

Open Source Technologies

DCAP203/DCAP410



L OVELY
P ROFESSIONAL
U NIVERSITY



OPEN SOURCE TECHNOLOGIES

Copyright © 2013, Sumita Verma
All rights reserved

Produced & Printed by
EXCEL BOOKS PRIVATE LIMITED
A-45, Naraina, Phase-I,
New Delhi-110028
for
Lovely Professional University
Phagwara

CONTENTS

Unit 1:	MySQL	1
Unit 2:	Working with MySQL	18
Unit 3:	Apache Server	28
Unit 4:	Apache Server Installation in Window	42
Unit 5:	PHP	64
Unit 6:	Building Blocks of PHP	81
Unit 7:	Functions	112
Unit 8:	Working with Strings, Date and Time	131
Unit 9:	Working with Forms	149
Unit 10:	Cookies	164
Unit 11:	Directories and Files	173
Unit 12:	Images	189
Unit 13:	Stored Procedure	204
Unit 14:	Connecting to MySQL with PHP	222

SYLLABUS

Open Source Technologies

Objectives: To impart understanding of essentials of open source technologies. Open source technologies course is designed to enable web developers and others with limited programming experience to build dynamic database driven e-commerce websites using the PHP programming language.

S.No.	Description
1.	MySQL: Current and Future Versions of MySQL, Installing MySQL. Basic Security Guidelines. Privilege System and Working with user Privileges.
2.	Apache Server: Versions of Apache. Choosing Appropriate Installation Method. Installing on Windows. Apache Configuration File Structure. Apache Log File. Starting Apache for First Time.
3.	PHP: Versions of PHP. Installation of PHP. Php.ini basics. Testing Installation. Building Blocks of PHP: Variables, Data Types, Operators & Expressions, Constants. Switching Flow. Loops, Code Blocks and Browser Output.
4.	Functions: Meaning, Calling, Defining a Function. Return Value from User-defined Function. Saving State with 'Static' Function. Testing for Existence of Function. Arrays: What are Arrays, Creating Arrays, Array Related Functions. Objects: Creating an Object. Object Inheritance.
5.	Working with String, Dates & Time: Formatting String with PHP. Using Date and Time Functions with PHP. Other String, Date/Time Functions.
6.	Forms: Creating Simple Input Form. Accessing Form Input with User Defined Arrays, HTML and PHP Code on a Single Page. Using Hidden Fields to Save State. Redirecting User. Working with File Upload.
7.	Cookies: Introducing Cookies, Setting Cookies, Deleting Cookies with PHP, Session Function Overview, Starting Session, Working with Session Variables. Destroying Sessions and Unsetting Variables.
8.	Files and Directories: Include Files with include(). Validating Files. Creating Files, Deleting Files, Opening a File for Reading, Writing, Appending.
9.	Images: Understanding Image Creation Process, Necessary Modifications to PHP, Drawing a New Image, Modifying Existing Images, Image Creation from User Input.
10.	Stored Procedures: What are Transactions, What are Stored Procedures. Connecting to MySQL with PHP: Working with MySQL Data.

Unit 1: MySQL

Notes

CONTENTS

Objectives

Introduction

1.1 Current and Future Versions of MySQL

1.1.1 How to Get MySQL

1.2 Installing MySQL

1.2.1 Installing MySQL on Linux/Unix

1.2.2 Installing MySQL on Mac OS X

1.2.3 Installing MySQL on Windows

1.3 Basic Security Guidelines

1.3.1 Starting MySQL

1.3.2 Securing Your MySQL Connection

1.4 Summary

1.5 Keywords

1.6 Review Questions

1.7 Further Readings

Objectives

After studying this unit, you will be able to:

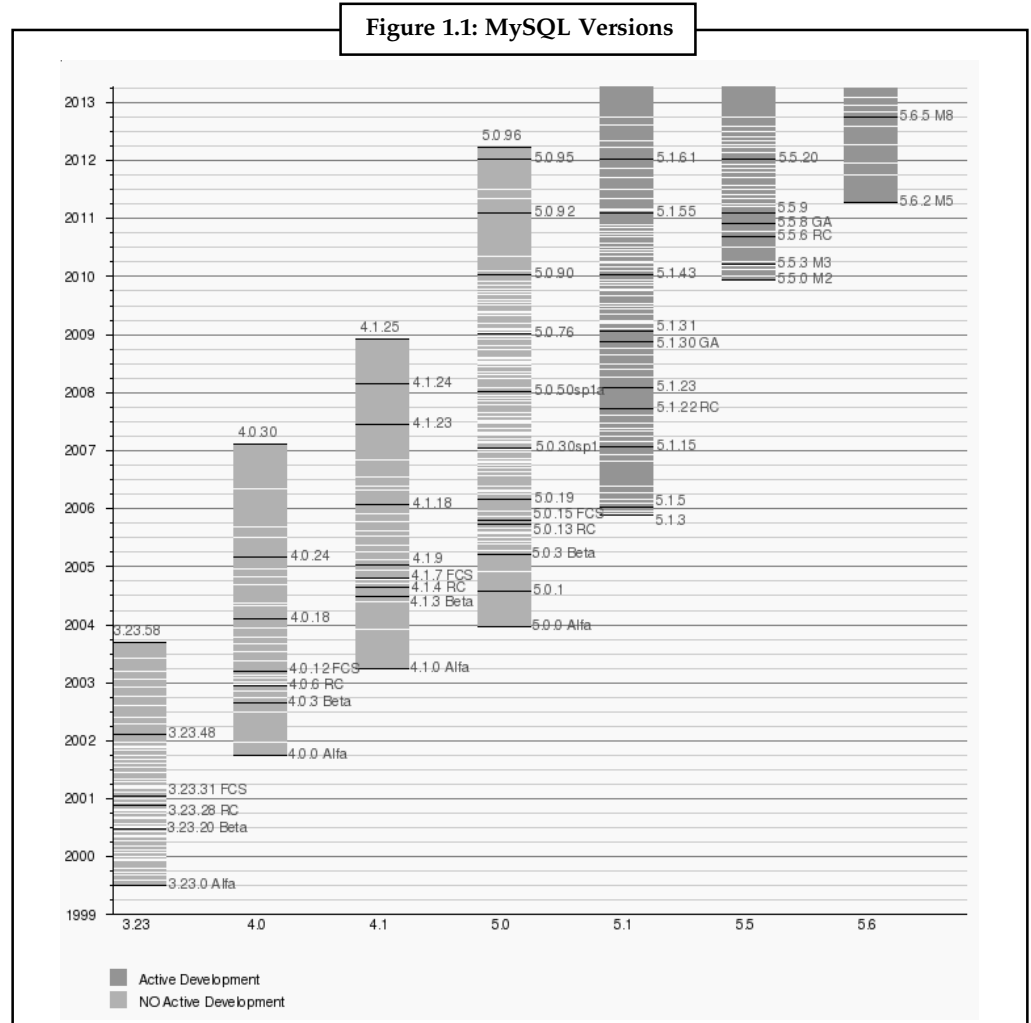
- Explain the Versions of MySQL
- Discuss the Installation Process of MySQL
- Understand the Basic Security Guidelines in MySQL

Introduction

MySQL is an open source Relational Database Management System (RDBMS). It provides multi-user access to a number of databases. The SQL phrase stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, MyBB, Drupal and other software.

1.1 Current and Future Versions of MySQL

The version that we will discuss is MySQL 4.0.21. This version number is read as minor release number 21 of the major version 4.0 software. MySQL AB was a software company that was founded in 1995 but acquired by Sun Microsystems in 2008 which was then, in 2010, acquired by Oracle Corporation. MySQL AB is the creator of MySQL which releases different versions of MySQL after bug changes and security enhancements.



Source: <http://en.wikipedia.org/wiki/MySQL>

1.1.1 How to Get MySQL

To get access to MySQL visit the website <http://www.MySQL.com/>. It contains various editions of MySQL to download including MySQL Enterprise Edition, MySQL Cluster CGE and MySQL Community Edition along with for all platforms, installer packages for Mac OS X, and RPMs and source code files for Linux/Unix platforms.



Did u know? MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, <http://en.wikipedia.org/wiki/MySQL> - cite_note-9 Google, Facebook, Twitter, Flickr, Nokia.com and YouTube.

Self Assessment

Notes

State whether the following statements are true or false:

1. MySQL is a proprietary Relational Database Management System (RDBMS).
2. MySQL provides single-user access to a number of databases.
3. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.
4. LAMP is an acronym for Linux, Application, MySQL, Perl/PHP/Python.
5. Drupal is an application using MySQL.

1.2 Installing MySQL

Now we will discuss the steps to be followed in order to install MySQL on different operating systems.

1.2.1 Installing MySQL on Linux/Unix

You use RPMs or binaries to install MySQL on your system. For both the methods two basic files that are needed are: MySQL-server and MySQL-client packages. If you get a dependency failure when trying to install MySQL packages (Example: error: removing these packages would break dependencies: libmysqlclient.so.10 is needed by ...), you should also install the MySQL-shared-compat package, which includes both the shared libraries for backward compatibility.



Notes RPM distributions of MySQL often are provided by other vendors. They may differ in features and capabilities from those built by MySQL AB.

To see all files in an RPM package (Example: a MySQL-server RPM), run a command like this:

```
shell> rpm -qp1 MySQL-server-VERSION.i386.rpm
```

To perform a standard minimal installation, install the server and client RPMs:

```
shell> rpm -i MySQL-server-VERSION.i386.rpm
```

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

To install only the client programs, install just the client RPM:

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

Oracle provides a set of binary distributions of MySQL. These include binary distributions in the form of compressed **tar** files (files with a **.tar.gz** extension) for a number of platforms, as well as binaries in platform-specific package formats for selected platforms. MySQL compressed **tar** file binary distributions have names of the form **mysql-VERSION-OS.tar.gz**, where **VERSION** is a number, and **OS** indicates the type of operating system for which the distribution is intended. To install MySQL from a compressed **tar** file binary distribution, your system must have GNU gunzip to uncompress the distribution and a reasonable **tar** to unpack it. If your **tar** program supports the **z** option, it can both uncompress and unpack the file. GNU **tar** is known to work. The standard **tar** provided with some operating systems is not able to unpack the long file

Notes

names in the MySQL distribution. You should download and install GNU **tar**, or if available, use a pre-installed version of GNU tar. Usually this is available as **gnutar**, **gtar**, or as **tar** within a GNU or Free Software directory, such as `/usr/sfw/bin` or `/usr/local/bin`. GNU **tar** is available from <http://www.gnu.org/software/tar/>.



Caution If you have previously installed MySQL using your operating system native package management system, such as `yum` or `apt-get`, you may experience problems installing using a native binary. Make sure your previous MySQL previous installation has been removed entirely (using your package management system), and that any additional files, such as old versions of your data files, have also been removed. You should also check the existence of configuration files such as `/etc/my.cnf` or the `/etc/mysql` directory have been deleted.

If your system does not already have a user and group for **mysqld** to run as, you may need to create one. The following commands add the `mysql` group and the `mysql` user. You might want to call the user and group something else instead of `mysql`. If so, substitute the appropriate name in the following instructions. The syntax for **useradd** and **groupadd** may differ slightly on different versions of Unix, or they may have different names such as **adduser** and **addgroup**.

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
```

Because the user is required only for ownership purposes, not login purposes, the **useradd** command uses the `-r` option to create a user that does not have login permissions to your server host. Omit this option to permit logins for the user (or if your **useradd** does not support the option).

To obtain and unpack the distribution, pick the directory under which you want to unpack the distribution and change location into it. The example here unpacks the distribution under `/usr/local`. The instructions, therefore, assume that you have permission to create files and directories in `/usr/local`. If that directory is protected, you must perform the installation as root.

```
shell> cd /usr/local
```

Unpack the distribution, which creates the installation directory. Then create a symbolic link to that directory. **tar** can uncompress and unpack the distribution if it has `z` option support:

```
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
```

The **tar** command creates a directory named `mysql-VERSION-OS`. The `ln` command makes a symbolic link to that directory. This enables you to refer more easily to the installation directory as `/usr/local/mysql`.

If your **tar** does not have `z` option support, use **gunzip** to unpack the distribution and **tar** to unpack it. Replace the preceding **tar** command with the following alternative command to uncompress and extract the distribution:

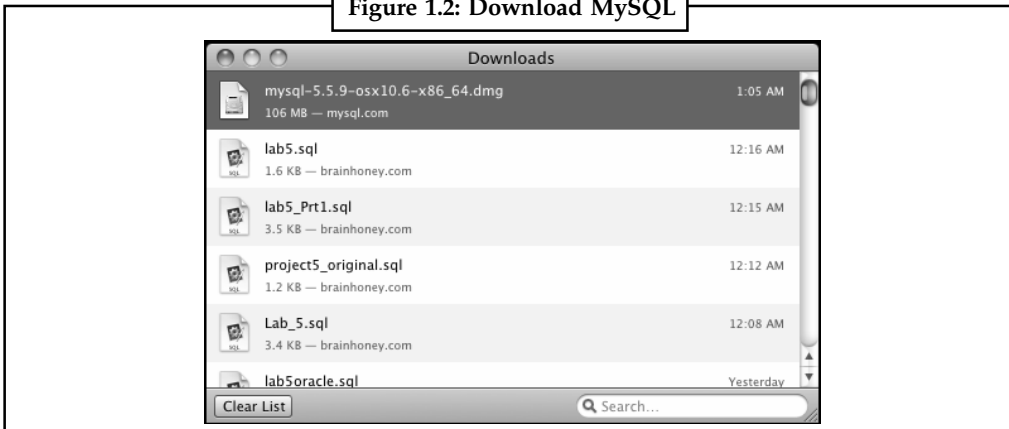
```
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
```

1.2.2 Installing MySQL on Mac OS X

To install MySQL on your Mac machine follow the steps below:

Step 1: You can download MySQL for the Mac OS X. After the download completes, open the file folder in the download directory.

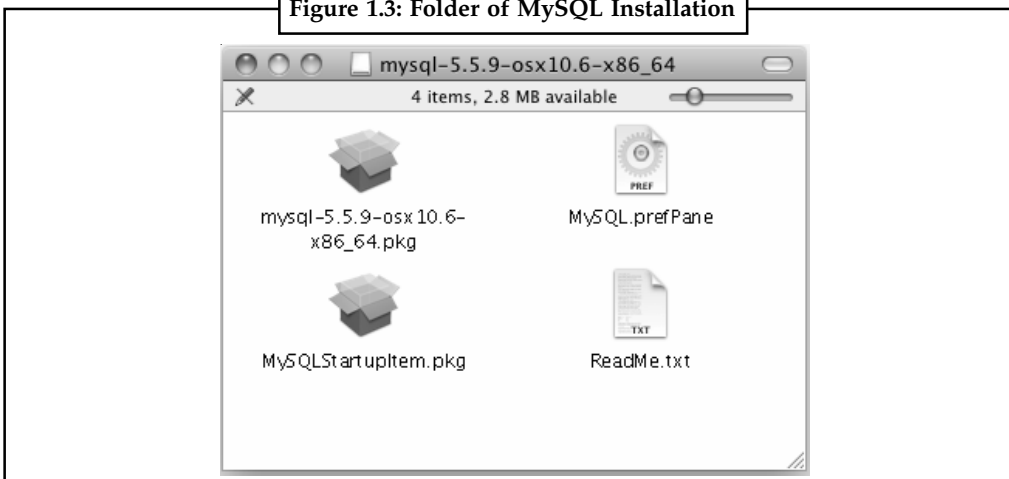
Figure 1.2: Download MySQL



Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Step 2: The open file folder and it will look like the following. Launch the mysql-5.5.9-osx 10.6-x86_64.pkg file, which installs the product.

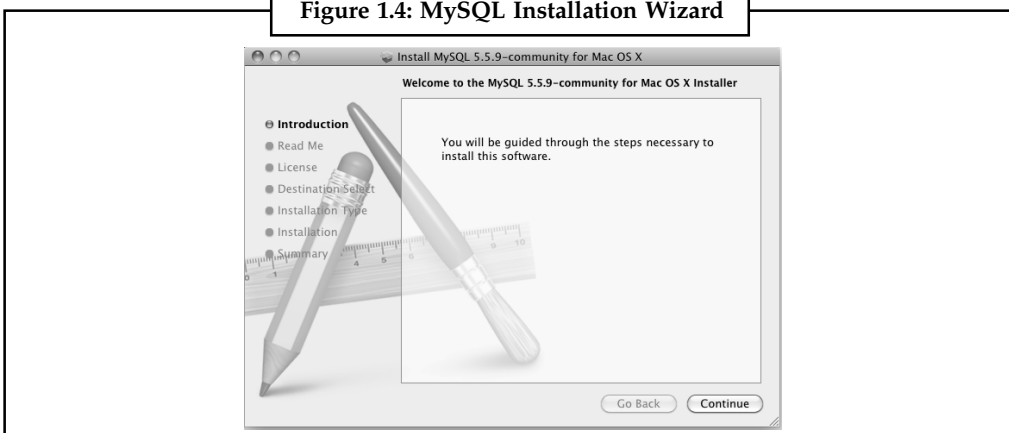
Figure 1.3: Folder of MySQL Installation



Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Step 3: After launching the executable, you are now on the first page of the Install MySQL 5.5.9 installation application. Click the *Continue* button.

Figure 1.4: MySQL Installation Wizard



Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Notes

Step 4: This page contains the instructions, you can pause to read them or continue with these instructions. Click the *Continue* button to proceed.

Figure 1.5: MySQL Installation Wizard



Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Step 5: This page contains the General Public License (GPL). You agree or stop the installation. Click the *Continue* button to proceed.

Figure 1.6: MySQL Installation Wizard



Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Step 6: The following overlay dialog contains your agreement. Click the *Agree* button to proceed.

Figure 1.7: MySQL Installation Wizard



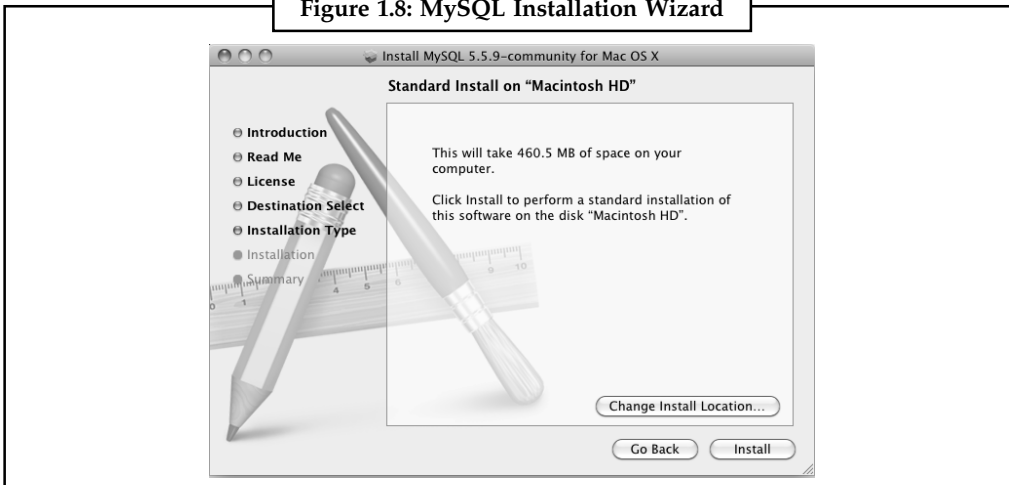
Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Step 7: There are fewer options in this installation than the Windows installation. While you can change the installation location, the software installs by default in the `/usr/local/mysql` directory. The installation requires that you have a `mysql` user account on the operating system, and you

don't need to do anything because one exists as part of the default Mac OS X installation. Click the *Install* button to proceed.

Notes

Figure 1.8: MySQL Installation Wizard



Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Step 8: This dialog requires the system administrator's password. Enter the valid password and click the *OK* button to proceed.

Figure 1.9: MySQL Installation Wizard



Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Step 9: Depending on the system, this could take more than a minute. All you can do is wait.

Figure 1.10: MySQL Installation Wizard

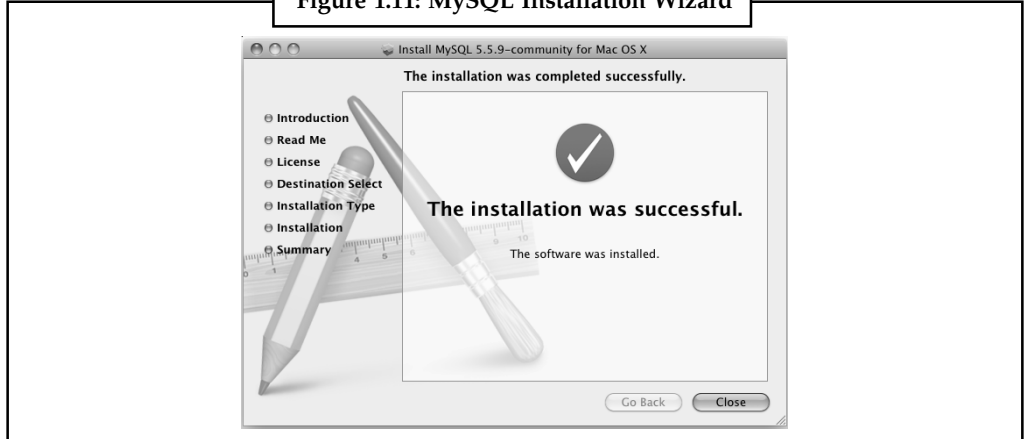


Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

Notes

Step 10: This page tells you that you've completed the installation. Click the *Close* button to proceed.

Figure 1.11: MySQL Installation Wizard




Source: <http://blog.mclaughlinsoftware.com/2011/02/10/mac-os-x-mysql-install/>

1.2.3 Installing MySQL on Windows

To install MySQL on Windows, follow the steps below:

Step 1: Log into the computer as an administrator. This will give you administrator rights, which will make the installation smoother.

 *Notes* Once installed, the program doesn't need to be run as an administrator.

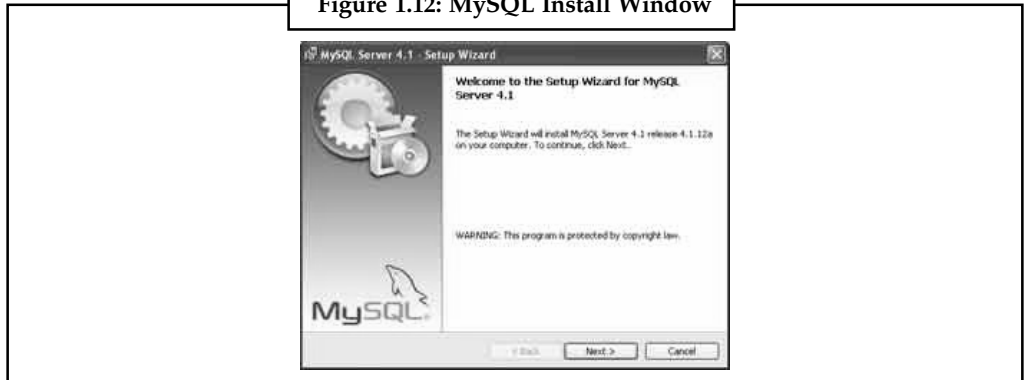
Step 2: Download the free MySQL Server Community Edition. Be sure to download a version that includes a Windows Installer. Save the file on your Windows Desktop. If you're not sure which version to select, download MySQL Installer for Windows.

Step 3: Double-click the downloaded file. The installation file comes as a .zip file, so double-clicking it should initiate your unzipping software and take you to an archive folder.

Step 4: Double-click Setup.exe. (It should be the only file inside your archive.) This will initialize the setup process.

Step 5: Click Next. This begins the setup.

Figure 1.12: MySQL Install Window



Source: <http://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC>

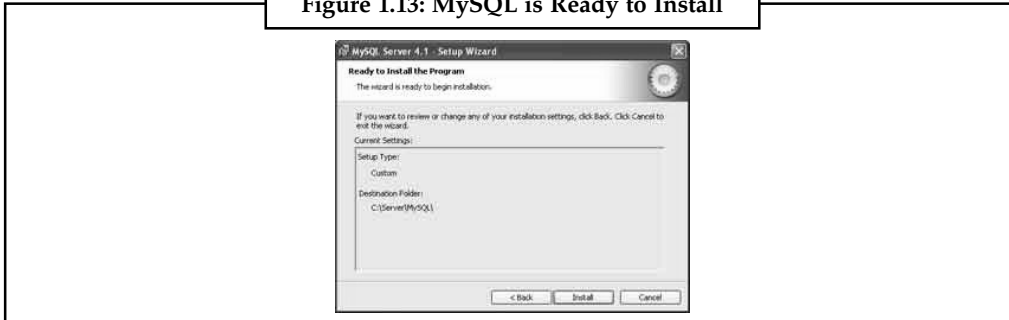
Notes

Step 6: Click Custom > Next. This will enable you to specify where you want to install MySQL. If you have Apache installed in C:\Server, you will want to install MySQL in the same directory.

- In the next window, highlight MySQL Server, and then click the Change.
- In the next window, in the text box Folder Name, change the directory to C:\Server\MySQL\ exactly as it's written here and then click OK.
- On the next window, click Next. Now MySQL is ready to install.

Step 7: Click Install. Wait while the program self-installs.

Figure 1.13: MySQL is Ready to Install



Source: <http://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC>

Step 8: Click Skip Sign-Up and then Next. Once the installation is complete, you will be presented with a MySQL Sign-Up window. Skip signing up with MySQL for now since you can sign up later if you like. Once you've skipped, your new dialogue box should say Wizard Completed.

Step 9: Configure MySQL. Leave the check box Configure the MySQL Server Now checked and click Finish.

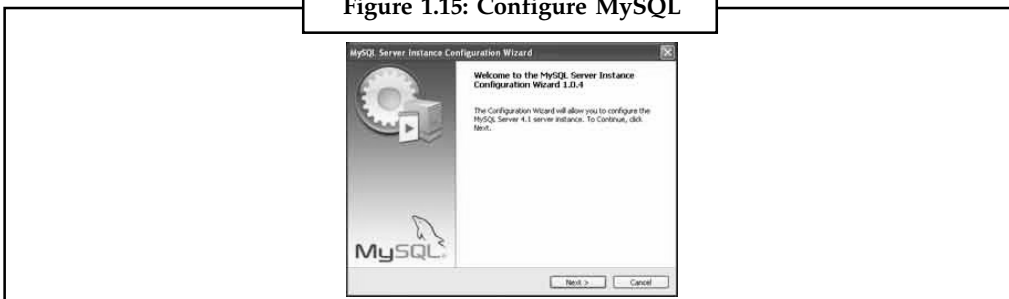
Figure 1.14: MySQL Success



Source: <http://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC>

Step 10: Click Next. This will initialize the configuration setup.

Figure 1.15: Configure MySQL



Source: <http://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC>

Notes

Step 11: Check Standard Configuration and then click Next. This is the default configuration and is recommended for most users.

Step 12: Make sure Install As Windows Service and Launch the MySQL Server Automatically are checked, then click Next.

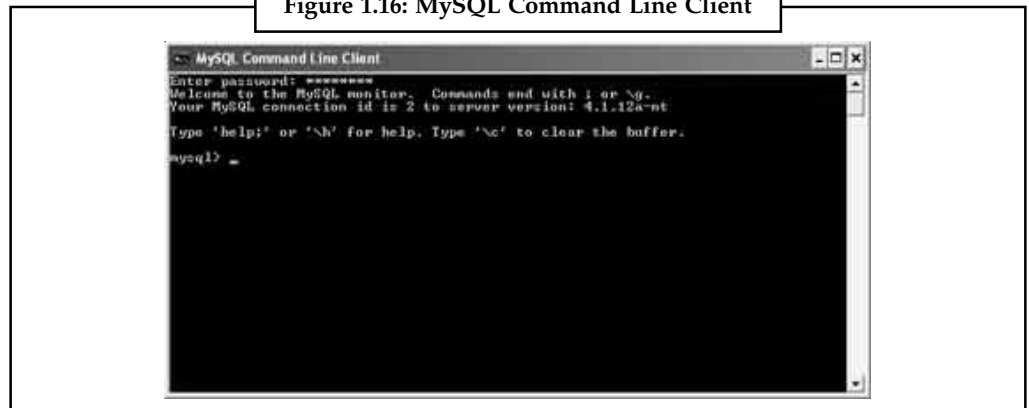
Step 13: Create a root password. Type in what you want your root password to be and make sure Enable root access from remote machines is checked. Make sure you choose a difficult to guess password and write it down so you don't forget it. Click Next.

Step 14: Click Execute. This will start the MySQL server. After MySQL has done its thing, click Finish.

Step 15: From the Windows task bar, go to Start > All Programs > MySQL > MySQL Server 4.x > MySQL Command line client. This will open a command window asking you for a password.

Step 16: Enter your root password and hit Enter. This should initiate the program.

Figure 1.16: MySQL Command Line Client



Source: <http://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC>

Self Assessment

Fill in the blanks:

6. You use or to install MySQL on your system.
7. For both the methods two basic files that are needed are: and packages.
8. If you get a dependency failure when trying to install MySQL packages, you should also install the package.
9. The binary distributions of MySQL are compressed files.
10. The command is used to add a group through command line.

1.3 Basic Security Guidelines

While accessing MySQL via your Internet service provider, the non-root user should be prevented from modifying certain aspects of server security .It is necessary to consider fully protecting the entire server host (not just the MySQL server) against all types of applicable attacks: eavesdropping, altering, playback, and denial of service. MySQL uses security based on Access Control Lists (ACLs) for all connections, queries, and other operations that users can attempt to perform. There is also support for SSL-encrypted connections between MySQL clients and servers.

1.3.1 Starting MySQL

Notes

When running MySQL, follow these guidelines:

1. Do not ever give anyone (except MySQL root accounts) access to the user table in the MySQL database.
2. Learn how the MySQL access privilege system works . Use the GRANT and REVOKE statements to control access to MySQL. Do not grant more privileges than necessary. Never grant privileges to all hosts.
 - ❖ Try `mysql -u root`. If you are able to connect successfully to the server without being asked for a password, anyone can connect to your MySQL server as the MySQL root user with full privileges.
 - ❖ Use the SHOW GRANTS statement to check which accounts have access to what. Then use the REVOKE statement to remove those privileges that are not necessary.
3. Do not store clear text passwords in your database. If your computer becomes compromised, the intruder can take the full list of passwords and use them. Instead, use SHA1(), MD5(), or some other one-way hashing function and store the hash value.
4. **Do not choose passwords from dictionaries.** Special programs exist to break passwords. Even passwords like “xfish98” are very bad. Much better is “duag98” which contains the same word “fish” but typed one key to the left on a standard QWERTY keyboard. Another method is to use a password that is taken from the first characters of each word in a sentence.



Example: “Four score and seven years ago” results in a password of “Fsasya”. The password is easy to remember and type, but difficult to guess for someone who does not know the sentence. In this case, you can additionally substitute digits for the number words to obtain the phrase “4 score and 7 years ago”, yielding the password “4sa7ya” which is even more difficult to guess.

5. **Invest in a firewall.** This protects you from at least 50% of all types of exploits in any software. Put MySQL behind the firewall or in a demilitarized zone (DMZ).
 - ❖ Try to scan your ports from the Internet using a tool such as nmap. MySQL uses port 3306 by default. This port should not be accessible from untrusted hosts. As a simple way to check whether your MySQL port is open, try the following command from some remote machine, where *server_host* is the host name or IP address of the host on which your MySQL server runs

