Unit 1

TCC 242/05
Web Database Application

Introduction to Web Development and PHP
Wawasan Open University is Malaysia’s first private not-for-profit tertiary institution dedicated to adult learners. It is funded by the Wawasan Education Foundation, a tax-exempt entity established by the Malaysian People’s Movement Party (Gerakan) and supported by the Yeap Chor Ee Charitable and Endowment Trusts, other charities, corporations, members of the public and occasional grants from the Government of Malaysia.

The course material development of the university is funded by Yeap Chor Ee Charitable and Endowment Trusts.

© 2011 Wawasan Open University. Except where otherwise noted, this work is licensed under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

This course material was published to support the learning of students registered with Wawasan Open University. Wawasan Open University does not grant any degree, certification or credits based solely on your completion of this course material.

Wawasan Open University
Jalan Sultan Ahmad Shah,
10050 Penang.
Tel: (604) 2180333  Fax: (604) 2279214
Email: enquiry@wou.edu.my
Website: www.wou.edu.my
Contents

Unit 1  Introduction to Web Development and PHP

Course overview 1
Unit overview 3
Unit objectives 3

1.1 Understanding web development 5

Objectives 5
Introduction 5
Web development — A brief description 5
Methodologies for web development 8
Databases on the Web 16
Suggested answers to activities 20

1.2 Installing and configuring Apache, MySQL and PHP 23

Objectives 23
Introduction 23
Installing LAMP 23
Testing the configurations of Apache, PHP and MySQL 26
Suggested answers to activities 35
Course Overview

Welcome to *TCC 242/05 Web Database Application*.

This is a course that emphasises both theory and application. It introduces the concepts, methods and software that will help you create web database applications. A web database application is a database-driven web application that enables websites such as online shopping, Internet banking or social networking sites to be created. In this course, you will learn how to build web database applications that incorporate authentication and security, and that adhere to industry standards with respect to database-driven websites.

This course module has five study units. Each unit takes three to four weeks to complete. The course is structured so that each unit builds on previous ones.

In Unit 1, you will be introduced to the fundamentals of web application development, web servers and the syntax of the PHP scripting language. PHP is a powerful server-side scripting language that allows you to build dynamic websites. You will also be shown how to install Apache web server, PHP and MySQL on your computer. Unit 2 will focus on the basics of PHP. Here, you will learn PHP data types, customised functions and control structures. Unit 3 will discuss the intermediate-level features of PHP such as strings, arrays and files. The discussion of database integration begins in Unit 4. You will learn how PHP interacts with MySQL and MySQL library functions. The final unit of this course will cover the principles of using sessions and cookies for saving state information on a web page.

The material in this module is presented in a simple, step-by-step format, and it includes many examples and web programming exercises. Each unit has specific objectives, activities and self-tests. The activities and self-tests allow you to check your progress and improve your understanding of what you have learnt. The practice exercises at the end of each unit allow you to practise the skills that you have learned so that you can develop proficiency in real-world problem solving.

By the end of this course, you should be able to:

1. Install and configure Apache web server, PHP and MySQL.
2. Create a web database application incorporating functions and control structures using PHP.
3. Manipulate strings using PHP.
5. Work with database records and manage user accounts.

6. Use PHP to manipulate database records.

7. Use hidden forms, query strings, cookies and sessions to save state information.
Unit Overview

This unit introduces you to web development methodologies. It introduces the Apache web server, PHP and MySQL, and discusses the rationale of using databases on the web. You will also learn how to install and configure the Apache web server, PHP and MySQL on both Linux and Windows systems. In addition, you will create web pages that use basic PHP scripts.

Unit Objectives

By the end of this unit, you should be able to:

1. Discuss web development methodologies.

2. Install, configure and run the Apache web server, PHP and MySQL on your computer.

3. Create web pages using PHP scripts.
1.1 Understanding Web Development

Objectives

By the end of this section, you should be able to:

1. Describe web development methodologies.

2. Describe how information is exchanged between servers and clients.

3. Explain why databases are used on the World Wide Web.

Introduction

A web database application is a database-driven web application. This type of application is used in Internet banking websites such as Maybank2U.com, online shopping websites such as Amazon.com and social networking sites like Facebook. For sites like Amazon.com, even a few minutes of downtime could cost millions of dollars in lost revenue. Many dotcom failures in the early nineties were caused by the improper management and underestimation of the scope and features of web development. In this section, we are going to look at the architecture of a web application, starting with the methodologies of web development. We will also show how databases are used on the Web.

Web development — A brief description

A website is a place for publishing information using web pages. A web page is a collection of text, multimedia and hyperlinks that allow users to navigate from one page to another. Web development is defined as the conception, design and implementation of a web entity created to serve a specific purpose. The purpose could be to sell something, promote products or publish information.

Websites can be categorised as personal or professional. Personal sites are mainly for people to share information about themselves or their products. They may also just want to get attention, promote their skills or post their curriculum vitae online to improve their chances of getting a job. Recently, people have started to publish virtual journals called blogs that describe events in their lives, their interests or their personal opinions on various topics.

Professional sites are created as portals that showcase businesses and provide information on their products and services. Some of these portals enable businesses to sell products online and act as virtual storefronts for receiving orders and payment. Such sites can directly generate revenue for businesses.
Websites can also be categorised as static or dynamic. **Static websites** are developed with fixed-content pages (pages with content that does not change dynamically) that do not contain any programmatically controlled content. The content will only change when the website designer manually modifies it.

As the pace of change and business information flows have become more rapid, static websites are no longer effective at showcasing fast-moving businesses. **Dynamic websites** are more effective tools for such businesses. Dynamic websites are driven by dynamic content that is loaded or created using programming and databases.

The following are some of the popular platforms that are used to create dynamic websites.

**LAMP (Linux, Apache, MySQL and PHP)**

This is a popular platform for creating dynamic websites because it is **open source**, free of charge and easily adaptable. This course will go into the details of **MySQL** and **PHP**, while also briefly describing Linux and Apache. **Linux** is an open-source operating system, **Apache** is an open-source web server, **MySQL** is an open-source database and **PHP** is an open-source scripting language.

**Microsoft .NET**

The .NET framework is a platform that was created by Microsoft. It has a huge library that supports several programming languages like Visual Basic and C#. It also provides many ready-made tools, templates and applications that assist developers in creating software. However, to use these tools and templates, developers have to purchase the appropriate licenses from Microsoft.

**Java Struts and JSP**

Java Struts and Java Server Pages (JSP) are technologies for developing dynamic websites with the Java programming language. Java provides a standardised Integrated Development Environment (IDE), but it is not as intuitive as the Microsoft Visual Studio IDE.

**What platform should I use?**

Since LAMP is free and open-source, you should consider using it for web development projects in which cost is the main constraint. If the software you are dealing with is written in Java, JSP or Struts could be better candidates. If you are looking for ease of starting, ready-made tools and templates, technical support and more robust applications, you should consider .NET.
When choosing a platform, you should also consider the following important aspects of web development:

1. Security
2. Scalability
3. Search engine optimisation
4. Portability
5. Maintenance
6. Administration

---

**Web Reference**

You can read more about the Open-Source Software movement at:


---

**Activity 1.1**

1. Web development is defined as

   A. a concept that describes the growth of an online business.
   B. the conception, design and implementation of a web entity to serve a purpose.
   C. a reference guide for writing a web programming language.
   D. the philosophy of how to create and run an online business.

2. Which of the following is not a platform for creating dynamic websites?

   A. JSP
   B. LAMP
   C. MS.NET
   D. Internet Explorer
3. Personal sites are mainly for people to share information about themselves or their products. People nowadays publish virtual journals called ____________ that describe events in their lives, their interests or their opinions.

4. ____________ websites have pages with fixed content, which is not programmatically controlled.

5. LAMP is one of the most popular platforms for creating websites because it is open source, free of charge and easily adaptable. True / False?

6. What are the important aspects of web development that you should consider when choosing a platform?

---

**Methodologies for web development**

Some web development projects involve minor modifications to existing websites, while others may involve major development work. To make sure that projects are delivered on time and within budget, we need to look into several development methodologies.

**Waterfall model (also known as the traditional model)**

In this model, systems development is divided into five well-defined sequential stages. The stages are *(Figure 1.1)*:

1. Analysis: Identifying the problem and formulating how to translate it into the software domain as a specification.
2. Design: Designing the system.
3. Implementation: Implementing the system for end users in a real-world scenario.
4. Testing: Testing the system for functionality and bugs, against the initial specification.
5. Delivery/Support: Handing over the finished product to the customer and providing software, hardware and usability support.

These stages must be performed sequentially. Hence, the final product will be delivered when all the stages have been completed.
The advantages of using the waterfall model in systems development are:

1. It enables you to plan and organise the tasks needed to achieve the main objective.

Your objective in systems development is to develop a usable system and deliver it to the user on time. You need to organise your team members’ schedules in order to meet the deadline. The waterfall model enables you to see where you are heading and it clearly differentiates between the various stages of the development process. Thus, by using this model as your fundamental framework, you can plan and organise the workload for your project.

2. Short development times.

Using the waterfall model will enable short development times as there is no iterative process involved. Therefore, it enables you to reach your goal in a shorter time. However, the quality of your system will be affected if you do not properly complete the tasks in one stage before moving on to the next.

3. Low cost.

Due to the direct approach to the objective taken by this model, we can shorten development time. Hence, manpower, documentation and utility costs can be reduced.
The disadvantages of the waterfall model are:

1. Not flexible to changes.

   The model has a rigid sequence. Hence, you cannot start a new stage until you have completed the previous stage. In addition, it does not let you backtrack when you discover problems. As such, errors will accumulate from stage to stage.

2. System limitations are not discovered until later stages in the development cycle.

   This model does not allow you to backtrack; hence when you discover an error in a stage, the error will be carried forward to the subsequent stages. For example, if you make an error in the analysis stage, the error will be carried to the delivery stage. Therefore, the error will exist in the final product.

3. Unable to see the whole product until it is finished.

   We usually need to see a product before purchasing it. The waterfall model does not give us the luxury to see the product until the final stage. Thus, if the user does not like the product, it is too late for changes. If you restart the project, the cost of development will increase substantially.

For these reasons, the waterfall model is impractical in the real world. When we develop a system, it is often difficult to anticipate problems that may arise. In addition, users’ demands change from time to time but the waterfall model is not flexible enough to cope with the changes. Consequently, we need a model that is flexible and able to cope with changes. This leads us to the next model, namely the iterative development model.

**Iterative development model**

This model is similar to the waterfall model in that we still have to go through the analysis, design, implementation, testing and delivery stages. However, an iterative process has been added to the model, so that each stage can provide feedback to the stages that precede it. This iterative process enables you to refine the system or amend errors before progressing to the next stage.

As you can see in Figure 1.2, the iterative process has been added to the waterfall model to create the iterative development model. This model enables feedback to be iterated between stages until you are satisfied with the result. For example, you can iterate between the analysis and design stages until the result is acceptable. If problems are discovered at the implementation stage, you can go back to either the design stage or the analysis stage.
The advantages of using the iterative development model in systems development are:

1. Problems can be discovered and rectified at an early stage.

   You still carry out the various stages, but not in a manner as rigid as in the waterfall model. Errors discovered at an early stage can be rectified. As such, the accumulation of errors from stage to stage can be eliminated.

2. Can cope with changes.

   At any point of the process, you can decide to continue, amend or stop the project. This flexibility is important because unexpected circumstances or changes in user's requirements can arise. By providing flexibility during the development of a system, changes can be made by the developer, stakeholders and users without unduly increasing costs.

3. A better quality system.

   Due to the flexibility of iteration, you can refine each stage until you are ready to move on. In addition, when you iterate between stages, you may discover new ideas or better ways to implement the system.
The disadvantages of using the iterative development model are:

1. Limited application range.

   The iterative development model is only applicable for developing business applications and not really suitable for technical applications. Hence, its focus is narrower.

2. The process can be messy.

   Due to its flexibility, there might be so many changes that it is difficult to keep track of all of them. In addition, you can be so drawn to new ideas as you keep iterating between the stages that the main objective of the system becomes lost in the process.

3. Costs can increase.

   Excessive iteration can increase development time. When development time is increased, development costs will increase as well.

Prototype approaches

The essence of iterative design is to build, test and then refine the design. We need an approach that enables us to quickly build a system and obtain useful feedback. Prototyping is such an approach. There are three types of prototyping:

1. **Throwaway (rapid) prototyping technique**

   Throwaway prototyping is also known as rapid prototyping. A prototype built using this technique will not be used as part of the final system. This technique requires that the prototype be built quickly so that information can be promptly collected and used in the development life cycle.

   ![Figure 1.3 The throwaway (rapid) prototyping technique](image-url)
As you can see from Figure 1.3, a prototype will be built after the preliminary requirements analysis. Then you will evaluate the prototype. If the prototype is adequate, you may move to the final requirements specification. Here, adequacy means that the design knowledge gained from the prototype is good enough to derive the final requirements. If the prototype is found to be inadequate during testing, you need to rebuild and then re-evaluate the specifications/ideas until they are acceptable to the user. In the throwaway technique, the prototype will be discarded upon completion of the final requirements specification. This method enables you to see the overall features of the system before it is completed.

2. Incremental technique

The idea here is to save time between the specification and delivery stages. This requires a compromise between the developer and the clients. A large system will be broken down into components and installed separately. The developer will first set up the skeleton of the system at the client’s premises. Then, the installation of different components is carried out progressively. This incremental approach uses one overall design, even though different components are added at different times. At the end of this progressive process, the entire system will be in place. The advantage of this approach is that it enables a prototype to be tested at the location where the finished system will be used. Changes to the critical features of the system are possible as it is being built by adding components incrementally.

![Diagram](image)

Figure 1.4 Incremental technique

Note: Req = requirements specification, Arch = architectural design, Det = detailed design, Impl = implementation and testing, Int = integration and testing.
3. Evolutionary technique

In the evolutionary prototyping approach, a prototype is first developed. It will then be evaluated and continually evolved into a fully functional system. The advantage of this method is that it can cope with changes during and after development.

![Diagram](image.png)

**Figure 1.5** The evolutionary technique

### Extreme programming

Extreme programming (XP) is one of the popular Agile systems development methods. It is a model that works in a rapidly changing environment. The XP model encourages developers to work in pairs to promote synergy and to support each other. Developers are also required to write their own test scripts for the software that they have created. This model assumes that developers are good at teamwork and are highly committed to their work.

An XP team emphasises on customer satisfaction as it is expected to quickly deliver tangible results. Thus, this approach needs the heavy involvement of the customer’s representatives in development meetings. The XP model is optimised for small team projects that require less than 20 people.

### Scrum

Scrum is another Agile approach to software development. Scrum uses iterative development to address requirement changes. The model refers to these repetitions as sprints that usually last for 3 to 4 weeks. Each sprint needs to achieve some targets. For example, Sprint #1 may be required to deliver a part of the project such as the web authentication logic and mechanism.

In this approach, the entire team will spend 10 to 15 minutes in “standup” meetings every morning before they start work. Team members will update each other on tasks that were performed after the last standup meeting, tasks to be accomplished
on that day and problems that they faced. They will not resolve the problems during the standup meeting. The problems will be solved outside of the meeting so that team members who are not involved with the problem can go about their own tasks.

At the end of a sprint, the customer’s representative will be invited to a demonstration of the features that were implemented in that particular sprint. For example, the developers may demonstrate the login box, user registration page and password retrieval mechanism that were developed based on the targets set for that sprint. Scrum also works for large projects (Schwalbe 2004).

Web Reference

The following web references present a comprehensive view of various web development methodologies:

1. “Successful Web Development Methodologies”
   
   http://articles.sitepoint.com/article/successful-development

2. “Extreme Programming: A gentle introduction”
   
   http://www.extremeprogramming.org/

3. “Scrum Methodology”
   
   http://scrummethodology.com/

Activity 1.2

State whether each of the following is True or False.

1. In the methodology of web development, many developers agree with the waterfall model, which states that, in practice, it is feasible to finish a stage perfectly before moving to the next stage because customers will never change their requirements.

2. Extreme Programming is a model that works in a rapidly changing environment, encourages developers to work in pairs and requires developers to write their own test scripts for the software they have developed.
Databases on the Web

When we surf the Internet, we use a web browser such as Internet Explorer or Firefox to request information from a web server. The request could be submitted as a form, by clicking on hyperlinks or by entering URLs in our web browser. The web server will then respond to our web browser. At the basic level, the Web has a client/server architecture in which the web server is the server and the web browser is the client. Figure 1.6 shows the client-server architecture of the Web.

![Figure 1.6 Client-server architecture of the Web](image)

The web server is not a dedicated storage area. Complex data manipulation operations, especially on commercial sites, should be handled by a separate database server. Figure 1.7 shows how a web browser requests a resource from a database server and how the database and web servers respond to the request. The browser is still the client, but the web server is now the middle tier and the database server is the database tier of the architecture.

![Figure 1.7 A web browser requesting a resource from a database server](image)

Most of the websites that we visit today are powered by web database applications. Web-based email, e-commerce sites and online shopping sites are all database-driven. To build a practical and powerful website, you need to develop a web database application.

In order to build web database applications, we need to have a web server such as Apache and a database server such as MySQL. Apache is one of the popular web servers and works best in Unix/Linux-based systems. It can also be implemented in Windows-based systems. As it is open-source, developers can write code to extend the functionality of Apache. It is an extremely quick and stable web server.

MySQL is particularly suitable for web applications. You can install and use MySQL at no cost. MySQL is fast and powerful when used for building small- to medium-sized databases. As it is open source, updates and feature improvements are frequently added.
The key consideration here is to come up with an efficient and reliable way for the web server to communicate with the database. Using PHP is one of the better ways of accomplishing this task. PHP is a powerful, open-source, server-side scripting language particularly suited to web database applications because of integration tools available in the web server and database environments. PHP provides the capability to interact with almost all popular database servers.

Web Reference

The following web references present a comprehensive view of websites and web applications:

1. “Web Site vs Web Application”
http://www.tonymarston.net/php-mysql/web-site-vs-web-application.html

2. “PHP”
http://www.php.net/

3. “Apache”
http://www.apache.org/

Activity 1.3

1. What happens to the web server when we click on a hyperlink in a web page?

2. Why is MySQL suitable for web applications?

3. Apache is a popular web server that works in Unix systems but cannot run under Windows. True / False?
Summary

In this section, you have learnt the basics of web development. The various web development methodologies were briefly described and their advantages and disadvantages were discussed. You were also introduced to the application of databases on the Web. In the next section, you will be given step-by-step instructions on how to install Apache, MySQL and PHP on your computer. You will use these open-source software to build web database applications.

Self-test 1.1

1. A website is a place for publishing information using web pages. A web page may contain text, images and hyperlinks for users to navigate from one page to another. True / False?

2. Professional sites are created to describe a business, provide contact information and to generate revenues for a business. True / False?

3. As business information flows and the pace of change have become more rapid, dynamic websites have become effective tools for achieving business success. Dynamic websites have dynamic content that is loaded by ____________ or ______________.

4. Linux is an open-source ______________, Apache is an open-source ______________, MySQL is an open-source ______________, and PHP is an open-source ______________.

A. web server; operating system; database; programming language
B. programming language; operating system; database; web server
C. programming language; web server; database; operating system
D. operating system; web server; database; web programming language

5. Describe how incremental prototyping is useful in web development.
6. What are the two popular Agile models of web development?

7. An XP team spends 10 to 15 minutes in a “standup” meeting every morning before its members start work. True / False?

8. What is Scrum?

9. Complex data manipulation operations, especially on commercial sites, should be handled by a separate ______________.

   A. data server  
   B. programming language  
   C. web server  
   D. web browser

10. PHP is an open-source programming language that is particularly suited to web database applications because of integration tools available in the web server and database environments. True / False?
Suggested answers to activities

Feedback

Activity 1.1

1. B

2. D

3. blogs

4. Static

5. True

6. • Security
   • Scalability
   • Search engine optimisation
   • Portability
   • Maintenance
   • Administration

Activity 1.2

1. False

2. True

Activity 1.3

1. When we click on a hyperlink, the web browser requests information or a resource from a web server. The web server will then respond to the web browser and deliver the information or resources that were requested.
2. MySQL is open source and free, so you can install and use it without paying anything. It is fast and powerful when used for building small- to medium-sized databases. Updates and feature improvements are frequently added to MySQL.

3. False
1.2 Installing and Configuring Apache, MySQL and PHP

Objectives

By the end of this section, you should be able to:

1. Install and configure the LAMP system under Ubuntu on your computer.

2. Configure the Apache web server to work with PHP.

3. Interact with MySQL using basic SQL queries.

Introduction

This section introduces you to the general techniques for installing the Apache web server, MySQL and PHP on a Linux (Ubuntu) system. Apache is one of the most popular web servers and works best on Unix or Linux systems. However, it works well on the Windows system too. Understanding the techniques to install and configure a web server is essential to hosting your own web application. The following installation guide is for the Linux system. For the Windows system, please refer to the Appendix.

Note: The following guide uses Linux Ubuntu 10.10. You can download Ubuntu from www.ubuntu.com. It is assumed that you have basic knowledge of installing and operating Ubuntu. For more information on Ubuntu installation, please refer to the following web pages:


Installing LAMP

Linux (Ubuntu) will be the platform for the Apache, PHP and MySQL components that you will be installing and using soon. You will be installing LAMP (Linux, Apache, MySQL and PHP) packages from within Ubuntu. Once you have an understanding of how to set up LAMP on Ubuntu, you will be able set it up on other Linux versions as well.
Installation steps

1. Open the terminal window (also known as the command line or console window). On the Ubuntu desktop, go to Applications > Accessories > Terminal. You will then see a command-line window like the one shown in Figure 1.8.

![Figure 1.8 Opening a terminal window in Ubuntu](image)

2. Enter the following command in the command-line window:

   ```bash
   sudo apt-get install lamp-server`
   
   This is the general command used to install a program and package in the Ubuntu system. The command `sudo` means that you will be running the command with root (administrator) privileges. You will be prompted to enter a password. Notice the caret (^) symbol at the end of the command. It is not a typo and must be included.
3. Hit the <Enter> key after the command and you will be shown the packages that need to be installed. Type \texttt{Y} and hit <Enter> to begin the installation.

![Figure 1.9 Packages that need to be installed for LAMP](image)

4. The installation begins and you will be prompted to enter a new password for the MySQL root user (Figure 1.10). Type \texttt{wawasan} as the password for our purposes. Hit <Enter> to continue. If there is no response or the on-screen <Ok> button is not highlighted in red, hit the <Tab> key to move the focus to the <Ok> button.

![Figure 1.10 Entering a new password for the MySQL root user](image)
5. You will be asked to confirm the password. Re-type the same password and the installation will then continue until it is complete.

6. That's it, the LAMP installation is done! Congratulations! Now you just need to run some configuration processes and tests to confirm your installation.

Activity 1.4

1. What is LAMP and why do we want to install it?

2. ___________________________ is the single line command that will install the LAMP (Linux, Apache, MySQL and PHP) packages from Ubuntu.

Testing the configurations of Apache, PHP and MySQL

You can use the following steps to test the configurations of Apache, PHP and MySQL. You will first test the Apache server.

Apache

1. At this stage, the Apache web server should have been installed on your computer. Type the following in the terminal to check the version of Apache that was installed: apache2 -v

![Figure 1.11 Checking the version of Apache that was installed]
2. The version will be displayed. Check if the Apache service is running by typing `service apache2 status`.

![Figure 1.12 Confirming that the Apache service is running](image)

3. If the service is not running, type the following command to start the service:

   ```bash
   sudo service apache2 start
   ```

4. Once you have confirmed that the Apache service is running, you can test the Apache server by entering `http://localhost` on the web browser. If the installation is successful, you will be able to see the default Apache page that has the “It works!” headline (Figure 1.13). If you cannot see this page, visit the ‘error.log’ file located in the ‘/var/log/apache2/’ folder to see what went wrong with your installation.

![Figure 1.13 Testing your Apache installation](image)
PHP

1. After confirming that Apache is running, we can then test the PHP installation. By default, the document root directory for Apache is the ‘/var/www/’ directory. Note that you can change the document root directory path in the ‘/etc/apache2/sites-available/default’ file under the “DocumentRoot” section.

2. Let’s retain the default document root directory as ‘/var/www/’ and create a ‘test.php’ file in it. Open the terminal and call up the nano editor by entering:

   sudo nano /var/www/test.php

3. Enter the following PHP code into the new text file:

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

   ![Figure 1.14 Entering PHP code using the nano editor](image)

4. Hit <Ctrl> + <x> to exit the editor. Save the code you entered earlier by typing Y and hitting <Enter> for the default format option.

5. To ensure that the changes are updated in Apache, restart the Apache server using the following code on the terminal line:

   ```bash
   sudo service apache2 restart
   ```
6. You can now test the page you created by entering `http://localhost/test.php` on a web browser. You should get the PHP information page shown in Figure 1.15.

![Figure 1.15 Viewing PHP information on the Apache server](image)

**Web Reference**

To learn more about configuring Apache with PHP, please read the following:

“Ubuntu 10.10 — PHP5 — Scripting Language” at: https://help.ubuntu.com/10.10/serverguide/C/php5.html#php5-configuration
**MySQL**

1. After the installation, you should make sure that the MySQL database is bound to your localhost IP address. At the terminal, type:

   ```
   cat /etc/hosts | grep localhost
   ```

2. You can also verify the localhost bind address at ‘my.cnf’ of MySQL.

   ```
   cat /etc/mysql/my.cnf | grep bind-address
   ```

3. Notice that the bind-address = 127.0.0.1 (Figure 1.16)

4. Check that the MYSQL service is running by typing:

   ```
   service mysql status
   ```

   ![Figure 1.17 Checking that the MySQL service is running](image)
5. If the service is not running, type the following command to start the service:

   `sudo service mysql start`

6. After confirming that the MYSQL service is running, check the databases in the MySQL version that was installed. Follow the steps below to do this.

7. Type the following to enter the MySQL monitor:

   `mysql -h localhost -u root -p`

8. You will be prompted to enter a password. Enter the password `wawasan` that you set earlier.

9. You will then be in the MySQL monitor with the “mysql>” prompt showing on-screen. All commands used here will end with a semicolon “;” or “\g”. View your pre-installed databases by typing:

   `show databases;`

Do not forget to add the semicolon before you execute your command. You will then see some pre-installed databases listed. However, you might not have databases that are identical to those shown in the following figure.
10. You can list all the tables in the database by typing:

a. `use mysql` to use the 'mysql' database.

b. `show tables;` to list all the tables in the database.

![Figure 1.20 The MySQL monitor showing a list of tables]

11. Next, you can list the users in your MySQL system. In this case, you should already have a localhost root user with the password that you entered. Type:

```
SELECT User, Host, Password FROM mysql.user;
```

Note that the password is hashed for security purposes.

![Figure 1.21 The MySQL monitor showing a user list]

If everything works, you should get the results shown in Figure 1.21. If so, congratulations, your MySQL installation is fine.
Type `exit` twice to exit the MySQL monitor and terminal. Notice that you have just used the SQL query language to obtain the user list. The SQL query that you used was:

```
SELECT User, Host, Password FROM mysql.user;
```

The query is an example of the SQL `SELECT` statement, which is used to select data from a database. The general syntax of this statement is:

```
SELECT column_name(s) FROM table_name;
```

Do not worry about SQL commands for now. We are just giving you some basic knowledge on how you can interact with MySQL using the SQL query language. You will go into the details of SQL when you learn about MySQL in Unit 4.

---

**Web Reference**

To learn more about MySQL post-installation procedures, please visit “Unix Postinstallation Procedures” at:


---

**Activity 1.5**

1. Once you have confirmed that the Apache service is running, you can test the Apache server by entering `http://_________________________` on your web browser.

2. In a PHP content page, you enter the code `_____________ ________________` to test that PHP is working and to display information about PHP, Apache and MySQL on your system.
Summary

This section discussed the common techniques for installing Apache, MySQL and PHP in the Linux system. We have chosen to install the LAMP packages via a package manager because it is easier to do so. We are sure that you will be able to do the same by following the step-by-step guide provided in this section. We have also prepared a version of the installation guide for the Windows operating system. For Windows, we will be installing the components individually so that you will be exposed to different ways of doing things. If you need to install the packages in Windows, please refer to the Appendix.

In the next section, we will discuss basic PHP syntax and language constructs. You will also learn how PHP is embedded in a web page.

Self-test 1.2

1. In the Linux system, what is the command to check the Apache 2 service status?

2. You can change the document root directory path of the ‘/etc/apache2/sites-available/default’ file under the __________ section.
   A. ApacheRoot
   B. DefaultDirectory
   C. DocumentRoot
   D. RootDirectory

3. What is the command to enter the localhost MySQL monitor with root privileges?

4. All commands used in the MySQL monitor will end with either a __________ or __________.
   A. semicolon, \g
   B. comma, \g
   C. semicolon, \end
   D. comma, \end
Suggested answers to activities

Feedback

Activity 1.4

1. LAMP is a software bundle consisting of Linux, Apache, MySQL and PHP. We install LAMP to build an Apache web server in the Linux operating system. Apache will host a web database application via PHP, the middle-tier programming language and MySQL, the database component.

2. `sudo apt-get install lamp-server`

Activity 1.5

1. localhost or 127.0.0.1

2. `<?php phpinfo(); ?>`
1.3 Creating Basic PHP Scripts

Objectives

By the end of this section, you should be able to:

1. Describe and use basic PHP syntax and language constructs.
2. Embed PHP code in a web page.

Introduction

PHP, as a server-side scripting language, is intended largely for the Web environment. PHP has gained quite a following in recent times, and it is one of the forerunners of the Open-Source Software movement. Its popularity is due to its syntax, which is similar to that of the C programming language, and its simplicity. Basically, PHP allows a static web page to become a dynamic one. “PHP” is an acronym that stands for “Pre-Hypertext Processor”. This means that PHP code is parsed on a server before it creates HTML. PHP file names end with the “.php” extension. This extension notifies a web server that PHP code needs to be parsed before HTML is displayed on a web browser.

Overview of PHP syntax

PHP syntax is relatively straightforward. In this section, you will learn the general rules and basic syntax of PHP. The following are three basic rules that apply to PHP:

1. PHP statements are enclosed within the opening tag `<?php` and closing tag `?>`:

   ```php
   <?php PHP code ?>
   ```

   or

   ```php
   <?php
   PHP code
   ?>
   ```

   It does not matter if both the tags are on one line or if the second one starts on a new line. To a server, the code will show up as one continuous line, regardless of tabs or new lines. It is up to us to arrange the code so that it is organised and easily understood by people who may want to modify the code.
2. PHP statements end with a semicolon:

```php
<?php
    PHP statement one;
    PHP statement two;
?>
```

3. PHP comments are represented by double slashes `//--` for a single-line comment or by `/*` and `*/` for several lines of comment:

```plaintext
// This is a single-line comment

/*
   This is a multiple-line comment
*/
```

Comments are used throughout the code to explain the purpose of the code to people. We encourage you to use as many comments as you think are necessary, so that other developers can easily understand your programming.

Note: The following discussion assumes that you have basic knowledge of HTML.

---

**Web Reference**

For a tutorial on HTML, please refer to the following website:


---

**Creating your first web page with PHP**

As you may already know, web pages are actually generated by HTML code. A HTML document begins with a `<html>` tag and ends with a `</html>` tag. A typical HTML page is shown below:

```html
<html>
    <head>
        <title> HTML document </title>
    </head>
    <body>
        Content of the page.
    </body>
</html>
```
If we save the content above into a file with the extension “.html” (e.g., ‘first.html’), we can display this page in a web browser like Internet Explorer or Firefox.

We will be able to see the HTML source code that generated this web page by going to View > Source in Internet Explorer.

Notice that every opening tag, such as <head>, will always end with an ending tag that comes with a forward slash, such as </head>. PHP works in almost the same way. Every PHP opening tag will also end with an ending tag. The difference is that PHP code is hidden from web page visitors as PHP is a server-side scripting language. The only code that the visitors can see is the resulting HTML output.
Let’s start creating our first PHP script by using the `echo` function to print text in a web browser. We will reproduce the HTML code that we wrote earlier by using the PHP `echo` function. The `echo` function is used inside the PHP tags. We also change the content in the title tag to “PHP document” as shown below.

```
<html>
  <head>
    <title>PHP document</title>
  </head>
  <body>
    <?php
      echo "Content of the page."
    ?>
  </body>
</html>
```

We then save the content above into a file with the “.php” extension (e.g., ‘first.php’). A web page with the “.html” extension can be displayed from anywhere on the local disk. But as PHP is a server-side page, the file needs to be placed on the Apache server that we installed earlier. The location to store the server pages is ‘/var/www/’ for Linux or ‘C:\Program Files\Apache Software Foundation\Apache2.2\htdocs’ for Windows. Save the ‘first.php’ file there and display this page in a web browser by entering `localhost/first.php`.

![Image of a web browser displaying a PHP document](image.jpg)

**Figure 1.24** Web page displayed by embedded PHP code
Notice that you get the same result as that of the earlier HTML page except that the title of the page is now “PHP document”. Access the source code by going to View > Source on Internet Explorer. You will see HTML code that is nearly identical to the code that generated the HTML-only web page.

You should notice from looking at the source code that the PHP tags and `echo` function are all hidden from web page visitors. They only get to see the resulting HTML output. This will provide more security and flexibility to your PHP code.

PHP can also be written as a pure PHP program without being enclosed in HTML tags. Pure PHP code can be used to store business logic such as calculations and a separate HTML file is then used as the presentation layer. The advantage of doing this is that the same business logic can be reused for different presentations. You will learn more about the use of pure PHP code in the following units. An example is shown below.

```php
<?php
    echo "<html>
    <head>
        <title> Pure PHP document </title>
    </head>
    <body>
        Content of the page.
    </body>
</html>
?>
Congratulations, you have just created your very first PHP page.

**Activity 1.6**

1. “PHP” in the context of the scripting language for web development is an acronym that stands for “Personal Hypertext Protocol”. True / False?

2. A single line of PHP code ends with ____________.
   
   A. double slashes (//)
   B. an asterisk followed by a slash (*/)
   C. the hash key (#)
   D. a semicolon (;)

3. A HTML document begins with a ___________ tag and ends with a ___________ tag.
   
   A. <body>, </body>
   B. <head>, </head>
   C. <html>, </html>
   D. <title>, </title>
Web Reference

To learn more about basic PHP syntax, please visit:


Summary

You should now have some idea of what PHP code looks like and how PHP can be integrated into HTML code to create dynamic web pages. To summarise, you should now be familiar with:

1. PHP basic syntax.
2. The purpose of using comments throughout program code.
3. How PHP, as a server-side scripting language, gives you flexibility.
4. How to create a HTML web page using a pure PHP program.

In the next unit, you will learn about variables and data types, and learn how to create functions and control structures. You will also write a PHP program that contains a function that you will write yourself.

Self-test 1.3

1. PHP file names end with a .php extension to notify the server that PHP code needs to be parsed before the HTML web page is displayed in the user’s web browser. True / False?

2. The more comments you put into your PHP program code, the more processing time is needed by the web server to display a web page in a browser. True / False?
3. PHP code is always enclosed within the opening tag _______ and the closing tag ________.
   A. <!-- ... -->
   B. <% ... %>
   C. <? ... ?>
   D. <?php ... ?>

4. PHP comments are represented by ________ for a single-line comment and by _________ for several lines of comment.
   A. <?php and <? ... ?>
   B. ‘ and ” ... “
   C. // and <!-- ... -->
   D. // and /* ... */

5. The following PHP function is used to print text in a web browser:
   A. write
   B. echo
   C. println
   D. printing

6. Which of the following tags will not be visible in the web page source code when you go to View > Source on Internet Explorer?
   A. <?php
   B. <body>
   C. </html>
   D. <head>

7. What is the output produced in a web browser by the following PHP code?
   ```php
   <?php echo "This is TCC242\05"; ?>
   ```
   A. This is TCC242\05
   B. This is TCC24205
   C. This is TCC242\05
   D. “This is TCC242\05”
8. What is the output produced by the following PHP code?

```php
<?php echo "echo"; echo";"; ?>
```

A. echo;
B. echo
C. echo"; echo";
D. ;

9. What is the PHP code that produces the following result in a web browser?

```php
<?php
```

A. ```<?php echo"<?php "; ?>```  
B. ```<?php echo"&lt;?php "; ?>```  
C. ```<?php echo"\<?php "; ?>```  
D. ```<?php echo"\<?php "; ?>```  

10. What is the PHP code that produces the following result in a web browser?

```
18/01/2011 11:41:02
```

A. ```<?php echo date("H:i:s d/m/Y") ?>```  
B. ```<?php echo date("H:i:s m/d/Y") ?>```  
C. ```<?php echo date("m/d/Y H:i:s") ?>```  
D. ```<?php echo date("d/m/Y H:i:s") ?>```  

Suggested answers to activity

### Feedback

**Activity 1.6**

1. False
2. D
3. C
Summary of Unit 1

Summary

In this unit, you learnt about the development of different types of web sites. You also learnt about popular programming platforms that are used to create dynamic websites. You looked at several development methodologies, including the two popular development models in the Agile group of methodologies, which are used to ensure that web projects are delivered on time and within their scope and budget. You have also seen how databases are used on the Web.

In the second section, you learnt how to install, configure, run and test the Apache web server, MySQL and PHP on Linux systems. The installation procedures for Windows systems are covered in the Appendix. In the final section, you studied basic PHP syntax and discovered how PHP, as a server-side scripting language, gives you flexibility. You also learnt to write and execute a pure PHP program.

In the next unit, you will focus on PHP variables and data types, and learn about the constructs that PHP provides for controlling program execution flow.
Unit Practice Exercises

1. Create a PHP file in the ‘/var/www/Unit1’ directory (or ‘<Apache Directory>\htdocs\Unit1’ in Windows) and name it ‘MySite01.php’.

The PHP web page will consist of a title, a welcome message, your name and a message stating the current day, as laid out in the following table.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</td>
<td>My Site 01</td>
</tr>
<tr>
<td>Welcome Message</td>
<td>&lt;body&gt;&lt;h1&gt;…&lt;/h1&gt;&lt;/body&gt;</td>
<td>Welcome, {Your name}!</td>
</tr>
<tr>
<td>Current Day</td>
<td>&lt;?php&lt;br&gt;  // set the default time zone&lt;br&gt;  date_default_timezone_set('Asia/Kuala_Lumpur');&lt;br&gt;  // use echo function to write&lt;br&gt;  the string&lt;br&gt;  ...&lt;br&gt;  // Today’s date&lt;br&gt;  echo date(&quot;d F, Y&quot;);&lt;br&gt;?&gt;</td>
<td>Today is 09 January, 2011</td>
</tr>
</tbody>
</table>

Create the page as shown in the following figure.

Welcome, John!

Today is 09 January, 2011
2. In this exercise, you will modify the web page from question 1. Duplicate the PHP file from question 1 in the ‘/var/www/Unit1’ directory (or ‘<Apache Directory>\htdocs\Unit1’ in Windows) and rename it ‘MySite02.php’.

   a. Copy and paste the contents of the “About Wawasan” message into the file.

   b. Insert the footer note that includes your e-mail address.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>My Site 02</td>
</tr>
<tr>
<td>About Wawasan page</td>
<td><code>&lt;p&gt;…&lt;/p&gt;</code></td>
<td>{Copy About Wawasan page from <a href="http://wou.edu.my/about_overview.aspx%7D">http://wou.edu.my/about_overview.aspx}</a></td>
</tr>
<tr>
<td>Footer note</td>
<td><code>&lt;br&gt; &lt;span style=&quot;font-size:x-small&quot;&gt;The site is developed by &lt;a href=&quot;mailto:{your_email}&quot;&gt;{Your_Name}&lt;/a&gt;&quot;</code></td>
<td>e.g., “The site is developed by John Doe”</td>
</tr>
</tbody>
</table>

The results should look like the figure below.

![Welcome, John!](image)
3. In this exercise, you will modify the web page from question 2. Duplicate the PHP file from question 2 in the `/var/www/Unit1` directory (or `\Apache\Directory\htdocs\Unit1` in Windows) and rename it `MySite03.php`.

   a. Add the current time after the date by using the PHP function shown below.

   b. Insert Javascript code to update the clock every second.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td><code>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>My Site 03</td>
</tr>
<tr>
<td><strong>PHP current time</strong></td>
<td><code>// echo a space before the current time</code> <code>&lt;?php echo date(&quot;G:i:s&quot;); ?&gt;</code></td>
<td>e.g., 2:39:14</td>
</tr>
</tbody>
</table>
| **Insert Javascript into the `<head>`** | `<script>` ... `</script>`                                           | function showTime () {
| **clockFace id placement** | `<span id="clockFace"`>`<?php echo date("G:i:s"); ?></span>`              | e.g., 2:39:14   |
| **Set interval to showTime in body tag** | `<BODY onload="setInterval(showTime, 1000)">` | e.g., Time will auto update 2:39:15…16…17 |

The resulting web page should look like the one shown in the figure below.
Compare your results with the figure below.

4. In this exercise, you will modify the web page from question 3. Duplicate the PHP file from question 3 in the '/var/www/Unit1' directory (or '<Apache Directory>\htdocs\Unit1' in Windows) and name it 'MySite04.php'.
   
   a. Use the PHP random function to load a colour.
   
   b. Use the randomly loaded colour as the background colour of the web page.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>&lt;Head&gt;&lt;Title&gt;...&lt;/Title&gt;&lt;/Head&gt;</td>
<td>My Site 04</td>
</tr>
<tr>
<td>PHP random function</td>
<td>&lt;?php echo(dechex(rand(0,255))).dechex(rand(0,255)).dechex(rand(0,255));?&gt;</td>
<td>e.g., 296746</td>
</tr>
<tr>
<td>Insert the PHP random function into the body tag 'bgcolor'</td>
<td>&lt;BODY bgcolor=&quot;#&lt;?php echo(dechex(rand(0,255))).dechex(rand(0,255)).dechex(rand(0,255));?&gt;&quot; ...&gt;</td>
<td>Background colour will change every time the page is refreshed.</td>
</tr>
</tbody>
</table>

The following figure shows the result of the modification. Of course, changes in background colour can be better observed in a web browser than on a black-and-white paper page!
Welcome, John!

Today is 09 January, 2011 2:59:00

Wawasan Open University is Malaysia’s first private, not-for-profit, open learning institution, a lifelong learning community without borders. We use flexible approaches to make higher education accessible to all - anytime, anywhere. Our self-paced learning environment is specially designed for the convenience and accessibility of adult learners of any age, gender, ethnicity and background. Funded by charitable trusts, corporations and the public, we aim to lower the personal cost of learning and make it more affordable. If you wish to study for accredited qualifications or for personal enrichment without leaving your home or job, then Wawasan Open is for you.

The site is developed by John Doe.
Suggested Answers to Self-tests

Feedback

Self-test 1.1

1. True

2. True

3. programming languages, databases

4. D

5. Incremental prototyping:
   a. Allows large systems to be developed in phases with each release providing additional capabilities.
   b. In this model, the most important features of a system are developed to completion first and less important features are added later.
   c. Speeds up the implementation of a project.

6. Extreme Programming and Scrum.

7. False

8. Scrum is an Agile model that uses iterative development periods called sprints that usually last for 3 to 4 weeks. Each sprint achieves some targets. The team needs to meet up every morning before they start work to discuss development issues. At the end of a sprint, features that have been completed in that particular sprint are demonstrated to the customer's representative.

9. A

10. True
Self-test 1.2

1. `service apache2 status`

2. C

3. `mysql -h localhost -u root -p` (and enter the root password)

4. A

Self-test 1.3

1. True

2. False

3. D

4. D

5. B

6. A

7. C

8. A

9. B

10. D
Suggested Answers to Unit Practice Exercises

Feedback

1. MySite01.php

```html
<HTML>
<HEAD>
<TITLE>My Site 01</TITLE>
</HEAD>
<BODY>
<h1>Welcome, John!</h1>
<?php
// set the default time zone
date_default_timezone_set('Asia/Kuala Lumpur');
// Today's date
echo "Today is ";
echo date("d F, Y");
echo " ";
?>
</BODY>
</HTML>
```

2. MySite02.php

```html
<HTML>
<HEAD>
<TITLE>My Site 02</TITLE>
</HEAD>
<BODY>
<h1>Welcome, John!</h1>
<?php
// set the default time zone
date_default_timezone_set('Asia/Kuala Lumpur');
// Today's date
echo "Today is ";
echo date("d F, Y");
echo " ";
?>
<p>
</p>
<br>
<span style="font-size:x-small">The site is developed by <a href="mailto:johndoe@nothing.com">John Doe</a>"/></span>
</BODY>
</HTML>
```
3. MySite03.php

<HTML>
<HEAD>
<TITLE>My Site 03</TITLE>
<script>
function showTime () {
var time = new Date()
var hour = time.getHours()
var minute = time.getMinutes()
var sMin = (minute<10) ? "0" + minute : minute
var second = time.getSeconds()
var sSecs = (second<10) ? "0" + second : second
var strTime = hour + "" + sMin + "" + sSecs
document.getElementById("clockFace").innerHTML = strTime;
}
</script>
</HEAD>
<BODY onload="setInterval(showTime, 1000)">
<h1>Welcome, John!</h1>
<?php
// set the default time zone
date_default_timezone_set('Asia/Kuala_Lumpur');
// Today’s date
echo "Today is ";
echo date("d F, Y");
// echo a space before the current time
echo " ";
?>
<span id="clockFace">
<?php echo date("G:i:s");
?></span>
<p>
</p>
<br>
<span style="font-size:x-small">The site is developed by <a href="mailto:johndoe@nothing.com">John Doe</a>
</span>
</BODY>
</HTML>
4. MySite04.php

```
<html>
<head>
<title>My Site 04</title>
<script>
function showTime () {
  var time = new Date();
  var hour = time.getHours();
  var minute = time.getMinutes();
  var sMin = (minute<10) ? "0" + minute : minute;
  var second = time.getSeconds();
  var sSecs = (second<10) ? "0" + second : second;
  var strTime = hour + ":" + sMin + ":" + sSecs;
  document.getElementById("clockFace").innerHTML = strTime;
}
</script>
</head>
<body bgcolor="<?php echo(dechex(rand(0,255)).dechex(rand(0,255)).dechex(rand(0,255)));">
<hi>Welcome, John!</hi>
<?php
  // set the default time zone
  date_default_timezone_set('Asia/Kuala_Lumpur');
  // Today's date
  echo "Today is ";
  echo date("d F, Y");
  // echo a space before the current time
  echo " ";
  ?>
  <span id="clockFace"><?php echo date("G:i:s")?></span>
  </p>
  </p>
  <br>
  <span style="font-size:x-small">The site is developed by <a href="mailto:johndoe@nothing.com">John Doe</a></span>
</body>
</html>
```
Appendix

Introduction

This appendix introduces you to the general techniques for installing Apache, MySQL and PHP on the Windows operating system. The installation steps for the Linux system were covered in section 1.2.

