     $record = dbase_get_record($db, $index);

     print("H3>Record $index/H3>\n");

     //loop over fields
     for($index2 = 0; $index2  $numFields;
$index2++)
     {
       print("B>Field $index2:/B>");
       print($record[$index2]);
       print("<BR>\n");

    }

     //print deletion status
     print("B>Deleted:/B> ");
     print($record["deleted"]);
     print("<BR>\n");
  }

  //close connection
  dbase_close($db);
?>
```

Keyword

Integer is colloquially defined as a number that can be written without a fractional component.

array dbase_get_record_with_names(integer database, integer record)

This function behaves like dbase_get_record, except that instead of being indexed by **integers**, fields are indexed by their names.

```
 <?
   //connect to database
   $db = dbase_open("customer.dbf", 2);

   // get every record
   for($index = 1; $index = dbase_numrecords($db);
 $index++)
   {
     $record = dbase_get_record_with_names($db, $index);
     print("H3>Record $index/H3>\n");


     //loop over fields
     while(list($key, $value) = each($record))
     {
       print("B>Field $key: $value/B><BR>\n");
     }
  }

  //close connection
  dbase_close($db);
?>
```

integer dbase_numfields(integer database)

The dbase_numfields function returns the number of fields for the given database.

integer dbase_numrecords(integer database)

The dbase_numrecords function returns the number of records in the database.

integer dbase_open(string filename, integer mode)

Use dbase_open to get a dbase identifier. This integer is needed for identifying which database to operate on. The mode may be 0 for read-only, 1 for write-only, or 2 for allowing both reading and writing. FALSE is returned if the database cannot be opened.

boolean dbase_pack(integer database)

When rows are deleted in a dbase database, they are simply marked for deletion. Use the dbase_pack function to permanently remove these rows, thus packing the database.

```
<?
  //connect to database
  $db = dbase_open("customer.dbf", 2);

  //removed rows marked for deletion
  dbase_pack($db);

  //close connection
  dbase_close($db);
?>
```

boolean dbase_replace_record(integer database, array record, integer record_number)

Use dbase_replace_record to change the contents of a record. The record argument must have one element for each field defined in the database. Record numbers start counting at one.

```
<?
  $db = dbase_open("customer.dbf", 2);

  $Record = array("John Smith", 200.00, "19990901","Y");
  dbase_replace_record($db, $Record, 1);

  dbase_close($db);
?>
```

8.2 DBM-STYLE DATABASE ABSTRACTION

The DBA functions abstract communications with databases that conform to the style of Berkeley DB database systems. Rather than storing complex records, a DBM database simply stores key/value pairs. This is similar to an associative array.

The functions replace a set of functions that allow just one type of DBM database. These new functions allow for choosing the underlying system from within your PHP code rather than compiling PHP for a single DBM implementation. You choose a type of database when you open a connection, and the rest of the functions perform accordingly. Sascha Schumann added these functions to PHP.

Keyword

Database connection is a facility in computer science that allows client software to talk to database server software, whether on the same machine or not.

dba_close(integer link)

The dba_close function closes a link to a database. The link argument is an integer returned by the dba_open or dba_popen functions. If you choose not to close a **database connection**, PHP will close it for you.

boolean dba_delete(string key, integer link)

The dba_delete function removes an entry from a database. You must supply both the key and a valid link to a database, as supplied by dba_open or dba_popen. The success of the delete is returned as a boolean.

```
<?
  // open database in write mode
  $db = dba_popen('inventory', 'w', 'gdbm');

  if($db)
  {
    //check for record
```

```
    if(dba_exists('3', $db))
    {

      // remove item 3
      dba_delete('3', $db);
    }
    else
    {
      print('Record does not exist');
    }

    // close database
    dba_close($db);
  }
  else
  {
    print('Database does not exist');
  }
?>
```

boolean dba_exists(string key, integer link)

The dba_exists function tests for the presence of a key. The link argument must be an integer returned by the dba_open or dba_popen functions. The description of dba_delete has an example of using dba_exists.

string dba_fetch(string key, integer link)

Use the dba_fetch function to retrieve a record.

```
<?
  // open database in write mode
  $db = dba_popen('inventory', 'r', 'gdbm');

  if($db)
  {
    //loop over each record
    for($key = dba_firstkey($db); $key; $key=dba_nextkey($db))
    {
      print("$key = ");

      //fetch this record
      print(dba_fetch($key, $db));

     print("<BR>\n");
    }
  }
  else
  {
    print('Database does not exist');
  }

?>
```

string dba_firstkey(integer link)

The dba_firstkey function returns the first key in the database. If the database is empty, FALSE will be returned. As the example for dba_fetch shows, dba_firstkey and dba_nextkey may be used to traverse the entire database.

boolean dba_insert(string key, string value, integer link)

Use dba_insert to add a record to the database. The success of the insert is returned. Trying to insert a record that already exists is not allowed. If you need to update a record, use dba_replace.

```
<?
  // open database in write mode
  $db = dba_popen('inventory', 'w', 'gdbm');

  if($db)
  {
    //check for record
    if(dba_exists('3', $db))
    {
      //item 3 exists, set inventory to 150
      dba_replace('3', '150', $db);
    }
    else
    {
     //item 3 doesn't exists, insert it
     dba_insert('3', '150', $db);
   }

   // close database
   dba_close($db);
 }
 else
 {
   print('Database does not exist');
 }
?>
```

string dba_nextkey(integer link)

The dba_nextkey function returns the next key from the database. When there are no keys left, false is returned. The description of dba_fetch shows a typical use of dba_nextkey and dba_firstkey together.

integer dba_open(string filename, string mode, string type, ...)

Use dba_open to establish a link to a dbm-style database. A positive integer will be returned if the open is successful, FALSE if it fails. The filename argument is simply the path to a database. The mode argument can be one of four characters that control input and output of data. Table 2 lists the four modes.

Table 2. DBA open modes

Mode	Description
c	If the database does not exist, it will be created. Reads and writes may be performed.
n	If the database does not exist, it will be created. If it does exist, all records will be deleted. Reads and writes may be performed.
r	Only reads may be performed.
w	Reads and writes may be performed. If the file does not exist, an error occurs.

The type argument chooses the underlying database engine. Table 3 describes the four types. You may also supply any number of optional arguments that will be passed directly to the underlying engine.

Table 3. DBA database engine codes

Code	Description
dbm	This code represents the original style of DBM database as developed at Berkeley.
ndbm	This code stands for a newer version of the DBM standard with less restrictions than dbm.
gdbm	The GNU Database Manager is the result of project by GNU. You can download gdbm from the GNU FTP server < ftp://ftp.gnu.org/gnu/gdbm>.
db2	This code stands for a database package developed by Sleepycat software that is based on the original Berkeley source code. In fact, the founders wrote the original DBM at Berkeley. You can get more information and download software at their Web site: .
cdb	CDB is a package for creating constant databases—that is, databases that are read created and read from only. This offers a performance advantage with the tradeoff being that none of the writing functions work. To download the sofware, visit < ftp://koobera.math.uic.edu/www/cdb.html >.

When your script finishes executing, the database link will close automatically. You may choose to close it sooner with dba_close, and this may save some small amount of memory. Contrast this function to dba_popen, which attempts to reuse links.

boolean dba_optimize(integer link)

Use dba_optimize to optimize a database, which usually consists of eliminating gaps between records created by deletes. This function returns TRUE on success. Some underlying engines do not support optimizations, in which case this function will have no effect

```
<?
  // open database in write mode
  $db = dba_popen('inventory', 'w', 'gdbm');

  if($db)
  {
    //optimize database
    dba_optimize($db);

    // close database
    dba_close($db);
  }
  else
  {
    print('Database does not exist');
  }
?>
```

integer dba_popen(string filename, string mode, string type, ...)

The dba_popen function behaves identically to dba_open with one difference: Links are not closed. They remain with the process until the process ends. When you call dba_popen, it first tries to find an existing link. Failing that, it will create a new link. You never call dba_close on a link returned by dba_popen.

Since the links are pooled on per-process basis, this functionality offers no benefit when using PHP as a stand-alone executable. When used as an Apache module, there may be some small performance benefit due to the unique way Apache uses child processes.

boolean dba_replace(string key, string value, integer link)

Use dba_replace to update the value of an existing record. As with the other DBA functions, a valid link as returned by dba_open or dba_popen should be used for the link argument.

boolean dba_sync(integer link)

The dba_sync function will synchronize the view of the database in memory and its image on the disk. As you insert records, they may be cached in memory by the

underlying engine. Other processes reading from the database will not see these new records until **synchronization**.

Keyword

Synchronization is the capability to control the access of multiple threads to any shared resource.

```
<?
   // open database in write mode
   $db = dba_popen('inventory', 'w', 'gdbm');


  if($db)
  {
    for($n=1; $n=10; $n++)
    {
      //insert row
      dba_insert($n, '', $db);

      //synchronize
      dba_sync($db);
    }

    // close database
    dba_close($db);
  }
  else
  {
    print('Database does not exist');
  }
?>
```

Keyword

filePro is a proprietary DBMS and RAD system originally developed by Howard Wolowitz as The Electric File Clerk in 1978.

8.3 FILEPRO

filePro is a relational database by fP Technologies. There are versions for win32 and SCO UNIX. PHP only supports reading from filePro databases. Further information can be found at the filePro Web site. To enable filePro functions dynamically, use the dl function to load the appropriate extension. You can also compile **filePro** support into the PHP module. Chad Robinson wrote the filePro extension.

boolean filepro(string directory)

The filepro function starts a connection to a map file. Information about the database is stored in memory and used by the other FilePro functions.

```
<?
  //get information about database
  filepro("/fp/store");

  print("TABLE>");

  //create headers that contain
  //field names, type, width
  print("TR>\n");
  for($col=1; $col = filepro_fieldcount(); $col++)
  {
    print("TH>");
    print(filepro_fieldname($col));
    print(" ");
    print(filepro_fieldtype($col));
    print(" ");
    print(filepro_fieldwidth($col));

   print("/TH>");
  }
  print("/TR>\n");


  //loop over each row
  for($row=1; $row = filepro_rowcount(); $row++)
  {
    print("TR>\n");

    //output fields
    for($col=1; $col = filepro_fieldcount(); $col++)
    {
      print("TD>");
      print(filepro_retrieve($row, $col));
      print("/TD>");
    }

    print("/TR>\n");
  }

  print('/TABLE>');

?>
```

boolean filepro(string directory)

The filepro function starts a connection to a map file. Information about the database is stored in memory and used by the other FilePro functions.

```
<?
  //get information about database
  filepro("/fp/store");

  print("TABLE>");

  //create headers that contain
  //field names, type, width
  print("TR>\n");
  for($col=1; $col = filepro_fieldcount(); $col++)
  {
   print("TH>");
   print(filepro_fieldname($col));
   print(" ");
   print(filepro_fieldtype($col));
   print(" ");
   print(filepro_fieldwidth($col));

   print("/TH>");
 }
 print("/TR>\n");


 //loop over each row
 for($row=1; $row = filepro_rowcount(); $row++)
 {
   print("TR>\n");

   //output fields
   for($col=1; $col = filepro_fieldcount(); $col++)
   {
      print("TD>");
      print(filepro_retrieve($row, $col));
      print("/TD>");
    }

    print("/TR>\n");
  }

  print('/TABLE>');

?>
```

integer filepro_fieldcount()

Use filepro_fieldcount to find the number of fields.

string filepro_fieldname(integer field_number)

The filepro_fieldname function returns the name of the field for the given field number.

string filepro_fieldtype(integer field_number)

The filepro_fieldtype function returns the edit type for the given field number.

string filepro_retrieve(integer row, integer field)

The filepro_retrieve function returns the value for the specified field on the specified row.

integer filepro_rowcount()

Use filepro_rowcount to find the number of rows.

integer filepro_fieldwidth(integer field_number)

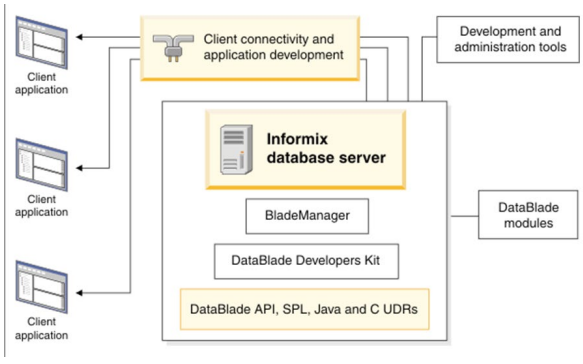
Use the filepro_fieldwidth function to find the width of the specified field.

8.4 INFORMIX

Informix makes several specialized relational database servers for Windows NT and UNIX. Like Oracle and Sybase, Informix products are intended for demanding situations. We cannot begin to discuss the unique features of Informix database servers, but you can learn more about them at their home site.

PHP includes support for two parts of the Informix API, ODS and IUS. The functions that begin with ifx_, such as ifx_pconnect, are part of ODS and should be available to you. The IUS functions begin with ifxus_, such as ifxus_create_slob. These will be available only if you have IUS libraries. Keep in mind that ODBC drivers for Informix are available, so you could pursue using PHP's ODBC functions instead.

Jouni Ahto, Christian Cartus, and Danny Heijl collaborated to create the Informix extension.



integer ifx_affected_rows(integer result)

The ifx_affected_rows function returns the number of rows selected, inserted, updated, or deleted, depending on the query. If the query was a select, the number is only an estimate. You can use this function on a result identifier returned by ifx_prepare in order to avoid executing queries that will return large result sets.

boolean ifx_blobinfile_mode(integer mode)

Use ifx_blobinfile function to control how to work with blobs. If mode is 0, blobs are saved in memory. If mode is 1, blobs are saved to disk.

boolean ifx_byteasvarchar(integer mode)

Use ifx_byteasvarchar to control how byte blobs are returned in queries. If mode is 0, blob IDs are returned. If mode is 1, blob contents are returned.

boolean ifx_close(integer link)

The ifx_close function closes a database connection created by ifx_connect. If the optional link argument is left out, the last-opened connection is closed.

integer ifx_connect(string database, string user, string password)

The ifx_connect function returns a connection to an Informix database. All of the arguments are optional and will draw their values from the php.ini file if necessary. If a connection cannot be established, FALSE is returned. If you attempt to connect again after successfully connecting, the original connection identifier is returned. The connection will be closed automatically when the script ends, but you can close it manually with ifx_close. The ifx_pconnect function creates a persistent connection.

integer ifx_copy_blob(integer blob)

The ifx_copy_blob function makes a copy of an existing blob and returns the identifier to the new blob.

integer ifx_create_blob(integer type, integer mode, string data)

The ifx_create_blob function creates a blob in the database. The type argument can be 1 for text or 0 for byte. The mode argument is set to 0 if the data argument contains data to place in the blob. The mode is set to 1 if the data argument is a path to a file.

```
<?
  //connect to database
  if(!($dbLink = ifx_pconnect("mydb@ol_srv1", "
leon", "secret"))
  {
    print("Unable to connect!<BR>\n");
    exit();
  }

  //create blob and add to array
  $blob[] = ifx_create_blob(0, 0, "This is a message");

  //insert message
  $Query = "INSERT INTO message " .
    "VALUES (3,'My Title', ?)";

  if(!($result = ifx_query($Query, $dbLink, $blob))
  {
    print("Unable to insert message!<BR>\n");

    //print Informix error message
    print(ifx_error() . "<BR>\n");
    print(ifx_errormsg() . "<BR>\n");
  }

  //free result identifier
  ifx_free_result($result);

  //close connection
  ifx_close($dbLink);
?>
```

integer ifx_create_char(string data)

The ifx_create_char function creates a character object. An identifier for the character object is returned.

boolean ifx_do(integer result)

The ifx_do function executes a query prepared with ifx_prepare. The result argument must be as returned by ifx_prepare.

string ifx_error()

Use ifx_error to fetch the error produced by the last query. The first character of the returned string is a flag reporting the type of condition. A space means no error. E is an error. N means no more data is available. W is a warning. A ? signals an unknown error condition. If anything other than a space is returned in the first character, the string will contain extra information, including an error code.

string ifx_errormsg(integer error)

Use ifx_errormsg to fetch a description of an error given its code. This is the same numerical code returned by the ifx_error function. If no error code is supplied, the description of the last error is returned.

array ifx_fetch_row(integer result, integer position) array ifx_fetch_row(integer result, string position)

The ifx_fetch_row function returns a row from a result set after executing a select. The returned array will contain elements named in the query indexed by column names. If cursor type was set using IFX_SCROLL, then you may use the position argument. This can be an integer, or one of these strings: FIRST, NEXT, LAST, PREVIOUS, CURRENT.

array ifx_fieldproperties(integer result)

The ifx_fieldproperties function returns an array containing information about each column in a result set. The elements of the array are indexed by the column names. Each element contains a list of properties separated by semicolons. The parts of this string are type, length, precision, scale, and a flag for whether the column can be null. The type is one of the strings listed in ifx_fieldtypes.

array ifx_fieldtypes(integer result)

The ifx_fieldtypes function returns an array describing the type for eachcolumn in the result set. Possible values are SQLBOOL, SQLBYTES, SQLCHAR, SQLDATE, SQLDECIMAL, SQLDTIME, SQLFLOAT, SQLINT, SQLINT8, SQLINTERVAL, SQLLVARCHAR, SQLMONEY, SQLNCHAR, SQLNVCHAR, SQLSERIAL, SQLSERIAL8, SQLSMFLOAT, SQLSMINT, SQLTEXT, SQLUDTFIXED, SQLVCHAR. The returned array is indexed by the names of the columns.

boolean ifx_free_blob(integer blob)

The ifx_free_blob function deletes the specified blob from the database.

boolean ifx_free_char(integer character)

The ifx_free_char function deletes a character object from the database.

boolean ifx_free_result(integer result)

The ifx_free_result function frees memory associated with a result set.

string ifx_get_blob(integer blob)

The ifx_get_blob function returns the contents of a blob.

array ifx_getsqlca(integer result)

The ifx_getsqlca function returns an array of the values in the sqlerrd struct from the underlying Informix API.

integer ifx_htmltbl_result(integer result, string options)

The ifx_htmltbl_result function prints an HTML table containing all the rows in the result set. The optional options argument will be placed inside the table tag.

boolean ifx_nullformat(integer mode)

When mode is 0, ifx_nullformat will cause all null columns to be returned as empty strings. If mode is 1, they are returned as NULL, a four-character string.

integer ifx_num_fields(integer result)

The ifx_num_fields function returns the number of columns in the result set.

integer ifx_num_rows(integer result)

Use ifx_num_rows to get the exact number of rows already fetched for a result set. To get an approximate number of rows in a result set, use ifx_affected_rows.

integer ifx_pconnect(string database, string user, string password)

The ifx_pconnect function is similar to ifx_connect, except that connections are not closed until the Web server process ends. Persistent links remain available for subsequent connection attempts by other scripts.