```
shell> telnet server_host 3306
```

If **telnet** hangs or the connection is refused, the port is blocked, which is how you want it to be. If you get a connection and some garbage characters, the port is open, and should be closed on your firewall or router, unless you really have a good reason to keep it open.

6. Applications that access MySQL should not trust any data entered by users, and should be written using proper defensive programming techniques.
7. Do not transmit plain (unencrypted) data over the Internet. This information is accessible to everyone who has the time and ability to intercept it and use it for their own purposes. Instead, use an encrypted protocol such as SSL or SSH. MySQL supports internal SSL connections. Another technique is to use SSH port-forwarding to create an encrypted (and compressed) tunnel for the communication.

Notes

8. Learn to use the **tcpdump** and **strings** utilities. In most cases, you can check whether MySQL data streams are unencrypted by issuing a command like the following:

```
shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings
```

This works under Linux and should work with small modifications under other systems.



Caution If you do not see cleartext data, this does not always mean that the information actually is encrypted. If you need high security, consult with a security expert.

1.3.2 Securing Your MySQL Connection

The problem of security arises in case we have a system being shared amongst various users. Leaving your workstation unattended can lead to anyone deleting important data, inserting incorrect data, or shutting down the server. To prevent this use a screen saver or lock screen mechanism with a password. To prevent data interception over the internet, the best way is to encrypt it. To access a remote machine use SSH which provides all transmissions to and from the remote machine to be encrypted. You can also use a Web-based administration interface, such as phpMyAdmin over a secure HTTP connection.

Self Assessment

Fill in the blanks:

11. MySQL uses security based on
12. Use the statement to check which accounts have access to what.
13. Use the statement to remove the privileges that are not necessary.
14. MySQL uses port by default.
15. To check whether your MySQL port is open, use command.



Case Study

Big Fish Selects MySQL Cluster to Serve Real-time Web Recommendations

Introduction

The world's largest producer of casual games has selected MySQL Cluster to power its real-time recommendations platform. High velocity data ingestion, low latency reads, online scaling and the operational simplicity delivered by MySQL Cluster enables Big Fish to increase customer engagement and deliver targeted marketing, providing a more personalized experience to its users.

Company Overview

Founded in 2002, Big Fish is the world's largest producer of casual games; dedicated to bringing engaging entertainment to everyone, anywhere, on any device.

Through its data-driven platform, millions of consumers seeking engaging entertainment easily discover and play PC and mobile games created by Big Fish's network of more than 600 development partners and its in-house Big Fish Studios.

Contd...

The company has distributed more than 2 billion games from a growing catalog of 3,000+ unique PC/Mac titles and 300+ unique mobile games, and offers cross-platform interactive game streaming via its cloud gaming service, Big Fish Unlimited.

Big Fish's games are played in more than 150 countries across 13 local language sites. The company is headquartered in Seattle, WA, with multiple regional offices in North America and Europe.

Personalizing the User Experience

The global video gaming market is experiencing explosive growth with forecasts estimating revenues to reach \$82bn by 2017. Competition is intense, and so to differentiate services and engage users, progressive gaming companies such as Big Fish are seeking solutions to more fully personalize the customer experience.

Using Business Intelligence (BI) and predictive analytics Big Fish can segment customers based on a range of demographic and behavioural indicators. This enables Big Fish to serve highly targeted recommendations and marketing, precisely personalized to a user's individual preferences.

Project Requirements

The Marketing Management Service (MMS) is designed to serve real-time recommendations to Big Fish customers in real time. The project team established a set of success requirements for the database that would power the MMS recommendations engine:

- **High-Velocity Data Ingest:** Data had to be extracted daily from the BI system to the recommendations database, with the ability to scale the frequency and volume of data loads in the future. Each user could be categorized into multiple market segmentations, and these segmentations can also change daily;
- **Low-Latency Access:** Personalized content had to be rendered to each page view, without impacting user response times;
- **On-Line Scalability:** With data volumes and velocity increasing, coupled with the desire to use MMS across new services in the future, it was vital that the database could scale quickly to meet new and unpredictable demands. At the same time, database scaling had to be an online operation, ensuring continuous platform availability to gamers;
- **99.999% Availability:** The service had to be reliable and highly available as downtime would significantly impact the user experience;
- **Enterprise-Level Support & Strong Technology Roadmap:** MMS is a strategic investment for Big Fish, and so the project team wanted access to the highest levels of 24x7 support. This requirement was coupled with the security of working with a proven technology that the developers knew had a strong roadmap.

Technology Selection

Big Fish already has an extensive deployment of MySQL databases powering web applications, including the storefront. They knew MySQL could power the recommendations database, but would require additional engineering efforts to implement database sharding to support data ingest and future scaling needs, coupled with a Memcached layer for low-latency reads.

As a result, they began evaluations of MySQL Cluster, in addition to other database technologies. Using MySQL Cluster, the Engineering teams were able to leverage their existing MySQL skills, enabling them to reduce operational complexity when compared to introducing a new database to the Big Fish environment.

Contd...

Notes

At the same time, they knew MySQL Cluster, backed by Oracle, provided the long-term investment protection they needed for the MMS platform.

Through their evaluation, the Big Fish engineering team identified MySQL Cluster was best able to meet their technical requirements.

Write Performance to Support High Velocity Data Ingest

MySQL Cluster's multi-master architecture, coupled with the ability to auto-shard (partition) tables across nodes, enables Big Fish to meet the data load demands of MMS.

During testing, MySQL Cluster was able to insert 187,000 records per second while simultaneously serving 3,000 SELECTs per second with just two data nodes and one MySQL Server.

They found MySQL Cluster scaled linearly as they tested the loading of 25m to 1bn records.

Low Latency Access

Using MySQL Cluster's in-memory tables, Big Fish are able to serve up personalized recommendations to their gamers with sub-millisecond responsiveness.

On-Line Scalability

Adding nodes to a running cluster, without downtime, will enable Big Fish to scale the recommendations database to support the higher capacity and performance demanded by the MMS roadmap. MySQL Cluster's use of horizontal scalability across low-cost commodity nodes also reduces TCO of the project.

Continuous Availability

The shared-nothing, distributed design of MySQL Cluster coupled with integrated replication and self-healing recovery ensures high availability of the recommendations platform, all without DevOps intervention.

SQL and NoSQL APIs

Big Fish can select the best access method to meet their specific requirements. Data is inserted into the cluster using the NoSQL Java ClusterJ API, and accessed by the MMS recommendations engine using the full power of SQL.

Project Implementation

- User data is replicated from the MySQL databases powering the gaming storefront to the Big Fish BI platform;
- User data is analyzed and segmented within the BI platform;
- Recommendations are loaded as user records into MySQL Cluster using the NoSQL Cluster_J (Java) API;
- The SQL API from the MySQL Servers then delivers personalized content to gamers in real-time, initially serving over 15m sessions per day.

"Gaming is a highly dynamic environment. It can be very difficult to predict performance demands. MySQL Cluster's online scalability provides the assurances we need to meet future requirements and we can use MySQL Cluster Manager to automate scaling and cluster administration - which makes DevOps very happy!" Isaac Hawley, Software Developer, Big Fish

MySQL Cluster is distributed across commodity x86 hardware running the Linux operating system, in the following configuration:

- 3 x Application (Access) nodes running the native Java NoSQL ClusterJ API and MySQL Server.

Contd...

- 2 x Data Nodes with a hot-spare, automatically provisioned by MySQL Cluster Manager, providing a self-healing backup in the event of a node outage.

Big Fish runs MySQL Cluster on Virtual Machines in their Development & Testing environment, and uses physical hardware in production. MySQL Cluster's flexibility enables them to move seamlessly between the virtual and physical environments.

Big Fish has subscribed to MySQL Cluster CGE providing the Engineering team with access to 24x7 Oracle Premier Support and MySQL Cluster Manager, which reduces operational overhead by providing:

- Automated configuration and reconfiguration of MySQL Cluster;
- Automated online node addition for on-demand scaling.

The Future with Oracle's MySQL Cluster

It is expected that the initial deployment of MMS will be expanded to include all new web site developments, and to additional channels and gaming platforms. In addition, data volumes and loading frequencies are expected to increase, with the possibility of streaming real-time updates from the BI system to the recommendations platform. The online scalability of MySQL Cluster will enable Big Fish to meet these new requirements.

Big Fish also has the flexibility to support future projects that include complex queries on live, web-based data. Using MySQL Cluster's Adaptive Query Localization, JOIN operations are pushed down to the data nodes, where they execute on local copies of the data before aggregating results to the application - enabling real-time analytics.

In summary, MySQL Cluster has the performance, availability and operational ease-of-use to support Big Fish' requirements today, while also providing a future platform for scale and new application development in the future.

Questions

1. Study and analyze the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: <http://www.mysql.com/why-mysql/case-studies/mysql-cs-bigfish.html>

1.4 Summary

- MySQL is an open source Relational Database Management System (RDBMS).
- It provides multi-user access to a number of databases.
- MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack.
- To get access to MySQL visit the website <http://www.MySQL.com/>.
- You use RPMs or binaries to install MySQL on your system.
- For both the methods two basic files that are needed are: MySQL-server and MySQL-client packages.
- Oracle provides a set of binary distributions of MySQL. These include binary distributions in the form of compressed tar files.
- To install MySQL from a compressed tar file binary distribution, your system must have GNU gunzip to uncompress the distribution and a reasonable tar to unpack it.

Notes

- While accessing MySQL via your Internet service provider, the non-root user should be prevented from modifying certain aspects of server security.
- MySQL uses security based on Access Control Lists (ACLs) for all connections, queries, and other operations that users can attempt to perform.

1.5 Keywords

ACL: Access Control Lists used by MySQL to handle security of the system.

GNU: General Public License.

GRANT: Statement used to grant the privileges that are necessary for the users.

LAMP: Linux, Apache, MySQL, Perl/PHP/Python open source web application software stack.

RDBMS: Relational Database Management System.

REVOKE: Statement used to remove the privileges that are not necessary from the users.

SQL: Structured Query Language.

TAR: Extension of zipped binaries used for MySQL installation.

1.6 Review Questions

1. What is MySQL?
2. How do you obtain MySQL?
3. Give the steps to install MySQL on Linux/Unix using binaries.
4. Explain the process of installing MySQL on Linux/Unix using RPM.
5. How do you install MySQL on Mac OS X?
6. Explain the steps to install MySQL on Windows.
7. How does MySQL guarantee security of the system?
8. Explain the steps to be followed to secure your system.
9. What are GRANT and REVOKE commands?
10. How do we use the RPM command?

Answers: Self Assessment

- | | |
|-------------------------------|------------------------|
| 1. False | 2. False |
| 3. True | 4. False |
| 5. True | 6. RPM, Binaries |
| 7. MySQL Server, MySQL Client | 8. MySQL Shared Compat |
| 9. TAR | 10. groupadd |
| 11. ACL | 12. SHOW GRANT |
| 13. REVOKE | 14. 3306 |
| 15. telnet | |

1.7 Further Readings

Notes



Books

Anthony Molinaro, SQL Cookbook.

O reilly, Head First SQL.

Russell Dyer, MySQL in a Nutshell.

Sheldon and Moes, Beginning MySQL.



Online links

dev.mysql.com/doc/refman/5.5/en/mysql-installer-gui.html

<http://www.cs.duke.edu/csl/docs/mysql-refman/installing.html#linux-rpm>

<http://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC>

http://www.yaldex.com/php_tutorial_3/ch02lev1sec7.html

Unit 2: Working with MySQL

CONTENTS

Objectives

Introduction

2.1 Privilege System in SQL

2.2 Working with User Privileges

2.2.1 Adding Users through MySQL

2.2.2 Removing Privileges of MySQL

2.3 Summary

2.4 Keywords

2.5 Review Questions

2.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain SQL Privilege
- Discuss Working with User Privileges

Introduction

MySQL provides privileges that apply in different contexts and at different levels of operation:

- (a) Administrative privileges enable users to manage operation of the MySQL server. These privileges are global because they are not specific to a particular database.
- (b) Privileges for database objects such as tables, indexes, views, and stored routines can be granted for specific objects within a database, for all objects of a given type within a database (for example, all tables in a database), or globally for all objects of a given type in all databases).
- (c) Database privileges apply to a database and to all objects within it. These privileges can be granted for specific databases, or globally so that they apply to all databases.

We will now discuss Database privileges that are supported by MySQL.

- **CREATE:** Enables creation of new databases and tables.
- **DROP:** Enables you to drop (remove) existing databases, tables, and views.
- **LOCK TABLES:** Enables the use of explicit LOCK TABLES statements to lock tables for which you have the SELECT privilege. This includes the use of write locks, which prevents other sessions from reading the locked table.
- **REFERENCES:** This privilege gives the user the right to create foreign keys that point to tables of the specified database.
- **ALTER:** The ALTER privilege enables use of ALTER TABLE to change the structure of tables. ALTER TABLE also requires the CREATE and INSERT privileges. Renaming a table

requires ALTER and DROP on the old table, ALTER, CREATE, and INSERT on the new table.

- **DELETE:** Enables rows to be deleted from tables in a database.
- **INSERT:** Enables rows to be inserted into tables in a database. INSERT is also required for the ANALYZE TABLE, OPTIMIZE TABLE, and REPAIR TABLE table-maintenance statements.
- **SELECT:** The SELECT privilege enables you to select rows from tables in a database. SELECT statements require the SELECT privilege only if they actually retrieve rows from a table. Some SELECT statements do not access tables and can be executed without permission for any database. It is also needed for other statements that read column values.
- **UPDATE:** Enables rows to be updated in tables in a database.
- **INDEX:** Enables you to create or drop (remove) indexes. INDEX applies to existing tables. If you have the CREATE privilege for a table, you can include index definitions in the CREATE TABLE statement.
- **CREATE TEMPORARY TABLES:** Enables the creation of temporary tables using the CREATE TEMPORARY TABLE statement.
- **CREATE VIEW:** Enables use of CREATE VIEW.
- **CREATE ROUTINE:** It is needed to create stored routines (procedures and functions).
- **ALTER ROUTINE:** It is needed to alter or drop stored routines (procedures and functions).
- **EXECUTE ROUTINE:** This privilege gives the user the right to invoke existing stored procedures and functions of the specified database.
- **ALL or ALL PRIVILEGES:** It stands for “all privileges available at a given privilege level” (except GRANT OPTION). For example, granting ALL at the global or table level grants all global privileges or all table-level privileges.

2.1 Privilege System in SQL

The primary function of the MySQL privilege system is to authenticate a user who connects from a given host and to associate that user with privileges on a database such as SELECT, INSERT, UPDATE and DELETE. Additional functionality includes the ability to have anonymous users and to grant privileges for MySQL-specific functions such as LOAD DATA INFILE and administrative operations.

There are some things that you cannot do with the MySQL privilege system:

- You cannot explicitly specify that a given user should be denied access. That is, you cannot explicitly match a user and then refuse the connection.
- You cannot specify that a user has privileges to create or drop tables in a database but not to create or drop the database itself.
- A password applies globally to an account. You cannot associate a password with a specific object such as a database, table, or routine.

Internally, the server stores privilege information in the grant tables of the database named mysql. The MySQL server reads the contents of these tables into memory when it starts and bases access-control decisions on the in-memory copies of the grant tables. The MySQL privilege system ensures that all users may perform only the operations permitted to them. As a user,

Notes

when you connect to a MySQL server, your identity is determined by *the host from which you connect* and *the user name you specify*. When you issue requests after connecting, the system grants privileges according to your identity and *what you want to do*. MySQL considers both your host name and user name in identifying you because there is no reason to assume that a given user name belongs to the same person on all hosts.



Example: The user joe who connects from office.example.com need not be the same person as the user joe who connects from home.example.com. MySQL handles this by enabling you to distinguish users on different hosts that happen to have the same name: You can grant one set of privileges for connections by joe from office.example.com, and a different set of privileges for connections by joe from home.example.com. To see what privileges a given account has, use the SHOW GRANTS statement.

```
SHOW GRANTS FOR 'joe'@'office.example.com';
```

```
SHOW GRANTS FOR 'joe'@'home.example.com';
```

The Two-Step Authentication Process

MySQL access control involves two stages when you run a client program that connects to the server:

Stage 1: The server accepts or rejects the connection based on your identity and whether you can verify your identity by supplying the correct password.

Stage 2: Assuming that you can connect, the server checks each statement you issue to determine whether you have sufficient privileges to perform it.



Example: If you try to select rows from a table in a database or drop a table from the database, the server verifies that you have the SELECT privilege for the table or the DROP privilege for the database.

Self Assessment

State whether the following statements are true or false:

1. Administrative privileges enable users to manage operation of the MySQL server.
2. Administrative privileges are local ones.
3. Database privileges apply to a database and but not to objects within it.
4. Database privileges are granted for specific databases, or globally so that they apply to all databases.
5. LOCK TABLES do not include the use of write locks.
6. REFERENCES gives the user the right to create primary keys that point to tables of the specified database.
7. If you have the CREATE privilege for a table, you can include index definitions in the CREATE TABLE statement.
8. You cannot explicitly specify that a given user should be denied access in MySQL privilege system.

2.2 Working with User Privileges

By default a single user system will have the user to access all databases and perform any operations on them. But this needs to be controlled in order to prevent malicious data operations specially in case of multiple users. To do that, we have some MySQL commands that we will discuss below.

2.2.1 Adding Users through MySQL

To add a new user we can make use of the GRANT command. It is also used to provide access or privileges on the database objects to the users. The Syntax for the GRANT command is:

```
GRANT privileges.
ON databasename.tablename.
TO username@host.
IDENTIFIED BY "password";
```



Example:

```
mysql> CREATE USER 'abc'@'localhost' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'abc'@'localhost'
-> WITH GRANT OPTION;
mysql> CREATE USER 'abc'@'%' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'abc'@'%'
-> WITH GRANT OPTION;
mysql> CREATE USER 'admin'@'localhost';
mysql> GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost';
mysql> CREATE USER 'dummy'@'localhost';
```

The privileges you can grant are shown in Table 2.1.

Table 2.1: Grant Privileges

ALL [PRIVILEGES]	Grant all privileges at specified access level except GRANT OPTION
ALTER	Enable use of ALTER TABLE
ALTER ROUTINE	Enable stored routines to be altered or dropped
CREATE	Enable database and table creation
CREATE ROUTINE	Enable stored routine creation
CREATE TEMPORARY TABLES	Enable use of CREATE TEMPORARY TABLE
CREATE USER	Enable use of CREATE USER, DROP USER, RENAME USER, and REVOKE ALL PRIVILEGES
CREATE VIEW	Enable views to be created or altered
DELETE	Enable use of DELETE
DROP	Enable databases, tables, and views to be dropped
EVENT	Enable use of events for the Event Scheduler

Contd...

Notes

EXECUTE	Enable the user to execute stored routines
FILE	Enable the user to cause the server to read or write files
GRANT OPTION	Enable privileges to be granted to or removed from other accounts
INDEX	Enable indexes to be created or dropped
INSERT	Enable use of INSERT
LOCK TABLES	Enable use of LOCK TABLES on tables for which you have the SELECT privilege
PROCESS	Enable the user to see all processes with SHOW PROCESSLIST
REFERENCES	Not implemented
RELOAD	Enable use of FLUSH operations
REPLICATION CLIENT	Enable the user to ask where master or slave servers are
REPLICATION SLAVE	Enable replication slaves to read binary log events from the master
SELECT	Enable use of SELECT
SHOW DATABASES	Enable SHOW DATABASES to show all databases
SHOW VIEW	Enable use of SHOW CREATE VIEW
SHUTDOWN	Enable use of mysqladmin shutdown
SUPER	Enable use of other administrative operations such as CHANGE MASTER TO, KILL, PURGE BINARY LOGS, SET GLOBAL, and mysqladmin debug command
TRIGGER	Enable trigger operations
UPDATE	Enable use of UPDATE
USAGE	Synonym for “no privileges”

Source: <http://dev.mysql.com/doc/refman/5.1/en/grant.html>

2.2.2 Removing Privileges of MySQL

In order to revoke privileges from an account, you use the MySQL REVOKE statement. The syntax of MySQL REVOKE statement is as follows:

```
REVOKE privilege_type [(column_list)][, priv_type [(column_list)]]...
ON [object_type] privilege_level
FROM user [, user]...
```

You specify a list of privileges that you want to revoke from an account right after the REVOKE keyword. You need to separate privileges by comma. The ON clause specifies the privilege level at that privileges are to be revoked. After FROM keyword, you specify the account that you want to revoke the privileges. You can specify multiple accounts in the FROM clause. You separate the accounts by comma. In order to revoke privileges from an account, you must have GRANT OPTION privilege and privileges that you are revoking. To revoke all privileges, you use the following MySQL REVOKE syntax:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user]...
```

To execute the above command, you must have the global CREATE USER privilege or the UPDATE privilege for the *mysql* database.

To revoke proxy user, you use the REVOKE PROXY command as follows:

```
REVOKE PROXY ON user FROM user [, user]...
```

A proxy user is a valid user in MySQL who can impersonate as another user therefore the proxy user has all privileges of the user that it impersonates.

Before revoking privileges of a user, it is good practice to check if the user has the privileges by using the SHOW GRANTS statement as follows:

```
SHOW GRANTS FOR user;
```



Example: Suppose *rfc* account has privileges SELECT, UPDATE and DELETE in the *classicmodels* sample database. If you want to revoke UPDATE and DELETE privileges from the *rfc* account, you can do so as follows:

Step 1: Check the privileges of *rfc* account using SHOW GRANTS statement:

```
SHOW GRANTS FOR 'rfc'@'localhost';
GRANT SELECT, UPDATE, DELETE ON 'classicmodels'.* TO 'rfc'@'localhost'
```

If you have not followed the previous tutorial on granting privileges to user, you can first grant the SELECT, UPDATE and DELETE privileges for *rfc* account that connects from *localhost* to the *classicmodels* database as follows:

```
GRANT SELECT, UPDATE, DELETE ON classicmodels.* TO 'rfc'@'localhost';
```

Step 2: You can revoke the UPDATE and DELETE privileges from the *rfc* account:

```
REVOKE UPDATE, DELETE ON classicmodels.* FROM 'rfc'@'localhost';
```

Step 3: You can check the privileges of the *rfc* account again using the SHOW GRANTS command.

```
SHOW GRANTS FOR 'rfc'@'localhost';
GRANT SELECT ON 'classicmodels'.* TO 'rfc'@'localhost'
```

If you want to revoke all privileges of the *rfc* account, you run the following command:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'rfc'@'localhost';
```

If you check the privileges of the *rfc* account again, you will see the *rfc* account has no privilege.