Installing Apache

The Apache web server will be the server hosting the PHP and MySQL components that you will be using soon. The following guide works for Windows XP SP 3 and above. For other versions of Windows, please read the notes for Win32 at http://httpd.apache.org.

Installation steps

1. Visit www.apache.org, scroll down and look for the HTTP Server hyperlink and click on it (Figure 1.27).

Figure 1.27 The HTTP Server hyperlink at www.apache.org
2. You will arrive at the Apache HTTP Server Project page. Look for the Apache HTTP Server 2.2 latest and stable releases, usually located on the second row. Avoid selecting the alpha releases. In this guide, the **Apache HTTP Server 2.2.17** is selected. Click on the **Download** hyperlink (Figure 1.28).

![Figure 1.28 Download hyperlink of Apache HTTP Server 2.2.17](image)

3. This will take you to the download page. Look for the section entitled “Apache HTTP Server (httpd) 2.2.17 is the best available version” and select **Win32 Binary without crypto (no mod_ssl) (MSI Installer): httpd-2.2.17-win32-x86-no_ssl.msi** (Figure 1.29). At this stage, security is not our major concern, so it is fine to select the installer without crypto. Download the MSI Installer to your computer's hard disk.

![Figure 1.29 Win32 Binary without crypto](image)
4. Once the download is complete, launch the installer by double-clicking on its icon. This will bring up the “Apache HTTP Server 2.2 — Installation Wizard”.

![Figure 1.30 Apache HTTP Server 2.2 — Installation wizard](image)

5. Click on **Next** to begin the installation process.

![Figure 1.31 Apache HTTP Server 2.2 — License agreement](image)
6. You will be shown the License Agreement (Figure 1.31). Select the button to accept the License Agreement and then click on Next >. You will then be shown the “Read This First” screen. Read it if you want to know more about Apache.

![Figure 1.32 Apache HTTP Server 2.2 — Read this first](image)

7. Continue by clicking on Next > and you should arrive at the Server Information screen (Figure 1.33). Enter the following information:

   a. Network Domain: domain_name.com (or something similar)

   b. Server Name: testpc.domain_name.com (or something similar)

   c. Administrator’s Email Address: admin@domain_name.com (or something similar)

   d. Select the button to install Apache programs and shortcuts for ‘All Users’ so that the server will be available to anyone who arrives at your site.

   e. Click on Next > to continue.
Next, select the Setup Type (Figure 1.34). You may choose “Custom” so that you will know in detail the components and features that you are installing.
9. You may read the feature description of each component on the “Custom Setup” screen. Leave the default installation path as it is and click on Next > to continue.

![Figure 1.35 Apache HTTP Server 2.2 — Custom Setup](image1.jpg)

10. You are now ready to install the program. Click on Install (Figure 1.36) and let the installation process begin.

![Figure 1.36 Apache HTTP Server 2.2 — Ready to install](image2.jpg)
11. When the installation is complete, click on Finish (Figure 1.38).

![Figure 1.37 Apache HTTP Server 2.2 — Installing](image)

![Figure 1.38 Apache HTTP Server 2.2 — Installation completed](image)
Installing PHP

PHP is a widely used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. The following guide works for Windows XP SP 3 and above. For other versions of Windows, please read the notes for Windows systems at http://php.net/manual/en/install.windows.php.

Installation steps

1. Visit www.php.net, look for the downloads hyperlink at the top of the web page and click on it (Figure 1.39).

2. Look for the PHP 5.3 current stable releases. In this guide, we will be using PHP 5.3.4. Scroll to the Windows Binaries section and click on http://windows.php.net/download/.
3. You will arrive at a download page showing various versions of PHP. Since we are using Apache 2.2 from apache.org, we will select the VC6 x86 Thread Safe version for non-CGI applications. Click on VC6 x86 Thread Safe, Zip format.

![VC6 x86 Thread Safe, Zip format](image)

**Figure 1.41** PHP.net — VC6 x86 Thread Safe, Zip format


4. Save the zip file to your local disk. Then, unzip the package with any standard unzip program. We recommend you unzip the package to C:\ and rename it ’C:\php’. Be careful not to insert spaces in between the directory name to avoid the web server string concatenation problem.

5. The next step is to set up a valid configuration file for PHP, ‘php.ini’. Go to ‘C:\php’ and look for the file named ‘php.ini-development’. There are two ‘ini’ files distributed in the zip file, ‘php.ini-production’ and ‘php.ini-development’ **(Figure 1.42)**. The ‘php.ini-production’ file is optimised for performance and security. For the purposes of learning, you will be using the ‘php.ini-development’ file for simplicity and fewer configuration steps. You can switch to ‘php.ini-production’ once you are more familiar with PHP. Therefore, rename ‘php.ini-development’ as ‘php.ini’. 
6. Next, you will make a copy of the ‘php.ini’ file (Figure 1.43) and place it in the ‘C:\WINDOWS’ directory so that Apache can find it.

![Figure 1.42 Renaming php.ini-development as php.ini](image1)

![Figure 1.43 Copy the php.ini file into C:\WINDOWS](image2)
7. Finally, you need to copy the ‘php5ts.dll’ file (since we are using PHP version 5) from ‘C:\php’ into the ‘C:\Program Files\Apache Group\<Apache Version>\bin’ directory, which is named ‘C:\Program Files\Apache Software Foundation\Apache2.2\bin’ in this case.

![Copy php5ts.dll from C:\php into the <Apache Version>\bin directory](image)

You have now completed the installation of both Apache and PHP on the Windows system. Next, you are going to configure Apache and PHP.

**Configuring Apache and PHP**

At this point, the Apache web server should already have been installed on your computer. You should be able to see the “Apache Service” taskbar icon (**Figure 1.45**) at the lower right-hand corner of your Windows desktop screen if you had accepted all the default installation features.

![Apache Service taskbar icon](image)
If you cannot see the icon, the Apache service can be accessed via the Start menu or from the program installation directory, which by default is at ‘C:\Program Files\Apache Software Foundation\Apache2.2\bin’.

Figure 1.46 The Apache service can be accessed via the Start menu

Once you have confirmed that the Apache service is running, you can test the Apache server by entering http://localhost in the web browser. If the installation was successful, you will be able to see the default Apache page with the “It works!” headline. If this web page is not displayed, visit the ‘error.log’ file located in the ‘C:\Program Files\Apache Software Foundation\Apache2.2\logs’ folder to see what went wrong with your installation. The setup may fail if Apache is trying to share HTTP port 80 with another web server, such as Windows IIS, or an application, such as a firewall.

Figure 1.47 Test the Apache installation with http://localhost

After verifying that your Apache installation works fine, you then need to add some code to the configuration file, ‘httpd.conf’, which can be found in the ‘C:\Program Files\Apache Software Foundation\Apache2.2\conf’ folder, before you can run PHP. This file can be opened with any text editor, for instance Notepad. You are advised to make a backup copy of the ‘httpd.conf’ file before you start making any modifications to it.
Configuring Apache to recognise PHP files

1. Go to the ‘C:\Program Files\Apache Group\Apache Version\conf’ directory, which is ‘C:\Program Files\Apache Software Foundation\Apache2.2\conf’ in this case. Make a copy of the ‘httpd.conf’ file and name it ‘httpd.conf.backup’.

2. Open ‘httpd.conf’ with a text editor like Notepad. Use the text editor to find the “LoadModule” section. Scroll to the end of the “LoadModule” section and add the following code after the last #LoadModule line (Figure 1.49).

   #Insert the following to set up the PHP 5 module for Apache 2.2.
   #Make sure your path matches the location of this file.
   LoadModule php5_module "c:/php/php5apache2_2.dll"

3. Note that the words followed by the hash character, “#” are interpreted as comments and they will not be executed. Please make sure the path ‘php5apache2_2.dll’ matches the location of the existing file.
4. Next, find the “AddType” section. Scroll to the end of the “AddType” section and add the following code after the last line of this section.

```
# In order for Apache to recognise a PHP file as one that needs to be
# parsed with the PHP engine, you need to add the following two lines
AddType application/x-httpd-php .php
# For syntax highlighted .phps files, also add
AddType application/x-httpd-php-source .phps
```

![Figure 1.50 Adding the AddType code to httpd.conf](image)

5. Save the changes you made to ‘httpd.conf’ and close the file. Launch the Apache Service Monitor (Figure 1.51) from the Apache Service taskbar icon by right clicking on the icon and then selecting Open Apache Monitor. You can also launch the Monitor from the start > All Programs menu if you did not install the taskbar icon.

![Figure 1.51 The Apache Service Monitor](image)
6. Select **Apache 2.2** and then click the **Restart** button. Note that after you restart the service, there will be a “PHP/5.3.4” label shown next to the “Apache/2.2.17 (Win32)” label at the bottom left-hand side of the monitor window (Figure 1.52). This shows that you have successfully configured Apache with PHP. If not, check for errors and revisit the steps above.

![Apache Service Monitor](image)

**Figure 1.52** An additional “PHP/5.3.4” label situated next to the “Apache/2.2.17 (Win32)” label indicates that you have successfully configured Apache with PHP.

Once Apache is configured to recognise PHP, you can test a PHP script. By default, the document root directory for Apache is the ‘C:\Program Files\<Apache Group>\<Apache Version>\conf’ directory, which is ‘C:\Program Files\Apache Software Foundation\Apache2.2\htdocs’ in this case. Note that you can change the document root directory path in the ‘httpd.conf’ file under the “DocumentRoot” section.

Let us now use the default document root directory (‘C:\Program Files\Apache Software Foundation\Apache2.2\htdocs’) and create a ‘test.php’ file in it. Using Notepad, enter the following code into a new text file:

```html
<?php
phpinfo();
?>
</body>
</html>
```
Save this file as ‘test.php’ in the ‘C:\Program Files\Apache Software Foundation\Apache2.2\htdocs’ directory. You can now test the page you created in a web browser by entering \texttt{http://localhost/test.php}. The PHP information page shown in the following figure should be displayed.

![PHP Information Page](image1)

\textbf{Figure 1.53} The output of test.php in a web browser

\section*{Installing MySQL}

MySQL will be used as the database to hold the information that is going to be accessed by the web application that you will learn to create. The following guide describes how to install a MySQL database. It works for Windows XP SP 3 and above. For other versions of Windows, please read “Installing MySQL on Microsoft Windows” at \url{http://dev.mysql.com/doc/refman/5.5/en/windows-installation.html}.

\subsection*{Installation steps}

1. Visit \url{www.mysql.com} and click on the \textbf{Downloads (GA)} tab (Figure 1.54).

![MySQL.com Downloads (GA) Tab](image2)

\textbf{Figure 1.54} MySQL.com — Downloads (GA) tab
2. On the download page, scroll to the “MySQL Community Server” section and click on the **DOWNLOAD** hyperlink shown in Figure 1.55.

![Figure 1.55 MySQL.com — MySQL Community Server download section](image)

3. Select the **Microsoft Windows** platform, choose the **Windows (x86, 32-bit)**, **MSI Installer** by clicking on the **Download** button next to it (Figure 1.56).

![Figure 1.56 MySQL Microsoft Windows (x86, 32-bit), MSI Installer](image)
4. You will be brought to a registration page. You may register or simply skip to the download page by clicking on “No thanks, just take me to the downloads!”.

![Registration page](image)

**Figure 1.57 MySQL.com — Registration page**

5. On the download page, scroll to the “Asia” section, select the nearest mirror site and click on its HTTP download hyperlink.

![Select the nearest mirror site](image)

**Figure 1.58 MySQL.com — Select the nearest mirror site**
6. Download the MSI Installer to your local disk. Once the download is complete, launch the installer by double-clicking on the icon. This will bring up the “MySQL Server Setup Wizard”.

![MySQL Server Setup Wizard](image1)

**Figure 1.59** MySQL Server Setup Wizard

7. Click on **Next** to start the setup wizard for MySQL Server.

![MySQL Server Setup — License Agreement](image2)

**Figure 1.60** MySQL Server Setup — License Agreement
8. After accepting the License Agreement and clicking on **Next**, you will be brought to the “Choose Setup Type” screen. Click on the **Custom** button.

![MySQL Server Setup — Choose setup type](image)

**Figure 1.61** MySQL Server Setup — Choose setup type

9. On the “Custom Setup” screen, click on the **Browse** button to re-select the program destination path.

![MySQL Server Setup — Custom setup](image)

**Figure 1.62** MySQL Server Setup — Custom setup
10. In the “Folder name:” box, enter ‘C:\mysql’ and click on OK to close the “Change destination folder” window. Click on Next in the “Custom Setup” screen to continue.

![MySQL Server Setup — Changing the destination folder](image1)

**Figure 1.63** MySQL Server Setup — Changing the destination folder

11. You are now ready to install the program. Click on Install and let the installation begin.

![MySQL Server Setup — Ready to install](image2)

**Figure 1.64** MySQL Server Setup — Ready to install
12. During installation, the MySQL Enterprise subscription window may pop up. Simply click on **Next** twice and this will bring you to the end of the installation.

![MySQL Server Setup — Installing](image1.png)

**Figure 1.65** MySQL Server Setup — Installing

![MySQL Enterprise subscription window](image2.png)

**Figure 1.66** MySQL Server Setup — MySQL Enterprise subscription window
13. When the installation is complete, leave the “Launch the MySQL Instance Configuration Wizard” checkbox checked and click on Finish to end the installation.

![MySQL Server Setup — Setup completed](image)

Next, you will use the “MySQL Instance Configuration Wizard” to configure the MySQL installation in the ‘my.ini’ configuration file.

**Configuring and running MySQL**

The “MySQL Instance Configuration Wizard” will be launched once the setup is completed. You can also launch the wizard manually from the start > All Programs menu in Windows.

![Launching the MySQL Instance Configuration Wizard from the program menu](image)
Use the following steps in the wizard to configure MySQL.

**Configuration steps**

1. On the welcome screen of the wizard, click on Next > to begin configuration.

![MySQL Server Instance Configuration Wizard welcome screen](image)

**Figure 1.69** MySQL Server Instance Configuration Wizard welcome screen

2. Select the “Detailed Configuration” option to learn more about and have control over the configuration. Click on Next > to continue.

![MySQL Server Instance Configuration Wizard](image)

**Figure 1.70** MySQL Configuration — Detailed configuration selection
3. MySQL will be installed on your personal computer intended for personal use, so we assume that other applications will be running and MySQL will have to use minimal system resources. Thus you will select the “Developer Machine” option.

4. The “Multifunctional Database” option is for regular use and will enable both InnoDB (for fast transactions) and MyISAM (high speed) storage engines to divide resources equally. Select this option, then click on **Next >**.
5. For better performance, you may want to place the InnoDB Tablespace in a separate location like in a RAID storage system. But for your learning purposes, you will place it in the “G:” drive and the “Installation Path”. Click on Next >.

![MySQL Configuration — InnoDB tablespace settings](image)

**Figure 1.73** MySQL Configuration — InnoDB tablespace settings

6. We can set the limits of concurrent connections to the MySQL Server but since you will be the only user connecting to MySQL in this case, this setting is not important at the moment. Select the “Decision Support (DSS)/OLAP” option, then click on Next >.

![MySQL Configuration — number of concurrent connections](image)

**Figure 1.74** MySQL Configuration — number of concurrent connections
7. We need to enable TCP/IP for MySQL to run properly. In any Windows installation that has a firewall enabled, you need to open the MySQL port 3306 for connections from your firewall configuration. Check the “Enable TCP/IP Networking” option, retain the default port number as 3306 and ensure that the “Add firewall exception for this port” checkbox is checked. You will also enable strict mode so that MySQL enforces rules to control and handle invalid or missing input values. Check the “Enable Strict Mode” box and then click on **Next >**.

![Figure 1.75 MySQL Configuration — enabling TCP/IP, firewall exception and strict mode](image)

8. You will now select the default character set. If you will be working with international languages that require the UTF8 character set, you should choose the “Best Support for Multilingualism” option. Otherwise, select the “Standard Character Set” option as the default server character set. Then click on **Next >**.

![Figure 1.76 MySQL Configuration — Standard character set option](image)
9. In the MySQL Server Instance Configuration screen, selecting the “Install As Windows Service” option makes MySQL Server start automatically when Windows boots up. Check the boxes as shown in the figure below and then click on **Next >**.

![MySQL Configuration — Windows service and add the bin directory to the PATH variable](image)

**Figure 1.77 MySQL Configuration**

10. We recommend that you set the root password for your MySQL Server to increase server security and to avoid any login and permission difficulties. You will use the password *wawasan*. Do not check the “Anonymous Account” option. Click on **Next >** after setting the password.

![MySQL Configuration — Security settings](image)

**Figure 1.78 MySQL Configuration**

Security settings
11. Now the final screen will be displayed. Click on the **Execute** button to start the configuration process.

![MySQL Configuration — Executing the configuration](image1.png)

**Figure 1.79** MySQL Configuration — Executing the configuration

12. Once the configuration file is created, you will see a summary as shown in the figure below. Click on **Finish** to exit the wizard.

![MySQL Configuration — Summary of the tasks performed](image2.png)

**Figure 1.80** MySQL Configuration — Summary of the tasks performed
If for some reason you need to reinstall and reconfigure MySQL Server, follow the steps below to remove MySQL Server first.

1. Go to Start > Control Panel > Add or Remove Programs, select MySQL Server 5.5 and then click on the Remove button.

2. Once the uninstall process is completed, manually remove the MySQL program folder ‘C:\mysql’.

3. Navigate to ‘C:\Documents and Settings\All Users\Application Data’ and remove the MySQL folder. If you cannot see the ‘Application Data’ folder, go to Start > Control Panel > Folder Options, select the View tab and under the ‘Hidden files and Folders’ section, choose Show hidden files and folders.

4. If you are told to restart the system, follow the on-screen instructions to reboot Windows before you continue.

5. Go back to the steps above to install, configure and run MySQL Server.

**Test your installation to ensure that MySQL Server is running**

After the installation, you should make sure that MySQL Server is running.

1. Go to Start > Run. Type `cmd` in the ‘Open:’ box.

![Figure 1.81 The Run applet in Windows](image)

2. Change the directory by typing `cd C:\mysql\bin` and pressing <Enter>.
3. Then type `mysql -h localhost -u root -p` to enter the MySQL monitor. You will be prompted to enter a password. Type the password `wawasan` that you set earlier and press <Enter>.

![MySQL Monitor](image1.png)

**Figure 1.82** Entering MySQL monitor

4. You will now be in the MySQL monitor with the `mysql>` prompt displayed. As indicated in the message, all commands used here will end with ; or \g. View your pre-installed databases by typing `show databases;`. Do not forget to add the semicolon before you press <Enter>. You will notice that there are four pre-installed databases listed.

![MySQL Monitor — Show databases](image2.png)

**Figure 1.83** MySQL Monitor — Show databases
5. You can list all the tables in the database by typing:

a. use mysql to use the mysql database, and then

b. show tables; to list all the tables in the database.

![Figure 1.84 MySQL Monitor — Show tables](image)

6. Next, you can list the users of your MySQL database. In this case you should only have a localhost root user with the password that you entered. Type `SELECT User, Host, Password FROM mysql.user;`. Take note that the password is hashed for security purposes.

![Figure 1.85 MySQL Monitor — Show user list](image)

If everything works, you should get the results shown in the figure above. Congratulations, your MySQL Server installation now works. Type `exit` twice to exit the MySQL monitor and the Windows command prompt. Do not worry about all the commands for now. We will go into details when we discuss MySQL in Unit 4.
Summary

This appendix discussed the common techniques for installing Apache, MySQL and PHP on a Windows system. We have chosen to install the components separately so that you know how to setup each component. If you need to install Apache, MySQL and PHP in a Linux system, please refer to section 1.2.
References


## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>An open-source web server.</td>
</tr>
<tr>
<td>Client/Server</td>
<td>An architecture in which the web server is the server and the web browser is the client.</td>
</tr>
<tr>
<td>Comments</td>
<td>Are used throughout program source code to explain the purpose of the code.</td>
</tr>
<tr>
<td>Dynamic websites</td>
<td>Websites that have dynamic content created by web programming languages and databases.</td>
</tr>
<tr>
<td>Echo</td>
<td>PHP function used to print text in a web browser.</td>
</tr>
<tr>
<td>Extreme Programming</td>
<td>Also known as XP. It is a software development model used in rapidly changing development environments; it encourages developers to work in pairs.</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language. It is the predominant markup language for web pages.</td>
</tr>
<tr>
<td>LAMP</td>
<td>Linux, Apache, MySQL and PHP.</td>
</tr>
<tr>
<td>Linux</td>
<td>An open-source operating system.</td>
</tr>
<tr>
<td>MySQL</td>
<td>An open-source relational database management system (RDBMS).</td>
</tr>
<tr>
<td>MySQL monitor</td>
<td>A command area that gives a user the ability to query MySQL server.</td>
</tr>
<tr>
<td>Open-source</td>
<td>Any software program whose source code is available for modification and is free of charge.</td>
</tr>
<tr>
<td>Personal sites</td>
<td>Websites where people share information about themselves or their products.</td>
</tr>
<tr>
<td>PHP</td>
<td>Hypertext Preprocessor. A server-side scripting language.</td>
</tr>
<tr>
<td>Portals</td>
<td>Online platforms or online showcases.</td>
</tr>
<tr>
<td>Professional sites</td>
<td>Websites that provide information on a business and/or generate revenue for a business.</td>
</tr>
<tr>
<td><strong>RAD</strong></td>
<td><strong>Rapid Application Development.</strong> A development approach in which developers work with an evolving prototype.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Root</strong></td>
<td><strong>A user with administrator privileges in the Linux system.</strong></td>
</tr>
<tr>
<td><strong>Scrum</strong></td>
<td><strong>A development approach that uses iterative development to address requirement changes.</strong></td>
</tr>
<tr>
<td><strong>Static websites</strong></td>
<td><strong>Websites that have fixed content and do not contain any programmatically controlled content.</strong></td>
</tr>
<tr>
<td><strong>Sudo</strong></td>
<td><strong>Used to give a Linux command with root (administrator) privileges.</strong></td>
</tr>
<tr>
<td><strong>Terminal</strong></td>
<td><strong>Command line or console window in the Linux system.</strong></td>
</tr>
<tr>
<td><strong>Web database application</strong></td>
<td><strong>A web application that involves a database, in which the browser is the client, the web server is the middle tier and the database server is the database tier.</strong></td>
</tr>
<tr>
<td><strong>Web development</strong></td>
<td><strong>Conception, design and implementation of a web entity created to serve a specific purpose.</strong></td>
</tr>
<tr>
<td><strong>Web page</strong></td>
<td><strong>A document on the World Wide Web that may contain text, images and hyperlinks.</strong></td>
</tr>
</tbody>
</table>
Unit 2
TCC 242/05
Web Database Application

Data Types, Functions and Control Structures
Wawasan Open University is Malaysia’s first private not-for-profit tertiary institution dedicated to adult learners. It is funded by the Wawasan Education Foundation, a tax-exempt entity established by the Malaysian People’s Movement Party (Gerakan) and supported by the Yeap Chor Ee Charitable and Endowment Trusts, other charities, corporations, members of the public and occasional grants from the Government of Malaysia.

The course material development of the university is funded by Yeap Chor Ee Charitable and Endowment Trusts.

© 2011 Wawasan Open University. Except where otherwise noted, this work is licensed under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

This course material was published to support the learning of students registered with Wawasan Open University. Wawasan Open University does not grant any degree, certification or credits based solely on your completion of this course material.

Wawasan Open University
Jalan Sultan Ahmad Shah,
10050 Penang.
Tel: (604) 2180333   Fax: (604) 2279214
Email: enquiry@wou.edu.my
Website: www.wou.edu.my
Contents

Unit 2  Data Types, Functions and Control Structures

Unit overview 1
Unit objectives 1

2.1 Working with data types and building expressions 3
Objectives 3
Introduction 3
What are data types? 3
Building expressions 9
Suggested answers to activities 16

2.2 Working with functions 19
Objectives 19
Introduction 19
What are functions? 19
Applying PHP built-in functions 19
Creating customised functions 21
Suggested answers to activities 30

2.3 Control structures 31
Objectives 31
Introduction 31
The if, elseif and else statements 31
The switch statement 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>for</code> loop</td>
<td>36</td>
</tr>
<tr>
<td>The <code>while</code> and <code>do-while</code> loops</td>
<td>38</td>
</tr>
<tr>
<td>Suggested answers to activities</td>
<td>43</td>
</tr>
<tr>
<td>Summary of Unit 2</td>
<td>45</td>
</tr>
<tr>
<td>Unit practice exercises</td>
<td>47</td>
</tr>
<tr>
<td>Suggested answers to self-tests</td>
<td>53</td>
</tr>
<tr>
<td>Suggested answers to unit practice exercises</td>
<td>57</td>
</tr>
<tr>
<td>References</td>
<td>63</td>
</tr>
<tr>
<td>Glossary</td>
<td>65</td>
</tr>
</tbody>
</table>
Unit Overview

This unit introduces you to the fundamental components of the PHP scripting language such as data types and functions. It also introduces you to control structures such as loops and conditional statements.

The first section discusses several PHP data types and explains how these data types are used. It also explains how data type conversion is performed and how expressions are built. The second section describes the use of functions for efficient coding. You will learn to use built-in PHP functions as well as create your own customised functions. The third section discusses how to control program flow with control structures. It explains the use of conditional statements and various types of loops.

Unit Objectives

By the end of this unit, you should be able to:

1. Define the data types used in PHP.
2. Explain how expressions are written in PHP.
3. Implement customised functions.
4. Apply various control structures such as loops and conditional statements.
2.1 Working with Data Types and Building Expressions

Objectives

By the end of this section, you should be able to:

1. Identify the main PHP data types.
2. Explain how data types are used.
3. Perform data type conversions.
4. Explain how to build expressions in PHP.

Introduction

You created your very first PHP web page in Unit 1 and it was written in basic PHP syntax. In this section, you will be exposed to more PHP code in a systematic manner. Let us first take a look at the data types that are supported in PHP and see how they are used. Later in this section, we will examine how PHP expressions are created.

What are data types?

In Malaysia, there are different ethnic groups such as Malay, Chinese, Indian and other minority groups. Each of these groups has its own set of characteristics. The same concept applies to data in PHP as data values in PHP are grouped based on their characteristics. These different groups of data are called data types.

There are two main categories of data types, namely:

1. Scalar types
2. Compound types

Scalar types

Scalar types store single items of data. The four primitive scalar types in PHP are:

1. Boolean
2. Integer
3. Float

4. String

**Boolean**

The Boolean data type is the simplest one. Its value can only be either TRUE or FALSE. Alternatively, you can use a non-zero value (usually “1”) to represent TRUE and a zero (“0”) value to represent FALSE. The values are also not case-sensitive, which means that TRUE, True and true are identical and are all valid. Consider the following examples:

```php
$open = True;  // $open is TRUE
$open = TRUE;  // $open is TRUE
$open = 1;     // $open is TRUE
$open = 0;     // $open is FALSE
$open = false; // $open is FALSE
```

Each line above is called an assignment statement. The combination of a dollar sign ($) and letters (open) on the left-hand side of the “equals” sign (which is called the **assignment operator** in PHP and does not have the same meaning as the equals sign in mathematics) in each line above is called a **variable** in PHP. We will discuss variables after the data type topic but at this stage we will just say that each statement above assigns a Boolean value (TRUE or FALSE) to the variable `$open` as shown by the comment line after each statement.

**Integer**

Any number, either positive or negative, that does not have a fractional part is an integer. The number zero is also an integer. An integer is optionally preceded by a sign (− or +). Integers can be written in base 8 (octal), base 10 (decimal) or base 16 (hexadecimal) formats. The following examples show how a variable ($i$) is assigned different integer values:

```php
$i = 0567;       // octal integer
$i = 24205;     // decimal integer
$i = -2011;     // negative integer
$i = 0x1B;      // hexadecimal integer
```

Note: To use octal notation, precede the number with “0” (zero). To use hexadecimal notation, precede the number with “0x”.

Float (Floating-point numbers)

Floating-point numbers are numbers that have fractional parts. They are also known as real numbers. In PHP, floating-point numbers are known as either floats or doubles. They are used when an integer value cannot adequately represent the preciseness of a number. The following examples show how a variable ($f$) is assigned different floating-point values:

```
$f = 0.567; // floating point number
$f = 5.0; // floating point number
$f = 2.42e5; // in exponential format
$f = 2.0E-11; // in engineering format
```

String

A string is a sequence of characters that are treated as a contiguous group and delimited by single or double quotes. Generally, PHP does not limit the size of a string; its size depends on the amount of computer memory available. In the following examples, the characters within the quote marks form strings; for example, `simple string` is a string.

```
echo 'simple string'; // String
echo 'It can embed a new line'; // Strings can embed new lines
echo 'I’ll be back'; // Escape sequence \
echo "Print capital A: \x41"; // Escape sequence \x41 // outputs A
```

In all the examples above, the `echo` function causes the strings to be displayed on the computer screen. You will learn more about strings and string library functions in Unit 3.

Compound types

Compound data types store multiple items of data under a single representation. The compound data types are:

1. Arrays
2. Objects
Arrays

An array is an ordered map that associates values to keys. It is an indexed collection of data values. Each item in an array is accessed using a simple integer reference, called a key, corresponding to the item. For example, to create a list of states in Malaysia, you could use an integer indexed array called $state, as shown below:

```php
$state[0] = "Kuala Lumpur";
$state[1] = "Penang";
...
$state[13] = "Johor";
```

Arrays will be discussed in more detail in Unit 3. Do not worry if you do not completely understand the concept at this point.

Objects

The term “object” refers to an essential component in object-oriented programming. An object is a particular instance of a class. In PHP, an object is created using the new statement. Here is how you can define a class (called program), instantiate an object (called $instance) of the class and then use the object to do something (output “An action”):

```php
class program // Define a class
{
    function do_work()
    {
        echo “An action”;
    }
}

$instance = new program; // Instantiate an object
$instance -> do_work(); // Output: An action
```

Web Reference

The following web reference presents a comprehensive view of classes and objects:

Variables

A variable is a “container” that can hold different values of a particular data type. In PHP, variables are represented by a dollar sign ($) followed by the name of the variable. The names of PHP variables are case-sensitive. This means that $var and $Var are two different variables.

PHP does not require explicit type definition in variable declaration. This means that if a variable, such as $a, is assigned a string value, it becomes a string. If it is assigned an integer value, it becomes an integer. You do not need to specifically declare the type before using it. As PHP is a flexible language, it allows variables to be set to any data type. You could first define a variable as a string and then later replace the string with a number, as shown in the example below:

```php
$var = "two four two";
$var = 242;
```

Type conversion

You can convert values from one data type to another using type casting (type conversion). This can be done by placing the casting operator of the intended type in front of the variable that you intend to convert. Refer to Table 2.1 for a list of PHP casting operators.

<table>
<thead>
<tr>
<th>Casting operators</th>
<th>Cast to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(bool)</td>
<td>Booleans</td>
</tr>
<tr>
<td>(int)</td>
<td>Integers</td>
</tr>
<tr>
<td>(float) or (double)</td>
<td>Floating-point number</td>
</tr>
<tr>
<td>(string)</td>
<td>Strings</td>
</tr>
</tbody>
</table>

Table 2.1 Type casting operators

For example, if you want to cast an integer to a double, here is how you would write the code:

```php
$var = (double)242; // $var = 242.0
```

If you cast a double value to an integer, the decimal part will be truncated:

```php
$var = (int)2.8; // $var = 2
```
Type juggling

PHP supports type juggling. This means that PHP is able to perform automatic type conversion of a variable to best fit the situation in which the variable is referenced. Consider the following example:

```php
$var = "242";  //Variable $var is assigned a string value
$a = $var + 1.1; //Addition of $var and a float value
echo $a;       //Output: 243.1
```

Here, the value in the variable `$var` has been automatically converted from type `string` to type `float` during the addition process, resulting in the float number 243.1 being output.

---

Web Reference

The following web reference presents a comprehensive view of type juggling:


---

Activity 2.1

1. The four primitive scalar types are ____________________.
   
   A. true, false, 0, 1
   B. array, object, null, resources
   C. float, integer, string, boolean
   D. mixed, number, char, bool

2. Which of the following is not a valid statement?
   
   A. $a = ON;
   B. $a = -1;
   C. $a = tRuE;
   D. $a = 5;

3. Floating-point numbers are also known as ________________, ________________, or ________________.
4. Write a single quote echo statement to output I’m learning $php now.

5. Arrays and objects are compound data types that can store multiple items of data under a single representation. True/False?

6. List all the type casts that are allowed in PHP.

---

Building expressions

You have now learnt most of the data types that are supported by PHP. Now let us start looking into building simple expressions. In school, you learnt to build mathematics expressions that involved simple arithmetic operations. In programming, an expression is a phrase representing an action in a program. An expression consists of operands and operators. For example, the following simple expression assigns the integer value 8 to the variable $a:

$$a = 8;$$

In this example, the operator is represented by the symbol “=” and the operands are $a$ and 8. Operands and operators are discussed next.

Operands and operators

Operands are the data input into an expression while operators are the symbols used to manipulate the operands in an expression.

Operands

In the examples below, $a$, $sum$, $value1$ and $value2$ are all operands.

$$a++;$$

$$sum = value1 + value2;$$

Arithmetic operators

The basic arithmetic operators are shown in Table 2.2.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>−</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
</tbody>
</table>
Assignment operators

The assignment operators used in PHP are shown in Table 2.3. As you can see, arithmetic operators can be combined with the assignment operator. The combined assignment operators are used when a variable and a second value are to be manipulated and the result of this manipulation is to be assigned to the variable itself.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Assignment</td>
</tr>
<tr>
<td>+=</td>
<td>Addition assignment</td>
</tr>
<tr>
<td>-=</td>
<td>Subtraction assignment</td>
</tr>
<tr>
<td>*=</td>
<td>Multiplication assignment</td>
</tr>
<tr>
<td>/=</td>
<td>Division assignment</td>
</tr>
<tr>
<td>%=</td>
<td>Modulus assignment</td>
</tr>
<tr>
<td>.=</td>
<td>Concatenation assignment</td>
</tr>
</tbody>
</table>

Table 2.3 Assignment operators

For example, the results of both the expressions below will be identical:

```php
$var = $var + 8;
$var += 8;
```

Increment and decrement operators

The increment and decrement operators are listed in Table 2.4. They are used to shorten statements and thus make code easier to read and understand.

<table>
<thead>
<tr>
<th>Operators</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>Increment</td>
</tr>
<tr>
<td>--</td>
<td>Decrement</td>
</tr>
</tbody>
</table>

Table 2.4 Increment and decrement operators
For example, the results of both the expressions below will be identical:

```php
$var += 1;
$var ++;
```

PHP also supports C language pre- and post-increment and decrement operators.

<table>
<thead>
<tr>
<th>Example</th>
<th>Name</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>++$a</td>
<td>Pre-increment</td>
<td>Increments $a by one, then returns $a.</td>
</tr>
<tr>
<td>$a++</td>
<td>Post-increment</td>
<td>Returns $a, then increments $a by one.</td>
</tr>
<tr>
<td>--$a</td>
<td>Pre-decrement</td>
<td>Decrements $a by one, then returns $a.</td>
</tr>
<tr>
<td>$a--</td>
<td>Post-decrement</td>
<td>Returns $a, then decrements $a by one.</td>
</tr>
</tbody>
</table>

Table 2.5 Pre- and post-increment and decrement operators

Examples:

```php
$a = 8;
echo $a++;    // Output: 8
echo $a;      // Output: 9

$a = 8;
echo ++$a;    // Output: 9
echo $a;      // Output: 9

$a = 8;
echo $a--;    // Output: 8
echo $a;      // Output: 7

$a = 8;
echo --$a;    // Output: 7
echo $a;      // Output: 7
```

Web Reference

The following web reference presents a comprehensive view of operators:

Operator precedence

Operator precedence determines the order in which operations in an expression will be performed. PHP follows the standard precedence rules used in mathematics. You will need to refer to the online PHP manual listed in the web reference below to learn about operator precedence in PHP. Operator precedence is the reason why the first statement below can be written as the second statement because the multiplication operator has a higher precedence than the addition operator. The parentheses in the second statement forces the operation \( \$quantity \times 8 \) to be evaluated first. Both statements will produce the same result.

\[
\begin{align*}
\$total &= \$tax + \$quantity \times 8; \\
\$total &= \$tax + (\$quantity \times 8);
\end{align*}
\]

Operator associativity

Operator associativity determines how multiple operators that have the same precedence in an expression are grouped. Associativity can be either from left to right or right to left. For example, the following statements can be written from left to right or right to left and will produce the same result.

\[
\begin{align*}
\$a &= 2 \times 4 \times 2 \times 5; & \text{// Equals 80} \\
\$a &= (((2 \times 4) \times 2) \times 5); & \text{// Equals 80} \\
\$a &= (2 \times (4 \times (2 \times 5))); & \text{// Equals 80}
\end{align*}
\]

However, in the next example, writing from left to right and from right to left will produce different results.

\[
\begin{align*}
\$a &= 2 \times 4 \div 2 \times 5; & \text{// Equals 20} \\
\$a &= (((2 \times 4) \div 2) \times 5); & \text{// Also equals 20} \\
\$a &= (2 \times (4 \div (2 \times 5))); & \text{// Equals 0.8}
\end{align*}
\]

Generally, it is good practice to make groupings explicit by using parentheses to avoid mistakes and improve readability.

Web Reference

To learn about PHP operator precedence and associativity, please refer to the following:

Activity 2.2

Write the value of the variable $y$ after each statement below is executed. Start each case with $x = 5$ and $y = 0$.

1. $y = x + 2;$
2. $y = x - 2;$
3. $y = x * 2;$
4. $y = x / 2;$
5. $y = x \% 2;$
6. $y = x.x;$
7. $y = x++;$
8. $y = x--;$
9. $y += x;$
10. $y -= x;$
11. $y *= x;$
12. $y /= x;$
13. $y %= x;$
14. $y .= x;$
15. $y = ++x;$
16. $y = --x;$
17. $y = x + 2 * 5;$
18. $y = (x + 2) * 5;$
19. $y = (x * (4 * (x * 5)));$
20. $y = (x * (4 / (x * 5)));$
Summary

Data that share a set of characteristics are grouped into a specific data type. The two main categories of data types in PHP are scalar types and compound types.

Scalar data types store a single item of data. The four primitive scalar types are:

1. Boolean — the simplest data type; its value can only be either TRUE or FALSE.
2. Integer — any whole number without a fractional part.
3. Float — numbers with a fractional part; also known as real numbers.

Compound data types store multiple items of data under a single representation. There are two compound data types:

1. Arrays — ordered map that associates values to keys.
2. Objects — an instance of a class.

You can convert values from one data type to another by using type casting. PHP is able to perform automatic type conversion of a value to best fit the situation in which it is referenced. This is known as type juggling.

A variable is a “container” that can hold different values of a particular data type.

An expression is a phrase representing an action in a program. It consists of operands and operators. The types of operators discussed in this section were:

1. Arithmetic operators
2. Assignment operators
3. Incrementing and decrementing operators

Operator precedence and associativity were also briefly discussed in this section.

You also learnt how to build simple expressions with operators and operands. In the next section, you will learn about functions.
Self-test 2.1

1. An integer is a number that contains a fractional part and can be used to represent monetary values, weights and distances. True / False?

2. A string is a sequence of characters that are treated as a contiguous group and delimited by single or double quotes. True / False?

3. The two compound data types in PHP are __________ and __________.
   A. null, resources
   B. arrays, objects
   C. floats, integers
   D. strings, booleans

4. It is often useful to aggregate a series of similar data values together, arranging and referencing them in some specific way. This data structure, known as a/an __________, is formally defined as an indexed collection of data values.
   A. boolean
   B. object
   C. data type
   D. array

5. Describe how data types can be adapted by type juggling in PHP.

6. What is an expression in PHP?

7. The output of the two following PHP statements will be 1. True / False?
   ```php
   $a = 2;
   echo $a--; 
   ```

8. What is operator precedence?
9. What is the value of $\text{var}$ after the following type casting is performed?

$$\text{var} = (\text{int})188.9;$$

A. 188  
B. 189  
C. 190  
D. 188.9

10. In PHP, the names of variables are not case-sensitive. This means that $\text{var}$ and $\text{Var}$ are identical variables. True / False?

Suggested answers to activities

Feedback

Activity 2.1

1. C

2. A

3. floats, doubles, real numbers

4. `echo 'I\'m learning $\text{php} now';`

5. True

6. The casts allowed are:
   a. (int), (integer) — cast to integer
   b. (bool), (boolean) — cast to boolean
   c. (float), (double), (real) — cast to float
   d. (string) — cast to string
   e. (array) — cast to array
   f. (object) — cast to object
Activity 2.2

1. 7
2. 3
3. 10
4. 2.5
5. 1
6. 55
7. 5
8. 5
9. 5
10. −5
11. 0
12. 0
13. 0
14. 05
15. 6
16. 4
17. 15
18. 35
19. 500
20. 0.8
2.2 Working with Functions

Objectives

By the end of this section, you should be able to:

1. Discuss the use of functions for efficient coding.
2. Use PHP built-in functions.
3. Create your own customised functions.

Introduction

The e-commerce sites and web database applications of today rely heavily on features such as mortgage payment calculators and shopping carts. These features provide users with the ability to perform repetitive tasks. Providing a convenient way to perform these repetitive tasks within a section of code is a key concept in modern programming languages like PHP. This is achieved through the use of functions. In this section, you will learn how to use and create functions in PHP.

What are functions?

A function (also known as a method) is a reusable group or block of code that performs a specific task. You can give a PHP function values to work with and it will process the values and output the result. If the task that a function performs needs to be modified, the function is the single point at which you need to make the modification. In other words, it gives you the convenience of a single point of reference. This reduces the possibility of programming errors and enables programs to be maintained easily.

Applying PHP built-in functions

PHP has more than a thousand built-in functions (refer to the online PHP manual for information about these functions). Some functions require you to add the `include()` or `require()` functions before you can use them. These two functions allow you to insert the contents of a PHP file into another PHP file before the server executes it. They are used to create functions, headers, footers and to reuse certain elements on multiple pages. They are identical except that:

1. `include()` generates a warning if it cannot find a file.
2. `require()` generates an error if it cannot find a file.
You can call the function that you want from within a program by simply entering the function name. For example, if you want to calculate the square root of 64, you can call the PHP `sqrt()` function:

```php
$var = sqrt(64); // returns 8
echo $var; // displays 8
```

You can also directly output the function's result without first assigning the value to a variable:

```php
echo sqrt(64); // returns and displays 8
```

You may also output the function result within a larger string by using `printf()`.

```php
printf(“The square root of 64 is %d.”, sqrt(64));
```

### The `printf()` function

The `printf()` function works like the `echo` function but provides more formatting capability. The `printf()` function is used to output a mixture of static text and dynamic information stored within one or more variables. It separates static and dynamic data into two sections and allows you to control how dynamic information is rendered on the screen in terms of position and precision. Its general format is:

```php
int printf(string $format [, mixed $args [, mixed $... ]])
```

Example:

```php
printf(“TCC%d/05: Web Database Application”, 242);
```

Output: TCC242/05: Web Database Application

In the previous example, note how an integer (242) was inserted into a static string. The `%d` is a **type specifier** which represents an integer value placed between “TCC” and “/05”. **Table 2.6** lists the commonly used type specifiers.

<table>
<thead>
<tr>
<th>Type specifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%d</td>
<td>Signed decimal number</td>
</tr>
<tr>
<td>%f</td>
<td>Floating point number</td>
</tr>
<tr>
<td>%s</td>
<td>String</td>
</tr>
<tr>
<td>%o</td>
<td>Octal number</td>
</tr>
<tr>
<td>%x</td>
<td>Lowercase hexadecimal number</td>
</tr>
</tbody>
</table>

**Table 2.6** Type specifiers
Here is another example of how to output two values using `printf()`:

```php
$q = 5;
$costPerPen = 1.98;
$amt = $q * $costPerPen;
printf("The total cost of %d pens is RM%.2f", $q, $amt);
```

Output: The total cost of 5 pens is RM9.90

Note: The specifier `%2f` is used to adjust the number of decimal points or the precision of the floating-point number.

---

### Activity 2.3

1. What are functions and why do you want to use them?

2. ____________ is the PHP built-in function that you can use if you need to raise $x$ to the power of $y$. (Hint: You can find this function in the PHP online manual)

3. What is the output of the following statement?

```php
printf("%d bottles of drinking water cost RM%.2f", 10, 21.20);
```

---

### Creating customised functions

You can create functions that have the logic and calculations that you require. You can even build entire function libraries. The format for a typical function is:

```
Function FunctionName(parameters)
{
    Function body
}
```

As an example, consider the following function, which is called `generateHeader()`:

```php
Function GenerateHeader()
{
    echo "*****";
    echo "Welcome to WOU";
    echo "*****";
}
```
After defining the function, you can call it by using

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

The function will give you the output:

*****Welcome to WOU*****

**Passing parameters by value**

**Parameters** are variables within a function. They are passed to the function by the program that calls the function. The function can then read and change the parameters locally. When calling a function that has more than one parameter, you can separate each parameter with a comma. The following function has two parameters `$var1` and `$var2`:

```php
function PrintWords($var1, $var2)
{
    echo $var1;
    echo $var2;
}
```

The function `PrintWords` can be called with the following statement, which also passes two string parameters “Web” and “Application” to the function:

```php
PrintWords("Web", "Application");
```

The output will be:

Web Application

When you pass parameters to a function in this way, you are passing them by value. In other words, any changes you make to those values within the function are ignored outside the function.
Note

Many functions take multiple parameters. The syntax or the prototype of such a function is as follows:

```php
bool function (mixed $needle, array $haystack [, bool $strict])
```

This function returns a boolean value, TRUE on success (if the $needle was found in the $haystack) or FALSE on failure (if the $needle was not found in the $haystack). The first parameter is named $needle and it can be of many different data types, so we call it mixed. This mixed $needle (what we are looking for) can be either a scalar value (string, integer or float), or an array. The array $haystack (the array we are searching in) is the second parameter. The third optional parameter is named $strict. All optional parameters are placed in square brackets "[]".

Passing parameters by reference

If you need the changes made to a parameter within a function to be reflected outside the function, you can pass the parameter by reference. You pass a parameter by reference by adding an ampersand symbol in front of the parameter. Here is an example:

```php
$amount = 10.00;
$tax = 0.05;
function NewTaxAmount(&$amount, $tax)
{
    // Change the $amount variable
    $amount += $amount * $tax;
    // Add 1% to $tax variable.
    $tax += 0.01;
}
NewTaxAmount($amount, $tax);
echo "Tax is $tax <br />

```

Note that in this example the value of $tax was changed in the function but this change was not reflected outside whereas the value of $amount was also changed in the function and the change was reflected outside the function.
Default parameter values

When writing a function, you sometimes want to have the flexibility not to use all the parameters in a function call. PHP allows you to define default parameter values during function declaration. The following example illustrates this:

```php
function NewTaxAmount($amount, $tax=0.06)
{
    $amount += $amount * $tax;
    echo "Total amount: $amount";
}
```

The default value of $tax, which is 0.06, will only be used if the function call does not include enough parameters. If only one parameter is provided, $tax=0.06 will be applied. The following example shows only one parameter being passed in the call to the function NewTaxAmount:

```php
$amount = 10.00;
NewTaxAmount($amount); //Total amount: 10.6
```

Default parameter values must be placed at the end of the parameter list and are required to be constant expressions. This allows you to call a function like NewTaxAmount() without passing the second parameter. A constant is an identifier for a fixed value. As the name suggests, the value of a constant cannot be changed during the execution of the script.