```
<?
  //connect to database
  if(!($dbLink = ifx_pconnect("mydb@ol_srv1", "leon", "secret"))
  {
    print("Unable to connect!<BR>\n");
    exit();
  }
  //treat blobs as varchars
  ifx_textasvarchar(TRUE);

  //get a record from the message table
  $Query = "SELECT Title, Body FROM message " .
    "WHERE ID = 3 ";
  if(!($result = ifx_query($Query, $dbLink))
  {
    print("Unable to query message table!<BR>\n");
```

```
  }

  //print results in HTML table
  ifx_htmltbl_result($result);

  //free result identifier
  ifx_free_result($dbLink);

  //close connection
  ifx_close($dbLink);
?>
```

integer ifx_prepare(string query, integer link, integer cursor_type, array blob_id)

The ifx_prepare function parses a query but does not execute it. Otherwise it operates identically to ifx_query. To execute the query, use ifx_do.

```
<?
  //connect to database
  if(!($dbLink = ifx_pconnect("mydb@ol_srv1", "leon", "secret"))
  {
    print("Unable to connect!<BR>\n");
    exit();
  }

//get message about PHP
$Query = "SELECT ID, Title FROM message " .
  "WHERE Title like '%PHP%' ";
if(!($result = ifx_prepare($Query, $dbLink, IFX_SCROLL))
{
  print("Unable to query message table!<BR>\n");
}

if(ifx_affectedrows($result)  100)
{
  //execute query
  ifx_do($result);

  //fetch each row, print a link
  while($row = ifx_fetch_row($result, "NEXT"))
  {
    print("A HREF=\"get.php?id={$row["ID"]}\">");
    print("{$row["Title"]}/A><BR>\n");
  }
}
else
{
  print("Too many results to display on one page.<BR>\n");
}
```

```
  //free result identifier
  ifx_free_result($dbLink);

  //close connection
  ifx_close($dbLink);
?>
```

integer ifx_query(string query, integer link, integer cursor_type, array blob_id)

The ifx_query function executes a query and returns a result identifier, which most of the other Informix functions require. The link argument is as returned by ifx_pconnect, but if you leave it out, the last link established will be used. If the query is a select, you may use the IFX_SCROLL and IFX_HOLD constants for the cursor type.

If performing an update or insert, you may use a ? in the query and match it to an entry in the blob_id argument. Each entry must be a value returned by ifx_create_blob. Selects that return blob columns will return blob identifiers by default, but you can override this functionality with ifx_textasvarchar.

Note that both cursor_type and blob_id are optional. The ifx_query function will allow you to specify an array of blob identifiers for the third argument.

boolean ifx_textasvarchar(integer mode)

Use ifx_textasvarchar to control how text blobs are returned in queries. If mode is 0, blob IDs are returned. If mode is 1, blob contents are returned.

boolean ifx_update_blob(integer blob, string data)

The ifx_update_blob function changes the contents of a blob.

boolean ifx_update_char(integer character, string data)

The ifx_update_char function changes the contents of a character object.

ifxus_close_slob

You can use ifxus_close_slob as an alias for ifxus_free_slob.

integer ifxus_create_slob(integer mode)

Use ifxus_create_slob to create a slob object. The object identifier is returned. The modes listed in Table 4 may be combined with | operators.

Table 4. Informix Slob Modes

Value	Informix API Constant
1	LO_RDONLY
2	LO_WRONLY
4	LO_APPEND
8	LO_RDWR
16	LO_BUFFER
32	LO_NOBUFFER

boolean ifxus_free_slob(integer slob)

The ifxus_free_slob function deletes a slob object.

integer ifxus_open_slob(integer slob, integer mode)

Use ifxus_open_slob to get an identifier for an existing slob object. The modes listed in Table 4 may be combined with | operators.

string ifxus_read_slob(integer slob, integer bytes)

The ifxus_read_slob function returns data from the specified slob object. The bytes argument specifies the number of bytes to return.

integer ifxus_seek_slob(integer slob, integer mode, integer offset)

The ifxus_seek_slob function moves the current cursor position within a slob object. The mode argument controls where the offset is applied. If mode is 0, offset is applied to the beginning. If mode is 1, offset is applied to the current position. If mode is 2, offset is applied to the end of the slob object.

integer ifxus_tell_slob(integer slob)

The ifxus_tell_slob function returns the current position of a cursor inside a slob object.

integer ifxus_write_slob(integer slob, string data)

The ifxus_write_slob function writes data to an open slob object. The number of bytes written is returned.

8.5 INTERBASE

InterBase is a full-featured database that spent much of its life as closed-source and proprietary. In January 2000, Inprise released InterBase under an open source license, allowing everyone access to the source code. InterBase is the first **open source database** to be compliant with the SQL 92 standard. Under commercial development for more than 16 years, InterBase compares favorably to Oracle, Sybase, and IBM's DB2. We discuss the PHP functions for communicating with InterBase, but a tutorial on InterBase itself is out of scope.

Keyword

Open source database is any database application with a codebase that is free to view, download, modify, distribute, and reuse.

InterBase support was added to PHP by Jouni Ahto. Later work was done by Andrew Avdeev and Ivo Panacek. At the time of writing, InterBase functions in PHP were not complete, but the recent change in licensing will probably encourage developers. You also have the option of using the ODBC functions.

boolean ibase_blob_add(integer blob, string data)

The ibase_blob_add function adds data to a blob. You must create the blob first with ibase_blob_create.

boolean ibase_blob_cancel(integer blob)

Use ibase_blob_cancel to discard a blob you have created with ibase_blob_create.

boolean ibase_blob_close(integer blob)

The ibase_blob_close function writes changes made to a blob to the database.

integer ibase_blob_create(integer link)

The ibase_blob_create function creates a new blob. The link argument is optional and will default to the last-opened connection. A blob identifier is returned.

boolean ibase_blob_echo(string blob)

The ibase_blob_echo function prints the contents of the named blob to the browser.

string ibase_blob_get(integer blob, integer bytes)

Use ibase_blob_get to get the specified number of bytes from a blob.

integer ibase_blob_import(integer file) integer ibase_blob_import(integer link, integer file)

The ibase_blob_import function creates a blob and places the contents of an open file into it. The file argument must be a file identifier as returned by fopen. You may call ibase_blob_import with or without an open link. The file is closed after the import. The blob identifier is returned.

object ibase_blob_info(string blob)

Use ibase_blob_info to get information about a blob. An object is returned with the following properties: isnull, length, maxseg, numseg, stream.

integer ibase_blob_open(string blob)

Use ibase_blob_open to get a blob identifier for an existing blob.

boolean ibase_close(integer link)

Use ibase_close to close a connection created by ibase_connect. If a link is not specified, the last opened link will be closed. The default transaction will be committed, and other transactions will be rolled back.

boolean ibase_commit(integer link)

The ibase_commit function commits the default transaction on the specified link, or the last link if none is specified.

integer ibase_connect(string path, string user, string password)

Use ibase_connect to connect to an InterBase database. You must specify a path to database file. The user and password arguments may be omitted. They default to those set in php.ini using the ibase.default_user and ibase.default_password directives.

A connection identifier is returned that is used by most of the other functions. When the script ends, the connection will be closed for you, but you can close it manually with ibase_close. A second connection attempt to the same database as the same user will return the same connection identifier.

Compare this function to ibase_pconnect.

string ibase_errmsg()

Use ibase_errmsg to get the last error message. FALSE is returned when no error message is available.

integer ibase_execute(integer query, value bind, ...)

Use ibase_excute to execute a query prepared with ibase_prepare. If the query contained ? placeholders, you must supply matching bind values following the query identifier. A result identifier is returned if executing a select query.

object ibase_fetch_object(integer result, integer blob)

The ibase_fetch_object function returns an object that contains a property for each column in the next result row. The name of the property will match the name of the column. The blob argument is optional. If set to IBASE_TEXT, blob columns will be returned as text. Otherwise the blob identifier is returned. FALSE is returned when no rows remain.

array ibase_fetch_row(integer result, integer blob)

The ibase_fetch_row function operates identically to ibase_fetch_object except that an array is returned. Instead of being referenced by name, columns are referenced by number, starting with 0.

array ibase_field_info(integer result, integer field)

Use ibase_field_info to get information about a column in a result set. An associative array is returned containing the following elements: alias, length, name, relation, type.

boolean ibase_free_query(integer query)

Use ibase_free_query to free memory associated with a prepared query.

boolean ibase_free_result(integer result)

Use ibase_free_result to free memory associated with a result set.

integer ibase_num_fields(integer result)

The ibase_num_fields function returns the number of fields in a result set.

ibase_pconnect(string path, string user, string password, string character_set)

The ibase_pconnect function works similarly to ibase_connect. The difference is that connections are not closed by PHP or by your script. They persist with the server process to be reused when later script executions need identical connections.

integer ibase_prepare(string query) integer ibase_prepare(integer link, string query)

Use ibase_prepare to prepare a query for later execution with ibase_execute. If you leave out the link argument, the last-opened link will be used. A query identifier is returned.

integer ibase_query(string query, value bind, ...) integer ibase_query(integer link, string query, value bind, ...)

The ibase_query function executes a query on an open connection. You may skip the link identifier, causing the last-opened connection to be used. If the query contains ? placeholders, you must match them with bind values that follow the query argument. A result identifier is returned. It is used with functions such as ibase_fetch_row.

```
<?
  //connect to database
  if(!($dbLink = ibase_connect("mydatabase.gdb", "leon", "secret"))
  {
    print("Unable to connect!<BR>\n");
    exit();
  }


 //begin transaction
 $dbTran = ibase_trans(IBASE_DEFAULT, $dbLink);

 //insert a message using bind parameters
 $Query = "INSERT INTO message " .
   "VALUES (?, ?, ?) ";

 if(!($result = ibase_query($dbLink, $Query, $inputID, $inputTitle,
$inputBody))
 {
   print("Unable to insert row!<BR>\n");
   exit();
 }

 //release memory
 ibase_free_result($result);


 //dump table
 print("<TABLE BORDER=\"1\">\n");
 $Query = "SELECT * FROM message ";

 if(!($result = ibase_query($dbLink, $Query))
 {
   print("Unable to query table!<BR>\n");
   exit();
 }

 //print headers
 print("<TR>\n");
 for($i=0; $i<ibase_num_fields($result); $i++)
 {
   $info = ibase_field_info($result, $i);
   print("<TH>{$info["name"]}</TH>\n");
 }
 print("</TR>\n");

 //get all rows
 while($row = ibase_fetch_row($result))
 {
   print("<TR>\n");
   for($i=0; $iibase_num_fields($result); $i++)
   {
     print("<TD>$row[$i]/<TD>\n");
   }
   print("</TR>\n");
 }
```

```
  print("</TABLE>\n");

  //release memory
  ibase_free_result($result);

  //commit transaction
  ibase_commit($dbTran);

  //close connection


  ibase_close($dbLink);
?>
```

boolean ibase_rollback(integer link)

The ibase_rollback function causes a transaction to roll back. The default transaction on the specified link is rolled back. The last link will be used if none is specified.

boolean ibase_timefmt(string format)

Use ibase_timefmt to set the format for datetime columns. The format string should follow the rules of strftime. The default format is "%m/%d/%Y %H:%M:%S".

integer ibase_trans(integer flags, integer link)

The ibase_trans function returns a transaction identifier. The flags can be any combination of the constants listed in Table 5. Use | to combine them. IBASE_DEFAULT matches InterBase read, write, snapshot, and wait properties. The link argument is optional. The last connection established will be used if it's left out.

Table 5. InterBase Constants

IBASE_COMMITED
IBASE_CONSISTENCY
IBASE_DEFAULT
IBASE_NOWAIT
IBASE_READ
IBASE_TEXT
IBASE_TIMESTAMP

8.6 MSQL

mSQL or Mini SQL is a lightweight database management system from Hughes Technologies. There are a handful of mSQL functions that exist simply for backward compatibility. We have chosen to leave them out. Only some of them are documented in the official PHP 3 manual. Their use is described in the PHP 2 documentation. For reference they are msql, msql_createdb, msql_dbname, msql_dropdb, msql_freeresult, msql_listdbs, msql_list-fields, msql_listtables, msql_numfields, msql_numrows, msql_ selectdb, msql_tablename.

Two mSQL extensions exist, one for mSQL version 1 and one for version 2. Zeev Suraski wrote both mSQL extensions.

integer msql_affected_rows(integer link)

The msql_affected_rows function returns the number of rows involved in the previous query made on the given link. The link argument must be an integer returned by msql_connect or msql_pconnect.

boolean msql_close(integer link)

The msql_close function closes the link to a database. If the link argument is left out, the last-opened link is closed. Only links opened by msql_connect may be closed. Using this function is not strictly necessary, since all nonpersistent links are automatically closed when a script ends.

integer msql_connect(string host, string username, string password)

The msql_connect function attempts to connect to the mSQL server at the specified host. If the host argument is left out, the local host will be assumed. A link identifier is returned. In the case where an open link exists, it will be returned rather than establishing a second link. The connection is automatically closed at the end of the script.

You may add a colon and a port number to the host argument.

Did You Know?

In 1993–94 David Hughes developed a network monitoring and management system called Minerva. The design of this system required a database management system to store its configuration and working data. To enable future portability, Hughes elected to use a Structured Query Language interface between the application and the database management system, despite the fact that at the time there was no free or inexpensive SQL database management implementation available.

boolean msql_create_db(string database, integer link)

The msql_create_db function attempts to create a database. The link argument is optional. If left out, the last-opened link will be used.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_create_db("store", $Link);
  msql_close($Link);
?>
```

boolean msql_data_seek(integer result, integer row)

Use msql_data_seek to move the internal row pointer to the specified row in a result set. The result argument is as returned by msql_query.

```
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Result = msql_query("SELECT Name FROM customer", $Link);

  // jump to tenth customer
  msql_data_seek($Result, 10);

  $Row = msql_fetch_row($Result);

  print($Row[0]);

  msql_close($Link);
?>
```

integer msql_db_query(string database, string query, integer link)

The msql_db_query function is identical to msql_query except that it specifies a database rather than using the database selected with msql_ select_db. The query is executed in the specified database and a result identifier is returned.

```
<?
  $Link = msql_connect("msql.clearink.com");
  $Query = "DELETE FROM customer";
  $Result = msql_db_query("store", $Query, $Link);

  $RowsAffected = msql_affected_rows($Link);
  print($RowsAffected . " rows deleted.");

  msql_close($Link);
?>
```

boolean msql_drop_db(string database, integer link)

The msql_drop_db function removes an entire database from the server. The link argument is optional and, if omitted, the last connection opened will be used.

```
<?
  $Link = msql_connect("msql.clearink.com");

  if(msql_drop_db("store", $Link))
  {
    print("Database deleted!");
  }
  else
  {
    print("Database not deleted: ");

   print(msql_error());
  }

 msql_close($Link);
?>
```

string msql_error()

Use msql_error to retrieve the last error message returned by an mSQL function.

array msql_fetch_array(integer result, integer type)

The msql_fetch_array function returns an array of the data for the current row. The result argument is as returned by msql_query. By default, result columns are returned in two elements each: one referenced by number and one referenced by field name. The optional type argument controls which elements are created. MSQL_NUM signals that only numbered elements be created. MSQL_ASSOC signals that only named elements be created. If you want both, you can explicitly request it with MSQL_BOTH.

Compare this function to msql_fetch_row and msql_fetch_object.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM customer";
  $Result = msql_query($Query, $Link);

  //fetch each row
  while($Row = msql_fetch_array($Result, MSQL_ASSOC))
  {
    print($Row["FirstName"] . "<BR>\n");
  }

  msql_close($Link);
?>
```

object msql_fetch_field(integer result, integer field)

The msql_fetch_field function returns an object with properties that describe the specified field. The field argument may be left out, and the next unfetched field will be returned. The properties of the object are listed in Table 6.

```
$Link = msql_connect("msql.clearink.com");
msql_select_db("store", $Link);

$Query = "SELECT * FROM item i, SKU s ";
$Query .= "WHERE i.SKU = s.ID ";
$Result = msql_query($Query, $Link);

// get description of each field
while($Field = msql_fetch_field($Result))
{
  print("Name: " . $Field->name . "<BR>\n");
  print("Table: " . $Field->table . "<BR>\n");
  print("Not Null: " . $Field->not_null . "<BR>\n");
  print("Primary Key: " . $Field->primary_key"<BR>\n");
  print("Unique: " . $Field->unique . "<BR>\n");
  print("Type: " . $Field->type . "<BR>\n<BR>\n");
}

  msql_close($Link);
?>
```

Table 6. Properties of msql_fetch_field Object

Property	Description
name	Name of the column
not_null	TRUE if the column cannot be null
primary_key	TRUE if the column is a primary key
table	Name of the table the column is from
type	Datatype of the column
unique	TRUE if the column is a unique key

object msql_fetch_object(integer result)

The msql_fetch_object function returns an object with a property for each column of the resulting row. Each call to msql_fetch_object gets the next row from the results, or returns FALSE when there are none left.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item";
  $Result = msql_query($Query, $Link);

  while($Row = msql_fetch_object($Result))
  {
    print("$Row->ID: $Row->Name<BR>\n");
  }

  msql_close($Link);
?>
```

array msql_fetch_row(integer result)

The msql_fetch_row function returns an array with one element for each resulting column. FALSE is returned when no results are left. Columns are referenced by integers starting at zero. Compare this function to msql_fetch_array and msql_fetch_object.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item";
  $Result = msql_query($Query, $Link);

  while($Row = msql_fetch_row($Result))
  {
    print($Row[0] . ": " . $Row[1] . "<BR>\n");
  }

  msql_close($Link);
?>
```

boolean msql_field_seek(integer result, integer field)

Use msql_field_seek to move the internal field pointer to the specified field.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item i, SKU s ";
  $Query .= "WHERE i.SKU = s.ID ";
  $Result = msql_query($Query, $Link);

  // get description of each field
  // starting with the third
  msql_field_seek($Result, 2);


while($Field = msql_fetch_field($Result))
{
   print("Name: " . $Field->name . "<BR>\n");
   print("Table: " . $Field->table . "<BR>\n");
   print("Not Null: " . $Field->not_null ."<BR>\n");
   print("Primary Key: " . $Field->primary_key"<BR>\n");
   print("Unique: " . $Field->unique . "<BR>\n");
   print("Type: " . $Field->type . "<BR>\n<BR>\n");
}

  msql_close($Link);
?>
```

string msql_fieldflags(integer result, integer field)

The msql_fieldflags function returns all the flags turned on for the specified field. These may be primary key, unique, and not null.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item";
  $Result = msql_query($Query, $Link);
  print("Field 0 flags are " .msql_fieldflags($Result, 0));

  msql_close($Link);
?>
```

integer msql_fieldlen(integer result, integer field)

The msql_fieldlen function returns the length of the specified field.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item";
  $Result = msql_query($Query, $Link);

  print("Field 0 length is " . msql_fieldlen($Result, 0));

  msql_close($Link);
?>
```

string msql_fieldname(integer result, integer field)

The msql_fieldname function returns the name of the specified field.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item";

 $Result = msql_query($Query, $Link);

 print("Field 0 is " . msql_fieldname($Result, 0));

 msql_close($Link);
?>
```

string msql_fieldtable(integer result, integer field)

The msql_fieldtable function returns the name of the table for the specified field.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item";
  $Result = msql_query($Query, $Link);

  print("Field 0 is from " . msql_fieldtable($Result, 0));

  msql_close($Link);
?>
```

string msql_fieldtype(integer result, integer field)

The msql_fieldtype function returns the type of the specified field.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item";
  $Result = msql_query($Query, $Link);

  print("Field 0 is " . msql_fieldtype($Result, 0));

  msql_close($Link);
?>
```

boolean msql_free_result(integer result)

When a script ends, all results are freed. If memory is a concern while your script is running, use msql_free_result.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "INSERT INTO store VALUES (0, 'Martinez')";
  $Result = msql_query($Query, $Link);

  msql_free_result($Result);

  msql_close($Link);
?>
```

integer msql_list_dbs(integer link)

The msql_list_dbs function returns a result identifier as if the database were queried with msql_query. Any of the functions for fetching rows or fields may be used to get the names of the databases. The link argument is optional. If left out, the last-opened connection will be used.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Result = msql_list_dbs($Link);
  while($row_array = msql_fetch_row($Result))
  {
    print($row_array[0] . "<BR>\n");
  }

  msql_close($Link);
?>
```

integer msql_list_fields(string database, string tablename, integer link)

The msql_list_fields function returns a result identifier as if the database were queried with msql_query. Any of the functions for fetching rows or fields may be used to get the names of the fields. The link argument is optional. If left out, the last-opened connection will be used.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Result = msql_list_fields("store", "item",$Link);
  while($row_array = msql_fetch_row($Result))
  {

    print($row_array[0] . "<BR>\n");
  }

 msql_close($Link);
?>
```

integer msql_list_tables(string database, integer link)

The msql_list_tables function returns a result identifier as if the database were queried with msql_query. Any of the functions for fetching rows or fields may be used to get the names of the fields. The link argument is optional. If left out, the last-opened connection will be used.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Result = msql_list_tables("store", $Link);
  while($row_array = msql_fetch_row($Result))
  {
    print($row_array[0] . "<BR>\n");
  }

  msql_close($Link);
?>
```

integer msql_num_fields(integer result)

Use msql_num_fields to get the number of fields in a result set.