```
SHOW GRANTS FOR 'rfc'@'localhost';
GRANT USAGE ON *.* TO 'rfc'@'localhost'
```



Notes USAGE privilege means no privileges in MySQL.

The effect of MySQL REVOKE statement depends on the privilege level as follows:

- Changes that are made to the global privileges only take effect when the client connects to the MySQL in the subsequent sessions. The changes are not applied to all current connected accounts.
- The change of database privileges is applied after the next USE statement.
- Table and column privilege's changes are applied to all queries that are issued after the changes are made.

Self Assessment

Fill in the blanks:

9. To add a new user we can make use of the command.

Notes

10. The privilege enables the use of events for the Event Scheduler.
11. The privilege enables the user to see all processes with SHOW PROCESSLIST.
12. enables the user to ask where master or slave servers are.
13. is a synonym for “no privileges”.
14. In order to revoke privileges from an account, you use the statement.
15. To revoke proxy user, you use the command.



Case Study

Zillow.com Deploys MySQL Cluster for High Growth with High Availability

Introduction

Zillow.com is an online real estate community where home-owners, buyers, sellers, and real estate agents and professionals can find and share vital information about homes, for free. Launched in early 2006 with Zestimate™ values and data on millions of U.S. homes, Zillow has since “opened up” the site to community input, data and dialogue, including Home Q&A. Zillow’s goal is to help people become smarter about real estate - what homes are worth, what’s for sale, and what local experts have to say about real estate and individual homes. One of the most-visited real estate websites, Zillow was the only online company named by Advertising Age magazine to its 2006 “Marketing 50” list of the most powerful consumer brands.

Their ‘Growing’ Challenge

As soon as Zillow launched, it immediately generated a massive amount of Web traffic. As the company expanded its data and services and started to promote its distinctive brand, interest in Zillow continued to climb. Zillow’s founders knew from the company’s inception that the site’s ability to quickly process and manage massive amounts of data — in real time — would be key to its success. The company identified a need for a low latency and reliable Web database that would enable them to continue to increase the capacity of their infrastructure indefinitely without sacrificing performance.

Their Scale-Out Solution

Zillow uses MySQL Cluster — a special high availability version of MySQL — as the high transaction, low latency, Web-centric database to conduct critical data processing for Zillow.com, in real time. The open source database has helped Zillow scale on a single interface to meet many of the database requirements that a high-growth Web site requires. While Zillow uses a variety of databases to address different data processing needs, MySQL Cluster on inexpensive hardware has proven to be a high availability and low-cost solution for its specialized website application. In an average month, Zillow.com now handles more than 4 million unique visitors.

MySQL Cluster combines the world’s most popular open source database with a fault tolerant database clustering architecture to deliver mission-critical database applications with 99.999 percent availability. With the performance and high throughput required to meet the most demanding applications, MySQL Cluster enables incremental, linear scaling without having to invest in expensive hardware. Scaling-out in a horizontal architecture is made possible through MySQL’s simple-yet-powerful replication technology, and helps

Contd...

many of the world's most popular online businesses — such as Zillow — grow gracefully and cost-effectively.

Zillow has also taken advantage of MySQL's expert consulting and support services. MySQL professional services provided Zillow with prototyping and capacity planning to make it easy for the company to launch its service. According to Zillow, its database administrators were able to quickly learn MySQL Cluster's simple interface and deliver highly available capabilities due to MySQL AB's responsive technical support team and thorough documentation.

As a result of their success with MySQL Cluster, Zillow is continuing to evaluate the database for new applications that could benefit from an affordable, Scale-Out architecture.

Questions

1. Study and analyze the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: <http://www.mysql.com/why-mysql/case-studies/mysql-cs-zillow.html>

2.3 Summary

- MySQL provides privileges that apply in different contexts and at different levels of operation.
- Administrative privileges enable users to manage operation of the MySQL server.
- Privileges for database objects can be granted for specific objects within a database, for all objects of a given type within a database.
- Database privileges apply to a database and to all objects within it.
- The primary function of the MySQL privilege system is to authenticate a user who connects from a given host and to associate that user with privileges on a database.
- To add a new user we can make use of the GRANT command.
- It is also used to provide access or privileges on the database objects to the users.
- In order to revoke privileges from an account, you use the MySQL REVOKE statement.
- Changes that are made to the global privileges only take effect when the client connects to the MySQL in the subsequent sessions.
- The change of database privileges is applied after the next USE statement.
- Table and column privilege's changes are applied to all queries that are issued after the changes are made.

2.4 Keywords

Index: It enables you to create or drop (remove) indexes.

References: This privilege gives the user the right to create foreign keys that point to tables of the specified database.

Reload: Enable use of FLUSH operations.

Replication Client: It enable the user to ask where master or slave servers are.

Notes

Replication Slave: It enables replication slaves to read binary log events from the master.

Routine. It is needed to create stored routines (procedures and functions).

Super: It enables use of other administrative operations such as CHANGE MASTER TO, KILL, PURGE BINARY LOGS, SET GLOBAL, and mysqladmin debug command.

Usage: Synonym for “no privileges”.

2.5 Review Questions

1. What are privileges?
2. Explain the types of privileges.
3. Give the syntax and use of REFERENCES command.
4. What is The Two-Step Authentication Process?
5. What is the purpose of using INDEX command?
6. List the privileges available with GRANT option.
7. Explain MySQL privilege system.
8. What is GRANT ALL?
9. How do you create a new user?
10. Explain REVOKE command with example.

Answers: Self Assessment

- | | |
|------------------|------------------------|
| 1. True | 2. False |
| 3. False | 4. True |
| 5. False | 6. False |
| 7. True | 8. True |
| 9. GRANT | 10. EVENT |
| 11. PROCESS | 12. REPLICATION CLIENT |
| 13. USAGE | 14. REVOKE |
| 15. REVOKE PROXY | |

2.6 Further Readings



Books

Anthony Molinaro, SQL Cookbook.

O reilly, Head First SQL.

Russell Dyer, MySQL in a Nutshell.

Sheldon and Moes, Beginning MySQL.

Notes



Online links

http://dev.mysql.com/doc/refman/5.1/en/privileges-provided.html#priv_all

http://docs.oracle.com/cd/F49540_01/DOC/server.815/a67772/privs.htm

http://en.wikipedia.org/wiki/Microsoft_SQL_Server

<http://www.mysqltutorial.org/mysql-revoke.aspx>

Unit 3: Apache Server

CONTENTS

Objectives

Introduction

3.1 Versions of Apache

3.1.1 Apache HTTP Server Version 2.0

3.1.2 Apache Mod

3.1.3 Hello World

3.1.4 Module Definition

3.1.5 Standard Module Stuff

3.1.6 Config Slots

3.2 Choosing Appropriate Installation Method

3.2.1 Installation Options

3.2.2 What You Need

3.2.3 Binary Installation

3.2.4 RPM (Red Hat Packet Manager) Installation

3.2.5 Build from Source

3.2.6 Starting Apache

3.2.7 Customize

3.2.8 Restarting Apache

3.3 Summary

3.4 Keywords

3.5 Review Questions

3.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain the Versions of Apache
- Discuss the Different Installation Method for Apache

Introduction

The Apache HTTP Server or Apache is a web server software program notable for playing a key role in the initial growth of the World Wide Web. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently named Oracle iPlanet Web Server). Typically Apache is run on a Unix-like operating system, and was developed for use on Linux. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating

systems, including Unix, FreeBSD, Linux, Solaris, Novell NetWare, OS X, Microsoft Windows, OS/2, TPF, and eComStation. Released under the Apache License, Apache is open-source software. Apache was originally based on NCSA HTTPd code. The NCSA code has since been removed from Apache, due to a rewrite. Since April 1996 Apache has been the most popular HTTP server software in use. As of December 2012 Apache was estimated to serve 63.7% of all active websites and 58.49% of the top servers across all domains.

3.1 Versions of Apache

3.1.1 Apache HTTP Server Version 2.0

emWare, Inc. (a Device Networking Company) released the original version having the look and feel similar to Apache's, can be found here in a zip file.



Did u know? In 2009, Apache became the first web server software to surpass the 100 million website milestone.

3.1.2 Apache Mod

You can include custom mods in Apache in two ways.

- You can build them into Apache but that would mean rebuilding all of Apache for every little change/addition you make to your mod.
- The second way to include a mod is to build Apache to load mods at start up. It build them with the following commands:

```
>./configure --prefix=PREFIX --enable-so
>make
>make install
```

Then it makes the user and group in httpd.conf match my own. They will look something like what is given below:

```
User billyboebob
Group billyboebob
```

3.1.3 Hello World

Given below is the code for Hello World.

```
#include "httpd.h"
#include "http_config.h"
#include "http_core.h"
#include "http_log.h"
#include "http_main.h"
#include "http_protocol.h"
#include "http_request.h"
#include "util_script.h"
#include "http_connection.h"
```

Notes

```
/* This example just takes a pointer to the request record as its only
 * argument */
static int hello_handler(request_rec *r)
{
    /* We decline to handle a request if hello-handler is not the value
     * of r->handler */
    if (strcmp(r->handler, "hello-handler")) {
        return DECLINED;
    }
    /* The following line just prints a message to the errorlog */
    ap_log_error(APLOG_MARK, APLOG_NOERRNO|APLOG_NOTICE, 0, r->server,
        "mod_hello: %s", "Before content is output");

    /* We set the content type before doing anything else */
    ap_set_content_type(r, "text/html");

    /* If the request is for a header only, and not a request for
     * the whole content, then return OK now. We don't have to do
     * anything else. */
    if (r->header_only) {
        return OK;
    }

    /* Now we just print the contents of the document using the
     * ap_rputs and ap_rprintf functions. More information about
     * the use of these can be found in http_protocol.h */
    ap_rputs("<HTML>\n", r);
    ap_rputs("\t<HEAD>\n", r);
    ap_rputs("\t\t<TITLE>\n\t\t\tHello There\n\t\t\t</TITLE>\n", r);
    ap_rputs("\t</HEAD>\n\n", r);
    ap_rputs("<BODY BGCOLOR=\"#FFFFFF\">\n" ,r);
    ap_rputs("<H1>Hello </H1>\n", r);
    ap_rputs("Hello world\n", r);
    ap_rprintf(r, "<br>A sample line generated by ap_rprintf<br>\n");
    ap_rputs("</BODY></HTML>\n" ,r);

    /* We can either return OK or DECLINED at this point. If we return
     * OK, then no other modules will attempt to process this request */
    return OK;
}
```

Notes

```

/* Each function our module provides to handle a particular hook is
 * specified here. See mod_example.c for more information about this
 * step. Suffice to say that we need to list all of our handlers in
 * here. */
static void x_register_hooks(apr_pool_t *p)
{
    ap_hook_handler(hello_handler, NULL, NULL, APR_HOOK_MIDDLE);
}

/* Module definition for configuration. We list all callback routines
 * that we provide the static hooks into our module from the other parts
 * of the server. This list tells the server what function will handle a
 * particular type or stage of request. If we don't provide a function
 * to handle a particular stage / callback, use NULL as a placeholder as
 * illustrated below. */
module AP_MODULE_DECLARE_DATA hello_module =
{
    STANDARD20_MODULE_STUFF,
    NULL, /* per-directory config creator */
    NULL, /* directory config merger */
    NULL, /* server config creator */
    NULL, /* server config merger */
    NULL, /* command table */
    x_register_hooks, /* other request processing hooks */
};

```

Save this code with the file name 'mod_hello.c'. Compile it using an apache tool called apxs which is located in the bin Directory of the Apache 2.0.x build.

```
`apxs -c -i -a mod_hello.c'.
```

This module merely prints Hello World to your browser when you make a request that this handler handles. To add the handler we write:

```

<Location /ourmap>
SetHandler helloworld2-handler
</Location>

```

3.1.4 Module Definition

An initial 'module' to describe an apache module. To define it, we write:

```
module AP_MODULE_DECLARE_DATA module_name=
```

3.1.5 Standard Module Stuff

The macro 'STANDARD20 MODULE STUFF' predefines the first eight out of the 14 slots in apache 2.0 module. It is defined in http_config.h.

3.1.6 Config Slots

We now discuss some config slots in apache 2.0.

Commands: This slot allows us to capture and handle the special commands sent to the Apache server in the HTTP Header. These are typically used to configure the module before the associated registered hooks are called.

Register Hooks: This is a pointer to a function that details the hooks that this module handles. The convention is for the function to be called, of all things, 'register_hooks'. There are many potential hooks but ap_hook_handler is the one we need.

helloworld2_handler: Our actual handler takes the form of:

```
static int handler_name (request_rec *r)
```

Our first action in the handler is to set the return messages content type to 'text/html'. We send the return header with:

```
ap_send_http_header(r);
```

Now we can send some text. In our case this is the 'Hello'. We finish up by sending the macro OK.

Self Assessment

State whether the following statements are true or false:

1. Apache was the first viable alternative to the Netscape Communications Corporation web server.
2. Apache is not open-source software.
3. An initial 'module' to describe an apache module.
4. The macro 'STANDARD20 MODULE STUFF' predefines the first eight out of the 16 slots in apache 2.0 module.
5. STANDARD20 MODULE STUFF is defined in "http_protocol.h".

3.2 Choosing Appropriate Installation Method

Apache 2.0 has a lot of features making it very popular. Some of them include:

- Multiple queries may be processed as threads or processes. The process management has been relocated to a separate module, called the multiprocessing module (MPM). Depending on the MPM, Apache 2 responds to queries in different ways, with different effects on the performance and the use of modules.
- The innards of Apache have been thoroughly revised. Apache now uses a new, special base library (Apache Portable Runtime, APR) as the interface to system functions and for the memory management. Moreover, important and widespread modules, such as mod_gzip (successor: mod_deflate) and mod_ssl, which have a profound impact on the processing of requests, are now integrated more fully in Apache.
- Apache 2 supports the Internet protocol IPv6.

- A new mechanism now enables manufacturers of modules to specify the desired loading sequence for modules. Thus, users are no longer required to do this themselves. The order in which modules are executed is often significant. Previously, it was determined by means of the loading sequence. For example, a module that only gives authenticated users access to certain resources must be run first to prevent the pages from being displayed to users who do not have any access permissions.
- Queries to and replies from Apache can be processed with filters.
- Support for files that are larger than 2 GB (large file support, LFS) on 32-bit systems.
- Some of the newer modules are only available for Apache 2.
- Multi-language error responses.

3.2.1 Installation Options

Apache HTTPD Web Server can be downloaded from the Apache website <http://www.apache.org>. We can install apache with binaries, source or RPM.

3.2.2 What You Need

To install Apache, you will need a computer with Linux operating system along with its root access. For binary and source installations, the tar and gunzip Unix utilities.

3.2.3 Binary Installation

A binary is pre-fab, which means someone else has gone to the trouble of configuring and building the software for you. Binaries are compiled for a particular operating system. In other words, you must use a binary built specifically for FreeBSD on your FreeBSD machine and a Linux binary on your Linux box. You need to be sure to grab the correct binary; if you don't see a binary for your particular operating system, you must choose a different method of installation. Apache binaries are usually a version or two behind the current source distribution. This means you don't reap the benefits of the latest bug fixes and feature enhancements. Because binaries are pre-configured, you don't have much opportunity to alter the way the software works. If you're a newcomer, you may not care about this loss of flexibility. Fortunately most Apache binaries include a full source distribution, providing you with the best of both worlds — play now, learn later. Now let's install a binary. Point your browser at <http://www.apache.org/dist/binaries> and download the binary for your operating system. You'll most likely be presented with a directory containing multiple versions of Apache in various compressed forms. If you can't find a binary for your operating system, choose either the "From Scratch" or "Using an RPM" (if you are running Red Hat Linux) method. Now let's uncompress the archive using the handy combination of gunzip and tar. You should replace the "apache_1.3.9-i386-whatever-freebsd3.2.tar.gz" text below with the name of the gzip'd file you downloaded.

```
gunzip < apache_1.3.9-i386-whatever-freebsd3.2.tar.gz | tar xvf -
```

Some of you may be lucky enough to have a version of tar that is capable of taking care of both tasks.

```
tar xvzf apache_1.3.9-i386-whatever-freebsd3.2.tar.gz
```

Either way, you should end up with an apache_1.3.x directory, with x being the particular sub-version of Apache you downloaded. Move into the newly created directory.

```
cd apache_1.3.x
```


Notes

Binary distributions contain an install script; mine's called `install.bindist.sh`. If your binary does not seem to contain such an install script, take a look at the `README.bindist` and/or `INSTALL.bindist` documents for further information;

To run the installed script,

```
./install.bindist.sh
```

This command should install the various bits of the Apache distribution into the appropriate locations; the default is usually to install everything under `/usr/local/apache`.

3.2.4 RPM (Red Hat Packet Manager) Installation

Apache Web Server is typically included in most Linux distributions. If you selected Apache as part of your installation it will already be installed and you won't need to install it again (unless you're upgrading). If Apache is already installed, you may also need to configure the web server to suit your requirements.

To check if Apache is already installed,

```
rpm -qa | grep apache
```

Install Apache

```
rpm -Ivh /<path_to_apache_files>/apache*
```

This installs all files in that directory with a name beginning with `apache`.

Configure Apache

Configuration is achieved by entering directives into the `httpd.conf` file (in the `/etc/httpd/conf` directory). Apache provides a default `httpd.conf` file with directives, some of which are commented out.

Depending on your requirements, you might not even need to change anything in the `httpd.conf` file, however there are some directives that you should customize for your site (`ServerName`, `ServerAdmin` etc).

Here are some of the more useful directives.

Server Configuration

```
#ServerName new.host.name
```

Sets the host name of the server. If not specified, the server attempts to deduce it from its own IP address. By default, this is commented out. To use it, simply remove the comments and specify the server name.

```
ServerAdmin you@your.address
```

Allows you to specify the administrator's email address for administrative issues. Simply replace the email address with your own (or your administrator's).

```
DocumentRoot /var/www/html
```

Specifies the root directory for your web files (`index.html` etc).

```
ServerRoot /etc/httpd
```

Specifies where your web server configuration, error, and log files are kept.

```
Redirect [ status ] url-path url
```

Maps an old URL into a new one. This can be useful if you move a page or directory to a new location, or if you delete a file and want to redirect users to another file.

Access Control

Access by remote users can be controlled by: Order allow, deny, Allow from all.

Individual Directories

To add directives to individual directories, you can either:

- Use the .htaccess file in the respective directory, or
- Type the following into the httpd.conf file:

```
<Directory directory> ... </Directory>
```



Notes You can also use the <Location> tag to add directives according to URL, <File> tag to add directives for individual files, and the <VirtualHost addr[:port]...> tag to add directives for a virtual host.

Test the web server

1. Type: /etc/rc.d/init.d/httpd start (starts the web server)
2. Open web browser and type http://localhost/ into the address bar
3. Type: /etc/rc.d/init.d/httpd stop (stops the web server)



Notes If the web server fails to start because it can't determine the server name, you may need to open the httpd.conf file, uncomment the **#ServerName** directive and add **localhost** for your server name (or your fully qualified domain name).

Type the following command, substituting the name of the .rpm you're using for apache-1.3.9-4.i386.rpm.

```
rpm -ivh apache-1.3.9-4.i386.rpm
```

RPM should grind away, displaying its progress with a primitive ##### progress bar. Barring any errors, you're done.

3.2.5 Build from Source

Building Apache from source may seem a daunting proposition to newcomers, but the Apache developers have done a wonderful job of making the task about as simple as could be. Just three more commands than a binary installation and you skip the arduous task of figuring out which binary is the right one for your particular operating system. Now let's uncompress that archive using gunzip and tar. You should replace the apache_1.3.11.tar.gz below with the name of the gzip'd file you downloaded.

```
gunzip < apache_1.3.11.tar.gz | tar xvf -
```

You should end up with an apache_1.3.x directory, x being the particular sub-version of Apache you downloaded. Move into the newly created directory.

```
cd apache_1.3.x
```

Notes

Now we'll use the configure and make commands to (you guessed it) configure, make, and install Apache. If you've not already done so, now would be the time to become root.

```
./configure
```

Your screen should look something like:

```
# ./configure
```

```
Configuring for Apache, Version 1.3.11
```

```
...
```

```
Creating Makefile
```

```
Creating Configuration.apaci in src
```

```
Creating Makefile in src
```

```
+ configured for Linux platform
```

```
+ setting C compiler to gcc
```

```
+ setting C pre-processor to gcc -E
```

```
+ checking for system header files
```

```
+ adding selected modules
```

```
+ checking size of various data types
```

```
+ doing sanity check on compiler and options
```

```
...
```

```
Creating Makefile in src/modules/standard
```

Unless errors were reported your Apache installation is now configured and we can move on.

Make

Your screen should look something like:

```
# make
```

```
==> src
```

```
make[1]: Entering directory `src/httpd/apache_1.3.11'
```

```
make[2]: Entering directory `src/httpd/apache_1.3.11/src'
```

```
==> src/regex
```

```
...
```

```
[several lines later]
```

```
...
```

```
gcc -DLINUX=2 -DUSE_HSREGEX -DUSE_EXPAT -I../lib/expat-lite
```

```
-DNO_DL_NEEDED `../apaci` -o ab -L../os/unix
```

```
-L../ap ab.o -lap -los -lm -lcrypt
```

```
make[2]: Leaving directory `src/httpd/apache_1.3.11/src/support'
```

```
<=== src/support
```

```
make[1]: Leaving directory `src/httpd/apache_1.3.11'
```

```
<=== src
```

```
#
```

Finally, you're ready to install your Apache build.

Notes

```
# make install
```

3.2.6 Starting Apache

Apache comes bundled with a script titled `apachectl` that facilitates starting, stopping and restarting Apache. Run the follow command to start Apache:

```
apache_install_dir/bin/apachectl start
```

Run the follow command to start Apache in SSL mode:

```
apache_install_dir/bin/apachectl startssl
```

To stop apache, run the following command:

```
apache_install_dir/bin/apachectl stop
```

After you start, you can test your installation of Apache. Once Apache is running, type the following address in your web browser: `http://localhost/`. If your installation was successful and Apache is running, you should see a test page displaying a message to that effect.

3.2.7 Customize

`/etc/httpd/conf` stores configuration text file on a Red Hat machine. In case you can't locate them, follow the steps below:

1. Login as root
2. Type: `cd /`
3. Type: `find -name httpd.conf`

After entering the `/etc/httpd/conf` you will see:

- *httpd.conf* – Server configuration settings
- *access.conf* – Security settings for Apache.
- *srm.conf* – MIME definitions and default document names for files on the server.

3.2.8 Restarting Apache

To see the effect of changes made to server configuration files, we need to restart our server. To do that,