Returning values from a function

Sometimes you need a function to return the result of a calculation. For example, by passing the loan amount, loan period and interest rate to a function that calculates loan instalments, you would expect it to return the monthly instalment amount.

To return a value from a function, you use the return statement in the function.

```php
function SumNumbers($var1 = 0, $var2 = 0, $var3 = 0)
{
    $total = $var1 + $var2 + $var3;
    return $total;
}
```

//Call the function and print the result
$sum = SumNumbers(23, 24, 25);
echo "The sum of 23 + 24 + 25 = ($sum)";
```

In this case, the value returned by the function will be assigned to the variable $sum.
You may also want the function to return a value to indicate if a parameter is not valid:

```
function Divide($number, $divider)
{
    if ($divider == 0) {
        // Checking to avoid division by 0.
        return false;
    }
    $result = $number/$divider;
    return $result;
}
```

The `return` statement ends the function execution. If there are any statements after the `return` statement, they will not be executed. In this example, if `$divider` is equal to 0, the `return false` statement will be executed and the function stops without proceeding to the calculation of `$result`.

**Returning multiple values from a function**

When you want to return multiple values from a function, you need to return an array set. For example:

```
function MultipleOperation($input1, $input2)
{
    $total = ($input1 + $input2);
    $difference = ($input1 - $input2);
    $ret = array("tot"=>$total, "diff"=>$difference);
    return $ret;
}

$result = MultipleOperation (20, 10);
echo $result ['tot']; // 30
echo $result ['diff']; // 10
```

**Recursive functions**

Recursive functions are functions that call themselves. A recursive function is often very useful in dividing a complex problem into a number of simpler ones. It will iterate through the simpler cases until the problem is resolved. Here is an example of a recursive function being used to calculate a loan payment.
function InterestPayment ($amount, $interest, $years)
{
    if ($years > 0)
    {
        echo $amount;
        echo "<br />";
        $amount = $amount + $amount * $interest;
        return (InterestPayment($amount, $interest, $years-1));
        // Here the function calls itself
    }
    else
        return ($amount);
}

// Apply the function
echo InterestPayment (100, 0.18, 10);
// Output:
// 100 (At year 0)
// 118 (At year 1, 100 + 100 * 18%)
// 139.24 (At year 2, 118 + 118 * 18%)
// 164.3032 (At year 3, 139.24 + 139.24 * 18%)
// 193.877776 ...
// 228.77577568 ...
// 269.9554153024 ...
// 318.54739005683 ...
// 375.88592026706 ...
// 443.54538591513 ...
// 523.383555379866 (Final payment at year 10)

The recursion blocks are:

    InterestPayment($amount, $interest, 10)
    InterestPayment($amount, $interest, 9)
    InterestPayment($amount, $interest, 8)
    ...
    InterestPayment($amount, $interest, 1)

When $years = 0, the function will return $amount. The recursive function will always have a conditional expression that serves to end the recursion. Otherwise, the program will perform infinite recursions. In this example, the conditional expression is if ($years > 0) and the step expression is $years-1. When the condition $years > 0 is no longer true, the recursion will end and the function will return the final $amount.
Web Reference

To learn more about functions, please visit “Functions” at:


Activity 2.4

1. What is the output of the following block of code?

```php
function PrintText($var1, $var2, $var3)
{
    echo $var3;
    echo $var1;
    echo "by ";
    echo $var2;
}
PrintText("parameters ", "value ", "passing ");
```

2. In the function `TotalBill(100)` below, what is the default percentage value of `$service` when the function is called?

```php
function TotalBill($amount, $service=0.12) {...}
```

3. When you need a function to return the result of a calculation, you use the _____________ statement in the function.

4. What is the output of the following recursive function?

```php
function simple ($num)
{
    if ($num > 0)
    {
        echo $num;
        return simple($num-1);
    }
}
simple(5);
```
Summary

A function is a group or block of code that performs a specific task. It can be reused when needed. It reduces the possibility of programming errors and enables easy maintenance. PHP has more than a thousand built-in functions and you can also create your own functions.

Parameters are variables within a function. They are passed to a function by the program that calls it. The function can read and change parameters locally. If you need the changes made to a parameter within a function to be reflected outside the function, you can pass the parameters by reference. PHP also allows you to define default parameter values during function declaration.

To return a value from a function, you use the `return` statement in the function. The `return` statement ends function execution. When you need to return multiple values, you need to return an array set.

Recursive functions are functions that call themselves. They are very useful for dividing complex problems into simpler cases. They iterate through the simpler cases until the problem is resolved.

In the next section, you will learn about control structures.

Self-test 2.2

1. Some PHP built-in functions require you to add the _________ or _________ statement before you can use them.

2. What is the built-in function that you can use to round a number to the nearest integer? (Hint: You can find it in the online PHP manual.)
   A. ceil()
   B. round()
   C. nearest()
   D. floor()

3. Write a function named `PrintOutput` that generates the output “I come I see I conquer” when the function call `PrintOutput("see", "conquer", "I", "come")` is made.
4. What is the output of the following code?

```php
$cost = 10.00;
$tax = 0.05;
function EstimateCost(&$cost, $tax)
{
    $cost = $cost + ($cost * $tax);
    $tax += 4;
}
EstimateCost($cost, $tax);
echo "Tax is ". $tax * 100 ."% <br />
echo "Cost is: $".$cost;
```

5. What is the output of the following code?

```php
function CarSalesTax($price, $tax="")
{
    $total = $price + ($price * $tax);
    echo "Total cost: $$total";
}
CarSalesTax(49800.00);
```

6. What is the output of the following code?

```php
function BikeSalesTax($price, $tax=.05)
{
    return $price + ($price * $tax);
}
$price = 1800;
$total = BikeSalesTax($price);
echo "Total cost: $$total";
```

7. What is the output of the following code?

```php
function Sum ($start, $end, $amount=0)
{
    if ($start > $end)
    { return ($amount);
    }
    else
    { $amount += $start;
        return (Sum($start+1, $end, $amount));
    }
}
echo Sum(1, 5);
Suggested answers to activities

Feedback

Activity 2.3

1. A function (also known as a method) is a group or block of code that performs a specific task. It can be reused when needed. You can give a PHP function values to work with and it will process the values and output the result. If the task that a function performs needs to be modified, the function is the single point at which you need to make the modification. In other words, it gives you the convenience of a single point of reference. You use functions because they reduce the possibility of programming errors and enable easy maintenance.

2. `pow(x, y)`

3. 10 bottles of drinking water cost RM21.20

Activity 2.4

1. passing parameters by value

2. 12 %

3. return

4. 54321
2.3 Control Structures

Objectives

By the end of this section, you should be able to:

1. Explain how to control program flow with control structures.
2. Apply conditional statements.
3. Use various types of loops.

Introduction

When you want to control how your code flows by specifying which statements in an application are to be executed, you will need to use control structures. There are two control structures in PHP: conditional statements and looping statements. The conditional statement controls a program using logic to distinguish between different conditions based on an input value. The common conditional statements are `if-else` and `switch`. A looping statement provides a repeating sequence of instructions that is executed until a specific condition is fulfilled. The more common forms of looping statements in PHP are the `for`, `foreach`, `while` and `do while` statements. You will learn about these statements in this section.

The `if, elseif and else` statements

The `if` statement structure begins with an `if` clause, which is made up of the keyword `if` followed by a true/false expression in parentheses. Subsequent statements in the structure will be located within a block delimited by curly braces `{ }`. The `if` clause says, “If the expression in parentheses is true, execute the statements located within the curly braces. If it is false, ignore these statements.” The format of the `if` statement is as follows:

```php
if (expr)
{
    Statement1;
    Statement2;
}
```

If the curly braces are omitted, only one line will be executed following the `if` statement. This rule also applies to other control structures.

```php
if (expr)
    statement1_only
```
If you want to compare two numbers and then print the result of the comparison, you may use the if statement as in the following example.

```php
$var1 = 10;
$var2 = 20;
if ($var2 > $var1)
{
    echo "var2 is greater than var1";
}
```

If $var2 is greater than $var1, the statement “var2 is greater than var1” will be displayed. In this example, $var2 = 20 and $var1 = 10, thus the conditional expression $var2 > $var1 is true and the statement is displayed.

What happens if $var2 is smaller than $var1? In the previous example, nothing will happen if the conditional expression evaluates to false. If you want your program to be able to do something when the if statement evaluates to false, you can use the elseif clause. The elseif clause says, “If the previous if statement was false, check if this statement is true. Execute it if it is true and skip it if it is not.” Thus, the elseif statement is only evaluated when the preceding if statement is false, otherwise it is skipped. The following example illustrates this:

```php
$var1 = 20;
$var2 = 10;
if ($var2 > $var1)
{
    echo "var2 is greater than var1";
}
elseif ($var2 < $var1)
{
    echo "var2 is smaller than var1";
}
```

In this example, $var2 is indeed smaller than $var1, thus the statement “var2 is smaller than var1” is displayed.

Another form of the if statement uses the else clause. The else clause says, “If all the preceding tests fail, execute this code”. Here is how you would use the if-else statement:

```php
$var1 = 10;
$var2 = 10;
if ($var2 > $var1)
{
    echo "var2 is greater than var1";
}
elseif ($var2 < $var1)
{
```
```php
echo "var2 is smaller than var1";
}
else
{
    echo "var2 is equal to var1";
}
```

In this example, $var2 is equal to $var1, so the else clause is executed and the words "var2 is equal to var1" are displayed.

---

**Web Reference**

To learn more about if-else statements, please visit:


---

**Activity 2.5**

1. The if clause says “If this conditional expression is true, the program will execute the following statements. If it is false, ignore these statements”. True / False?

2. What is the output of the following PHP code?

```php
$d=date("D");
if ($d=="Sat")
    echo "Have a nice weekend!";
elseif ($d=="Sun")
    echo "Have a nice Sunday!";
else
    echo "Have a nice day!";
```
The *switch* statement

In the previous subsection, you learned to use the *if-elseif-else* statement combination. There are certain occasions when you will need to use many *elseif* clauses to test a single variable or expression with different values. The following example illustrates this:

```php
if ($command == "p")
{
    play();
}
elseif ($command == "r")
{
    record();
}
elseif ($command == "a")
{
    pause();
}
else
{
    stop();
}
```

In such a case, there will be a lot of repeated code. The *switch* statement allows you to avoid these repetitions as it compares the value of a certain variable or expression to a list of possible values and executes an action corresponding to a value if a match is found. Here is how to rewrite the preceding example using the *switch* statement:

```php
switch ($command)
{
    case "p":
        play();
        break;
    case "r":
        record();
        break;
    case "a":
        pause();
        break;
    default:
        stop();
        break;
}
```
In this example, the value of the variable \$command (the test variable) is compared against the value listed in each case statement from top to bottom until a match is found. When a match is found, the statement in the matching case will be executed. For example, if \$command equals "p", then play() is executed. The break statement causes the program to exit the switch statement block once a case statement has been executed. If you forget the break statement at the end of a case statement, the program will continue executing all statements until the end of the switch block.

The default statement appears after the last case statement and the statement(s) it contains will be executed if no case statement value matches the test variable. In the previous example, if the test variable \$command does not match any case, then stop() will be executed.

---

**Web Reference**

To learn more about switch statements, please visit:


---

**Activity 2.6**

1. In a switch statement, if you forget the break statement at the end of a case statement, the program will still be able to automatically prevent execution from running into the next case. True / False?

2. Write a switch statement to print out “section 1” when \$var is 1, “section 2” when \$var is 2, and so on up to “section 5” when \$var = 5. Print “No section between 1 to 5” if \$var is other than 1 to 5.
The *for* loop

You have seen conditional statements using *if-else* and *switch*. Another useful mechanism in programming is the looping statement. A looping statement accomplishes a task by repeating a sequence of instructions until a specific condition is satisfied.

Let us first look at the most commonly used loop, which is called the *for* loop. The *for* loop is slightly complex in terms of syntax. The basic syntax of the *for* loop is:

```plaintext
for([initialisation]; [condition]; [step])
{
  body statements
}
```

[Initialisation] is the first expression to run in the loop. It initialises variables or carries out any actions that are needed before the execution of the body statements. The [condition] expression is evaluated before each execution of the body statements. If it is true, the body statements will be executed, if it is false, the *for* loop will exit. [Step] specifies an action, such as variable increment or decrement, to be performed after each execution of the body statements. Consider the *for* loop example below that prints numbers from 0 to 9.

```plaintext
for ($i = 0; $i < 10; $i++) {
  echo $i; // Output: 0123456789
}
```

First, the variable `$i` is initialised to 0. Then the conditional expression, `$i < 10` is evaluated. Since `$i = 0`, which is less than 10, the condition is true and the program will enter the body statement and print the value of `$i`, which is 0 at this point.

Then the loop increments `$i` using `$i++`. Now `$i = 1`. Next, `$i` enters the conditional expression again to be evaluated against `$i < 10`. Since `$i` is still less than 10, the program will execute the body statement and the value of `$i`, which is now 1, will be printed.

Each time the loop runs, `$i` will be incremented and the conditional expression is evaluated. When `$i` finally reaches 10, which means that `$i < 10` is false, the body statement will not be executed and the program will exit the loop.

Thus, the result of the loop execution is: 0123456789
The foreach loop

The foreach loop is a special form of the standard for loop. The foreach loop is used to iterate through an array and to manipulate the values in it. For instance, it is commonly used to print out all the values in an array. You will deal with arrays in Unit 3 but for now consider the following example:

```php
$array = array("Toyota", "Nissan", "Honda", "Perodua");
foreach ($array as $var) {
    echo ($var . "<br />");
}
```

In this example, the as keyword within the foreach parentheses is used to sequentially assign each item in the array to the variable, $var. Each item is then displayed on its own line by the echo statement.

---

**Web Reference**

To learn more about for loop statements, please visit:


---

**Activity 2.7**

1. A for loop consists of:

   ```php
   for ([initialisation]; [condition]; [step])
   {
       body statements
   }
   ```

   What is the typical execution sequence when the loop starts running?

   A. [initialisation] → [body statements] → [condition] → [step]
   B. [initialisation] → [body statements] → [step] → [condition]
   C. [initialisation] → [condition] → [step] → [body statements]
   D. [initialisation] → [condition] → [body statements] → [step]

2. Write a for loop program to print out 54321.
The while and do-while loops

The while loop

The while loop is the simplest type of loop in PHP. It tells PHP to repeatedly execute the body statement(s) as long as the conditional expression is true. The syntax of the while loop is shown below:

```
[initialisation]   //optional
while([condition])
{
    body statements
    [steps]       //optional
}
```

The conditional expression [condition] is checked at the beginning of each iteration of the loop. This means that the body statements will not be run even once if the conditional expression is evaluated to false at the beginning of the first iteration. The example below shows an alternative way of writing the for loop that prints 0123456789 using the while statement.

```
$i = 0;
while ($i < 10)
{
    echo $i++;        // Output: 0123456789
}
```

Do-while loops

The do-while loop has a similar syntax and purpose as the while loop except that its conditional expression is checked at the end of each iteration instead of at the beginning. This means that the body statement(s) will execute at least once. The syntax of the do-while loop is shown below:

```
[initialisation]   //optional
do
{
    body statements
    [steps]       //optional
} while([condition]);
```

The code block below will print out “Hi” even though $c is initialised to 6, which makes the conditional expression $c < 5 false because the conditional expression is only evaluated at the end of the loop.
$c = 6;
do
{
    echo 'Hi';
} while ($c < 5);

So, when do you use the do-while loop? Since the body statement(s) will always execute once regardless of the status of the conditional expression, you can use it to run a pre-checking procedure before the loop is executed. For example, you may use a do-while loop to perform file existence checking before you start to work on a file or you may want to get confirmation input from a user before running the loop.

---

Web Reference

To learn more about do-while loop statements, please visit:


---

Break and continue

You have already seen how the break statement was used in the switch block. The break statement is also useful when you need to end the execution of for, foreach, while, or do-while loops. The following is an example of the application of the break statement in a while loop. Note that the conditional expression for the while loop in this example is set to true to make it an infinite loop on purpose. Then the break statement is used to end the execution of the loop when $i equals 10.

$i = 0;
while (true)
{
    if ($i >= 10)
        break;
    echo $i++;
    // Output: 0123456789
}

The continue statement is used within looping structures to skip the rest of the current looping statement. It forces the program to continue execution at the conditional expression and begin the next iteration of the loop. In the example below, when $i is equal to 2, the continue statement will be executed and the echo $i statement will be skipped. Thus the number 2 will not be printed. Execution continues at $i++ where $i will become 3 and the loop then continues.
for ($i = 0; $i < 10; $i++) {
    if ($i == 2)
        continue;
    echo $i; // Output: 013456789
}

Activity 2.8

1. What is the difference between the `while` loop and the `do-while` loop?

2. What happens when a program encounters the `break` statement and what happens when the program encounters the `continue` statement?

Summary

There are two main control structures in PHP: the conditional statement and the looping statement. The conditional statement controls the program by using logic to distinguish between different conditions based on an input value. The `if-else` statement is used to execute some task if a condition is true and another task if the condition is false. The `switch` statement allows you to compare the value of a certain variable to a list of possible values and execute an action when a match is found.

The looping statement repeats a sequence of instructions until a specific condition is fulfilled. The `for` loop is used to execute a block of code a specified number of times or while a specified condition is true. Its basic syntax includes initialisation, condition and step. The `foreach` loop is a special form of the standard `for` loop. The common use of the `foreach` loop is to iterate through an array. The `while` loop repeatedly executes statements as long as its conditional expression is true. The `do-while` loop has a similar syntax and purpose as the `while` loop except that the conditional expression is checked at the end of each iteration instead of at the beginning.

To summarise, you should now be familiar with:

- Conditional statements like `if-else` and `switch`.
- Looping statements like `for`, `foreach`, `while` and `do-while`.
- The use of the `break` and `continue` statements.
Self-test 2.3

1. The conditional statement repeats a sequence of instructions until a specified condition is fulfilled to accomplish a programming task. True / False?

2. Consider the following PHP code:

```php
$secretNumber = 555;
$guessNumber = 123; // change variable value here
if ($guessNumber == $secretNumber) {
    echo “Congratulations!”;
}
elseif (abs ($guessNumber - $secretNumber) < 10) {
    echo “You’re getting close!”;
}
else {
    echo “Sorry!”;
}
```

What is the output when

a. $guessNumber = 500;

b. $guessNumber = 550;

c. $guessNumber = 555;

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

```php
$category = “weather”; switch ($category) {
    case “news”:
        echo “<p>What’s happening around the world</p>”;
    case “weather”:
        echo “<p>Your weekly forecast</p>”;
    case “sports”:
        echo “<p>Latest sports highlights</p>”;
    default:
        echo “<p>Welcome to my Web site</p>”;
}
4. Print out the following lines with a for loop:

1 km = 0.621400 miles
2 km = 1.242800 miles
3 km = 1.864200 miles
4 km = 2.485600 miles
5 km = 3.107000 miles

5. What is the output of the following PHP code?

```php
$arr = array("HTML", "XML", "XHTML", "CSS", "JavaScript");
foreach ($arr as $a) {
    if ($a == "XML")
        continue;
    echo ($a . "<br /> ");
}
```

6. Print out the following lines with a while loop.

1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16

7. What is the output of the following PHP code?

```php
$count = 11;
do {
    printf("%d squared = %d <br /> ", $count, pow($count, 2));
} while ($count < 10);
```

8. What is the output of the following PHP code?

```php
$gal = 1;
for (;;) {
    if ($gal > 5)
        break;
    printf("%d gallon = %.2f liters <br /> ", $gal, $gal*3.7854);
    $gal++;}
```
Suggested answers to activities

Feedback

Activity 2.5

1. True

2. The result varies depending on what the current day is.

   Mon-Fri: Have a nice day!
   Sat: Have a nice weekend!
   Sun: Have a nice Sunday!

Activity 2.6

1. False

2. $var = 0; \ // assign different values here
   switch ($var)
   {
     case 1:
       echo "section 1";
       break;
     case 2:
       echo "section 2";
       break;
     case 3:
       echo "section 3";
       break;
     case 4:
       echo "section 4";
       break;
     case 5:
       echo "section 5";
       break;
     default:
       echo "No section between 1 to 5";
   }
Activity 2.7

1. D

2. for ($i=5; $i>=1; $i--)
   {
       echo $i;
   }

Activity 2.8

1. The difference is that the code embedded within a while statement could possibly never be executed, whereas the code embedded within a do-while statement will always execute at least once.

2. A break statement will immediately end the execution of a control structure block. The continue statement causes execution of the current loop iteration to end and recommence at the beginning of the loop.
Summary of Unit 2

Summary

In the first section of this unit, you learnt about the two main categories of data types found in PHP, which are the scalar and compound types. You were also introduced to variables, type conversion and type juggling. Additionally, you learnt to build simple expressions with operands and operators.

In the second section, you learnt how to apply PHP built-in functions and create customised functions. You also learnt to pass parameters by value and by reference to functions. You saw how functions return values and how recursive functions are used.

In the final section, you studied control structures that involve conditional statements and looping statements. You learnt about the if-else statement, switch statement, for loop, foreach loop, while loop and do-while loop.

You are now well on your way towards developing a strong foundation for building web database applications! In the next unit, you will learn how to manipulate strings and arrays and you will also begin to work with files.
1. Create a PHP file in the ‘/var/www/Unit2’ directory (or ‘<Apache Directory>\htdocs\Unit2’ in Windows) and name it ‘MySite05.php’. In this file, you are to write code that will display a web page with a message based on the current time of day. For example, if it is in the morning, have the site display “Good Morning.”.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</td>
<td>My Site 05</td>
</tr>
<tr>
<td>Greeting Message</td>
<td>Use <code>&lt;h1&gt;</code> markup, <code>date_default_timezone_set('Asia/Kuala_Lumpur');</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use <code>date(&quot;G&quot;)</code> to get the hour value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:00 – 11:00: Good Morning,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12:00 – 18:00: Good Afternoon,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>19:00 – 4:00: Good Evening,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>You may use either if-else or switch statement.</td>
<td></td>
</tr>
</tbody>
</table>

Create the page shown in **Figure 2.1**.

**Figure 2.1** Greeting message
2. In this exercise, you will modify the My Site web page from question 1. Duplicate the PHP file from question 1 in the ‘/var/www/Unit2’ directory (or ‘<Apache Directory>\htdocs\Unit2’ in Windows) and name it ‘MySite06.php’.

   a. Save the greeting message into a new file called ‘header.php’.

   b. Call the header file from ‘MySite06.php’ with include “header.php”.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</td>
<td>My Site 06</td>
</tr>
<tr>
<td>Use the include function</td>
<td>include &quot;header.php&quot;</td>
<td>{the greeting message}</td>
</tr>
</tbody>
</table>

3. In this exercise, you will modify the My Site web page from question 2. Duplicate the PHP file from question 2 in the ‘/var/www/Unit2’ directory (or ‘<Apache Directory>\htdocs\Unit2’ in Windows) and name it ‘MySite07.php’.

   a. Print the following lines with a while loop.

   Espresso Roast Grade 1 earns 1 loyalty points
   Espresso Roast Grade 2 earns 4 loyalty points
   Espresso Roast Grade 3 earns 9 loyalty points
   Espresso Roast Grade 4 earns 16 loyalty points
   Espresso Roast Grade 5 earns 25 loyalty points

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</td>
<td>My Site 07</td>
</tr>
<tr>
<td>PHP current time</td>
<td>Use a while loop and the pow() function</td>
<td></td>
</tr>
</tbody>
</table>

Compare your results with Figure 2.2.
4. In this exercise, you will modify the My Site web page from question 3. Duplicate the PHP file from question 3 in the '/var/www/Unit2' directory (or '<Apache Directory>/htdocs/Unit2' in Windows) and name it 'MySite08.php'.

   a. Print a table using the following information:

   ```html
   <table border="1" width="50%">
   <tr>
   <th>Coffee</th>
   <th>Cost</th>
   <th>Quantity</th>
   <th>Sub Total</th>
   </tr>
   <?php ...
   print table here
   ?>
   ```

   b. Use a `for` loop to print out 5 rows of items as shown below in the PHP tag.

<table>
<thead>
<tr>
<th>Coffee</th>
<th>Cost</th>
<th>Quantity</th>
<th>Sub total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Espresso Roast Grade 1</td>
<td>RM10.95</td>
<td>5</td>
<td>RM54.75</td>
</tr>
<tr>
<td>Espresso Roast Grade 2</td>
<td>RM21.90</td>
<td>4</td>
<td>RM87.60</td>
</tr>
<tr>
<td>Espresso Roast Grade 3</td>
<td>RM32.85</td>
<td>3</td>
<td>RM98.55</td>
</tr>
<tr>
<td>Espresso Roast Grade 4</td>
<td>RM43.80</td>
<td>2</td>
<td>RM87.60</td>
</tr>
<tr>
<td>Espresso Roast Grade 5</td>
<td>RM54.75</td>
<td>1</td>
<td>RM54.75</td>
</tr>
</tbody>
</table>
c. Use the following simulation data to assist you.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>My Site 08</td>
</tr>
<tr>
<td>Coffee</td>
<td>Espresso Roast Grade with the number equivalent to the row number.</td>
<td>Row no. 1 means Espresso Roast Grade 1</td>
</tr>
<tr>
<td>Cost</td>
<td>Use $\text{pricePerOz} = 10.95$; Each grade increment will increase the cost, which is $\text{pricePerOz}$ multiplied by grade number.</td>
<td>Espresso Roast Grade 2 = 2 x $\text{pricePerOz}$ Espresso Roast Grade 5 = 5 x $\text{pricePerOz}$ Hint: Use <code>printf()</code> with floating 2 decimal points</td>
</tr>
<tr>
<td>Quantity</td>
<td>Quantity is as shown in the table above</td>
<td>Espresso Roast Grade 1 → Quantity: 5 Espresso Roast Grade 5 → Quantity: 1</td>
</tr>
<tr>
<td>Sub Total</td>
<td>Sub Total is the cost multiplied by quantity</td>
<td>Hint: Use <code>printf()</code> with 2 decimal points</td>
</tr>
</tbody>
</table>

5. In this exercise, you will modify the My Site web page from question 4. Duplicate the PHP file from question 4 in the ‘/var/www/Unit2’ directory (or ‘<Apache Directory>\htdocs\Unit2’ in Windows) and name it ‘MySite09.php’.

a. Display the grand total of the cost of all the items.

b. Display the total loyalty points earned based on the chart in question 3.

c. Use the following simulation data to assist you.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>My Site 09</td>
</tr>
<tr>
<td>Grand total</td>
<td>The grand total is the total of the sub totals.</td>
<td>Hint: Use <code>printf()</code> with 2 decimal points.</td>
</tr>
<tr>
<td>Total points</td>
<td>Total points are equivalent to quantity multiplied by points earned on each grade of coffee.</td>
<td>Points for Espresso Roast Grade 2 = 4 x 4 Points for Espresso Roast Grade 5 = 1 x 25</td>
</tr>
</tbody>
</table>
See Figure 2.3.

**Figure 2.3** Displaying the grand total and total points earned

d. If you are required to display only Espresso Roast Grade 4 and above, how would you modify your `for` loop? (Hint: Use the `if` and `continue` statements.)
Feedback

Self-test 2.1

1. False
2. True
3. B
4. D
5. Because of PHP’s lax attitude toward type definitions, variables are sometimes automatically cast to best fit the circumstances in which they are referenced. PHP’s automatic type conversion can be illustrated by the addition operation. If either operand in an addition operation is a float, then both operands are evaluated as floats, and the result will be a float. Otherwise, the operands will be interpreted as integers, and the result will also be an integer.

6. An expression is a phrase representing a particular action in a program. All expressions consist of at least one operand and one or more operators.

7. False

8. Operator precedence determines the order in which operators evaluate the operands surrounding them. PHP follows the standard precedence rules used in mathematics.

9. A

10. False

Self-test 2.2

1. include(), require()

2. B
3. function PrintOutput($var1, $var2, $var3, $var4)
{
    echo $var3 . "$.var4." ";
    echo $var3 . "$var1." ";
    echo $var3 . "$var2." ";
}

4. Tax is 5%; Cost is 10.5

5. Total cost: $49800

6. Total cost: $1890

7. 15

Self-test 2.3

1. False

2. a. Sorry!
   
   b. You're getting close!
   
   c. Congratulations!

3. Output:

   Your weekly forecast
   Latest sports highlights
   Welcome to my Web site

4. Code:

   for ($i = 1; $i <= 5; $i++)
   {
       printf("%d km = %f miles <br />", $i, $i*0.62140);
   }
5. Output:

HTML
XHTML
CSS
JavaScript

6. Code:

```php
$count = 1;
while ($count < 5) {
    printf("%d squared = %d <br />", $count, pow($count, 2));
    $count++;
}
```

Output: 11 squared = 121

8. Output:

1 gallon = 3.79 liters
2 gallon = 7.57 liters
3 gallon = 11.36 liters
4 gallon = 15.14 liters
5 gallon = 18.93 liters
Suggested Answers to Unit Practice Exercises

Feedback

1. MySite05.php

<HTML>
<HEAD>
<TITLE>My Site 05</TITLE>
</HEAD>
<BODY>
<h1>
<?php
// Either one is acceptable
// if-else statement
date_default_timezone_set('Asia/Kuala_Lumpur');
if ((date("G") >= 5) AND (date("G") <= 11 ))
  echo "Good Morning";
if ((date("G") >=12) AND (date("G") <=18))
  echo "Good Afternoon";
if ((date("G") >= 19) OR (date("G") <= 4))
  echo "Good Evening";

// switch statement
switch (date("G"))
{
  case 5:
  case 6:
  case 7:
  case 8:
  case 9:
  case 10:
  case 11:
    echo "Good Morning";
    break;
  case 12:
  case 13:
  case 14:
  case 15:
  case 16:
  case 17:
  case 18:
    echo "Good Afternoon";
    break;

</HTML>
2. MySite06.php

```php
<?php include "header.php"; ?>
```

header.php

```php
<?php
    date_default_timezone_set('Asia/Kuala_Lumpur');
    if ((date("G") >=5) AND (date("G") <= 11 ))
        echo "Good Morning";
    if ((date("G") >=12) AND (date("G") <=18))
        echo "Good Afternoon";
    if ((date("G") >= 19) OR (date("G") <= 4))
        echo "Good Evening";
?>
```

3. MySite07.php

```php
<?php include "header.php";

$items = 5;
$count = 1;
while ($count <= $items)
{
    printf("Espresso Roast Grade %d earns %d loyalty points \n", $count, pow($count, 2));
    $count++;
}

?>
```

4. MySite08.php

```php
<?php include "header.php";

$items = 5;
$count = 1;
while ($count <= $items)
{
    printf("Espresso Roast Grade %d earns %d loyalty points \n", $count, pow($count, 2));
    $count++;
}

<table border="1" width="50%">
<tr>
    <th>Coffee</th>
    <th>Cost</th>
    <th>Quantity</th>
    <th>Sub Total</th>
</tr>
<?php

$pricePerOz = 10.95;
```
5. MySite09.php

<HTML>
 <HEAD>
  <TITLE>My Site 09</TITLE>
 </HEAD>
 <BODY>
 <?php include "header.php";
 $items = 5;
 $count = 1;
 while ($count <= $items)
 {
  printf("Espresso Roast Grade %d earns %d loyalty points \n", $count, pow($count, 2));
  $count++;
 }
 ?>
 <table border="1" width="50%">
  <tr>
   <th>Coffee</th>
   <th>Cost</th>
   <th>Quantity</th>
   <th>Sub Total</th>
  </tr>
  <?php
  $pricePerOz = 10.95;
  $grandTotal = 0;
  $totalPoints = 0;
  for($i = 1; $i <= $items; $i++)
  {
   echo "<tr>";
   echo "<td>Espresso Roast Grade \$i . \</td>";
   $price = $pricePerOz * $i;
   printf ("<td>RM%.2f</td>", $price);
   $q = $items - $i + 1;
   $p = $q * pow($i, 2);
   echo "<td>\$q .\</td>";
   $subTotal = $price * $q;
   printf ("<td>RM%.2f</td>", $subTotal);
   echo "</tr>";
  }
  echo "</table>";
?>
</BODY>
</HTML>
for($i = 1; $i <= $items; $i++)
{
    //Part d. Uncomment below to display Espresso
    Roast Grade 4 and above
    //if ($i <= 3)
    //continue;
    echo "<tr>";
    echo "<td>Espresso Roast Grade "$i."</td>"
    $price = $pricePerOz * $i;
    printf("<td>RM%.2f</td>", $price);
    $q = $items - $i + 1;
    $p = $q * pow($i, 2);
    $totalPoints += $p;
    echo "<td>".$q."</td>";
    $subTotal = $price * $q;
    printf("<td>RM%.2f</td>", $subTotal);
    $grandTotal += $subTotal;
    echo "</tr>";
}
echo "</table>";

printf("<p>Grand total: <strong>RM%.2f</strong></p>" , $grandTotal);
printf("Total points earned: %d", $totalPoints);
?>
</BODY>
</HTML>
References


Wikibooks (2010) *PHP programming*, http://en.wikibooks.org/wiki/PHP, used under a Creative Commons Attribution-ShareAlike licence: http://creativecommons.org/licenses/by-sa/3.0/
WAWASAN OPEN UNIVERSITY

TCC 242/05  Web Database Application
Glossary

Arithmetic operators
The basic arithmetic operators in PHP are the addition, subtraction, multiplication, division, modulus and concatenation operators.

Array
An ordered map that associates values to keys. It is an indexed collection of data values.

Assignment operator
An operator that is used to assign a value to a variable.

Boolean
The simplest data type. Its value can only be either TRUE or FALSE.

Break statement
Used to end execution of the current switch, for, foreach, while or do-while structure.

Casting
Converting values from one data type into another data type. In PHP, this can be done by placing the intended type in front of the variable that you intend to convert.

Compound types
Compound data types store multiple items of data under a single representation. In PHP, arrays and objects are compound data types.

Conditional statement
Controls a program by using logic to distinguish between different conditions based on an input value.

Continue statement
Used within looping structures to skip the rest of the current iteration. It forces execution to continue at the evaluation of the conditional expression and thus begins the next iteration.

Data type
A group of data values that share a set of characteristics. Data types in PHP are represented by names like boolean, integer and array.

Decrementing operator
An operator that decrements a value by one, such as “−−”. It is used to shorten a statement and thus make code easier to read and understand.
Do-while loop

The do-while loop has a similar syntax and purpose as the while loop except that its conditional expression is checked at the end of each iteration instead of at the beginning.

Expression

An expression is a phrase representing an action in a program. It consists of operands and operators.

Floating-point number

A number that has a fractional part; also known as a real number.

Function

A function (also known as a method) is a reusable group or block of code that performs a specific task.

For-loop

The for loop is a statement that enables code to be executed repeatedly.

Foreach-loop

The foreach loop is a special form of the standard for loop. The foreach loop is used to iterate through the values in an array.

If-else statement

A conditional statement that is used to select one of several blocks of code to be executed.

Incrementing operator

An operator that increments a value by one, such as “++”. It is used to shorten a statement and thus make code easier to read and understand.

Integer

A number without a fractional part.

Looping statement

Repeats a sequence of instructions until a specific condition is fulfilled.

Object

The essential component in object-oriented programming. Objects refer to an instance of a class.

Operands

The inputs to expressions.

Operator associativity

Determines how multiple operators that have the same precedence are grouped.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator precedence</td>
<td>Determines the order in which operations in an expression will be performed.</td>
</tr>
<tr>
<td>Operators</td>
<td>Any symbol used to manipulate data in an expression.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Variables within a function.</td>
</tr>
<tr>
<td>Recursive functions</td>
<td>Functions that call themselves.</td>
</tr>
<tr>
<td>Scalar types</td>
<td>Scalar data types store single items of data. The four primitive scalar types in PHP are boolean, integer, float and string.</td>
</tr>
<tr>
<td>String</td>
<td>A series of characters. Strings are delimited by single or double quotes.</td>
</tr>
<tr>
<td>Switch statement</td>
<td>A conditional statement that is used to select one of a number of blocks of code for execution.</td>
</tr>
<tr>
<td>Type juggling</td>
<td>Automatic type conversion of a value to best fit the situation in which it is referenced.</td>
</tr>
<tr>
<td>Variable</td>
<td>A “container” that can hold different values of a data type.</td>
</tr>
<tr>
<td>While-loop</td>
<td>A loop that repeatedly executes statements within its structure as long as the conditional expression is true.</td>
</tr>
</tbody>
</table>
Unit 3

TCC 242/05
Web Database Application

Manipulating Strings, Arrays and Working with Files
Wawasan Open University is Malaysia's first private not-for-profit tertiary institution dedicated to adult learners. It is funded by the Wawasan Education Foundation, a tax-exempt entity established by the Malaysian People’s Movement Party (Gerakan) and supported by the Yeap Chor Ee Charitable and Endowment Trusts, other charities, corporations, members of the public and occasional grants from the Government of Malaysia.

The course material development of the university is funded by Yeap Chor Ee Charitable and Endowment Trusts.

© 2011 Wawasan Open University. Except where otherwise noted, this work is licensed under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

This course material was published to support the learning of students registered with Wawasan Open University. Wawasan Open University does not grant any degree, certification or credits based solely on your completion of this course material.
## Contents

### Unit 3  Manipulating Strings, Arrays and Working with Files

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit overview</td>
<td>1</td>
</tr>
<tr>
<td>Unit objectives</td>
<td>1</td>
</tr>
<tr>
<td><strong>3.1 Strings</strong></td>
<td>3</td>
</tr>
<tr>
<td>Objectives</td>
<td>3</td>
</tr>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>String length</td>
<td>3</td>
</tr>
<tr>
<td>Trimming whitespace or other characters</td>
<td>4</td>
</tr>
<tr>
<td>String padding</td>
<td>5</td>
</tr>
<tr>
<td>Comparing two strings</td>
<td>7</td>
</tr>
<tr>
<td>Extracting substrings</td>
<td>8</td>
</tr>
<tr>
<td>Finding the position of a substring</td>
<td>9</td>
</tr>
<tr>
<td>Replacing substrings</td>
<td>10</td>
</tr>
<tr>
<td>Suggested answers to activities</td>
<td>15</td>
</tr>
<tr>
<td><strong>3.2 Arrays</strong></td>
<td>17</td>
</tr>
<tr>
<td>Objectives</td>
<td>17</td>
</tr>
<tr>
<td>Introduction</td>
<td>17</td>
</tr>
<tr>
<td>Creating arrays</td>
<td>18</td>
</tr>
<tr>
<td>Outputting arrays</td>
<td>19</td>
</tr>
<tr>
<td>Adding and removing array elements</td>
<td>19</td>
</tr>
<tr>
<td>Extracting items from arrays</td>
<td>21</td>
</tr>
<tr>
<td>Determining array size</td>
<td>22</td>
</tr>
</tbody>
</table>
Array traversal 23
Sorting arrays 24
Multidimensional arrays 26
Suggested answers to activities 34

3.3 Files 37

Objectives 37
Introduction 37
Opening a file 37
Closing a file 38
Reading a file 39
Writing a file 41
Deleting a file 43
Reading directory contents 43
Checking file or directory existence 44
Suggested answers to activities 47

Summary of Unit 3 49
Unit practice exercises 51
Suggested answers to self-tests 57
Suggested answers to unit practice exercises 63
References 69
Glossary 71
Unit Overview

This unit discusses techniques for manipulating strings, arrays and files using PHP. The first section discusses how string library functions are used. It also explains how you can modify strings to achieve the results or formatting that you require. The second section describes the use of arrays to deal with collections of data. It also shows you some common applications of multidimensional arrays. The third section discusses the processes involved in the creation as well as the removal of files. Functions for dealing with the file system are also covered in this section.

Unit Objectives

By the end of this unit, you should be able to:

1. Describe and use PHP string library functions.
2. Describe and use PHP array library functions.
3. Describe and use PHP functions for working with files and directories.
3.1 Strings

Objectives

By the end of this section, you should be able to:

1. Describe basic PHP string functions.
2. Format and compare strings.
3. Find, extract and replace strings.
4. Apply various string functions to solve programming problems.

Introduction

In Unit 2, strings were defined as a series of characters. Strings are one of the most commonly used data types in PHP scripts. In this section, you will learn how to manipulate strings. PHP provides a large library of string functions to help you manage and manipulate strings. You will be focusing on some of the more useful PHP string functions in this section.

String length

Obtaining the length of a string is a very common task when writing PHP scripts. The length of a string can be determined with the `strlen()` function. It returns the length of a string as an integer or 0 if the string is empty. The syntax of the function is:

```php
int strlen ( string $string )
```

The following example counts and verifies the length of a given username string. Note that the comment line at the end of the code listing shows the output:

```php
$username = strlen("alice");
if (strlen($username) < 6)
    echo "Error: Username must be at least six characters";
else
    echo "Username is valid!";
// Output: Error: Username must be at least six characters
```

In this case, the output will be the error message since the string “alice” is only five characters long whereas the username should be at least six characters in length.
Trimming whitespace or other characters

When your PHP script gets string input from a user through a web form, it may consist of whitespace characters before or after the text. Whitespace characters are spaces or tabs, which the user may sometimes accidentally include in a web form. It is important that you remove or trim whitespaces to avoid errors during string manipulation. PHP provides three functions to trim leading and trailing whitespaces, and whitespaces on both sides of a string. The syntax of each of the functions is shown below:

```php
string ltrim ( string $str [, string $charlist ] )
string rtrim ( string $str [, string $charlist ] )
string trim ( string $str [, string $charlist ] )
```

If the optional second parameter ($charlist) is omitted, these functions remove the following whitespaces:

1. space (" ")
2. tab (\t)
3. newline (\n)
4. carriage return (\r)
5. NULL (\x00)

The examples below illustrate the use of these functions:

```php
$var = trim(" Hypertext Preprocessor \n"); //Result: “Hypertext Preprocessor"

$var = ltrim(" Hypertext Preprocessor \n"); //Result: “Hypertext Preprocessor \n"

$var = rtrim(" Hypertext Preprocessor \n"); //Result: “ Hypertext Preprocessor"
```

The optional second parameter, $charlist, allows you to specify characters to trim. Simply list all characters that you want to trim in this parameter. With the symbol ". . .", you can specify a range of characters or digits to be trimmed. For example:

```php
$var = trim("01 FEB 2012", "0..9"); // Trims digits and spaces
echo $var; // Prints “FEB”
```
String padding

Padding is used for spacing out strings to reduce clutter and enhance readability. PHP has a handy function called **str_pad()** that performs string padding. The syntax of the function is:

```php
string str_pad ( string $input , int $pad_length [, string $pad_string [, int $pad_type ]] )
```

The function increases the number of characters of an input string, `$input`, to a new length of `$pad_length` characters. If the optional parameter `$pad_string` is not supplied, the input is padded with blank spaces. You can use `$pad_type` to pad a string on:

1. the left (`$pad_type = STR_PAD_LEFT`)
2. the right (`$pad_type = STR_PAD_RIGHT`)
3. or both sides (`$pad_type = STR_PAD_BOTH`)

By default, the string will be padded on the right.

The following example shows the use of `str_pad()` without optional parameters:

```php
echo str_pad("web", 6);
// Prints "web " followed by three spaces
```

The following example shows the use of `str_pad()` with optional parameters. The function is enclosed within the HTML `<pre>` tag. Text in the `<pre>` tag is displayed in a fixed-width font with the spaces and line breaks being preserved. These tags enable us to see the effects of string padding in a web browser.