```
<?
   $Link = msql_connect("msql.clearink.com");
   msql_select_db("store", $Link);

   $Query = "SELECT * FROM item i, SKU s ";
   $Query .= "WHERE i.SKU = s.ID ";
   $Result = msql_query($Query, $Link);

   print(msql_num_fields($Result));

   msql_close($Link);
?>
```

integer msql_num_rows(integer result)

The msql_num_rows function returns the number of rows in the result set.

```
<?
   $Link = msql_connect("msql.clearink.com");
   msql_select_db("store", $Link);

   $Query = "SELECT * FROM item i, SKU s ";
   $Query .= "WHERE i.SKU = s.ID ";
   $Result = msql_query($Query, $Link);

   print(msql_num_rows($Result));

   msql_close($Link);
?>
```

integer msql_pconnect(string host)

The msql_pconnect function is identical to the msql_connect function except that the connection will not be closed when the script ends. This has meaning only when PHP is compiled as an Apache module. These are called persistent links because they live as long as the server process.

```
<?
   $Link = msql_pconnect("localhost");
?>
```

integer msql_query(string query, integer link)

Use msql_query to execute a query. The database used will be the one specified in a call to the msql_select_db function. The link argument is optional. The last connection made will be used if it is left out.

msql_regcase

This is an alias to sql_regcase.

string msql_result(integer result, integer row, string field)

The msql_result function returns a single field value for the given row. The field argument can be interpreted in two ways. If it is a number, it will be used as a field offset, starting with zero. Otherwise, it will be considered to be a column name.

The msql_result function is relatively slow. Its use should be avoided in favor of faster functions such as msql_fetch_array.

```
<?
  $Link = msql_connect("msql.clearink.com");
  msql_select_db("store", $Link);

  $Query = "SELECT * FROM item i, SKU s ";
  $Query .= "WHERE i.SKU = s.ID ";
  $Result = msql_query($Query, $Link);

  $numRows = msql_num_rows($Result);

  for($index = 0; $index  $numRows; $index++)
  {
    $item_ID = msql_result($Result, $index, "item.ID");
    $item_Name = msql_result($Result, $index, "item.Name");

    print("$item_ID: $item_Name<BR>\n");
  }

  msql_close($Link);
?>
```

boolean msql_select_db(string database, integer link)

Use msql_select_db to select the database against which to make queries. As with most other mSQL functions, the link identifier is not required.

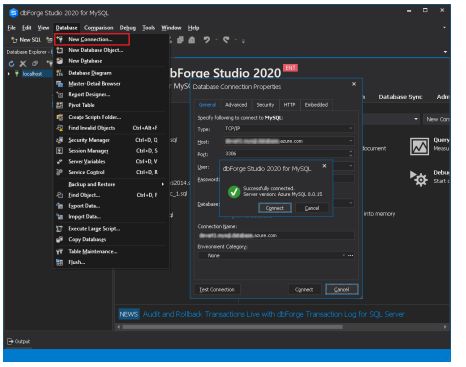
8.7 MYSQL

MySQL is a relational database with a license that allows you to use it cost-free for most noncommercial purposes. It shares many features with mSQL because it was originally conceived as a faster, more flexible replacement. Indeed, MySQL has delivered on these goals. It easily outperforms even commercial databases. Not surprisingly, MySQL is the database of choice for many PHP developers.

As with mSQL, there were MySQL functions in PHP2 that are still supported in PHP3, but their use is discouraged. We have chosen to leave these functions out of the reference. The functions we have left out are mysql, mysql_cre- atedb, mysql_dbname,

mysql_dropdb, mysql_fieldflags, mysql_field-len, mysql_fieldname, mysql_fieldtable, mysql_fieldtype, mysql_ freeresult, mysql_listdbs, mysql_listfields, mysql_listtables, mysql_numfields, mysql_numrows, mysql_selectdb, mysql_tablename.

The MySQL extension was written by Zeev Suraski.



integer mysql_affected_rows(integer link)

The mysql_affected_rows function returns the number of rows affected by the last query made to the specified database connection link. If the link argument is omitted, the last-opened connection is assumed. If the last query was an unconditional delete, zero will be returned. If you want to know how many rows were returned by a select statement, use mysql_num_rows.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //update some invoices
  $Query = "UPDATE invoice " .
    "SET Active = 'Y' " .
    "WHERE ID  100 ";
  $dbResult = mysql_query($Query, $dbLink);

  //let user know how many rows were updated
  $AffectedRows = mysql_affected_rows($dbLink);
  print("$AffectedRows rows updated.<BR>\n");
?>
```

boolean mysql_change_user(string user, string password, string database, integer link)

Use mysql_change_user to change the user for a database connection. The database and link arguments are optional. If left out, the current database and the link last opened are used. If the user cannot be changed, the current connection remains open with the original user. This function requires MySQL version 3.23.3 or newer.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //switch to admin user
  mysql_change_user("admin", "secret", "freetrade", $dbLink);
?>
```

boolean mysql_close(integer link)

Use mysql_close to close the connection to a database. The connect must have been opened with mysql_connect. Use of this function is not strictly necessary, as all nonpersistent links are closed automatically when the script finishes. The link argument is optional, and when it's left out, the connection last opened is closed.

```
<?
  // open connection
  $Link = mysql_connect("localhost", "httpd", "");

  // close connection
  mysql_close($Link);
?>
```

integer mysql_connect(string host, string user, string password)

The mysql_connect function begins a connection to a MySQL database at the specified host. If the database is on a different port, follow the hostname with a colon and a port number. You may alternatively supply a colon and the path to a socket if connecting to localhost. This might be written as localhost:/tmp/sockets/mysql. All the arguments are optional and will default to localhost, the name of the user executing the script, and an empty string, respectively. The user executing the script is typically httpd, the Web server.

Connections are automatically closed when a script finishes execution, though they may be closed earlier with mysql_close. If you attempt to open a connection that is already open, a second connection will not be made. The identifier of the previously open connection will be returned.

FALSE is returned in the event of an error.

```
<?
  //establish connection
  if(!($dbLink = mysql_connect("localhost:3606", "freetrade", "")))
  {
    print("mysql_connect failed!<BR>\n");
  }

  //select database
  if(!(mysql_select_db("freetrade", $dbLink)))
  {
    print("mysql_select_db failed!<BR>\n");
    print(mysql_errno() . ": ");
    print(mysql_error() . "<BR>\n");
  }
```

Remember

Note that you must open a connection with an account that has permission to create databases. If you leave out the link argument, the last-opened connection will be used.

boolean mysql_create_db(string database, integer link)

Use mysql_create_db to create a new database.

```
<?
  // open connection
  $dbLink = mysql_connect("localhost", "admin", "secret");

  //create database
  mysql_create_db("garbage", $dbLink);
?>
```

boolean mysql_data_seek(integer result, integer row)

The mysql_data_seek function moves the internal row pointer of a result set to the specified row. Use this function with mysql_fetch_row to jump to a specific row. The result argument must have been returned from mysql_query or a similar function.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //get states from tax table
  $Query = "SELECT State " .
    "FROM tax ";
  $dbResult = mysql_query($Query, $dbLink);

  //jump to fifth row
  mysql_data_seek($dbResult, 4);
```

```
  //get row
  $row = mysql_fetch_row($dbResult);

  //print state name
  print($row[0]);
?>
```

integer mysql_db_query(string database, string query, integer link)

The mysql_db_query function executes a query on the specified database and returns a result identifier. If the link argument is omitted, the last-opened link will be used, or a new one will be created if necessary.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //truncate session table
  $Query = "DELETE FROM session ";
  $dbResult = mysql_db_query("freetrade", $Query, $dbLink);
?>
```

boolean mysql_drop_db(string database, integer link)

Use mysql_drop_db to delete a database. If the link argument is omitted, the lastopened link will be used.

```
<?
  //open connection
  $dbLink = mysql_connect("localhost", "admin", "secret");

  //drop garbage database
  if(mysql_drop_db("garbage", $dbLink))
  {
    print("Database dropped.BR>");
  }
  else
  {
    print("Database drop failed!BR>");
  }
?>
```

integer mysql_errno(integer link)

The mysql_errno function returns the error number of the last database action. If the optional link identifier is left out, the last connection will be assumed.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //try to execute a bad query (missing fields)


  $Query = "SELECT FROM tax ";
  if(!($dbResult = mysql_query($Query, $dbLink)))
  {
    // get error and error number
    $errno = mysql_errno($dbLink);
    $error = mysql_error($dbLink);

    print("ERROR $errno: $error<BR>\n");
  }
?>
```

string mysql_error(integer link)

Use mysql_error to get the textual description of the error for the last database action. If the optional link identifier is left out, the last connection will be assumed.

array mysql_fetch_array(integer result, integer type)

The mysql_fetch_array function returns an array that represents all the fields for a row in the result set. Each call produces the next row until no rows are left, in which case FALSE is returned. By default, each field value is stored twice: once indexed by offset starting at zero and once indexed by the name of the field. This behavior can be controlled with the type argument. If the MYSQL_NUM constant is used, elements will be indexed by field numbers only. If the MYSQL_ASSOC constant is used, elements will be index by field names only. You can also use MYSQL_BOTH to force the default.

Compare this function to mysql_fetch_object and mysql_fetch_row.

object mysql_fetch_field(integer result, integer field)

Use the mysql_fetch_field function to get information about a field in a result set. Fields are numbered starting with zero. The return value is an object with properties described in Table 7.

If the field argument is left out, the next field in the set will be returned. This behavior allows you to loop through each field easily.

Table 7. Properties of mysql_fetch_field Object

Property	Description
blob	TRUE if the column is a blob
max_length	Maximum length
multiple_key	TRUE if the column is a nonunique key
name	Name of the column
not_null	TRUE if the column cannot be null
numeric	TRUE if the column is numeric
primary_key	TRUE if the column is a primary key
table	Name of the table
type	Type of the column
unique_key	TRUE if the column is a unique key
unsigned	TRUE if the column is unsigned
zerofill	TRUE if the column is zero-filled

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //get everything from address table
  $Query = "SELECT * " .
    "FROM address a, user u " .
    "WHERE u.Address = a.ID ";
  $dbResult = mysql_query($Query, $dbLink);

  // get description of each field
  while($Field = mysql_fetch_field($dbResult))
  {
    print("$Field->table, $Field->name, $Field->type<BR>\n");
  }
?>
```

array mysql_fetch_lengths(integer result)

Use mysql_fetch_lengths to get an array of the maximum lengths for each of the fields in a result set.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //get everything from address table
  $Query = "SELECT * " .
    "FROM address ";
  $dbResult = mysql_query($Query, $dbLink);

  //get field lengths
  $lengths = mysql_fetch_lengths($dbResult);

   //print length of the third column
   print($lengths[2]);
?>
```

object mysql_fetch_object(integer result)

The mysql_fetch_object function is similar to mysql_fetch_array and mysql_fetch_row. Instead of an array, it returns an object. Each field in the result set is a property in the returned object. Each call to mysql_fetch_object returns the next row, or FALSE if there are no rows remaining. This allows you to call mysql_fetch_object in the test condition of a while loop to get every row.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //get unique cities from address table
  $Query = "SELECT DISTINCT City, StateProv " .
    "FROM address ";
  $dbResult = mysql_query($Query, $dbLink);

  // get each row
  while($row = mysql_fetch_object($dbResult))
  {
    // print name
    print("$row->City, $row->StateProv<<BR>\n");
  }
?>
```

array mysql_fetch_row(integer result)

The mysql_fetch_row function returns an array that represents all the fields for a row in the result set. Each call produces the next row until no rows are left, in which case FALSE is returned. Each field value is indexed numerically, starting with zero.

Compare this function to mysql_ fetch_array and mysql_fetch_object. There is not much difference in performance between these three functions.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //get unique cities from address table
  $Query = "SELECT City, StateProv " .
    "FROM address ";
  $dbResult = mysql_query($Query, $dbLink);


  //get each row
  while($row = mysql_fetch_row($dbResult))
  {
    // print city, state
    print("$row[0], $row[1]<<BR>\n");
  }
?>
```

string mysql_field_flags(integer result, integer field)

Use mysql_field_flags to get a description of the flags on the specified field. The flags are returned in a string and separated by spaces. The flags you can expect are auto_increment, binary, blob, enum, multiple_key, not_null, primary_key, timestamp, unique_key, unsigned, and zerofill. Some of these flags may be available only in the newest versions of MySQL.

integer mysql_field_len(integer result, integer field)

Use mysql_field_len to get the maximum number of characters to expect from a field. The fields are numbered from zero.

string mysql_field_name(integer result, integer field)

Use mysql_field_name to get the name of a column. The field argument is an offset numbered from zero.

boolean mysql_field_seek(integer result, integer field)

The mysql_field_seek function moves the internal field pointer to the specified field. The next call to mysql_fetch_field will get information from this field.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //get everything from address table
  $Query = "SELECT * " .
    "FROM address ";
  $dbResult = mysql_query($Query, $dbLink);


  //skip to second field
  mysql_field_seek($dbResult, 1);

  //get description of each field
  while($Field = mysql_fetch_field($dbResult))
  {
    print($Field->table, $Field->name, $Field->type<BR>\N");
  }
?>
```

string mysql_field_table(integer result, integer field)

The mysql_field_table function returns the name of the table for the specified field. If an alias is used, as in the example, the alias is returned.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //get everything from user table
  //get everything from address table
  $Query = "SELECT * " .
    "FROM address a, user u " .
    "WHERE u.Address = a.ID ";

  $dbResult = mysql_query($Query, $dbLink);

  $Fields = mysql_num_fields($dbResult);
  for($i = 0; $i  $Fields; $i++)
  {
    print(mysql_field_table($dbResult, $i) . "<BR>\n");
  }
?>
```

string mysql_field_type(integer result, integer field)

Use mysql_field_type to get the type of a particular field in the result set.

boolean mysql_free_result(integer result)

Use mysql_free_result to free any memory associated with the specified result set. This is not strictly necessary, as this memory is automatically freed when a script finishes executing.

```
<?
  // connect to server
  $Link = mysql_connect("localhost", "httpd", "");

  // select the 'store' database
  mysql_select_db("store", $Link);

  // get everything from customer table
  $Query = "SELECT * FROM customer ";
  $Result = mysql_query($Query, $Link);

  // free result set
  mysql_free_result($Result);
?>
```

integer mysql_insert_id(integer link)

After inserting into a table with an auto_increment field, the mysql_insert_id function returns the id assigned to the inserted row. If the link argument is left out, the most recent connection will be used.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //select the 'freetrade' database
  mysql_select_db("freetrade", $dbLink);

  //insert a row
  $Query = "INSERT INTO user (Login, Password) " .
    "VALUES('leon', 'secret') ";
  $dbResult = mysql_query($Query, $dbLink);

  //get id
  print("ID is " . mysql_insert_id($dbLink));
```

integer mysql_list_dbs(integer link)

The mysql_list_dbs function queries the server for a list of databases. It returns a result pointer that may be used with mysql_fetch_row and similar functions.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //get list of databases
  $dbResult = mysql_list_dbs($dbLink);

  //get each row
  while($row = mysql_fetch_row($dbResult))
  {
    // print name
    print($row[0] . "<BR>\n");
  }
?>
```

integer mysql_list_fields(string database, string table, integer link)

The mysql_list_fields function returns a result pointer to a query on the list of fields for a specified table. The result pointer may be used with any of the functions that get information about columns in a result set: mysql_field_flags, mysql_field_len, mysql_field_name, mysql_field_type. The link argument is optional.

integer mysql_list_tables(string database, integer link)

Use mysql_list_tables to get a result pointer to a list of tables for a specified database. The result pointer may be used in any of the functions for fetching rows from a result set. The link argument is optional.

```
<?
  //connect to server as freetrade user, no password
  $dbLink = mysql_pconnect("localhost", "freetrade", "");

  //get list of tables
  $dbResult = mysql_list_tables("freetrade", $dbLink);

  //get each row
  while($row = mysql_fetch_row($dbResult))
  {
        //print name
        print($row[0] . "<BR>\n");
  }
?>
```

integer mysql_num_fields(integer result)

The mysql_num_fields function returns the number of fields in a result set.

integer mysql_num_rows(integer result)

The msyql_num_rows function returns the number of rows in a result set.

integer mysql_pconnect(string host, string user, string password)

The mysql_pconnect function operates like mysql_connect except that the connection will be persistent. That is, it will not be closed when the script ends. The connection will last as long as the server process lasts, so that if a connection is attempted later from the same process, the overhead of opening a new connection will be avoided.

```
<?
  //open persistent connection
  $dbLink = mysql_pconnect("localhost", "freetrade", "");
?>
```

integer mysql_query(string query, integer link)

Use mysql_query to execute a query. If the link argument is omitted, the last connection made is used. If there has been no previous connection, PHP will connect to the local host. If the query performs an insert, delete, or update, a boolean value will be returned. Select queries return a result identifier.

string mysql_result(integer result, integer row, string field)

The mysql_result function returns the value of the specified field in the specified row. The field argument may be a number, in which case it is considered a field offset. It may also be the name of a column, either with the table name or without. It could also be an alias.