```
cd /usr/local/apache2/bin
./apachectl
```

Self Assessment

Fill in the blanks:

6. The process management in apache 2.0 has been relocated to a separate module, called the
7. Apache now uses a new, special base library as the interface to system functions and for the memory management.
8. Apache 2 supports the Internet protocol

Notes

9. Apache 2 supports files that are larger than GB on 32-bit systems.
10. To uncompress the archive, we use the handy combination of and
11. To run the installed script, we write
12. The default is usually to install everything under
13. To check if Apache is already installed, we use
14. Configuration is achieved by entering directives into the file.
15. Apache comes bundled with a script titled that facilitates starting, stopping and restarting.



Case Study

PCextreme on Apache Cloudstack

PCextreme is one of the leading Internet Service Providers in Netherlands. The company offers a wide range of ready-to-use services including web hosting, collocation, dedicated servers, domain names and managed services. As a pioneer in the affordable hosting market, PCextreme operates with a level of scale and efficiency that allows them to combine reliability with competitive prices. What started out as a hobby for Wido den Hollander, CTO of PCextreme, quickly grew to become one of the leading hosting services providers in the Netherlands, serving over 40,000 customers and hosting 100,000 websites in two datacenters. PCextreme has deployed 120 racks of Supermicro server in those two datacenters.

The Challenge: Increase Business Agility to Enhance Competitiveness

To sustain rapid growth and maintain leadership in the affordable hosting market, PCextreme had to constantly innovate and expand their services to meet new demands. Key challenges included responding to the desire of customers to control their infrastructure without always having to use managed services, as well as achieving the flexibility to scale PCextreme services up and down based on customers' needs. Given rising energy prices in Europe, power consumption was another primary concern.

Adapting to these requirements called for a cloud solution that could easily integrate with PCextreme's existing environment, allow them to manage their infrastructure more efficiently and reduce energy costs—all while providing customers the control and flexibility that they demanded.

The Solution: Apache CloudStack

PCextreme looked at a variety of different cloud orchestration platforms for the foundation of their public clouds, and ultimately selected Apache CloudStack. The platform's product maturity, ease of implementation, open APIs and ability to seamlessly integrate with existing systems served as key factors in the decision. Wido den Hollander, CTO of PCextreme, recalls, "I carried CloudStack and a competing product's documentation with me during a three hour train ride. By the time my journey was over, a decision was made."

The openness of the CloudStack solution made it very easy for PCextreme to enhance the platform with features, such as integrating a new storage solution (the Ceph object store) with CloudStack, that were important for their business. Hollander, in addition of being the CTO of PCextreme, is an active CloudStack and libvirt developer and participant in

Contd...

Notes

the Ceph community. He is now an active committer to the Apache CloudStack project, and was invited to join the Podling Project Management Committee (PPMC), the technical leadership arm of Apache CloudStack. "It is very easy for small companies to contribute to the Apache Foundation, be heard and make a difference," remarks Hollander. Given PCextreme's interest in large-scale storage, Hollander took on the Ceph integration in CloudStack and is the sole contributor to this project.

PCextreme Cloud based on CloudStack is currently in beta phase and being used in production by 200 customers. It consists of 192 cores and 1.5TB of RAM using the KVM hypervisor. The image catalogue known as secondary storage is a single 2u machine with 20TB of disks. The primary storage used for the running instances is made of a 4u BSD node running the Z file system (ZFS). For additional cheaper storage, a Ceph deployment is used. The company's current Ceph deployment totals 250 TB.

In an attempt to augment CloudStack platform with new features and improvements, Hollander actively participates in storage refactoring to enable support for all the great features of Ceph such as snapshotting and cloning. He also looks forward to see support for long lasting instances and a better user interface.

The Benefit: Flexibility, Savings and Growth

CloudStack gave PCextreme a turnkey solution to offer cloud services to their customers with self-service provisioning and management customized to each unique environment. CloudStack's ability to integrate with existing systems enabled PCextreme to roll out their services to the market in less than four months. The open platform also made it possible for PCextreme to build all the tools around CloudStack in-house, giving them much-needed flexibility and customization.

CloudStack allows PCextreme the flexibility to scale resources up and down based on need, enabling the company to achieve a new level of efficiency and drive down overall cost. The ability to get more computing power out of the same datacenter saves energy and further reduces cost. The CloudStack APIs gives PCextreme customers flexibility to have their services spread over multiple companies and networks without being locked into a single vendor's solution. Hollander believes this is going to be a big game changer for many companies who are looking at expansion while minimizing risk. Customers also benefit from a toolkit that allows them to manage both traditional and cloud workloads ensuring flexibility and ease of migration.

The open source nature of Apache CloudStack also means that a company like PCextreme can invest development time to enhance the platform with the features that it needs. Hollander sets a great example by investing his time to contribute to the success of the CloudStack project while enabling much-needed services for his customers.

Today, the PCextreme cloud is powering a wide range of high-performance workloads across the globe. PCextreme sees the cloud as a catalyst for growth, and plans to not only continue to sell innovative cloud services publicly, but also to leverage CloudStack to augment their traditional business offerings and their internal infrastructure.

Questions

1. Study and analyze the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: <http://cloud.dzone.com/articles/case-study-pcextreme-apache>

3.3 Summary

- The Apache HTTP Server or Apache is a web server software program notable for playing a key role in the initial growth of the World Wide Web.
- Typically Apache is run on a Unix-like operating system, and was developed for use on Linux.
- You can include custom mods in Apache in two ways- You can build them into Apache but that would mean rebuilding all of Apache for every little change/addition you make to your mod or you can include a mod is to build Apache to load mods at start up.
- The macro 'STANDARD20 MODULE STUFF' predefines the first eight out of the 14 slots in apache 2.0 module. It is defined in http_config.h.
- To install Apache, you will need a computer with Linux operating system along with its root access. For binary and source installations, the tar and gunzip Unix utilities.
- A binary is pre-fab, which means someone else has gone to the trouble of configuring and building the software for you.
- Binaries are compiled for a particular operating system.
- Apache Web Server is typically included in most Linux distributions.
- We can use a RPM to install apache easily.

3.4 Keywords

access.conf: Security settings for Apache.

APR: Apache Portable Runtime, the special base library that acts as an interface to system functions and for the memory management.

Binary: Pre-fabricated file used for installation of apache.

httpd.conf: Server configuration settings.

IPV6: Internet Protocol Version 6.

MPM: Multiprocessing Module Responsible for process management.

RPM: Red Hat Packet Manager used for installation of apache.

srm.conf: MIME definitions and default document names for files on the server.

3.5 Review Questions

1. What do you mean by the Apache HTTP Server?
2. How can we include custom mods in Apache?
3. What are the Config Slots in macro 'STANDARD20 MODULE STUFF'?
4. Explain the term Configure Apache.
5. Give the steps for Binary Installation of apache.
6. How can we install apache using RPM (Red Hat Packet manager)?
7. What do you mean by build from source?
8. Give the command to customize an apache server.

9. Give the steps to test a web server.
10. How do you start and stop an apache server?

Notes

Answers: Self Assessment

- | | |
|----------------------------------|---------------------------------|
| 1. True | 2. False |
| 3. True | 4. False |
| 5. False | 6. Multiprocessing module (MPM) |
| 7. Apache Portable Runtime (APR) | 8. IPv6 |
| 9. 2 | 10. gunzip and tar |
| 11. ./install.bindist.sh | 12. /usr/local/apache |
| 13. rpm -qa grep apache | 14. httpd.conf |
| 15. apachectl | |

3.6 Further Readings



Books

Ben Laurie and Peter Laurie, Apache: The Definitive Guide.

Peter Wainwright, Professional Apache.

Rich Bowen and Ken Coar, Apache Cookbook, Second Edition - Solutions and Examples for Apache Administration.

The Apache HTTP Server Reference Manual (for Apache version 2.2.17), Apache Software Foundation.



Online links

<http://httpd.apache.org/docs/>

http://wiki.openoffice.org/wiki/Documentation/Building_Guide_AOO/Step_by_step

http://www.onlamp.com/pub/a/apache/2000/02/24/installing_apache.html?page=3

<http://www.packetwatch.net/documents/papers/apache20.pdf>

Unit 4: Apache Server Installation in Window

CONTENTS

Objectives

Introduction

- 4.1 Installation Method in Windows
 - 4.1.1 Building from Source
 - 4.1.2 Installing a Binary
- 4.2 Apache Configuration File Structure
 - 4.2.1 Directives
 - 4.2.2 Containers
 - 4.2.3 Conditional Evaluation
 - 4.2.4 ServerRoot
 - 4.2.5 Per Directory Configuration Files
- 4.3 Apache Log File
 - 4.3.1 Error Logs
 - 4.3.2 Apache Access Log File
 - 4.3.3 Tracking Website
 - 4.3.4 Log Rotation
- 4.4 Starting Apache for First Time
 - 4.4.1 Using Apache with Microsoft Windows
- 4.5 Summary
- 4.6 Keywords
- 4.7 Review Questions
- 4.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Define Installation Method in Widows
- Explain Apache Configuration File Structure
- Understand Log File
- Explain starting Apache for First Time

Introduction

Apache is generally recognized as the world's most popular Web server (HTTP server). Originally designed for Unix environments, the Apache Web server has been ported to Windows and other network operating systems. The name "Apache" derives from the word "patchy" that the Apache

developers used to describe early versions of their software. The Apache Web server provides a full range of Web server features, including CGI, SSL, and virtual domains. Apache also supports plug-in modules for extensibility. Apache is free software, distributed by the Apache Software Foundation that promotes various free and open source advanced Web technologies.

4.1 Installation Method in Windows

All Linux distributions comes with Apache. However, it is recommended to download latest Apache source code, compile and install on Linux. This will make it easier to upgrade Apache on an ongoing basis immediately after a new patch or release is available for download from Apache.

4.1.1 Building from Source

The steps to be followed to build apache from source are given below:

- **Download Apache**

Download the latest version from Apache HTTP Server Project. Current stable release of Apache is 2.2.9. Move the source to /usr/local/src and extract it as shown below.

```
# cd /usr/local/src
# gzip -d httpd-2.2.9.tar.gz
# tar xvf httpd-2.2.9.tar
```

- **Install Apache**

View all configuration options available for Apache using ./configure --help (two hyphen in front of help). The most commonly used option is --prefix={install-dir-name} to install Apache on a user defined directory.

```
# cd httpd-2.2.9
# ./configure --help
```

In the following example, Apache will be compiled and installed to the default location /usr/local/apache2 with the DSO capability. Using the --enable-so option, you can load modules to Apache at runtime via the Dynamic Shared Object (DSO) mechanism, rather than requiring a recompilation.

```
# ./configure --enable-so
# make
# make install
```



Notes During the ./configure, you may get the following error message.

```
# ./configure --enable-so
configure: error: no acceptable C compiler found in $PATH
See 'config.log' for more details.
configure failed for srclib/apr
```

Install the gcc and the dependent modules as shown below and try ./configure again to fix the above issue.

Notes

```
# rpm -ivh gcc-4.1.2-14.el5.i386.rpm glibc-devel-2.5-18.i386.rpm glibc-headers-2.5-18.i386.rpm kernel-headers-2.6.18-53.el5.i386.rpm
Preparing... ##### [100%]
1:kernel-headers ##### [ 25%]
2:glibc-headers ##### [ 50%]
3:glibc-devel ##### [ 75%]
4:gcc ##### [100%]
```

- Start Apache and verify installation

```
# cd /usr/local/apache2/bin
# ./apachectl start
```

Go to `http://local-host`, which should display the default message "It Works!"

4.1.2 Installing a Binary

Binary distributions are archived copies of the compiled Apache source and contain a complete Apache server, all supporting scripts and configuration files, a complete copy of the source code, and an installation script to install the server into the desired location. They don't, however, contain scripts to automatically start and stop Apache with the operating system. Using Unix as an example, first download either the compress archive (suffix `.Z`) or the gzip archive (suffix `.gz`). For other platforms, such as OS/2, you may also find a ZIP archive available. Because gzipped archives are compressed more efficiently than compressed ones, you should get the gzip archive if you have gzip available to decompress it. If you don't, this might be a good time to install it.

Once the archive is downloaded, unpack it:

```
$ gunzip apache_1.3.28-i686-whatever-linux22.tar.gz
$ tar xvf apache_1.3.28-i686-whatever-linux22.tar
```

This archive is for a Linux server, kernel version 2.2 or higher, running on a machine with at least an Intel Pentium II processor. Archives for other platforms are named appropriately in the relevant platform subdirectory. When a platform is available for more than one processor architecture, as Linux is, be careful to download the correct binary distribution—a Sparc or Alpha distribution is of little use to an Intel server, for example.

You can also extract the archive in one step, leaving the archive in a compressed state to conserve disk space:

```
$ gunzip -c apache_1.3.28-i686-whatever-linux22.tar.gz | tar xvf -
```

On systems with the GNU version of tar, which includes all Linux and BSD platforms, you can also extract the archive in one step (note the extra `z` in `zxvf` to signify that the archive is compressed):

```
$ tar zxvf apache_1.3.28-i686-whatever-linux22.tar.gz
```

On systems that don't have gzip installed, download the `.Z` archive and use the standard Unix uncompress utility instead:

```
$ uncompress apache_1.3.28-i686-whatever-linux22.tar.Z
```

This is actually the most complex part of the process. Once the archive is unpacked, go into the newly created Apache directory:

```
$ cd apache_1.3.28
```

Then run the included installation script:

```
$ ./install-bindist.sh
```

If you want to install Apache somewhere other than `/usr/local/apache`, give the installation script the path you want to use, for example:

```
$ ./install-bindist.sh /home/httpd/
```

This should produce a working Apache installation in the desired location. If you're installing on a Unix server and want to install into a standard system location, you'll need to have root privileges to perform this step. After the installation is complete, you may remove both the archive and the unpacked archive directory. You're now ready to configure Apache.



Did u know? An interesting point of binary distributions is that you can create them yourself, using the source distribution. This allows you to create a fully customized server that matches your needs and then distribute it yourself.

Self Assessment

State whether the following statements are true or false:

1. Apache is free software.
2. View all configuration options available for Apache using `./configure`
3. Apache will be compiled and installed to the default location `/usr/apache2`
4. You can load modules to Apache at runtime via the Dynamic Shared Object (DSO) mechanism.

4.2 Apache Configuration File Structure

Apache HTTP Server is configured by placing directives in plain text configuration files. The main configuration file is usually called `httpd.conf`. The location of this file is set at compile-time, but may be overridden with the `-f` command line flag. In addition, other configuration files may be added using the `Include` directive, and wildcards can be used to include many configuration files. Any directive may be placed in any of these configuration files. Changes to the main configuration files are only recognized by `httpd` when it is started or restarted. The server also reads a file containing mime document types; the filename is set by the `TypesConfig` directive, and is `mime.types` by default.

4.2.1 Directives

Directives placed in the main configuration files apply to the entire server. If you wish to change the configuration for only a part of the server, you can scope your directives by placing them in `<Directory>`, `<DirectoryMatch>`, `<Files>`, `<FilesMatch>`, `<Location>`, and `<LocationMatch>` sections. These sections limit the application of the directives which they enclose to particular file system locations or URLs. They can also be nested, allowing for very fine grained configuration. `httpd` has the capability to serve many different websites simultaneously. This is called Virtual Hosting. Directives can also be scoped by placing them inside `<VirtualHost>` sections, so that they will only apply to requests for a particular website. Although most directives can be placed in any of these sections, some directives do not make sense in some contexts.

Notes



Example: Directives controlling process creation can only be placed in the main server context.

Syntax: This indicates the format of the directive as it would appear in a configuration file. This syntax is extremely directive-specific, and is described in detail in the directive's definition. Generally, the directive name is followed by a series of one or more space-separated arguments. If an argument contains a space, the argument must be enclosed in double quotes. Optional arguments are enclosed in square brackets. Where an argument can take on more than one possible value, the possible values are separated by vertical bars "|". Literal text is presented in the default font, while argument-types for which substitution is necessary are *emphasized*. Directives which can take a variable number of arguments will end in "..." indicating that the last argument is repeated.

Directives use a great number of different argument types. A few common ones are defined below.

- **URL:** A complete Uniform Resource Locator including a scheme, hostname, and optional pathname as in `http://www.example.com/path/to/file.html`
- **URL-path:** The part of a *url* which follows the scheme and hostname as in `/path/to/file.html`. The *url-path* represents a web-view of a resource, as opposed to a file-system view.
- **file-path:** The path to a file in the local file-system beginning with the root directory as in `/usr/local/apache/htdocs/path/to/file.html`. Unless otherwise specified, a *file-path* which does not begin with a slash will be treated as relative to the `ServerRoot`.
- **directory-path:** The path to a directory in the local file-system beginning with the root directory as in `/usr/local/apache/htdocs/path/to/`.
- **Filename:** The name of a file with no accompanying path information as in `file.html`.
- **Regex:** A Perl-compatible regular expression. The directive definition will specify what the *regex* is matching against.
- **Extension:** In general, this is the part of the *filename* which follows the last dot. However, Apache recognizes multiple filename extensions, so if a *filename* contains more than one dot, each dot-separated part of the filename following the first dot is an *extension*. For example, the *filename* `file.html.en` contains two extensions: `.html` and `.en`. For Apache directives, you may specify *extensions* with or without the leading dot. In addition, *extensions* are not case sensitive.
- **MIME-type:** A method of describing the format of a file which consists of a major format type and a minor format type, separated by a slash as in `text/html`.
- **env variable:** The name of an environment variable defined in the Apache configuration process. Note this is not necessarily the same as an operating system environment variable.

Default: If the directive has a default value (*i.e.*, if you omit it from your configuration entirely, the Apache Web server will behave as though you set it to a particular value), it is described here. If there is no default value, this section should say "*None*".



Notes The default listed here is not necessarily the same as the value the directive takes in the default `httpd.conf` distributed with the server.

Context: This indicates where in the server's configuration files the directive is legal. It's a comma-separated list of one or more of the following values:

- **server config:** This means that the directive may be used in the server configuration files (e.g., httpd.conf), but **not** within any <VirtualHost> or <Directory> containers. It is not allowed in .htaccess files at all.
- **virtual host:** This context means that the directive may appear inside <VirtualHost> containers in the server configuration files.
- **Directory:** A directive marked as being valid in this context may be used inside <Directory>, <Location>, <Files>, and <Proxy> containers in the server configuration files, subject to the restrictions outlined in Configuration Sections.
- **.htaccess:** If a directive is valid in this context, it means that it can appear inside *per*-directory .htaccess files. It may not be processed, though depending upon the overrides currently active.

The directive is *only* allowed within the designated context; if you try to use it elsewhere, you'll get a configuration error that will either prevent the server from handling requests in that context correctly, or will keep the server from operating at all *i.e.*, the server won't even start. The valid locations for the directive are actually the result of a Boolean OR of all of the listed contexts. In other words, a directive that is marked as being valid in "server config, .htaccess" can be used in the httpd.conf file and in .htaccess files, but not within any <Directory> or <VirtualHost> containers.

Override: This directive attribute indicates which configuration override must be active in order for the directive to be processed when it appears in a .htaccess file. If the directive's context doesn't permit it to appear in .htaccess files, then no context will be listed. Overrides are activated by the AllowOverride directive, and apply to a particular scope (such as a directory) and all descendants, unless further modified by other AllowOverride directives at lower levels. The documentation for that directive also lists the possible override names available.

Status: This indicates how tightly bound into the Apache Web server the directive is; in other words, you may need to recompile the server with an enhanced set of modules in order to gain access to the directive and its functionality. Possible values for this attribute are:

- **Core:** If a directive is listed as having "Core" status, that means it is part of the innermost portions of the Apache Web server, and is always available.
- **MPM:** A directive labeled as having "MPM" status is provided by a Multi-Processing Module. This type of directive will be available if and only if you are using one of the MPMs listed on the Module line of the directive definition.
- **Base:** A directive labeled as having "Base" status is supported by one of the standard Apache modules which is compiled into the server by default, and is therefore normally available unless you've taken steps to remove the module from your configuration.
- **Extension:** A directive with "Extension" status is provided by one of the modules included with the Apache server kit, but the module isn't normally compiled into the server. To enable the directive and its functionality, you will need to change the server build configuration files and re-compile Apache.
- **Experimental:** "Experimental" status indicates that the directive is available as part of the Apache kit, but you're on your own if you try to use it. The directive is being documented for completeness, and is not necessarily supported. The module which provides the directive may or may not be compiled in by default; check the top of the page which describes the directive and its module to see if it remarks on the availability.

Notes

Module: This quite simply lists the name of the source module which defines the directive.

Compatibility: If the directive wasn't part of the original Apache version 2 distribution, the version in which it was introduced should be listed here. In addition, if the directive is available only on certain platforms, it will be noted here.

4.2.2 Containers

Directives in the configuration files may apply to the entire server, or they may be restricted to apply only to particular directories, files, hosts, or URLs. If directives are not inside a container, they belong to the default server scope (server config) and apply to the server as a whole.

The default Apache directive containers are discussed below:

`<Directory>` and `</Directory>` are used to enclose a group of directives that will apply only to the named directory, sub-directories of that directory, and the files within the respective directories. Any directive that is allowed in a directory context may be used. *Directory-path* is either the full path to a directory, or a wild-card string using Unix shell-style matching. In a wild-card string, `?` matches any single character, and `*` matches any sequences of characters. You may also use `[]` character ranges. None of the wildcards match a `'/'` character, so `<Directory /*/
public_html>` will not match `/home/user/public_html`, but `<Directory /home/*/public_html>` will match.



Example:

```
<Directory /usr/local/httpd/htdocs>  
Options Indexes FollowSymLinks  
</Directory>
```



Caution Be careful with the *directory-path* arguments. They have to literally match the file system path which Apache uses to access the files.

`<DirectoryMatch>` and `</DirectoryMatch>` are used to enclose a group of directives which will apply only to the named directory and *sub-directories of that directory* (and the files within), the same as `<Directory>`. However, it takes as an argument a regular expression.



Example:

```
<DirectoryMatch "^/www/(.+)?[0-9]{3}">
```

would match directories in `/www/` that consisted of three numbers.

The `<Files>` directive limits the scope of the enclosed directives by filename. It is comparable to the `<Directory>` and `<Location>` directives. It should be matched with a `</Files>` directive. The directives given within this section will be applied to any object with a basename (last component of filename) matching the specified filename. `<Files>` sections are processed in the order they appear in the configuration file, after the `<Directory>` sections and `.htaccess` files are read, but before `<Location>` sections.



Notes `<Files>` can be nested inside `<Directory>` sections to restrict the portion of the file system they apply to.

The `<FilesMatch>` directive limits the scope of the enclosed directives by filename, just as the `<Files>` directive does. However, it accepts a regular expression.



Example:

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

would match most common Internet graphics formats.

The `<Location>` directive limits the scope of the enclosed directives by URL. It is similar to the `<Directory>` directive, and starts a subsection which is terminated with a `</Location>` directive. `<Location>` sections are processed in the order they appear in the configuration file, after the `<Directory>` sections and `.htaccess` files are read, and after the `<Files>` sections. `<Location>` sections operate completely outside the file system. This has several consequences. Most importantly, `<Location>` directives should not be used to control access to file system locations. Since several different URLs may map to the same file system location, such access controls may be circumvented.

The `<LocationMatch>` directive limits the scope of the enclosed directives by URL, in an identical manner to `<Location>`. However, it takes a regular expression as an argument instead of a simple string.



Example:

```
<LocationMatch "/(extra|special)/data">
```

would match URLs that contained the substring `/extra/data` or `/special/data`.

4.2.3 Conditional Evaluation

There are two basic types of containers. Most containers are evaluated for each request. The enclosed directives are applied only for those requests that match the containers. The `<IfDefine>`, `<IfModule>`, and `<IfVersion>` containers, on the other hand, are evaluated only at server startup and restart. If their conditions are true at startup, then the enclosed directives will apply to all requests. If the conditions are not true, the enclosed directives will be ignored. The `<IfDefine>` directive encloses directives that will only be applied if an appropriate parameter is defined on the `httpd` command line. For example, with the following configuration, all requests will be redirected to another site only if the server is started using `httpd -DClosedForNow`:

```
<IfDefine ClosedForNow>
  Redirect / http://otherserver.example.com/
</IfDefine>
```

The `<IfModule>` directive is very similar, except it encloses directives that will only be applied if a particular module is available in the server. The module must either be statically compiled in the server, or it must be dynamically compiled and its `LoadModule` line must be earlier in the configuration file. This directive should only be used if you need your configuration file to work whether or not certain modules are installed. It should not be used to enclose directives that you want to work all the time, because it can suppress useful error messages about missing modules.

In the following example, the `MimeMagicFile` directive will be applied only if `mod_mime_magic` is available.

```
<IfModule mod_mime_magic.c>
  MimeMagicFile conf/magic
</IfModule>
```


Notes

4.2.4 ServerRoot

The ServerRoot directive sets the directory in which the server lives. Typically it will contain the subdirectories conf/ and logs/. Relative paths in other configuration directives (such as Include or LoadModule) are taken as relative to this directory.

```
ServerRoot "/home/httpd"
```

The default location of ServerRoot may be modified by using the `—` prefix argument to configure.

4.2.5 Per Directory Configuration Files

Apache allows for decentralized management of configuration via special files placed inside the web tree. The special files are usually called .htaccess, but any name can be specified in the AccessFileName directive. Directives placed in .htaccess files apply to the directory where you place the file, and all sub-directories. The .htaccess files follow the same syntax as the main configuration files. Since .htaccess files are read on every request, changes made in these files take immediate effect. The server administrator further controls what directives may be placed in .htaccess files by configuring the AllowOverride directive in the main configuration files. When the server finds an .htaccess file (as specified by AccessFileName) it needs to know which directives declared in that file can override earlier configuration directives. AllowOverride is valid only in <Directory> sections specified without regular expressions, not in <Location>, <DirectoryMatch> or <Files> sections. When this directive is set to None, then .htaccess files are completely ignored. In this case, the server will not even attempt to read .htaccess files in the filesystem. When this directive is set to All, then any directive which has the .htaccess Context is allowed in .htaccess files.

The *directive-type* can be one of the following groupings of directives.

- AuthConfig
- FileInfo
- Indexes
- Limit

Self Assessment

Fill in the blanks:

5. The main configuration file is usually called
6. The location of httpd.conf is set at compile-time, but may be overridden with the command line flag.
7. The filename is set by the TypesConfig directive, and is by default.
8. httpd has the capability to serve many different websites simultaneously. This is called
9. indicates where in the server's configuration files the directive is legal.
10. The directive sets the directory in which the server lives.

4.3 Apache Log File

In order to effectively manage a web server, it is necessary to get feedback about the activity and performance of the server as well as any problems that may be occurring. The Apache HTTP Server provides very comprehensive and flexible logging capabilities.

4.3.1 Error Logs

All apache errors diagnostic information other errors found during serving requests are logged to this file. Location of error log is set using ErrorLog directive. If there is any problem, you should first take a look at this file using cat, grep or any other UNIX / Linux text utilities. This apache log file often contain details of what **went wrong and how to fix it**. Default error log file location:

- RHEL / Red Hat / CentOS / Fedora Linux Apache error file location - **`/var/log/httpd/error_log`**
- Debian / Ubuntu Linux Apache error log file location - **`/var/log/apache2/error.log`**
- FreeBSD Apache error log file location - **`/var/log/httpd-error.log`**

To find exact apache log file location, you can use grep command:

```
# grep ErrorLog /usr/local/etc/apache22/httpd.conf
# grep ErrorLog /etc/apache2/apache2.conf
# grep ErrorLog /etc/httpd/conf/httpd.conf
```

Sample output:

```
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a
ErrorLog "/var/log/httpd-error.log"
```

4.3.2 Apache Access Log File

The server access log records all requests processed by the server. The location and content of the access log are controlled by the CustomLog directive. The LogFormat directive can be used to simplify the selection of the contents of the logs. Storing the information in the access log is only the start of log management. The next step is to analyze this information to produce useful statistics. Log analysis in general is beyond the scope of this document, and not really part of the job of the web server itself. Various versions of Apache httpd have used other modules and directives to control access logging, including mod_log_referer, mod_log_agent, and the TransferLog directive. The CustomLog directive now subsumes the functionality of all the older directives. The format of the access log is highly configurable. The format is specified using a format string that looks much like a C-style printf(1) format string.

4.3.3 Tracking Website

Log files are critical to **managing Apache**. Managing these logs can provide advance security warnings, provide detailed information in terms of who is visiting your site and where they are coming from and can help you troubleshoot the use of your server resources. The two main log files are listed below from a **CentOS server**.

The access_log file gives you information on who is using your web server and the error_log provides information for troubleshooting.

```
/var/log/httpd/access_log
/var/log/httpd/error_log
```

The log format is seen here:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%[Referer]i\" \"%[User-Agent]i\"" combinedInformation
%
```

Notes

The identifier with descriptions are given below:

```
Host    %h      IP address making request of server
ident   %l      identd daemon
authuser %u      authentication requests
Date    %t      date and time of the request
request %r      requested client services
status  %>s     three-digit status code
bytes   %b      number of bytes sent to client
Referrer  %{Referer} Web page from which this client came
User Agent %{User-Agent} Browser information
```

Hostname lookups are also available and can be turned on, though they are turned off by default. Hostname will try to evaluate if the IP Address is really associated with hostname that is says. This may be a good way to evaluate hostnames on a limited basis but it does require considerable server resources to carry out.

4.3.4 Log Rotation

On even a moderately busy server, the quantity of information stored in the log files is very large. The access log file typically grows 1 MB or more per 10,000 requests. It will consequently be necessary to periodically rotate the log files by moving or deleting the existing logs. This cannot be done while the server is running, because Apache will continue writing to the old log file as long as it holds the file open. Instead, the server must be restarted after the log files are moved or deleted so that it will open new log files. By using a *graceful* restart, the server can be instructed to open new log files without losing any existing or pending connections from clients. However, in order to accomplish this, the server must continue to write to the old log files while it finishes serving old requests. It is therefore necessary to wait for some time after the restart before doing any processing on the log files. A typical scenario that simply rotates the logs and compresses the old logs to save space is:

```
mv access_log access_log.old
mv error_log error_log.old
apachectl graceful
sleep 600
gzip access_log.old error_log.old
```

Another way to perform log rotation is using piped logs. Apache httpd is capable of writing error and access log files through a pipe to another process, rather than directly to a file. This capability dramatically increases the flexibility of logging, without adding code to the main server. In order to write logs to a pipe, simply replace the filename with the pipe character “|”, followed by the name of the executable which should accept log entries on its standard input. Apache will start the piped-log process when the server starts, and will restart it if it crashes while the server is running. Piped log processes are spawned by the parent Apache httpd process, and inherit the user id of that process. This means that piped log programs usually run as root. It is therefore very important to keep the programs simple and secure. One important use of piped logs is to allow log rotation without having to restart the server. The Apache HTTP Server includes a simple program called rotatelogs for this purpose.



Example: To rotate the logs every 24 hours, you can use:

```
CustomLog "|/usr/local/apache/bin/rotatelogs/var/log/access_log 86400" common
```

A similar, but much more flexible log rotation program called cronolog is available at an external site. As with conditional logging, piped logs are a very powerful tool, but they should not be used where a simpler solution like off-line post-processing is available.

Self Assessment

Notes

Fill in the blanks:

11. Location of error log is set using directive.
12. The location and content of the access log are controlled by the directive.