```php
echo "<pre>"; // The <pre> tag defines preformatted text.
echo str_pad("Topics 3.1", 50, " ", STR_PAD_BOTH) . "<br>";
echo str_pad("Intro", 30, ".") . str_pad("Page 1", 20, ".", STR_PAD_LEFT) . "<br>";
echo str_pad("String Length", 30, ".") . str_pad("Page 3", 20, ".", STR_PAD_LEFT) . "<br>";
echo str_pad("Trim whitespace", 30, ".") . str_pad("Page 6", 20, ".", STR_PAD_LEFT) . "<br>";
echo str_pad("String Padding", 30, ".") . str_pad("Page 8", 20, ".", STR_PAD_LEFT) . "<br>";
echo "</pre>";
```
The output of this script will be:

```
Topics 3.1
Intro.......................................Page 1
String Length...............................Page 3
Trim whitespace.............................Page 6
String Padding..............................Page 8
```

### Activity 3.1

1. The three optional `$pad_type` parameters used in the `str_pad()` function are

   - A. STR_PAD_TOP, STR_PAD_BOTTOM, STR_PAD_BOTH
   - B. STR_PAD_IN, STR_PAD_OUT, STR_PAD_BOTH
   - C. STR_PAD_LTRIM, STR_PAD_RTRIM, STR_PAD_BOTH
   - D. STR_PAD_LEFT, STR_PAD_RIGHT, STR_PAD_BOTH

2. Which of the following is not a valid whitespace?

   - A. NULL (\x00)
   - B. white (#FF0000)
   - C. newline (\n)
   - D. carriage return (\r)

3. PHP provides three functions to trim ________ and ________ whitespaces, and whitespaces on __________ of a string.

4. Write PHP code to output the length of the string “I’m learning php now”.

5. Write PHP code that trims the trailing whitespaces of the string “Log Report \t” and shows the results of the trimming.

6. What is the output of the following code?

   ```php
   echo "<pre>";
   echo str_pad("Vegetables", 20)." are good for health";
   echo "</pre>";
   ```
Comparing two strings

String comparison is used to determine whether two strings are identical. Although you can compare two strings using the equality operator (==), this may cause unexpected or unsafe results for strings with binary content or multi-byte encoding. There are two basic string comparison functions. The syntax of each of the functions is:

```php
int strcmp ( string $str1 , string $str2 )
int strncmp ( string $str1 , string $str2 , int $len )
```

Note:

You may have heard about binary content being unsafe when you read about functions in PHP. What does binary safe mean? The term binary safe is related to the handling of binary data. Binary data is encoded as a string of 1s and 0s. If you use Microsoft Word to open a Word document (binary content), then you are actually handling the document in a binary safe way. Binary data need to be handled differently than other data types to avoid the data becoming corrupt.

The `strcmp()` function performs case-sensitive comparison of two strings. It returns

1.  $< 0$ if $str1$ is less than $str2$
2.  $> 0$ if $str1$ is greater than $str2$
3.  $0$ if they are equal

The following examples show you how to use the function:

```php
echo strcmp("a", "b"); // Returns -1
echo strcmp("b", "a"); // Returns 1
echo strcmp("linux", "linux"); // Returns 0
echo strcmp("linux", "Linux"); // Returns 1
```

The `strncmp()` function takes a third parameter, $len$, which restricts the comparison to a specified number of characters. For example, to compare the first seven characters of two strings, you will use:

```php
echo strncmp("windowsxp", "windows7", 7); // Returns 0
```

The result will be different when you change the third parameter to eight:

```php
echo strncmp("windowsxp", "windows7", 8); // Returns 1
```
The `strncmp()` function performs case-sensitive comparisons. If you want to compare strings with case-insensitivity, you can use `strcasecmp()` or `strncasecmp()`:

```c
int strcasecmp ( string $str1 , string $str2 )
int strncasecmp ( string $str1 , string $str2 , int $len )
```

The following examples show the use of `strcasecmp()` and `strncasecmp()`:

```php
echo strcasecmp("linux", "Linux");    // Returns 0
echo strncasecmp("windowsxp", "WINDOWS7", 7); // Returns 0
```

### Extracting substrings

The `substr()` function allows you to extract part of a string or “substring” from a source string with a predefined start position (`$start`). The syntax of the function is:

```php
string substr ( string $source , int $start [, int $length ] )
```

Consider the following string:

```php
$plane = "Airbus-A380";
```

1. If `$start` is positive, the returned string will start at the specified position of the source string, counting from zero. For example:

```php
echo substr($plane, 3);    // Returns “bus-A380”
```

2. If `$start` is negative, the returned string will start at the specified position counting in reverse from the end of the source string. For example:

```php
echo substr($plane, -4);   // Returns “A380”
```

If the optional `$length` parameter is specified, the number of characters specified by this parameter is returned.
Consider the same source string used in the previous example:

$plane = “Airbus-A380”;

1. If $length is positive, the string returned will contain $length characters counting from the start position of the substring. For example:

   echo substr($plane, 3, 5); // Returns “bus-A”

2. If $length is negative, the returned string will have $length characters omitted from the end of the string. For example:

   echo substr($plane, 3, -1); // Returns “bus-A38”

3. Here is an example when both the $start and $length parameters are negative:

   echo substr($plane, -4, -1); // Returns “A38”

Finding the position of a substring

When manipulating a string, you may need to find the position of a substring. The strpos() function is used to do this. Its syntax is:

   int strpos ( string $source , string $substr [, int $offset ] )

The function returns the position (counting from 0) of the first case-sensitive character of $substr if it is in the $source string. Consider the following examples:

   $source = “first come first served”;
   echo strpos($source, “f”); // Returns 0
   echo strpos($source, “e”); // Returns 9

If the optional $offset parameter is used, the search starts from the offset position in the source string. The following example statements start to search from the 11th character of the previous source string:

   echo strpos($source, “f”, 10); // Returns 11, targets 2nd “first”
   echo strpos($source, “e”, 10); // Returns 18, targets “served”
The `strpos()` function performs case-sensitive search. If you want to search without case-sensitivity, you will use the following syntax:

```php
int stripos ( string $source , string $substr [, int $offset ] )
```

The following example shows how `stripos()` returns a result different from that of `strpos()` when applied to the source string used in the previous example:

```php
echo strpos($source, "F"); // Since "F" is not found, it returns boolean FALSE

echo stripos($source, "F"); // Returns 0
```

### Replacing substrings

The `substr_replace()` function replaces the source string with a new string beginning from the `$start` position to the end of the source string. The syntax of the function is:

```php
string substr_replace(string $source, string $replacement, int $start [, int $length])
```

It is used in the same way as the substring extraction function `substr()`, except that there is an additional parameter (i.e., `$replacement`) for this function.

Consider the following string:

```
$plane = "Boeing-B787";
```

1. If `$start` is positive, the replacement string will start at the position specified by `$start` in the source string, counting from zero. For example:

```php
echo substr_replace($plane, "A380", 3); //Returns "BoeA380"
```

2. If `$start` is negative, the replacement string will start at the position specified by `$start`, counting from the end of the source string. For example:

```php
echo substr_replace($plane, "A380", -4); //Returns "Boeing-A380"
```
If the optional $length parameter is used, a maximum of $length characters are replaced. Consider the same source string used in the previous example:

$plane = "Boeing-B787";

1. If $length is positive, the string replacement function will replace only $length characters beginning from the start position. For example:

```php
echo substr_replace($plane, "A380", 3, 2);
// Returns "BoeA380g-B787"
```

2. If $length is negative, the function will stop replacing at $length characters from the end of the source string. For example:

```php
echo substr_replace($plane, "A380", 3, -1);
// Returns "BoeA3807"
```

3. Here is an example when both $start and $length parameters are negative:

```php
echo substr_replace($plane, "A380", -4, -1);
// Returns "Boeing-A3807"
```

Web Reference

The following web reference presents a comprehensive view of string functions:


Activity 3.2

1. The int strcmp ( string $str1 , string $str2 ) function performs a case-sensitive comparison of two strings. It returns _________ if $str1 is less than $str2, _________ if $str1 is greater than $str2, and _________ if they are equal.

A. positive, zero, negative  
B. zero, negative, positive  
C. negative, positive, zero  
D. positive, negative, zero
2. Which of the following will not produce zero?

A. `echo strcmp("android", "android");`
B. `echo strncmp("android3.1", "android3.2", 10);`
C. `echo strcasecmp("android", "Android");`
D. `echo strncasecmp("android3.1", "Android3.2", 9);`

3. Write the output of

`echo substr("Today is a PHP day", -7, -4);`

4. Write the output of

```php
$source = "Today is a PHP day";
echo stripos($source, "p");
```

5. You are given `$source = "Accomplish the same goal"`. Write code using the `substr_replace()` function to replace “same” with “different” so that the source string becomes “Accomplish the different goal”.

6. Write the output of the following code:

```php
$original = "PHP has string functions to manipulate strings";
$pos = strpos($original, "string functions");
$pos_end = $pos + strlen("string functions");
$new = substr_replace($original, " that give you an incredible ability", $pos_end, 0);
echo $new;
```
Summary

In this section, you learnt a number of PHP string library functions that help you to manage and manipulate strings. The following functions were discussed:

1. `strlen()` — Returns string length
2. `trim()` — Trims whitespaces or other characters
3. `str_pad()` — String padding
4. `strcmp()` — Compares two strings
5. `substr()` — Extracts substrings
6. `strpos()` — Finds the position of a substring
7. `substr_replace()` — Replaces substrings

In the next section, you will learn about arrays.

Self-test 3.1

1. Obtaining string length is a very common task when writing PHP scripts. The length of a string can be determined by the `strlen()` function. True / False?

2. Why do you need to use the string comparison function to compare two strings instead of the equality operator (==)?

3. The output of the following code is ____________.

   ```php
   $productCode = "AF8765";
   if ( strlen($productCode) < 6 )
   {
       echo "That product code is not valid";
       exit;
   }
   ```

   A. That product code is not valid  
   B. AF8765  
   C. <empty>  
   D. exit
4. The following function trims digits and spaces. What is the expected output?

```php
$var = trim("64oid brings Nintendo 64 emulation via channel 64", "0..9 ");
```

A. 64oid brings Nintendo 64 emulation via channel 64
B. oid brings Nintendo emulation via channel
C. oid brings Nintendo emulation via channel
D. oid brings Nintendo 64 emulation via channel

5. Use the `str_pad()` and `echo()` functions to print the following table by enclosing the code within the `<pre>` tag.

```
Table
A.................Pg 1
B.................Pg 2
C.................Pg 3
```

6. Consider the following variables. Write a `strcmp()` function to check if the two variables are the same. If they are the same, display “Passwords match!”. If they are different, display “Passwords do not match!”.

```php
$pswd = "supersecret";
$pswd2 = "supersecret2";
```

7. You are given the following log file variables:

```php
$file = "index.html";
$log = "192.168.1.11:/www/htdocs/index.html:[2012/02/10:20:36:50]";
```

A. What is the position ($pos) of $file in $log?
B. What is the string length ($filelen) of $file?
C. Calculate the position ($pos2) of the timestamp by using $pos, $filelen and an integer.
D. Using $log, $pos2 and a `substr()` function, print the statement: “The file index.html was accessed on: 2012/02/10:20:36:50”.

8. Write code that puts the substring “introduced in this chapter” into the string “Many functions will be used within your PHP applications” so that you get the string “Many functions introduced in this chapter will be used within your PHP applications”. Hint: Get the position and string length of the substring “Many functions”.

```php

```
Suggested answers to activities

Feedback

Activity 3.1

1. D

2. B

3. leading, trailing, both sides

4. `echo strlen("I’m learning php now");` // Returns 20

5. `echo rtrim("Log Report \t");` // Returns "Log Report"

6. Vegetables are good for health
   // Prints additional ten spaces after Vegetables

Activity 3.2

1. C

2. B

3. PHP

4. 11

5. `echo substr_replace($source, "different", 15, 4);` // OR
   `echo substr_replace($source, "different", -9, 4);` // OR
   `echo substr_replace($source, "different", -9, -5);`

6. PHP has string functions that give you an incredible ability to manipulate strings.
3.2 Arrays

Objectives

By the end of this section, you should be able to:

1. Handle collections of data with PHP.
2. Explain the concept of arrays.
3. Implement data collection using arrays.
4. Apply various types of array library functions.
5. Describe common applications of arrays.

Introduction

As a programmer of e-commerce web applications, you will spend a lot of time dealing with data sets. Some of these data sets might include customer details such as name, age or gender, while other data sets, such as one for an online shopping cart, would contain details like product name, quantity ordered and price. By using PHP, you will be able to deal with these data sets in a more organised and structured way. PHP allows you to store, update, retrieve and sort items in these data sets. These data sets are called **arrays**.

In Unit 2, you were introduced to compound data types that are able to store multiple items of data under a single representation. One of these compound types is the array. In this section, you will learn to use arrays. **Figure 3.1** shows a list of products that can be stored in an array.

![Figure 3.1 A list of products that can be stored in an array](image)
Creating arrays

Arrays can be created in two ways. The first involves using the function `array()`.
The syntax of the function is:

```
array([item1 [,item2 ... [,itemN]]])
```

For example, to create an array for the items in Figure 3.1, we can write:

```
$product = array( "Watch", "Shoe", "Cap" );
```

An array is actually an ordered map. A map is a type that associates values to keys.
A key may only be an integer or a string, whereas a value may be of any type.
It is possible to define the value associated with a key in the following way:

```
array( key => value , ... )
```

The following example shows how you can create an array called `$states` and
map its keys to corresponding values:

```
$states = array("PEN" => "Penang",
               "SEL" => "Selangor",
               "JHR" => "Johor");
```

The second way of creating an array is to use square brackets. This method allows
you to set up an array by directly setting the values. The syntax is:

```
$arr[key] = value;
```

Examples:

```
$product[0] = "Watch";
$product[1] = "Shoe";
$product[2] = "Cap";
$states["PEN"] = "Penang";
$states["SEL"] = "Selangor";
$states["JHR"] = "Johor";
```
Outputting arrays

You can print out individual values in an array by accessing the values with their key. The following statements print out values from the arrays created in the preceding example:

```php
echo $product[0]; // Prints "Watch"
echo $states["PEN"]; // Prints "Penang"
```

You can also use the `foreach` loop to iterate through and to print all the values in an array as stated in Unit 2. Remember that the `as` keyword in the `foreach` bracket is used to assign each item in the array to the variable, `$var`.

```php
foreach ($product as $var) {
    echo ($var . " "); // Prints "Watch Shoe Cap"
}
```

There is a function that can print out array contents for testing purposes called `print_r()`. The following examples print out the contents of the arrays `$product` and `$state` using this function:

```php
print_r($product);
print_r($states);
```

Adding and removing array elements

PHP provides functions that add or remove elements from an array. These tasks can be carried out by using the queue implementation functions, namely `array_unshift`, `array_push`, `array_shift` and `array_pop`.

**Unshift**

The `array_unshift()` function adds elements to the beginning of an array. You can add multiple elements into an array all at the same time. The syntax of the function is:

```php
int array_unshift ( array &$array , mixed $var [, mixed $... ] )
```

For example, to insert two elements to the front of an array:

```php
$platform = array ("delivery", "payment");
array_unshift ($platform, "cart", "content");  // $platform = array ("cart", "content", "delivery", "payment");
```
Push

The `array_push()` function adds elements to the end of an array. You can add multiple elements into an array all at the same time. The syntax of the function is:

```php
int array_push ( array &array , mixed $var [, mixed $... ] )
```

For example, to insert two elements to the end of an array:

```php
$platform = array ("delivery", "payment");
array_push ($platform, "service", "maintenance");
// $platform = array ("delivery", "payment", "service", "maintenance");
```

Shift

The `array_shift()` function removes the first element from an array and returns the item. The syntax of the function is:

```php
mixed array_shift ( array &array )
```

For example, to remove the first element from an array:

```php
$platform = array ("cart", "content", "delivery", "payment");
$firstElement = array_shift ($platform);
// $platform = array ("content", "delivery", "payment");
// $firstElement = "cart"
```

Pop

The `array_pop()` function removes the last element from an array and returns the item. The syntax of the function is:

```php
mixed array_pop ( array &array )
```

For example, to remove the last element from an array:

```php
$platform = array ("cart", "content", "delivery", "payment");
$lastElement = array_pop ($platform);
// $platform = array ("cart", "content", "delivery");
// $lastElement = "payment"
```
Extracting items from arrays

You can extract items from arrays by using the `list()` function. The `list()` function assigns variables as if they were an array. It is used to assign a list of variables in one operation. The `list()` function is useful when you are extracting information from a database or file. The syntax of the function is:

```
array list ( mixed $varname [, mixed $... ] )
```

For example, if an array of items is assigned to `$info` as follows:

```
$info = array("Value meal", "burger", "fries", "coke");
```

Then the `list()` function can be used to extract all the items from `$info` and assign them to variables:

```
list($set, $food, $sideline, $drink) = $info;
```

The extracted variables can then be used to print out a sentence:

```
echo "$set is made up of $food, $sideline and $drink.";
// Output: “Value meal is made up of burger, fries and coke.”
```

---

**Web Reference**

To learn more about arrays, please visit:


---

**Activity 3.3**

1. What are arrays and why do we want to use them?

2. Arrays can be created in two ways. The first involves using the function ____________ and the second way of creating an array is to use ____________.
3. Use the syntax `array(key => value, ...)` to store the following table. Then use the `print_r` function to print out the array.

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>YearsEmployed</th>
</tr>
</thead>
<tbody>
<tr>
<td>E908766</td>
<td>11</td>
</tr>
<tr>
<td>E915637</td>
<td>8</td>
</tr>
<tr>
<td>E987560</td>
<td>2</td>
</tr>
</tbody>
</table>

4. PHP provides convenient ways for you to add or remove elements from an array. These tasks can be done using the queue implementation functions _____________.

A. array_left, array_right, array_up and array_down
B. array_in, array_out, array_push and array_pull
C. array_pull, array_unpage, array_unpop and array_up
D. array_unshift, array_push, array_shift and array_pop

5. The `list()` function assigns variables as if they were an array. It is used to assign a list of items in an array to variables in one operation. True / False?

---

**Determining array size**

You can determine the size of an array by using the `count()` function. This function will return the number of values found in an array. The syntax of the function is:

```
int count ( array $var [, int $mode ] )
```

The following example shows how you can obtain the size of an array:

```php
$commercePlatform = array("cart", "content", "delivery", "payment");
echo count($commercePlatform); // Output: 4
```

The `count()` function does not detect infinite recursion. If the optional parameter is set to `COUNT_RECURSIVE` (or 1), `count()` will recursively count the array. This is particularly useful for counting all the elements of a multidimensional array. We will look at an example of this in the section on multidimensional arrays.
Array traversal

In certain circumstances, you will need to visit every array element and perform a task upon it. Every array has an internal pointer that points to the “current” value (element) in the array. The following functions operate using the internal array pointer:

next

The `next()` function sets the internal pointer to the next element and returns the element. Here is an example of its use:

```php
$service = array("contact center", "knowledge base", "self-service", "live help", "email", "video connect");
echo next($service); // Output: "knowledge base"
echo next($service); // Output: "self-service"
```

prev

The `prev()` function sets the internal pointer to the previous element and returns the element. It may return FALSE if the pointer currently resides at the first position in the array. The example below is a continuation from the previous example in which the internal pointer is currently at "self-service":

```php
echo prev($service); // Output: "knowledge base"
```

reset

The `reset()` function sets the internal pointer to the first element in an array and returns the first element. It is usually used when you need to review an array multiple times or when sorting has been completed. The example below is a continuation from the previous example in which the internal pointer is currently at “knowledge base”:

```php
echo reset($service); // Output: "contact center"
```

current

The `current()` function returns the current element and does not move the internal pointer. The example below is a continuation from the previous example:

```php
echo current($service); // Output: "contact center"
```
The `end()` function sets the internal pointer to the last element and returns the last element. The example below is a continuation from the previous example:

```php
echo end($service); // Output: “video connect”
```

The `each()` function returns the current element, then sets the internal pointer to the next element. The `each()` function is typically used in conjunction with key/value pairs from an array to traverse an array. For example:

```php
$food = array("a" => "burger", "b" => "fries", "c" => "coke");
list($key, $val) = each($food); // moves to next element
echo "$key is $val “; // Output: “a is burger”
list($key, $val) = each($food); // moves to next element
echo "$key is $val “; // Output: “b is fries”
list($key, $val) = each($food); // moves to next element
echo "$key is $val “; // Output: “c is coke”
```

Sorting arrays

Data sets that you extract from databases or files are usually not well arranged or categorised. If you need to do some serious data analysis, you have to segregate the items into groups according to specified criteria. This is what we call data sorting. Several techniques for sorting arrays are described next.

Normal sort

You can sort an array by using the `sort()` function. It will help you to arrange the elements from the lowest to the highest value. The syntax of the function is:

```php
bool sort ( array &$array [, int $sort_flags ] )
```

For example, we can sort an array containing marks that students in a class obtained:

```php
$marks = array(80, 60, 20, 100, 40);
sort($marks);
print_r($marks);
```
The optional parameter $sort_flags may be used to modify the sorting behaviour using the following values:

1. SORT_REGULAR — compare items normally (do not change types)
2. SORT_NUMERIC — compare items numerically
3. SORT_STRING — compare items as strings
4. SORT_LOCALE_STRING — compare items as strings, based on the current locale

Reverse sort

If you need to sort an array in descending or reverse order, you can use the rsort() function. The syntax of the function is:

```php
bool rsort ( array &$array [, int $sort_flags ] )
```

Other than providing reverse sorting, this function’s usage and optional features are identical to that of sort().

The following code example performs reverse sorting:

```php
$marks = array(80, 60, 20, 100, 40);
rsort($marks);
print_r($marks);
```

Sorting an array while maintaining key/value pairs

To sort an array while maintaining the key/value pair associations, you can use the asort() function. The syntax of the function is:

```php
bool asort ( array &$array [, int $sort_flags ] )
```

Consider the following key/value pairs:

```php
$meal = array("food" => "burger", "sideline" => "fries", "drink" => "coke");
```

If the sort() function is used on the array:

```php
sort($meal);
print_r($meal);
```
If the `asort()` function is used instead:

```php
asort($meal);
print_r($meal);
```

**Sorting an array in reverse order while maintaining key/value pairs**

If you need to sort an array in descending or reverse order while maintaining key/value pairs, you can use the `arsort()` function. The syntax of the function is:

```php
bool arsort ( array &$array [, int $sort_flags ] )
```

Other than providing reverse sorting, this function's usage and optional features are identical to that of `asort()`.

Consider the key/value pairs array again:

```php
$meal = array("food" => "burger", "sideline" => "fries", "drink" => "coke");
```

Using the `arsort()` function on `$meal`:

```php
arsort($meal);
print_r($meal);
```

**Multidimensional arrays**

You have learnt that arrays are a list of keys and values. What if you need to store more information like that found in the following table of fares for flights to Singapore?

<table>
<thead>
<tr>
<th>Code</th>
<th>From</th>
<th>Fare</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEN</td>
<td>Penang</td>
<td>RM176</td>
</tr>
<tr>
<td>KUL</td>
<td>Kuala Lumpur</td>
<td>RM83</td>
</tr>
<tr>
<td>LGK</td>
<td>Langkawi</td>
<td>RM205</td>
</tr>
</tbody>
</table>

*Figure 3.2 Flight fare table*
In PHP, elements in an array can also be an array, thus allowing for multidimensional arrays. You can use a two-dimensional array to store the fare information shown in Figure 3.2. The information can be stored as follows:

```php
```

To access an element of a two-dimensional array, you need to indicate the row and column in which the element is located. The top row is row 0 and the leftmost column is column 0. The following examples show how to display the contents of the array of flight fares:

```php
echo $SgpFlight[0][0]." | ".SgpFlight[0][1]." | ".SgpFlight[0][2]."<br ";
// Output: PEN | Penang | 176
echo $SgpFlight[1][0]." | ".SgpFlight[1][1]." | ".SgpFlight[1][2]."<br ";
// Output: KUL | Kuala Lumpur | 83
echo $SgpFlight[2][0]." | ".SgpFlight[2][1]." | ".SgpFlight[2][2]."<br ";
// Output: LGK | Langkawi | 205
```

We can also use a nested for loop to obtain the same result:

```php
for ($row = 0; $row < 3; $row++)
{
    for ($col = 0; $col < 3; $col++)
    {
        echo $SgpFlight[$row][$col]." | ";
    }
    echo "<br ";
}
// Output: PEN | Penang | 176
// Output: KUL | Kuala Lumpur | 83
// Output: LGK | Langkawi | 205
```

Remember that you can get the size of an array by using the count() function. However, to count the sub-elements inside a two-dimensional array, you need to include the COUNT_RECURSIVE optional parameter.

```php
echo count($SgpFlight); // Output: 3
echo count($SgpFlight, COUNT_RECURSIVE); // Output: 12
```
In this example, you have a total of 3 parent elements plus the 9 sub-elements in the recursive array, which gives you a total of 12.

Multidimensional arrays mean that you can use array elements to hold other arrays and these arrays can in turn hold more arrays. The example above showed the fares of flights to Singapore. You can store even more information, such as the fares of flights to Bangkok and the Gold Coast, by using a three-dimensional array. Refer to Figure 3.3 to see how this three-dimensional array is represented graphically.

**Figure 3.3** The three-dimensional array to store flight fare information

Here is how we can create the three-dimensional array:

```php
$MultipleFlights = array(
    array(
        array("PEN", "Penang", 1176),
        array("KUL", "Kuala Lumpur", 1083),
        array("LGK", "Langkawi", 1205)
    ),
    array(
        array("PEN", "Penang", 276),
        array("KUL", "Kuala Lumpur", 183),
        array("LGK", "Langkawi", 305)
    ),
    array(
        array("PEN", "Penang", 176),
        array("KUL", "Kuala Lumpur", 83),
        array("LGK", "Langkawi", 205)
    ));
```
We can then use nested `for` loops to display the contents of the multidimensional array:

```php
for ($layer = 0; $layer < 3; $layer++)
{
    echo "Flight".$layer."<br/>";
    for ($row = 0; $row < 3; $row++)
    {
        for ($col = 0; $col < 3; $col++)
        {
            echo $MultipleFlights[$layer][$row][$col]." | ";
        }
        echo "<br/>";
    }
}
```

The output of the code block will be:

```
Flight0
PEN | Penang | 1176 |
KUL | Kuala Lumpur | 1083 |
LGK | Langkawi | 1205 |
Flight1
PEN | Penang | 276 |
KUL | Kuala Lumpur | 183 |
LGK | Langkawi | 305 |
Flight2
PEN | Penang | 176 |
KUL | Kuala Lumpur | 83 |
LGK | Langkawi | 205 |
```

Although there is no limit to the number of dimensions an array can have in PHP, most real-world problems require arrays with three or less dimensions.

---

**Web Reference**

To learn more about array functions, please visit:

Activity 3.4

1. You can determine the size of an array by using the __________ function.

2. In certain circumstances, you will need to visit every array element and perform a task upon it. The following functions provide ways of traversing an array using the internal array pointer:

   A. incoming(), outgoing(), every() and adjust()
   B. come(), goto(), take() and put()
   C. current(), reset(), end() and each()
   D. shift(), push(), pull() and unpop()

3. If you need to sort an array in reverse order while maintaining key/value pairs, you can use the __________ function.

4. Write PHP code that stores the information in the following table in a multidimensional array and then prints the array using a nested for loop.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>ElementName</th>
<th>AtomicNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li</td>
<td>Lithium</td>
<td>3</td>
</tr>
<tr>
<td>Ti</td>
<td>Titanium</td>
<td>22</td>
</tr>
<tr>
<td>Pb</td>
<td>Lead</td>
<td>82</td>
</tr>
</tbody>
</table>

Summary

Arrays allow you to deal with data sets in a more organised and structured way. Using arrays, you can store, update, retrieve and sort groups of items. In this section, we discussed the following array library functions:

1. array() or $arr[key] = value — Creates arrays
2. array( key => value , ... ) — Key/value mapping
3. $arr[key] or foreach() or print_r() — Outputs arrays
4. array_unshift() and array_push() — Add elements
5. array_shift() and array_pop() — Remove elements

6. list() — Extracts arrays

7. count() — Determines array size

8. next(), prev(), reset(), current(), end(), each() — Array traversal

9. sort(), rsort(), asort(), arsort() — Sort arrays

We also discussed multidimensional arrays and looked at how they are applied. In the next section, we will discuss files and directories, and how to read from and write to a file. PHP file system library functions will also be introduced in the next section.

Self-test 3.2

1. Create an array based on the table of accessories shown below.

<table>
<thead>
<tr>
<th>Accessories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
</tr>
<tr>
<td>Lubricant</td>
</tr>
<tr>
<td>Wax</td>
</tr>
</tbody>
</table>

a. Use a foreach loop to print out the array with a space between the items.

b. Use a for loop to print out the array with a line “ | ” after each item.

2. Based on the table below, create an array using the array() function and also the square brackets method. Use the print_r() function to print the arrays.

<table>
<thead>
<tr>
<th>Code</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>Somalia</td>
</tr>
<tr>
<td>SUD</td>
<td>Sudan</td>
</tr>
<tr>
<td>MAU</td>
<td>Mauritania</td>
</tr>
</tbody>
</table>
3. Rewrite the contents of the array after each operation below:

a. $fruits = array ("strawberry", "kiwi", "banana");
   array_unshift ($fruits, "apple");

b. $fruits = array ("strawberry", "kiwi", "banana");
   array_push ($fruits, "durian");

c. $fruits = array ("strawberry", "kiwi", "banana");
   array_shift ($fruits);

d. $fruits = array ("strawberry", "kiwi", "banana");
   array_pop ($fruits);

4. You are given $var = array("Mary", "format", "information"); Write an echo function to print “Mary wants to format output information.” by extracting the elements of the array with a list() function.

5. What is the output of the following code?

   $prices = array("Battery" => 200,
                   "Lubricant" => 80,
                   "Wax" => 30);
   while (list($accessories, $price) = each ($prices))
       echo "$accessories = $$price <br>";

6. Write PHP code to determine the size of the following arrays:

a. $accounts = array ("saving", "fixed deposit", "current", "loan", "islamic", "foreign", "stock");

b. $num = array(array(87, 54, 76, 89, 30),
                    array(34, 67, 73, 49, 27),
                    array(16, 88, 25, 59, 31));

by applying the COUNT_RECURSIVE parameter.
7. You are given the following array:

```php
$commodities = array("metal", "energy", "fuel", "grains", "agricultural", "solar");
```

Determine each of the elements using the internal array pointer in sequence from a. to e.:

- a. echo next($commodities);
- b. echo end($commodities);
- c. echo prev($commodities);
- d. echo reset($commodities);
- e. echo current($commodities);

8. Consider the following “News” sections:

```
World | Sci-Tech | Business | Entertainment | Sports | Health
```

Create an array to store the data.

- a. Sort the array so that it displays the elements in alphabetical order. Print out the array using the `print_r()` function.
- b. Sort the array so that it displays the elements in reverse order. Print out the array using the `print_r()` function.

9. Consider the following table:

```
<table>
<thead>
<tr>
<th>Code</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>North</td>
</tr>
<tr>
<td>E</td>
<td>East</td>
</tr>
<tr>
<td>S</td>
<td>South</td>
</tr>
<tr>
<td>W</td>
<td>West</td>
</tr>
</tbody>
</table>
```

Create an array to store the data in the table.

- a. Sort the array so that it displays the elements in alphabetical order while maintaining key/value pairs. Print out the array using the `print_r()` function.
b. Sort the array so that it displays the elements in reverse order while maintaining key/value pairs. Print out the array using the `print_r()` function.

10. Store the following BWF World Ranking table in a multidimensional array and print the array using a nested for loop

<table>
<thead>
<tr>
<th>Rank</th>
<th>Player name</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chong Wei LEE</td>
<td>MAS</td>
</tr>
<tr>
<td>2</td>
<td>Taufik HIDAYAT</td>
<td>INA</td>
</tr>
<tr>
<td>3</td>
<td>Dan LIN</td>
<td>CHN</td>
</tr>
<tr>
<td>4</td>
<td>Peter Hoeg GADE</td>
<td>DEN</td>
</tr>
<tr>
<td>5</td>
<td>Long CHEN</td>
<td>CHN</td>
</tr>
</tbody>
</table>

Suggested answers to activities

Feedback

Activity 3.3

1. Arrays are defined as data sets that share similar characteristics or a data type. We use arrays to organise data in a structured way that allows us to store, update, retrieve and sort a group of data items easily.

2. `array(); square brackets`

3. ```
   $emp = array("E908766" => 11, 
   "E915637" => 8, 
   "E987560" => 2);
   print_r ($emp);
```  

4. D

5. True
Activity 3.4

1. count()

2. C

3. arsort()

4. $Periodic = array(array("Li", "Lithium", 3),
               array("Ti", "Titanium", 22),
               array("Pb", "Lead", 82));
   for ($row = 0; $row < 3; $row++)
   {
       for ($col = 0; $col < 3; $col++)
       {
           echo $Periodic[$row][$col]." | ";
       }
   }
   echo "<br />
   // Li | Lithium | 3 |
   // Ti | Titanium | 22 |
   // Pb | Lead | 82 |
3.3 Files

Objectives

By the end of this section, you should be able to:

1. Describe the basic file operations in PHP.
2. Perform a variety of operations on files using PHP.
3. Apply PHP file system library functions.

Introduction

When working with web applications, it is often necessary to store information in a permanent form. There are two primary ways to store data: files and databases. In this section you will learn how to interact with files. You will learn about databases in Unit 4. It is easier and quicker to use files to perform simple tasks. Files are usually used to store web logs, configurations and server settings. To manipulate files, PHP provides a number of file-related functions.

Opening a file

The `fopen()` function opens a file in a mode that you specify and returns a handle. Using this handle you can read or write to a file via a file pointer. The file pointer is used to denote the present position in the file. When you open a file for reading, the file pointer points to the beginning of the file. The syntax for this function is:

```php
resource fopen ( string $filename , string $mode [, bool $use_ include_path [, resource $context ]])
```

The following is a list of the common modes used as the second parameter:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>Open for reading only; place the file pointer at the beginning of the file.</td>
</tr>
<tr>
<td>r+</td>
<td>Open for reading and writing; place the file pointer at the beginning of the file.</td>
</tr>
<tr>
<td>w</td>
<td>Open for writing only; place the file pointer at the beginning of the file and delete contents. If the file does not exist, attempt to create it.</td>
</tr>
</tbody>
</table>
Let us now look at how the `fopen()` function is used:

```php
$Hdl = fopen("data.txt", "r"); // Open a read-only handle
$Hdl = fopen("data.txt", "w"); // Open the write handle
```

In the first example above, you will notice that the file was opened for reading only by specifying "r" as the mode. The `fopen()` function only prepares the file for another operation. Other than establishing the handle, it does nothing else. You will need to use other functions to actually perform the read and write operations.

### Closing a file

The `fclose()` function closes a file after you finish using it. The syntax of the function is:

```php
bool fclose ( resource $handle )
```

To close the file that you opened in the previous example, you will use:

```php
fclose($Hdl); // Close the file
```

When a PHP script finishes executing, all open files are automatically closed. Although it is not strictly necessary to close a file using the `fclose()` function after opening it, it is considered good programming practice to do so.
Reading a file

PHP provides many functions for reading data from a file. If you just want to make all the contents of a file available for you to work with, you can use the `file_get_contents()` function. However, if you want each line of a file stored in an array, you will use the `file()` function. For total control over the task of reading from a file, the `fread()` function is used.

The `file_get_contents()` and `file()` functions do not require you to use the open and close file commands. These functions are good for quick, one-time file operations. If you plan on performing multiple operations on a file, it is best to use `fopen()` in conjunction with `fread()`, `fwrite()` and `fclose()` as this is more efficient. The functions for reading data from a file are described in detail next.

file_get_contents

The syntax of the `file_get_contents()` function is:

```php
string file_get_contents ( string $filename [, bool $use_include_path = false [, resource $context [, int $offset = -1 [, int $maxlen ]]]] )
```

Assume that you have a text file named ‘file.txt’ in the same directory as your PHP source code file. The following example code causes the contents of the file to be printed:

```php
$myfile = file_get_contents('file.txt');
echo $myfile;  // Print all contents from the file
```

Using the optional parameters, you can tell `file_get_contents()` to search through the paths specified in the `$use_include_path` parameter. The resource `$context` parameter refers to a stream context. A `stream` is a resource object which can be read from or written to in a linear fashion. The `$offset` parameter indicates where the reading starts from in the original stream. You can indicate the maximum length of data read using `$maxlen`.

file

The syntax of the `file()` function is:

```php
array file ( string $filename [, int $use_include_path = 0 [, int $offset = 0 [, resource $context ]]] )
```
The following example shows how the function reads an entire file into an array. Each item in the array corresponds to a line in the file.

```php
$Lines = file('file.txt');
foreach($Lines as $Key => $Line) {
    $LineNum = $Key + 1;
    echo "Line $LineNum: $Line <br>";
}
```

The result of the code above is:

- Line 1: Line 1 content
- Line 2: Line 2 content
- Line 3: Line 3 content

The $use_include_path and resource $context parameters operate in a manner identical to those described in the file_get_contents subsection.

### fread

The syntax of the `fread()` function is:

```php
string fread ( resource $handle , int $length )
```

The following code shows how this function is used. The code opens a file (`file.txt`), reads a number of bytes (as specified in the `$length` parameter) from the file and returns a string. The file is then closed and the string is printed.

```php
$handle = fopen('file.txt', 'r');
$string = fread($handle, 64);
fclose($handle);
echo $string; // Print 64 bytes from the contents of the file
```

Note that with ASCII encoding, storing a character in a file requires one byte. A Unicode character requires two bytes.

---

**Web Reference**

To learn more about file system functions, please visit:

Activity 3.5

1. An internal array pointer is used to denote the present position in a file. When you open a file for reading, the internal array pointer points to the beginning of the file. True / False?

2. A file named `preface.txt` contains the following two lines:

   What is a web database application?
   It brings databases and the Web together.

   What is the output of the following code?

   a. `$Source = file('preface.txt');
      foreach($Source as $Counter => $Display) {
          $Num = $Counter + 1;
          echo "Going to $Num: $Display <br>";
      };

   b. `$handle = fopen('preface.txt', 'r');
      $string = fread($handle, 34);
      fclose($handle);

Writing to a file

The function `file_put_contents()` enables a file to be written to. Writing to a file can also be performed by the `fwrite()` function in conjunction with `fopen()` and `fclose()`.

Generally, there are two modes you can supply to the `fopen()` function when you write to a file. The first mode, ‘w’, deletes the original contents of the file and replaces them with what you write. The second mode, ‘a’, appends the new content just after the original contents of the file. The functions that are used to write to a file are discussed next.

`file_put_contents`

The syntax of the `file_put_contents()` function is:

```
int file_put_contents ( string $filename , mixed $data [, int $flags = 0 [, resource $context ]]] )
```
The following is an example of how to write to a file. In this example, the original contents of the file are completely replaced by new content:

```
$content = 'New content.;
$file_put_contents('file.txt', $content);
$myfile = file_get_contents('file.txt');
echo $myfile; // Prints: “New content.”
```

The next example shows you how to append new content to the original contents:

```
$appContent = ' and other languages are no different.';
$file_put_contents('file.txt', $appContent, FILE_APPEND);
$myfile = file_get_contents('file.txt');
echo $myfile; // Prints: “Original content and other languages are no different.”
```

The `file_put_contents()` function will also attempt to create the file if it does not exist. Note that this function is binary safe.

`fwrite`

The syntax of the `fwrite()` function is:

```
int fwrite ( resource $handle , string $string [, int $length ] )
```

The following example shows how this function is used to write to a file. It replaces the original contents of the file with new content:

```
$handle = fopen('file.txt', 'w'); // Open and delete all contents
$data = “New content.”;
fwrite($handle, $data);
fclose($handle);
echo file_get_contents('file.txt'); // Prints: “New content.”
```

The next example shows how `fwrite()` is used to append new content to the original contents:

```
$handle = fopen('file.txt', 'a'); // Open file for appending
$data = “ and other languages are no different.”;
fwrite($handle, $data);
fclose($handle);
echo $myfile; // Prints: Original content + and other languages are no different.
```
Deleting a file

The `unlink()` function helps to delete a file. The syntax of the function is:

```php
bool unlink ( string $filename [, resource $context ] )
```

For example, the following statement deletes a file called ‘file.txt’:

```php
unlink (‘file.txt’); // Delete the file called ‘file.txt’
```

To `unlink`, the user must have permissions to write to the directory that contains the file. Take note that if a user has write permissions to a server directory, the user may delete files from that server directory regardless of who owns them.

Reading directory contents

Sometimes, you may need to list all the files and child directories in a given directory. To do this, you need to first open the directory, then read the directory to list the files and subdirectories, and finally close the directory. The following subsection describes the functions that are used to open, close and read a directory:

Opening a directory

The `opendir()` function opens a directory stream specified by a path. The syntax of the function is:

```php
resource opendir(string path)
```

Closing a directory handle

The `closedir()` function closes the directory stream. Its syntax is as follows:

```php
void closedir(resource directory_handle)
```

Parsing directory contents

The `readdir()` function returns each element in a directory. Its syntax is as follows:

```php
string readdir(int directory_handle)
```
For example, the following code lists all the files and child directories in a given directory:

```php
$dh = opendir("C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\");
while ($file = readdir($dh))
{
    echo "$file \n";
}
closedir($dh);
```

Checking file or directory existence

The `file_exists()` function checks whether a file or directory exists. Its syntax is:

```php
bool file_exists ( string $filename )
```

For example, the following code checks whether a file called ‘file.txt’ exists:

```php
echo file_exists('file.txt'); // Return true if the file exists; false otherwise.
```

Web Reference

To learn more about the PHP file system, please visit:


Activity 3.6

1. The function `file_write_contents()` enables the process of writing to a file. This function will also attempt to create the file if it does not exist and it is binary safe. True / False?

2. To delete a file, you will use the function `delete($filename)`. True / False?
3. What does the following PHP code do?

```php
$dh = opendir("C:"嘭;
while ($file = readdir($dh))
{
    echo "$file <br />嘭")
}
closedir($dh);"嘭
```

**Summary**

PHP provides a number of file-related functions that are used to manipulate files. In this section, we discussed the following file-related functions:

1. `fopen()` — Opens a file
2. `fclose()` — Closes a file
3. `file_get_contents()`, `file()`, `fread()` — Read a file
4. `file_put_contents()`, `fwrite()` — Write a file
5. `unlink()` — Deletes a file
6. `opendir()`, `closedir()`, `readdir()` — Manipulate a directory
7. `file_exists()` — Checks file or directory existence

You should now know how to write PHP code to:

1. Open and close a file.
2. Read from and write to a file.
3. Read directory contents and check file or directory existence.
Self-test 3.3

1. The `file_get_contents()` and `file()` functions do not require you to use the `fopen()` and `fclose()` functions. True / False?

2. All of the following file modes allow you to open a file for writing except:
   
   A. `fopen("doc.txt", "r+");`
   B. `fopen("doc.txt", "w");`
   C. `fopen("doc.txt", "a");`
   D. `fopen("doc.txt", "r");`

3. Consider a text file named ‘users.txt’ that has the following contents:

Ale ale@php.com
Nicole nicole@php.com
Laura laura@php.com

Write PHP code to print the contents from the file using the following functions:

a. `fread()`  
   (Assume that 64 bytes of the contents are to be printed)

b. `file()`

c. `file_get_contents()`

4. Using the `file_put_contents()` function, create a text file named ‘km_miles.txt’ that has the following contents:

1 km = 0.621400 miles
2 km = 1.242800 miles
3 km = 1.864200 miles

Then append the following data into the file using the `fwrite()` function.

4 km = 2.485600 miles
5 km = 3.107000 miles
5. Write PHP code to list all files and child directories in 'C:\Program Files'.

6. Write PHP code that:
   a. Checks that a file exists and if it does, deletes the file.
   b. Checks that a subdirectory does not exist and if it does not, creates the subdirectory.

Suggested answers to activities

Feedback

Activity 3.5
1. False

2. a. Going to 1: What is a web database application?
   Going to 2: It brings databases and the Web together.

   b. What is a web database application

Activity 3.6
1. False

2. False

3. It will list all the directories and files in the C drive.
Summary of Unit 3

Summary

In the first section of this unit, you learnt how to use PHP string library functions to manipulate strings. You focused on string-related tasks that frequently appear in web application design, such as trimming whitespaces, comparing strings and extracting strings. You also learnt how to perform string padding. In the second section, you learnt to handle collections of data by using arrays. You also learnt about the various array library functions, as well as the common applications of arrays and multidimensional arrays. In the final section, you learnt how to work with files. You were exposed to the various file and directory manipulation functions that PHP provides.

In the next unit, you will be introduced to the very interesting topic of connecting a web page to a MySQL database server using PHP.
UNIT 3 51
Manipulating strings, arrays and working with files

Unit Practice Exercises
1. In this exercise, you will modify the “MySite09” web page from Unit 2. Copy
the ‘MySite09.php’ file from Unit 2 into the ‘/var/www/Unit3’ directory (or
‘<Apache Directory>\htdocs\Unit3’ in Windows) and name it ‘MySite10.php’.
Your task is to print the same table that you created in ‘MySite09.php’ but
you will use a two-dimensional array to store the data this time. You should
obtain a web page like the one shown in Figure 3.5.
Note: You will need to copy the “include” files like ‘header.php’ over to the
Unit3 directory.
Object

Markup

Value/Example

Title

<Head><Title>…</Title>
</Head>

My Site 10

Storing
elements
into the
array

Create additional variables as
follows:

array_push ($element,
$name.$i );

$name = “Espresso Roast
Grade”;
$currency = “RM”;
$elements = array();

$price =
$pricePerOz * $i;
array_push ($element,
$currency.$price );

By using the same for loop,
create a new array in the loop:
$element = array();

Remove all the HTML tags but
retain the formulas.
Store each column data into
the array $element by using
the array_push() function.
See examples in the righthand side column of this
table and complete all other
column data.
Maintain the formulas:
$totalPoints += $p;
$grandTotal +=
$subTotal;

Print array

By using a nested for loop,
print out the array.
Insert HTML table tags into
the nested for loop to print
out the table. See example in
the right-hand side column.

for ($row = 0; $row <
$items; $row++)
{
echo “<tr>”;
for ($col = 0; $col
< count($element);
$col++)
{
echo “<td>”;
…
…


2. In this exercise, create two new PHP files in the '/var/www/Unit3' directory (or '\Apache Directory\htdocs\Unit3' in Windows) and name them 'MySite11.php' and 'MySite11-register.php'.

a. In 'MySite11.php', create a registration form by using the HTML code shown in the table below.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;My Site 11&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td></td>
</tr>
<tr>
<td>Registration form</td>
<td>Insert into the HTML body tag.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;form name=&quot;registration_form&quot; method=&quot;post&quot; action=&quot;MySite11-register.php&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name: <code>&lt;input type=&quot;text&quot; name=&quot;name&quot;&gt;</code>&lt;br /&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Email: <code>&lt;input type=&quot;text&quot; name=&quot;email&quot;&gt;</code>&lt;br /&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Password: <code>&lt;input type=&quot;password&quot; name=&quot;password&quot;&gt;</code>&lt;br /&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Confirm Password: <code>&lt;input type=&quot;password&quot; name=&quot;password2&quot;&gt;</code>&lt;br /&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mobile Phone Number: <code>&lt;input type=&quot;text&quot; name=&quot;phone&quot;&gt;</code>&lt;br /&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;input type=&quot;submit&quot; value=&quot;Register&quot;&gt;</code>&lt;br /&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;/form&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;/pre&gt;</td>
<td></td>
</tr>
</tbody>
</table>
You should obtain a web page like the one shown in Figure 3.6.

![Figure 3.6 User registration form](image)

b. In ‘MySite11-register.php’, you are going to manipulate various string functions according to the instructions given in the table below.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP page</td>
<td>&lt;?PHP ... ?&gt;</td>
<td></td>
</tr>
<tr>
<td>Obtain form values</td>
<td>Use the $_POST[&lt;field name&gt;] to obtain values from the user registration form in question 2.a.</td>
<td>$_POST['name'];</td>
</tr>
<tr>
<td>Trim whitespace</td>
<td>Trim whitespace and store data into the following variables: $name $email $password $password2 $phone</td>
<td>$name = trim($_POST['name']);</td>
</tr>
<tr>
<td>String length</td>
<td>Use a string function to validate the length of the string in the $name field. Any string less than 6 characters in length will produce an error message and cause the exit() function to execute.</td>
<td>// If error, echo &quot;Invalid name&quot;; exit;</td>
</tr>
</tbody>
</table>


| String position and substring extraction | Use a string function to extract the part of the string before the @ symbol from the $email field. If the @ symbol is not found, an error message will be displayed and the exit() function will be executed. | // If error, echo "Invalid email"; exit; |
| String comparison | Use a string function to compare $password and $password2. If they are not the same, an error message will be displayed and the exit() function will be executed. | // If error, echo "Invalid passwords"; exit; |
| Substring and substring replacement | Check to make sure that the phone number is exactly 10 digits. Use several string functions to transform the phone number from 01XXXXXXXX into the international format (+60)XX-XXXXXXX. | // E.g., 0124567890 becomes (+60)12-4567890 |
| Test the form | Fill up the registration form, 'MySite11.php' with valid and invalid data to test the string validation in the fields. See Figure 3.7. | Figure 3.8 shows an example of the page displayed when the data is invalid. |
3. In this exercise, you will modify the “MySite11” and “MySite11-register” web pages from question 2. Copy both files into the ‘/var/www/Unit3’ directory (or ‘<Apache Directory>htdocs\Unit3’ in Windows) and name them ‘MySite12.php’ and ‘MySite12-register.php’ respectively.

a. In ‘MySite12.php’, modify the form action to action="MySite12-register.php".