In general, this function is very slow. It's better to use mysql_fetch_row or a similar function.

```
<?
      //connect to server as freetrade user, no password
      $dbLink = mysql_connect("localhost", "freetrade", "");

      //select the 'freetrade' database
      mysql_select_db("freetrade", $dbLink);

      // get everything from customer table
      $Query = "SELECT * FROM " .
            "user u " .
            "WHERE u.Login like 'A%' ";
      $dbResult = mysql_query($Query, $dbLink);
```

```
        // get number of rows
        $rows = mysql_num_rows($dbResult);

        for($i = 0; $i  $rows; $i++)
        {
                $name = mysql_result($dbResult, $i, "u.Login");
                print("$name<BR>\n");
        }
?>
```

boolean mysql_select_db(string database, integer link)

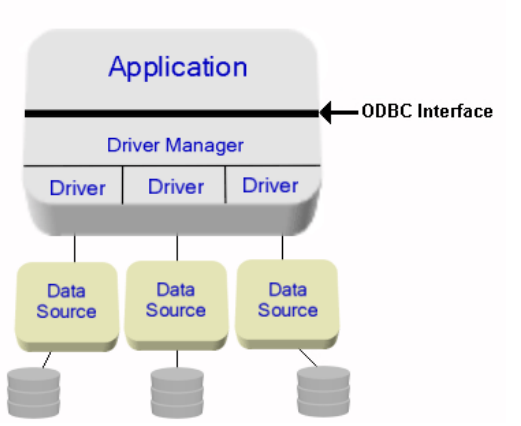
Use mysql_select_db to select the default database.

8.8 ODBC

Open Database Connectivity (ODBC) has become an industry standard for communicating with a database. The model is simple. Client software is designed to use an ODBC API. Vendors write drivers that implement this API on the client side and talk natively to their database on the server side. This allows application developers to write one application that can communicate with many different databases simply by changing the driver, which is an external file.

ODBC uses SQL as its language for communicating with any database, even when the database is not relational. Microsoft offers drivers that allow you to query text files and Excel workbooks.

Microsoft has offered free ODBC drivers for some time, but only for their operating systems. ODBC drivers for UNIX are harder to come by. Most database manufacturers offer drivers, and there are third parties, like Intersolv, that sell optimized drivers for both Windows and UNIX platforms.



Most of the database with native support in PHP can also be accessed via ODBC. There are also numerous databases that can only be accessed viaODBC by PHP. Two that others have tried are Solid and Empress.

Stig Bakken, Andreas Karajannis and Frank Kromann have contributed to the creation of the ODBC extension.

boolean odbc_autocommit(integer connection, boolean on)

The odbc_autocommit function sets whether queries are automatically committed when executed. By default this is on. The connection argument is an integer returned by the odbc_connect or odbc_pconnect functions. This function has to be used intelligently, as not all ODBC drivers support commits and rollbacks.

```
<?
      $Connection = odbc_connect("store", "sa", "sa");

      // turn off autocommit
      odbc_autocommit($Connection, FALSE);
?>
```

boolean odbc_binmode(integer result, integer mode)

Use odbc_binmode to set the way binary columns return data for a result set. When binary data are returned by the driver, each byte is represented by hexadecimal codes. By default, PHP will convert these codes into raw binary data. If you have to use the odbc_longreadlen function to set the maximum length of long data to anything other than zero, then the modes in Table 8 apply. If the maximum read length is zero, the data are always converted to raw binary data.

```
<?
       // get a GIF from a database and send it to browser

       // connect to database
       $Connection = odbc_connect("store", "admin", "secret");

       // execute query
       $Query = "SELECT picture ";

$Query .= "FROM employee ";
$Query .= "WHERE id=17 ";
$Result = odbc_do($Connection, $Query);

// make sure binmode is set for binary pass through
odbc_binmode($Result, ODBC_BINMODE_PASSTHRU);

// make sure longreadlen mode
// is set for echo to browser
odbc_longreadlen($Result, 0);
```

```
// get the first row, ignore the rest
odbc_fetch_row($Result);

// send header so browser knows it's a gif
header("Content-type: image/gif");

// get the picture
odbc_result($Result, 1);
```

Table 8. ODBC binary data modes

Mode	Description
ODBC_BINMODE_PASSTHRU	Pass through as binary data
ODBC_BINMODE_RETURN	Return as hexadecimal codes
ODBC_BINMODE_CONVERT	Return with data converted to a string

odbc_close(integer connection)

Use odbc_close to close a connection to a database. If there are open transactions for the connection, an error will be returned and the connection will not be closed.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT price ";
        $Query .= "FROM catalog ";
        $Query .= "WHERE id=10 ";
        $Result = odbc_do($Connection, $Query);


         odbc_fetch_row($Result)

         $price = odbc_result($Result, 1);

         print("$price<BR>\n");

         odbc_close($Connection);
?>
```

odbc_close_all()

The odbc_close_all function closes every connection you have open to ODBC data sources. Like odbc_close, it will report an error if you have an open transaction on one of the connections.

```
<?
        // connect to database three times
        $Connection1 = odbc_connect("store", "guest", "guest");
        $Connection2 = odbc_connect("store", "guest", "guest");
        $Connection3 = odbc_connect("store", "guest", "guest");

        // close all the connections
        odbc_close_all();
?>
?>
```

boolean odbc_commit(integer connection)

Use odbc_commit to commit all pending actions for the specified connection. If automatic commit is turned on, as is default, this function has no effect. Also, make sure your driver supports transactions before using this function.

```
<?
         // connect to database
         $Connection = odbc_connect("store", "guest", "guest");

         // turn off autocommit
         odbc_autocommit($Connection, FALSE);

         // put everything on sale
         $Query = "UPDATE catalog ";
         $Query .= "SET price = price * 0.9 ";
         $Result = odbc_do($Connection, $Query);

        // commit
        if(odbc_commit($Connection))
        {
              print("Commit successful!<BR>\n");
        }

        odbc_close($Connection);
?>
```

integer odbc_connect(string dsn, string user, string password, integer cursor_type)

Use odbc_connect to connect to an ODBC data source. The user and password arguments are required, so if your driver does not require them, pass empty strings. The optional cursor_type argument forces the use of a particular cursor so that you may avoid problems with some ODBC drivers. Using the SQL_CUR_USE_ODBC constant for cursor type may avoid problems with calling stored procedures or getting row numbers.

string odbc_cursor(integer result)

Use odbc_cursor to fetch the name of a cursor for a result set.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        print("Cursor: " . odbc_cursor($Result) . "<BR>\n");

        while(odbc_fetch_row($Result))
        {
              $name = odbc_result($Result, 1);
              $price = odbc_result($Result, 2);

              print("$name: $price<BR>\n");
        }

        odbc_close($Connection);
?>
```

integer odbc_do(integer connection, string query)

Use odbc_do to execute a query on a connection. A result identifier is returned, and is used in many of the other functions for fetching result data.

integer odbc_exec(integer connection, string query)

The odbc_exec function is an alias for odbc_do.

integer odbc_execute(integer result, array parameters)

The odbc_execute function executes a prepared statement. The result argument is an identifier returned by odbc_prepare. The parameters argument is an array that must be passed by reference and will be set with the value of the result columns. See odbc_prepare for an example of use. integer odbc_fetch_into(integer result, array fields) integer odbc_fetch_into(integer result, integer row, array fields) The odbc_fetch_into function gets the specified row for the specified result set and puts the columns into the fields array. The fields argument must be passed by reference. The number of columns in the result set is returned. The row argument may be omitted, in which case the next row in the set is returned.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        while(odbc_fetch_into($Result, &$fields))

        {
              $name = $fields[0];
              $price = $fields[1];
              print("$name: $price<BR>\n");
        }

        odbc_close($Connection);
?>
```

boolean odbc_fetch_row(integer result, integer row)

Use odbc_fetch_row to get a row of data from a result set. The data for the row is stored in internal memory, ready to be retrieved with the odbc_result function. The row argument is optional and, if left out, the next available row will be returned. FALSE will be returned when there are no more rows in the result set.

integer odbc_field_len(integer result, integer field)

Use odbc_field_len to get the length of a field in a result set. Fields are numbered starting with one.

```
<?
       // connect to database
       $Connection = odbc_connect("store", "guest", "guest");


        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        print(odbc_field_len($Result, 1));

        odbc_close($Connection);
?>
```

string odbc_field_name(integer result, integer field)

Use odbc_field_name to get the name of a field in a result set. Fields are numbered starting with one.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        print(odbc_field_name($Result, 1));

        odbc_close($Connection);
?>
```

string odbc_field_type(integer result, integer field)

Use odbc_field_type to get the type of a field in a result set. Fields are numbered starting with one.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        print(odbc_field_type($Result, 1));

        odbc_close($Connection);
?>
```

boolean odbc_free_result(integer result)

Use odbc_free_result to free the memory associated with the result set. This is not strictly necessary, but it's a good idea if you are worried about running out of memory. If autocommit is disabled and you free a result set before calling odbc_commit, the transaction will be rolled back.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "gu

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        // free the result set
        odbc_free_result($Result);
```

boolean odbc_longreadlen(integer result, integer length)

Use odbc_longreadlen to set the maximum length for values of any columns of type long. This includes binary columns such as longvarbinary. By default the maximum length is zero, which has the special meaning of causing fetched columns to be echoed to the browser. Any other positive number will cause returned values to be truncated to the specified length.

Note that it is not always apparent that a field is considered to be a long by the ODBC driver.

A memo column in Microsoft Access is a long. Data appearing in the wrong place are a sign of fetching a long where you did not expect it. One strategy to avoid these problems is to always call longreadlen.

integer odbc_num_fields(integer result)

Use odbc_num_fields to find the number of fields in the result set.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");


        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        print(odbc_num_fields($Result));

        odbc_close($Connection);
?>
```

integer odbc_num_rows(integer result)

The odbc_num_rows function returns the number of rows in the set, or the number of rows affected by a delete or insert if the driver supports it.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        print(odbc_num_rows($Result));

        odbc_close($Connection);
?>
```

integer odbc_pconnect(string dsn, string user, string password)

The odbc_pconnect function operates similarly to odbc_connect. A connection is attempted to the specified Data Source Name (DSN) and a connection identifier is returned. The connection should not be closed with odbc_close. It will persist as long as the Web server process. The next time a script executes odbc_pconnect, PHP will first check for existing connections.

```
<?
        // connect to database
        $Connection = odbc_pconnect("store", "guest", "guest");
?>
```

integer odbc_prepare(integer connection, string query)

The odbc_prepare function parses a query and prepares it for execution. A result identifier that may be passed to odbc_execute is returned. Preparing statements can be more efficient than making the driver reparse statements. This is usually the case where you have many rows to insert into the same table. To specify a value to be filled in later, use a question mark.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // prepare query
        $Query = "INSERT INTO catalog (ID, Name, Price) ";
        $Query .= "VALUES(?, ?, ?) ";
        $Result = odbc_prepare($Connection, $Query);

        // insert
        // 0, 2000 Calendar, 20.00
        // 1, 2001 Calendar, 20.50
        // 2, 2002 Calendar, 21.00
        for($index = 2000; $index = 2002; $index++)
```

```
        {
              $values[0] = $index-2000;
              $values[1] = "$index Calendar";
              $values[2] = 20.00 + (0.50 * ($index-2000));

              odbc_execute($Result, $values);
        }

        odbc_close($Connection);
?>
```

string odbc_result(integer result, string field)

Use odbc_result to get the value of a field for the current row. Fields may be referenced by number or name. If by using numbers, start counting fields with 1. If you specify a field by name, do not include the table name.

This function is affected by the settings controlled by odbc_binmode and odbc_longreadlen. An important fact to keep in mind is that while in most cases the value of the field will be returned, fields that contain long data will be echoed to the browser instead by default. Use odbc_longreadlen to change this behavior.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";


      $Result = odbc_do($Connection, $Query);

      while(odbc_fetch_row($Result))
      {
      $name = odbc_result($Result, 1);
      $price = odbc_result($Result, 2);
            print("$name: $price<BR>\n");
      }

      odbc_close($Connection);
?>
```

integer odbc_result_all(integer result, string format)

The odbc_result_all function will dump all the rows for a result set to the browser. The number of rows is returned. The dumped rows will be formatted in a table. The field names will be printed in a header row with TH tags. The optional format argument will be inserted inside the initial table tag so that you may set table attributes.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // execute query
        $Query = "SELECT name, price ";
        $Query .= "FROM catalog ";
        $Result = odbc_do($Connection, $Query);

        // dump all results
        odbc_result_all($Result, "BORDER=1");

        odbc_close($Connection);
?>
```

boolean odbc_rollback(integer connection)

Use odbc_rollback to abandon all pending transactions. By default all queries are automatically committed, but this behavior may be modified with odbc_autocommit. Not all databases support transactions.

```
<?
        // connect to database
        $Connection = odbc_connect("store", "guest", "guest");

        // turn off autocommit
        odbc_autocommit($Connection, FALSE);


         // put everything on sale
         $Query = "UPDATE catalog ";
         $Query .= "SET price = price * 0.9 ";
         $Result = odbc_do($Connection, $Query);

         // rollback
         odbc_rollback($Connection);

         odbc_close($Connection);
?>
```

integer odbc_setoption(integer id, integer function, integer option, integer parameter)

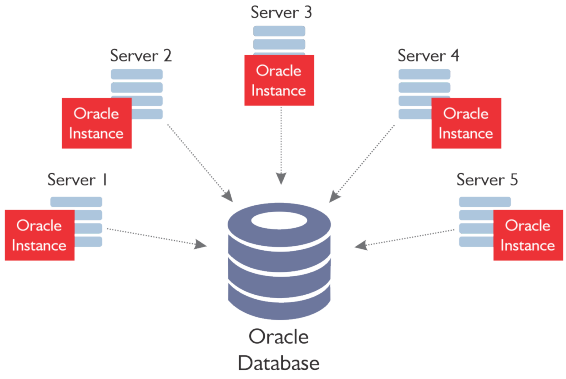
The odbc_setoption function changes the configuration of the ODBC driver for an entire connection or a single result set. Its purpose is to allow access to any ODBC setting in order to avoid problems with buggy ODBC drivers. To use this function, you ought to understand ODBC in greater detail than the average user. You will need to know the values of the various options available to you.

The id argument is either a connection identifier or a result set identifier. Since odbc_setoption wraps two C API functions, SQLSetConnectOption and SQLSetStmtOption,

you must specify which to use with the function argument. The option argument is an integer that identifies one of the many options available on the ODBC driver. The parameter argument is the value to use with the option.

8.9 ORACLE

Oracle is one of the most popular relational databases in the world. It is an industrial strength engine preferred by large corporations using databases of exceeding complexity. Oracle database administrators are scarce and command high salaries.



PHP supports two generations of Oracle libraries, Version 7 and Version 8. The functions that use Oracle 7 begin with ora_, such as ora_logon. The functions that work with Oracle 8 begin with oci, such as ocilogon. The Oracle 8 library supports connecting to older Oracle databases. We have included descriptions of the older functions because it's possible you are in the situation of not having access to the newer libraries. Aside from compiling Oracle support into PHP, you may also load an extension using the dl function.

The Oracle 7 functions require two environment variables to be set: ORACLE_HOME and ORACLE_SID. They are most likely not set for your Web server, so you must use the putenv function to set them.

Thies Arntzen, Stig Bakken, Mitch Golden, Andreas Karajannis, and Rasmus Lerdorf contributed to the Oracle 7 extension. Oracle 8 support was added to PHP by Stig Bakken and Thies Arntzen.

Table 9 lists constants created by the Oracle 8 Extension. When Oracle is installed, it creates a test user. The login is scott and the password is tiger.

Table 9. All Oracle 8 Constants

OCI_ASSOC
OCI_BOTH
OCI_B_BFILE
OCI_B_BIN
OCI_B_BLOB
OCI_B_CFILEE
OCI_B_CLOB
OCI_B_CURSOR
OCI_B_ROWID
OCI_COMMIT_ON_SUCCESS
OCI_DEFAULT
OCI_DESCRIBE_ONLY
OCI_DTYPE_FILE
OCI_DTYPE_LOB
OCI_DTYPE_ROWID
OCI_D_FILE
OCI_D_LOB
OCI_D_ROWID
OCI_EXACT_FETCH
OCI_NUM
OCI_RETURN_LOBS
OCI_RETURN_NULLS
SQLT_BFILEE
SQLT_BLOB
SQLT_CFILEE
SQLT_CLOB
SQLT_RDD

boolean ocibindbyname (integer statement, string placeholder, reference variable, integer length, integer type)

The ocibindbyname function binds an Oracle placeholder to a PHP variable. You must supply a valid statement identifier as created by ociparse, the name of the placeholder, a reference to a PHP variable, and the maximum length of the bind data. You may use a value of -1 to use the length of the variable passed as the variable argument.

The optional type argument specifies a data type and is necessary if you wish to bind to an abstract data type. Use one of the following constants to set the data type: OCI_B_BLOB, OCI_B_CFILE, OCI_B_CLOB, OCI_B_FILE, OCI_B_ROWID. Make sure you use ocinewdescriptor before binding to an abstract data type. You also need to use -1 for the length argument.