4.4 Starting Apache for First Time

Microsoft Windows does things a little differently. Apache on Windows offers a few other ways to manage things, more in line with the expected Windows way of doing things.



Example: You can install Apache as an NT service, or start and stop it from the Start menu. However, there are also several ways to start and stop Apache from the command line as well.

4.4.1 Using Apache with Microsoft Windows

We will now elaborate upon the process to use apache with Windows.

Operating System Requirements: The primary Windows platform for running Apache 2.5 is Windows 2000 or later. Always obtain and install the current service pack to avoid operating system bugs. Apache HTTP Server versions later than 2.2 will not run on any operating system earlier than Windows 2000.

Downloading Apache for Windows: The Apache HTTP Server Project itself does not provide binary releases of software, only source code. If you cannot compile the Apache HTTP Server yourself, you can obtain a binary package from numerous binary distributions available on the Internet.

Customizing Apache for Windows: Apache is configured by the files in the conf subdirectory. These are the same files used to configure the Unix version, but there are a few different directives for Apache on Windows.

The main differences in Apache for Windows are:

- Because Apache for Windows is multi-threaded, it does not use a separate process for each request, as Apache can on Unix. Instead there are usually only two Apache processes running: a parent process, and a child which handles the requests. Within the child process each request is handled by a separate thread.

The process management directives are also different:

MaxConnectionsPerChild: Like the Unix directive, this controls how many connections a single child process will serve before exiting. However, unlike on Unix, a replacement process is not instantly available. Use the default MaxConnectionsPerChild 0, unless instructed to change the behavior to overcome a memory leak in third party modules or in-process applications.



Caution The server configuration file is reread when a new child process is started. If you have modified httpd.conf, the new child may not start or you may receive unexpected results.

Notes

ThreadsPerChild: This directive is new. It tells the server how many threads it should use. This is the maximum number of connections the server can handle at once, so be sure to set this number high enough for your site if you get a lot of hits. The recommended default is ThreadsPerChild 150, but this must be adjusted to reflect the greatest anticipated number of simultaneous connections to accept.

- The directives that accept filenames as arguments must use Windows filenames instead of Unix ones. However, because Apache may interpret backslashes as an “escape character” sequence, you should consistently use forward slashes in path names, not backslashes.
- While filenames are generally case-insensitive on Windows, URLs are still treated internally as case-sensitive before they are mapped to the filesystem. For example, the <Location>, Alias, and ProxyPass directives all use case-sensitive arguments. For this reason, it is particularly important to use the <Directory> directive when attempting to limit access to content in the filesystem, since this directive applies to any content in a directory, regardless of how it is accessed. If you wish to assure that only lowercase is used in URLs, you can use something like:

```
RewriteEngine On
RewriteMap lowercase int:tolower
RewriteCond %{REQUEST_URI} [A-Z]
RewriteRule (.*) ${lowercase:$1} [R,L]
```

- When running, Apache needs write access only to the logs directory and any configured cache directory tree. Due to the issue of case insensitive and short 8.3 format names, Apache must validate all path names given. This means that each directory which Apache evaluates, from the drive root up to the directory leaf, must have read, list and traverse directory permissions. If Apache2.5 is installed at C:\Program Files, then the root directory, Program Files and Apache2.5 must all be visible to Apache.
 - Apache for Windows contains the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the \Apache2.5\modules directory. To activate these or other modules, the new LoadModule directive must be used. For example, to activate the status module, use the following (in addition to the status-activating directives in access.conf):
- ```
LoadModule status_module modules/mod_status.so
```
- Apache can also load ISAPI (Internet Server Application Programming Interface) extensions such as those used by Microsoft IIS and other Windows servers.



*Notes* Apache **cannot** load ISAPI Filters, and ISAPI Handlers with some Microsoft feature extensions will not work.

- When running CGI scripts, the method Apache uses to find the interpreter for the script is configurable using the ScriptInterpreterSource directive.
- Since it is often difficult to manage files with names like .htaccess in Windows, you may find it useful to change the name of this per-directory configuration file using the AccessFilename directive.
- Any errors during Apache startup are logged into the Windows event log when running on Windows NT. This mechanism acts as a backup for those situations where Apache is not yet prepared to use the error.log file. You can review the Windows Application Event Log

by using the Event Viewer, e.g. Start - Settings - Control Panel - Administrative Tools - Event Viewer.

Notes

**Running Apache as a Service:** Apache comes with a utility called the Apache Service Monitor. With it you can see and manage the state of all installed Apache services on any machine on your network. To be able to manage an Apache service with the monitor, you have to first install the service (either automatically via the installation or manually). You can install Apache as a Windows NT service as follows from the command prompt at the Apache bin subdirectory:

```
httpd.exe -k install
```

If you need to specify the name of the service you want to install, use the following command. You have to do this if you have several different service installations of Apache on your computer. If you specify a name during the install, you have to also specify it during any other -k operation.

```
httpd.exe -k install -n "MyServiceName"
```

If you need to have specifically named configuration files for different services, you must use this:

```
httpd.exe -k install -n "MyServiceName" -f "c:\files\my.conf"
```

If you use the first command without any special parameters except -k install, the service will be called Apache2.5 and the configuration will be assumed to be conf\httpd.conf.

Removing an Apache service is easy. Just use:

```
httpd.exe -k uninstall
```

The specific Apache service to be uninstalled can be specified by using:

```
httpd.exe -k uninstall -n "MyServiceName"
```

Normal starting, restarting and shutting down of an Apache service is usually done via the Apache Service Monitor, by using commands like NET START Apache2.5 and NET STOP Apache2.5 or via normal Windows service management. Before starting Apache as a service by any means, you should test the service's configuration file by using:

```
httpd.exe -n "MyServiceName" -t
```

You can control an Apache service by its command line switches, too. To start an installed Apache service you'll use this:

```
httpd.exe -k start -n "MyServiceName"
```

To stop an Apache service via the command line switches, use this:

```
httpd.exe -k stop -n "MyServiceName"
```

or

```
httpd.exe -k shutdown -n "MyServiceName"
```

You can also restart a running service and force it to reread its configuration file by using:

```
httpd.exe -k restart -n "MyServiceName"
```

By default, all Apache services are registered to run as the system user (the LocalSystem account). The LocalSystem account has no privileges to your network via any Windows-secured mechanism, including the file system, named pipes, DCOM, or secure RPC. It has, however, wide privileges locally.

## Notes



*Caution* Never grant any network privileges to the LocalSystem account! If you need Apache to be able to access network resources, create a separate account for Apache as noted below.

It is recommended that users create a separate account for running Apache service(s). If you have to access network resources via Apache, this is required.

1. Create a normal domain user account, and be sure to memorize its password.
2. Grant the newly-created user a privilege of Log on as a service and Act as part of the operating system. On Windows NT 4.0 these privileges are granted via User Manager for Domains, but on Windows 2000 and XP you probably want to use Group Policy for propagating these settings. You can also manually set these via the Local Security Policy MMC snap-in.
3. Confirm that the created account is a member of the Users group.
4. Grant the account read and execute (RX) rights to all document and script folders (htdocs and cgi-bin for example).
5. Grant the account change (RWXD) rights to the Apache logs directory.
6. Grant the account read and execute (RX) rights to the httpd.exe binary executable.

It is usually a good practice to grant the user the Apache service runs as read and execute (RX) access to the whole Apache2.5 directory, except the logs subdirectory, where the user has to have at least change (RWXD) rights.

If you allow the account to log in as a user and as a service, then you can log on with that account and test that the account has the privileges to execute the scripts, read the web pages, and that you can start Apache in a console window. If this works, and you have followed the steps above, Apache should execute as a service with no problems.

*Error code 2186* is a good indication that you need to review the “Log On As” configuration for the service, since Apache cannot access a required network resource. Also, pay close attention to the privileges of the user Apache is configured to run as.

When starting Apache as a service you may encounter an error message from the Windows Service Control Manager. For example, if you try to start Apache by using the Services applet in the Windows Control Panel, you may get the following message:

```
Could not start the Apache2.5 service on \\COMPUTER
```

```
Error 1067; The process terminated unexpectedly.
```

You will get this generic error if there is any problem with starting the Apache service. In order to see what is really causing the problem you should follow the instructions for Running Apache for Windows from the Command Prompt.

If you are having problems with the service, it is suggested you follow the instructions below to try starting httpd.exe from a console window, and work out the errors before struggling to start it as a service again.

*Running Apache as a Console Application:* Running Apache as a service is usually the recommended way to use it, but it is sometimes easier to work from the command line, especially during initial configuration and testing.

To run Apache from the command line as a console application, use the following command:

```
httpd.exe
```

Apache will execute, and will remain running until it is stopped by pressing Control-C.

You can also run Apache via the shortcut Start Apache in Console placed to Start Menu —> Programs —> Apache HTTP Server 2.5.xx —> Control Apache Server during the installation. This will open a console window and start Apache inside it. If you don't have Apache installed as a service, the window will remain visible until you stop Apache by pressing Control-C in the console window where Apache is running in. The server will exit in a few seconds. However, if you do have Apache installed as a service, the shortcut starts the service. If the Apache service is running already, the shortcut doesn't do anything.

If Apache is running as a service, you can tell it to stop by opening another console window and entering:

```
httpd.exe -k shutdown
```

Running as a service should be preferred over running in a console window because this lets Apache end any current operations and clean up gracefully.

But if the server is running in a console window, you can only stop it by pressing Control-C in the same window.

You can also tell Apache to restart. This forces it to reread the configuration file. Any operations in progress are allowed to complete without interruption. To restart Apache, either press Control-Break in the console window you used for starting Apache, or enter

```
httpd.exe -k restart
```

if the server is running as a service.

Note for people familiar with the Unix version of Apache: these commands provide a Windows equivalent to `kill -TERM pid` and `kill -USR1 pid`. The command line option used, `-k`, was chosen as a reminder of the kill command used on Unix.

If the Apache console window closes immediately or unexpectedly after startup, open the Command Prompt from the Start Menu —> Programs. Change to the folder to which you installed Apache, type the command `httpd.exe`, and read the error message. Then change to the logs folder, and review the `error.log` file for configuration mistakes. Assuming `httpd` was installed into `C:\Program Files\Apache Software Foundation\Apache2.5\`, you can do the following:

```
c:
cd "\Program Files\Apache Software Foundation\Apache2.5\bin"
httpd.exe
```

Then wait for Apache to stop, or press Control-C. Then enter the following:

```
cd ..\logs
more < error.log
```

When working with Apache it is important to know how it will find the configuration file. You can specify a configuration file on the command line in two ways:

- `-f` specifies an absolute or relative path to a particular configuration file:

```
httpd.exe -f "c:\my server files\anotherconfig.conf"
```

or

```
httpd.exe -f files\anotherconfig.conf
```



**Notes**

- -n specifies the installed Apache service whose configuration file is to be used:  
`httpd.exe -n "MyServiceName"`

In both of these cases, the proper ServerRoot should be set in the configuration file.

If you don't specify a configuration file with -f or -n, Apache will use the file name compiled into the server, such as `conf\httpd.conf`. This built-in path is relative to the installation directory. You can verify the compiled file name from a value labelled as `SERVER_CONFIG_FILE` when invoking Apache with the -V switch, like this:

```
httpd.exe -V
```

Apache will then try to determine its ServerRoot by trying the following, in this order:

1. A ServerRoot directive via the -C command line switch.
2. The -d switch on the command line.
3. Current working directory.
4. A registry entry which was created if you did a binary installation.
5. The server root compiled into the server. This is /apache by default, you can verify it by using `httpd.exe -V` and looking for a value labelled as `HTTPD_ROOT`.

If you did not do a binary install, Apache will in some scenarios complain about the missing registry key. This warning can be ignored if the server was otherwise able to find its configuration file.

The value of this key is the ServerRoot directory which contains the conf subdirectory. When Apache starts it reads the `httpd.conf` file from that directory. If this file contains a ServerRoot directive which contains a different directory from the one obtained from the registry key above, Apache will forget the registry key and use the directory from the configuration file. If you copy the Apache directory or configuration files to a new location it is vital that you update the ServerRoot directive in the `httpd.conf` file to reflect the new location.

**Testing the Installation:** After starting Apache (either in a console window or as a service) it will be listening on port 80 (unless you changed the Listen directive in the configuration files or installed Apache only for the current user). To connect to the server and access the default page, launch a browser and enter this URL:

```
http://localhost/
```

Apache should respond with a welcome page and you should see "It Works!". If nothing happens or you get an error, look in the `error.log` file in the logs subdirectory. If your host is not connected to the net, or if you have serious problems with your DNS (Domain Name Service) configuration, you may have to use this URL:

```
http://127.0.0.1/
```

If you happen to be running Apache on an alternate port, you need to explicitly put that in the URL:

```
http://127.0.0.1:8080/
```

Once your basic installation is working, you should configure it properly by editing the files in the conf subdirectory. Again, if you change the configuration of the Windows NT service for Apache, first attempt to start it from the command line to make sure that the service starts with no errors.

Because Apache **cannot** share the same port with another TCP/IP application, you may need to stop, uninstall or reconfigure certain other services before running Apache. These conflicting

services include other WWW servers, some firewall implementations, and even some client applications (such as Skype) which will use port 80 to attempt to bypass firewall issues.

## Self Assessment

State whether the following statements are true or false:

13. Apache is configured by the files in the conf subdirectory.
14. Apache for Windows is not multi-threaded.
15. Apache for Windows contains the ability to load modules at runtime.



Case Study

### Apache Binary Backdoors on Cpanel-based Servers

For the last few months we have been tracking server level compromises that have been utilizing malicious Apache modules (Darkleech) to inject malware into websites. However, during the last few months we started to see a change on how the injections were being done. On cPanel-based servers, instead of adding modules or modifying the Apache configuration, the attackers started to replace the Apache binary (httpd) with a malicious one. This new backdoor is very sophisticated and we worked with our friends from ESET to provide this report on what we are seeing.

#### *Detection*

In our previous posts, we recommended the utilization of tools like “rpm -Va” or “rpm -qf” or “dpkg -S” to see if the Apache modules were modified. However, those techniques won’t work against this backdoor. Since cPanel installs Apache inside /usr/local/apache and does not utilize the package managers, there is no single and simple command to detect if the Apache binary was modified.

They also keep the same timestamp on the binary, so you can’t see by the date of the file. A good and reliable way to identify the modified binary is by searching for “open\_tty” on the httpd directory:

```
grep -r open_tty /usr/local/apache/
```

If it finds open\_tty in your Apache binary, it is likely compromised, since the original Apache binary does not contain a call to open\_tty. Another interesting point is that if you try to just replace the bad binary with a good one, you will be denied, because they set the file attribute to immutable. So you have to run chattr -ai before replacing it:

```
chattr -ai /usr/local/apache/bin/httpd
```

#### *Injections*

The compromised binary doesn’t change anything in the site in terms of utilization or how the sites looks, however on some random requests (once per day per IP address) instead of just displaying the content, it also adds a malicious redirect. That causes the browser to load content from what seems to be random domains:

http://893111632ce77ff9.aliz.co.kr/index.php (62.212.130.115)

http://894651446c103f0e.after1201.com (62.212.130.115)

http://328aaaf8978cc492.ajintechno.co.kr (62.212.130.115)

Contd...

Notes

http://23024b407634252a.ajaxstudy.net (62.212.130.115)  
http://cdb9156b281f7b01.ajuelec.co.kr (62.212.130.115)  
http://894651446c103f0e.after1201.com (62.212.130.115)

..

And many others like that. So if a browser requests a javascript file, it would return a 302 (redirect) pointing to:

**Location:** http://dcb84fc82e1f7b01.alarm-gsm.be/index.php?j=originalfilebase64

Where "originalfilebase64" is a base64 encoded string of the URL that was requested. That allows the attackers to return the malware along with the original content. Once the malware is loaded it will redirect the site to spammy sites (most often porn pages). At the sites we analyzed, they were being pushed to http://amazingtubesites.org (seems off-line now). On some cases we also saw the redirection going to the Blackhole Exploit kit.

Note that those URL's change very often and the ESET team has identified more than 30,000 variations of them.

#### **The Backdoor**

Our friends from ESET (Marc-Etienne, Olivier Bilodeau and Pierre-Marc Bureau) also analyzed the binary and discovered a nasty hidden backdoor.

Linux/Cdorked.A is one of the most sophisticated Apache backdoor we have seen so far. Although we are still processing the data, our Livegrid system reports hundreds of compromised servers and thousands of potential victims. The backdoor leaves no traces on the hard drive of compromised hosts other than its modified httpd binary. All the information related to the backdoor is stored in shared memory, the configuration is pushed by the attacker through obfuscated HTTP requests that aren't logged in normal Apache logs. This means that no command and control information is stored anywhere on the system.

The HTTP server is equipped with a reverse connect backdoor that can be triggered via a special HTTP GET request. It is invoked when a request to a special path is done with a query string in a particular format, containing the hostname and port to connect. The client IP of the HTTP dialog is used as a key to decrypt the query string as a 4 byte XOR key. Additionally, IP specified in X-Real-IP or X-Forwarded-For headers will override the client IP as the XOR key. This means we can craft a X-Real-IP header that will in effect be a "\x00\x00\x00\x00" key...

As you can see, the attackers don't need any files to act as a backdoor and just use the Apache binary for it.

#### **The Random URLs**

One thing that strikes us as very suspicious is that most of random domains being used as the first level redirection are coming from legitimate sites with their DNS hosted at dothost.co.kr:

```
ajaxstudy.net name server ns1.dothost.co.kr.
ajaxstudy.net name server ns2.dothost.co.kr.
```

We are still unsure if those are compromised accounts or if the attackers got some type of access to their DNS to inject random sub domains to domains hosted there. We are still tracking how those URL's changed, so we will have to post more details later.

Contd...

**Final Thoughts**

When attackers get full root access to the server, they can do anything they want. From modifying configurations, to injecting modules and replacing binaries. However, their tactics are changing to make it even harder for admins to detect their presence and recover from the compromise.

We also don't have enough information to pinpoint how those servers are initially being hacked, but we are thinking through SSHD-based brute force attacks.

We will keep monitoring these attacks and we will provide more information as we get them.

**Questions**

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: <http://blog.sucuri.net/2013/04/apache-binary-backdoors-on-cpanel-based-servers.html>

**4.5 Summary**

- Apache is generally recognized as the world's most popular Web server (HTTP server).
- Apache is free software, distributed by the Apache Software Foundation that promotes various free and open source advanced Web technologies.
- Apache HTTP Server is configured by placing directives in plain text configuration files. The main configuration file is usually called httpd.conf.
- Changes to the main configuration files are only recognized by httpd when it is started or restarted.
- Directives placed in the main configuration files apply to the entire server. If you wish to change the configuration for only a part of the server, you can scope your directives by placing them in <Directory>, <DirectoryMatch>, <Files>, <FilesMatch>, <Location>, and <LocationMatch> sections.
- Directives in the configuration files may apply to the entire server, or they may be restricted to apply only to particular directories, files, hosts, or URLs using containers.
- The ServerRoot directive sets the directory in which the server lives.
- Apache allows for decentralized management of configuration via special files placed inside the web tree. The special files are usually called .htaccess.
- The Apache HTTP Server provides very comprehensive and flexible logging capabilities.
- Error Logs are the file where all apache errors diagnostic information other errors found during serving requests are logged.
- The server access log records all requests processed by the server.

**4.6 Keywords**

**Access Log:** It records all requests processed by the server.

**Binary Distributions:** They are archived copies of the compiled Apache source.

**Directives:** To change the configuration for only a part of the server.

**Notes**

**Error Logs:** It stores all apache errors diagnostic information other errors found during serving requests.

**MIME-type:** A method of describing the format of a file which consists of a major format type and a minor format type, separated by a slash as in text/html.

**Regex:** A Perl-compatible regular expression.

**Syntax:** This indicates the format of the directive as it would appear in a configuration file.

**URL:** A complete Uniform Resource Locator including a scheme, hostname, and optional pathname.

### 4.7 Review Questions

1. Explain the Installation Method in Windows.
2. Give the steps to Build apache from Source.
3. What is a binary?
4. How do we install apache with binaries?
5. What are sections?
6. What are directives? Explain with examples.
7. Does apache allow decentralized management of configuration? If yes, how?
8. Explain the two kinds of logging in Apache.
9. What is log rotation?
10. Explain the steps to configure apache with Microsoft windows.

### **Answers: Self Assessment**

- |                 |                    |
|-----------------|--------------------|
| 1. True         | 2. False           |
| 3. False        | 4. True            |
| 5. httpd.config | 6. -f              |
| 7. mime.types   | 8. Virtual Hosting |
| 9. Context      | 10. ServerRoot     |
| 11. Error Log   | 12. Custom Log     |
| 13. True        | 14. False          |
| 15. True        |                    |

### 4.8 Further Readings



Books

Ben Laurie and Peter Laurie, Apache: The Definitive Guide.

Peter Wainwright, Professional Apache.

Rael Dornfest, Getting, Installing, and Running Apache.

Rich Bowen, Daniel Lopez Ridruejo, Allan Liska, Apache Administrator's Handbook.

**Notes**



*Online links*

<http://dev.mysql.com/doc/refman/5.0/en/windows-installation.html>

<http://dev.mysql.com/doc/refman/5.0/en/windows-source-build.html>

<http://httpd.apache.org/docs/2.2/mod/core.html#locationmatch>

<http://net.tutsplus.com/tutorials/apache-2-basic-configuration-on-unix-like-systems/>

## Unit 5: PHP

### CONTENTS

Objectives

Introduction

5.1 Versions of PHP

5.1.1 PHP 3.0

5.1.2 PHP 4

5.1.3 PHP 5

5.2 Installation of PHP

5.2.1 Windows

5.2.2 Installing with PHP Windows Installer

5.2.3 Go-Pear.Org

5.2.4 Prerequisites

5.2.5 Going PEAR

5.3 Changing php.ini File

5.4 Testing Installation

5.4.1 All-in-One Packages

5.4.2 The PHP Installer

5.4.3 Manual Installation

5.5 Summary

5.6 Keywords

5.7 Review Questions

5.8 Further Readings

### Objectives

After studying this unit, you will be able to:

- Discuss PHP Versions
- Explain the Installation of PHP
- Elaborate upon the Basics of PHP Installation
- Explain Testing Installation
- Understand PHP String Handling Function

### Introduction

PHP is a server scripting language that is embedded in HTML, and is a powerful tool for making dynamic and interactive Web pages. It is a widely-used, free, and efficient alternative to

competitors such as Microsoft's ASP. It started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994. PHP is a recursive acronym for "PHP: Hypertext Preprocessor". It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server. PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time. It supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time. Common uses of PHP include performing system functions, handle forms, add, delete, modify elements within your database, access cookies variables and set cookies, restrict users to access some pages of your website and data encryption.

## **5.1 Versions of PHP**

Let us now discuss the versions available in PHP.

### **5.1.1 PHP 3.0**

It was the first version that closely resembles PHP as it exists today. Finding PHP/FI 2.0 still inefficient and lacking features they needed to power an eCommerce application they were developing for a university project, Andi Gutmans and Zeev Suraski of Tel Aviv, Israel, began yet another complete rewrite of the underlying parser in 1997. Approaching Rasmus online, they discussed various aspects of the current implementation and their redevelopment of PHP. In an effort to improve the engine and start building upon PHP/FI's existing user base, Andi, Rasmus, and Zeev decided to collaborate in the development of a new, independent programming language. This entirely new language was released under a new name, that removed the implication of limited personal use that the PHP/FI 2.0 name held. It was renamed simply 'PHP', with the meaning becoming a recursive acronym - PHP: Hypertext Preprocessor.

One of the biggest strengths of PHP 3.0 was its strong extensibility features. In addition to providing end users with a mature interface for multiple databases, protocols, and APIs, the ease of extending the language itself attracted dozens of developers who submitted a variety of modules. Arguably, this was the key to PHP 3.0's tremendous success. Other key features introduced in PHP 3.0 included object-oriented programming support and a far more powerful and consistent language syntax.

In June, 1998, with many new developers from around the world joining the effort, PHP 3.0 was announced by the new PHP Development Team as the official successor to PHP/FI 2.0. Active development of PHP/FI 2.0, which had all-but ceased as of November of the previous year, was now officially ended. After roughly nine months of open public testing, when the announcement of the official release of PHP 3.0 came, it was already installed on over 70,000 domains around the world, and was no longer limited to POSIX-compliant operating systems. A relatively small share of the domains reporting PHP as installed were hosted on servers running Windows 95, 98, and NT, and Macintosh. At its peak, PHP 3.0 was installed on approximately 10% of the web servers on the Internet.

### **5.1.2 PHP 4**

By the winter of 1998, shortly after PHP 3.0 was officially released, Andi Gutmans and Zeev Suraski had begun working on a rewrite of PHP's core. The design goals were to improve



**Notes**

performance of complex applications, and improve the modularity of PHP's code base. Such applications were made possible by PHP 3.0's new features and support for a wide variety of third party databases and APIs, but PHP 3.0 was not designed to handle such complex applications efficiently.

The new engine, dubbed 'Zend Engine' (comprised of their first names, Zeev and Andi), met these design goals successfully, and was first introduced in mid 1999. PHP 4.0, based on this engine, and coupled with a wide range of additional new features, was officially released in May 2000, almost two years after its predecessor. In addition to the highly improved performance of this version, PHP 4.0 included other key features such as support for many more web servers, HTTP sessions, output buffering, more secure ways of handling user input and several new language constructs.

### 5.1.3 PHP 5

PHP 5 was released in July 2004 after long development and several pre-releases. It is mainly driven by its core, the *Zend Engine 2.0* with a new object model and dozens of other new features.

PHP's development team includes dozens of developers, as well as dozens others working on PHP-related and supporting projects, such as PEAR, PECL, and documentation, and an underlying network infrastructure of well over one-hundred individual web servers on six of the seven continents of the world. Though only an estimate based upon statistics from previous years, it is safe to presume PHP is now installed on tens or even perhaps hundreds of millions of domains around the world.

### Self Assessment

State whether the following statements are true or false:

1. PHP is a client side scripting language.
2. PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
3. PHP 3.1 was the first version of PHP.
4. PHP 3.0 did not support third party databases and APIs.
5. PHP 5 was released in July 2004.

## 5.2 Installation of PHP

PEAR is the PHP Extension and Application Repository, a collection of open source classes that work together. Developers can use PEAR classes to generate HTML, make SOAP requests, send MIME mail, and a variety of other common tasks. PHP 4.3 includes the first stable release of PEAR. When using PHP, the PEAR Package Manager is already installed unless one has used the `./configure` option — `without-pear`.

If one uses a version of PHP that is supplied by Unix/Linux/BSD distributors it may be necessary to manually install PEAR. If you want to re-install the Package Manager, you can use the following provisional way:

```
$ wget http://pear.php.net/go-pear.phar
$ php go-pear.phar
```



*Notes* If the process just exits without any output, your syslog will probably contain the following lines:

```
suhosin[4705]: ALERT - Include filename ('phar://go-pear.phar/index.php')
is an URL that is not allowed
```

```
(attacker 'REMOTE_ADDR not set', file '/root/go-pear.phar', line 1236)
```

To work around this problem, enable the phar in `/etc/php5/conf.d/suhosin.ini`:

```
suhosin.executor.include.whitelist = phar
```

Notes

### 5.2.1 Windows

To install PHP on windows the first step is to install a binary distribution of PHP from <http://www.php.net/downloads.php>. The default location where the PHP install will end up is `C:\PHP`.