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;My Site 12&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>My Site 12</td>
</tr>
<tr>
<td>Form action</td>
<td>Modify the form action</td>
<td><code>&lt;form name=&quot;registration_form&quot; method=&quot;post&quot; action=&quot;MySite12-register.php&quot;&gt;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP page</td>
<td>&lt;?PHP ... ?&gt;</td>
<td>My Site 12</td>
</tr>
<tr>
<td>Writing a file</td>
<td>Create a writeLog($message) function that is able to timestamp the current date and time together with the $message into a 'register.log' file. Use the example shown in the right-hand side column to help you.</td>
<td>date_default_timezone_set('Asia/Kuala_Lumpur');&lt;br&gt;$date = date(&quot;d F, Y&quot;);&lt;br&gt;$time = date(&quot;G:i:s&quot;);&lt;br&gt;...&lt;br&gt;$content = $date.' ,'.$time.' ,'.$message;&lt;br&gt;...</td>
</tr>
<tr>
<td>Use the function</td>
<td>Use the function to write the following values to the log file together with their corresponding text labels.</td>
<td>writeLog(&quot;<strong>Log registration details</strong>&lt;br/&gt;&quot;);&lt;br&gt;writeLog(&quot;Name: &quot;.$name.&quot;&lt;br/&gt;&quot;);&lt;br&gt;...</td>
</tr>
<tr>
<td>Read and display the log file</td>
<td>Display the contents of the 'register.log' file.</td>
<td>You should obtain a page similar to the one shown in Figure 3.9.</td>
</tr>
</tbody>
</table>

Figure 3.9 Displaying the contents of the log file
Suggested Answers to Self-tests

Feedback

Self-test 3.1

1. False

2. Although you can compare two strings using the equality operator (==), this may cause unexpected or unsafe results for strings with binary content or multi-byte encoding.

3. C

4. D

5. echo “<pre>”;
   echo str_pad(“Table”, 20, “ “, STR_PAD_BOTH) . “
   echo str_pad(“A”, 10, “.”) . str_pad(“Pg 1”, 10, “.”, STR_PAD_LEFT) . “
   echo str_pad(“B”, 10, “.”) . str_pad(“Pg 2”, 10, “.”, STR_PAD_LEFT) . “
   echo str_pad(“C”, 10, “.”) . str_pad(“Pg 3”, 10, “.”, STR_PAD_LEFT) . “
   echo “</pre>”;

6. if (strcmp($pswd,$pswd2) != 0)
   echo “Passwords do not match!”;
else
   echo “Passwords match!”;

7. a. $pos = strpos($log, $file);
   echo $pos; // 25

   b. $filelen = strlen($file);
   echo $filelen; // 10

   c. $pos2 = $pos + $filelen + 2;
   echo $pos2; // 37
d. $timestamp = substr($log, $pos2, -1);
echo "The file $file was accessed on: $timestamp";
// The file index.html was accessed on: 2012/02/10:20:36:50

8. $original = “Many functions will be used within your PHP applications”;
   // Get the position of ‘Many functions’
   $pos = strpos($original, “Many functions”);
   // Find where ‘Many functions’ ends by adding the length of ‘Many functions’
   $pos_end = $pos + strlen(“Many functions”);
   // Replace new substring content
   $new = substr_replace($original, “ introduced in this chapter”, $pos_end, 0);

Self-test 3.2

1. $accessories = array(“Battery”, “Lubricant”, “Wax”);
   foreach ($accessories as $var) {
       echo ($var . “ “);
   }
   // Battery Lubricant Wax
   for ($col = 0; $col < 3; $col++)
   {
       echo $accessories[$col].” | “;
   }
   // Battery | Lubricant | Wax |

2. $Array1 = array(“SOM” => “Somalia”,
   “SUD” => “Sudan”,
   “MAU” => “Mauritania”);
   print_r($Array1);

   $Array2[“SOM”] = “Somalia”;
   $Array2[“SUD”] = “Sudan”;
   $Array2[“MAU”] = “Mauritania”;
   print_r($Array2);
3. a. $fruits = array ("apple", "strawberry", "kiwi", "banana");

   b. $fruits = array ("strawberry", "kiwi", "banana", "durian");

   c. $fruits = array ("kiwi", "banana");

   d. $fruits = array ("strawberry", "kiwi");

4. list($who, $action, $what) = $var;
   echo "$who wants to $action output $what.";

5. Battery = $200
   Lubricant = $80
   Wax = $30

6. a. count($accounts); // 7

   b. count($num, COUNT_RECURSIVE); // 18

7. a. energy

   b. solar

   c. agricultural

   d. metal

   e. metal


   a. sort($news);
   print_r($news);

   b. rsort($news);
   print_r($news);
9. $direction = array("N" => "North",
    "E" => "East",
    "S" => "South",
    "W" => "West");

a. asort($direction);
   print_r($direction);

b. arsort($direction);
   print_r($direction);

10. $BwfRanking = array(array(1, “Chong Wei LEE”, “MAS”),
    array(2, “Taufik HIDAYAT”, “INA”),
    array(3, “Dan LIN”, “CHN”),
    array(4, “Peter Hoeg GADE”, “DEN”),
    array(5, “Long CHEN”, “CHN”));

for ($row = 0; $row < 5; $row++)
{
    for ($col = 0; $col < 3; $col++)
    {
        echo $BwfRanking[$row][$col]." | ";
    }
    echo "<br />
}

// 1 | Chong Wei LEE | MAS |
// 2 | Taufik HIDAYAT | INA |
// 3 | Dan LIN | CHN |
// 4 | Peter Hoeg GADE | DEN |
// 5 | Long CHEN | CHN |

Self-test 3.3

1. True

2. D

3. a. $handle = fopen('user.txt', 'r');
   $data = fread($handle, 64);
   fclose($handle);
   echo $data;
b. $data = file('user.txt');
    foreach($data as $line) {
        echo $line." <br>
    }

c. $data = file_get_contents('user.txt');
    echo $data;

4. $content = “1 km = 0.621400 miles\n”.
   “2 km = 1.242800 miles\n”.
   “3 km = 1.864200 miles\n”;
   file_put_contents('km_miles.txt', $content);
   // Open for appending
   $handle = fopen('km_miles.txt', 'a');
   $extra = “4 km = 2.485600 miles\n”.
   “5 km= 3.107000 miles”;
   fwrite($handle, $extra);
   fclose($handle);

5. $dh = opendir("C:\Program Files");
    while ($file = readdir($dh))
    {
        echo "$file <br />”;
    }
    closedir($dh);

6. a. if (file_exists('afile.txt'))
    {
        unlink ('afile.txt');
    }

   b. if (!file_exists('subdir'))
    {
        mkdir ('subdir');
    }
Suggested Answers to Unit Practice Exercises

Feedback

1. MySite10.php

```php
<?php

$items = 5;
$count = 1;
while ($count <= $items)
{
    printf("Espresso Roast Grade %d earns %d loyalty points <br />", $count, pow($count, 2));
    $count++;
}

$table = '<table border="1" width="50%">
<tr><th>Coffee</th><th>Cost</th><th>Quantity</th><th>Sub Total</th></tr>';

<?php
$pricePerOz = 10.95;
$grandTotal = 0;
$totalPoints = 0;
$name = "Espresso Roast Grade ";
$currency = "RM";
$elements = array();

for($i = 1; $i <= $items; $i++)
{
    $element = array();
    array_push ($element, $name.$i);
    array_push ($element, $currency.$price);
    $price = $pricePerOz * $i;
    array_push ($element, $price);
    $totalPoints += $price;
    $grandTotal += $price;
}
$table .= $elements;
$table .= '</table>';";

echo $table;

```
```php
$q = $items - $i + 1;
$p = $q * pow($i, 2);
array_push ($element, $q );
$subTotal = $price * $q;
array_push ($element, $currency.$subTotal );
$totalPoints += $p;
$grandTotal += $subTotal;
array_push ($elements, $element );
}

for ($row = 0; $row < $items; $row++)
{
    echo "<tr>";
    for ($col = 0; $col < count($element); $col++)
    {
        echo "<td>";
        echo $elements[$row][$col];
        echo "</td>";
    }
    echo "</tr>";
}
echo "</table>";
printf("<p>Grand total: <strong>RM%.2f</strong></p>", $grandTotal);
printf("Total points earned: %d", $totalPoints);
?>
</BODY>
</HTML>

2. a. MySite11.php

```
b. MySite11-register.php

<?PHP

// Trim whitespace of all items
$name = trim($_POST['name']);
$email = trim($_POST['email']);
$password = trim($_POST['password']);
$password2 = trim($_POST['password2']);
$phone = trim($_POST['phone']);

// Make sure string length of name is > 6
if (strlen($name) < 6)
{
    echo "Invalid name"
    exit;
}

// Substring email as username
$symbolPos = strpos($email, "@");  
if ($symbolPos == FALSE)
{
    echo "Invalid email"
    exit;
}
$username = substr($email, 0, $symbolPos);

// Compare if two passwords are the same
if (strcmp ($password, $password2) != 0)
{
    echo "Invalid passwords"
    exit;
}

// Make sure string length of phone is 10
if (strlen($phone) != 10)
{
    echo "Invalid phone"
    exit;
}

// Replace 0124567890 with (+60)12-4567890

$intmobile = substr_replace($phone, "(+60)1", 0, 2);
$part1 = substr($intmobile, 0, 7);
$part2 = substr($intmobile, 7);
$part1 = $part1 . ";";
$intmobile = $part1 . $part2;
?>

3 a. MySite12.php

<HTML>
  <HEAD>
    <TITLE>MySite12</TITLE>
  </HEAD>
  <BODY>
    <pre>
      <form name="registration_form" method="post"
action="MySite12-register.php">
        Name: <input type="text" name="name"><br />
        Email: <input type="text" name="email"><br />
        Password: <input type="password" name="password"><br />
        Confirm Password: <input type="password" name="password2"><br />
        Mobile Phone Number: <input type="text" name="phone"><br />
        <input type="submit" value="Register">
      </form>
    </pre>
  </BODY>
</HTML>

b. MySite12-register.php

<?PHP
  // Trim whitespace of all items
  $name = trim($_POST['name']);
  $email = trim($_POST['email']);
  $password = trim($_POST['password']);
  $password2 = trim($_POST['password2']);
  $phone = trim($_POST['phone']);

  // Make sure string length of name is > 6
  if (strlen($name) < 6)
  {
    echo "Invalid name";
    exit;
  }
?>
// Substring email as username
$symbolPos = strpos($email, "@");
if ($symbolPos == FALSE)
{
    echo "Invalid email";
    exit;
}
$username = substr($email, 0, $symbolPos);

// Compare if two passwords are the same
if (strcmp ($password, $password2) != 0)
{
    echo "Invalid passwords";
    exit;
}

// Make sure string length of phone is 10
if (strlen($phone) != 10)
{
    echo "Invalid phone";
    exit;
}

// Replace 0124567890 with (+60)12-4567890
$intmobile = substr_replace($phone, "(+60)1", 0, 2);
$part1 = substr($intmobile, 0, 7);
$part2 = substr($intmobile, 7);
$part1 = $part1 . "-";
$intmobile = $part1 . $part2;

// logging
writeLog("**Log registration details**<br />");
writeLog("Name: ".$name."<br />");
writeLog("Email: ".$email."<br />");
writeLog("Password: ".$password."<br />");
writeLog("Username: ".$username."<br />");
writeLog("International mobile: ".$intmobile."<br />");

echo file_get_contents('register.log');

function writeLog($message)
{
    date_default_timezone_set('Asia/Kuala Lumpur');
    $date = date("d F, Y");
    $time = date("G:i:s");

    $handle = fopen('register.log', 'a');

    fwrite($handle, "$date $time $message
");
    fclose($handle);
}
$content = $date.'_'.$time.'_'.$message;
fwrite($handle, $content);
fclose($handle);
?>

register.log sample
07 March, 2011,2:12:17 **Log registration details**
07 March, 2011,2:12:17 Name: John Doe
07 March, 2011,2:12:17 Email: johndoe@gmail.com
07 March, 2011,2:12:17 Password: P@ssw0rd
07 March, 2011,2:12:17 Username: johndoe
07 March, 2011,2:12:17 International mobile: (+60)12-4567890
References


Wikibooks (2010) *PHP programming*, http://en.wikibooks.org/wiki/PHP, used under a Creative Commons Attribution-ShareAlike licence: http://creativecommons.org/licenses/by-sa/3.0/
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>An array is an ordered map that associates values to keys. It is an indexed collection of data values. Arrays can be used to organise data in a structured way that allows us to store, update, retrieve and sort groups of data items easily.</td>
</tr>
<tr>
<td>Binary-safe function</td>
<td>A binary-safe function is essentially one that treats its input as a raw stream of data without any specific format. Binary data is encoded as a string of 1s and 0s. Binary data needs to be handled differently; otherwise the data will become corrupt. A binary-safe function does not modify the binary data that it handles and thus does not corrupt the data.</td>
</tr>
<tr>
<td>Multidimensional array</td>
<td>An array that is created by using array elements to hold other arrays, which, in turn, can hold more arrays.</td>
</tr>
<tr>
<td>Padding</td>
<td>Padding is used for spacing-out strings to reduce clutter and enhance readability.</td>
</tr>
<tr>
<td>Sort</td>
<td>Segregate items into groups according to a specified criterion.</td>
</tr>
<tr>
<td>Traversal</td>
<td>Move from one array element to another and perform a task upon each element. Usually, traversal is performed using the internal array pointer.</td>
</tr>
<tr>
<td>Whitespace</td>
<td>The following characters are considered to be whitespace in PHP: spaces, tabs, newline, carriage return and NULL.</td>
</tr>
</tbody>
</table>
Unit 4

TCC 242/05
Web Database Application

Working with MySQL and PHP
Wawasan Open University is Malaysia's first private not-for-profit tertiary institution dedicated to adult learners. It is funded by the Wawasan Education Foundation, a tax-exempt entity established by the Malaysian People's Movement Party (Gerakan) and supported by the Yeap Chor Ee Charitable and Endowment Trusts, other charities, corporations, members of the public and occasional grants from the Government of Malaysia.

The course material development of the university is funded by Yeap Chor Ee Charitable and Endowment Trusts.

© 2011 Wawasan Open University. Except where otherwise noted, this work is licensed under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

This course material was published to support the learning of students registered with Wawasan Open University. Wawasan Open University does not grant any degree, certification or credits based solely on your completion of this course material.

Wawasan Open University
Jalan Sultan Ahmad Shah,
10050 Penang.
Tel: (604) 2180333  Fax: (604) 2279214
Email: enquiry@wou.edu.my
Website: www.wou.edu.my
# Contents

## Unit 4  Working with MySQL and PHP

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit overview</td>
<td>1</td>
</tr>
<tr>
<td>Unit objectives</td>
<td>1</td>
</tr>
<tr>
<td><strong>4.1 Working with relational databases</strong></td>
<td>3</td>
</tr>
<tr>
<td>Objectives</td>
<td>3</td>
</tr>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Relational database concepts</td>
<td>3</td>
</tr>
<tr>
<td>Modelling real-world objects</td>
<td>9</td>
</tr>
<tr>
<td>Avoiding redundant data</td>
<td>10</td>
</tr>
<tr>
<td>Single item single column</td>
<td>11</td>
</tr>
<tr>
<td>Avoiding empty attributes</td>
<td>12</td>
</tr>
<tr>
<td>Unique keys</td>
<td>12</td>
</tr>
<tr>
<td>Creating questions to ask a database</td>
<td>13</td>
</tr>
<tr>
<td>Suggested answers to activity</td>
<td>15</td>
</tr>
<tr>
<td><strong>4.2 Managing databases and user accounts</strong></td>
<td>17</td>
</tr>
<tr>
<td>Objectives</td>
<td>17</td>
</tr>
<tr>
<td>Introduction</td>
<td>17</td>
</tr>
<tr>
<td>Creating a database</td>
<td>17</td>
</tr>
<tr>
<td>Using a database</td>
<td>19</td>
</tr>
<tr>
<td>Deleting a database</td>
<td>19</td>
</tr>
<tr>
<td>Creating tables</td>
<td>19</td>
</tr>
<tr>
<td>Deleting a table</td>
<td>22</td>
</tr>
</tbody>
</table>
Altering a table

Setting up new accounts for database users

Granting access privileges

Managing account passwords

Suggested answers to activities

4.3 The bridge connecting MySQL and PHP

Objectives

Introduction

Connecting to MySQL server

Querying a database

Inserting data into a database

Retrieving data from a database

Updating records in a database

Deleting records from a database

Using phpMyAdmin

Suggested answers to activity

Summary of Unit 4

Unit practice exercises

Suggested answers to self-tests

Suggested answers to unit practice exercises

Appendix

References

Glossary
Unit Overview

This unit discusses techniques for manipulating MySQL databases. You will be shown how to create database records and manage user accounts in MySQL server. In addition, you will learn how PHP interacts with MySQL server.

The first section describes the key concepts of relational databases. It explains the basic principles of data modelling and database design. The second section discusses database tables. It also describes MySQL access privileges and how to set up new accounts for database users. The third section discusses how to connect to MySQL server with PHP. It explains the use of PHP to query a database in order to insert, retrieve, modify and delete database records.

Unit Objectives

By the end of this unit, you should be able to:

1. Describe relational database design concepts.
2. Create MySQL relational databases.
3. Manage user accounts in MySQL databases.
4. Access data in MySQL databases.
5. Connect a PHP web application to MySQL server.
6. Use MySQL library functions to query databases.
4.1 Working with Relational Databases

Objectives

By the end of this section, you should be able to:

1. Explain the concept of relational databases.
2. Model data with real-world objects.
3. Explain how data in relational databases are represented as collections of records.
4. Discuss the key principles of relational database design.

Introduction

You should by now have a good understanding of PHP. It is now time for you to move on to the next stage of creating a web database application, which is the task of integrating MySQL with PHP. You were introduced to MySQL in Unit 1 and shown how to install and configure MySQL on your computer. MySQL is a type of relational database server. In this section, you will first learn the fundamental concepts of relational databases.

Relational database concepts

A relational database is a database that organises its data into collections of tables, attributes, records, values, keys, schemas and relationships. Predicate logic is used to both describe the information contained in the database and to query it.

There are many advantages of storing and accessing data in a relational database compared to storing and accessing data in a flat file. One of the main advantages of a relational database is its built-in access privilege system. A relational database also provides faster access to data and simplifies querying. For example, a relational database in a human resource management application allows you to easily obtain data about employees, such as where they are from and what department they are in. You will next be introduced to the elements in the structure of a relational database.
Tables

Tables (also known as relations) are sets of data values that make up relational databases. In Figure 4.1, the sample table called EMPLOYEES stores employee ID numbers, names, departments and locations.

<table>
<thead>
<tr>
<th>EMPLOYEES</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeID</td>
</tr>
<tr>
<td>168</td>
</tr>
<tr>
<td>787</td>
</tr>
<tr>
<td>251</td>
</tr>
</tbody>
</table>

Figure 4.1 Employee data stored in a table

Attributes

Attributes or fields are the columns in a table. They have unique names and distinctive data types. For example, in Figure 4.1 the EmployeeID attribute is an integer whereas the other three attributes are strings.

Records

Records or tuples are rows in a table. In the previous example, each record in the table represents different attributes of an employee. Since the information is in a tabular format, all the records have the same attributes.

Values

Records consist of a set of values that correspond to the attributes. Each value has a fixed data type specified by its attribute.

Keys

A key is an attribute that has a unique value for each row in a table. For example, in Figure 4.1, we can take EmployeeID as the key for the table. We select EmployeeID because it is unique for each employee and will guarantee that information will not be repeated. We know that EmployeeID 168 belongs to “John Doe” in the US. Therefore, EmployeeID 168 cannot be assigned to another person. This unique key is also known as the primary key for that table. You should not take Location as the primary key since another employee, namely “Alice Carol”, is also located in the US. In this case, Name, Department and Location are not unique for each employee.
You will understand tables, attributes, records, values and keys better when you start creating tables in later sections.

A typical database consists of multiple tables. These tables can be interlinked using a key. **Figure 4.2** shows a second table called BONUS PAY, which has been added into the database that contains the EMPLOYEES table. It stores the amounts of the bonus paid to employees. By referring to the EmployeeID, we can see the amount of the bonus each employee has received. For example, the employee with EmployeeID 168, “John Doe”, received $2500 in bonuses.

Although EmployeeID is the primary key in the EMPLOYEES table, it is called a **foreign key** in the BONUS PAY table. The BONUS PAY table has its own primary key, namely TransactionID. In a relational database, a foreign key is used by a table to reference other tables that have related information.

![Figure 4.2 Linking to a related table with a foreign key](image)

### Schemas

A database schema is the design of a database. It can be considered to be the blueprint that is used to create the database. A schema defines the tables, attributes and relationships in a database. A schema will not usually include any data values. The database schema for the example shown in **Figure 4.2** can be represented in text form as shown below:

```sql
employees (EmployeeID, Name, Department, Location)
bonus_pay (TransactionID, EmployeeID, Amount, Date)
```

Note that the underlined attributes are the primary and foreign keys of each table.
Relationships

There are three types of relationships in a relational database, namely:

1. one-to-one
2. one-to-many
3. many-to-many

A one-to-one relationship means that one record in a table corresponds to only one record in a related table. For example, we could have placed the Location field in the EMPLOYEES table of Figure 4.1 in another table called LOCATION. That would create a one-to-one relationship between the two tables. Figure 4.3 shows this one-to-one relationship.

![Figure 4.3 One-to-one relationship](image)

A one-to-many relationship is a relationship where one record in a table is linked to many records in another table. In the example shown in Figure 4.2, one particular employee received two bonuses — EmployeeID 168 appears twice in the BONUS PAY table. Figure 4.4 shows this one-to-many relationship.
A many-to-many relationship is a relationship in which many records in a table are linked to many other records in another table. For example, looking at the two tables EMPLOYEES and TRAINING shown in Figure 4.5, you will see that one employee can attend several training courses. Conversely, each of the training courses can be attended by a number of employees. This many-to-many relationship can be represented by a new table called EMPLOYEE_TRAINING. This table will contain only the foreign keys from the EMPLOYEES and TRAINING tables to show which employee has attended which training course.
### Activity 4.1

1. What is a relational database?

2. ________ or fields are columns in a table. Each has a unique name and contains data.

3. Records or tuples are ________ in a table. Each record in a table consists of a number of attributes.
4. A relational database is a database that organises its data into collections of ___________.

A. statements, identifiers, keywords and expressions  
B. primary keys and foreign keys  
C. tables, attributes, records, values, keys, schemas and relationships  
D. one-to-one, one-to-many and many-to-many relationships

5. A database schema is the set of table designs for a database. A schema shows the attributes of each table and indicates the primary key and any foreign key it possesses. True / False?

6. Create a schema based on the table below, where FlightID is the primary key and Code is the foreign key.

<table>
<thead>
<tr>
<th>FlightID</th>
<th>Code</th>
<th>Place</th>
<th>Time</th>
<th>Fare</th>
<th>DepartureGate</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA08</td>
<td>PEN</td>
<td>Penang</td>
<td>20:00</td>
<td>RM176</td>
<td>B1</td>
</tr>
<tr>
<td>WA88</td>
<td>KUL</td>
<td>Kuala Lumpur</td>
<td>21:00</td>
<td>RM83</td>
<td>B2</td>
</tr>
<tr>
<td>WA77</td>
<td>LGK</td>
<td>Langkawi</td>
<td>08:00</td>
<td>RM205</td>
<td>B1</td>
</tr>
</tbody>
</table>

Modelling real-world objects

When creating a database, you will need to model real-world items and their relationships. Once you have a clear idea of how each item is related to the other items, you can store that information in a data model. A data model is a set of rules and conventions that assist in transforming loosely defined relationships in the real world into a finite model by means of reusable concepts (Relational Database Design 2010).

In general, each class of the real-world items (e.g., employees, trees, vehicles) that you model will need a table of its own. For example, if you want to store information about employees, you will need to identify the attributes of an employee which are common to all employees. Only then will you be able to model the employees table. Figure 4.1, Figure 4.2 and Figure 4.5 show how data about employees are modelled in terms of general details, bonus pay and training.

For the real-world object that we just mentioned (i.e., employees), you can apply the data modelling concept by creating at least three tables in the database, namely EMPLOYEES, BONUS PAY and TRAINING.
Avoiding redundant data

At this point you might be wondering why general information about employees such as name, department and location cannot be stored in the BONUS PAY table. Figure 4.6 shows what happens when you actually try do this. The shaded columns in the table contain redundant data.

<table>
<thead>
<tr>
<th>TransactionID</th>
<th>Amount</th>
<th>Date</th>
<th>EmployeeID</th>
<th>Name</th>
<th>Department</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>3000.00</td>
<td>1-Aug-2011</td>
<td>168</td>
<td>John Doe</td>
<td>Accounting</td>
<td>US</td>
</tr>
<tr>
<td>14</td>
<td>3000.00</td>
<td>1-Sep-2011</td>
<td>168</td>
<td>John Doe</td>
<td>Accounting</td>
<td>US</td>
</tr>
<tr>
<td>17</td>
<td>3000.00</td>
<td>1-Oct-2011</td>
<td>168</td>
<td>John Doe</td>
<td>Accounting</td>
<td>US</td>
</tr>
<tr>
<td>18</td>
<td>3000.00</td>
<td>1-Nov-2011</td>
<td>168</td>
<td>John Doe</td>
<td>Accounting</td>
<td>US</td>
</tr>
</tbody>
</table>

**Figure 4.6** Redundant data in the BONUS PAY table

From Figure 4.6, you can see that the name, department and location data of the employee named “John Doe” are repeated a number of times in the BONUS PAY table. As your list of employees grows, and since they will be paid bonuses every year, there will be a lot of duplicated information in your table as time goes by. This will take up a lot of space in your database. Therefore, you need to avoid storing duplicate or redundant data so that the performance and efficiency of the database is optimised.

Redundant data are also subject to update anomalies, which are situations where inconsistent data are created when a database is updated. This violates the integrity of the data, which leads us being unsure of the correctness of the data (Welling 2008). For example, if “John Doe” moves to a new location, you will have to update the Location attribute for all the transactions that appear in the BONUS PAY table shown in Figure 4.6. In this case, the BONUS PAY table has four rows stating John Doe’s location and you will thus have to update the table four times. Besides that, you will need to ensure that all the Location values are consistent throughout the table. For example, different ways of recording “US”, such as writing “United States”, will cause data conflicts for “John Doe”. In addition, removing transactions involving John Doe from the BONUS PAY table means that you risk losing his location information. In general, you want to design your database so that it is free from update anomalies.
Single item single column

You should store only a single item in each column. This means that you should only store one item in each attribute of each record. For example, if you want to track how many training sessions an employee has attended, you may add a column called TrainingAttended to the EMPLOYEES table as shown in Figure 4.7 and store the training IDs in this column.

<table>
<thead>
<tr>
<th>EMPLOYEES</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeID</td>
</tr>
<tr>
<td>168</td>
</tr>
<tr>
<td>787</td>
</tr>
<tr>
<td>251</td>
</tr>
</tbody>
</table>

*Figure 4.7* The TrainingAttended attribute in each record has multiple values

However, note that by doing this you are actually nesting a whole table into one column. This will lead to problems when you want to answer questions like “How many employees have attended project management (TrainingID=83) training sessions?” If your table is set up as shown in Figure 4.7, the TrainingAttended field in each record will have to be separately analysed to see if a matching TrainingID is inside it.

You should instead create a new table to hold the training data. For example, you can create a new table called EMPLOYEE_TRAINING as shown in Figure 4.8. This new table makes it easier to count the number of employees who have attended a particular training session.

<table>
<thead>
<tr>
<th>EMPLOYEE_TRAINING</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeID</td>
</tr>
<tr>
<td>168</td>
</tr>
<tr>
<td>251</td>
</tr>
<tr>
<td>787</td>
</tr>
<tr>
<td>787</td>
</tr>
<tr>
<td>251</td>
</tr>
</tbody>
</table>

*Figure 4.8* The EMPLOYEE_TRAINING table
Avoiding empty attributes

There are two ways to add a non-mandatory or optional attribute to a table. One way would be to add a new column to the table. For example, you can add an additional column called Note to the TRAINING table.

<table>
<thead>
<tr>
<th>TRAINING</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrainingID</td>
</tr>
<tr>
<td>81</td>
</tr>
<tr>
<td>82</td>
</tr>
<tr>
<td>83</td>
</tr>
</tbody>
</table>

![Figure 4.9 Null value in the Note column](image)

However, if you have many records in the TRAINING table and if there is only a small chance that a note is written in a record, the table will end up with many empty attributes or “null values” in the Note column. This will waste storage space and the null values may also cause errors when used in numerical functions like those that calculate totals. A null value in a table cell also creates doubts as it is not known whether the cell was intentionally left empty or if data has not been entered yet.

To avoid this problem, you can create a new table called TRAINING_NOTES. This table will only store the training courses that have a note attached to them. The training courses that have no notes will not be stored in this table so that the creation of null values is avoided.

<table>
<thead>
<tr>
<th>TRAINING_NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrainingID</td>
</tr>
<tr>
<td>81</td>
</tr>
<tr>
<td>82</td>
</tr>
<tr>
<td>83</td>
</tr>
</tbody>
</table>

![Figure 4.10 Dedicated table for training notes](image)

Unique keys

You should make sure that the primary key you select is really unique. In the previous examples, primary keys were created for the EMPLOYEES (EmployeeID), BONUS PAY (TransactionID) and TRAINING (TrainingID) tables. This was done because these objects do not have a unique identifier that can ensure that each record is unique. For tables like EMPLOYEE_TRAINING, you do not need to create a unique identifier since both the foreign keys (EmployeeID and TrainingID) found in EMPLOYEE_TRAINING can be treated as unique identifiers.
Creating questions to ask a database

An important database design principle is to always create questions that you want to ask the database that you are designing. As long as the database that you are designing contains the relationships that can answer these questions, the design is considered to be good.

Web Reference

To learn more about designing relational databases, please visit:


Summary

A relational database is a database that organises its data into collections of tables, attributes, records, values, keys, schemas and relationships.

When creating a database, you can model it on real-world items and their relationships. You can store information about the items and relationships in data models. In general, you have to design your database so that it is free from update anomalies. The fundamental concept is to store only one item in each record field. You should create a new table instead of storing a nested table in a column. You should also avoid having too many null values in your table. Always select the right unique key and always create questions that you want to ask the database that you are designing.

In the next section, you will learn how to create and work with databases and user accounts.
Self-test 4.1

1. What are update anomalies?

2. For some many-to-many relationship tables, you do not need to create a unique identifier since the foreign keys can be treated as unique identifiers. True / False?

3. One of the principles in designing a relational database is to
   A. always assign a unique key to a table immediately.
   B. always insert records into the database.
   C. always create tables in the database.
   D. always create questions that you want to ask the database.

4. Consider the following tables that respectively show the various types of camels and the countries with the highest camel populations in the world.

   **CAMEL**
<table>
<thead>
<tr>
<th>TypeID</th>
<th>CamelType</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Dromedary</td>
</tr>
<tr>
<td>T2</td>
<td>Bactrian</td>
</tr>
<tr>
<td>T3</td>
<td>Wild Bactrian</td>
</tr>
<tr>
<td>T4</td>
<td>F1 Hybrid</td>
</tr>
<tr>
<td>T5</td>
<td>F2 Hybrid</td>
</tr>
</tbody>
</table>

   **POPULATION**
<table>
<thead>
<tr>
<th>Code</th>
<th>Country</th>
<th>Population</th>
<th>CamelTypesFound</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>Somalia</td>
<td>3,860</td>
<td>T1, T3, T5</td>
</tr>
<tr>
<td>SUD</td>
<td>Sudan</td>
<td>14,660</td>
<td>T2, T4, T5</td>
</tr>
<tr>
<td>MAU</td>
<td>Mauritania</td>
<td>37,000</td>
<td>T1, T5</td>
</tr>
</tbody>
</table>

   a. Can you identify the design problem in the POPULATION table?

   b. How can you resolve the problem?
5. Consider the following table that shows the countries with the highest camel populations in the world, which has a record field for the total amount of donations received by each country.

<table>
<thead>
<tr>
<th>Code</th>
<th>Country</th>
<th>Population</th>
<th>Donation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>Somalia</td>
<td>3,860</td>
<td></td>
</tr>
<tr>
<td>SUD</td>
<td>Sudan</td>
<td>14,660</td>
<td></td>
</tr>
<tr>
<td>MAU</td>
<td>Mauritania</td>
<td>37,000</td>
<td></td>
</tr>
</tbody>
</table>

a. Can you identify the design problem in the POPULATION table?

b. How can you resolve the problem?

Suggested answers to activity

Feedback

Activity 4.1

1. A relational database is a database that organises its data into collections of tables, rows, attributes and domains. Predicate logic is used to both describe the information contained in the database and to query information from it.

2. Attributes

3. rows

4. C

5. True

6. WAIR_FLIGHT (FlightID, Code, Place, Time, Fare, DepartureGate)
4.2 Managing Databases and User Accounts

Objectives

By the end of this section, you should be able to:

1. Create databases and relation structures.
2. Create tables based on schemas.
3. Delete and alter tables.
4. Describe the MySQL access privilege system.
5. Perform user authentication from a given host.
6. Manage and assign account passwords.

Introduction

In this section, you will learn how to manage databases and user accounts in MySQL server via the MySQL monitor. You will reuse the data model that was discussed in the previous section. Here is the schema of that data model.

```
employees (EmployeeID, Name, Department, Location)
bonus_pay (TransactionID, EmployeeID, Amount, Date)
training (TrainingID, Trainer, CourseName, Date)
employee_training (EmployeeID, TrainingID)
training_notes (TrainingID, Note)
```

The database that you will create with this schema will be called “human_resource”.

Creating a database

You will now create a database in MySQL server via the MySQL monitor. In Unit 1, you learnt how to access the MySQL monitor. To run MySQL monitor, open the terminal window in Ubuntu. In Windows go to Start > Run and then type cmd.

1. At the terminal prompt, type the following to enter the MySQL monitor:

```
mysql -h localhost -u root -p
```
2. You will be prompted to enter a password. Enter *wawasan*, which is the password you set earlier.

3. You will then be in the MySQL monitor and the mysql> prompt will be displayed. Remember that all commands used here will end with a semicolon ";" or a "\g". View your pre-installed databases by typing `show databases;`.

![MySQL Monitor showing pre-installed databases](image)

Figure 4.11 MySQL Monitor showing pre-installed databases

Note that permission to view or use databases on MySQL server is based on the user’s privileges in the MySQL Access Privilege System. We will discuss this topic later in the unit. Since you login as the `root`, you will have all the administrative privileges including the ability to create a new database.

To create a database, you will use the `CREATE DATABASE` command. The syntax of the command is:

```
CREATE DATABASE db_name;
```

For example, to create a database called `human_resource`, use the following syntax:

```
mysql> CREATE DATABASE human_resource;
```

Once you run the command, you will receive a confirmation message:

```
Query OK, 1 row affected (0.03 sec).
```

You can verify whether the database has been created by typing:

```
show databases;
```
Using a database

After the database is created, you can set it as your default working database by using the `USE` command:

```
mysql> USE human_resource;
```

After issuing the `USE` command, you will receive a confirmation message:

```
Database changed
```

You may also directly log in to the database through MySQL monitor by indicating its name in the following command line:

```
%> mysql -h localhost -u root -p human_resource
```

Deleting a database

You may remove a database that you created by using the `DROP` command. The syntax of the command is:

```
DROP DATABASE db_name;
```

To remove the database named `human_resource`, use the following command:

```
mysql> DROP human_resource;
```

Creating tables

In this section, you will learn how to create tables using the schemas in the `human_resource` database. If you have removed it in the previous subsection, please re-create the database.

Before you work on the database, make sure that you are using the `human_resource` database by running the `USE` command:

```
mysql> USE human_resource;
```

A table is created within a database by using the `CREATE TABLE` statement. The syntax of the `CREATE TABLE` statement is:

```
CREATE TABLE table_name (  
col_name data_type  
[NOT NULL | NULL]  
[AUTO_INCREMENT] [PRIMARY] KEY],  
...);
```
Based on the schema `employees(EmployeeID, Name, Department, Location)`, the following command will create a table called `employees`:

```
mysql> CREATE TABLE employees (  
    EmployeeID INT NOT NULL PRIMARY KEY,  
    Name VARCHAR(50) NOT NULL,  
    Department VARCHAR(20) NOT NULL,  
    Location VARCHAR(40) NOT NULL );
```

The command that you used above names the first column or field of the table `EmployeeID`, assigns to the field the integer (`INT`) data type, and states that it must not be a null value (`NOT NULL`). It is assigned as the `PRIMARY KEY`. The other three fields have the string data type (`VARCHAR`). `VARCHAR(50)` means that the field can store a variable-length string with a maximum of 50 characters.

If you are currently using another database and would like to create a table in `human_resource`, simply type the table name with the target database name like this:

```
human_resource.employees
```

View the table that you created by using the `SHOW TABLES` command:

```
mysql> SHOW TABLES;
```

The output of this command will be something like the following:

```
+--------------------------+
| Tables_in_human_resource |
+--------------------------+
| employees                |
+--------------------------+
1 row in set (0.00 sec)
```

The output above does not describe much about the structure of the table. To view the structure in detail, you need to use the `DESCRIBE` command:

```
mysql> DESCRIBE employees;
```
This command will display the following information:

+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
|-------------+-------------+------|-----|--------+-------+
| EmployeeID  | int(11)     | NO   | PRI | NULL   |       |
| Name        | varchar(50) | NO   |     | NULL   |       |
| Department  | varchar(20) | NO   |     | NULL   |       |
| Location    | varchar(40) | NO   |     | NULL   |       |
+-------------+-------------+------|-----|--------+-------+

4 rows in set (0.00 sec)

Now create the `bonus_pay` table based on the schema: `bonus_pay (TransactionID, EmployeeID, Amount, Date). The statement to do this is:

```sql
mysql> CREATE TABLE bonus_pay (  
    TransactionID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    EmployeeID INT NOT NULL,  
    Amount FLOAT(10,2) NOT NULL,  
    Date DATE NOT NULL );
```

The `TransactionID` field, which is the primary key for this table, is created with the `AUTO_INCREMENT` facility. `AUTO_INCREMENT` allows a unique number to be auto-generated when you insert a new record into the table. By default, the starting value for `AUTO_INCREMENT` is 1 and it will increment by 1 for each new record.

The data type of `Amount` is `FLOAT(10,2)`, which means that the values of this field can have a maximum of 10 digits with 2 decimal places. The `Date` field has the `DATE` data type.

Create the other tables in the `human_resource` database using the following schemas:

```sql
training (TrainingID, Trainer, CourseName, Cost)
employee_training (EmployeeID, TrainingID)
training_notes (TrainingID, Note)
```

You can use the statements shown below to create these tables.

```sql
mysql> CREATE TABLE training (  
    TrainingID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    Trainer VARCHAR(50) NOT NULL,  
    CourseName VARCHAR(40) NOT NULL,  
    Cost FLOAT(10,2) NOT NULL );
```
mysql> CREATE TABLE employee_training (  
    EmployeeID INT NOT NULL,  
    TrainingID INT NOT NULL,  
    PRIMARY KEY (EmployeeID, TrainingID));

mysql> CREATE TABLE training_notes (  
    TrainingID INT NOT NULL PRIMARY KEY,  
    Note TEXT );

Note that when creating the employee_training table, you indicated multiple fields of primary keys by using the special PRIMARY KEY clause in the last line:

    PRIMARY KEY (EmployeeID, TrainingID));

In this clause, the fields are separated by using a comma. Lastly, the data type of Note is TEXT, which is used for accepting long strings such as text documents.

Instead of running the CREATE TABLE statements one by one in MySQL monitor, you can store all the CREATE TABLE statements in one external SQL file and execute them together. Creating the external SQL file has a few advantages. One advantage is that you can edit the commands in a text editor before you execute them; doing this will minimise errors. Another advantage is that you can reuse the file to create the same tables in another database.

To create an SQL file, copy all the CREATE TABLE statements into a file called 'hr_tbls.sql'. Then exit MySQL monitor and execute the following command from the Linux terminal or Windows command prompt:

    mysql -h localhost -u root -D human_resource -p < hr_tbls.sql

You may want to remove all the existing tables in the human_resource database prior to trying out the above command. Refer to the next section on how to remove tables.

Note: The Appendix in this unit has the complete contents of 'hr_tbls.sql'.

Deleting a table

You can remove (delete) a table in a database by using the DROP TABLE command. For example, if you want to remove the table called employees, you will execute the following command:

    DROP TABLE employees;
You can remove several tables at the same time by separating the table names with commas in the `DROP TABLE` command:

```
DROP TABLE employees, bonus_pay, training;
```

**Altering a table**

Often, you will need to revise and improve upon the structures of the tables in your database. You do not have to delete and re-create a table each time you need to make a change to its structure. Instead, you can change the table structure with the `ALTER TABLE` statement.

For example, if you want to add an additional `Address` field to the `employees` table, you can use the `ADD COLUMN` clause to indicate the new field name and data type:

```
ALTER TABLE employees ADD COLUMN Address VARCHAR(200);
```

By default, the new field that you inserted will be placed in the last position in the table. If you want to specify the position of the new field, you can place the positioning keywords `FIRST` or `AFTER` directly after a field name:

```
ALTER TABLE employees ADD COLUMN Address VARCHAR(200) AFTER Department;
```

You can use the `CHANGE` clause to modify field properties. In case you forgot to insert the `NOT NULL` clause to the new field, you can modify the new field by using the following statement:

```
ALTER TABLE employees CHANGE Address VARCHAR(200) NOT NULL;
```

Another modification that you can make to a table is to set the auto increment to start from a value other than the default value 1. To set the `AUTO_INCREMENT` index in the `training` table to start from 80, you can use the following statement after creating the table:

```
ALTER TABLE training AUTO_INCREMENT = 80;
```

Lastly, you can remove a field from a table by using the `DROP` clause. For example, if you want to remove the `Address` field from the `employees` table, you can use the following statement:

```
ALTER TABLE employees DROP Address;
```
Web Reference

To learn more about managing databases and tables, please visit:


Activity 4.2

1. To create a database, you will use the __________ command.

2. Which of the following command(s) can be used to set a database called db_name as the default working database after it is created?

   A. mysql> USING db_name;
   B. mysql> CHANGE DATABASE db_name;
   C. %> mysql -h localhost -u root -p db_name
   D. All of the above

3. You may remove the database called db_name that you have created by using the __________ command.

   A. DELETE DATABASE db_name;
   B. DROP DATABASE db_name;
   C. REMOVE DATABASE db_name;
   D. All of the above

4. What are the option(s) that you can set when you create a new table with the CREATE TABLE statement?

   A. PRIMARY KEY;
   B. AUTO_INCREMENT;
   C. NOT NULL | NULL;
   D. All of the above

5. You can use the SHOW TABLES or DESCRIBE command to view tables that you have created. True / False?
6. Write an SQL statement to create the table below (without inserting the records). Take AtomicNumber as the primary key and allow both Symbol and ElementName to have a maximum length of 5 and 20 respectively. All fields must not be null.

<table>
<thead>
<tr>
<th>PERIODIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
</tr>
<tr>
<td>Li</td>
</tr>
<tr>
<td>Ti</td>
</tr>
<tr>
<td>Pb</td>
</tr>
</tbody>
</table>

Setting up new accounts for database users

Just as you would lock the door when you leave your house, you will need to secure the data in your database. Remember that the driving factor of any database-driven application is data. An organisation cannot operate if it loses its data. A database system like MySQL provides several security options for protecting your data. This section will describe the MySQL Access Privilege System. It will explain how you can create users, grant privileges and manage passwords.

The MySQL Access Privilege System protects your data from unauthorised access or modification. MySQL allows you to grant different access privileges to different users. For example, you may want to provide a clerk with only read access to certain table columns while providing an analyst with write access to those columns.

The MySQL Access Privilege System is based on the concepts of authentication and authorisation. Authentication involves connecting the user to the database server. Authorisation specifies what privileges the user has in executing queries. MySQL user account information, which enables authentication and authorisation to be performed, is stored in the `mysql` database, which is installed by default.

Let us now create a new MySQL account by using the `CREATE USER` statement. The syntax is as follows:

```
CREATE USER user [IDENTIFIED BY [PASSWORD] 'password'];
```

Create an account called “johndoe” at localhost with the password `wawasan`:

```
CREATE USER johndoe@localhost IDENTIFIED BY 'wawasan';
```

After the previous statement is executed, you will receive a confirmation message:

```
Query OK, 0 rows affected (0.00 sec).
```
You can then verify the current user list by entering:

   SELECT User, Host, Password FROM mysql.user;

You should be able to see an additional user called johndoe at localhost. Note that you were logged in as the root account and this means that you have the global CREATE USER privilege for the mysql database.

You can rename an existing MySQL account by using the RENAME USER statement. The general syntax is as follows:

   RENAME USER old_user TO new_user;

To rename the account “johndoe” at localhost to “babydoe”:

   RENAME USER johndoe@localhost TO babydoe@localhost;

You may remove an existing MySQL account by using the DROP USER statement. The general syntax is as follows:

   DROP USER user;

For example, to drop the account “babydoe” at localhost, you will type:

   DROP USER babydoe@localhost;

After using this statement, you should check the current user list to see whether the user “babydoe” has been removed.

**Granting access privileges**

Recall that the MySQL Access Privilege System allows you to grant different access privileges to different users. You can use the GRANT and REVOKE commands to manage access privileges. To illustrate the application of these commands, let us first create a user account.

Create a user account called johndoe@localhost:

   CREATE USER johndoe@localhost IDENTIFIED BY 'wawasan';
By default, the new user account that you have just created will have no privileges. You may grant privileges to a user by using the GRANT command. The general syntax is as follows:

```
GRANT privilege_type [(column_list)]
ON {table_name | * | *.* | database_name.*}
TO user_name [IDENTIFIED BY 'password'];
```

You will now reuse the `human_resource` database that you created earlier. Assume that you want to grant `johndoe@localhost` all user rights to the entire `human_resource` database. The following is the statement that you would use:

```
GRANT ALL ON human_resource.* TO johndoe@localhost;
```

`ALL` means all user rights including the right to select, insert, update and delete tables and records in the database. The asterisk (`*`) means all tables from `human_resource`.