```
<?
        //set-up data to insert
        $NewEmployee = array(
              array(8001, 'Smith', 'Clerk'),
              array(8002, 'Jones', 'Analyst'),
              array(8003, 'Atkinson', 'President')
              );

        //connect to database
        $Connection = ocilogon("scott", "tiger");

//assemble query
$Query = "INSERT INTO emp (EMPNO, ENAME, JOB, HIREDATE) ";
$Query .= "VALUES (:empno, :ename, :job, SYSDATE ) ";
$Query .= "RETURNING ROWID INTO :rowid ";

//parse query
$Statement = ociparse($Connection, $Query);

//create descriptor the abstract data type
$RowID = ocinewdescriptor($Connection, OCI_D_ROWID);

//bind input and output variables
ocibindbyname($Statement, ":empno", &$EmployeeNumber, 32);
ocibindbyname($Statement, ":ename", &$EmployeeName, 32);
ocibindbyname($Statement, ":job", &$Job, 32);
ocibindbyname($Statement, ":rowid", &$RowID, -1, OCI_B_ROWID);

         //loop over each new employee
         while(list($key, $EmployeeInfo) = each($NewEmployee))
         {
               list($EmployeeNumber, $EmployeeName, $Job) =
 $EmployeeInfo;

               //execute query, do not automatically commit
               ociexecute($Statement, OCI_DEFAULT);

               print("$EmployeeNumber has ROWID $RowID<BR>\n");
         }

         //free the statement
         ocifreestatement($Statement);
```

```
        //undo the inserts
        //Normally, you won't do this, if we undo the inserts
        //each time, we can run the example over and over
        ocirollback($Connection);

        //close connection
        ocilogoff($Connection);
?>
```

boolean ocicancel(integer statement)

The ocicancel function fetches the next row from a statement. Internally it calls the OCIStmtFetch function, which is part of OCI, specifying zero for the number of rows. In every other way, it is identical to ocifetch.

boolean ocicolumnisnull(integer statement, value column)

Use ocicolumnisnull to test whether a column is null. You may specify columns by number, in which case columns are numbered starting with 1, or you may specify columns by name.

string ocicolumnname(integer statement, integer column)

The ocicolumnname function returns the name of a column given the column number. Columns are numbered starting with 1.

integer ocicolumnsize(integer statement, value column)

The ocicolumnsize function returns the size of a column. You may specify columns by number, in which case columns are numbered starting with 1, or you may specify columns by name

value ocicolumntype(integer statement, integer column)

Use ocicolumntype to get the type of the specified column. You may specify columns by number, in which case columns are numbered starting with 1, or you may specify columns by name. The name of the type will be returned if it is one of the following: BFILE, BLOB, CHAR, CLOB, DATE, LONG RAW, LONG, NUMBER, RAW, REFCURSOR, ROWID, VARCHAR. Otherwise, an integer code from the data type will be returned.

boolean ocicommit(integer connection)

The ocicommit function commits all previous statements executed on the connection. By default, statements are committed when executed. You can override this functionality when you call ociexecute, by specifying a mode.

boolean ocidefinebyname(integer statement, string column, reference variable, integer type)

The ocidefinebyname function associates a column with a PHP variable. When the statement is executed, the value of the column will be copied into the variable. The statement argument must be an integer returned by ociparse. The column name must be written in upper case, otherwise Oracle will not recognize it. Unrecognized column names do not produce errors. Since the variable you pass in ocidefinebyname will be modified, you need to pass it by reference. That mean preceding it with an ampersand (&).

The type argument appears to be necessary only if you are attaching to an abstract data type, such as a ROWID. Abstract data types require ocinewdescriptor be used prior to ocidefinebyname. If the type argument is left out, the variable will be set as a nullterminated string.

```
<?
        //connect to database
        $Connection = ocilogon("scott", "tiger");

        //assemble query
        $Query = "SELECT ENAME, HIREDATE ";
        $Query .= "FROM emp ";
        $Query .= "WHERE JOB='CLERK' ";

        //parse query
        $Statement = ociparse($Connection, $Query);

    //associate two columns with variables
    ocidefinebyname($Statement, "ENAME", &$EmployeeName);
    ocidefinebyname($Statement, "HIREDATE", &$HireDate);

    //execute query
    ociexecute($Statement);

    //fetch each row
    while(ocifetch($Statement))
    {
          print("$EmployeeName was hired $HireDate<BR>\n");
    }
        //free the statement
        ocifreestatement($Statement);

        //close connection
        ocilogoff($Connection);
?>
```

array ocierror(integer identifier)

If an error has occurred, the ocierror function returns an associative array that describes it. If no error has occurred, FALSE is returned. The identifier argument may be either a statement identifier or a connection identifier. The returned array will have two elements, code and message. You may also call ocierror with no argument to get information about a failed login.

boolean ociexecute(integer statement, integer mode)

Use ociexecute to execute a statement. The mode argument is optional. It controls whether the statement will be committed after execution. By default, OCI_COMMIT_ON_EXECUTE is used. If you do not wish to commit the transaction immediately, use OCI_DEFAULT.

boolean ocifetch(integer statement)

The ocifetch function prepares the next row of data to be read with ociresult. When no rows remain, FALSE is returned.

```
<?
        //connect to database
        $Connection = ocilogon("scott", "tiger");

        //assemble query
        $Query = "SELECT * ";
        $Query .= "FROM emp ";

        //parse query
        $Statement = ociparse($Connection, $Query);

        //execute query
        ociexecute($Statement);

       //check that the query executed sucessfully
       if($Error = ocierror($Statement))
       {
             print($Error["code"] . ": " . $Error["message"]
<BR>\n");
             exit;
       }

       //start HTML table
       print("<TABLE>\n");

   //build headers from column information
   print("<TR>\n");
   for($i=1; $i = ocinumcols($Statement); $i++)
   {
```

```
        print("<TH>");

        //print a line like "<TH>ENAME VARCHAR2(10)</TH>"
        print(ocicolumnname($Statement, $i) . " ");
        print(ocicolumntype($Statement, $i));
        print("(" . ocicolumnsize($Statement, $i) . ")");

        print("</TH>\n");
}
print("</TR>\n");


//fetch each row
while(ocifetch($Statement))
{
      print("<TR>\n");

      //loop over each column
      for($i=1; $i = ocinumcols($Statement); $i++)
      {
            //print a line like "<TD>SMITH/TD>"
            print("<TD>");
            if(ocicolumnisnull($Statement, $i))
            {
                  print("(null)");

            }
            else
            {
                  print(ociresult($Statement, $i));
            }
            print("</TD>\n");
       }

       print("</TR>\n");
 }

 //close table

 print("</TABLE>\n");

 //free the statement
 ocifreestatement($Statement);

 //close connection
 ocilogoff($Connection);
```

boolean ocifetchinto(integer statement, reference data, integer mode)

Use ocifetchinto to get the next row of data from an executed statement and place it in an array. The data argument will contain an array that by default will be indexed by integers starting with 1. The optional mode argument controls how the array is indexed. You may add the constants listed in Table 10 to get the features you desire.

Table 10. Constants for Use with ocifetchinto

Constant	Description
OCI_ASSOC	Return columns indexed by name
OCI_NUM	Return columns indexed by number
OCI_RETURN_NULLS	Create elements for null columns
OCI_RETURN_LOBS	Return values of LOBs instead of descriptors

```
<?
        //connect to database
        $Connection = ocilogon("scott", "tiger");

        //assemble query
        $Query = "SELECT * ";
        $Query .= "FROM emp ";

        //parse query
        $Statement = ociparse($Connection, $Query);

        //execute query
        ociexecute($Statement);

        //check that the query executed sucessfully
        if($Error = ocierror($Statement))
        {
              print($Error["code"] . ": " . $Error["message"]
 "<BR>\n");
              exit;
        }

        //start HTML table
        print("<TABLE>\n");

        //fetch each row
        while(ocifetchinto($Statement, $Data,
```

```
        OCI_NUM + OCI_RETURN_NULLS + OCI_RETURN_LOBS))
    {
        print("<TR>\n");

        //loop over each column
        while(list($key, $value) = each($Data))
        {
            //print a line like "<TD>SMITH</TD>"
            print("<TD>$value</TD>\n");
        }

        print("</TR>\n");
    }
  //close table
  print("</TABLE>\n");

  //free the statement
  ocifreestatement($Statement);

        //close connection
        ocilogoff($Connection);
?>
```

integer ocifetchstatement(integer statement, reference data)

The ocifetchstatement function places an array with all the result data in the data argument and returns the number of rows. The data array is indexed by the names of the columns. Each element is an array itself which is indexed by integers starting with zero. Each element in this subarray corresponds to a row.

boolean ocifreecursor(integer cursor)

Use ocifreecursor to free the memory associated with a cursor you created with ocinewcursor.

boolean ocifreestatement(integer statement)

Use ocifreestatement to free the memory associated with a statement. The statement argument is an integer returned by ociparse.

ociinternaldebug(boolean on)

The ociinternaldebug function controls whether debugging information is generated. The debug output will be sent to the browser. It is off by default, of course.

boolean ocilogoff(integer connection)

Use ocilogoff to close a connection.

integer ocilogon(string user, string password, string sid)

The ocilogon function establishes a connection to an Oracle database. The identifier it returns is used to create statements, cursors, and descriptors. The user and password arguments are required. The optional sid argument specifies the server; if it is left out, the ORACLE_SID environment variable will be used.

If you attempt to create a second connection to the same database, you will not really get another connection. This means that commits or rollbacks affect all statements created by your script. If you want a separate connection, use ocinlogon instead.

integer ocinewcursor(integer connection)

Use ocinewcursor to create a cursor. The cursor identifier that is returned is similar to a statement identifier. Use ocifreecursor to free the memory associated with a cursor. You can use a cursor to get the data returned by a stored procedure.

```
<?
        //open connection
        $Connection = ocilogen ("scott", "tiger");

        //create cursor

 $Cursor = ocinewcursor ($Connection);

 //create statement that calls a stored procedure
 $Query = "BEGIN ";
 $Query .= "docalculation;(:price); ";
 $Query .= "END; ";
 $Statement = ociparse ($Connection, $Query);

 //bind placeholder to cursor
 ocibindbyname ($Statement, "price", &$Cursor, -1, OCI_B_CURSOR)

 //execute statement
 ociexecute ($Statement);

 //execute cursor
 ociexecute($Cursor);

 //loop over results in cursor
 while (ocifetchinto ($Cursor, &$Results))
 {
        print("$Results<BR>\n")
 }

 //free memory for cursor
 ocifreecursor ($Cursor);
```

```
//free memory for statement
ocifreestatement ($Statement);