Figure 5.1: PHP Setup Wizard Welcome Screen



Source: [http://foundationphp.com/tutorials/php\\_installer.php](http://foundationphp.com/tutorials/php_installer.php)

### 5.2.2 Installing with PHP Windows Installer

The new Windows installer bases `php.ini` on `php.ini-recommended`, which imposes stricter standards than `php.ini-dist`, the version normally recommended for a development environment. This is generally to be welcomed, because it forces you to create scripts that are more secure. However, you need to be aware of the following differences:

- The wizard turns off the display of error messages and writes them instead to a log file. For development purposes, change the settings in `php.ini` like this: `display_errors = On`, `log_errors = Off`
- Magic quotes are turned off. Although many scripts rely on the use of magic quotes (the automatic insertion of backslashes in front of single and double quotes).
- Short open tags are turned off. This means that you must use the full opening PHP tag (`<?php`). Since this is the recommended practice, you should leave the setting unchanged.

Notes

### 5.2.3 Go-Pear.Org

To install the PEAR Installer and the PHP Foundation Classes make use of the website Go-Pear.Org. It consists of stable releases and makes the installation of PEAR very easy as it figures out the user's directory layout on its own. It is cross platform and can be run from the command line and from your web server.


### 5.2.4 Prerequisites

A CGI or CLI version of PHP is needed to execute go-pear outside the web server. This is because go-pear is written in PHP. When we install the web server PHP module, the CLI version of PHP is installed too.

To check if CLI is available or not run,

```
php -v
```

The php command is installed in the /usr/local/bin directory on UNIX, or c:\php on Windows by default.



*Notes* In Windows, the CLI version of PHP may also be called php-cli and so type php-cli instead of php.


### 5.2.5 Going PEAR

The base installation that comes with the PHP distribution, contains all the stuff that is needed to run the PEAR installation tools etc. If you have a recent installation of PHP, you can relax: The PEAR base installation is already there, unless you have compiled your PHP with the ./configure flag —without-pear. The packages that do not come with PHP can be installed with the PEAR package manager. Apart from installing packages, the PEAR package manager also handles some other tasks: It can create new packages on your machine, manage a registry of installed packages, check dependencies and it can interact with services on pear.php.net, and other PEAR compatible channel servers to get information about available packages.

*Windows:* After you have downloaded and installed PHP, you have to manually execute the batch file located in e.g. c:\php\go-pear.bat. The setup will ask you some questions and afterwards the PEAR Package Manager will be installed in the path, which you have specified during installation. Finally you have to add that installation path to your PATH environment. Either do this manually (Start > Control Panel > System > Environment) or run (double-click) the newly generated PEAR\_ENV.reg that's now found in the PHP source directory. After that you can access the PEAR Package Manager by running the command pear in a Windows Command Prompt. To update your PEAR installation, request <http://pear.php.net/go-pear> in your browser and save the output to a local file go-pear.php. You can then run

```
php go-pear.php
```

in a Windows Command Prompt to start the update process.



*Notes* After changing php.ini, you will need to restart your web server.

To check if PEAR is working, both pear and pecl tools should be available everywhere from the command line. For that to work, pear's binary directory should be in your PATH variable.

To verify it works, simply type pear. A list of commands should be shown:

```
$ pear
Commands:
build Build an Extension From C Source
bundle Unpacks a Pecl Package
channel-add Add a Channel
...
```

You should further test that PEAR is up to date:

```
$ pear version
PEAR Version: 1.7.2
PHP Version: 5.2.6RC4-pl0-gentoo
Zend Engine Version: 2.2.0
Running on: Linux ...
```

To use PEAR and PEAR compatible packages in your applications, you normally include them into your PHP scripts using require\_once(). For this to work, PEAR's php\_dir must be a part of PHP's include path.

1. First, check where PEAR installs .php files:

```
$ pear config-get php_dir
/usr/share/lib/php/
```



*Notes* For Windows and the default installation of WampServer it would typically be C:\wamp\bin\php\php5.2.6\pear

2. Now it's time to find which configuration file is used by your PHP installation. (Make sure WampServer is running or else this will not output anything to screen.) On the command line, execute:

```
$ php -ini
Configuration File (php.ini) Path: /etc/php/cli-php5
Loaded Configuration File: /etc/php/cli-php5/php.ini
Scan for additional .ini files in: /etc/php/cli-php5/ext-active
Additional .ini files parsed: /etc/php/cli-php5/ext-active/php_gtk2.ini,
/etc/php/cli-php5/ext-active/xdebug.ini
```

#### **My Initial Results:**

```
Configuration File (php.ini) Path: C:\Windows
Loaded Configuration File: C:\wamp\bin\php\php5.2.6\php.ini
Scan for additional .ini files in: (none)
Additional .ini files parsed: (none)
```

To see which php.ini is used by PHP on your web server, create a file with only <?php phpinfo(); ?> as the contents, and save it in your local web root as check\_php.php. Open the file in your browser as http://localhost/check\_php.php, to find the path to the php.ini file your web server is using.

**Notes**

3. Now check PHP's include\_path setting on command line:

```
php -c /path/to/php.ini -r `echo get_include_path()."\n";`
```

To check PHP's include\_path in your web server, create a file with only

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

as the contents, and save it in your local web root as check\_php.php. Open the file in your browser as http://localhost/check\_php.php, to verify the include\_path your web server is using.

In every case, PEAR's php\_dir should be in the include path. If not, add it in your system's php.ini.

4. Create a new .php file with the following contents (I named it as

```
"test_for_system_file.php") :

<?php
require_once `System.php` ;

?>
```

System.php is shipped with every PEAR installation and thus should be on your computer, too. Open the file with the browser from your web server, and also try it on command line. There should be no output. A message like:

```
Warning: require_once(System.php): failed to open stream:
No such file or directory in /path/to/test.php on line 2
```

This means that your include path is not correct.

*To install PEAR with Command line installer follow the steps below:*

After getting PEAR working on your machine (see Installation) you most likely want to install some packages. This guide shows people new to the PEAR command line installer how to get started.

The general command to install a PEAR package named "foo" is

```
pear install foo
```

Typing this and pressing return, the package will be downloaded and installed on your computer. It does not matter if you write the package name in lowercase, UPPERCASE or MixedCase - the installer will find the package by lowercasing the name.

When a package is already installed, you will get the following message:

```
pear install foo
Ignoring installed package pear/foo
Nothing to install
```

This happens even if there is a newer version of the package! The correct command to upgrade to the latest version is

```
pear upgrade foo
upgrade ok: channel://pear.php.net/Foo-1.2.3
```

If the package already has the latest version, you will get a message similar to the following: Ignoring installed package pear/Foo

```
Nothing to upgrade
```

## Notes

A package often requires other packages to be installed to function correctly. Such a relation is called a dependency. The PEAR installer has full support for dependencies; it can automatically install required and/or optional dependencies if you wish so.

If you try to install a package with required dependencies, you will get an error that the installation failed. Looking deeper and actually reading the messages shows you that the package needs dependencies that are not installed on your system:

```
pear install html_page2
Did not download dependencies: pear/HTML_Common,
use -alldeps or -onlyreqdeps to download automatically
pear/HTML_Page2 requires package "pear/HTML_Common" (version >= 1.2)
No valid packages found
install failed
```

You have several choices:

- Install dependent packages by hand
- Let PEAR automatically install necessary dependencies only
- Let PEAR automatically install necessary and optional dependencies

The first method can be a painful and daunting process, because dependent packages itself can have dependencies.

Both other methods just require a switch to the install command, either `—onlyreqdeps` (install required dependencies only) or `—alldeps` (install all dependencies).

```
pear install -onlyreqdeps html_page2
WARNING: "pear/HTML_Common" is deprecated in favor of "pear/HTML_Common2"
downloading HTML_Page2-0.5.0beta.tgz ...
Starting to download HTML_Page2-0.5.0beta.tgz (15,467 bytes)
.....done: 15,467 bytes
downloading HTML_Common-1.2.4.tgz ...
Starting to download HTML_Common-1.2.4.tgz (4,519 bytes)
...done: 4,519 bytes
install ok: channel://pear.php.net/HTML_Common-1.2.4
install ok: channel://pear.php.net/HTML_Page2-0.5.0beta
```

You can download individual packages for example, off-line installation on a second machine just as you would install or upgrade a package:

```
pear download Foo
```

After downloading, you will have a file like `Foo-1.2.3.tgz` if the latest version of `Foo` was 1.2.3.

Installing it is as easy as typing

```
pear install Foo-1.2.3.tgz
```

Example using `Image_GraphViz-1.2.1.tgz`

```
C:\pear download Image_GraphViz-1.2.1.tgz
```

```
C:\pear install -alldeps Image_GraphViz-1.2.1.tgz
```

Notes

**Self Assessment**

Fill in the blanks:

- 6. A collection of open source classes that work together constitutes a .....
- 7. The default location where the PHP install will end up is .....
- 8. A ..... or ..... version of PHP is needed to execute go-pear outside the web server.
- 9. To check if CLI is available or not run, execute the command .....
- 10. The php command is installed in the ..... directory on UNIX by default.

**5.3 Changing php.ini File**

In order to change the php.ini file, check /usr/local/php/lib, or the lib subdirectory of the PHP installation location you used at configuration time for a Linux/Unix system. Whereas, on Windows, this file should be in the Windows directory. The php.ini file has two kinds of Directives: values and flags. Value takes in the name of the directive and a value separated by an equal sign and the Flag directives take the form of a directive name and a positive or negative term separated by an equal sign.

**5.4 Testing Installation**

We will now discuss PHP installation on Windows with Apache 2.2.

**5.4.1 All-in-One Packages**

It is a way to install PHP easily as it contains Apache, PHP, MySQL and other applications in a single installation file.



*Example:* XAMPP is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.

**5.4.2 The PHP Installer**

Another way to install PHP is using the PHP installer. But if you want to learn the details of the system, a manual installation is preferred.

**5.4.3 Manual Installation**

**Step 1:** Download and unzip the latest version of PHP

Download the latest zipped distribution of PHP from <http://php.net>. Unzip it. The recommendation is to put all the PHP stuff in a folder just off of the root drive (avoid whitespace), like C:\PHP.

**Step 2:** Rename/copy php.ini-recommended to php.ini

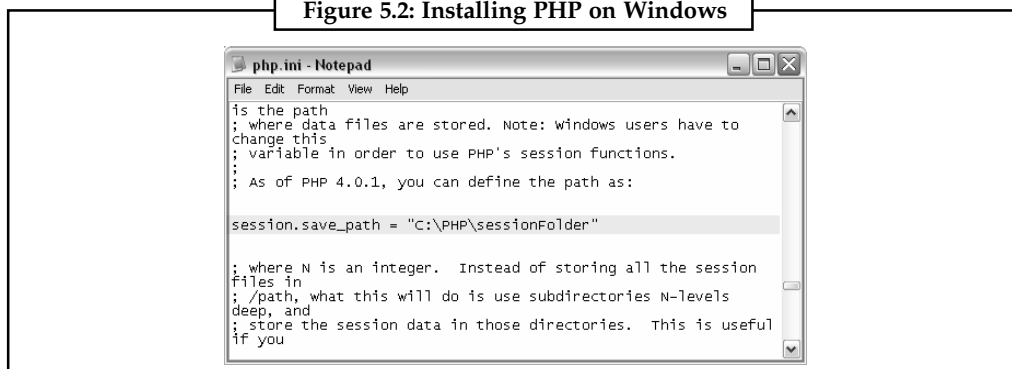
In your PHP directory, you'll find a couple of php.ini-\* files. They are pre-configured settings for a PHP install that you can use as an initial setup. The php.ini-recommended is the most secure, hence, the recommended one; just rename it to php.ini and copy to your Windows directory.

**Step 3:** Create a session state directory and point the session.save\_path variable in php.ini to it

Notes

This is optional, but recommended step. PHP does not need sessions, but it's something that will most likely be useful. Create a session directory somewhere on the server. I created C:\PHP\sessionFolder. This directory will hold many small files with session variable information for PHP. Now change the value of the session.save\_path variable in php.ini to be the full path to that directory (session.save\_path=C:\PHP\sessionFolder).

Figure 5.2: Installing PHP on Windows



Source: [http://webcheatsheet.com/PHP/install\\_and\\_configure.php#apacheconfig](http://webcheatsheet.com/PHP/install_and_configure.php#apacheconfig)

**Step 4:** Setup the PHP extensions

You need to point PHP to the directory that holds the extension libraries and you need to uncomment the desired extensions.

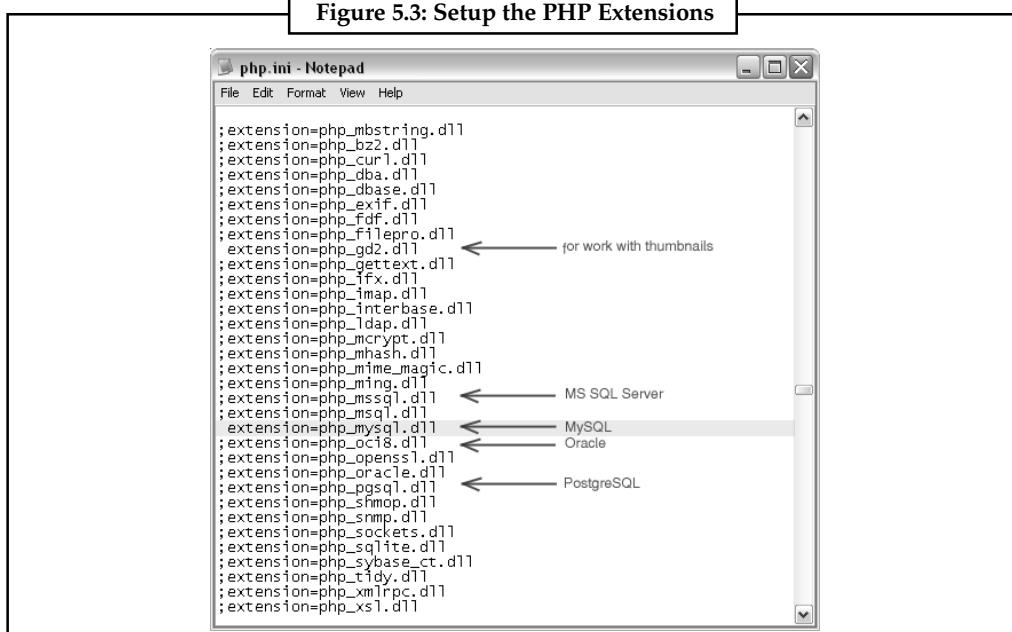
- Point PHP to the correct directory:

Set extension\_dir in php.ini to "C:\PHP\ext" (extension\_dir = "C:\PHP\ext")

- Uncomment the ones you want to use.

It's important to be sure that php\_mysql.dll extension is uncommented (for PHP 5 or newer).

Figure 5.3: Setup the PHP Extensions

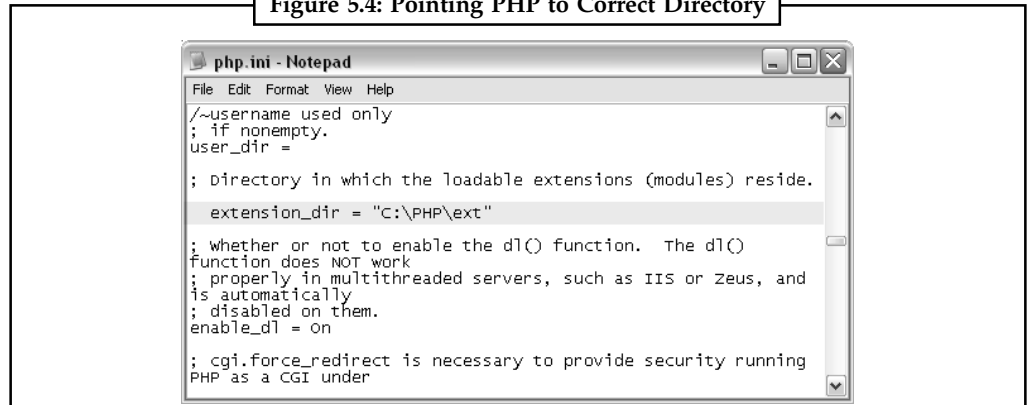


Source: [http://webcheatsheet.com/PHP/install\\_and\\_configure.php#apacheconfig](http://webcheatsheet.com/PHP/install_and_configure.php#apacheconfig)



Notes

Figure 5.4: Pointing PHP to Correct Directory



Source: [http://webcheatsheet.com/PHP/install\\_and\\_configure.php#apacheconfig](http://webcheatsheet.com/PHP/install_and_configure.php#apacheconfig)

Step 5: Make sure that PHP folder is in the system path

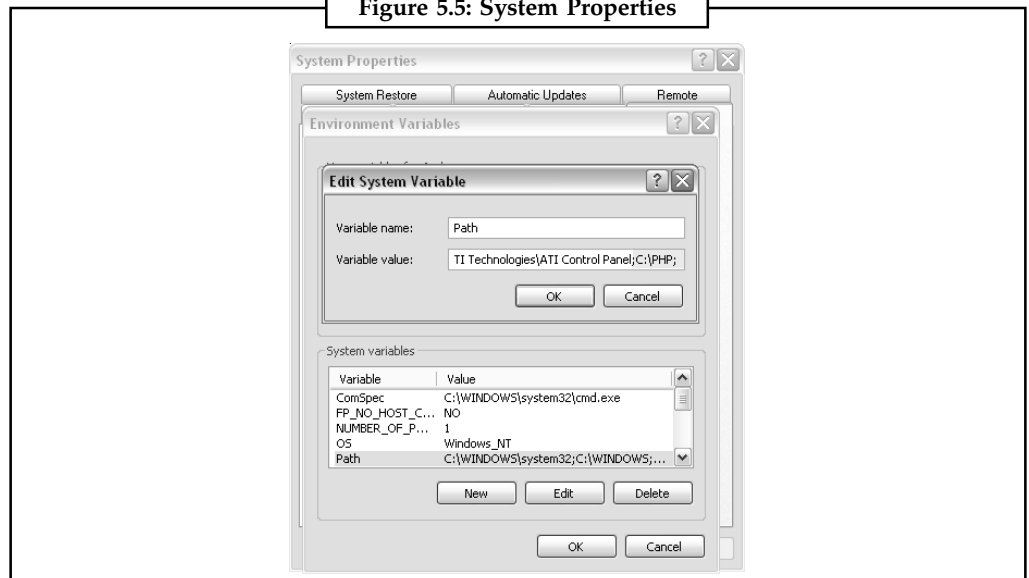
You should add "C:\PHP" to the server's PATH environment variable:

- Right-click on My Computer, choose Properties
- Flip to the Advanced tab
- Click the Environment Variables button
- Double-click the Path variable in the list of System variables.
- Either add "C:\PHP;" to the beginning or ";C:\PHP" to the end (sans quotes, not both).
- Restart IIS for it to take effect.

(To restart IIS you should right-click the local computer in the left pane of IIS Manager, click on All Tasks -> Restart IIS... -> OK)

Instead, you can copy all non-php dll files from C:\PHP to C:\Windows\System32 (or somewhere else in the server's PATH), but the first is the preferred method since it keeps the installation in one place, making upgrading or uninstalling easier.

Figure 5.5: System Properties



Source: [http://webcheatsheet.com/PHP/install\\_and\\_configure.php#apacheconfig](http://webcheatsheet.com/PHP/install_and_configure.php#apacheconfig)

**Step 6: Configure IIS****Notes**

For these steps, open IIS Manager

(Start -> Control Panel -> Administrative Tools -> Internet Information Services (IIS) Manager).

Then add new extension (.php)

- Expand the local computer in the left pane
- Right-click on "Web Sites" in the left pane, then click "Properties" in the menu that pops up
- Flip top the Home Directory tab
- Click "Configuration"
- Flip to the Mappings tab
- Click Add...
- Enter the full path to php5isapi.dll in the "Executable" textbox (Browse... to find it more easily if you need to)
- Enter ".php" in the Extension textbox
- Select radial button Limit to, enter "GET,POST,HEAD"
- Click OK all the way out

This will apply to every website. This sets up IIS to actually respond to requests for php files. Until now, IIS hadn't known what to do with php files, you just told it to pass them through php5isapi.dll.

**Step 7: Configure Apache Web Server**

If you want PHP to work with your Apache server, you will need to modify your Apache configuration file to load it. There are two ways to configure Apache to use PHP: one is to configure it to load the PHP interpreter as an Apache module. The other is to configure it to run the PHP interpreter as a CGI binary. Unless you have a particular reason for running PHP as a CGI binary, you will probably want to load PHP as a module in Apache, since it runs more efficiently that way. To configure Apache to load PHP as a module to parse your PHP scripts you should make some changes in the Apache configuration file, "httpd.conf", typically found in "c:\Program Files\Apache Group\Apache\conf\". It also can be accessed from your program files menu.

- Search for the section that has a series of commented out "LoadModule" statements. Add the following line after all the LoadModule statements:

```
LoadModule php5_module "c:/php/php5apache2.dll"
```

- Search for "AddType" and add the following line after the last "AddType" statement:

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

If you need to support other file types, like ".php3" and ".phtml", simply add them to the list, like this:

```
AddType application/x-httpd-php .php3
```

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

**Step 8: Test your setup**

Create a new file named test.php in one of the websites. Expand the websites folder in the left pane of IIS Manager to see a list of existing websites. Right-click on a website -> Properties ->

**Notes**

Home Directory -> Local Path will show you where the website root directory is. Create test.php file with the following line: <?php phpinfo(); ?>

With your browser go to http://localhost/test.php

After loading test.php, you should see some information about your PHP installation. Be sure to scroll all the way down to the bottom to see that there were no errors. Pay attention to "Configuration File (php.ini) Path" field. Field's value is current location of php.ini file and you should make appropriate changes in it.

**Self Assessment**

State whether the following statements are true or false:

11. The php.ini file has two kinds of Directives: values and flags.
12. Flag takes in the name of the directive and a value separated by an equal sign.
13. Value directives take the form of a directive name and a positive or negative term separated by an equal sign.
14. httpd.conf is typically found in "c:\Program Files\Apache Group\Apache\conf\".
15. We expand the Web Sites folder in the left pane of IIS Manager to see a list of existing websites.



Case Study

**How Colgate Used Online Video, Social Media and Mobile to Drive Engagement and Purchase Intent**

Colgate-Palmolive had a unique marketing challenge in launching Colgate Wisp, its new mini disposable toothbrush. Colgate began introducing the mini brush in April 2009 with help from Big Fuel, a social media marketing agency. The mini brush created a new product category for Colgate and meant marketing to a young, urban target—18- to 25- year-old men and women—a demographic the personal care giant doesn't typically focus dedicated attention on. It was clear that the company needed to figure out how to introduce the product into relevant conversations and contexts where its college student and young professional target hangs out.

**Challenge:** Colgate wanted to get Wisp into the hands of young, urban consumers who are active daters. The audience is active and mobile and dating opportunities can be created in an instant via text. "Wisp is almost a brand new product category," said Avi Savar, Founding Partner and CEO of Big Fuel. "It's an on-the-go product. The biggest challenge for us was making the product and brand relevant to the young consumer market."

Not surprisingly, Colgate turned to social media to help it launch a multi-pronged campaign. But who wants to "friend" or follow a disposable toothbrush on Facebook? Colgate and Big Fuel tackled the challenge by conducting a lot of research. Big Fuel worked up several creative strategies and testing the concepts, "We wanted to know, what does this product represent or mean to the audience?" Savar said.

Typically, Colgate talks to moms, but with Wisp, the marketer knew it needed unique social media components to introduce the product and seed interest. Big Fuel worked closely with Y&R and VML, Colgate's creative and digital agencies respectively on the TV campaign, microsite, online banners and social media elements.

Contd...

## Notes

**Strategy:** Big Fuel came up with a “Be More Kissable” creative platform that positioned Colgate Wisp as a kind of technology advancement that it believed would connect with the target audience. The idea centred around self-confidence: “Everyone wants to be more kissable not just within the context of a physical kiss, but all the time. Feeling kissable is about feeling confident. From a social media standpoint, we thought it was a good platform,” Savar explained. Colgate thought so too.

The concept, one of four that Big Fuel developed, was tested in four different markets. The linchpin involved creating irreverent online video content and syndicating it on YouTube and other video-sharing hubs. Along with a strategy focus on online video, Colgate Wisp developed a Facebook application and a “Be the Face of Wisp” photo contest. At the heart of the strategy was online video. Big Fuel developed a series of viral videos, partnering with eight different publishers including CollegeHumor and YourTango and Web celebrities like Kip Kay, known for his how-to and prank videos, to syndicate the content. It released eight wacky videos targeting niche interests among the target audience, contextually integrating Colgate Wisp into how-to, comedy and talk show-genre video content. The goal was to achieve a seamless content integration with no heavy brand sell. Online video syndication offered Colgate the potential to scale its vast consumer target.

The photo contest sought to identify the most kissable person in America: Participants who entered the contest uploaded a photo to colgatewisp.com and received a widget that enabled friends to vote for them. The widget was shared via the Facebook and MySpace networks and via the microsite. “It was like a syndicated version of „Are you hot or not? Savar said.

Big Fuel turned the contest into a social experience by enabling the widget to syndicate the photo content. Participants uploaded their photo, chose a specific Wisp color and placed it in the photo as an overlay. The contest enabled segmentation by geographic area as well. For example, when a man entered the contest, he could choose to look only at women in Chicago who entered the contest and decide whether they were kissable or not.

On average, Big Fuel reports that there were 11 votes cast per person or one individual voting on 11 different people. To drive brand engagement further, Big Fuel created a Facebook app called Spin the Wisp. Once the app was installed, it had the names of the consumer’s Facebook friends. Consumers could have the app randomly pick Facebook friends for the game or they could handpick up to 16 people to fill it. The Wisp landed on exotic locations and flavors—a woman could send a virtual kiss from Paris to her crush. Spin the Wisp became a novel way to flirt.

**Results:** Big Fuel reports that a Real Life Twitter video produced with CollegeHumor netted more than 1.7 million plus views. The video featured man-in-the-street style interviews by a stand-up comic who walked around blurting out things like: “I just found this new wisp. Anybody want a kiss?”

The Kip Kaye video “Quick Draw Gadget” in which Kip constructs a quick draw gadget out of a Colgate Wisp, has generated more than 1 million views. In total, the eight videos in the “Be More Kissable” series racked up more than 4.1 million views on YouTube as of late June 2010.