You can create a new account and grant privileges at the same time by using the GRANT statement. For instance, the following statement grants `SELECT` and `INSERT` privileges to a new user account called `marydoe@localhost`, which has the password `wawasan`:

```
GRANT SELECT, INSERT ON human_resource.* TO marydoe@localhost
IDENTIFIED BY 'wawasan';
```

You may remove privileges from a user by using the REVOKE command. The syntax is as follows:

```
REVOKE privilege_type [(column_list)]
ON {table_name | * | *.* | database_name.*}
FROM user_name;
```

To remove the `INSERT` privilege granted earlier to `marydoe@localhost`:

```
REVOKE INSERT ON human_resource.* FROM marydoe@localhost;
```

You may view all the privileges granted to a user by using SHOW GRANTS command. The syntax is as follows:

```
SHOW GRANTS FOR user;
```
For example, to view all the privileges granted to johndoe@localhost and marydoe@localhost you will type:

```
SHOW GRANTS FOR johndoe@localhost;
SHOW GRANTS FOR marydoe@localhost;
```

![Figure 4.12 Viewing all the privileges granted to johndoe@localhost and marydoe@localhost](image)

You may also view the privileges of a currently logged-in user by using the `CURRENT_USER()` command as follows:

```
SHOW GRANTS FOR CURRENT_USER();
```

### Managing account passwords

During user account creation, you have the option to assign or not assign a password to the user account. If you are the administrator creating a new user account, you would expect the default password that you generated to be eventually overwritten. You may assign or change a password for an existing account by using the `SET PASSWORD` statement. The syntax is as follows:

```
SET PASSWORD [FOR user] = { PASSWORD('some password')
  | 'encrypted password');
```

To change the password for marydoe@localhost to wawasan2:

```
SET PASSWORD FOR marydoe@localhost = PASSWORD('wawasan2');
```
You may also use `SET PASSWORD` to change a currently logged-in user's password by removing the `FOR` clause:

```
SET PASSWORD = PASSWORD('wawasan2');
```

Note that `SET PASSWORD` may be recorded in server logs or in a history file such as `~/.mysql_history`, which means that plain text passwords may be read by anyone having read access to that information (MySQL 5.5 Reference Manual 2011).

To avoid sending plain text passwords to the server logs, you may directly specify the hash value that `PASSWORD()` will return for your password. You can use the `SELECT PASSWORD(str)` statement to perform a one-way (irreversible) encryption. For example, to know the hash value for `wawasan2`, you can use the following statement:

```
SELECT PASSWORD('wawasan2');
```

This statement will return you the hash value:

```
*9CEA96BAEC56D8CB17C813A152A918DC23EB100.
```

Next, you can use this hash value to assign the hash value of your password directly by using the `SET PASSWORD` statement without the `PASSWORD()` function clause. To set the password for `marydoe@localhost` using the hash value of `wawasan2`:

```
SET PASSWORD FOR marydoe@localhost = '9CEA96BAEC56D8CB17C813A152A918DC23EB100';
```

Notice the difference between using the `SET PASSWORD` command with and without the `PASSWORD()` function clause. Similarly, you can assign the hash value when you use the `CREATE USER` or `GRANT` statements by specifying the hash value preceded by the keyword `PASSWORD`:

```
CREATE USER marydoe@localhost IDENTIFIED BY PASSWORD '9CEA96BAEC56D8CB17C813A152A918DC23EB100';
```

(Equivalent to: `CREATE USER marydoe@localhost IDENTIFIED BY 'wawasan2';`)

```
GRANT SELECT, INSERT ON human_resource.* TO marydoe@localhost IDENTIFIED BY PASSWORD '9CEA96BAEC56D8CB17C813A152A918DC23EB100';
```

(Equivalent to: `GRANT SELECT, INSERT ON human_resource.* TO marydoe@localhost IDENTIFIED BY 'wawasan2';`
Web Reference

The following web reference presents a comprehensive view of MySQL account management statements:


Activity 4.3

1. The MySQL Access Privilege System is based on the concepts of _________ and _________.

2. Write an SQL statement to create a user called william at localhost who has the password p@ssw0rd.

3. Write an SQL statement to rename the user william at localhost to bill at localhost.

4. You may remove privileges granted to a user by using the GRANT command. True / False?

5. Can you explain the result of the execution of the following SQL statement?

   GRANT SELECT ON army.* TO kenny@localhost IDENTIFIED BY 'p@ssw0rd';

6. Write an SQL statement to view the privileges of a currently logged-in user.
Summary

In this section, you learnt how to create a database in MySQL server via MySQL monitor. To create a database, you use the `CREATE DATABASE` command. After the database is created, you can set it as your default working database by using the `USE` command. You may remove a database by using the `DROP` command.

You also learnt how to create tables using schemas. You can create a table by using the `CREATE TABLE` statement. You can use the `SHOW TABLES` command or the `DESCRIBE` command to view the tables in a database. To remove a table, use the `DROP TABLE` command, and to change the table structure use the `ALTER TABLE` statement.

The MySQL Access Privilege System protects your data from unauthorised access or modification. You can create and manage a MySQL user account by using the `CREATE USER`, `RENAME USER` and `DROP USER` statements. You may grant privileges to a user by using the `GRANT` statement and remove privileges by using the `REVOKE` statement. You may set a password for an existing account by using the `SET PASSWORD` statement.

In the next section, you will learn how to connect to MySQL server using PHP.
Self-test 4.2

1. By using an SQL statement, create a database called `stock_market` and set it as your current working database.

2. Write an SQL statement to create the table shown below (without inserting the records). Take ID as the primary key and put it in auto_increment mode. Allow the Symbol and StockName fields to have maximum lengths of 10 and 20 respectively. The Price, Changes and PercentRate fields will have maximum lengths of 10, 8 and 6 respectively, and each will have 2 decimal places (float type fields). All fields must not be null.

<table>
<thead>
<tr>
<th>ID</th>
<th>Symbol</th>
<th>StockName</th>
<th>Price</th>
<th>Changes</th>
<th>PercentRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IXIC</td>
<td>Nasdaq</td>
<td>2,741.83</td>
<td>−29.68</td>
<td>−1.07</td>
</tr>
<tr>
<td>2</td>
<td>N225</td>
<td>Nikkei 225</td>
<td>9,555.26</td>
<td>−164.44</td>
<td>−1.69</td>
</tr>
<tr>
<td>3</td>
<td>HSI</td>
<td>Hang Seng</td>
<td>23,976.37</td>
<td>−326.70</td>
<td>−1.34</td>
</tr>
</tbody>
</table>

3. Based on question 2 above, write an SQL statement to insert a new column called Region (as shown below), between the ID field and the Symbol field. Allow Region to have a maximum length of 15. This field can accept null values.

<table>
<thead>
<tr>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
</tr>
<tr>
<td>Asia</td>
</tr>
<tr>
<td>Asia</td>
</tr>
</tbody>
</table>

4. After the SQL statements shown below are executed, which user account(s) will still be in the database?

```
CREATE USER joseph@localhost IDENTIFIED BY 'p@ssw0rd';
CREATE USER patrick@localhost IDENTIFIED BY 'p@ssw0rd';
CREATE USER alex@localhost IDENTIFIED BY 'p@ssw0rd';
RENAME USER patrick@localhost TO patrick2@localhost;
DROP USER joseph@localhost, alex@localhost;
```
5. Write SQL statements to grant all user rights in a database called `accounting` to an existing user, `leeann@localhost`, and grant read-only permission to a new user, `maria@localhost`, who has the password `p@ssw0rd`.

6. Based on the database in question 5, write SQL statements to remove insertion and updating rights from `leeann@localhost` and to display the privileges granted to both `leeann@localhost` and `maria@localhost`.

7. As a root user, how would you change the password for an existing user named `leeann@localhost` to `12345678` using a MySQL statement? In addition, use an SQL statement to display the hash value for `12345678`.

8. You are given the following password and its hash value:

```
+-------------------------------------------+
| PASSWORD('p@ssw0rd') |
+-------------------------------------------+
| *D7E39C3AF517EC9EF7086223B036E0B4F22821F8 |
+-------------------------------------------+
```

What is the difference between the two SQL statements shown below?

a. CREATE USER doe@localhost IDENTIFIED BY 'p@ssw0rd';

b. CREATE USER doe@localhost IDENTIFIED BY PASSWORD 'D7E39C3AF517EC9EF7086223B036E0B4F22821F8';

9. Which of the following statements is an invalid way to set a password?

A. CREATE USER william@localhost IDENTIFIED BY 'p@ssw0rd';
B. RENAME USER bill@localhost TO william@localhost IDENTIFIED BY 'p@ssw0rd';
C. SET PASSWORD FOR william@localhost = PASSWORD ('p@ssw0rd');
D. REVOKE ALL ON database.* FROM william@localhost IDENTIFIED BY 'p@ssw0rd';
Suggested answers to activities

Feedback

Activity 4.2

1. CREATE DATABASE

2. C

3. B

4. D

5. True

6. CREATE TABLE PERIODIC (
   Symbol VARCHAR(5) NOT NULL,
   ElementName VARCHAR(20) NOT NULL,
   AtomicNumber INT NOT NULL,
   PRIMARY KEY (AtomicNumber));

Activity 4.3

1. authentication and authorisation

2. CREATE USER william@localhost IDENTIFIED BY 'p@ssw0rd';

3. RENAME USER william@localhost TO bill@localhost;

4. False

5. All tables in the army database will provide read-only privilege to the user account called kenny@localhost which has the password p@ssw0rd.

6. SHOW GRANTS FOR CURRENT_USER();
4.3 The Bridge Connecting MySQL and PHP

Objectives

By the end of this section, you should be able to:

1. Describe the process of connecting to MySQL server with PHP.
2. Describe the PHP syntax used in querying databases.
3. Use PHP to implement the insertion, selection, modification and deletion of records in MySQL databases.
4. Install and use the web-based MySQL administration tool.

Introduction

MySQL has been supported by PHP since its early versions. These two prominent technologies have been used together to create powerful, dynamic database-driven web applications.

You will be using MySQL Improved or mysqli, which comes with the updated MySQL extension in PHP 5. The mysqli extension offers all of the functionality of its predecessor mysql, in addition to some new features. You will be introduced to how the mysqli extension is used to connect to the database server, and to run queries to get data and perform modification tasks. You will start by learning how to connect to MySQL server with PHP script.

Connecting to MySQL server

To connect to MySQL server, you will need to use the host name, user name and password connection variables. In this section, you will reuse the user account that you created in the previous section. It has the following connection variables:

<table>
<thead>
<tr>
<th>Host name</th>
<th>User name</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost</td>
<td>johndoe</td>
<td>wawasan</td>
</tr>
</tbody>
</table>

Figure 4.13 Connection variables
You will use the PHP `mysqli_connect` function to create a connection command. The connection command will open a new connection to MySQL server. The general syntax of the command is as follows:

```php
mysqli_connect([string $host [, string $username [, string $passwd [, string $dbname [, int $port, [string $socket]]]]]])
```

You will supply four parameters to this function: the server host name, the user name, the user password and the name of the database. Since you are setting up on the localhost server, your server host name is `localhost`. If you are running a web server on one system and MySQL on another system, you can replace `localhost` with the IP address or domain name of the system MySQL resides on. The `mysqli_connect` function returns a connection object that you can use to communicate with the database.

In keeping with the object-oriented paradigm, the new `mysqli` extension is encapsulated into a class. As a result, you need to first instantiate the `mysqli` class by using the `new` keyword. You will store the connection object in a variable called `$mysqli`:

```php
$mysqli = new mysqli("localhost", "johndoe", "wawasan", "human_resource");
```

Here, you are trying to establish a connection with the MySQL server. If you want to be notified of an error so that you can stop your program from executing further, you can retrieve error codes and error messages using the following predefined syntax of the `mysqli` class:

```php
class mysqli {
    int connect_errno
    string connect_error
}
```

The following PHP code attempts to connect to MySQL server and retrieves error codes and messages if an error occurs during the connection attempt:

```php
$mysqli = new mysqli("localhost", "johndoe", "wawasan", "human_resource");
if ($mysqli->connect_errno) {
    printf("Unable to connect to the database:<br /> %s", $mysqli->connect_error);
    exit();
}
```
For example, if an incorrect password is provided, you will see the following message and the program will be terminated by the `exit()` function:

```
Unable to connect to the database:
Access denied for user 'johndoe'@'localhost' (using password: YES)
```

### Note: Invoking fields

You may have seen the operator "->" in the object-oriented context. This operator is used to invoke fields. This is different from the variables that are prefixed with a dollar sign. An object is correlated with its specific field’s value and is referenced like this:

```php
$obj -> field
```

Since the `mysqli` class includes the fields `connect_errno` and `connect_error`, that is how you get the referred fields below under the object `$mysqli`:

```php
$mysqli->connect_errno
$mysqli->connect_error
```

After you have finished interacting with the database, it is always good practice to close the connection at the end of the script. To close the MySQL connection, you will use the `close()` function as shown below:

```php
$mysqli->close();
```

### Querying a database

We can use the PHP `query()` method to query a database. Queries involve data insertion, retrieval, update and deletion. The `query()` method’s syntax is as follows:

```php
class mysqli {
    mixed query(string query [, int resultmode])
}
```

The optional `resultmode` parameter accepts two values:

1. **MYSQLI_STORE_RESULT**: The default setting that returns the result as a buffered set and ensures that all data are available for retrieval.

2. **MYSQLI_USE_RESULT**: Returns the result as an unbuffered set; the data will be retrieved as needed from the server.
After you have set up the query (it could be an insertion, retrieval, update or deletion), you need to send the query to MySQL by running the PHP statement below and assigning it to the $result variable. By default, you will use the MYSQLI_STORE_RESULT setting to get the full buffered set since you do not have a large resulting data set:

```php
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
```

Just like when you establish a connection with MySQL server, you can be notified if an error happens during the execution of a query so that you can stop your program from executing further. For successful queries, $mysqli->query will return the resulting object, whereas for a failed query it will return FALSE. Since we know that $mysqli->query or $result returns FALSE on failure, we can use the following statement to catch the error:

```php
if(!$result) {
    echo($mysqli->error);
    exit();
}
```

### Inserting data into a database

You have created a database and the table structures. You have also established a connection to MySQL server. However, at this stage your database is still empty since no records have been inserted yet. You will now learn how to store data into the database that you have created.

You can insert data into a database by using the SQL `INSERT` command. The syntax is as follows:

```
INSERT [INTO] table [(column1, column2, column3, ...)] VALUES (value1, value2, value3,...);
```

You will now insert some data into the `human_resource` database's `employees` table by using the SQL `INSERT` command and the PHP code shown below:

```php
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
```
Notice that the string value that you inserted is enclosed by a pair of quotes but for numbers or dates, you do not need them. In this example, you filled up the table columns by following the order given in the schema below:

```
employees (EmployeeID, Name, Department, Location)
```

If you want to put data into specific columns, you can list the columns and their corresponding values. For example, to store only EmployeeID and Name:

```
$query = "INSERT INTO employees (EmployeeID, Name) VALUES (168, 'John Doe')";
```

or

```
$query = "INSERT INTO employees SET
    EmployeeID = 168,
    Name = 'John Doe'";
```

Note that running the INSERT command above may cause an error to occur because in the existing employees table you indicated that the Department and Location fields are not to be NULL. To run the command successfully, you need to alter these fields to accept NULL values.

You can also insert multiple records into a table at the same time. Each row is placed in its own set of brackets and is separated by a comma. You can place all the INSERT statements into an SQL file just like the one you created in the previous section for the CREATE statements. For convenience, we have put together an SQL INSERT script that populates the database. This script can be found in the file 'hr_tbls.sql', which is given in the Appendix. This SQL script is used in the same way as was described in the previous section.

### Retrieving data from a database

To retrieve data from a database, you can use the SQL SELECT command. The general syntax of this command is as follows:

```
SELECT items FROM tables [WHERE condition] [GROUP BY group_type]
[HAVING where_definition] [ORDER BY order_type];
```

The following statements perform a simple query that retrieves data from the Name and Department fields in the employees table:

```
$query = "SELECT Name, Department FROM employees";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
```
You can iterate through the result set by using a `while` loop with the `list` and `fetch_row()` functions and print out the results with the `echo` function:

```php
while(list($name, $dept) = $result->fetch_row())
    echo "$name | $dept <br/>";
```

**Output:**

John Doe | Accounting  
Shion Lee | Engineering  
Alice Carol | Marketing  

The `list()` function was introduced in Unit 3. If you recall, the `list()` function assigns variables as if they were an array. It is used to assign a list of variables ($name, $dept) in one operation. The `mysqli` extension offers the ability to manage result sets using indexed arrays by using the `fetch_row()` method.

The resulting output displays the two columns Name and Department. If you want to retrieve all the columns and rows from the `employees` table, you can use the wildcard operator, `*`, which will match all columns and rows from the `employees` table:

```php
$query = "SELECT * FROM employees";  
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);  
while(list($id, $name, $dept, $location) = $result->fetch_row())
    echo "$id | $name | $dept | $location <br/>";
```

**Output:**

168 | John Doe | Accounting | US  
251 | Shion Lee | Engineering | Malaysia  
787 | Alice Carol | Marketing | US  

To quickly test an array result, remember that you can use the `print_r()` function within the HTML `<pre>` tag:

```php
while ($myrow = $result->fetch_row())
{
    echo "<pre>";
    print_r($myrow);
    echo "</pre>";
}
```
To access a specific record in a table, you need to specify some criteria by using the WHERE clause. For example, to retrieve the record of an employee with employeeID = 168:

```sql
$query = "SELECT * FROM employees WHERE employeeID = 168";
```

**Output:**

168 | John Doe | Accounting | US

You may use the wildcard character % and the LIKE operator to search for partial matches in your database. For example, to retrieve the record of an employee who has the name “Lee”, you can use the following statement:

```sql
$query = "SELECT * FROM employees WHERE name LIKE ' %Lee%'";
```

**Output:**

251 | Shion Lee | Engineering | Malaysia

If you want to display query results in a particular order, you can use the ORDER BY clause. For example, to retrieve the records of employees in alphabetical order by Name:

```sql
$query = "SELECT * FROM employees ORDER BY Name";
```

**Output:**

787 | Alice Carol | Marketing | US
168 | John Doe | Accounting | US
251 | Shion Lee | Engineering | Malaysia
To do the above in reverse order, use the desc (descending) keyword:

```sql
query = "SELECT * FROM employees ORDER BY Name DESC";
```

You may also perform simple calculations on the data such as AVG (average), COUNT (count items), MIN (minimum value), MAX (maximum value) and SUM (sum of values). For example, to get the average bonus amount from the bonus_pay table:

```sql
$query = "SELECT AVG(Amount) FROM bonus_pay";
```

**Output:**

2075.00

To get more detailed information, we can group data by using the **GROUP BY** clause. For example, to get the bonus amount for each employee from the bonus_pay table:

```sql
$query = "SELECT EmployeeId, SUM(Amount) FROM bonus_pay GROUP BY EmployeeId"
```

**Output:**

<table>
<thead>
<tr>
<th>EmployeeId</th>
<th>Bonus Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>168</td>
<td>5500.00</td>
</tr>
<tr>
<td>251</td>
<td>800.00</td>
</tr>
<tr>
<td>787</td>
<td>2000.00</td>
</tr>
</tbody>
</table>

### Updating records in a database

You will inevitably have to update records in a database. For example, you may need to update the cost of training. The **UPDATE** statement can be used to modify data in a record. The general syntax of this statement is as follows:

```sql
UPDATE tablename SET column1=expression1, column2=expression2,... [WHERE condition]
```

For example, if you want to increase the training cost stored in the training table by 20%, you can execute the following statement:

```sql
$query = "UPDATE training SET Cost = Cost * 1.2";
```
If you just want to change a particular record, you can attach the WHERE clause to the UPDATE statement. For example, if Alice (EmployeeID 787) has been transferred to the Sales department, you can update the employees table with the following statement:

```sql
$query = "UPDATE employees SET Department = 'Sales' WHERE EmployeeID = 787";
```

**Deleting records from a database**

To delete a record from a database, you will use the DELETE statement:

```
DELETE FROM table [WHERE condition]
```

For example, to remove all records from the employee_training table, you will use the following statement:

```sql
$query = "DELETE FROM employee_training";
```

If you just want to remove a particular record from the employee_training table, you will attach the WHERE clause:

```sql
$query = "DELETE FROM employee_training WHERE EmployeeID = 787";
```

---

**Activity 4.4**

1. In keeping with the object-oriented paradigm, the mysql extension is encapsulated into a class. As a result, we need to first instantiate the mysql class using the new keyword. True / False?

2. Consider the following information:

<table>
<thead>
<tr>
<th>Host name</th>
<th>User name</th>
<th>Password</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost</td>
<td>paul_dubois</td>
<td>p@ssw0rd</td>
<td>epub</td>
</tr>
</tbody>
</table>

Using the information in the table, write a PHP connection string to MySQL server using the mysqli extension.
3. What is the purpose of the following command?

$table->close();

4. We use the `query()` method to send a query to a database. Queries involve the __________ of data.

A. insertion, retrieval, updating and deletion
B. primary key and foreign key
C. tables, attributes, records, values, keys, schemas and relationships
D. one-to-one, one-to-many and many-to-many relationships

5. Which of the following is a valid data insertion query?

C. “INSERT INTO books SET Price = 39.95, Author = ‘Jonathan Stephens’, BookID = 123”
D. All of the above

6. You can retrieve all the columns and rows from a database table by using the wildcard operator, *, which will match all columns and rows. True / False?

7. What is the possible result of the following query?

$table = “UPDATE product SET Price = Price * 0.94”;

A. The product price will be increased by 94%
B. The product price will have a discount of 94%
C. The product price will be increased by 6%
D. The product price will have a discount of 6%
Using phpMyAdmin

When your database grows bigger, you will have more complex transactions, inter-joined-tables and complicated entity-relationship diagrams. When that happens, managing databases from the MySQL monitor (command-line) alone will no longer be adequate. Fortunately, you have a GUI-based database management tool called phpMyAdmin to assist you.

The phpMyAdmin tool is a web-based MySQL administration tool written in PHP that is recognised by web hosting providers around the world. It is a stable, feature-rich and open-source tool released under the GNU General Public License. It is viewed in a browser, which means that you can easily manage your MySQL database from a remote location if you have an Internet connection. Administrators can perform almost all the operations that MySQL database server offers using phpMyAdmin.

Do the following to install phpMyAdmin in Linux Ubuntu 10.10:

1. Run the Terminal.

2. Enter the following:

   ```bash
   sudo apt-get install libapache2-mod-auth-mysql phpmysql
   ```

3. You will then be shown the configuration applet. Select <Yes> to proceed with dbconfig-common.

![Figure 4.14 Configuring phpMyAdmin](image-url)
4. Next, you will be asked to enter a password for the MySQL root account. If you have followed our LAMP installation steps, use wawasan as the root account password.

![Figure 4.15 Enter the password for the MySQL root account](image1)

5. Next, you will be asked to enter a password for the phpMyAdmin account. Enter wawasan for the phpMyAdmin account password too. Confirm your password if the confirmation screen is displayed.

If you are asked to choose a web server, hit the <space bar> to select apache2.

![Figure 4.16 Select apache2 as the web server](image2)
6. You should get the following screen upon successful completion of the installation.

![Figure 4.17 Successful completion of the installation](image)

7. Open a web browser and enter `http://localhost/phpmyadmin/index.php` to run phpMyAdmin. The login screen shown in Figure 4.18 will be displayed.

![Figure 4.18 phpMyAdmin login screen](image)
8. You can login with the username root and the password wawasan. Congratulations, you have successfully installed phpMyAdmin. You can view the human_resource database and the tables that you have created and browse the records that you have inserted using phpMyAdmin.

Figure 4.19 Using phpMyAdmin to view database tables and to browse records in phpMyAdmin

To install phpMyAdmin in the Windows system:

1. Visit the phpMyAdmin homepage to download the file ‘phpMyAdmin-3.x.x-english.zip’.

2. Unzip the package into the Apache2 ‘htdocs’ directory (e.g., C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\phpmyadmin).

3. Create a file called ‘config.inc.php’ in the ‘phpmyadmin’ directory. Enter the following PHP code into ‘config.inc.php’ by using a text editor.

```php
<?php
    $cfg['blowfish_secret'] = 'ba17c1ec07d65003';
    // use a value of your choice
    $i=0;
    $i++;
    $cfg['Servers'][$i]['auth_type'] = 'cookie';
?>
```

4. Save and then close the file.

6. Login with the username root and the password wawasan. Congratulations, you have successfully installed phpMyAdmin on Windows.

---

### Web Reference

To learn more about phpMyAdmin, please visit:


---

### Summary

This section discussed the process of connecting to MySQL server with PHP scripts. You can use the PHP mysqli_connect function to create a connection command. To send a query to a database, you can use the query() method. You can insert data into a database by using the SQL `INSERT` command. To retrieve data from a database, you can use the SQL `SELECT` command. The `UPDATE` statement will be used to make changes to your existing data. To delete a record from a database, you will use the `DELETE` statement.

When your database grows bigger, managing databases from the MySQL monitor alone will no longer be adequate. Fortunately, you have a GUI-based database management tool called phpMyAdmin to assist you. The process of installing this tool on both the Linux Ubuntu and Windows platforms was described.

To summarise, you should now be familiar with:

1. Querying and inserting data into a database.
2. Retrieving, updating and deleting records from a database.
3. Installing phpMyAdmin.
Self-test 4.3

1. Consider the following information:

<table>
<thead>
<tr>
<th>Host name</th>
<th>User name</th>
<th>Password</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost</td>
<td>stefan_hinz</td>
<td>p@ssw0rd</td>
<td>docbook</td>
</tr>
</tbody>
</table>

With the information given in the table, write a PHP script to connect to MySQL server using the \texttt{mysqli} extension. If the connection fails, display the error codes and error messages, and then terminate the program.

2. Write PHP code to insert all three records shown in the table below into a MySQL table called \texttt{PERIODIC}. Assume that the table already exists and has the schema: \texttt{PERIODIC} (\texttt{Symbol}, \texttt{ElementName}, \texttt{AtomicNumber}). You do not need to create the table.

<table>
<thead>
<tr>
<th>PERIODIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol: Li</td>
</tr>
<tr>
<td>Symbol: Ti</td>
</tr>
<tr>
<td>Symbol: Pb</td>
</tr>
</tbody>
</table>

3. Consider the following schema for a database called \texttt{HOTEL}:

\texttt{hotel (HotelID, Name, City)}

\texttt{room (RoomID, HotelID, Type, Price)}

\texttt{guest (GuestID, Fullname, Address)}

\texttt{booking(BookingID, HotelID, GuestID, DateFrom, DateTo, RoomID)}

Write PHP code to execute the following queries:

a. List the full details of all hotels.

b. List the Name and City of all hotels in Georgetown.

c. List the Fullname and Address of all guests staying in Georgetown, alphabetically ordered by Fullname.

d. List all superior rooms with a price below $100 per night, in descending order of Price.

e. List the bookings for which no DateTo has been specified.
4. Based on the HOTEL schema in question 3, write PHP code for the following queries:
   
a. How many hotels are there?

b. What is the average price of a room?

c. List the total number of rooms according to the room type.

d. List the average price of rooms according to the room type.

5. Based on the HOTEL schema in question 3, write PHP code for the following queries:
   
a. Increase the price of all rooms by 10%.

b. Discount the price of superior rooms by 20%.

c. Delete the guest record that has GuestID = 618.

d. Delete all records from the booking table.

6. What is phpMyAdmin?

Suggested answers to activity

Feedback

Activity 4.4

1. False

2. $mysqli = new mysqli("localhost", "paul_dubois", "p@ssw0rd", "epub");

3. The command is used to close the MySQL connection. It is good practice to close the connection at the end of a script.

4. A

5. D

6. True

7. D
Summary of Unit 4

Summary

In this unit, you learnt about the concepts of relational database design. The key principles of database design, such as object modelling, avoiding redundant data and empty attributes, atomic column values, unique key selection, and creating questions for a database, were briefly explored. Additionally, you learnt that data in relational databases are organised into tables, attributes, records, values, keys, schemas and relationships.

Next, you were taught how to manage databases. You were given descriptions of the various types of MySQL database library functions that are used to retrieve and manipulate data. You were also shown how to set up and manage new accounts for database users. Finally, you learnt how to connect to MySQL server with PHP. You were shown PHP statements that are used to insert, select, modify and delete records in a MySQL database.

In the next unit, which is the final unit of this course, you will study the principles of using sessions and cookies for saving state information. You will then be able to build a complete web database application.
Unit Practice Exercises

1. In this exercise, you will create a simple product page and a simple shopping cart. First, using SQL scripts, you will create a database called SHOPPING that has the tables shown below. Store all your code in a file named `var/www/Unit4/create_table.sql` (or `<Apache Directory>/htdocs/Unit4/create_table.sql` in Windows). Fields should not have null values unless otherwise indicated.

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerID</td>
<td>INT</td>
<td>Primary key, auto_increment, starts from 100</td>
<td></td>
</tr>
<tr>
<td>Fullname</td>
<td>VARCHAR(50)</td>
<td>Customer's full name</td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td>VARCHAR(50)</td>
<td>Customer's email address</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>CHAR(41)</td>
<td>Customer's hashed password</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>TEXT</td>
<td>Customer's shipping address. Can be a null value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductID</td>
<td>CHAR(5)</td>
<td>Primary key, unique product ID</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR(50)</td>
<td>Product title</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>TEXT</td>
<td>Product description. Can be a null value.</td>
<td></td>
</tr>
<tr>
<td>Price</td>
<td>DEC(6,2)</td>
<td>Price of the product up to 999.99</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORDERS</th>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderID</td>
<td>INT</td>
<td>Primary key, auto_increment, starts from 1000</td>
<td></td>
</tr>
<tr>
<td>CustomerID</td>
<td>INT</td>
<td>ID of customer who placed the order</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>DATE</td>
<td>Order date</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>DEC(8,2)</td>
<td>Total amount up to 99999.99</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRODUCT_ORDERS</th>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderID</td>
<td>INT</td>
<td>Foreign key from ORDER (Set as primary key)</td>
<td></td>
</tr>
<tr>
<td>ProductID</td>
<td>CHAR(5)</td>
<td>Foreign key from PRODUCT (Set as primary key)</td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>INT</td>
<td>Quantity ordered</td>
<td></td>
</tr>
</tbody>
</table>
2. The SHOPPING_CART table is a temporary item holder for the customer who is shopping online. After the customer has checked-out, the items from SHOPPING_CART will be moved to PRODUCT_ORDERS for permanent storage. Insert the following product information into the PRODUCT table. Store your code into a file called ‘/var/www/Unit4/insert_product.sql’ (or ‘<Apache Directory>htdocs\Unit4\insert_product.sql’ in Windows).

<table>
<thead>
<tr>
<th>PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductID</td>
</tr>
<tr>
<td>22103</td>
</tr>
<tr>
<td>97902</td>
</tr>
<tr>
<td>12948</td>
</tr>
<tr>
<td>12983</td>
</tr>
</tbody>
</table>
3. Load ‘create_table.sql’ from question 1 and ‘insert_product.sql’ from question 2 into MySQL server using your root privilege. Create a PHP page to display all products from the PRODUCT table. Store it in the ‘/var/www/Unit4’ directory (or ‘<Apache Directory>htdocs\Unit4’ in Windows) and name it ‘MySite13.php’.

The description of the page is as follows:

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</td>
<td>My Site 13</td>
</tr>
</tbody>
</table>
| Display products in a table | Create a two-column table. The left column will display ProductID and Name. The right column will display all fields from the PRODUCT table and a Quantity text box together with an “Add to cart” button. Besides the “Add to cart button”, add a hidden field called ‘prodID’ to store the value of ProductID. The form tag should link to ‘MySite14.php’. | `<form method='POST' action='MySite14.php'>...
<input type='text' name='qty' size='2'>
<input type='hidden' name='prodID' value='$ProductID'>
...</form>` |

You should obtain a page like the one shown in Figure 4.20.

![Figure 4.20 Displaying all products from the PRODUCT table](image-url)
4. Next, you will create a shopping cart page in the ‘/var/www/Unit4’ directory (or ‘<Apache Directory>/htdocs/Unit4’ in Windows) and name it ‘MySite14.php’.

In ‘MySite14.php’, you are going to store Quantity and ProductID in the SHOPPING_CART table. Based on the ProductID in SHOPPING_CART, you will display all product information from the PRODUCT table on the shopping cart page. Show the total price for each item ordered and the total order amount. Provide a link to return to ‘MySite13.php’ and a “Checkout” button.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>My Site 14</td>
</tr>
<tr>
<td>Store the Quantity and ProductID in the SHOPPING_CART table</td>
<td>Retrieve the qty and prodID from MySite13.php.</td>
<td>$qty =$_POST['qty']; $prodID = $_POST['prodID'];</td>
</tr>
</tbody>
</table>
| Display all product information from the PRODUCT table on the shopping cart page | Use a while loop to get the ProductID values from the SHOPPING_CART table and another nested while loop to get all product information from the PRODUCT table. | while(…FROM SHOPPING_CART...)  
  
  while (…FROM PRODUCT...) |
| Show the total price of each item ordered and the total order amount. | Show the Quantity as an input text field and create a new column called Total (Price × Quantity). Accumulate the overall Order Total during looping and display it below the shopping cart table. Store the $OrderTotal in a hidden field in order to pass it to the checkout page. | $Total = number_format($Price * $CartQty, 2);  
$OrderTotal += $Total;  
<input type='hidden' name='orderTotal' value='$OrderTotal'> |
| Provide a link to return to ‘MySite13.php’ and a “Checkout” button | The form tag should link to ‘MySite15.php’. | `<form method="POST" action="MySite15.php">  
  ...
  <a href="MySite13.php">Keep Shopping</a>  
  <input type="submit" name="Submit" value="Checkout »">` |

You should obtain a page like the one shown in Figure 4.21.
5. When customers click on the “Checkout” button that you created in question 4, a confirmation page showing customer billing and shipping information will be displayed. Create a PHP file in the ‘/var/www/Unit4’ directory (or ‘<Apache Directory>/htdocs/Unit4’ in Windows) and name it ‘MySite15.php’. You will retrieve the dummy customer information that was stored in the CUSTOMER table earlier.

You should obtain a page like the one shown in Figure 4.22.
6. Lastly, you will create a confirmation page to show customer billing details and the items purchased. Create a PHP file in the `/var/www/Unit4` directory (or `<Apache Directory>/htdocs/Unit4` in Windows) and name it `MySite16.php`. You will insert the order into the ORDERS table, copy the items in the SHOPPING_CART table to the PRODUCT_ORDERS table, remove the items from SHOPPING_CART and show billing information and the items purchased.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td><code>My Site 16</code></td>
</tr>
</tbody>
</table>
| Insert the order into the ORDERS table | `Insert the order into the ORDERS table. CustomerID and Total can be obtained from the previous page by using the $_POST[] function. Let the OrderID auto generate by inserting a NULL. The OrderID can be obtained by invoking $mysqli->insert_id. Set today's date as the Date.` | `date_default_timezone_set('Asia/Kuala_Lumpur');
$custID = $_POST['CustID'];
$OrderTotal = $_POST['orderTotal'];
$today = date("y-m-d");
$orderID = $mysqli->insert_id;` |
| Copy the SHOPPING_CART items to the PRODUCT_ORDERS table | `Retrieve ProductID and Quantity from SHOPPING_CART and insert them into PRODUCT_ORDERS together with the OrderID.` | `$query = "SELECT ...
while(...
{
    $query2 = "INSERT_` |
| Remove the items from SHOPPING_CART | `Remove all records from the shopping cart.` | `DELETE FROM...` |
Show billing information
List customer details from the CUSTOMER table using the $custID that you used earlier.

Show the items purchased
Use a while loop to get the ProductID from the PRODUCT_ORDERS table and another nested while loop to get all product information from the PRODUCT table, neglecting the Description. Show the Order Total from the $OrderTotal that you obtained earlier. Create a link to return to the product page.

<h1>Thank you for your order</h1>
<p>Bill to: </p>

while(...FROM PRODUCT_ORDERS ...)
{
while (...FROM PRODUCT...)
...
<a href="MySite13.php">Back to Shopping</a>

You should obtain a page like the one shown in Figure 4.23.

Figure 4.23 Displaying billing information and a summary of purchased items
Suggested Answers to Self-tests

Feedback

Self-test 4.1

1. An update anomaly is a situation where inconsistent data is created when a database is updated. The integrity of the data is violated and we no longer know which data item is correct and which is incorrect.

2. True

3. D

4. a. The design problem is in the CamelTypesFound column because it is nesting an entire table in one column. It does not follow the “single item single column” strategy. When you want the answer to questions like “How many types of Dromedary (T1) camels are found in Somalia?”, the database cannot count the matching TypeIDs. Each of the attributes will have to be analysed to see if it matches the TypeID inside.

   b. The problem is resolved if a new table called CAMELTYPE_COUNTRIES is created as shown in the figure below.

```
<table>
<thead>
<tr>
<th>Code</th>
<th>TypeID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>T1</td>
</tr>
<tr>
<td>SOM</td>
<td>T3</td>
</tr>
<tr>
<td>SOM</td>
<td>T5</td>
</tr>
<tr>
<td>SUD</td>
<td>T2</td>
</tr>
<tr>
<td>SUD</td>
<td>T4</td>
</tr>
<tr>
<td>SUD</td>
<td>T5</td>
</tr>
<tr>
<td>MAU</td>
<td>T1</td>
</tr>
<tr>
<td>MAU</td>
<td>T5</td>
</tr>
</tbody>
</table>
```
5. a. The design problem is in the Donation column because chances of donations being received by each country may be small (no donations are recorded in the table). This will cause many empty attributes or null values to be present in the Donation column. This type of data model wastes storage and null values may cause problems when numerical functions like calculating the total are executed. Null values also create uncertainty as the question will arise whether the field is intentionally left empty or data has not been entered yet.

b. To resolve the problem, you can create a new table called COUNTRIES_DONATION. This table only stores the code of countries that have received donations. The code of the countries that have not received any donation will not be stored to avoid storing null values.

<table>
<thead>
<tr>
<th>COUNTRIES_DONATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
</tr>
</tbody>
</table>

### Self-test 4.2

1. CREATE DATABASE stock_market;
   USE stock_market;

2. CREATE TABLE STOCK_MARKET {
   ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
   Symbol VARCHAR(10) NOT NULL,
   StockName VARCHAR(20) NOT NULL,
   Price FLOAT(10,2) NOT NULL,
   Changes FLOAT(8,2) NOT NULL,
   PercentRate FLOAT(6,2) NOT NULL ;
}

3. ALTER TABLE STOCK_MARKET ADD COLUMN Region VARCHAR(15) NULL AFTER ID;

4. patrick2@localhost

5. GRANT ALL ON accounting.* TO leeann@localhost;
   GRANT SELECT ON accounting.* TO maria@localhost IDENTIFIED BY 'p@ssw0rd';

6. REVOKE INSERT, UPDATE ON accounting.* FROM leeann@localhost;
   SHOW GRANTS FOR leeann@localhost;
   SHOW GRANTS FOR maria@localhost;
7. SET PASSWORD FOR leeann@localhost = PASSWORD ('12345678');
SELECT PASSWORD('12345678');
+-------------------------------------------+
| *84AAC12F54AB666ECFC2A83C676908C8BBC381B1 |
+-------------------------------------------+

8. Both SQL statements will produce the same results. The first statement uses a plain text password and the second one uses a hash value. The second statement will not send a plain text password to the server logs. Therefore, only the hash value can be read by anyone who can view the server logs.

9. B

Self-test 4.3

1. $mysqli = new mysqli("localhost", "stefan_hinz", "p@ssw0rd", "docbook");
   if ($mysqli->connect_errno) {
     printf("%s: %s", $mysqli->connect_errno, $mysqli->connect_error);
   }

2. $query = "INSERT INTO PERIODIC VALUES
   ('Li', 'Lithium', 3),
   ('Ti', 'Titanium', 22),
   ('Pb', 'Lead', 82) ";

3. a. $query = "SELECT * FROM hotel";

   b. $query = "SELECT Name, City FROM hotel
   WHERE City = 'Georgetown'";

   c. $query = "SELECT Fullname, Address FROM guest
   WHERE Address LIKE '%Georgetown%'
   ORDER BY Fullname";

   d. $query = "SELECT * FROM room
   WHERE Type = 'superior' AND Price < 100
   ORDER BY Price DESC";

   e. $query = "SELECT * FROM booking
   WHERE DateTo IS NULL";
4. a. $query = "SELECT COUNT(*) FROM hotel";
   
b. $query = "SELECT AVG(Price) FROM room";
   
c. $query = "SELECT Type, COUNT(Type) FROM room GROUP BY Type";
   
d. $query = "SELECT Type, AVG(Price) FROM room GROUP BY Type";

5. a. $query = "UPDATE room SET Price = Price * 1.1";
   
b. $query = "UPDATE room SET Price = Price * 0.8 WHERE Type = 'superior'";
   
c. $query = "DELETE FROM guest WHERE GuestID = 618";
   
d. $query = "DELETE FROM booking";

6. The phpMyAdmin tool is a web-based MySQL administration tool written in PHP and it is recognised by web hosting providers around the world. It is a stable and feature-rich tool. It is viewed on a browser, which means that you can easily manage your MySQL database from a remote location that is connected to the Internet. Administrators can perform almost all operations that MySQL database server offers using phpMyAdmin.
Suggested Answers to Unit Practice Exercises

Feedback

1. create_table.sql

    /****/ CREATE TABLE statements */***/
    CREATE DATABASE SHOPPING;
    USE SHOPPING;

    /****/ CREATE TABLE statements */***/
    CREATE TABLE CUSTOMER ( 
        CustomerID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
        Fullname VARCHAR(50) NOT NULL,  
        Email VARCHAR(50) NOT NULL,  
        Password CHAR(41) NOT NULL,  
        Address TEXT );

    CREATE TABLE PRODUCT ( 
        ProductID INT NOT NULL PRIMARY KEY,  
        Name VARCHAR(50) NOT NULL,  
        Description TEXT,  
        Price DEC(6,2) NOT NULL );

    CREATE TABLE ORDERS ( 
        OrderID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
        CustomerID INT NOT NULL,  
        Date DATE NOT NULL,  
        Total DEC(8,2) NOT NULL );

    CREATE TABLE PRODUCT_ORDERS ( 
        OrderID INT NOT NULL,  
        ProductID INT NOT NULL,  
        Quantity INT NOT NULL,  
        PRIMARY KEY (OrderId, ProductID));

    CREATE TABLE SHOPPING_CART ( 
        CartID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
        SessionID CHAR(50),  
        ProductID INT NOT NULL,  
        Quantity INT NOT NULL );

    /****/ ALTER TABLE statement */***/
    ALTER TABLE CUSTOMER AUTO_INCREMENT = 100;
    ALTER TABLE ORDERS AUTO_INCREMENT = 1000;
2. insert_product.sql

```sql
/****** INSERT statements ******/
INSERT INTO PRODUCT VALUES
(22103, 'Tribute Blend', 'Sun-dried beans from Ethiopia, inspired by our collaboration with Ethiopian farmers.', 13.95),
(97902, 'Pike Place Roast', 'The smoother finish and soft acidity are attributed to a proprietary blend of high-altitude arabica beans.', 9.95),
(12948, 'French Roast', 'Blunt, smoky flavours are the objective in creating this, our darkest roasted coffee.', 10.95),
(12983, 'Sumatra', 'The concentrated spicy, herbal notes and earthy aroma are the telltale signatures of this well-loved coffee.', 10.95);
```

```sql
INSERT INTO CUSTOMER VALUES
(NULL, 'John Doe', 'johndoe@web.com', PASSWORD('wawasan'), '1 Seattle, Washington');
```

3. MySite13.php

```php
<?php
  $mysqli = new mysqli("localhost", "root", "wawasan", "shopping");
  if ($mysqli->connect_errno) {
    printf("%s: %s", $mysqli->connect_errno, $mysqli->connect_error);
    $mysqli->connect_error);
  }
  $query = "SELECT * FROM PRODUCT";
  $result = $mysqli->query($query, MYSQLI_STORE_RESULT);
  if(!$result) {
    echo($mysqli->error);
    exit();
  }
  while(list($ProductID, $Name, $Desc, $Price) = $result->fetch_row())
  {
    echo "<tr>";
    echo "<td>";
```

echo "#$ProductID<br />$Name";
echo "</td>";
echo "<td>";
echo "$Name<br />$Desc<br />$$Price";
echo "<form method='POST' action='MySite14.php'/>";
    echo "#$ProductID Qty:";
    echo "<input type='text' name='qty' size='2'/>";
    echo "<input type='hidden' name='prodID' value='$ProductID'>";
    echo "<input type='submit' name='Submit' value='Add to cart'/>";
    echo "</form>";
    echo "</td>";
</tr>

$mysqli->close();
?>
</table>
</BODY>
</HTML>


<html><head>
<title>My Site 14</title>
</head>
<body>
<?php
$mysqli = new mysqli("localhost", "root", "wawasan", "shopping");
if ($mysqli->connect_errno) {
    printf("%s: %s", $mysqli->connect_errno, $mysqli->connect_error);
    $mysqli->close();
} else {
    $mysqli = new mysqli("localhost", "root", "wawasan", "shopping");
    if ($mysqli->connect_errno) {
        printf("%s: %s", $mysqli->connect_errno, $mysqli->connect_error);
    }
    $qty =$_POST['qty'];
    $prodID = $_POST['prodID'];
    $query = "INSERT INTO SHOPPING_CART (Quantity, ProductID) VALUES ('$qty','$prodID')";
    $result = $mysqli->query($query);
    if(!$result) {
        echo($mysqli->error);
        exit();
    }
}?>
<h1>My Coffee Store cart</h1>
<form method="POST" action="MySite15.php">
<?php
$query = "SELECT ProductID, Quantity FROM SHOPPING_CART";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
$orderTotal = 0;
while (list($CartProdID, $CartQty) = $result->fetch_row()) {
    $query = "SELECT * FROM PRODUCT WHERE ProductID = '$CartProdID';";
    $result2 = $mysqli->query($query, MYSQLI_STORE_RESULT);
    if(!$result2) {
        echo($mysqli->error);
        exit();
    }

    while(list($ProductID, $Name, $Desc, $Price) = $result2->fetch_row()) {
        echo "<tr>
        <td><input type='text' name='qty' size='2' value='$CartQty'></td>
        <td>#$ProductID</td>
        <td>$Name</td>
        <td>$Desc</td>
        <td>$$Price</td>
        $Total = number_format($Price * $CartQty, 2);
        <td>$$Total</td>
        </tr>
    }
    $orderTotal += $Total;
}
$mysqli->close();
?>
5. MySite15.php

```php
<?php
    $mysqli = new mysqli("localhost", "root", "wawasan", "shopping");
    if ($mysqli->connect_errno) {
        printf("%s: %s", $mysqli->connect_errno, $mysqli->connect_error);
    }
    $OrderTotal = $_POST['orderTotal'];
    $query = "SELECT CustomerID, Fullname, Email, Address FROM CUSTOMER";
    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);
    if(!$result) {
        echo($mysqli->error);
        exit();
    }
    while(list($CustomerID, $Fullname, $Email, $Address) = $result->fetch_row()) {
        echo "<tr><td>Customer ID</td>
```
```
6. MySite16.php