//close connection
ocilogoff ($Connection);
```

string ocinewdescriptor(integer connection, integer type)

The ocinewdescriptor function allocates memory for descriptors and LOB locators. The type defaults to being a file, but you may specify OCI_D_FILE, OCI_D_LOB, or OCI_D_ROWID.

integer ocinlogon(string user, string password, string sid)

The ocinlogon function establishes a unique connection to an Oracle database. The identifier it returns is used to create statements, cursors, and descriptors. The user and password arguments are required. The optional sid argument specifies the server, and if left out, the ORACLE_SID environment variable will be used.

Compare this function to ocilogon and ociplogon.

integer ocinumcols(integer statement)

The ocinumcols function returns the number of columns in a statement.

integer ociparse(integer connection, string query)

The ociparse function creates a statement from a query. It requires a valid connection identifier.

integer ociplogon(string user, string password, string sid)

The ociplogon function establishes a persistent connection to an Oracle database. These connections exist as long as the server process. When you request a persistent connection, you may get a connection that already exists, thus saving the overhead of establishing a connection. The returned identifier is used to create statements, cursors, and descriptors. The user and password arguments are required. The optional sid argument specifies the server, and if left out, the ORACLE_SID environment variable will be used. Compare this function to ocilogon and ocinlogon.

string ociresult(integer statement, value column)

Use ociresult to get the value of a column on the current row. The column may be identified by number or name. Columns are numbered starting with 1. Results are returned as strings, except in the case of LOBs, ROWIDs, and FILEs.

boolean ocirollback(integer connection)

Use ocirollback to issue a rollback operation on the given connection. By default, calls to ociexecute are committed automatically, so be sure to override this functionality if you wish to use ocirollback.

integer ocirowcount(integer statement)

The ocirowcount function returns the number of rows affected by an update, insert, or delete.

string ociserverversion(integer connection)

Use ociserverversion to get a string describing the version of the server for a connection.

integer ocisetprefetch(integer statement, integer size)

The ociprefetch function sets the size of a buffer that Oracle uses to prefetch results into. The size argument will be multiplied by 1024 to set the actual number of bytes.

string ocistatementtype(integer statement)

Use ocistatementtype to get a string that describes the type of the statement. The types you can expect are ALTER, BEGIN, CREATE, DECLARE, DELETE, DROP, INSERT, SELECT, UNKNOWN, and UPDATE.

boolean ora_bind(integer cursor, string variable, string parameter, integer length, integer type)

The ora_bind function binds a PHP variable to a parameter in an Oracle query. This causes data to flow between the two entities. You must call ora_parse before binding any variables. The type parameter is optional. It specifies whether data may go only into or out of the Oracle parameter. By default, data may go both ways. The type may be defined using the following constants: ORA_BIND_IN, ORA_BIND_INOUT, ORA_BIND_OUT.

ROLE MODEL

LARRY ELLISON: FORMER CHIEF EXECUTIVE OFFICER (CEO) OF ORACLE CORPORATION

Lawrence Joseph Ellison (born August 17, 1944) is an American businessman and investor who is the co-founder, executive chairman, chief technology officer (CTO) and former chief executive officer (CEO) of Oracle Corporation. As of April 2021, he was listed by Bloomberg Billionaires Index as the ninth-wealthiest person in the United States and as the tenth-wealthiest in the world, with a fortune of $93.9 billion, increased from $57.3 billion in 2018. He is also the owner of the 41st largest island in the United States, Lanai in the Hawaiian Islands with a population of just over 3000.

Early life and education

Larry Ellison was born in New York City, to an unwed Jewish mother. His biological father was an Italian-American United States Army Air Corps pilot. After Ellison contracted pneumonia at the age of nine months, his mother gave him to her aunt and uncle for adoption. He did not meet his biological mother again until he was 48.

Ellison moved to Chicago's South Shore, then a middle-class neighborhood. He remembers his adoptive mother as warm and loving, in contrast to his austere, unsupportive, and often distant adoptive father, who had chosen the name Ellison to honor his point of entry into the United States, Ellis Island. Louis Ellison was a government employee who had made a small fortune in Chicago real estate, only to lose it during the Great Depression.

Although Ellison was raised in a Reform Jewish home by his adoptive parents, who attended synagogue regularly, he remained a religious skeptic. At age thirteen, Ellison refused to have a bar mitzvah celebration. Ellison states: "While I think I am religious in one sense, the particular dogmas of Judaism are not dogmas I subscribe to. I don't believe that they are real. They're interesting stories. They're interesting

mythology, and I certainly respect people who believe these are literally true, but I don't. I see no evidence for this stuff." Ellison says that his fondness for Israel is not connected to religious sentiments, but rather due to the innovative spirit of Israelis in the technology sector.

Ellison attended South Shore High School in Chicago and later was admitted to University of Illinois at Urbana–Champaign and was enrolled as a premed student. At the university, he was named science student of the year. He withdrew without taking final exams after his sophomore year, because his adoptive mother had just died. After spending the summer of 1966 in California, he then attended the University of Chicago for one term, studying physics and mathematics. He did not take any exams, but he first encountered computer design there. In 1966, aged 22, he moved to Berkeley, California.

Early career and Oracle

While working at Ampex in the early 1970s, he became influenced by Edgar F. Codd's research on relational database design for IBM. That led in 1977 to the formation of the company which later became Oracle. Oracle became a successful database vendor to mid- and low-range systems, later competing with Sybase (created 1984) and Microsoft SQL Server (a port of Sybase created in 1989) which led to Ellison being listed by Forbes as one of the richest people in the world.

1977–1994

During the 1970s, after a brief stint at Amdahl Corporation, Ellison began working for Ampex Corporation. His projects included a database for the CIA, which he named "Oracle". Ellison was inspired by a paper written by Edgar F. Codd on relational database systems called "A Relational Model of Data for Large Shared Data Banks". In 1977, he founded Software Development Laboratories (SDL) with two partners and an investment of $2,000; $1,200 of the money was his.

In 1979, the company renamed itself Relational Software Inc. Ellison had heard about the IBM System R database, also based on Codd's theories, and wanted Oracle to achieve compatibility with it, but IBM made this impossible by refusing to share System R's code. The initial release of the Oracle Database in 1979 was called Oracle version 2; there was no Oracle version. In 1983, the company officially became Oracle Systems Corporation after its flagship product. In 1990, Oracle laid off 10% of its workforce (about 400 people) because it was losing money. This crisis, which almost resulted in the company's bankruptcy, came about because of Oracle's "up-front" marketing strategy, in which sales people urged potential customers to buy the largest possible amount of software all at once. The sales people then booked the value of future license sales in the current quarter, thereby increasing their bonuses. This became a problem when the future sales subsequently failed to materialize. Oracle eventually

had to restate its earnings twice, and had to settle class-action lawsuits arising from its having overstated its earnings. Ellison would later say that Oracle had made "an incredible business mistake".

Although IBM dominated the mainframe relational database market with its DB2 and SQL/DS database products, it delayed entering the market for a relational database on Unix and Windows operating systems. This left the door open for Sybase, Oracle, Informix, and eventually Microsoft to dominate mid-range systems and microcomputers. Around this time, Oracle fell behind Sybase. From 1990 to 1993, Sybase was the fastest-growing database company and the database industry's darling vendor, but soon it fell victim to merger mania. Sybase's 1996 merger with Powersoft resulted in a loss of focus on its core database technology. In 1993, Sybase sold the rights to its database software running under the Windows operating system to Microsoft Corporation, which now markets it under the name "SQL Server".

In his early years at Oracle, Ellison was named an Award Recipient in the High Technology Category for the Ernst and Young Entrepreneur of the Year Program.

1994–2010

In 1994, Informix overtook Sybase and became Oracle's most important rival. The intense war between Informix CEO Phil White and Ellison was front-page Silicon Valley news for three years. In April 1997, Informix announced a major revenue shortfall and earnings restatements. Phil White eventually landed in jail, and IBM absorbed Informix in 2001. Also in 1997, Ellison was made a director of Apple Computer after Steve Jobs returned to the company. Ellison resigned in 2002. With the defeat of Informix and of Sybase, Oracle enjoyed years of industry dominance until the rise of Microsoft SQL Server in the late 1990s and IBM's acquisition of Informix Software in 2001 to complement their DB2 database. As of 2013 Oracle's main competition for new database licenses on UNIX, Linux, and Windows operating systems comes from IBM's DB2 and from Microsoft SQL Server. IBM's DB2 still dominates the mainframe database market.

In 2005, Oracle Corporation paid Ellison a $975,000 salary, a $6,500,000 bonus, and other compensation of $955,100. In 2007, Ellison earned a total compensation of $61,180,524, which included a base salary of $1,000,000, a cash bonus of $8,369,000, and options granted of $50,087,100. In 2008, he earned a total compensation of $84,598,700, which included a base salary of $1,000,000, a cash bonus of $10,779,000, no stock grants, and options granted of $71,372,700. In the year ending May 31, 2009, he made $56.8 million. In 2006, Forbes ranked him as the richest Californian. In April 2009, after a tug-of-war with IBM and Hewlett-Packard, Oracle announced its intent to buy Sun Microsystems. On July 2, 2009, for the fourth year in a row, Oracle's board awarded Ellison another 7 million stock options. On August 22, 2009, it was reported that Ellison

would be paid only $1 for his base salary for the fiscal year of 2010, down from the $1,000,000 he was paid in fiscal 2009.

2010–present

The European Union approved Oracle's acquisition of Sun Microsystems on January 21, 2010, and agreed that Oracle's acquisition of Sun "has the potential to revitalize important assets and create new and innovative products". The Sun acquisition also gave Oracle control of the popular MySQL open source database, which Sun had acquired in 2008. On August 9, 2010, Ellison denounced Hewlett-Packard's board for firing CEO Mark Hurd, writing that "the HP board just made the worst personnel decision since the idiots on the Apple board fired Steve Jobs many years ago." (Ellison and Hurd were close personal friends.) Then on September 6, Oracle hired Mark Hurd as co-president alongside Safra Catz. Ellison remained in his current role at Oracle.

In March 2010, the Forbes list of billionaires ranked Ellison as the sixth-richest person in the world and as the third-richest American, with an estimated net worth of US$28 billion. On July 27, 2010, The Wall Street Journal reported that Ellison was the best-paid executive in the last decade, collecting a total compensation of US$1.84 billion. In September 2011, Ellison was listed on the Forbes list of billionaires as the fifth richest man in the world and was still the third richest American, with a net worth of about $36.5 billion. In September 2012, Ellison was again listed on the Forbes list of billionaires as the third richest American citizen, behind Bill Gates and Warren Buffett, with a net worth of $44 billion. In October 2012, he was listed just behind David Hamilton Koch as the eighth richest person in the world, according to the Bloomberg Billionaires Index. Ellison owns stakes in Salesforce.com, NetSuite, Quark Biotechnology Inc. and Astex Pharmaceuticals. In June 2012, Ellison agreed to buy 98 percent of the Hawaiian island of Lana'i from David Murdock's company, Castle & Cooke. The price was reported to be between $500 million and $600 million. In 2005, Ellison agreed to settle a four-year-old insider-trading lawsuit by offering to pay $100 million to charity in Oracle's name.

In 2013, according to the Wall Street Journal, Ellison earned $94.6 million. On September 18, 2014, Ellison appointed Mark Hurd to CEO of Oracle from his former position as President; Safra Catz was also made CEO, moving from her former role as CFO. Ellison assumed the positions of chief technology officer and executive chairman.

In November 2016, Oracle bought NetSuite for $9.3 billion. Ellison owned 35% of NetSuite at the time of the purchase making him $3.5 billion personally.

In 2017, Forbes estimated that Ellison was the 4th richest person in tech.

In June 2018, Ellison's net worth was about $54.5 billion, according to Forbes.

In December 2018, Ellison became a director on the board of Tesla, Inc., after purchasing 3 million shares earlier that year.

As of December 31, 2019, Ellison owns 36.2% of the shares of Oracle Corporation, and 1.7% of the shares of Tesla.

As of June 2020, Ellison is said to be the seventh wealthiest person in the world, with a net worth of $66.8 billion.

In December 2020, his net worth increased by $2.5 billion in a single week as Oracle's stock jumped by 4% between November 27 and December 4.

SUMMARY

- PHP offers support for many databases. Open source relational databases are well represented, as are many commercial products.
- The dbase_create function creates a dBase database. The fields argument is an array of arrays that describe fields. Each array may have up to four elements. In order, they are name, type, length, and precision.
- When rows are deleted in a dbase database, they are simply marked for deletion. Use the dbase_pack function to permanently remove these rows, thus packing the database.
- The DBA functions abstract communications with databases that conform to the style of Berkeley DB database systems.
- Use dba_optimize to optimize a database, which usually consists of eliminating gaps between records created by deletes.
- filePro is a relational database by fP Technologies. There are versions for win32 and SCO UNIX. PHP only supports reading from filePro databases.
- PHP includes support for two parts of the Informix API, ODS and IUS.
- The ifx_copy_blob function makes a copy of an existing blob and returns the identifier to the new blob.
- The ifx_query function executes a query and returns a result identifier, which most of the other Informix functions require.
- InterBase is a full-featured database that spent much of its life as closed-source and proprietary.
- mSQL or Mini SQL is a lightweight database management system from Hughes Technologies. There are a handful of mSQL functions that exist simply for backward compatibility.
- The msql_db_query function is identical to msql_query except that it specifies a database rather than using the database selected with msql_ select_db.
- The msql_list_dbs function returns a result identifier as if the database were queried with msql_query.
- MySQL is a relational database with a license that allows you to use it cost-free for most noncommercial purposes.
- Open database connectivity (ODBC) has become an industry standard for communicating with a database. The model is simple. Client software is designed to use an ODBC API.
- The odbc_pconnect function operates similarly to odbc_connect. A connection is attempted to the specified Data Source Name (DSN) and a connection identifier is returned.

KNOWLEDGE CHECK

1. **Which of the following DBMSs do not have a native PHP extension?**
   a. MySQL
   b. IBM DB/2
   c. Microsoft SQL Server
   d. None of the above
2. **In PHP in order to access MySQL database you will use:**
   a. mysqlconnect() function
   b. mysql-connect() function
   c. mysql_connect() function
   d. sql_connect() function
3. **Which one of the following databases has PHP supported almost since the beginning?**
   a. Oracle Database
   b. SQL
   c. SQL+
   d. MySQL
4. **The updated MySQL extension released with PHP 5 is typically referred to as .....**
   a. MySQL
   b. mysql
   c. mysqli
   d. mysqly
5. **Which one of the following lines need to be uncommented or added in the php.ini file so as to enable mysqli extension?**
   a. extension=php_mysqli.dll
   b. extension=mysql.dll
   c. extension=php_mysqli.dl
   d. extension=mysqli.dl

REVIEW QUESTIONS

1. What are the dBase field types? Discuss.
2. Write short note on DBM-style database abstraction.
3. Where can we find information about solving errors that might occur during backup and restore?
4. Discuss about open database connectivity (ODBC).
5. Differentiate between Version 7 and Version 8 oracle libraries.

Check Your Result

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

REFERENCES

1. Appelt, Dennis, Nadia Alshahwan, and Lionel Briand. "Assessing the impact of firewalls and database proxies on SQL injection testing." In Future Internet Testing, pp. 32-47. Springer International Publishing, 2014.
2. Bandhakavi, S., Bisht, P. and Madhusudan, P. (2007). CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluation. Proceeding of the ACM conference on computer and communication security.
3. Bhanderi, A. and Rawal, N. (2015). A review and Detection mechanism for SQL injection attacks. International Journal of Innovative Research in Science, Engineering & Technology, 4(12), 12446-12452.
4. Buehrer, Gregory, Bruce W. Weide, and Paolo AG Sivilotti. "Using parse tree validation to prevent SQL injection attacks." In Proceedings of the 5th international workshop on Software engineering and middleware, pp. 106-113. ACM, 2005.
5. Ettore, M., Dominic., L. and Giuliano., A. (2007). Automated Protection of PHP Applications Against SQL-injection Attacks. Software Maintenance and Reengineering, 11th European Conference IEEE CNF. Retrieved from http://ieeexplore.ieee.org
6. Francisco Jose Marques Vieira, "Realistic Vulnerability Injection in PHP Web Applications" Lisboa.2011.
7. Gehani, Narain , "The Database Book: Principles & Practice using MySQL",2007, 91-133, 279- 290.
8. Halfond, William GJ, and Alessandro Orso. "AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks." In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, pp. 174-183. ACM, 2005.
9. Johari, R. and Shanna, P. (2012). A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection. Proceedings of the International Conference on Communication Systems and Network Technologies, 453-458.
10. Kirti, R., Vishal, M. (2015). Security Engine for prevention of SQL Injection and CSS Attacks using Data Sanitization Technique. Proceeding to international journal of innovative Research in computer and communication engineering, 3(6).
11. Kumar, P. and Pateriya, P. (2012). A survey on SQL Injection attacks, detection and prevention techniques. Proceedings of the 3rd International Conference Computing Communication and Networking Technologies, 1-5.
12. Maori, O. & Schulman, "A Blind SQL Injection: Imperva ADC",2008, available at http://www.imperva.com/resources/adc/blind_sql_server_injection.html.

13. Stephen, T., Laurie, W. (2007). Using Automated Fix Generation to Secure SQL Statements. Software Engineering for Secure Systems IEEE CNF. Retrieved November 6, 2007, from http://ieeexplore.ieee.org
14. Subburaj, R., Varsha, V. (2016). Preventing Structured Query Language (SQL) Injection Attacks in Mobile Applications. Proceeding of International Journal of Control Theory and Applications ISSN: 0974–5572 © International Science Press 9(40).
15. Usage of server-side programming languages for websites, available at URL:http://w3techs.com/technologies/overview/programming_language/all, 2011.
16. Wei, M., M., Suraj, K. (2006). Preventing SQL Injection Attacks in Stored Procedures. Proceedings of the 2006 Australian Software Engineering Conference.
17. Win, Witty, and Hnin Htun. "A Simple and Efficient Framework for Detection of SQL Injection Attack." IJCCER 1, no. 2: 26-30, 2013.

CHAPTER

9

DESIGNING AND DEBUGGING

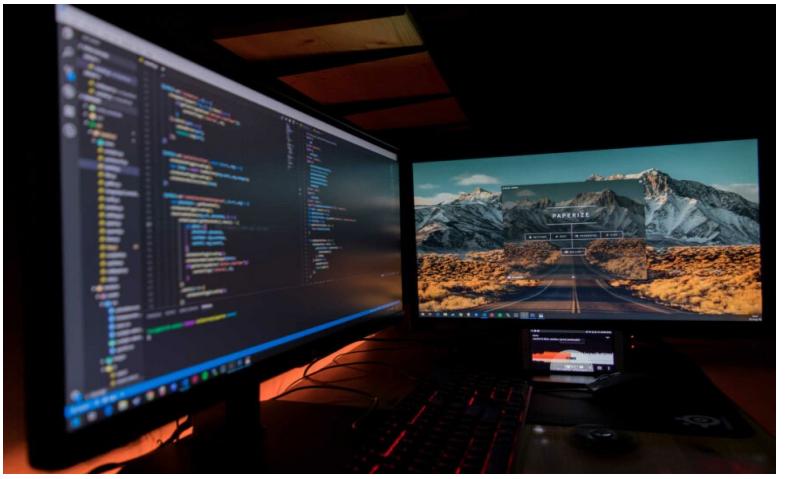
"If debugging is the process of removing software bugs, then programming must be the process of putting them in"

---Edsger Dijkstra

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Write requirements specifications
2. Write design documents
3. Discuss the benefits of concurrent versions system (CVS)
4. Deal with FreeEnergy system
5. Describe the uses of FastTemplate
6. Explain how to run a script regularly
7. Describe efficiency and debugging

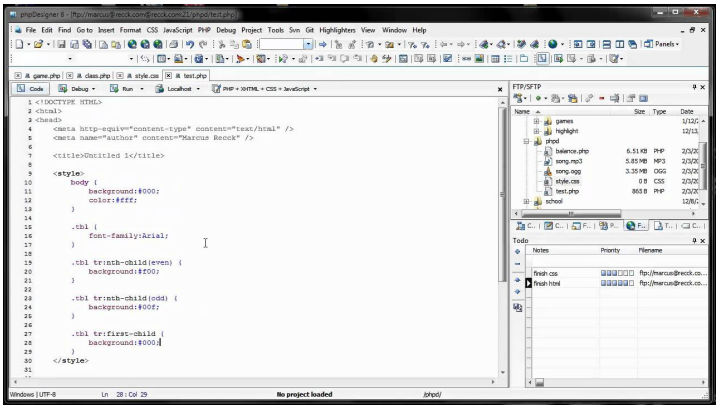


INTRODUCTION

Building a Web site with PHP is not the same as building a static Web site. If you choose simply to sprinkle PHP code

occasionally throughout the site, the effect may be minimal, of course. If you choose to use PHP to generate every page, you will find many opportunities for transforming patterns into functions.

Integration with HTML, elements such as opening and closing body tags can be put into a function or an included file. The consequence of this situation is that you no longer have a Web site. You have a Web application.



When this happens, it becomes more important to draw upon formal development techniques. Certainly, structured design is useful when building static Web sites. The case is made plainly in Web Site Engineering by Thomas Powell. The addition of PHP makes careful design critical.

Debugging is the process of finding and resolving bugs (defects or problems that prevent correct operation) within computer programs, software, or systems. Debugging tactics can involve interactive debugging, control flow analysis, unit testing, integration testing, log file analysis, monitoring at the application or system level, memory dumps, and profiling. Many programming languages and software development tools also offer programs to aid in debugging, known as debuggers.

9.1 WRITING REQUIREMENTS SPECIFICATIONS

Before you can design a system, it is important to understand what it's supposed to do. Too often this comes in the form of a verbal request such as, "We need a home page with a guest book and a visitor counter," which is never further defined. This usually leads to the building of a prototype that is 25 percent of what the client wants. Changes are made to the prototype, and the site is now 50 percent of what the client wants now. During the time the changes were made, the client has moved the target.

The solution to this problem is to set a target and stick with it. This should start with a statement of the goals for the project. In experience the most important question

left unasked is about motivation. When a client asks for a large animated scene to appear on their index page, often the motivation is a desire to seem familiar with leading-edge technology. Instead of blindly fulfilling the client's request, it is better to look for the best solution for the "Why?" A slick graphical design can say more about the client's attention to advances in technology.

Once you have asked "Why?" enough times, you should have a list of several goals for the project. These goals should suggest a set of requirements. If one of the system's goals is to generate more business, one requirement may be to raise visitor awareness of items in the client's catalog. This may evolve into a requirement that products appear throughout the site on a rotational basis. This could be implemented as banners or kickers strategically placed within the site. Don't, however, tie yourself down with design issues. This earliest stage of site development should concentrate solely on the goals of the system.

Keyword

Natural language is any language that has evolved naturally in humans through use and repetition without conscious planning or premeditation.

From a solid base of goals, you can begin to describe the system requirements. This usually takes the form of a requirements specification document, a formal description of the black-box behavior expected from the site. The goals will suggest a collection of functional requirements and constraints on the design. As we have said, having a goal of increasing sales suggests, among other things, that the site should raise customer awareness of catalog items. Another requirement could be that the site provide some free service to attract visitors. An example is a loan company offering a mortgage calculator. It is a good idea to informally explore possible solutions to requirements, but it's still important to keep design decisions out at this time.

The requirements specification is formal and structured, but it should be understandable by nonexperts in the implementation technology. The description of the system's behavior serves partially as a contract between the client and developer. Clear statements will eliminate misunderstandings that have a high cost later in development. That is not to say that the document should not be precise. When possible, state requirements in measurable terms. Constraining page size to 30K is an objective standard and easily tested. Requiring the

site to inspire confidence in the client company is not easily measurable, but sometimes it's all you have.

Table 1 lists six things toward which a requirements specification should aspire. It should only specify external behavior. Every requirement should be expressed as the answer to a "What?" question. It should specify constraints. These are best expressed as quantities: How many hits per day? Maximum page size? Maximum page depth? The requirements specification should allow you to change it later. While you should use **natural language**, do not write a long narrative. Number sections of the document and use diagrams where necessary. It should be a document that helps a future programmer learn about the system.

The requirements should pay attention to the entire life of the system. If the system needs to be able to recover from a catastrophic failure within an hour, write it into the specification. And the follow-up to this idea is that you should describe how the system deals with adversity—not just disaster, but also illegal user input. Some systems ignore user input that is not understood. How many times have you seen a "404 Document Not Found" error? It's nice when that page includes a link to the index page of the site.

Keeping these guidelines in mind, refer to Table 2, which outlines the structure of a requirements specification. The overview should be a page or less that reviews the goals of the site. If the goals were detailed in another document, make this document available. It is important to preserve the thought that went into the project at each phase. The requirements build on the goals, and in turn the design will build on the requirements. But being able to refer to the original goals of the system will be helpful to the designer and even the implementer.

Table 1. Properties of requirements specifications

Only specifies external system behavior
Specifies constraints on the implementation
Allows easy modification
Serves as a reference tool for system maintainers
Records forethought about the life cycle of the system
Characterizes acceptable responses to undesired events

Table 2. Requirements specification document structure

Overview of System Goals
Operating and Development Environments
External Interfaces and Data Flow
Functional Requirements
Performance Requirements
Exception Handling
Implementation Priorities
Foreseeable Modifications
Design Suggestions

The operating and development environments are sometimes overlooked in requirements specifications. This includes both the browser and the Web server. If you are developing an intranet application, you may be fortunate enough to optimize for a particular browser version. We have found that while a large company may impose a standard browser for the organization for which you've developed an application, another standard may apply to the users in another organization a thousand miles away.

The Web server is perhaps more under your control and certainly less finicky about differences in source code. If you are using PHP, most likely you will be using Apache. It's a good idea to use identical versions of both Apache and PHP for your development and live environments.

For the most part, your list of external interfaces will include the Internet connection between the browser and the Web server, the local file system, and possibly a database connection. We find it helpful to create a diagram that shows the relationship between data elements, the simplest of which might be a box labeled Browser connected to a box labeled Server. The line would have arrows at each end to show that information travels in both directions. This diagram is a description of the context, not a design of the data structure. Whether you will be using a database may be obvious, but which database may not. If the system will be storing data somehow, just show data flowing into a box that could be database or flat file. The goal is to describe how data moves around in the system.

The functional requirements will certainly be the largest part of the document. If you have drawn a data flow diagram, you may have a very good idea of how the system breaks up into modules. The more you can partition the functionality into distinct pieces, the easier it will be to group the functional requirements. We have written

many requirements documents for Web applications that are essentially **data warehouses**. Approach has been to dedicate a section to each of the major data entities. A project management application might have a collection of project descriptions, a collection of users, and a collection of comments. Each of these would have a section in the functional requirements that lists first all the information they store and then the ways the information can be manipulated.

Keyword

Data warehouse is a type of data management system that is designed to enable and support business intelligence (BI) activities, especially analytics

The performance requirements are constraints on the functionality. You may wish to outline a minimum browser configuration for use of the site. Maximum page weights are a good idea. If the client is dictating that a certain technology be used, it. It's good to know in advance that while you will be allowed to use PHP, you have to deal with Oracle and Internet Information Server on Windows NT.

The exception-handling describes how the system deals with adversity. The two parts of this are disaster and invalid input. Discuss what should happen if the Web server suddenly bursts into flame. Decide whether backups will be made hourly, daily, or weekly. Also decide how the system handles users entering garbage. For example, define whether filling out a form with a missing city asks the user to hit the back button or automatically redisplays the form with everything filled out and the missing field marked with a red asterisk.

If the client has a preference for the order of implementation, outline it. Our experience has been that, faced with a dire deadline before the project begins, the client will bargain for which functionality will appear in the first round. Other requirements may not be critical to the system, and the client is willing to wait. If there is a preference in this area, it is very important for the designer and implementers to know in advance.

Farther in the future are the foreseeable modifications. The client may not be ready to create a million-dollar e-commerce site just yet, but they may expect to ask you to plug this functionality into the site a year from now. It may not make sense to use an expensive database to implement a 50-item catalog, but building a strong foundation for later expansion will likely be worthwhile.

The last part of the requirements specification is a collection of design hints. This represents the requirements

writer's forethought about pitfalls for the designer. You might summarize a similar project. You might suggest a design approach.

9.2 WRITING DESIGN DOCUMENTS

Once you have created a requirements specification document, you will have to decide whether to write a design document. Often it is not necessary, especially when a few people are working on a small project. You may wish to choose key elements of a complete design document and develop them to the point of usefulness.

The first part of design is concerned with the architecture of the system. The system should be broken into sections that encompass broad groups of functionality. A **Web application** for project management might break down into a module that handles project information, a module that handles users, and a module that handles timesheet entries. An informational Web site can be broken down by the secondary pages—that is, the pages one click away from the home page. The "About Us" section serves to inform visitors about the company itself, while a catalog area is a resource for learning about the items the company sells Depending on the type of site, you should choose some sort of diagram that shows the subsystems and how they relate to each other. These are called entity relationship diagrams. We almost always create a page-flow diagram. Each node in the graph is a page as experienced by the user. Lines representing links connect the page to other pages on the site. Another useful diagram is one that shows the relationship between database tables. Nodes represent tables, and you may wish to list the fields inside boxes that stand for the tables. Lines connect tables and show how fields match. It is also helpful to indicate whether the relationship between the tables is one to one or one to many.

Keyword

Web application is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system of the device

The next phase of design is interface specification. This defines how subsystems communicate. It can be as simple as listing the URLs for each page. If the site has forms, all the fields should be enumerated. If you are tracking user sessions, you will want to specify how you will be doing this, with cookies or form variables. Define acceptable values for the

session identifier. If the site will be communicating with files or a database, this phase will define names of files or login information for databases.

The largest part of a design document is a detailed description of how each module works. At this point it's acceptable to specify exactly the method for implementing the module. For example, you may specify that a list of catalog items be presented using the UL tag. On the other hand, if it does not matter, we suggest leaving it out. The person writing the actual code will probably have the best idea for solving the problem.

We'd like to present some design ideas you may choose to adopt. PHP's dynamic nature allows for structural designs that cannot be achieved in plain HTML. It is a shame to waste this functionality by using PHP as a faster alternative to CGI. We encourage you to consider using PHP as the engine that powers a completely **dynamic Web site**.

Keyword

Dynamic web page is a web page that displays different content each time it's viewed. For example, the page may change with the time of day, the user that accesses the webpage, or the type of user interaction. There are two types of dynamic web pages.

9.3 USING CVS

CVS, Concurrent Versions System, is an open-source system that allows developers to track all changes to a project. A central repository stores the files that make up the project. Developers check out copies, modify them, and check them back in. The system records all changes, which allows team members to check out any previous version of any given file. The system also is able to merge differences if two developers make independent changes to the same file.

In the context of project management, we have come to believe CVS is essential. It allows developers to collaborate efficiently, even when they are in separate locations. Popular Net projects such as Apache, Linux, and PHP use CVS and probably would not be as successful if they did not.

Modularization Using include

Despite its name, the include function is not equivalent to C's preprocessor command of the same name. In many ways it is like a function call. Given the name of a file, PHP attempts to parse the file as if it appeared in place of the call to include. The difference from a function is that the code will be parsed only if the include statement is executed. You can take advantage of this by wrapping calls to include in if statements. The require function, on the other hand, will always include the specified file, even if it is inside an if block that is never executed. It has been discussed several times on the PHP mailing list that require is faster than include because PHP is able to inject the specified file into the script during an early pass across the code. However, this applies only to files specified by a static path. If the call to require contains a variable, it cannot be executed until the runtime. It may be helpful to adopt a rule of using require only when outside a compound statement and when specifying a static path.

Almost anything we write in PHP uses include extensively. The first reason is that it makes the code more readable. But the other reason is that it breaks the site into modules. This allows multiple people to work on the site at once. It forces you to write code that is more easily reused, within the existing site and on your next project. Most Web sites have to rely on repeating elements. Consistent navigation aids the user, but it is also a major problem when building and maintaining the site. Each page has to have a duplicate code block pasted into it. Making this a module and including it allows you to debug the code once, making changes quickly.

You can adopt a strategy that consists of placing functions into include modules. As each script requires a particular function, you can simply add an include. If your library of functions is small enough, you might place them all into one file. However, you likely will have pieces of code that are needed on just a handful of pages. In this case, you'll want this module to stand alone.

As your library of functions grows, you may discover some interdependencies. Imagine a module for establishing a connection to a database, plus a couple of other modules that rely on the database connection. Each of these two scripts will include the database connection module. But what happens when both are themselves included in a script? The database module is included twice. This may cause a second connection to be made to the database, and if any functions are defined, PHP will report the error of a duplicate function.

In C, programmers avoid this situation by defining constants inside the included files, and you can adopt a similar strategy. You can define a constant inside your

module. If this constant is already defined when the module is executed, control is immediately returned to the calling process. A function named printBold is defined in Listing 1. This function is needed in the script shown in Listing 2. We have purposely placed a bug in the form of a second include. The second time the module is included, it will return before redeclaring the function.

Listing 1 Preventing a Double Include