The two most recent videos for Colgate Wisp are College Humor POV “New Years Eve” which logged 1,255,872 views and Michelle Phans “Kissable Lips” video which has 1,791,352 views as of late June. All the videos were seeded on multiple video-sharing sites. The game saw a 10% click-through rate. Each time someone received a virtual kiss, they got a notification that appeared on their wall. The 10% click-through rate was based on the total number engagements vis-à-vis the notifications.

Contd...

## Notes

The average number of spins per install on Spin the Wisp was 7.6. There were more than 100,000 engagements and 40,000 + installations of the widget and more than 1 million unique impressions of the widget. There were 500,000 views of a faux Wisp infomercial. Overall, as of May, 2010, Big Fuel reported 6 million+ total engagements with the Wisp campaign (widget installs, video views, game plays, pass-alongs). Big Fuel considered “engagement” as active participation, meaning someone played the game, shared it, watched a video—there was a 10-second minimum on viewing—and commented on a video, Savar said.

**Key Takeaways:** Colgate learned the value of what an engagement is, according to Savar. “It was the first time they ever measured anything based on engagements. They are accustomed to the number of impressions.”

Now, Colgate is working to extend the engagement metric to its more mature brands. The brand has begun to understand what the value of video, game and other content is vs. framing content only within the context of an ad buy, Savar explained. While the campaign was in the market for four months, the videos and game continue to run.

**Next Steps:** Colgate has moved forward with content marketing and social marketing for others of its product brands. Colgate shot new videos for the Wisp product site and two additional viral video. The brand says it’s looking to turn customers into audiences and its brands into social identities.

### Questions

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: <http://www.aber.ac.uk/en/media/MMM3210-12.pdf>

## 5.5 Summary

- PHP is a server scripting language that is embedded in HTML.
- PHP is a recursive acronym for “PHP: Hypertext Preprocessor”.
- PHP 3.0 was the first version of PHP.
- PEAR is the PHP Extension and Application Repository, a collection of open source classes that work together.
- The default location where the PHP install will end up is C:\PHP.
- A CGI or CLI version of PHP is needed to execute go-pear outside the web server
- To check if CLI is available or not run, `php -v`
- The `php` command is installed in the `/usr/local/bin` directory on UNIX, or `c:\php` on Windows by default.
- The `php.ini` file has two kinds of Directives: values and flags.
- Value takes in the name of the directive and a value separated by an equal sign.
- Flag directives take the form of a directive name and a positive or negative term separated by an equal sign.

## 5.6 Keywords

Notes

**Apache:** A web server software program notable for playing a key role in the initial growth of the World Wide Web.

**Binary:** They are archived copies of the compiled source.

**Directive:** To change the configuration for only a part of the server.

**HTML:** Hyper Text Markup Language for data presentation.

**IIS:** Internet Information Server is a group of Internet servers.

**MIME:** A method of describing the format of a file which consists of a major format type and a minor format type, separated by a slash as in text/html.

**PEAR:** It is the PHP Extension and Application Repository.

**PHP:** It is a recursive acronym for PHP: Hypertext Preprocessor.

## 5.7 Review Questions

1. What is PHP?
2. Explain the PHP versions.
3. How do we install PHP with PEAR?
4. What are the steps to install PHP with PHP Win Installer?
5. What is go-pear?
6. How do we install go-pear in Windows?
7. Give the steps to install go-pear through command line.
8. Explain the step by step process to install PHP manually.
9. Explain how do we configure Apache?
10. How do you test your setup post installation?

### Answers: Self Assessment

- |           |                    |
|-----------|--------------------|
| 1. False  | 2. True            |
| 3. False  | 4. False           |
| 5. True   | 6. PEAR            |
| 7. C:/PHP | 8. CGI, CLI        |
| 9. php -v | 10. /usr/local/bin |
| 11. True  | 12. False          |
| 13. False | 14. True           |
| 15. True  |                    |

Notes

## 5.8 Further Readings



*Books*

Luke Welling and Laura Thomson, PHP and MySQL Web Development.

Lynn Beighley and Michael Morrison, Head First PHP & MySQL.

Peter MacIntyre and Rasmus Lerdorf , Programming PHP, Kevin Tatroe.

Robin Nixon , Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites.



*Online links*

[http://webcheatsheet.com/PHP/install\\_and\\_configure.php#apacheconfig](http://webcheatsheet.com/PHP/install_and_configure.php#apacheconfig)

<http://www.tizag.com/phpT/>

[http://www.tutorialspoint.com/php/php\\_introduction.htm](http://www.tutorialspoint.com/php/php_introduction.htm)

<http://www.w3schools.com/php/>

## Unit 6: Building Blocks of PHP

Notes

### CONTENTS

Objectives

Introduction

- 6.1 Building Blocks of PHP
  - 6.1.1 Variables
  - 6.1.2 Data Types
  - 6.1.3 Operators
  - 6.1.4 Expressions
  - 6.1.5 PHP Constants
  - 6.1.6 Switch Flow
  - 6.1.7 Loop Control Structures
  - 6.1.8 Browser Output
- 6.2 Summary
- 6.3 Keywords
- 6.4 Review Questions
- 6.5 Further Readings

### Objectives

After studying this unit, you will be able to:

- Discuss PHP Variables and Its Data Types
- Explain Types of Operators
- Elaborate the Concept of Expressions
- Discuss PHP Constants

### Introduction

The major reasons to use PHP include: PHP runs on different platforms (Windows, Linux, Unix, Mac OS X, etc.), it is compatible with almost all servers used today (Apache, IIS, etc.), it has support for a wide range of databases, it is easy to learn and is obviously free as its open source. We will now discuss about the basic building blocks of PHP like data types, literals, variables, and constants.

### 6.1 Building Blocks of PHP

Let us take each topic in detail now.

#### 6.1.1 Variables

A variable is a means of storing a value, such as text string "Hello World!" or the integer value 4. A variable can then be reused throughout your code, instead of having to type out the actual value over and over again.



**Notes**

In PHP you define a variable with the following form:

```
$variable_name = Value;
```



*Caution* If you forget that dollar sign at the beginning, it will not work.

Variable names are case-sensitive, so use the exact same capitalization when using a variable. The variables \$a\_number and \$A\_number are different variables in PHP's eyes. PHP does not require variables to be declared before being initialized. When you assign a text value to a variable, put quotes around the value. There are a few rules that you need to follow when choosing a name for your PHP variables.

- PHP variables must start with a letter or underscore “\_”.
- PHP variables may only be comprised of alphanumeric characters and underscores. a-z, A-Z, 0-9, or \_.
- Variables with more than one word should be separated with underscores. \$my\_variable
- Variables with more than one word can also be distinguished with capitalization. \$myVariable



*Example:*

```
<?php
$hello = "Hello World!";
$a_number = 4;
$anotherNumber = 8;
?>
```

The scope of a variable is the part of the script where the variable can be referenced/used. PHP has four different variable scopes:

**Local:** A variable declared within a PHP function is local and can only be accessed within that function. You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared. Local variables are deleted as soon as the function is completed.

**Global:** A variable that is defined outside of any function, has a global scope. Global variables can be accessed from any part of the script, EXCEPT from within a function. To access a global variable from within a function, use the global keyword. PHP also stores all global variables in an array called \$GLOBALS[index]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

**Static:** When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted. To do this, use the static keyword when you first declare the variable.

**Parameter:** A parameter is a local variable whose value is passed to the function by the calling code. Parameters are declared in a parameter list as part of the function declaration.

### Indirect References to Variables

Sometimes it is convenient to be able to have a variable name which can be set and used dynamically.

A normal variable is set with a statement such as:

```
<?php
$a = 'hello';
?>
```

This kind of variable takes the value of a variable and treats that as the name of a variable. In the above example, *hello*, can be used as the name of a variable by using two dollar signs i.e.

```
<?php
$$a = 'world';
?>
```

At this point two variables have been defined and stored in the PHP symbol tree: *\$a* with contents "hello" and *\$hello* with contents "world". Therefore, this statement:

```
<?php
echo "$a ${$a}";
?>
```

produces the exact same output as:

```
<?php
echo "$a $hello";
?>
```

i.e. they both produce: hello world.

## Managing Variables

We can manage variable using two kinds of constructs.

**isset():** The `isset()` function is used to check whether a variable is set or not. If a variable is already unset with `unset()` function, it will no longer be set. The `isset()` function return false if testing variable contains a NULL value. Its syntax is,

```
isset(variable1, variable2.....)
```

where variable 1 and 2 are the variables to be checked. It returns a Boolean TRUE if variable (variable1,variable2..) exists and has value not equal to NULL, FALSE otherwise.



*Example:*

```
<?php
$var1 = 'test';
var_dump(isset($var1));
?>
```

Output: true

**unset():** The `unset()` function destroys a given variable. Its syntax is,

```
unset (var1, var2.....)
```

where var 1 and 2 are the variables to be unset. It returns no value after execution.

## Notes



*Example:*

```
<? php
$xyz='abc';
echo 'Before using unset() the value of $xys is : '. $xyz.'
';
unset($xyz);
echo 'After using unset() the value of $xys is : '. $xyz; ?>
```

### **Output:**

Before using unset() the value of \$xys is : abc  
After using unset() the value of \$xys is :

## **Empty**

The empty() function is used to check whether a variable is empty or not. Its syntax is,  
empty(var\_name)

where var\_name is the name of the variable that needs to be checked. It returns FALSE if var\_name has a non-empty and non-zero value.

List of empty things:

- "0" (0 as a string)
- 0 (0 as an integer)
- "" (an empty string)
- NULL
- FALSE
- "" (an empty string)
- array() (an empty array)
- \$var\_name; (a variable declared but without a value in a class)



*Example:*

```
<?php
$ivar1=0;
$str1='Learning empty';
if (empty($ivar1))
{
echo '$ivar1.'" is empty or 0.
";
}
else
{
echo '$ivar1.'" is not empty or 0.
";
}
```

```

if (empty($istr1))
{
echo '$istr1'." is empty or 0.
";
}
else
{
echo '$istr1' ." string is not empty or 0.
";
}
?>

```

**Output:**

```

$ivar1 is empty or 0.
$istr1 string is not empty or 0.

```

**Superglobals**

Several predefined variables in PHP are “superglobals”, which means they are available in all scopes throughout a script. There is no need to do **global \$variable**; to access them within functions or methods. These superglobal variables are:

- **\$GLOBALS**: An associative array containing references to all variables which are currently defined in the global scope of the script. The variable names are the keys of the array.
- **\$\_SERVER**: `$_SERVER` is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server.
- **\$\_GET**: An associative array of variables passed to the current script via the URL parameters.
- **\$\_POST**: An associative array of variables passed to the current script via the HTTP POST method.
- **\$\_FILES**: An associative array of items uploaded to the current script via the HTTP POST method.
- **\$\_COOKIE**: An associative array of variables passed to the current script via HTTP Cookies.
- **\$\_SESSION**: An associative array containing session variables available to the current script.
- **\$\_REQUEST**: An associative array that by default contains the contents of `$_GET`, `$_POST` and `$_COOKIE`.
- **\$\_ENV**: An associative array of variables passed to the current script via the environment method.

**Self Assessment**

State whether the following statements are true or false:

1. Variable names in PHP are not case-sensitive.
2. PHP requires variables to be declared before being initialized.
3. Static variable is the one that are deleted when a function is completed.

## 6.1.2 Data Types

PHP has a total of eight data types which we use to construct our variables. Let us discuss each of them in detail.

### Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type. They correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so:

```
$int_var = 12345;$another_int = -12345 + 12345;
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x. For most common platforms, the largest integer is  $(2^{*31} - 1)$  (or 2,147,483,647), and the smallest (most negative) integer is  $(2^{*31} - 1)$  (or -2,147,483,647).

### Floating-Point Numbers

Floating point numbers could be written in two forms, fractional and exponential form.

- **Fractional form:** The number must have at least one digit (0-9). No comma or blanks are allowed within an integer. It must have a decimal point. It could be either positive or negative, default sign is positive.
- **Exponential form:** The exponent part is an "e" or "E" followed by an integer, which can be signed (preceded by "+" or "-").



Example:

```
<?php
$a = 5.205;
$b = 1.3e3;
$c = 6E-10; ?>
```

### Strings

Strings are sequence of characters. In PHP, a character is the same as a byte, therefore there are exactly 256 different characters possible. Long string is supported in PHP, in fact there is no practical bound to the size of strings. But PHP has no native support of Unicode.

A string literal can be specified in the following ways.

- **Single quoted:** The simple way to print a string is to enclose it in a single quotes (use the character '). If you want to print a single quote (') within a string then you must escape it with a backslash like many other language. If a backslash (\) needs to print before a single quote or at the end of the string then the backslash must occur twice. At the end of the every string a html break character has added for line breaking.



Example:

```
<?php
echo 'One line simple string'
?>
```

- **Double quoted:** When we want to print some special characters or the values of variables within a string then we enclose the string with double-quotes(") character.



Example:

```
<?php
$samt=2000;
$desc="My salary amount for this month is : ";
echo "$desc $samt
";
?>
```

We can use special characters in our code using double quotes.

Table 6.1: Special Characters

| Escape | Replaced by          |
|--------|----------------------|
| \n     | Linefeed (LF)        |
| \r     | Carriage return (CR) |
| \t     | horizontal tab (HT)  |
| \v     | vertical tab (VT)    |
| \f     | form feed (FF)       |
| \\     | Backslash            |
| \\$    | dollar sign          |
| \"     | double-quote         |

Source: <http://www.w3resource.com/php/data-types/strings.php>

## Here-Docs

It is a way to embed large pieces of text in your scripts which may include lots of double and single quotes.

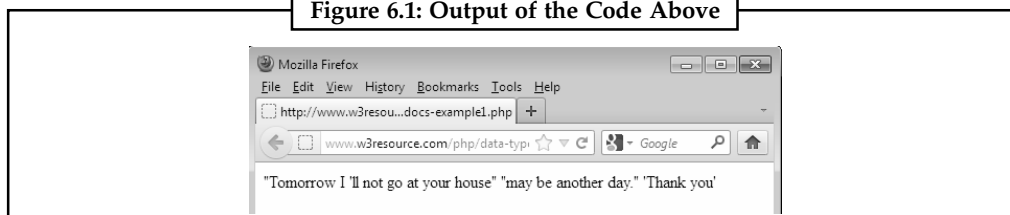


Example:

```
<?php
$mystring=<<<MYID
"Tomorrow I 'll not go at your house"
"may be another day."
'Thank you'
MYID;
echo $mystring;
?>
```

Output:

Figure 6.1: Output of the Code Above



Source: <http://www.w3resource.com/php/data-types/strings.php>

Notes



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

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

**Booleans**

Booleans is the easiest type. It can be either TRUE or FALSE. It is used in control structure like the testing portion of an if statement. To specify a Boolean literal, use the keywords **TRUE** or **FALSE**. Both are case-insensitive.



*Example:*

```
<?php
$height=100;
$width=50;
if ($width == 50)
{
echo "The width is 50";
}
?>
```

*Output:* The width is 50

**Table 6.2: List of Boolean Value Equivalents**

| Data Type      | True Value                          | False Value                                 |
|----------------|-------------------------------------|---------------------------------------------|
| Integer        | All non-zero values                 | 0                                           |
| Floating point | All non-zero values                 | 0.0                                         |
| Strings        | All other strings                   | Empty strings ("")<br>The Zero string ("0") |
| Null           | Never                               | Always                                      |
| Array          | If it contains at least one element | If it does not contain any elements         |
| Object         | Always                              | Never                                       |
| Resource       | Always                              | Never                                       |

*Source:* <http://www.w3resource.com/php/data-types/booleans.php>

To explicitly convert a value to Boolean, use the (*bool*) or (*boolean*) casts. However, in most cases the cast is unnecessary, since a value will be automatically converted if an operator, function or control structure requires a Boolean argument.



*Notes* 1 is considered **TRUE**, like any other non-zero (whether negative or positive) number.

## Null

## Notes

The special NULL value represents a variable with no value. NULL is the only possible value of type null. A variable is considered to be null if:

- it has been assigned the constant NULL.
- it has not been set to any value yet.
- it has been unset().

There is only one value of type null, and that is the case-insensitive constant NULL.



*Example:*

```
<?php
$var = NULL;
?>
```

Casting a variable to null using (*unset*) *\$var* will *not* remove the variable or unset its value. It will only return a **NULL** value.

## Resources

A resource is a special variable, holding a reference to an external resource. Resources are created and used by special functions. As resource variables hold special handlers to opened files, database connections, image canvas areas and the like, converting to a resource makes no sense. Thanks to the reference-counting system introduced with PHP 4's Zend Engine, a resource with no more references to it is detected automatically, and it is freed by the garbage collector. For this reason, it is rarely necessary to free the memory manually.



*Notes* Persistent database links are an exception to this rule. They are not destroyed by the garbage collector.

## Arrays

An array in PHP is actually an ordered map. A map is a type that associates *values* to *keys*. This type is optimized for several different uses; it can be treated as an array, list (vector), hash table (an implementation of a map), dictionary, collection, stack, queue, and probably more. As array values can be other arrays, trees and multidimensional arrays are also possible. In simpler words, an array is a collection of similar types.

*array()* construct

To create an array we use the *array()* construct as shown below:

```
array ([mixed $...])
```

“index => values”, separated by commas, define index and values. Index may be of type string or integer. When index is omitted, an integer index is automatically generated, starting at 0. If index is an integer, next generated index will be the biggest integer index + 1. Note that when two identical index are defined, the last overwrite the first.

Returns an array of the parameters. The parameters can be given an index with the => operator. Read the section on the array type for more information on what an array is.



Notes



Example:

```
a. $cars=array("Volvo", "BMW", "Toyota");
b. $age=array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
c. $cars[0]="Volvo";
 $cars[1]="BMW";
 $cars[2]="Toyota";
```

**Accessing Array Elements**

We can access an array element through its index value that can be numeric or string.



Example:

```
$employee_array[0] = "Bob";
$employee_array[1] = "Sally";
$employee_array[2] = "Charlie";
$employee_array[3] = "Clare";

echo "Two of my employees are "
. $employee_array[0] . " & " . $employee_array[1];
echo "
Two more employees of mine are "
. $employee_array[2] . " & " . $employee_array[3];
```

**Output:**

```
Two of my employees are Bob & Sally
Two more employees of mine are Charlie & Clare
```



Example:

```
$salaries["Bob"] = 2000;
$salaries["Sally"] = 4000;
$salaries["Charlie"] = 600;
$salaries["Clare"] = 0;

echo "Bob is being paid - $" . $salaries["Bob"] . "
";
echo "Sally is being paid - $" . $salaries["Sally"] . "
";
echo "Charlie is being paid - $" . $salaries["Charlie"] . "
";
echo "Clare is being paid - $" . $salaries["Clare"];
```

**Output:**

```
Bob is being paid - $2000
Sally is being paid - $4000
Charlie is being paid - $600
Clare is being paid - $0
```

## Modifying/Creating Array Elements

Notes

We can modify or create array elements using the `array()` and making use of its key value.



Example:

```
$cars=array("Volvo","BMW","Toyota"); //We created this array
$cars[1]="Ford"; //We changed the value from BMW to Ford
echo $cars[1];
```

**Output:** Ford

PHP supports a special notation, `$arr[]`, where the key is not specified. We can define values at run time.



Example:

```
$array_one = array(1, 2, 3); //Compile time values
$array_two[] = 1; //Runtime values
$array_two[] = 2;
$array_two[] = 3;
```

## Reading Array Values

As shown in examples above we can also read the array values with the key of the array.

## Accessing Nested Arrays (or Multi-Dimensional Arrays)

An array can also contain another array as a value, which in turn can hold other arrays as well. In such a way we can create two- or three-dimensional arrays. A multidimensional array is an array containing one or more arrays. In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.



Example:

```
$row_0 = array(1, 2, 3);
$row_1 = array(4, 5, 6);
$row_2 = array(7, 8, 9);
$multi = array($row_0, $row_1, $row_2);
$value = $multi[2][0];
echo("The value at row 2, column 0 is {$multi[2][0]}\n");
```

**Output:** The value at row 2, column 0 is 7.

## Traversing Arrays Using foreach

The `foreach` construct provides an easy way to iterate over arrays. `foreach` works only on arrays and objects, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. There are two ways to write `foreach` syntax:

```
foreach (array_expression as $value)
 statement
```

**Notes**

```
foreach (array_expression as $key => $value)
 statement
```

The first form loops over the array given by *array\_expression*. On each iteration, the value of the current element is assigned to *\$value* and the internal array pointer is advanced by one (so on the next iteration, you'll be looking at the next element).

The second form will additionally assign the current element's key to the *\$key* variable on each iteration.



*Notes* When *foreach* first starts executing, the internal array pointer is automatically reset to the first element of the array. This means that you do not need to call `reset()` before a *foreach* loop. As *foreach* relies on the internal array pointer changing it within the loop may lead to unexpected behaviour.

In order to be able to directly modify array elements within the loop precede *\$value* with `&`. In that case the value will be assigned by reference.



*Example:*

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value)
{
 $value = $value * 2;
} // $arr is now array(2, 4, 6, 8)
unset($value); // break the reference with the last element
?>
```

Referencing *\$value* is only possible if the iterated array can be referenced (i.e. if it is a variable). The following code won't work:

```
<?php
foreach (array(1, 2, 3, 4) as &$value)
{
 $value = $value * 2;
}
?>
```

**Traversing Arrays Using `list()` and `each()`**

We can also traverse an array using the `list()` and `each()` constructs.

- **`list()`**: PHP 5.5 added the ability to iterate over an array of arrays and unpack the nested array into loop variables by providing a `list()` as the value. **`list()`** only works on numerical arrays and assumes the numerical indices start at 0. Modification of the array during **`list()`** execution (e.g. using `list($a, $b) = $b`) results in undefined behavior. **`list()`** assigns the values starting with the right-most parameter.



Example:

```
<?php
$array = [
 [1, 2],
 [3, 4],
];
foreach ($array as list($a))
{
 echo "$a\n"; // Note that there is no $b here.
}
?>
```

**Output:**

```
1
3
```

- **reset()**: It rewinds *array*'s internal pointer to the first element and returns the value of the first array element. Returns the value of the first array element, or **FALSE** if the array is empty.



Example:

```
<?php
$array = array('step one', 'step two', 'step three', 'step four');
// by default, the pointer is on the first element
echo current($array) . "
\n"; // "step one"
// skip two steps
next($array);
next($array);
echo current($array) . "
\n"; // "step three"
// reset pointer, start again on step one
reset($array);
echo current($array) . "
\n"; // "step one"
?>
```

- **each()**: Return the current key and value pair from an array and advance the array cursor. After **each()** has executed, the array cursor will be left on the next element of the array, or past the last element if it hits the end of the array. You have to use **reset()** if you want to traverse the array again using **each**. Returns the current key and value pair from the *array*. This pair is returned in a four-element array, with the keys *0*, *1*, *key*, and *value*. Elements *0* and *key* contain the key name of the array element, and *1* and *value* contain the data. If the internal pointer for the array points past the end of the array contents, **each()** returns **FALSE**.

### Notes



Example:

```
<?php
$foo = array("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each($foo);
print_r($bar);
?>
```

\$bar now contains the following key/value pairs:

```
Array
(
 [1] => bob
 [value] => bob
 [0] => 0
 [key] => 0
)
```


```
<?php
$foo = array("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each($foo);
print_r($bar);
?>
```

\$bar now contains the following key/value pairs:

```
Array
(
 [1] => Bob
 [value] => Bob
 [0] => Robert
 [key] => Robert
)
```

### Additional Methods for Traversing Arrays

We also have two more language constructs `current()` and `next()` to make use of in arrays.



*Task* Create a small code to show the use of `current()` and `next()`

### Self Assessment

Fill in the blanks:

- 4. Floating point numbers could be written in two forms, ..... and ..... form.

5. .... is a way to embed large pieces of text in your scripts which may include lots of double and single quotes.

Notes

### 6.1.3 Operators

An operator is a special character or combination of characters that operates on variables. They can be grouped according to the number of values they take.

- Unary operators take only one value, for example ! (the logical not operator) or ++ (the increment operator).
- Binary operators take two values, such as the familiar arithmetical operators + (plus) and - (minus), and the majority of PHP operators fall into this category.
- Finally, there is a single ternary operator, ? :, which takes three values; this is usually referred to simply as the ternary or conditional operator.

PHP can only perform binary operations on two operands that have the same type. However, if the two operands have different types, PHP automatically converts one of them to the other's type.

#### Binary Operators

Binary operators are operators that deal with two arguments, both generally being either variables or constants.

**Table 6.3: Binary Operators**

| Operator | Use                         |
|----------|-----------------------------|
| +        | Addition                    |
| -        | Subtraction                 |
| *        | Multiplication              |
| /        | Division                    |
| %        | Remainder division          |
| =        | Assignment                  |
| ==       | Boolean equality comparison |
| >        | Boolean greater than        |
| <        | Boolean less than           |
| &&       | Boolean AND                 |
|          | Boolean OR                  |

Source: [http://www.msbware.com/view\\_article?article\\_id=109](http://www.msbware.com/view_article?article_id=109)

**Numeric Operators:** All the binary operators (except for the concatenation operator) work only on numeric operands. If one or both of the operands are non-numeric, they are converted to their numeric equivalents to perform the required action. They include +, -, \*, / and %.

**Concatenation Operator (.):** There are two string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side.

**Notes***Example:*

```
<?php
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"
$a = "Hello ";
$a .= "World!"; // now $a contains "Hello World!"
?>
```

**Assignment Operators**

The basic assignment operator is “=”. The left operand gets set to the value of the expression on the right. The value of an assignment expression is the value assigned. That is, the value of “ $a = 3$ ” is 3. This allows you to do some tricky things:

```
<?php
$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4.
?>
```

For arrays, assigning a value to a named key is performed using the “=>” operator. The precedence of this operator is the same as other assignment operators. In addition to the basic assignment operator, there are “combined operators” for all of the binary arithmetic, array union and string operators that allow you to use a value in an expression and then set its value to the result of that expression.

*Example:*

```
<?php
$a = 3;
$a += 5; // sets $a to 8, as if we had said: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // sets $b to "Hello There!", just like $b = $b . "There!";
?>
```



*Notes* The assignment copies the original variable to the new one (assignment by value), so changes to one will not affect the other. This may also have relevance if you need to copy something like a large array inside a tight loop.

**By-Reference Assignment Operator**

Assignment by reference is also supported in PHP. It uses the syntax, “ $\$var = \&\$othervar$ ;”. Assignment by reference means that both variables end up pointing at the same data, and nothing is copied anywhere.

*Example:*

```
<?php
$a = 3;
```

```

$b = &$a; // $b is a reference to $a
print "$a\n"; // prints 3
print "$b\n"; // prints 3
$a = 4; // change $a
print "$a\n"; // prints 4
print "$b\n"; // prints 4 as well, since $b is a reference to $a, which has
// been changed
?>

```

As of PHP 5, the new operator returns a reference automatically, so assigning the result of new by reference results in an **E\_DEPRECATED** message in PHP 5.3 and later, and an **E\_STRICT** message in earlier versions.

## Comparison Operators

Comparisons are used to check the relationship between variables and/or values. Comparison operators are used inside conditional statements and evaluate to either *true* or *false*. Here are the most important comparison operators of PHP.