```php
<?php
// INSERT INTO ORDERS
$mysqli = new mysqli("localhost", "root", "wawasan", "shopping");
if ($mysqli->connect_errno) {
    printf("%s: %s", $mysqli->connect_errno, $mysqli->connect_error);
}
$date_default_timezone_set('Asia/Kuala_Lumpur');
$custID = $_POST['CustID'];
$OrderTotal = $_POST['orderTotal'];
$today = date("y-m-d");
$query = "INSERT INTO ORDERS (OrderID, CustomerID, Date, Total) VALUES (NULL, '$custID', '$today', '$OrderTotal')";
$result = $mysqli->query($query);
// Get back OrderID

$mysqli->close();
?>
</table><input type="submit" name="Submit" value="Confirm">
</form>
</BODY>
</HTML>
```

```
<html>
<head>
<title>My Site 16</title>
</head>
<body>
<?php
// INSERT INTO ORDERS
$mysqli = new mysqli("localhost", "root", "wawasan", "shopping");
if ($mysqli->connect_errno) {
    printf("%s: %s", $mysqli->connect_errno, $mysqli->connect_error);
}
$date_default_timezone_set('Asia/Kuala_Lumpur');
$ custID = $_POST['CustID'];
$ OrderTotal = $_POST['orderTotal'];
$today = date("y-m-d");
$query = "INSERT INTO ORDERS (OrderID, CustomerID, Date, Total) VALUES (NULL, '$custID', '$today', '$OrderTotal')";
$result = $mysqli->query($query);
// Get back OrderID

$mysqli->close();
?>
</table><input type="submit" name="Submit" value="Confirm">
</form>
</BODY>
</HTML>
```
$orderID = $mysqli->insert_id;
if(!$result) {
    echo($mysqli->error);
    exit();
}
?>
<?php
// Copy SHOPPING_CART to PRODUCT_ORDERS
$query = "SELECT ProductID, Quantity FROM SHOPPING_CART";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
while(list($ProductID, $Quantity) = $result->fetch_row()) {
    $query2 = "INSERT INTO PRODUCT_ORDERS (OrderID, ProductID, Quantity) VALUES ('$orderID', '$ProductID', $Quantity)"
    $result2 = $mysqli->query($query2);
    if(!$result2) {
        echo($mysqli->error);
        exit();
    }
}
?>
<?php
// Delete SHOPPING_CART items
$query = "DELETE FROM shopping_cart";
$result = $mysqli->query($query);
if(!$result) {
    echo($mysqli->error);
    exit();
}
?>
<h1>Thank you for your order</h1>
<p>Bill to: </p>
<?php
$query = "SELECT CustomerID, Fullname, Email, Address FROM CUSTOMER WHERE CustomerID = '$custID'";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
while (list($CustomerID, $Fullname, $Email, $Address) = $result->fetch_row())
{
    echo "$Fullname (#$CustomerID)<br />";
    echo "$Email<br />";
    echo "$Address<br />";
}
?>
<p>Order ID <?php echo $orderID ?><br />
Summary of your purchase:<p>
<table border="0" width="50%">
<tr>
    <th>Qty.</th>
    <th>Item ID</th>
    <th>Item Name</th>
    <th>Price</th>
    <th>Total</th>
</tr>
<?php
$query = "SELECT ProductID, Quantity FROM PRODUCT_ORDERS WHERE OrderID = $orderID";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
while (list($CartProdID, $CartQty) = $result->fetch_row())
{
    $query2 = "SELECT * FROM PRODUCT WHERE ProductID = '$CartProdID'";
    $result2 = $mysqli->query($query2, MYSQLI_STORE_RESULT);
    if(!$result2) {
        echo($mysqli->error);
        exit();
    }
    while(list($ProductID, $Name, $Desc, $Price) = $result2->fetch_row())
    {
        echo "<tr>
            <td>$CartQty</td>
            <td>#$ProductID</td>
            <td>$Name</td>
            <td>$$Price</td>
            $Extprice = number_format($Price * $CartQty, 2);
            <td>$$Extprice</td>
        </tr>
    ";
}
$mysqli->close();
?
</table>
<table border="0" width="50%">
<tr><td>Order Total: $<?php echo number_format($OrderTotal, 2) ?></td></tr>
<tr><td><a href="MySite13.php">Back to Shopping</a></td></tr>
</table>
</BODY>
</HTML>
Appendix

human_resource database

hr_tbls.sql

/****** CREATE TABLE statements ******/
CREATE TABLE employees (  
EmployeeID INT NOT NULL PRIMARY KEY,  
Name VARCHAR(50) NOT NULL,  
Department VARCHAR(20) NOT NULL,  
Location VARCHAR(40) NOT NULL);

CREATE TABLE bonus_pay (  
TransactionID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
EmployeeID INT NOT NULL,  
Amount FLOAT(10,2) NOT NULL,  
Date DATE NOT NULL);

CREATE TABLE training (  
TrainingID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
Trainer VARCHAR(50) NOT NULL,  
CourseName VARCHAR(40) NOT NULL,  
Cost FLOAT(10,2) NOT NULL);

CREATE TABLE employee_training (  
EmployeeID INT NOT NULL,  
TrainingID INT NOT NULL,  
PRIMARY KEY (EmployeeID, TrainingID));

CREATE TABLE training_notes (  
TrainingID INT NOT NULL PRIMARY KEY,  
Note TEXT);

/****** ALTER TABLE statement ******/
ALTER TABLE training AUTO_INCREMENT = 80;

/****** INSERT statements ******/
INSERT INTO employees VALUES  
(251, ‘Shion Lee’, ‘Engineering’, ‘Malaysia’);

INSERT INTO bonus_pay VALUES  
(NULL, 168, 2500.00, ‘2011-06-01’),  
(NULL, 251, 800.00, ‘2011-06-01’),  
(NULL, 787, 2000.00, ‘2011-06-01’),  
(NULL, 168, 3000.00, ‘2011-07-01’);
INSERT INTO training VALUES
(NULL, 'Josh Low', 'Database Administration', 5000),
(NULL, 'Jason Tan', 'PHP Programming', 5000),
(NULL, 'Mohandas', 'Project Management', 8000);

INSERT INTO employee_training VALUES
(168, 83),
(251, 81),
(787, 81),
(787, 83),
(251, 82);

INSERT INTO training_notes VALUES
(168, 'Lunch and hi tea are provided');
References


## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Attributes or fields are columns in a database table. Each attribute has a unique name and can contain data of a certain type.</td>
</tr>
<tr>
<td>Data model</td>
<td>A data model is a set of rules and conventions that assist in transforming loose concepts in the real world into a finite model by means of reusable concepts.</td>
</tr>
<tr>
<td>Foreign key</td>
<td>A foreign key is a field in a database table that matches the primary key of another table.</td>
</tr>
<tr>
<td>Invoking fields</td>
<td>The operator <code>-&gt;</code> in the object-oriented context is used to invoke fields. An object is correlated with its specific field's value and is referenced like this: <code>Object -&gt; field</code>.</td>
</tr>
<tr>
<td>Key</td>
<td>A key is an attribute identifier in a database table. It is a unique attribute that can guarantee that the information in a record will not be repeated.</td>
</tr>
<tr>
<td>Many-to-many relationship</td>
<td>In a many-to-many relationship, many records in a table are linked to many records in another table.</td>
</tr>
<tr>
<td>One-to-many relationship</td>
<td>In a one-to-many relationship, one record in a table is linked to many records in another table.</td>
</tr>
<tr>
<td>One-to-one relationship</td>
<td>In a one-to-one relationship, one record in a table is related to only one record in another table.</td>
</tr>
<tr>
<td>phpMyAdmin</td>
<td>This is a web-based MySQL administration tool.</td>
</tr>
<tr>
<td>Primary key</td>
<td>A primary key is a unique key in a table.</td>
</tr>
<tr>
<td>Queries</td>
<td>SQL queries involve data insertion, retrieval, updating and deletion.</td>
</tr>
<tr>
<td>Records</td>
<td>Records or tuples are rows in a table. Each record in a table is made up of a number of attributes or fields.</td>
</tr>
</tbody>
</table>
Redundant data
Redundant data are duplicate data that will suffer from update anomalies.

Relational database
A relational database is a database that organises its data into collections of tables, rows, attributes and domains. Predicate logic is used to both describe the information contained in the database and to query it.

Schemas
A database schema is a set of table designs for a database. A schema shows a table's attributes, and it indicates the primary key and any foreign key the table possesses. Typically, a schema will not include any actual data values.

Update anomalies
An update anomaly is a situation that causes inconsistent data to be created when a database is updated. When this happens, the integrity of the data is violated and it is no longer known if the data are correct.
Unit 5

Managing State Information
Wawasan Open University is Malaysia's first private not-for-profit tertiary institution dedicated to adult learners. It is funded by the Wawasan Education Foundation, a tax-exempt entity established by the Malaysian People's Movement Party (Gerakan) and supported by the Yap Chor Ee Charitable and Endowment Trusts, other charities, corporations, members of the public and occasional grants from the Government of Malaysia.

The course material development of the university is funded by Yap Chor Ee Charitable and Endowment Trusts.

© 2011 Wawasan Open University. Except where otherwise noted, this work is licensed under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

This course material was published to support the learning of students registered with Wawasan Open University. Wawasan Open University does not grant any degree, certification or credits based solely on your completion of this course material.
# Contents

## Unit 5  Managing State Information

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit overview</td>
<td>1</td>
</tr>
<tr>
<td>Unit objectives</td>
<td>1</td>
</tr>
<tr>
<td><strong>5.1  Passing data between pages</strong></td>
<td>3</td>
</tr>
<tr>
<td>Objectives</td>
<td>3</td>
</tr>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Form elements</td>
<td>3</td>
</tr>
<tr>
<td>Query strings</td>
<td>6</td>
</tr>
<tr>
<td>Suggested answers to activities</td>
<td>11</td>
</tr>
<tr>
<td><strong>5.2  Using cookies to save state information</strong></td>
<td>13</td>
</tr>
<tr>
<td>Objectives</td>
<td>13</td>
</tr>
<tr>
<td>Introduction</td>
<td>13</td>
</tr>
<tr>
<td>What is a cookie?</td>
<td>13</td>
</tr>
<tr>
<td>Setting a cookie</td>
<td>14</td>
</tr>
<tr>
<td>Retrieving cookie data</td>
<td>15</td>
</tr>
<tr>
<td>Suggested answers to activity</td>
<td>19</td>
</tr>
<tr>
<td><strong>5.3  Using sessions to save state information</strong></td>
<td>21</td>
</tr>
<tr>
<td>Objectives</td>
<td>21</td>
</tr>
<tr>
<td>Introduction</td>
<td>21</td>
</tr>
<tr>
<td>Starting a session</td>
<td>21</td>
</tr>
<tr>
<td>Destroying a session</td>
<td>22</td>
</tr>
<tr>
<td>Setting and retrieving the session ID</td>
<td>22</td>
</tr>
</tbody>
</table>
Unit Overview

In web programming, state information is information relating to the current status of a web page. Websites use the Hypertext Transfer Protocol (HTTP) to deliver information via the Web when users send requests to them. However, HTTP is a stateless protocol, which means it has no record of state information from previous web requests made by a user. Therefore, alternative methods need to be used to gather and store information when the user moves from one page to another on a website. For example, when a user moves from the product selection page to the payment page of an e-commerce site, the information provided by the user in the product selection page needs to be passed on to the payment page.

This unit discusses the techniques of using sessions and cookies in web database applications to manage state information. It will also discuss how to track the interactions between a user and a web application by using PHP.

The first section discusses various techniques used to pass information such as data and variables between web pages. It explains how you can use hidden form fields and query strings. The second section describes how cookies are used. It shows you how to implement cookies with PHP. The third section discusses the use of sessions. It explains the methods of adding and reading sessions. It also covers the session management mechanism of a web database application.

Unit Objectives

By the end of this unit, you should be able to:

1. Explain various techniques used to pass state information between web pages.
2. Create and use cookies to save state information.
3. Create and destroy sessions and manage session variables.
5.1 Passing Data between Web Pages

Objectives

By the end of this section, you should be able to:

1. Describe how to pass information between web pages.
2. Discuss the use of form elements for passing information between web pages.
3. Explain the use of URLs for passing information between web pages.

Introduction

In web database applications, it is essential for developers to collect and pass information from one web page to another for processing. For example, if you host an e-commerce web application, you need to know the items selected and placed in the shopping cart by a user. You also need to keep track of whether the user is a returning customer or a new customer. In order to have a successful e-commerce web application, you need to handle the sign-in mechanism and the billing job as well.

Because the Hypertext Transfer Protocol (HTTP) that defines the rules of transferring data via the World Wide Web is a stateless protocol, it is incapable of retaining the current state of a web page. This means that each time a server receives a request from a client; it processes the request, sends a response and then forgets about the request. A new request from the client is independent of any of the previous requests. That is why you need to use HTML form elements, URLs, cookies or sessions to save state information. In this section, you will learn how to pass variables using form elements and URLs.

Form elements

Forms allow you to collect data from your visitors and send it to the server for storage, manipulation or to return some results to the visitors. PHP has various functions and techniques for handling input from HTML forms. You will use the special array variables $_GET[] and $_POST[] to handle form input.
Before we can use these special array variables, let us first take a look at a typical HTML form. Here is a simple form used to login to a web portal:

```html
<HTML>
  <HEAD>
    <TITLE>SIGN IN</TITLE>
  </HEAD>
  <BODY>
    <form method="POST" action="form-handler.php">
      Email: <input type="text" name="email"><br>
      Password: <input type="password" name="password"><br>
      <input type="submit" value="Login">
    </form>
  </BODY>
</HTML>
```

The example shows a form set up with a method and an action attribute. The method defines how the data from the form is to be transferred to the handler page. The handler page is defined by the action attribute. The method can be either GET or POST.

Let us first focus on the POST method. When POST is used, the data from a form will be put into a special server variable. To access it in PHP, you may use the built-in array variable, `$_POST[]`. The built-in array variable identifies the form values by referencing the name attribute of the input tags or any other name attribute of the form elements.

To retrieve the POST variables, we can use the following method:

```php
$email = $_POST['email'];
$password = $_POST['password'];
echo "Email : $email \n Password : $password";
```

If you enter johndoe@web.com in the email field and wawasan in the password field, you will get the following output:

```
Email : johndoe@web.com Password : wawasan
```

When POST is used, the posted data is completely invisible to the user. In a typical login mechanism, the data obtained will be checked against the records in the database and verified before the user is allowed access.

If you have some information that is not passed through any form field, you can create a hidden field to hold that information. A hidden field is similar to a text field except that it is not shown on the page. Hence, the visitor cannot type anything into the hidden field. This is a powerful feature that you can use to transfer data or variables from a form page into a handler page. A hidden field is represented by an INPUT element with type = hidden. You can place the hidden field anywhere you want as long as it is within the form tag and follows HTML rules.
The following example shows how a hidden field is used to store a user’s preferred language so that the server will be informed of this preference when it handles the form:

```html
<form method="POST" action="form-handler.php">
  Email: <input type="text" name="email"><br>
  <input type="hidden" name="Language" value="English">  
  Password: <input type="password" name="password"><br>
  <input type="submit" value="Login">
</form>
```

**Activity 5.1**

1. PHP uses special array variables to handle the input from HTML forms. They are ___________ and ___________.
   
   A. `$_SESSION[]` and `$_COOKIE[]`
   B. `$_GET[]` and `$_POST[]`
   C. `$_QUERY_STRING[]` and `$_ACTION[]`
   D. `$_HIDDEN[]` and `$_INPUT[]`

2. Which of the following is a valid way of writing a hidden field in HTML forms?
   
   A. `input hidden="type"`
   B. `hidden type="input"`
   C. `input type="hidden"`
   D. `type input="hidden"`

3. The PHP built-in array variables identify form values by referencing the __________ attribute of the _________ tags or any other name attribute of the form elements.

Query strings

A query string is a part of the Uniform Resource Locator (URL); it contains data to be passed to another page. The following is a typical URL containing a query string:

http://web/page.php?{query_string}

The question mark (after http://web/page.php in the previous example) is used as a separator and is not part of the query string. The main use of query strings is to transfer the contents of a HTML form. We discussed the POST method in the previous subsection. If the GET method is used in the form, all control names and values in the form will be appended as a query string to the end of the handler page URL that is specified in the action attribute.

For example:

<form method="GET" action="form-handler.php">
Email: <input type="text" name="email"><br>
Password: <input type="password" name="password"><br>
<input type="submit" value="Login">
</form>

To retrieve the GET variables, you can use the $_GET[] array:

$email = $_GET['email'];
$password = $_GET['password'];
echo "Email : $email \n Password : $password";

If you enter johndoe@web.com in the email field and wawasan in the password field, the URL address of the handler page will become:

form-handler.php?email=johndoe@web.com&password=wawasan

You can see that the GET method is not very appropriate for this example because you do not want the email address and password details to be displayed in the URL for security reasons. A better and more secure way would be to use the POST method when login details need to be sent to a web server.

Query string structure

A query string is usually made up of a series of field-value pairs. A form that submitted input1, input2, input3 will have its corresponding value1, value2, value3 as shown in the following query string:

input1=value1&input2=value2&input3=value3
Notice that the field-value pairs are separated by the ampersand (‘&’) symbol. The field-value pair itself uses the equals sign (‘=’) as its relational operator. If you had created hidden fields in the form, these fields will be included in the query string when the form is submitted with the GET method. For example, the following hidden field,

\[
\text{<input type=“hidden” name=“Language” value=“English”>}
\]

will generate the query string

\[
\ldots?Language=English&\ldots.
\]

In a nutshell, form content is encoded as a query string when the form is submitted via the GET method. On the other hand, form content submitted via the POST method will be sent as the body of the request and is not displayed in the URL.

---

**Web Reference**

The following web reference presents a comprehensive view of form elements and the processing of form data:

“W3C Recommendation: 17 Forms in HTML documents”,
http://www.w3.org/TR/REC-html40/interact/forms.html

---

**Activity 5.2**

1. What is a query string?

2. If the GET method is used in a form and if you enter Alice in the name field and Female in the gender field of the form, the URL of the target.php page will become:

   A. target.php?Female=gender&Alice=name
   
   B. target.php?name=Alice?gender=Female
   
   C. target.php?Female=gender?Alice=name
   
   D. target.php?name=Alice&gender=Female

3. What is the difference between submitting a HTML form using the POST method and the GET method? How do you retrieve field variables using each of these methods in PHP?
Summary

In this section, you looked at some mechanisms for passing data between web pages. You can use HTML forms to collect data from visitors to your website and then send the data to the server for processing.

When POST is used, the data will be put into a special server variable. To access it in PHP, you may use the built-in array variable \$_POST[]. The built-in array variable identifies the form values by referencing the name attribute of the input tags or any other name attribute of the form elements.

A query string is a part of the Uniform Resource Locator (URL); it contains data to be passed to another page. If the GET method is used in the form, all the form elements will be appended as a query string to the end of the handler page URL. A query string is usually made up of a series of field-value pairs.

You have learnt about form elements and the various PHP functions for passing data between pages. In the next section, you will look at cookies. Cookies are a powerful way of saving state information in PHP.

Self-test 5.1

1. The Hypertext Transfer Protocol (HTTP), which defines the rules of transferring data via the World Wide Web, is a stateful protocol; that is, it is capable of retaining a state between web pages. True / False?

2. You can create a __________ field to hold information that is not passed through any form field. This field is similar to a text field except that it does not appear on the web page.
3. Consider the following HTML form elements:

```html
<form method="POST" action="target.php">
<input type="text" name="quantity"><br>
<input type="hidden" name="productID">
</form>
```

To retrieve the field variables from ‘target.php’, you can use the following:

A. `$_POST[‘quantity’]` and `$_POST[‘productID’]`
B. `$_GET[‘quantity’]` and `$_GET[‘productID’]`
C. `$_POST[‘text’]` and `$_POST[‘hidden’]`
D. `$_GET[‘text’]` and `$_GET[‘hidden’]`

4. Write the PHP code to retrieve the following query string:

`http://server/web/handler.php?Country=Malaysia&Organizer=Disted+College&PhoneNumber=090-1234567`

5. Write PHP code to retrieve and echo information from the following online form using the POST method.
6. Consider the source code of the following HTML form and the details entered into the web form shown below the source code. Write the URL query string that will be generated by the GET method.

```html
<form method="GET" action="form-handler3.php">
  <b>Ordering information:</b><br>
  Charge to my <INPUT TYPE=radio NAME=cctype VALUE=MasterCard>MasterCard
  <INPUT TYPE=radio NAME=cctype VALUE=Visa>Visa<br>
  Credit card number: <INPUT TYPE=text NAME=ccnum SIZE=19 MAXLENGTH=19 >
  Expiration date (mm/yyyy): <INPUT TYPE=text NAME=expmo SIZE=2 MAXLENGTH=2 />
  <INPUT TYPE=text NAME=expyr SIZE=4 MAXLENGTH=4><br>
  Phone: <INPUT TYPE=text NAME=phone SIZE=25><br>
  E-mail address: <INPUT TYPE=text NAME=email SIZE=60><br>
  (Necessary if you want us to send you a confirmation message)<br>
  <INPUT TYPE="submit" VALUE="Submit">
</form>
```

![Online Order Form - Windows Internet Explorer](image)
Suggested answers to activities

Feedback

Activity 5.1

1. B

2. C

3. name, input

Activity 5.2

1. A query string is a part of the Uniform Resource Locator (URL); it contains data to be passed to another page. A typical URL containing a query string is:

   http://web/page.php?(query_string)

2. D

3. The HTML form content is encoded as a query string when the form is submitted via the GET method. On the other hand, the form content submitted via the POST method will be sent as the body of the request, not in the URL. To retrieve field variables via the POST method in PHP, the built-in array variable $_POST[] will be used. To retrieve field variables via the GET method in PHP, the built-in array variable $_GET[] will be used.
5.2 Using Cookies to Save State Information

Objectives

By the end of this section, you should be able to:

1. Explain what a cookie is in PHP.
2. Implement cookies with PHP.
3. Describe how to retrieve cookie data.

Introduction

Cookies are small text files that you can store on the computer of a visitor to your website. Cookies can be used to personalise a user’s experience of your website by presenting information in the same way during each visit or eliminating the need for repeated logins. Cookies can be used to automatically keep track of the browsing and buying habits of your visitors. Before we discuss how to implement cookies with PHP, let us get an idea of what a cookie is in the context of PHP.

What is a cookie?

A cookie is a small text file that is placed on the computer of a visitor by a website. It contains a unique tracking number that enables websites to remember or recognise visitors. Hence, cookies can be used to manage user information. When a user visits your website, you can create a cookie to store information about the user, such as preferences and browsing habits. When the user executes a request for another page, your server retrieves the user’s information from the cookie and uses it, for example, to eliminate the login prompt or to personalise the page.

A cookie can be used to store a session ID (SID). As the client navigates through your site, the SID is retrieved and used when necessary. Various items of data related to the SID will be furnished for use within the page. The SID will be explained in detail in the next section.

As a cookie can remain on the user’s browser even after a session ends (after the web browser or computer is shut down), the cookie can be read during other sessions as well. In other words, the persistency of the user’s preferences is maintained even over long periods of time. Some cookies are set to expire after a certain period of time to avoid the storage of outdated data.
You must understand that cookie acceptance is ultimately controlled by the user. There may be the possibility that a user will disable cookie support within the browser or will clear cookies from his or her computer. To remedy this situation, you can use the URL rewriting method. This method simply appends the SID to every local URL found within the requested page. It results in automatic SID propagation whenever the user clicks on one of your website’s links. Therefore, the URL rewriting method helps to maintain the SID even if the user disables cookie support. The following is an example that shows how URL rewriting is used to append the SID to a hyperlink:

```php
<?php session_start(); ?>
<a href="url_rewriting.php?self_defined=<?php echo session_id(); ?>" >URL rewriting</a>
<br/>
// Output:
//url_rewriting.php?self_defined=c8nus9v9qahllaaqcmqa5rf1od4
```

Another way is to turn off the `use_cookies` flag to get the SID and the default session name, PHPSESSID. We will discuss sessions more thoroughly in the next section.

```php
<?php
ini_set("session.use_cookies",0);
session_start(); ?>
<a href="url_rewriting.php?<?php echo SID; ?>" >
URL rewriting</a>
<br/>
// Output: url_rewriting.php?PHPSESSID=c8nus9v9qahllaaqcmqa5rf1od4
```

The URL rewriting method has its drawbacks. It does not allow persistency between sessions because the process, which automatically appends an SID to the URL, will no longer continue when the user leaves your website or starts a new session. There is also the possibility of a user copying a URL into an email and then sending it to another user. As long as the session has not expired, the session will be continued on the email recipient’s machine. This will cause a conflict if both users are simultaneously navigating the same website. It may also cause a breach of security to occur if the recipient of the link was not meant to see the data unveiled in that session. For these reasons, the cookie-based method is still preferable.

### Setting a cookie

To set a cookie in PHP, you will use the `setcookie()` function. Here is the syntax of this function:

```php
bool setcookie (string $name [, string $value [, int $expire ]])
```
The $name field is a mandatory field. The parameter $value can be a value such as the visitor's name. The $expire parameter is the time in seconds the cookie will exist before it expires. Specifying time() + 60 * 60 * 24 * 365 for the $expire parameter will keep the cookie alive for one year. If you do not set the expiry time, the cookie will expire once the web browser is closed.

As an example, let us set a cookie called “username” on Alice’s computer with an expiry time of one hour:

```php
<?php
  setcookie("username", "Alice", time()+3600);
?>
<html>
....
```

Note that cookies need to be placed at the very top of a web page, before the first `<html>` tag. Otherwise, you may get an error. If you want to set the cookie for a longer period of time like a month, you can use the time() function. To set the cookie to last for a month (60 sec/min * 60 min/hour * 24 hours/day * 30 days/month), use the following PHP code:

```
$expire=time()+60*60*24*30;
setcookie("username", "Alice", $expire);
```

In case you need to delete a cookie, you can set the cookie to an expired date or time by using the minus operator in the $expire field and leaving $value as an empty string. The following example code sets the cookie expiration date to one hour ago:

```
$expire=time()-3600;
setcookie("username", "", $expire);
```

**Retrieving cookie data**

To retrieve cookie data in PHP, you will use the $_COOKIE function. For example, to retrieve the value of a cookie named “username” and print it out, you can use:

```
echo $_COOKIE["username"];
```

To view all cookies available, you can use the print_r function:

```php
print_r($_COOKIE);
```
You may also check if a cookie has been set before you display it by using the `isset()` function. The following code uses the `isset()` function to check that a cookie has been previously set and then prints a greeting message to a customer returning to your website.

```php
if (isset($_COOKIE["username")))
    echo "Welcome " . $_COOKIE["username"] . "!\n</br>";
else
    echo "Welcome guest!\n</br>";
```

**Web Reference**

To learn more about setting cookies, please visit:


**Activity 5.3**

1. Explain what a cookie is and how it is used in a web application.

2. Some cookies are set to _________ after a certain period of time to avoid the storage of outdated data. There is the possibility that a user will _________ cookie support within the browser to protect his or her privacy.

3. The URL rewriting method appends the SID to every local URL found within the requested web page. This helps to maintain the SID even if a user disables cookies. True / False?

4. Which of the following is the PHP code that will create a cookie named “user” that will expire in one day?

   A. `create_cookie("user", "Shion", time()+60*60*24);`
   B. `cookie("user", "Alice", time()+60*60*24);`
   C. `setcookie("user", "Mary", time()+60*60*24);`
   D. `set_cookie("user", "John", time()+60*60*24);`

5. Write PHP code to retrieve and echo the cookies named “product”, “age”, “gender”, “location” and “language”.


Summary

In this section, you learnt about cookies and how to implement cookies with PHP. A cookie contains a unique tracking number that enables websites to remember or recognise visitors. Cookies can be used to manage user information, eliminate login prompts or to personalise a web page for a specific returning user.

To set a cookie in PHP, you use the `setcookie()` function and insert the following parameters: cookie name, cookie value and cookie expiration time. To retrieve cookie data, you use the `$_COOKIE` function with the cookie’s name as its parameter.

In the next section, you will learn how to use sessions to save state information. You will implement a session and work with session IDs and look at a session management example.

Self-test 5.2

1. Most of the time, information in a cookie is coded in a way that it is unreadable to any third party who happens to access your cookie folder. The only computer that can read and decode the information is the server that created the cookie in the first place. True / False?

2. What is the URL rewriting method? What are the pros and cons of using URL rewriting?

3. Which line of PHP code below will delete the cookie called “user”?

   A. `setcookie("user", "", time()-60*60*24);`
   B. `deletecookie("user", "", time()-60*60*24);`
   C. `unset_cookie("user", "", time()-60*60*24);`
   D. `delete_cookie("user", "", time()-60*60*24);`
4. Write PHP code to create cookies that will last for three months (assume that there are 30 days per month) based on the following table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>MPV</td>
</tr>
<tr>
<td>Colour</td>
<td>Black</td>
</tr>
<tr>
<td>Model</td>
<td>CX9</td>
</tr>
<tr>
<td>Price</td>
<td>200000</td>
</tr>
</tbody>
</table>

5. Based on the table in question 4, write PHP code to retrieve and echo all the data in the cookies.

6. Consider the following online form:

- a. Write PHP code to retrieve the information in the form by using the POST method.

- b. Create cookies that will last for two months based on the information retrieved in question 6a.
c. Retrieve and echo all the cookie data from question 6b. The web page that shows the retrieved data should look like the one in the following figure. By using the `isset()` function, check to see if a cookie has been set before you display it.

![Web page showing cookie data](image)

**Suggested answers to activity**

**Feedback**

**Activity 5.3**

1. A cookie is a small text file that contains a unique tracking number that enables websites to remember users when they make subsequent visits. When a user visits a web application, a cookie can be created and saved in the user’s browser to store information about the user, such as his or her preferences. As the user executes a request for another web page on the website, the server retrieves the user’s information from the cookie and uses it, for example, to personalise the page or to eliminate the need for the login prompt.

2. expire; disable

3. True

4. C

5. ```
   echo $_COOKIE["product"];
   echo $_COOKIE["age"];  
   echo $_COOKIE["gender"];  
   echo $_COOKIE["location"];  
   echo $_COOKIE["language"];  
```
5.3 Using Sessions to Save State Information

Objectives

By the end of this section, you should be able to:

1. Start and read sessions.
2. Pass information through sessions.
3. Explain the session management mechanism in web database applications.

Introduction

Passing values through a URL works fine provided that the information is not sensitive or confidential in nature. If you are transmitting information like usernames or passwords, better methods exist for passing the information and keeping it private. For this, you have the option of using either cookies or sessions. However, if you are looking for a method to maintain data only in a particular browsing session (meaning that the data will be removed once the user shuts down his or her web browser), you may want to consider using a session.

This section will introduce you to the implementation of sessions. You will learn about session creation, destruction and designation, as well as the retrieval of the session ID (SID) and the storage and retrieval of session variables.

Starting a session

A session is generally a temporary set of variables that exist only until a user shuts down his or her web browser. An example of a session variable is the session ID (SID), which is a variable that can be used to indicate that a user has been authenticated to login to your website. This information is stored temporarily for the PHP program to refer to throughout the browsing session.

A server distinguishes among multiple sessions being implemented on it simultaneously by referring to the SID. The SID is a unique identifier, usually in the form of a hash generated by a hash function. Every session is assigned a unique SID when the session is created. The browser retains the SID and uses it to let a server know which session to use.
Before you start using session variables, you need to start a session. This needs to be done on every web page that makes use of the session variables. It is important to note that a session must be started before anything else is coded. To start a session, you will use the following function:

```php
session_start();
```

Next, you need to decide what information will be stored in your session. You can store anything that you retrieve from the database in your session. Generally, the information that you will store is the username and other login information. It could also be a set of preferences obtained from a particular user. Of course, the information that can be stored in the session is not limited to user information. You can even store information such as the total cost of products selected by a user.

**Destroying a session**

Even if we know that a session has been pre-configured to expire based on an expiration time or when the user exits his or her browser, it is good to know how to cancel the session yourself. This is useful if a user logs off from the website. When the user clicks on the log off link, you can erase the session variables from memory or completely destroy the session from storage.

To erase all session variables stored in the current session, you can use the following function:

```php
session_unset();
```

This will effectively reset the session. However, `session_unset()` does not completely remove the session from the storage mechanism. If you want to completely destroy the session, you need to use the `session_destroy()` function. This will ensure the session is removed from the storage mechanism.

**Setting and retrieving the session ID**

The session ID (SID) is used to identify all session data tied to a particular user during a browsing session. Although PHP creates and uses the SID automatically, there are some circumstances in which you may want to set or retrieve it manually. Since the SID is unique for every browsing session, you can use it to keep track of sessions so that they can be easily differentiated.

Imagine that visitor A and visitor B are visiting your online shopping site at the same time. Visitor A’s SID is ABC123 and visitor B’s SID is XYZ789. Both visitors are adding your products into their shopping carts. You can identify the items selected by the visitors based on their SIDs. To do this, you will assign the SID ABC123 to all items (e.g., items J and K) selected by visitor A, and the SID XYZ789 to all items (e.g., items J and T) selected by visitor B. This will ensure
that the products selected by the visitors do not get mixed up. See Figure 5.1 for an illustration of this process.

![Figure 5.1 Use of SID in an e-commerce shopping cart](image)

You can force visitors to login before they are allowed to view or add products to the shopping cart, but by doing this you may miss the selling opportunity. Some visitors find it troublesome to register and login if they are still not sure about buying the products. By using the SID method, you can push the registration or login procedure to the end of the shopping process. By that time, the visitors have already selected the products that they are interested in, and thus, it will be more likely that they will go through the registration or login procedure.

To set or get the SID, you can use the following function:

```php
session_id([string sid])
```

To get the current SID, you can write the following PHP code:

```php
session_start();
echo "Your SID is " . session_id();
// Output: Your SID is 40j2m6nb798lpr57hvod849if0
```
If no parameter is passed to the function, it will return the current SID. If the optional SID is included, the current SID will be replaced with that value. Here is an example:

```php
session_start();
session_id("johnny123");
echo "Your SID is " . session_id();
// Output: Your SID is johnny123
```

Creating and deleting session variables

You can use session variables to manage users’ data or any other information that you intend to transfer from one web page to another. You can easily set and delete these variables just like any other variable. Once a session is created, variables may be stored in and retrieved from the `$_SESSION[]` array variable. Here is an example:

```php
session_start();
$_SESSION[’email’] = “johndoe@web.com”;
echo “Your email is “ . $_SESSION[’email’];
// Output: Your email is johndoe@web.com
```

To delete session variables, you will use the `unset()` function. For example:

```php
session_start();
$_SESSION[’email’] = “johndoe@web.com”;
unset($_SESSION[’email’]);
echo “Your email has been unset to empty: “ . $_SESSION[’email’];
// Output: Your email has been unset to empty:
```

Activity 5.4

1. Explain what `session` and `session ID` mean.

2. To start a session, you will use the `start_session()` function. True / False?

3. Which of the following is NOT a valid way of resetting a session in PHP?

   A. `session_unset();`
   B. `session_kill();`
   C. `unset($_SESSION[’name’]);`
   D. `session_destroy();`
4. Write PHP code to start a session, set a session variable called “username” to “lerdorf” and then print the retrieved value of the session variable.

Session management example

Now that you have been introduced to the basic functions that make session handling work, let us consider a real-world example of session management. This example illustrates a popular mechanism that automatically authenticates returning registered website users.

In web database applications, it is common to create a login form that is able to authenticate a visitor’s identity. However, once a visitor registers, logs in and is authenticated, it is convenient to allow the visitor to return to your site later without having to repeat the login process. If you want to keep your visitor logged in and not have to go through the login process for subsequent visits, you can use the \$_COOKIE function. If you want to maintain this mechanism only in the visitor’s current web browsing session due to security concerns, you can use the \$_SESSION management method. You just need to store a session variable once the visitor’s identity is authenticated. If the session variable is set, your visitors can log in to your website. Otherwise, you will show them a login form.

We will use the CUSTOMER table in the SHOPPING database that you created in the Unit 4 Practice Exercise for this example. If you have yet to create them, you may use the following SQL script to create the database and table. Refer to the Unit 4 subsections Creating a database and Creating tables for further details on executing the SQL script.

```
/****** CREATE DATABASE statements *******/
CREATE DATABASE SHOPPING;
USE SHOPPING;
/****** CREATE TABLE statements *******/
CREATE TABLE CUSTOMER (CustomerID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 Fullname VARCHAR(50) NOT NULL,
 Email VARCHAR(50) NOT NULL,
 Password CHAR(41) NOT NULL,
 Address TEXT );
/****** INSERT CUSTOMER detail *******/
INSERT INTO CUSTOMER VALUES
(NULL, ‘John Doe’, ‘johndoe@tcc242.com’, PASSWORD(‘wawasan’),
 ‘1 Seattle, Washington’);
```
The following PHP code is used to create and run session management for the auto-login process. Create a PHP extension file called 'example.php' to store the following code:

```php
<?php
    session_start();
    // Check if a session has been initiated previously?
    if (!isset($_SESSION['email'])) {
        // Did the user submit the form?
        if (isset($_POST['email'])) {
            $email = trim($_POST['email']);
            $password = trim($_POST['password']);

            // Connect to the MySQL server and select the database
            $mysqli = new mysqli("localhost", "root", "wawasan",
            "shopping");
            if ($mysqli->connect_errno) {
                printf("%s: %s", $mysqli->connect_errno, $mysqli->
                >connect_error);
            }

            // Look for the user in the customer table.
            $query = "SELECT * FROM customer WHERE Email = "
            "$email" AND Password = PASSWORD("$password");
            $result = $mysqli->query($query, MYSQLI_STORE_RESULT);
            if(!$result) {
                echo($mysqli->error);
                exit();
            }

            // If the user is located, return one row of results
            if ($result->num_rows == 1) {
                $_SESSION['email'] = $email;
                echo "You’ve successfully logged in. ";
            } else {
                // otherwise it should be an invalid login
                echo "Invalid email or password";
            }
        } else {
            // If the user has not logged in, show the login form
            include "login.php";
        }
    } else {
        // Substring email as username
        $symbolPos = strpos($_SESSION['email'], ";
        $username = substr($_SESSION['email'], 0, $symbolPos);

        // The user has returned. Offer a welcome note to the user.
        printf("Welcome back, %s!", $username);
    }
?>
```
The following is the code for creating the login form that is to be displayed if the user has not logged in before. You will create a file called ‘login.php’ to store the following code:

```html
<HTML>
  <HEAD>
    <TITLE>LOGIN</TITLE>
  </HEAD>
  <BODY>
    <p>
      <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
        Email:<br /><input type="text" name="email" size="50" />
        Password:<br /><input type="password" name="password" size="50" />
        <input type="submit" value="Login" />
      </form>
    </p>
  </BODY>
</HTML>
```

You should now test the session management mechanism by accessing ‘example.php’. Log in with the email: johndoe@tcc242.com and the password: wawasan. You should see the message “You've successfully logged in.” when you log in. Browse to another page and then return to ‘example.php’. You should then see the message “Welcome back, johndoe!”.

Visitors to your website should find this login mechanism helpful in reducing the hassle of logging in. You can change this mechanism to use the `$_COOKIE` function if you want to keep your visitor logged in for subsequent visits. The trick is to replace the `$_SESSION` function with the `$_COOKIE` function. Refer to the previous section on cookies if you want to do this. We will leave this as an exercise for you.

---

**Web Reference**

To learn more about PHP sessions, please visit:

Summary

In this section, you have learnt to save state information using sessions. A session is a temporary set of variables that exist only until you shut down your web browser. To start a session, you will use the `session_start()` function. To erase all session variables stored in the current session, you will use the `session_unset()` function. If you want to completely destroy the session, you can use the `session_destroy()` function.

The session ID (SID) keeps together all session data related to a user’s browsing session. To set or get the SID, you will use the `session_id([string sid])` function. Once a session is created, variables may be stored in and retrieved from the `$_SESSION[]` array variable. To delete the session variables, you will use the `unset()` function. In this section, you also saw how session management can be carried out through an auto-login example.

To summarise, you should now be familiar with:

1. Starting and reading sessions.
2. Working with a session ID.
3. Passing information through sessions.
4. The session management mechanism in a web application.

Self-test 5.3

1. Session variables can be set to exist even after the web browser has been terminated. True / False?
2. A session must be started before anything else is coded. True / False?
3. To start a session, you will use the ________ function. To completely destroy a session, you will use the ________ function.
4. Use PHP code with session ID to output the following results:
   
   My self-defined Session ID: 1d0g7rdpu0lsnc9j74vdathj2
5. Use PHP code to create session variables based on the following table.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID</td>
<td>session_id()</td>
</tr>
<tr>
<td>ServerAPI</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td>HTTP_ACCEPT_LANGUAGE</td>
<td>en-us</td>
</tr>
<tr>
<td>User-Agent</td>
<td>Mozilla/4.0</td>
</tr>
<tr>
<td>NUMBER_OF_PROCESSORS</td>
<td>2</td>
</tr>
<tr>
<td>mysqli</td>
<td>mysqliND 5.0.7</td>
</tr>
<tr>
<td>X-Powered-By</td>
<td>PHP/5.3.4</td>
</tr>
<tr>
<td>UA-CPU</td>
<td>x86</td>
</tr>
</tbody>
</table>

6. Based on the table given in question 5, write PHP code to retrieve and echo all the session data.

7. Consider the following online form:

   a. Use PHP code with the GET method to retrieve information from the online form.

   b. Create session variables for “Name”, “Contact” and “Email” based on the information retrieved from question 7a.
c. Retrieve and echo the variables and session data from question 7b. The result should look like the figure shown below. By using the `isset()` function, check whether a session has been set before you display the session variables.

d. Clear all the session variables created.

![Image of a web page with session data]

Suggested answers to activity

Feedback

Activity 5.4

1. A session is a temporary set of variables that exist only until you shut down your web browser. Sessions can be used to indicate that a person has logged in to your website. Sessions can also hold other variables that you may want to reference at any time. A session ID (SID) is a unique identifier, usually in the form of a hash generated by a hash function. When a session is created, it is assigned a SID, which the browser retains. The SID is used to let the server know which session to use.

2. False

3. B

4. `session_start();
$_SESSION['username'] = "lerdorf";
echo $_SESSION['username'];`
Summary of Unit 5

Summary

In this unit, you learnt about saving state information and passing this information across web pages. You used form elements to collect data and send them over to the server. You also learnt how to work with query strings.

You examined how cookies are used to save state information relating to visitors to your website. You were shown how to set and retrieve cookies. In addition, you studied how to create and maintain a session. You learnt several session operations that use the session ID and worked through a simple session management example.

The course materials should by now have equipped you with the necessary PHP and MySQL skills that you need to start building a dynamic web database application. You should now attempt the Unit Practice Exercise, in which you will build an e-commerce shopping cart.
Course Summary

Summary

Congratulations on completing this course. You should now have a good understanding of the fundamental topics in web database application development using PHP and MySQL.

In Unit 1, you were introduced to the fundamentals of web application development, Apache web server, PHP and MySQL. You were shown how to install these software packages on your computer. Basic PHP syntax was introduced in this unit as well. Unit 2 focused on the basics of the PHP scripting language. Various PHP data types, customised functions and control structures were introduced in this unit. Unit 3 discussed the intermediate level features of the PHP scripting language. The unit discussed strings, arrays and files. In Unit 4, you learnt how PHP is integrated with MySQL to build a web database application. The interaction process between PHP and MySQL, which uses PHP and MySQL library functions, was described in this unit. The final unit (Unit 5) of this course discussed how you can keep track of web page state information even when you use HTTP, which is a stateless protocol. The principles of using sessions and cookies to save state information were stated and explained in this unit.
Unit Practice Exercises

1. In this exercise, you are going to use most of the PHP and MySQL functions that you learnt from Unit 1 to Unit 5. You will enhance and complete the e-commerce shopping cart that you created in the Unit 4 Practice Exercise. If you have not already created a database call SHOPPING, please refer to question 1 of the Unit 4 Practice Exercise.

You will build the shopping cart based on the activity flowchart shown in Figure 5.2. First, the customer will arrive at the product selection page (‘store.php’). The customer will then add items to the cart (‘cart.php’) by specifying item quantities. In ‘cart.php’, the customer is able to update an item’s quantity or remove the item from the cart. After finalising item selection, the customer will proceed to checkout and the ‘login.php’ page will be displayed to prompt the customer to sign in. If the customer is a new customer, he or she will be sent to ‘register.php’ to register an account with your website. However, if the customer is already registered, he or she will use his or her email address and password to login. If the login is successful, the account details page (‘details.php’) will be displayed for the customer to verify details like shipping address and contact information. A page showing a summary of the items purchased will be displayed when the customer confirms an order.

![Figure 5.2 Shopping cart activity flow](image)

2. You will apply the session management technique that you learnt in this unit in your e-commerce site. You will add some useful links and separate the session and connection code so that they can be reused.

   a. The session will first check if the session variables are loaded. If they are loaded, load the customer’s full name into a session variable called “fullname” and greet the returning customer. Otherwise display “Welcome, Guest”.
b. You may also want to create some useful links that allow your customers to navigate to your product page, view the shopping cart or logout.

c. Since you want the links and database connections to appear on every page of your website, separate the code into two files: place the session code into ‘session.php’ and the database connection code into ‘connect.php’. Call them from the product page by using the include function.

d. Update the form action attribute value to ‘cart_update.php’ instead of ‘MySite14.php’. Add a hidden field for session ID to save state information.

You may copy ‘MySite13.php’ from Unit 4 or create a page that displays all the products from the PRODUCT table. Refer to the Unit 4 Practice Exercise for specific instructions. Store your web page in the '/var/www/Unit5' directory (or '\Apache Directory\htdocs\Unit5' in Windows) and name it 'store.php'.