```
<?
        /*
        ** Avoid double includes
        */
        $included_flag = 'INCLUDE_' . basename(__FILE__);
        if(defined($included_flag))
        {
                return(TRUE);
        }
        define($included_flag, TRUE);

        function printBold($text)
        {
                print("<B>$text</B>");
        }
?>
```

Listing 2 Attempting to Include a Module Twice

```
<?
        //load printBold function
        include("21-1.php");

        //try loading printBold function again
        include("21-1.php");

        printBold("Successfully avoided a second include");
?>
```

Did You Know?

Concurrent Versions System (CVS, also known as the Concurrent Versioning System) is a revision control system originally developed by Dick Grune in July 1986. CVS operates as a front end to RCS, an earlier system which operates on single files. It expands upon RCS by adding support for repository-level change tracking, and a client-server model.

9.4 FREEENERGY

We used the technique of including modules on several Web applications, and it led me to consider all the discrete elements of a Web page. Headers and footers are obvious, and so are other repeating navigational elements. Sometimes you can divide pages up into the content unique to the page, the stuff that comes before it, and the stuff that comes after it. This could be hard to maintain, however. Some of the HTML is

in one file, some in another. If nothing else you will need to flip between two editor windows.

Consider for a moment a Web page as an object—that is, in an object-oriented way. On the surface, a Web page is a pair of HTML tags containing HEAD tags and BODY tags. Regardless of the design or content of the page, these tags must exist, and inside them will be placed further tags. Inside the BODY tags a table can be placed for controlling the layout of the page. Inside the cells of the table are either links to other pages on the site or some content unique to the page.

FreeEnergy is a system that attempts to encapsulate major pieces of each page into files to be included on demand. Before we proceed, we want to state motivations clearly. Our first concern when developing a Web site is that it be correct and of the highest quality. Second is that it may be developed and maintained in minimal time. After these needs are addressed, we consider performance. Performance is considered last because of the relatively cheap cost of faster hardware. Moore's law suggests that eighteen months from now, CPU speed and memory capacity will have doubled for the same price. This doubling costs nothing but time. Also, experience has shown that a small minority of code contributes to a majority of the time spent processing.

The FreeEnergy system uses more calls to include than you'd find where you are simply making a few includes at the top of your pages. Hits to the file system do take longer than function calls, of course. You could place everything you might need in one large file and include it on every page, but you will face digging through that large file when you need to change anything. A trade has been made between the performance of the application and the time it takes to develop and maintain it.

We called this system FreeEnergy because it seems to draw power from the environment that PHP provides. The include function in PHP is quite unique and central to FreeEnergy, especially the allowance for naming a script with a variable. The content unique to a page is called a screen. The screen name is passed to a single PHP script, which references the screen name in a large array that matches the screen to corresponding layout and navigation modules.

The FreeEnergy system breaks Web pages into five modules: action, layout, navigation, screen, and utility. Action modules perform some sort of write function to a database, a file, or possibly to the network. Only one action module executes during a request, and they are executed before the screen module. An action module may override the screen module named in the request. This is helpful in cases where an action module is attempting to process a form and the submitted data are incomplete or otherwise unsatisfactory. Action modules never send data directly to the screen. Instead, they add messages to a stack to be popped later by the layout module. It is possible that an action module will send header information, so it's important that no output be produced.

Layout modules contain just enough code to arrange the output of screen and navigation modules. They typically contain table tags, as is the custom for controlling the layout of a Web page. Inside the table cells, calls to include are placed. They may be invoking navigation modules, or screen modules.

Navigation modules contain links and repeating elements. In the vernacular used by engineers we work with, these are "top nav," "bottom nav," and "side nav." Consider the popular site, Yahoo. Their pages generally consist of the same navigation across the top and some at the bottom. Their top nav includes their logo and links to important areas of their site. If the Yahoo site were coded in FreeEnergy, there would probably be a dynamic navigation module for generating the path to the current section, such as Home > Computers and Internet > Software > Internet > World Wide Web > Servers > Server Side Scripting > PHP.

Screen modules contain the content unique to the particular page being displayed. They may be plain HTML, or they may be primarily PHP code, depending on context. A press release is static. It can be prepared by someone unfamiliar with PHP. They only need to know that the screen module is an HTML fragment.

Any module may rely on a utility module in much the same way utility files are used in other contexts. Some utility modules are called each page load. Others are collections of functions or objects related to a particular database table.

All modules are collected in a modules directory that further contains a subdirectory for each module type. To enhance security, it is placed outside of the Web server's document root. Within the document root is a single PHP script, index.php. This script begins the process of calling successive modules and structuring their output with the standard HTML tags.

9.5 FASTTEMPLATE

Fast Template is is a PHP port of the Perl CGI:FastTemplate template engine originally written by Jason Moore jmoore@sober.com.

In 1999 the development of Fast Template was abandoned by CDI but survived as a good template system over the time. Since then several people tried to add new capabilities making the code work with newer PHP versions.

Since Fast Template is still useful in PHP projects and we decided to release a version with all modifications and new capabalities that were added.

Currently it includes the merged cache function, debug console, silent removal of not-assigned dynamic blocks, ability of including PHP code into templates.

We also added a new functionality: pattern assign. When variables or constants are the same as the template keys, these functions may be used as they are. Using these functions, can help you reduce the number of the assign functions in the PHP files, which is very useful for language files.

When are templates useful?

Templates are very useful for CGI programming, because adding HTML to your PHP code clutters your code and forces you to do any HTML modifications. By putting all of your HTML in seperate template files, you can let a graphic or interface designer change the look of your application without having to bug you, or let them muck around in your PHP code.

Why use FastTemplate?

Speed

FastTemplate parses with a single regular expression. It just does simple variable interpolation (i.e. there is no logic that you can add to templates - you keep the logic in the code). That's why it's has 'Fast' in it's name!

Flexibility

The API is robust and flexible, and allows you to build very complex HTML documents/ interfaces. It is also completely written in PHP and (should) work on Unix or NT. Also, it is not restricted to building HTML documents -- it could be used to build any ascii based document (postscript, XML, email - anything).

What are the steps to use FastTemplate?

The main steps are:

- define
- assign
- parse
- FastPrint

Variables

A variable is defined as:
<kbd>{([A-Z0-9_]+)}</kbd>

This means, that a variable must begin with a curly brace '{'. The second and remaining characters must be uppercase letters or digits 'A-Z0-9'. Remaining characters can include an underscore. The variable is terminated by a closing curly brace '}'.

For example, the following are valid variables:
{FOO}
{F123F}
{TOP_ OF_PAGE}

One interesting thing we have learned in years in the Web development business is that no two shops build Web applications the same way. Some have a mix of people who may be talented in certain areas, but all can do the same type of work. Others

enforce a strict separation between those who do HTML and those who write scripts. Others draw the line between a graphic design group and an engineering group that does HTML and scripting. It can become inefficient for an HTML group to be requesting changes from a scripting group.

FastTemplate addresses this problem by separating HTML from PHP. Templates are written in HTML and may contain special codes surrounded by curly braces. A PHP script loads the templates, defines values for the special codes and replaces them. The codes may be replaced with other templates or with data created by PHP. The implication is that people unable to work with PHP code will be comfortable working with template files that better resemble plain HTML. Small changes to the HTML can be made without interaction with the engineers, who might be grumpy about making changes. In addition, the engineers will not have to worry about novices introducing errors into their scripts.

9.6 MIDGARD

Another approach to Web site design with PHP is the Midgard project. The maintainers are Jukka Zitting and Henri Bergius. Rather than code a solution in PHP alone, they have pursued integrating PHP into their own application server. Midgard is capable of organizing more than 800,000 pages of content using a Web-based interface. For this reason it is ideal for operating Magazine sites.

Keyword

Content management system is software that helps users create, manage, and modify content on a website without the need for specialized technical knowledge.

Midgard is an open source persistent storage framework. It provides an object-oriented and replicated environment for building data-intensive applications.

Midgard also ships with MidCOM **content management system (CMS)** built on the Midgard framework. MidCOM's features include web-based authoring WYSIWYG interfaces and a component interface for installing additional web functionalities, including wikis and blogs.

Midgard is an open source project, of course. You can download an official release, or grab a snapshot through CVS. In order to install it, you must modify PHP slightly, but instructions are available at the Midgard site. Because it requires compilation, running Midgard on Windows is probably not worth the effort.

Ariadne

Still in beta at time of writing, Ariadne is a Web application framework from Muze, a development agency in the Netherlands. It is available under the GNU Public License. Auke van Slooten leads the project.

Ariadne stores PHP source code as objects in a MySQL database. These objects interact with each other using a virtual file system. A rich user interface is presented to the user through Web pages, but advanced users may dig deeper, as well. Another major component controls access rights for users or groups.

Preserving State and Providing Security

You may wish to secure your Web application by requiring visitors to identify themselves with a login and password. Requiring this page after page, though, would be very annoying. You may even want to track users through the site without actually identifying them. The process should be invisible and should not intrude on the experience.

One solution is to generate a random session identifier. This identifier must not be easy to guess and must be unique to each user. The session could be stored in a database or a file and passed in every link or form. The site simply checks that the session is valid each time a page is requested. If the session is invalid, you may display an error message, send the user back to the login page, or just generate a new session identifier, depending on context.

In a site that requires users to log in, the **session identifier** will be associated with a user identifier, which would be the key to a table of user information. You may also keep track of the last time the session requested a page and have all those with no activity in a given period, perhaps 15 minutes, expire. This protects users who walk away from their computers without explicitly logging out.

You may also choose to associate arbitrary variables with each session. This is relatively easy to implement with a relational database. Create a table where each row is uniquely identified by session identifier and variable name. Creating a variable is as easy as inserting a row into the table. You can

Keyword

Session identifier is a piece of data that is used in network communications to identify a session, a series of related message exchanges

fetch each variable with each request, or fetch them only as needed. Another approach would be serializing an array of values and storing it in a single table column.

Cloaking

When creating a plain HTML site, you confront two paths: create a site that works great in only one browser, or create a mediocre site that works in all browsers. PHP allows you to create a site that works great in any browser. The HTTP_USER_AGENT variable contains the string most browsers send to the Web server to identify themselves. This variable may be used to choose between versions of content. This cloaks the inner workings of the site from the browser. A seamless experience is provided to visitors, despite differences in browser capabilities.

In most cases browser name and version are sufficient, though operating system is also helpful. There are subtle differences between identical versions of browsers running on Windows or the Macintosh. One design element we have cloaked in the past is a JavaScript rollover, a graphic button that changes when the mouse is passed over it.

The label on the button may glow. This is accomplished in JavaScript by replacing the image. Unfortunately, this is not possible in older browsers. The code to accomplish this may be included only for browsers capable of executing it.

Another cloak we have used in the past involved graphic hard rules. An HTML trick is to create a single-pixel image and stretch it by setting the height and width attributes to values larger than one. This effect can be used to stretch the pixel into a line that becomes a hard rule. And unlike the HR tag, the rule can be any color. For older browsers that do not allow stretching of images, we send an HR tag instead to approximate the effect. Alternatively, we could have pointed to a graphic of the appropriate size.

Consider combining the strategy of cloaking with FreeEnergy. You can choose different layout modules for different browsers. The text-only Lynx browser does not allow

you to arrange elements using HTML tables, as is customary, and may jumble your content. Because the content is separated from the layout code, you could create a Lynx-friendly version of an entire site by creating a single layout module.

9.7 URLS FRIENDLY TO SEARCH ENGINES

Search engines such as Google and Alta Vista attempt to explore the entire Web. They have become an essential resource for Internet users, and anyone who maintains a public site benefits from being listed. Search engines use robots, or spiders, to explore pages in a Web site, and they index PHP scripts the same way they index HTML files. When links appear in a page, they are followed. Consequently, the entire site becomes searchable.

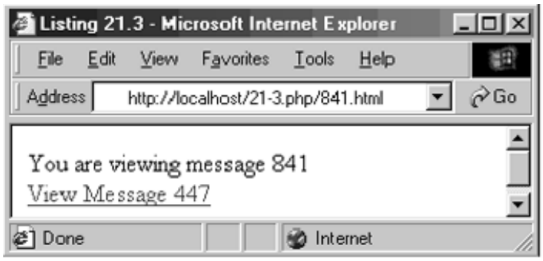
Unfortunately, robots will not follow links that appear to contain form variables. Links containing question marks may lead a robot into an endless loop, so they are programmed to avoid them. This presents a problem for sites that use form variables in links. Passing form variables in anchor tags is a natural way for PHP to communicate, but it can keep your pages out of the search engines. To overcome this problem, data must be passed in a format that resembles URLs.

First, consider how a Web server accepts a URI and matches it to a file. The URI is a virtual path, the part of the URL that comes after the host name. It begins with a slash and may be followed by a directory, another slash, and so forth. One by one, the Web server matches directories in the URI to directories in the filesystem. A script is executed when it matches part of the URI, even when more path information follows. Ordinarily this extra path information is thrown away, but you can capture it.

Look at Listing 3. This script works with Apache compiled for UNIX but may not work with other Web servers. It relies on the PATH_INFO environment variable, which may not be present in a different context. Each Web server creates a unique set of environment variables, although there is overlap.

You may be accessing the code in Listing 3 from the URL http://localhost/corephp/figures/21-5.php/1234.html. In this case, you are connecting to a local server that contains a directory named corephp/figures in its document root. A default installation of Apache might place this in /usr/local/apcache/htdocs. The name of the script is 21-5.php, and everything after the script name is then placed in the PATH_INFO variable. No file named 1234.html exists, but to the Web browser it appears to be an ordinary HTML document. It appears that way to a spider as well.

Listing 3 Using Path Info



We have introduced only the essential principles of this method. There are a few pitfalls, and a few enhancements to be pursued. Keep in mind that **Web browsers** do their best to fill in relative URLs, and using path information this way may foil their attempts to request images that appear in your scripts. Therefore, you must use absolute paths. You might also wish to name your PHP script so that it does not contain an extension. This is possible with Apache by setting the default document type, using the DefaultType configuration directive.

Keyword

Web browser is application software for accessing the World Wide Web. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the user's device.

9.8 RUNNING A SCRIPT REGULARLY

Both UNIX and Windows NT have facilities for running programs according to a schedule. In UNIX you can edit your crontab file, and in Windows you use the scheduling service. These are useful when you wish to perform some maintenance function as part of your PHP-powered site. You may write a script to download the list of files at the Slashdot site. Another script might rebuild the index for a local search engine. Both crontab and the scheduling service take a command line and execute it at a given time. If you're not familiar with the details, either read the main page for crontab, or type at /? in a Windows command shell.

You have two choices for invoking a PHP script from the command line. If you have PHP as a stand-alone executable, you can call it and use the path to the script for the only argument. This is probably the way to go on a Windows machine, because the installation provides php.exe. A UNIX installation likely will be compiled as an Apache module, and no stand-alone executable will be available. In this case, you

can use another program to make an HTTP connection. The text-only browser, Lynx is well suited for this purpose.

Remember, the Web server executes PHP scripts. Executing scripts from the root user's crontab will allow them greater ability to do damage. It is probably best to execute the script from the Web server's crontab. Using Lynx to run the script avoids this issue but raises another. Unless you put the script in a protected directory, anyone will be able to run it. Simply protect the script with a username and password. Lynx will allow you to specify these on the command line.

9.9 EFFICIENCY AND DEBUGGING

Efficiency should not be your first concern when writing code. You must first write code that works, and hopefully your second concern is keeping the code maintainable.

You will pick up some tactical design issues as you gain more experience in programming. These begin to gel as idioms—repeated structures applied to similar problems. Individuals and organizations tend to develop their own idioms, and you will notice them in code found in magazine articles and code repositories. Once you accept an idiom as your own, you can consider it a solved problem.

In most projects, a tiny minority of code is responsible for most of the execution time. Consequently, it pays to measure first and optimize the slowest section. If performance increases to acceptable levels, stop optimizing.

When a bug appears in your script, the time you spent writing meaningful comments and indenting will pay off. Sometimes just reading over troublesome code will reveal its flaws. Most of the time you will print incremental values of variables to understand the problem.

9.9.1 Measuring Performance

Three factors affect the time it takes to go from clicking on a link to seeing a completed Web page. First is the network. Your request must travel from the browser to the server, and the Web page must travel back to your browser.

The best way to measure how long a script runs is to print the time in important points in your script. Because most scripts take less than a second to run, you must use the microtime function. If you place the output in HTML comments, the display of the page will not be disturbed. Of course, the print statement itself will take some time. You can minimize this by simply printing the output of microtime instead of trying to convert its output into an integer. You can do the math later by hand or in a spreadsheet. The first HTML comment contains the time on the clock when the script begins. That's followed by time when the 10,000 cosine calculations have finished. Finally we see the time when the 10,000 lines are written to a file.

Output of microtime

```
<!-- 0.95462500 950996931 -->

<!-- 0.38373500 950996932 -->

<!-- 0.46406700 950996932 -->
```

The microtime function returns two numbers. The first is a fraction of a second, the other the number of seconds since January 1, 1970. Notice that from the first comment to the next, the number of seconds changed from 950996931 to 950996932. The fraction changed from 0.95462500 to 0.38373500. In total 0.42911 seconds elapsed. Doing the math for the second part shows it took 0.080332 seconds. If the performance of this script were not satisfactory, we'd first look into improving the first half. It takes five times longer to execute than the rest.

Measuring Script Performance

```
<?
print("\n<!-- " . microtime() . " -->\n");

  //fake some long calculation
  for($index = 0; $index  10000; $index++)
  {

    $value += (cos(time()%360));
  }

  print("\n<!-- " . microtime() . " -->\n");

  //write to file
  $fp = fopen("data.txt", "w");
  for($index = 0; $index < 10000; $index++)
  {
     fputs($fp, "Testing performance\n");
   }
   fclose($fp);

   print("\n<!-- " . microtime() . " -->\n");
?>
```

9.9.2 Fetching Database Query Results

For most of the databases supported by PHP, you can get columns in two ways. You can specify a value by row number and column name, or you can fetch rows one

at a time in an array. For MySQL this involves mysql_result and mysql_fetch_row, respectively.

Using mysql_result is much slower than the fetch functions. PHP has to work harder to find the exact piece of data you need. First, the specified row must be referenced. Then the data in that row must be searched for a column with a matching name. You can imagine that executing mysql_result several times inside a loop can add up to a very slow script. Each call has to start at the beginning and find the appropriate data element.

Alternatively, you may fetch an entire row into an object, as we have done in most of the examples so far. This allows you to reference the exact element without searching through the entire data set. The challenge is to match up the results of the query with the array elements. If you have created a query such as

```
SELECT *
FROM user u, employer e
WHERE u.Employer = e.ID
```

you may have a hard time. You will have to examine the structure of each table to see the order of the columns. A better approach is to specify the columns you need, leaving out any you will not use. This would transform the query into something like

```
SELECT u.ID, u.Name, e.Name

FROM user u, employer e
WHERE u.Employer = e.ID
```

which specifies only three columns. You can be sure, regardless of the order of the columns in the table, that the user ID will be column zero.

Another advantage is that, since the resulting data have been narrowed to three columns, each fetch will be much smaller. The savings go all the way back to the database, because it will return only three pieces of data times the number of rows. None of the unused rows from the first version of the query will be sent through the network from the database server to PHP. In turn, PHP does not need to put them into the array.

9.9.3 When to Store Content in a Database

When we speak of content, we mean static text, perhaps containing HTML. There is no rule saying that content should never be placed in a database, or that it should always be put in a database. In the case of a bulletin board, it makes sense to put each message in a database. Messages are likely to be added continually. It is convenient to treat them as units, manipulating them by their creation dates or authors. At the

other extreme, a copyright message that appears at the bottom of every page of a site is more suited to a text file that is retrieved with the require function.

Somewhere between these two extremes is a break-even point. The reason is that databases provide a tradeoff. They allow you to handle data incomplex ways. They allow you to associate several pieces of information around a common identifier. However, you trade away some performance, as retrieving data is slower than if you opened a file and read the contents.

Many Web sites are nothing more than a handful of pages dressed up in a common graphic theme. A hundred files in a directory are easy to manage. You can name each one to describe their contents and refer to them in a URL such as http://www.mysite.com/index.php?screen= about_us and still get the benefit of systematically generating the layout and navigation. Your PHP script can use the value of the screen variable as the name of a local file, perhaps in a directory named screens. Developers can work on the page contents as they are accustomed, because they know the code is stored in a file named about_us in a directory named screens.

When the content grows to a thousand pages, keeping each in a separate file starts to become unmanageable. A relational database will help you better organize the content. With a site so large it's likely that there will be many versions of the navigation. In a database it is easy to build a table that associates page content with navigation. You can also automate hyperlinks by creating one-way associations between pages. This would cause a link to automatically appear on a page.

The biggest problem with this approach is the lack of good tools for editing the site. Developers are used to fetching files into an editor via FTP. Asking these same people to start using a database shell is most likely out of the question. The cost of teaching SQL to anyone who might work on the site may eliminate any benefit gained when the content is put into the database. So, you are faced with creating your own tools to edit the content. The logical path is to create Web-based tools, since coding a desktop application is a major project in itself, especially if both Windows and Macintosh users are to be accommodated. As you might guess, Web-based site editors are less than ideal. However, with very large sites they become bearable, because the alternative of managing such a large static site is a greater evil, so to speak.

9.9.4 In-Line Debugging

There are times when code produces unexpected results. Examining the code reveals nothing. In this case the best thing to do is some in-line debugging. PHP scripts generate HTML to be interpreted by a browser, and HTML has a comment tag. Therefore, it is a simple matter to write PHP code that reports diagnostic information inside HTML comments

Often we create database queries dynamically, based on user input. A stray character or invalid user input can cause the query to return an error. Sometimes we will print the query itself. We also print the results of the error functions, such as mysql_error. The same applies to code unrelated to databases. Printing diagnostic information, even if it is as simple as saying "got here," can help.

9.9.5 Remote Debugging

You may enable remote debugging for all scripts by editing the configuration file php. ini, or you may use the debugger_on function. Once enabled, PHP will attempt to connect to a remote host and port each time a script is run.

You will need a port listening program to get the debugging information. There are numerous free port listeners for Windows. We have found Port Listener by Hauke X to work well. Similar programs exist for UNIX as well.

All debugging messages are sent to the listening host, regardless of the error reporting level set in php.ini. Messages are sent in a special format followed by a linefeed.

date time host(pid) type : message-data

The date is in the format YYYY-MM-DD. Time is in the format HH:MM:UUUUUU. The last six digits of time are the seconds and microseconds. The host is the name of the server and pid is the process identifier. The type is a special code described in Table 3. The rest of the line is a message terminated with a linefeed.

Using the remote debugger is as simple as executing a PHP script and watching the debug information appear in the port listener output.

Table 3. Debugging types

Type	Description
end	The end of an error message
frames	The number of frames in the call stack.
function	The name of the function where the error occurred.
location	The filename and line number that generated the error message.
message	A PHP error message.
start	This type signifies the beginning of a debugging message. The data-message for this line will be the type of error.

9.9.6 Simulating HTTP Connections

When writing PHP scripts, it is not necessary to understand every detail of the HTTP protocol. We would be straying to include a treatise here, but you ought to have enough understanding so that you could simulate a connect by using telnet. You may know that Web servers listen on port 80 by default. HTTP is a text-based protocol, so it's not hard to telnet directly to a Web server and type a simple request.

CASE STUDY

A CASE STUDY OF THE APPLICATION OF PHP DESIGN PATTERN'S STRATEGY PATTERN

This case study illustrates the application of strategy mode in PHP design pattern.

Strategic model

Definition

Policy patterns define a series of algorithms that encapsulate each algorithm and allow them to swap with each other. The policy pattern allows the algorithm to change independently of the users who use it.

Role analysis

- Abstract policy roles: policy classes, usually implemented by an interface or abstract class;
- Specific strategic roles: wrapping related algorithms and behaviors;
- Environmental roles: Hold a reference to a policy class that is ultimately used by the client.

Application scenarios

Multiple classes only differ from each other in table behavior. Policy patterns can be used to dynamically select specific actions to be performed at run time.

Different strategies (algorithms) need to be used in different situations, or strategies may be implemented in other ways in the future.

Hide the implementation details of the strategy (algorithm) to the customer, completely independent of each other.

Code implementation

```
<?php
/**
 * Created by PhpStorm.
 * Author: zhaorui
 * Date: 2019/2/27
```

```
 * Time: 10:55
 */
header('Content-Type:text/html;charset=utf-8');
// Abstract policy interface
abstract class Strategy{
  abstract function wayToSchool();
}
// Specific strategic roles
class BikeStrategy extends Strategy{
  function wayToSchool()
  {
    Echo "Go to school by bike". PHP_EOL;
  }
}
class BusStrategy extends Strategy{
  function wayToSchool()
  {
    Echo "go to school by bus". PHP_EOL;
  }
}
class TaxiStrategy extends Strategy{
  function wayToSchool()
  {
    Echo "Take a taxi to school." PHP_EOL;
  }
}
// Environmental role
class Context{
  private $strategy;
  function getStrategy($strategyName){
   try{
    $strategyReflection = new ReflectionClass($strategyName);
    $this->strategy = $strategyReflection->newInstance();
```

```
    }catch (ReflectionException $e){
      $this->strategy = "";
    }
  }
  function goToSchool(){
    $this->strategy->wayToSchool();
  }
}
// Testing
$context  = new Context();
$context->getStrategy('BusStrategy');
$context->goToSchool();
```

Operation results

Take a bus to school

Advantage

- The policy pattern provides a way to manage the related algorithmic families. The hierarchical structure of policy classes defines an algorithm or behavior family. Proper use of inheritance can transfer common code to the parent class, thus avoiding duplication of code.
- Policy patterns provide a way to replace inheritance relationships. Inheritance can handle a variety of algorithms or behaviors. If not using policy patterns, then there may be some subclasses of environment classes that use algorithms or behaviors, each of which provides different algorithms or behaviors. However, the user of the algorithm or behavior is mixed with the algorithm or the behavior itself. The logic of deciding which algorithm to use or which behavior to take is mixed with the logic of the algorithm or behavior, so that it is impossible to evolve independently. Inheritance makes it impossible to dynamically change an algorithm or behavior.
- Multiple conditional transfer statements can be avoided by using policy patterns. Multiple transfer statements are not easy to maintain. It combines the logic of which algorithm to adopt or which behavior to adopt with the logic of algorithm or behavior, and lists them all in a multiple transfer statement, which is more primitive and backward than the method of inheritance.

Disadvantages

The client must know all the policy classes and decide which one to use. This means that the client must understand the differences between these algorithms in order to select the appropriate algorithm class in time. In other words, the policy pattern only applies to all the algorithms or behaviors that the client knows.

Policy patterns create many policy classes, and each specific policy class generates a new class. Sometimes, policy classes can be designed to be shared by storing environment-dependent states in the client, so that policy class instances can be used by different clients. In other words, you can use the hedgehog pattern to reduce the number of objects.

SUMMARY

- Debugging is the process of finding and resolving bugs (defects or problems that prevent correct operation) within computer programs, software, or systems.
- Before you can design a system, it is important to understand what it's supposed to do.
- The operating and development environments are sometimes overlooked in requirements specifications. This includes both the browser and the Web server.
- The exception-handling section describes how the system deals with adversity. The two parts of this are disaster and invalid input.
- The largest part of a design document is a detailed description of how each module works. At this point it's acceptable to specify exactly the method for implementing the module.
- CVS, Concurrent Versions System, is an open-source system that allows developers to track all changes to a project. A central repository stores the files that make up the project.
- FreeEnergy is a system that attempts to encapsulate major pieces of each page into files to be included on demand. Before we proceed, we want to state motivations clearly.
- The FreeEnergy system uses more calls to include than you'd find where you are simply making a few includes at the top of your pages.
- Templates are very useful for CGI programming, because adding HTML to your PHP code clutters your code and forces you to do any HTML modifications.
- When creating a plain HTML site, you confront two paths: create a site that works great in only one browser, or create a mediocre site that works in all browsers.
- Web-based tools, since coding a desktop application is a major project in itself, especially if both Windows and Macintosh users are to be accommodated.

KNOWLEDGE CHECK

1. **What is gd in PHP?**
   a. A library
   b. A class
   c. A data type
   d. Nothing
2. **PHP-GTK developers are denoted by**
   a. php.gk
   b. php.gtk.dev
   c. php.dev
   d. php.pear
3. **For creating an image in PHP, which gd function is used?**
   a. create ()
   b. picture_create ()
   c. imagecreate ()
   d. createfromgd ()
4. **Which from the following bug occurs during program execution in PHP?**
   a. Runtime bugs
   b. Compile time bugs
   c. Logical bugs
   d. All of the above
5. **500 series codes indicate a**
   a. Client side error
   b. Server side error
   c. Redirection
   d. Success

REVIEW QUESTIONS

1. Discuss the properties of requirements specifications.
2. What do you understand by concurrent versions system? Explain.
3. What is Fast Template? When are templates useful?
4. Explain how fetch database query results.
5. When to store content in a database? Discuss.

Check Your Result

1. (a) 2. (b) 3. (c) 4. (d) 5. (b)

REFERENCES

1. Ahmadzadeh, M., D. Elliman and C. Higgins, 2005. An analysis of patterns of debugging among novice computer science students. Proceedings of the 10th Annual SIGCSE Conference on INNOVATION and Technology in Computer Science Education, (ITiCSE' 05), ACM Press, New York, pp: 84-88. DOI: 10.1145/1151954.1067472
2. Caballero, R., C. Hermans and H. Kuchen, 2007. Algorithmic debugging of java programs. Elect. Notes Theoretical Comput. Sci., 177: 75-89. DOI: 10.1016/j.entcs.2007.01.005.
3. Chang, C.P., C.P. Chu and Y.F. Yeh, 2008. Integrating in-process software defect prediction with association mining to discover defect pattern. Inform. Software Technol., 51: 375-384. DOI: 10.1016/j.infsof.2008.04.008.
4. Cheda, D. and J. Silva, 2009. State of the practice in algorithmic debugging. J. Elect. Notes Theoretical Comput. Sci., 246: 55-70. DOI: 10.1016/j.entcs.2009.07.015.
5. Chmiel, R., M.C. Loui, 2004. Debugging: From novice to expert. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Mar. 03-07, ACM Press, USA., pp: 17- 21. DOI: 10.1145/971300.971310
6. Coull, N., 2003. Helping novice programmers interpret compiler error messages. Proceedings of the 4th Annual LTSN-ICS Conference, (ALIC' 03), pp: 26-28.
7. Csallner, C. and Y. Smaragdakis, 2005. Check 'n' crash: Combining static checking and testing. Proceedings of the 27th International Conference on Software Engineering, May 15-21, ACM Press, St. Louis, MO, USA., pp: 422-431. DOI: 10.1145/1062455.1062533
8. Ebrahimi, A. and C. Schweikert, 2006. Empirical study of novice programming with plans and objects. SIGCSE Bull., 38: 52-54. DOI: 10.1145/1189215.1189169
9. Etheredge, J., 2004. CMeRun: Program logic debugging courseware for CS1/CS2 students. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Mar. 03-07, Norfolk, USA., pp: 22-25. DOI: 10.1145/1028174.971311
10. Garner, S., P. Haden and A. Robins, 2005. My program is correct but it doesn't run: A preliminary investigation of novice programmers' problems. Proceedings of the 7th Australasian Conference on Computing Education, (ACE' 05), Australian Computer Society, Inc. Darlinghurst, Australia, Australia, pp: 173-180.

Index

Level: Beginner to Advanced
Subject: Computer and Information Science

Basic Computer Coding: PHP

2nd Edition

PHP is an open source, server-side, HTML embedded scripting language used to create dynamic web pages. However, there are many languages which are used for web development or web programming. But among all of them PHP is the most popular web scripting language. So, let us find out why PHP is widely used for web development. PHP language has its roots in C and C++. PHP syntax is most similar to C and C++ language syntax. So, programmers find it easy to learn and manipulate.

The 2nd edition of this book is comprised of nine chapters. The book includes basic through advanced functionality, this book explains how to develop dynamic Web applications, such as guest books, chat rooms, and shopping carts.

ISBN 978-1-98467-609-2

BIBLIOTEX
Digital Library

www.3ge-learning.com
email: info@3ge-learning.com