The description of the page is as follows:

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>&lt;Head&gt;&lt;Title&gt;...&lt;/Title&gt;&lt;/Head&gt;</td>
<td>Store</td>
</tr>
</tbody>
</table>
| Apply session management| Call the session ID and start the session. Check if the session variable “fullname” is set:  
• If it is set, load the session variable and add a hyperlink to ‘logout.php’  
• If it is not set, print “Welcome, Guest” | session_id();  
session_start();  
if (isset($_SESSION['fullname']))  
{  
   // load the session variable and add a hyperlink to logout.php.  
}  
else  
{  
   // print a ‘Welcome, Guest’.  
} |
| Create useful links     | Create a table that looks neat and add:  
• a hyperlink to 'store.php' with the title: "Keep Shopping"  
• a hyperlink to 'cart.php' with the title: "View Cart"  
• a hyperlink to 'logout.php' with the title: "Logout". The logout page simply clears all the session variables and redirects to 'store.php' | echo "<table border='0' width='50%'><tr>";  
echo "<td><a href='store.php'>Keep Shopping</a></td>";  
echo "<td><a href='cart.php'>View Cart</a></td>";  
echo "<td align='right'><a href='logout.php'>Logout</a></td>";  
echo "<tr><td>";  
if (isset($_SESSION['fullname']))  
{  
   ...  
}  
else  
{  
   ...  
}|
Separate the session and connection code so that they can be reused on other PHP pages

- Create a new page called ‘session.php’ and move the session code and links there
- Create a new page called ‘connect.php’ and move the connection code there
- Add the include function to the pages in ‘store.php’

```php
<?php
if (isset($_SESSION['fullname']))
...
?>
```

```php
$conn = new mysqli...
...
?>
```

```php
session_id();
session_start();
include("session.php");
include("connect.php");
```

Update form action attribute value and add a hidden field

- Change form action attribute value from ‘MySite14.php’ to ‘cart_update.php’
- Add the session ID value to a hidden field called “sID”. Make sure the hidden field is placed within the `<form>` tag

```php
store.php
echo "<form method='POST' action='cart_update.php'>";
--
echo "<input type='hidden' name='sID' value='$sID'>";
--
echo "</form>";
```

The completed page should look like the one in Figure 5.3.
3. Next, you will enhance the shopping cart to support multiple users by using session IDs and adding the cart update and deletion features. Since each customer has a unique session ID during a browsing session, items added by each customer to the cart will not result in conflicts.

a. Enable your customer to view the cart. Enhance the shopping cart query method so that it can search for items based on the current session ID. If no items have been selected in a session, display “No items in your shopping cart”.

b. Provide the cart update feature to enable customers to update the quantities of the products selected.

c. Provide the cart item delete feature to enable customers to remove an item from the cart.

d. Save the total order amount into a session so that you do not have to carry this information from page to page.

e. Change the HTML main form action attribute value to ‘login.php’ instead of ‘MySite15.php’. Remove the duplicate “Keep Shopping” link.

You may copy ‘MySite14.php’ from Unit 4 or create a page that displays the SHOPPING_CART table based on the current session ID. Refer to the Unit 4 Practice Exercise for specific instructions. Store the page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>\htdocs\Unit5’ in Windows) and name it ‘cart.php’.

The description of the page is as follows:

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;…&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>Shopping Cart</td>
</tr>
<tr>
<td>Add the PHP session and connection codes</td>
<td>Retrieve the current SID and use the PHP include function</td>
<td><code>$sID = session_id();</code>&lt;br&gt;<code>session_start();</code>&lt;br&gt;<code>include(&quot;session.php&quot;);</code>&lt;br&gt;<code>include(&quot;connect.php&quot;);</code></td>
</tr>
<tr>
<td>Remove the existing INSERT query</td>
<td>Remove the INSERT INTO SHOPPING_CART query. You will later add it to ‘cart_update.php’. Check to see if an item has already been added before making the INSERT query</td>
<td>Remove:&lt;br&gt;<code>$query = “INSERT INTO SHOPPING_CART...</code></td>
</tr>
</tbody>
</table>
View cart
- Search for shopping cart items based on the current session ID
- Use `num_rows` to check for empty items. If there are no items in this session, display "No items in your shopping cart"

```php
$query = "SELECT ProductID, Quantity FROM SHOPPING_CART WHERE SessionID = '$sID';
```

```php
if ($result->num_rows == 0) {
    echo "</table><h3>No items in your shopping cart</h3>";
```

Provide the cart update feature
- Add a form into the "Qty." text field using a loop. Use the POST method and send form data to 'cart_update.php'
- Add two hidden fields named "prodID" and "sID" to store "$ProductID" and "$sID" respectively
- Add an "Update" button to the form

```php
while(list($ProductID, $Name, $Desc, $Price) = $result2->fetch_row()) {
    echo "<tr>"
    echo "$ProductID<br />
    echo "<input type='text' name='qty' size='2' value='$CartQty'><br />
    echo "$Name</td>"
```

Provide the item delete feature
- Add another form into the "Item ID" field using a loop. Use the POST method and send form data to 'cart_delete.php'
- Add two hidden fields named "prodID" and "sID" to store "$ProductID" and "$sID" respectively
- Add a "Delete" button to the form

```php
echo "</td><form ...
```

Save the total order amount into a session
- Place the value in the "orderTotal" hidden field into a session variable

```php
echo number_format($OrderTotal, 2);
$_SESSION['orderTotal'] = $OrderTotal;
```

Update the HTML main form action attribute value and remove the duplicate "Keep Shopping" link
- Change the main HTML form action attribute value from 'MySite15.php' to 'login.php' and move the open form tag to just before the "Checkout" button
- Remove the old "Keep Shopping" link after the table since the link is already available

```html
<tr>
    <td align="right">
        <form method="POST" action="login.php">
            <input type="submit" name="Submit" value="Checkout >"
        </form>
    </td></tr>
</table>
</BODY>
```

The shopping cart page should now look like the one shown in Figure 5.4.
4. Provide the cart update feature for customers to update the quantities of
   the products they selected.

   a. Check if the item has already been added. If the item exists, perform an
      UPDATE query. This will ensure that no duplicate items appear in the
      shopping cart in case the same item had been previously added.

   b. If the item is a new item, use the INSERT query (the query that you
      removed from ‘cart.php’) together with the SessionID.

   c. Print a message saying that the cart has been updated and display
      ‘cart.php’.

   Create a PHP page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>\`
   htdocs\Unit5’ in Windows) and name it ‘cart_update.php’.
5. Provide the cart item delete feature for customers to remove items from the cart.

   a. First check if the item has been added to the cart. This will ensure that item deletion can be successfully performed.

   b. If the item exists, perform the DELETE query.

   c. Print a message stating that the cart item has been deleted and display 'cart.php'.
Create a PHP page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>\htdocs\Unit5’ in Windows) and name it ‘cart_delete.php’.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/ Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
</table>
| Add the PHP connection code and retrieve variables                     | Use the PHP include function and retrieve the session ID and product ID using $_POST[] from the form | include("connect.php");
                                                                                     | $sID = $_POST['sID'];
                                                                                     | ...                                                                    |
| First check if the item has been added                                  | • Use the SessionID and ProductID to check for the item in SHOPPING_CART
                                                                                     • If the item does not exist, exit from the code| $query = "SELECT
                                                                                     SessionID, ProductID,
                                                                                     Quantity FROM SHOPPING_CART
                                                                                     WHERE SessionID = '$sID' AND ProductID = '$prodID'";
                                                                                     ...                                                                    |
                                                                                     if(!$result) {
                                                                                     echo($mysqli->error);
                                                                                     exit();
                                                                                     }
| If the item exists, perform the DELETE query                            | • If the item exists, perform the DELETE query using the SessionID and ProductID |
                                                                                     $query = "DELETE FROM
                                                                                     SHOPPING_CART WHERE
                                                                                     SessionID = '$sID' AND
                                                                                     ProductID = '$prodID'";
                                                                                     ...                                                                    |
| Print a message that the cart item has been deleted and display 'cart.php' | • Print "The item has been deleted" using the echo function
                                                                                     • Display 'cart.php' using the include function                         | echo "The item has been deleted<br/>
                                                                                     include("cart.php");
                                                              |

6. At the checkout stage, you need to get information from your customer. You will first create a sign-in page.

a. First check if the user has already logged in. If the user has already logged in, redirect the user to the details confirmation page.

b. Create a login form. Provide options for a new customer and a returning customer. For a returning customer, provide email address and password fields.

Create a PHP page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>\htdocs\Unit5’ in Windows) and name it ‘login.php’. 
The description of the page is as follows:

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
</table>
| Start a session and check for errors during authentication | • Add the session start function  
• Check if the session variable “fullname” is set. If it is, redirect to ‘details.php’  
• If there are errors during the authentication, retrieve and echo the $_GET[] array | session_start();  
if (isset($_SESSION['fullname'])) header(“Location: details.php”);  
if (isset($_GET['e'])) echo $_GET['e']; |
| Title | <Head><Title>…</Title></Head> | Sign in |
| Include the PHP session codes | Use the PHP include function | include(“session.php”); |
| Create a login form. Provide options for a new customer and a returning customer | • Provide a form using the POST method and set the value of the action attribute to ‘authenticate.php’  
• Provide an input field for the email address  
• Provide radio button inputs for new customers or returning customers  
• Provide a password field  
• Provide the “Submit” button | <form name="registration_form" method="post" action="authenticate.php">  
…  
<input type="text" name="email">…  
<input type="radio" name="customer_type" value="new" />…  
<input type="radio" name="customer_type" value="old" />…  
<input type="password" name="password">…  
<input type="submit" value="Next"> |

Your sign-in page should look like the one shown in Figure 5.5.
7. On the authentication page, check for customer types and authenticate the login information.

   a. Check for customer type. If the option is not selected, redirect to the login form and display a message asking the customer to select a customer type.

   b. If the “new customer” type is selected, redirect to the new customer registration page.

   c. For returning customers, process the information retrieved via $_POST. If the email address provided is invalid, redirect to the form and display a message telling the customer that the email address is invalid.

   d. Query the CUSTOMER table to check for email address and password matches. If the provided data match the data in the CUSTOMER table, apply session variables for “fullname” and “email” and redirect to ‘details.php’.

   e. If they do not match, redirect to the form and display an error message.

Create a PHP page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>\htdocs\Unit5’ in Windows) and name it ‘authenticate.php’.
The description of the page is as follows:

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
</table>
| Add the PHP session start function and retrieve and trim information from the form | • Add the session start function  
• Check if the session variable “fullname” is set. If it is, redirect to ‘details.php’  
• Retrieve and trim form information obtained via $_POST | ...  
```php
$email = trim($_POST['email']);
```  

| Check for customer type | If the option is not selected, redirect to the form and display an error message | if (!isset($_POST['customer_type']))  
```php
header("Location: ...
exit();
``` |

| For new customers | If the new customer type is selected, redirect to the new customer registration page  
• Copy the email variable to the registration page if the customer inserted an email address | ...  
```php
header("Location: register.php?email=$email");
exit();
``` |

| For returning customers, process the information retrieved | If the email address is invalid, redirect to the form and display an error message | ...  
```php
strpos($email, "@");
``` |

| Query the CUSTOMER table to check for email and password matches | Include a database connection  
• Make a query to the CUSTOMER table based on the login information  
• If login data match the data in the CUSTOMER table, apply session variables for “fullname” and “email” and redirect to ‘details.php’ | include("connect.php");  
```php
if ($result->num_rows == 1){  
$_SESSION['fullname'] = ...  
$_SESSION['email'] = ...  
header("Location: details.php");
``` |

| If login data does not match data in the CUSTOMER table | If they do not match, redirect to the form and display an error message | ... |
8. On the registration page, you will provide a form to collect the following customer information: Full Name, Email, Password, Confirm Password and Address.

Create a PHP page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>\htdocs\Unit5’ in Windows) and name it ‘register.php’.

The description of the page is as follows:

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve email query string and check for the error message query string. Include the PHP session code</td>
<td>• Retrieve email query string if it was passed from the login page &lt;br&gt;• If there are errors during the authentication, you will echo the error message query string by retrieving the $_GET[] array &lt;br&gt;• Use the PHP include function&lt;br&gt;... $GET['email']...&lt;br&gt;... $GET['e']...&lt;br&gt;include(&quot;session.php&quot;);</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td></td>
<td>Customer Registration</td>
</tr>
<tr>
<td>Create a registration form</td>
<td>• Provide a form that uses the POST method and set the value of the action attribute to 'reg_validation.php'&lt;br&gt;• Provide input fields for fullname, email, password, confirm password, and a text area for address&lt;br&gt;• Add in the email value retrieved from the login form&lt;br&gt;• Provide the “Register” button&lt;br&gt;... action=&quot;reg_validation.php&quot;&gt;&lt;br&gt;... &lt;input type=&quot;text&quot; name=&quot;email&quot; value=&quot;&lt;?php echo $email ?&gt;&quot;&gt;&lt;br&gt;... &lt;textarea name=&quot;address&quot; rows=&quot;3&quot; cols=&quot;16&quot;&gt;&lt;/textarea&gt;...</td>
<td></td>
</tr>
</tbody>
</table>
Your customer registration page should look like the one shown in Figure 5.6.

![Figure 5.6 New customer registration page](image)

9. On the registration validation page, you will validate the registration fields by using the string manipulation techniques you learnt in Unit 3.

   a. Retrieve and check the full name, email, password, re-entered password and address. If the information is invalid, redirect to the form and display an error message.

   b. If the information is valid, perform an INSERT query to save the data into the CUSTOMER table.

   c. Apply session variables for “fullname” and “email” and redirect to ‘details.php’.

Create a PHP page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>/htdocs/Unit5’ in Windows) and name it ‘reg_validation.php’.
The page is described in the following table.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve and trim information from the form</td>
<td>• Retrieve and trim form information using $_POST</td>
<td>$fullname = trim($_POST['fullname']);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Validate the retrieved information</td>
<td>• Make sure the string length of “fullname” is &gt; 3</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>• Check email address for the “@” symbol</td>
<td>strlen()...</td>
</tr>
<tr>
<td></td>
<td>• Make sure the string length of “password” is &gt; 4</td>
<td>strpos()...</td>
</tr>
<tr>
<td></td>
<td>• Ensure that the two passwords are the same</td>
<td>strcmp()...</td>
</tr>
<tr>
<td></td>
<td>• Make sure the address string is &gt; 5</td>
<td>...</td>
</tr>
<tr>
<td>INSERT into the database if all the information are valid</td>
<td>• Include the database connection code</td>
<td>include(&quot;connect.php&quot;);</td>
</tr>
<tr>
<td></td>
<td>• Perform an INSERT query to save the data into the CUSTOMER table</td>
<td>$query = &quot;INSERT INTO CUSTOMER...&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Apply session variables and redirect</td>
<td>• Start the session</td>
<td>session_start();</td>
</tr>
<tr>
<td></td>
<td>• Apply session variables for “fullname” and “email” and redirect to ‘details.php’</td>
<td>$_SESSION['fullname'] = ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>header(&quot;Location: ...&quot;</td>
</tr>
</tbody>
</table>

10. You will reuse the details page (‘details.php’) that was created earlier and make minimum changes to it in order to reflect the use of the session management concept.

   a. Retrieve customer information based on the email session variable.

   b. Change the HTML main form action attribute value to ‘summary.php’ from ‘MySite16.php’.

   c. Remove the “orderTotal” hidden field.

You may copy ‘MySite15.php’ from Unit 4 or create a page that shows customer details retrieved from the CUSTOMER table. Refer to the Unit 4 Practice Exercise for specific instructions. Save the page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>/htdocs/Unit5’ in Windows) and name it ‘details.php’.
The page is described in the following table.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;...&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>Confirm your details</td>
</tr>
</tbody>
</table>
| Add the PHP session and connection code                                | Use the PHP `include` function                                                     | `session_start();
include("session.php");
include("connect.php");`                                                                  |
| Retrieve customer information based on the email session variable      | • Provide a form that uses the POST method and set the action attribute value to 'reg_validation.php'
• Provide input fields for “fullname”, “email”, “password”, “confirm password” and a text area for “address”
• Add in the email value retrieved from the login form
• Provide the “Confirm” button                                           | `$sEmail = $_SESSION ['email'];
...
WHERE Email = "$sEmail"`;                                                            |
| Update the HTML main form action attribute value                       | • Update the HTML main form action attribute value to ‘summary.php’               | `<form method='POST'
action='summary.php'>
...`                                                                                                   |
| Remove the "orderTotal" hidden field                                  | • We no longer need to move the hidden field from page to page since we have already stored it in the session variable | Remove:
`<input type='hidden'
name='orderTotal'
value='$OrderTotal'>`                                                      |

The page should look like the one shown in **Figure 5.7**.
11. Lastly, you will enhance the summary page (‘summary.php’) so that it shows the customer’s billing details and the list of items purchased.

a. Update customer details in case there are any changes in the details page.

b. Update all $_POST['orderTotal'] using hidden fields to $_SESSION ['orderTotal'].

c. Modify the query that copies items from SHOPPING_CART to PRODUCT_ORDERS to include the clause “WHERE session id = ‘$sID’” since now you support multiple users based on their SID.

d. Also, modify the query that deletes items from SHOPPING_CART to include “WHERE session id = ‘$sID’”.

e. Rearrange the menu links to exclude the “View Cart” link.

You may copy ‘MySite16.php’ from Unit 4 or create a page that shows customer billing details and purchased items. Refer to the Unit 4 Practice Exercise for specific instructions. Store the page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>\htdocs\Unit5’ in Windows) and name it ‘summary.php’.

The page is described in the following table.

<table>
<thead>
<tr>
<th>Object</th>
<th>Markup/Instruction</th>
<th>Value/Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td><code>&lt;Head&gt;&lt;Title&gt;...&lt;/Title&gt;&lt;/Head&gt;</code></td>
<td>Summary</td>
</tr>
</tbody>
</table>
| Add the PHP session and connection code. Retrieve the SID and all variable fields from the details confirmation form | session_start(); $sID = session_id(); // UPDATE DETAILS $custID = $_POST ['CustID']; ...
$_SESSION['email'] = $email;
$_SESSION['fullname'] = $fullname;
include("connect.php"); | |
| Update customer details     | • Update customer details in case there are changes in the details page           | ...
$query = “UPDATE CUSTOMER SET ...
...
$_SESSION['orderTotal'] = $OrderTotal; | |
| Update all order total amounts | Change all $_POST['orderTotal'] to $_SESSION['orderTotal']                     | ...
$_SESSION['orderTotal'] = $OrderTotal; | |
Include session ID into the copy query since you now support multiple users based on SID

Include

| Session ID into the copy query since you now support multiple users based on SID | Modify the query that copies items from SHOPPING_CART to PRODUCT_ORDERS to include “WHERE session id = current SID” | --
| --FROM SHOPPING_CART WHERE SessionID = ‘$sID’; -- |

Modify the delete query to include session ID

Modify the delete query to include session ID

| Modify the query that deletes items from SHOPPING_CART by adding “WHERE session id = current SID” | -- | $query = "DELETE ... WHERE SessionID = ‘$sID’"; -- |

Rearrange the menu link

| Delete the “View Cart” link | Delete: | ...
| ...<a href='cart.php'>View Cart</a>... |

The page should look like the one shown in Figure 5.8.

![Figure 5.8 Billing and items purchased summary page](image)

12. Create a PHP page in the ‘/var/www/Unit5’ directory (or ‘<Apache Directory>/htdocs/Unit5’ in Windows) and name it ‘logout.php’. The logout page simply clears all the session variables and redirects to ‘store.php’.
13. Congratulations, you have completed your simple shopping cart system. To further enhance your shopping cart, you could include:

   a. Images of your products.

   b. A CSS style sheet to make your site look more presentable.

   c. A footnote to acknowledge the developer.

   d. The current date and a JavaScript running clock (you learnt these in the Unit 1 Practice Exercise).

Refer to Figure 5.9 for a sample of the enhancements described above.

Figure 5.9 Enhanced shopping cart
Suggested Answers to Self-tests

Feedback

Self-test 5.1

1. False

2. hidden

3. A

4. 

```php
$_GET['Country']; $_GET['Organizer']; $_GET['PhoneNumber'];
```

5. 

```php
$company_name = $_POST['company_name'];
$street_address = $_POST['street_address'];
$city = $_POST['city'];
$state = $_POST['state'];
$postal_code = $_POST['postal_code'];
```

```php
echo "Company name: " . $company_name;
```

```php
echo "<br>Street address: " . $street_address;
```

```php
echo "<br>City: " . $city;
```

```php
echo "<br>State: " . $state;
```

```php
echo "<br>Postal code: " . $postal_code;
```


Self-test 5.2

1. True

2. The URL rewriting method appends the SID to every local URL found within the requested page. It results in automatic SID propagation whenever the user clicks on one of those local links.

   Pros: The URL rewriting method helps to maintain the SID even if the user disables cookies.

   Cons: URL rewriting does not allow for persistence between sessions and it may cause problems if a user copies the rewritten URL into an e-mail and sends it to another user.
3. A

4. $expire=time()+60*60*24*30*3;
   setcookie("Type", "MPV", $expire);
   setcookie("Colour", "Black", $expire);
   setcookie("Model", "CX9", $expire);
   setcookie("Price", "200000", $expire);

5. echo $_COOKIE["Type"] . "<br>";
   echo $_COOKIE["Colour"] . "<br>";
   echo $_COOKIE["Model"] . "<br>";
   echo $_COOKIE["Price"] . "<br>";

6. a. $UnivInst = $_POST["UnivInst"];  
   $Year = $_POST["Year"];  
   $Faculty = $_POST["Faculty"];  
   $Division = $_POST["Division"];  
   $CGPA = $_POST["CGPA"];  
   $Scholarship = $_POST["Scholarship"];  

   b. $expire=time()+60*60*24*30*2;
      setcookie("UnivInst", $UnivInst, $expire);
      setcookie("Year", $Year, $expire);
      setcookie("Faculty", $Faculty, $expire);
      setcookie("Division", $Division, $expire);
      setcookie("CGPA", $CGPA, $expire);
      setcookie("Scholarship", $Scholarship, $expire);

   c. if (isset($_COOKIE["UnivInst"]))
      echo “University / Institution: “ . $_
      COOKIE["UnivInst"];  
      if (isset($_COOKIE["Year"]))
         echo “<br>Year: “ . $_COOKIE["Year"];  
      if (isset($_COOKIE["Faculty"]))
         echo “<br>Faculty of Studies: “ . $_
         COOKIE["Faculty"];  
      if (isset($_COOKIE["Division"]))
         echo “<br>Division: “ . $_COOKIE["Division"];  
      if (isset($_COOKIE["CGPA"]))
         echo “<br>CGPA Grade: “ . $_COOKIE["CGPA"];  
      if (isset($_COOKIE["Scholarship"]))
         echo “<br>Honour / Scholarship / Prizes Won: “ . $_COOKIE["Scholarship"];
Self-test 5.3

1. False

2. True

3. `session_start(); session_destroy()`

4. `session_start();
session_id("1d0g7rdpu01snq9jq74vdathj2");
echo “My self-defined Session ID: ” . session_id();`

5. `session_start();
$_SESSION[‘SID’] = session_id();
$_SESSION[‘ServerAPI’] = “Apache 2.0”;
$_SESSION[‘HTTP_ACCEPT_LANGUAGE’] = “en-us”;
$_SESSION[‘User-Agent’] = “Mozilla/4.0”;
$_SESSION[‘NUMBER_OF_PROCESSORS’] = "2";
$_SESSION[‘mysqli’] = “mysqli 5.0.7”;
$_SESSION[‘X-Powered-By’] = “PHP/5.3.4”;
$_SESSION[‘UA-CPU’] = “x86”;

6. echo “SID :” . $_SESSION[‘SID’];
echo “<br>ServerAPI :” . $_SESSION[‘ServerAPI’];
echo “<br>HTTP_ACCEPT_LANGUAGE :” . $_SESSION[‘HTTP_ACCEPT_LANGUAGE’];
echo “<br>User-Agent :” . $_SESSION[‘User-Agent’];
echo “<br>NUMBER_OF_PROCESSORS :” . $_SESSION[‘NUMBER_OF_PROCESSORS’];
echo “<br>mysqli :” . $_SESSION[‘mysqli’];
echo “<br>X-Powered-By :” . $_SESSION[‘X-Powered-By’];
echo “<br>UA-CPU :” . $_SESSION[‘UA-CPU’];

7. a. `$Title = $_GET[‘Title’];
   $Name = $_GET[‘Name’];
   $Contact = $_GET[‘Contact’];
   $ContactNumberType = $_GET[‘ContactNumberType’];
   $BestTimeToCall = $_GET[‘BestTimeToCall’];
   $Email = $_GET[‘Email’];
   $State = $_GET[‘State’];`
b. `session_start();
$_SESSION['Name'] = $Name;
$_SESSION['Contact'] = $Contact;
$_SESSION['Email'] = $Email;

c. if (isset($_SESSION['Name']))
   echo "Welcome " . $Title . " " . $_SESSION['Name'];
if (isset($_SESSION['Contact']))
   echo "<br>Contact: " . $_SESSION['Contact'] . "(" . $ContactNumberType . ")";
   echo "<br>Best time to call: " . $BestTimeToCall;
if (isset($_SESSION['Email']))
   echo "<br>Email: " . $_SESSION['Email'];
   echo "<br>Preferred branch: " . $State;

d. `session_unset();`
Suggested Answers to Unit Practice Exercises

Feedback

2. store.php

```html
<html>
  <head>
    <title>Store</title>
    <script src="clock.js" type="text/javascript"></script>
    <link href="style.css" type="text/css" rel="StyleSheet">
  </head>
  <body onload="setTimeout(showTime, 1000)">
    <h1>My Coffee Store.com</h1>
    <?php
    session_start();
    $sID = session_id();
    include("session.php");
    include("connect.php");
    $query = "SELECT * FROM PRODUCT";
    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);
    if(!$result) {
      echo($mysqli->error);
      exit();
    } else {
      while(list($ProductID, $Name, $Desc, $Price) = $result->fetch_row()) {
        echo "<table border='1' width='50%'>"
        echo "<tr>
          <td width='20%' align='center'>
            <img src='image/". $ProductID .".jpg' />
          </td>
          <td>
            <span class='header'>$Name<br />
            $Desc<br />
            <b>$$Price<br />
            </b>
            </form>
          </td>
        </tr>
        echo "</table>
      }
    }
  </body>
</html>
```


```php
session.php

<?php
    echo "<table border='0' width='50%'><tr>
        <td><a href='store.php'>Keep Shopping</a></td>
        <td align='right'><a href='cart.php'>View Cart</a></td></tr>
    echo "</tr></table>";
    if (isset($_SESSION['fullname'])) {
        echo "Welcome, ";
        echo $_SESSION['fullname'];
        echo "<td align='right'><a href='logout.php'>Logout</a></td>
    } else {
        echo "Welcome, Guest";
    }
    echo "</td></tr></table>";
    // set the default time zone
    date_default_timezone_set('Asia/Kuala_Lumpur');
    echo "Today is ";
    echo date('d F, Y');
    echo " <span id='clockFace'>";
    echo date('G:i:s');
    echo "</span>";
?>
```
connect.php

```php
<?php
    $mysqli = new mysqli("localhost", "root", "wawasan", "shopping");
    if ($mysqli->connect_errno) {
        printf("%s:%s", $mysqli->connect_errno, $mysqli->connect_error);
    }
?>
```

3. cart.php

```html
<html>
<head>
    <title>Shopping Cart</title>
    <script src="clock.js" type="text/javascript"></script>
    <link href="style.css" type="text/css" rel="StyleSheet">
</head>
<body oncreate="setInterval(showTime, 1000)">
<h1>My Coffee Store shopping cart</h1>
<?php
    session_id();
    session_start();
    $sID = session_id();
    include("session.php");
    include("connect.php");
?>
<table border="1" width="50%">
    <tr>
        <th>Qty.</th>
        <th>Item ID</th>
        <th>Item Name</th>
        <th>Item Info</th>
        <th>Price</th>
        <th>Total</th>
    </tr>
    <?php
    $query = "SELECT ProductID, Quantity FROM SHOPPING_CART WHERE SessionID = '$sID';";
    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);
    if (!$result) {
        echo($mysqli->error);
        exit();
    }
    $OrderTotal = 0;
```
if ($result->num_rows == 0) {
    echo "</table><h3>No items in your shopping cart</h3>";
    exit();
}
while (list($CartProdID, $CartQty) = $result->fetch_row())
{
    $query = "SELECT * FROM PRODUCT WHERE ProductID = '$CartProdID'";
    $result2 = $mysqli->query($query, MYSQLI_STORE_RESULT);
    if(!$result2) {
        echo($mysqli->error);
        exit();
    }
    while(list($ProductID, $Name, $Desc, $Price) = $result2->fetch_row())
    {
        echo"<tr>";
        echo "<td><form method='POST' action='cart_update.php'>";
        echo "<input type='hidden' name='prodID' value='$ProductID'>";
        echo "<input type='hidden' name='sID' value='$sID'>";
        echo "<input type='text' name='qty' size='2' value='$CartQty'><br />">";
        echo "<input type='submit' name='Update' value='Update'></form></td>";
        echo "<td><form method='POST' action='cart_delete.php'>";
        echo "#$ProductID<br />">";
        echo "<input type='hidden' name='prodID' value='$ProductID'>";
        echo "<input type='hidden' name='sID' value='$sID'>";
        echo "<input type='hidden' name='sID'>";
        echo "<input type='submit' name='Delete' value='Delete'></form>";
        echo "</td>";
    }
    $OrderTotal += $Total;
}
$mysqli->close();
?>
</table>
<table border="0" width="50%">
<tr>
<td align="right">
Order Total: $
<?php echo number_format($OrderTotal, 2);  
$_SESSION['orderTotal'] = $OrderTotal;  
?>
</td></tr>
<tr>
<td align="right">
<form method="POST" action="login.php">
<input type="submit" name="Submit" value="Checkout »">
</form>
</td></tr>
</table>
<?php include("footer.php"); ?>
</BODY>
</HTML>

4. cart_update.php

<?php
include("connect.php");
$sID = $_POST['sID'];
$qty =$_POST['qty'];
$prodID = $_POST['prodID'];
//Checking same items, same ID
$query = "SELECT SessionID, ProductID, Quantity  
FROM SHOPPING_CART WHERE SessionID = '$sID' AND  
ProductID = '$prodID'";
$result = $mysqli->query($query, MYSQLI_STORE_  
RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
// The item has already existed?
if ($result->num_rows == 1){
    $query = "UPDATE SHOPPING_CART  
SET Quantity = '$qty'  
WHERE SessionID = '$sID' AND ProductID =  
'$prodID'";
}
else {
    $query = "INSERT INTO SHOPPING_CART
(SessionID, Quantity, ProductID) VALUES ('$sID', '$qty', '$prodID');
}
$result = $mysqli->query($query);
if (!$result) {
    echo ($mysqli->error);
    exit();
}
echo "The cart has been updated<br />

include("cart.php");
?>

5. cart_delete.php

<?php
include("connect.php");
$sID = $_POST['sID'];
$prodID = $_POST['prodID'];
// Checking same items, same ID
$query = "SELECT SessionID, ProductID, Quantity FROM SHOPPING_CART WHERE SessionID = "'".$sID.'" AND ProductID = "'".$prodID.'"";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if (!$result) {
    echo ($mysqli->error);
    exit();
}
$query = "DELETE FROM SHOPPING_CART
    WHERE SessionID = "'".$sID.'" AND ProductID = "'".$prodID.'"";
$result = $mysqli->query($query);
if (!$result) {
    echo ($mysqli->error);
    exit();
}
echo "The item has been deleted<br />

include("cart.php");
?>
6. login.php

```php
<?php
session_start();
if (isset($_SESSION['fullname']))
    header("Location: details.php");
if (isset($_GET['e']))
    echo $_GET['e'];
?>

<HTML>
<HEAD>
<TITLE>Sign In</TITLE>
<script src="clock.js" type="text/javascript">
</script>
<link href="style.css" type="text/css" rel="StyleSheet">
</HEAD>
<BODY onload="setInterval(showTime, 1000)">
<h1>Sign In</h1>
<?php
include("session.php");
?>
<table>
<form name="registration_form" method="post" action="authenticate.php">
<tr><td>
Email: <input type="text" name="email"><br />
</td><tr><td align = 'right'
<input type="submit" value="Next">
</tr></form>
</table>
<?php include("footer.php"); ?>
</BODY>
</HTML>
```
7. authenticate.php

```php
<?PHP
session_start();
if (isset($_SESSION['fullname']))
    header("Location: details.php");
// Trim whitespace of all items
$email = trim($_POST['email']);
$password = trim($_POST['password']);
if (!isset($_POST['customer_type']))
{
    header("Location: login.php?e=Please select a
customer option");
    exit();
}
$customerType = $_POST['customer_type'];
if ($customerType == "new")
{
    header("Location: register.php?email=$email");
    exit();
}
$symbolPos = strpos($email, "@");
if ($symbolPos == FALSE)
{
    header("Location: login.php?e=Invalid email
address or password");
}
include("connect.php");
$query = "SELECT Fullname FROM customer WHERE Email = 
'email' AND Password = PASSWORD('password')";
$result = $mysqli->query($query, MYSQLI_STORE_ RESULT);
if (!$result) {
    echo($mysqli->error);
    exit();
}
if ($result->num_rows == 1)
{
    while ($row = mysqli_fetch_row($result))
        $_SESSION['fullname'] = $row[0];
        $_SESSION['email'] = $email;
    header("Location: details.php");
}
else
{
    header("Location: login.php?e=Invalid email
address or password");
}
?>
```
8. register.php

```php
<?php
    $email = "";
    if (isset($_GET['email']))
    {
        $email = $_GET['email'];
    }
    if (isset($_GET['e']))
    {
        echo $_GET['e'];
    }
    include("session.php");
?>

<HTML>
    <HEAD>
        <TITLE>Customer Registration</TITLE>
        <script src="clock.js" type="text/javascript">
        </script>
        <link href="style.css" type="text/css" rel="StyleSheet">
    </HEAD>
    <BODY onload="setInterval(showTime, 1000)">
    <h1>Customer Registration</h1>
    <table>
        <form name="registration_form" method="post" action="reg_validation.php">
            <tr><td>Full Name: </td><td><input type="text" name="fullname"></td></tr>
            <tr><td>Email: </td><td><input type="text" name="email" value="<?php echo $email ?>"></td></tr>
            <tr><td>Password: </td><td><input type="password" name="password"></td></tr>
            <tr><td>Confirm Password: </td><td><input type="password" name="password2"></td></tr>
            <tr><td>Address: </td><td><textarea name="address" rows="3" cols="19"></textarea></td></tr>
        </form>
    </table>
    <?php include("footer.php"); ?>
    </BODY>
</HTML>
```
9. reg_validation.php

```php
<?php

// Trim whitespace of all items
$fullname = trim($_POST['fullname']);
$email = trim($_POST['email']);
$password = trim($_POST['password']);
$password2 = trim($_POST['password2']);
/address = trim($_POST['address']);
// Make sure string length of name is > 3
if (strlen($fullname) < 3)
{
    header("Location: register.php?e=Invalid full
    name");
    exit();
}
// Check email for @ symbol
$symbolPos = strpos($email, "@");
if ($symbolPos == FALSE)
{
    header("Location: register.php?e=Invalid
    email");
    exit();
}
// Make sure string length of password is > 4
if (strlen($password) < 4)
{
    header("Location: register.php?e=Invalid
    password");
    exit();
}
// Compare if two passwords are the same
if (strcmp ($password , $password2 ) != 0)
{
    header("Location: register.php?e=Passwords not
    match");
    exit();
}
// Make sure address string is more than 5
if (strlen($address) < 5)
{
    header("Location: register.php?e=Invalid
    address");
    exit();
}
include("connect.php");
$query = "INSERT INTO CUSTOMER (CustomerID,
    Fullname, Email, Password, Address)
    VALUES (NULL, '$fullname', '$email', PASSWORD
    ('$password'), '$address')";
```
$result = $mysqli->query($query);
if(!$result) {
    echo($mysqli->error);
    exit();
}

session_start();
$_SESSION['fullname'] = $fullname;
$_SESSION['email'] = $email;
header("Location: details.php");
?>

10. details.php

<HTML>
<HEAD>
<TITLE>Confirm your details</TITLE>
<script src="clock.js" type="text/javascript"></script>
<link href="style.css" type="text/css" rel="StyleSheet"></HEAD>
<BODY onload="setInterval(showTime, 1000)">
<h1>Confirm your details</h1>
<?php
session_start();
include("session.php");
include("connect.php");
$sEmail = $_SESSION['email'];
$query = "SELECT CustomerID, Fullname, Email, Address FROM CUSTOMER WHERE Email = '{$sEmail}'";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
while(list($CustomerID, $Fullname, $Email, $Address) = $result->fetch_row()) {
    echo "<table border='1' width='50%'>";
    echo"<form method='POST' action='summary.php'";
    echo "<tr><td>Customer ID</td>";
    echo "<td><input type='text' name='CustomerID' value='$CustomerID' disabled><input type='hidden' name='CustID' value='$CustomerID'></td></tr>
    echo "<tr><td>Full Name</td>";
    echo "<td><input type='text' name='FullName' value='$Fullname'></td></tr>";
</form>"
};
?>

11. summary.php

```php
<?php
session_start();
$sID = session_id();
// UPDATE DETAILS
$custID = $_POST['CustID'];
$fullname = $_POST['FullName'];
$email = $_POST['Email'];
$address = $_POST['Address'];
// Update Session variables
$_SESSION['email'] = $email;
$_SESSION['fullname'] = $fullname;
include("connect.php");
$query = "UPDATE CUSTOMER SET
   Fullname = \\
   Email = \\
   Address = ";
```
WHERE CustomerID = $custID;
$result = $mysqli->query($query);
if(!$result) {
    echo($mysqli->error);
    exit();
}
// INSERT INTO ORDERS
date_default_timezone_set('Asia/Kuala_Lumpur');
//$custID = $_POST['CustID'];
$OrderTotal = $_SESSION['orderTotal'];
$today = date("y-m-d");
$query = "INSERT INTO ORDERS (OrderID, CustomerID, Date, Total) VALUES (NULL, '$custID', '$today', $OrderTotal)";
$result = $mysqli->query($query);
// Get back OrderID
$orderID = $mysqli->insert_id;
if(!$result) {
    echo($mysqli->error);
    exit();
}
// Copy SHOPPING_CART to PRODUCT_ORDERS WHERE
session id = current SID
$query = "SELECT ProductID, Quantity FROM SHOPPING_CART WHERE SessionID = '$sID';
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
    echo($mysqli->error);
    exit();
}
while(list($ProductID, $Quantity) = $result->fetch_row()) {
    $query2 = "INSERT INTO PRODUCT_ORDERS (OrderID, ProductID, Quantity) VALUES ('$orderID', '$ProductID', $Quantity)";
    $result2 = $mysqli->query($query2);
    if(!$result2) {
        echo($mysqli->error);
        exit();
    }
}
// Delete SHOPPING_CART items WHERE session id = current SID
$query = "DELETE FROM shopping_cart WHERE SessionID = '$sID';
$result = $mysqli->query($query);
if(!$result) {
    echo($mysqli->error);
exit();
}
echo "<table border='0' width='50%'><tr>
 echo "<td><a href='store.php'>Keep Shopping</a></td>
 echo "<td align='right'><a href='logout.php'>Logout</a>";
echo "</td></tr></table>";
// Some message
echo "<h1>Thank you for your order</h1>";
echo "<b>Bill to</b><br />";
// Print customer details
$query = "SELECT CustomerID, Fullname, Email, Address FROM CUSTOMER WHERE CustomerID = '$custID'";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
 echo($mysqli->error);
 exit();
}
while(list($CustomerID, $Fullname, $Email, $Address) = $result->fetch_row())
{
 echo "$Fullname (#$CustomerID)<br />";
 echo "$Email<br />";
 echo "$Address<br /><br />";
}
<p><b>Order ID</b> <?php echo $orderID ?><br />
<b>Summary of your purchase:</b></p>
<table border="0" width="50%">
<tr>
<th>Qty.</th>
<th>Item ID</th>
<th>Item Name</th>
<th>Price</th>
<th>Total</th>
</tr>
<?php
// Query purchased items from PRODUCTS_ORDER
$query = "SELECT ProductID, Quantity FROM PRODUCT_ORDERS WHERE OrderID = $orderID";
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);
if(!$result) {
 echo($mysqli->error);
 exit();
}
while (list($CartProdID, $CartQty) = $result->fetch_row())
{
$query2 = "SELECT * FROM PRODUCT WHERE ProductID = '$CartProdID'";
$result2 = $mysqli->query($query2, MYSQLI_STORE_RESULT);
if(!$result2) {
    echo($mysqli->error);
    exit();
}
while(list($ProductID, $Name, $Desc, $Price) = $result2->fetch_row()) {
    echo "<tr>
    echo "<td>$CartQty</td>
    echo "<td>#$ProductID</td>
    echo "<td>$Name</td>
    echo "<td>$$Price</td>
    $Extprice = number_format($Price * $CartQty, 2);
    echo "<td>$$Extprice</td>
    echo "</tr>
}

mysqli->close();
echo "<tr><td colspan='5' align='right'><b>Order Total: $" .
    echo number_format($OrderTotal, 2);
    echo "</b></td></tr></table>"
include("footer.php"); ?>
</BODY>
</HTML>

12. logout.php

<?PHP
    session_start();
    unset($_SESSION['email']);
    unset($_SESSION['orderTotal']);
    unset($_SESSION['fullname']);
    header("Location: store.php");
?>

13. a. image folder

12948.jpg
12983.jpg
22103.jpg
97902.jpg
b. style.css

HTML
{
    FONT-SIZE: 100%;
    padding: 0;
    margin: 0;
    border: 0;
}
BODY
{
    color:#FFF;
    background-color:$68c84;
    MARGIN: 0;
    FONT-FAMILY: Verdana,sans-serif;
    LETTER-SPACING: 0.01em;
    FONT-SIZE: 10px;
    text-align: center;
    padding: 0;
    /*overflow: scroll*/
}
TH
{
    FONT-FAMILY: Verdana, Arial;
    COLOR: #331c01;
    FONT-SIZE: 10px
}
TD
{
    FONT-FAMILY: Verdana, Arial;
    COLOR: #331c01;
    FONT-SIZE: 10px
}
form
{
    display:inline;
    padding: 0;
    margin:0;
} /* ---- links ---- */
a:link {color:#0b301b;}
a:visited {color:#0b301b;}
a:hover {color:$000; text-decoration:none}
a:active {color:$331c01; text-decoration:none}
p, h1, h2, h3, h4, h5, h6, ul, ol, dl, table
{margin:0 0 0; padding:0}
h1
{
    margin-top:1em;
    margin-bottom:0.2em;
Managing state information
c. footer.php

<br />
<span class="fine_print">The site is developed for WOU academic purposes<br />
by <a href="mailto:johndoe@nothing.com">John Doe</a></span>


d. clock.js

function showTime () {
    var time = new Date();
    var hour = time.getHours();
    var minute = time.getMinutes();
    var sMin = (minute<10) ? “0” + minute : minute;
    var second = time.getSeconds();
    var sSecs = (second<10) ? “0” + second : second;
    var strTime = hour + “:” + sMin + “:” + sSecs;
    document.getElementById(“clockFace”).innerHTML = strTime;
}

create_table.sql

/**************** CREATE TABLE statements *******/
CREATE DATABASE SHOPPING;
USE SHOPPING;

/**************** CREATE TABLE statements *******/
CREATE TABLE CUSTOMER (  
    CustomerID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    Fullname VARCHAR(50) NOT NULL,  
    Email VARCHAR(50) NOT NULL,  
    Password CHAR(41) NOT NULL,  
    Address TEXT );

CREATE TABLE PRODUCT (  
    ProductID INT NOT NULL PRIMARY KEY,  
    Name VARCHAR(50) NOT NULL,  
    Description TEXT,  
    Price DEC(6,2) NOT NULL );

CREATE TABLE ORDERS (  
    OrderID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    CustomerID INT NOT NULL,  
    ...
Date DATE NOT NULL,
Total DEC(8,2) NOT NULL );

CREATE TABLE PRODUCT_ORDERS ( 
OrderID INT NOT NULL,
ProductID INT NOT NULL,
Quantity INT NOT NULL,
PRIMARY KEY (OrderID, ProductID));

CREATE TABLE SHOPPING_CART ( 
CartID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
SessionID CHAR(50),
ProductID INT NOT NULL,
Quantity INT NOT NULL );

/******** ALTER TABLE statement *******/
ALTER TABLE CUSTOMER AUTO_INCREMENT = 100;
ALTER TABLE ORDERS AUTO_INCREMENT = 1000;

insert_product.sql

/******** INSERT statements *******/
INSERT INTO PRODUCT VALUES
(22103, 'Tribute Blend', 'Sun-dried beans from Ethiopia, inspired by our collaboration with Ethiopian farmers.', 13.95),
(97902, 'Pike Place Roast', 'The smoother finish and soft acidity are attributed to a proprietary blend of high-altitude arabica beans.', 9.95),
(12948, 'French Roast', 'Blunt, smoky flavours are the objective in creating this, our darkest roasted coffee.', 10.95),
(12983, 'Sumatra', 'The concentrated spicy, herbal notes and earthy aroma are the telltale signatures of this well-loved coffee.', 10.95);

INSERT INTO CUSTOMER VALUES
(NULL, 'John Doe', 'johndoe@tcc242.com', PASSWORD('wawasan'), '1 Seattle, Washington');
References


Wikibooks (2010) *PHP and MySQL programming/form handling*, http://en.wikibooks.org/wiki/PHP_and_MySQL_Programming/Form_Handling, used under a Creative Commons Attribution-ShareAlike licence: http://creativecommons.org/licenses/by-sa/3.0/


### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ampersand</td>
<td>This is the “&amp;” symbol. In a query string, the field-value pairs are separated by an ampersand.</td>
</tr>
<tr>
<td>Cookie</td>
<td>A cookie is a small text file that contains a unique tracking number that enables sites to remember users for all subsequent visits. When a user visits your website, a cookie can be created and saved on the user’s computer to store information about the user, such as his or her preferences. When the user executes a request for another web page on the same website, the server retrieves the user’s information from the cookie and uses it, for example, to eliminate the need for another login or to personalise the page.</td>
</tr>
<tr>
<td>Field-value pairs</td>
<td>A query string is usually made up of a series of field-value pairs.</td>
</tr>
<tr>
<td>GET</td>
<td>With the HTTP “GET” method, form data is appended to the URL specified by the action attribute (with a question-mark “?” as the separator) and this new URL is sent to the processing server.</td>
</tr>
<tr>
<td>Hidden field</td>
<td>A hidden field in a HTML form is similar to a text field except that it does not appear on the web page. It is used to transfer data or variables from the form page to the handler page. A hidden field is represented by an INPUT element with <code>type = hidden</code>. You can place the hidden field anywhere as long as it is within the form tag and follows HTML markup rules. If you created hidden fields in the form, these fields will be included in the query string as well when the form is submitted with the “GET” method.</td>
</tr>
<tr>
<td>POST</td>
<td>With the HTTP “POST” method, the form data set is included in the body of the form and sent to the processing server.</td>
</tr>
</tbody>
</table>
Query strings

A query string is a part of the Uniform Resource Locator (URL) that contains data to be passed to another web page. A typical URL containing a query string is as follows:

http://web/page.php?{query_string}

Session

A session is generally a temporary set of variables that exist only until you shut down your web browser. The information is stored temporarily for the PHP program to refer to throughout the browsing session.

Session ID (SID)

A session variable that identifies all session data related to a user’s browsing session. When a session is created, it is assigned a SID, which the browser retains to identify the session.

State information

In the context of web programming, state information is information relating to the current conditions of a web page. For example, the password that was used to login to the web page and the current user’s preferences are part of the state information.

URL rewriting

A method that appends the Session ID (SID) to every local URL found within a requested page. It results in automatic SID propagation whenever the user clicks on one of those local links. Therefore, it helps to maintain the SID even if the user disables cookies.