Assume:  $\$x = 4$  and  $\$y = 5$ ;

**Table 6.4: Comparison Operators**

| Operator | English                  | Example      | Result |
|----------|--------------------------|--------------|--------|
| ==       | Equal To                 | $\$x == \$y$ | False  |
| !=       | Not Equal To             | $\$x != \$y$ | True   |
| <        | Less Than                | $\$x < \$y$  | True   |
| >        | Greater Than             | $\$x < \$y$  | False  |
| <=       | Less than or Equal to    | $\$x <= \$y$ | True   |
| >=       | Greater than or equal to | $\$x >= \$y$ | false  |

Source: <http://www.tizag.com/phpT/operators.php>



*Task* Find out about === and !== operators.

## Logical Operators

There are following logical operators supported by PHP language.

**Table 6.5: Logical Operators**

| Example         | Name | Result                                               |
|-----------------|------|------------------------------------------------------|
| $\$a$ and $\$b$ | And  | True if both $\$a$ and $\$b$ are true.               |
| $\$a$ or $\$b$  | Or   | True if either $\$a$ or $\$b$ is true.               |
| $\$a$ xor $\$b$ | Xor  | True if either $\$a$ or $\$b$ is true, but not both. |
| ! $\$a$         | Not  | True if $\$a$ is not true.                           |
| $\$a$ && $\$b$  | And  | True if both $\$a$ and $\$b$ are true.               |
| $\$a$    $\$b$  | Or   | True if either $\$a$ or $\$b$ is true.               |

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



Notes

### Bitwise Operators

Bitwise operators allow evaluation and manipulation of specific bits within an integer.

**Table 6.6: Bitwise Operators**

| Example                       | Name               | Result                                                                                                    |
|-------------------------------|--------------------|-----------------------------------------------------------------------------------------------------------|
| <code>\$a &amp; \$b</code>    | And                | Bits that are set in both <code>\$a</code> and <code>\$b</code> are set.                                  |
| <code>\$a   \$b</code>        | Or (inclusive or)  | Bits that are set in either <code>\$a</code> or <code>\$b</code> are set.                                 |
| <code>\$a ^ \$b</code>        | Xor (exclusive or) | Bits that are set in <code>\$a</code> or <code>\$b</code> but not both are set.                           |
| <code>~ \$a</code>            | Not                | Bits that are set in <code>\$a</code> are not set, and vice versa.                                        |
| <code>\$a &lt;&lt; \$b</code> | Shift left         | Shift the bits of <code>\$a</code> <code>\$b</code> steps to the left (each step means "multiply by two") |
| <code>\$a &gt;&gt; \$b</code> | Shift right        | Shift the bits of <code>\$a</code> <code>\$b</code> steps to the right (each step means "divide by two")  |

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

### Unary Operators

Unary operators are operators that only deal with one argument (which is generally a single variable).

### Negation Operators

They are used to negate the value of the operand before which they are applied. There are two kinds of negation operators:

- **Bitwise (~):** It causes each bit to change. In case of a numeric operand, the bitwise negation of its bitwise representation (floating-point values are first converted to integers). In case of strings, a string of equal length, in which each character is the bitwise negation of its corresponding character in the original string.
- **Logical (!):** It causes the true value to become false and false to become true.

### Increment/Decrement Operators

The increment/decrement operators do not affect Boolean values. Decrementing NULL values has no effect too, but incrementing them results in 1.

**Table 6.7: Increment/Decrement Operators**

| Example            | Name           | Effect                                                              |
|--------------------|----------------|---------------------------------------------------------------------|
| <code>++\$a</code> | Pre-increment  | Increments <code>\$a</code> by one, then returns <code>\$a</code> . |
| <code>\$a++</code> | Post-increment | Returns <code>\$a</code> , then increments <code>\$a</code> by one. |
| <code>--\$a</code> | Pre-decrement  | Decrements <code>\$a</code> by one, then returns <code>\$a</code> . |
| <code>\$a--</code> | Post-decrement | Returns <code>\$a</code> , then decrements <code>\$a</code> by one. |

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



Example:

```
<?php
$i = 'W';
for ($n=0; $n<6; $n++) {
```

```

 echo ++$i . "\n";
}

?>

```

**Output:**

```

X
Y
Z
AA
AB
AC

```

**Cast Operators**

Type casting in PHP works much as it does in C: the name of the desired type is written in parentheses before the variable which is to be cast.

```

<?php
$foo = 10; // $foo is an integer
$bar = (boolean) $foo; // $bar is a boolean
?>

```

The casts allowed are:

- (int), (integer) - cast to integer
- (bool), (boolean) - cast to boolean
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object
- (unset) - cast to NULL (PHP 5)

(binary) casting and b prefix forward support was added in PHP 5.2.1



*Notes* Tabs and spaces are allowed inside the parentheses, so the following are functionally equivalent.



*Example:*

```

<?php
$foo = 10; // $foo is an integer
$bar = (boolean) $foo; // $bar is a boolean
?>

```

## Notes

**Silence Operator**

The operator @ silences error messages during the evaluation process of an expression.

**One and Only Ternary Operator**

Another conditional operator is the “?:” (or ternary) operator. The expression  $(expr1) ? (expr2) : (expr3)$  evaluates to  $expr2$  if  $expr1$  evaluates to **TRUE**, and  $expr3$  if  $expr1$  evaluates to **FALSE**.

Since PHP 5.3, it is possible to leave out the middle part of the ternary operator. Expression  $expr1 ? : expr3$  returns  $expr1$  if  $expr1$  evaluates to **TRUE**, and  $expr3$  otherwise.



*Notes* Please note that the ternary operator is a statement, and that it doesn't evaluate to a variable, but to the result of a statement. This is important to know if you want to return a variable by reference. The statement `return $var == 42 ? $a : $b;` in a return-by-reference function will therefore not work and a warning is issued in later PHP versions.

**Self Assessment**

State whether the following statements are true or false:

6. PHP can only perform binary operations on two operands that have the same type.
7. `true++ = false` in PHP.

**6.1.4 Expressions**

An expression is any combination of functions, values, and operators that resolve to a value. As a rule of thumb, if you can use it as if it were a value, it is an expression. Let us discuss some kinds of expressions used in PHP.

- We can term constants and variable to be the simplest kinds of expressions in PHP. Constants hold values that don't get changed during the runtime of a script whereas a variable can hold the same value or the value it holds can get changed during the runtime of a program.
- A function is a block of code which can be called from any point in a script after it has been declared. It is basically a compartmentalized PHP script designed to accomplish a single task. Furthermore, code contained within functions is ignored until the function is called from another part in the script. Functions are useful because they contribute to rapid, reliable, error-reducing coding, and increase legibility by tidying up complicated code sequences.



*Example:*

```
<?php
function arithmetic_mean($a, $b)
{
 $result = ($a + $b) / 2;
 return $result;
}
```

```
//print the results of calculation
echo arithmetic_mean(4,6) ,"
";
echo aritmetica_mean(3242,524543) ,"
";
?>
```

- PHP supports four scalar value types: integer values, floating point values, string values and Boolean and two composite (non-scalar) types: arrays and objects.
- Pre-increment, which is written ‘++\$variable’, increments the value before printing the final value and Post-increment, which is written ‘\$variable++’ increments the variable after reading its value.
- Conditional expressions use <, >, <=, >= and other conditional operators to be used in PHP.
- Combined operator-assignment expressions are another type of expressions where we can club the *assignment operator* with arithmetic operators to combine an assignment with a mathematical operation and also to perform string concatenations.



*Example:*

‘\$a = \$a + 3’ can also be written as ‘\$a+=3’

## Self Assessment

Fill in the blanks:

8. .... hold values that don’t get changed during the runtime.
9. .... increments the value before printing the final value.

### 6.1.5 PHP Constants

A constant is an identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script – case-sensitive by default. By convention, constant identifiers are always uppercase. The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.



*Example:*

```
<?php
define('INCHES_PER_YARD', 36);
?>
echo INCHES_PER_YARD';
```

**Output:** 36

It can often be useful to find out if a constant is actually defined. This can be achieved using the PHP `defined()` function. The `defined()` function takes the name of the constant to be checked as an argument and returns a value of `true` or `false` to indicate whether that constant exists. PHP provides a number of built-in constants that can be of significant use to the PHP web developer. In this section we will look at some of the more useful constants available.

Notes

**Table 6.8: PHP Script and Environment Related Constants**

| Constant Name        | Description                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------|
| __LINE__             | Contains the number of the line in the current PHP file (or include file) which is being executed by the PHP pre-processor. |
| __FILE__             | Contains the name of the file or include which contains the currently executing line of PHP code.                           |
| __FUNCTION__         | Contains the name of the PHP function which is currently executing                                                          |
| __CLASS__            | Contains the class which is currently in use                                                                                |
| __METHOD__           | Contains the name of the method in the current class which is currently executing                                           |
| PHP_VERSION          | Contains the version of PHP that is executing the script                                                                    |
| PHP_OS               | Contains the name of the Operating System hosting the PHP-processor                                                         |
| PHP_EOL              | Contains the Newline character for the host OS (differs between UNIX/Linux and Window for example)                          |
| DEFAULT_INCLUDE_PATH | The default path where PHP looks for include files                                                                          |

Source: [http://www.techotopia.com/index.php/PHP\\_Constants](http://www.techotopia.com/index.php/PHP_Constants) .

These constants provide information about the script which is currently executing, and also the environment in which the PHP web server module is running. These are of particular use when debugging scripts.

PHP provides a number of useful mathematical constants that can be used to save both programming time when writing a script, and processing time when performing calculations in a script. The following table provides a list of the mathematical constants available in PHP.

**Table 6.9: PHP Mathematical Constants**

| Constant Name | Description                    |
|---------------|--------------------------------|
| M_E           | Value of e                     |
| M_EULER       | Value of Euler's constant      |
| M_LNPI        | The natural logarithm of PI    |
| M_LN2         | The natural logarithm of 2     |
| M_LN10        | The natural logarithm of 10    |
| M_LOG2E       | Value of base-2 logarithm of E |
| M_PI          | The value of PI                |
| M_PI_2        | The value of PI/2              |
| M_PI_4        | The value of PI/4              |
| M_1_PI        | The value of 1/PI              |
| M_2_PI        | The value of 2/PI              |
| M_SQRTPI      | The square root of PI          |
| M_2_SQRTPI    | The value 2/square root of PI  |
| M_SQRT2       | The square root of 2           |
| M_SQRT3       | The square root of 3           |
| M_SQRT1_2     | The square root of 1/2         |

Source: [http://www.techotopia.com/index.php/PHP\\_Constants](http://www.techotopia.com/index.php/PHP_Constants) .

## Self Assessment

## Notes

Fill in the blanks:

10. A valid constant name starts with a ..... or .....
11. We use the ..... function to find out if a constant is actually defined.

### 6.1.6 Switch Flow

The *switch* statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the *switch* statement is for.



*Notes* Note that unlike some other languages, the continue statement applies to switch and acts similar to *break*.

The *switch* statement executes line by line (actually, statement by statement). In the beginning, no code is executed. Only when a *case* statement is found with a value that matches the value of the *switch* expression does PHP begin to execute the statements. PHP continues to execute the statements until the end of the *switch* block, or the first time it sees a *break* statement. If you don't write a *break* statement at the end of a case's statement list, PHP will go on executing the statements of the following case. In a *switch* statement, the condition is evaluated only once and the result is compared to each *case* statement. In an *elseif* statement, the condition is evaluated again. If your condition is more complicated than a simple compare and/or is in a tight loop, a *switch* may be faster. A special case is the *default* case. This case matches anything that wasn't matched by the other cases. The *case* expression may be any expression that evaluates to a simple type, that is, integer or floating-point numbers and strings. Arrays or objects cannot be used here unless they are dereferenced to a simple type.



*Example:*

```
<?php
switch ($i) {
 case 0:
 echo "i equals 0";
 break;
 case 1:
 echo "i equals 1";
 break;
 case 2:
 echo "i equals 2";
 break;
 default:
 echo "i is not equal to 0, 1 or 2";
}
?>
```

## Notes

**Self Assessment**

State whether the following statements are true or false:

12. The *case* expression can't be a floating-point number.
13. If you don't write a *break* statement at the end of a case's statement list, PHP will go on executing the statements of the following case.

**6.1.7 Loop Control Structures**

Loops execute a block of code a specified number of times, or while a specified condition is true. Let us discuss each loop in detail.

**While Loops**

The while loop executes a block of code while a condition is true. Its general syntax is,

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



*Example:*

The example below first sets a variable *i* to 1 ( $\$i=1$ ).

Then, the while loop will continue to run as long as *i* is less than, or equal to 5. *i* will increase by 1 each time the loop runs:

```
<?php
$i=1;
while($i<=5)
{
 echo "The number is " . $i . "
";
 $i++;
}
?>
```

**Output:**

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

**Loop Control: Break and Continue**

*Break* ends execution of the current *for*, *foreach*, *while*, *do-while* or *switch* structure. It accepts an optional numeric argument which tells it how many nested enclosing structures are to be

broken out of. *Continue on the other hand* is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration. It accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of. The default value is *1*, thus skipping to the end of the current loop.



*Example:*

```
$i = 10;
while (-$i)
{
 if ($i == 8)
 {
 continue;
 }
 if ($i == 5)
 {
 break;
 }
 echo $i . "\n"
}
```

Output:

```
9
7
6
```

### do-while Loops

A “do while” loop is a slightly modified version of the while loop. It *always* executes its block of code at least once. This is because the conditional statement is not checked until *after* the contained code has been executed.



*Example:*

```
$cookies = 0;
do
{
 echo "Mmmm...I love cookies! *munch munch munch*";
} while ($cookies > 1);
```

**Output:**

```
Mmmm...I love cookies! *munch munch munch*
```

The code segment “Mmmm...I love cookies!” was executed even though the conditional statement was false. This is because a do-while loop first *do’s* and secondly checks the while condition.



Notes

**For Loops**

This is similar to a WHILE loop in that it continues to execute the loop until a given condition is proven to be false. Its syntax is,

```
FOR (start; conditional; increment)
{
//code to execute.
}
```



Example:

```
<?php
for ($i=1; $i <= 10; $i++)
{
print $i . " is your number ";
}
?>
```

Our variable, in this case \$i, starts at 1 and continues to print it’s value until the condition of it being less than or equal to 10 is no longer true. This statement would become invalid once \$i hit 11, and therefore the loop would stop.

Perhaps surprisingly, infinite loops can sometimes be helpful in your scripts. As infinite loops never terminate without outside influence, the most popular way to use them is to break out of the loop and/or exit the script entirely from within the loop whenever a condition is matched. You can also rely on user input to terminate the loop – for example, if you are writing a program to accept people typing in data for as long as they want, it just would not work to have the script loop 30,000 times or even 300,000,000 times. Instead, the code should loop forever, constantly accepting user input until the user ends the program by pressing Ctrl-C.



Example:

```
<?php
for (;;)
{
print "In loop!\n";
}
?>
```

**Self Assessment**

Fill in the blanks:

- 14. The while loop executes a block of code while a condition is .....
- 15. The default value for continue is .....

**6.1.8 Browser Output**

Output is not buffered by default with PHP and the contents of the script is sent to the browser in chunks as it is created. PHP’s output buffering functions buffer the output giving control over when content is sent, allowing it to be compressed and allowing a mixture of headers and content to be coded without getting those “Cannot modify header information – headers already sent” error messages.

## Browser Output Control Functions

## Notes

- flush — Flush the output buffer
- ob\_clean — Clean (erase) the output buffer
- ob\_end\_clean — Clean (erase) the output buffer and turn off output buffering
- ob\_end\_flush — Flush (send) the output buffer and turn off output buffering
- ob\_flush — Flush (send) the output buffer
- ob\_get\_clean — Get current buffer contents and delete current output buffer
- ob\_get\_contents — Return the contents of the output buffer
- ob\_get\_flush — Flush the output buffer, return it as a string and turn off output buffering
- ob\_get\_length — Return the length of the output buffer
- ob\_get\_level — Return the nesting level of the output buffering mechanism
- ob\_get\_status — Get status of output buffers
- ob\_gzhandler — ob\_start callback function to gzip output buffer
- ob\_implicit\_flush — Turn implicit flush on/off
- ob\_list\_handlers — List all output handlers in use
- ob\_start — Turn on output buffering
- output\_add\_rewrite\_var — Add URL rewriter values
- output\_reset\_rewrite\_vars — Reset URL rewriter values



## Case Study

**Enterprise Solutions using Java-PHP**

**E***ngineering Challenge:* When David Berry assumed the role of CTO at LiveProcess, he inherited version 1.0 of the LiveProcess platform, a PHP based web application consisting of eight person years of code.

As the project moved forward, several of the existing functions and new feature requirements could be implemented better as background tasks. However, PHP on Apache is a user-initiated programming environment and requires user input to run PHP. As an experienced Java developer, David Berry knew that Java could handle the background tasks through multithreading and wanted the added Java benefits of integrated security and connection pooling.

The challenge became – could we integrate PHP with Java EE or would we need to replace PHP?

***Analysis***

Rewriting the PHP application to JSP, Struts, Spring, or JSF would take too much time so we focused our analysis on making PHP work with Java. We identified two solutions: a Java-PHP bridge or Quercus.

The Java-PHP bridge would consist of Java calling a running instance of Apache/PHP via RMI, but this would be cumbersome to deploy in a production environment.

Because Quercus runs as a Java Servlet and compiles PHP into Java, it could run the application with minor modifications and would allow the application to directly access

*Contd...*

Notes

Java objects. The Quercus solution would let us easily integrate container managed security, an open-source persistence library and a scheduler library.

**Findings**

In our trial, 90% of the application immediately ran on Quercus. The last 10% required a little recoding and the release of Resin 3.1.

After the application completely passed our regression tests using Quercus, we started to enhance the LiveProcess platform PHP code with Java. The first enhancement was to use Java EE container managed security to authenticate users and determine which PHP pages they could access. We did this by implementing a custom authentication class that used the existing user tables in our application. This allowed us to remove the “isLoggedIn” check that we did at the head of every PHP page.

The second area that we focused on was connecting PHP to a Java persistence library. This allowed us to use enterprise level Java features including connection pooling and prepared statement pooling, features which are not easily done in PHP.

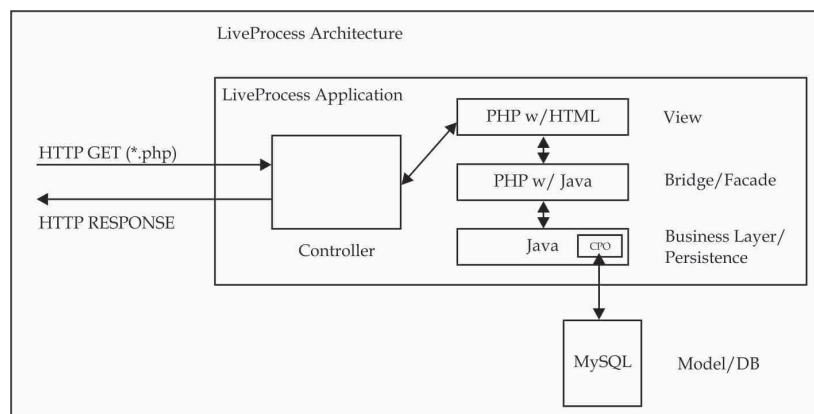
During our development process, we discovered that using object oriented PHP to develop a page template framework was superior to JSP or Struts. Our PHP template framework let us limit the web accessible PHP files to about six and the bulk of the PHP code is protected under WEB-INF by the Java EE container. The resulting PHP architecture offers significant flexibility, maintainability and security.

The benefits of Quercus Java-PHP architecture include:

- Ability to use PHP libraries
- Use of PHP5 object model for page templating
- PHP as Servlet better works than Zend solution or JSR223
- Multithreading
- Background tasks
- Event scheduling
- Java persistence
- Ability to use third party Java or PHP libraries to solve problems

**Result:** LiveProcess has chosen Quercus and Resin as their Platform of choice.

**Architecture Diagram:**



Contd...

**About LiveProcess:** LiveProcess is the leader in emergency preparedness' planning for the healthcare industry. The company, which was established in 2003, developed the first standardized software solution designed to help healthcare-related organizations prepare for and respond to emergencies. The LiveProcess platform provides a range of fully integrated tools to assist in emergency management including Hazard Vulnerability Analyses (HVAs), Incident Command System tools (ICS), Drills for Readiness & Compliance and Competency-based Training.

**About Caucho Technology:** Caucho Technology is an engineering company devoted to reliable open source and high performance Java-PHP solutions. Caucho is a Sun Microsystems licensee whose products include Resin application server, Hessian web services and Quercus Java-PHP solutions. Caucho Technology was founded in 1998 and is based in La Jolla, California.

#### Questions

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: [http://quercus.caucho.com/casestudies/Caucho\\_LiveProcess\\_casestudy.pdf](http://quercus.caucho.com/casestudies/Caucho_LiveProcess_casestudy.pdf)

## 6.2 Summary

- A variable is a means of storing a value.
- PHP has four different variable scopes: Local, Global, Parameter and Global.
- We can manage variables with `isset()`, `unset()` and `empty()` language constructs.
- PHP has a total of eight data types which we use to construct our variables: integer, float, Boolean, Strings, Array, Heredoc, Null and Resources.
- An array in PHP is actually an ordered map. A map is a type that associates values to keys.
- An operator is a special character or combination of characters that operates on variables.
- They can be grouped according to the number of values they take: unary, binary and ternary.
- A constant is an identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script.
- An expression is any combination of functions, values, and operators that resolve to a value.
- Loops execute a block of code a specified number of times, or while a specified condition is true.
- Loops can be of many kinds: for, while, do while, etc.

## 6.3 Keywords

**Array:** Array in PHP is actually an ordered map.

**Constant:** It is an identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script.

**Expression:** It is any combination of functions, values, and operators that resolve to a value.

**Notes**

**Infinite Loop:** A loop that is non-stop.

**Loops:** Execute a block of code a specified number of times, or while a specified condition is true.

**Operator:** It is a special character or combination of characters that operates on variables.

**Switch Statement:** It is similar to a series of IF statements on the same expression.

**Variable:** It is a means of storing a value.

### **6.4 Review Questions**

1. What are variables?
2. Explain the datatypes in PHP.
3. What are arrays? Why are they needed?
4. Differentiate between variable and constants in PHP.
5. Explain the various operators in PHP.
6. What are expressions?
7. Explain the use of break and continue statements.
8. Differentiate between while and do while loop.
9. Illustrate through an example the use of infinite loops.
10. Write a note about the PHP output buffering functions.

### **Answers: Self Assessment**

- |                  |                            |
|------------------|----------------------------|
| 1. False         | 2. False                   |
| 3. True          | 4. Fractional, Exponential |
| 5. Heredoc       | 6. True                    |
| 7. False         | 8. Constant                |
| 9. Pre increment | 10. Letter, Underscore     |
| 11. defined( )   | 12. False                  |
| 13. True         | 14. True                   |
| 15. One          |                            |

### **6.5 Further Readings**



*Books*

Luke Welling and Laura Thomson, PHP and MySQL Web Development.

Lynn Beighley and Michael Morrison, Head First PHP & MySQL.

Peter MacIntyre and Rasmus Lerdorf , Programming PHP, Kevin Tatroe.

Robin Nixon, Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites.

Notes



Online links

[http://docstore.mik.ua/oreilly/webprog/php/ch02\\_04.htm](http://docstore.mik.ua/oreilly/webprog/php/ch02_04.htm)

<http://www.php.net/manual/>

<http://www.tizag.com/phpT/>

<http://www.tutorialspoint.com/php/>

<http://www.w3schools.com/php/>

## Unit 7: Functions

### CONTENTS

Objectives

Introduction

7.1 User-Defined Functions

7.2 Function Scope

7.3 Returning Values by Value

7.4 Returning Values by Reference

7.5 Declaring Function Parameters

7.5.1 By Value Parameters

7.5.2 By Reference Parameters

7.6 Default Parameters

7.7 Static Variables

7.8 Arrays

7.8.1 Array ( ) Construct

7.8.2 Accessing Array Elements

7.8.3 Modifying/Creating Array Elements

7.8.4 Reading Array Values

7.8.5 Accessing Nested Arrays (or Multi-Dimensional Arrays)

7.8.6 Traversing Arrays Using Foreach

7.8.7 Traversing Arrays Using List( ) and Each( )

7.8.8 Reset( )

7.8.9 Additional Methods for Traversing Arrays

7.9 Objects

7.9.1 Creating an Object

7.9.2 Object Inheritance

7.10 Summary

7.11 Keywords

7.12 Review Questions

7.13 Further Readings

### Objectives

After studying this unit, you will be able to:

- Understand User-defined Functions
- Discuss Function Scope

- Explain Returning Values by Value
- Declaring Function Parameters
- Elaborate upon the Concept of Default Parameters
- Discuss Static Variables
- Understand the Concept of Arrays
- Discuss about Objects

## Introduction

A function is just a name we give to a block of code that can be executed whenever we need it. A function is a block of code which can be called from any point in a script after it has been declared. It is basically a compartmentalized PHP script designed to accomplish a single task. Furthermore, code contained within functions is ignored until the function is called from another part in the script. Functions are useful because they contribute to rapid, reliable, error-reducing coding, and increase legibility by tidying up complicated code sequences. It is good programming practice to use functions to modularize your code and to better provide reuse.



*Example:* You might have a company motto that you have to display at least once on every webpage. This is where you can use the same function on every page.

## 7.1 User-Defined Functions

A function will be executed by a call to the function. To create a function, the syntax is,

```
function functionName()
{
code to be executed;
}
```

PHP function guidelines:

- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)



*Example:*

A simple function that writes a name when it is called:

```
<?php
function writeName()
{
echo "Abc";
}

echo "My name is ";
writeName();
?>
```

**Output:** My name is Abc.



## Notes

**Self Assessment**

State whether the following statements are true or false:

1. A function is a block of code which can be called from any point in a script after it has been declared.
2. The function name can start with a number.

**7.2 Function Scope**

It's important to note that if you define a variable within a function, that variable is only available within that function; it cannot be referenced in another function or in the main body of your program code. This is known as a variable's scope. The scope of a variable defined within a function is local to that function. If a function needs to use a variable that is defined in the main body of the program, it must reference it using the "global" keyword, like this:



*Example:*

```
<?php function AddNumbers (){ global $sum = 2 + 2}$sum = 0AddNumbers (
)echo "2 + 2 = ".$sum?>
```

While the scope of a variable defined in a function is local to that function, a variable defined in the main body of code has a global scope. The "global" keyword tells PHP to look for a variable that has been defined outside the function. A variable defined within a PHP program script exists only while that script is running. When the script ends, the variable ceases to exist. A variable defined within a function exists only while that function is being processed; when the function ends, the variable ceases to exist.

**Self Assessment**

State whether the following statements are true or false:

3. The scope of a variable defined within a function is global for the entire program.
4. A variable defined in the main body of code has a local scope

**7.3 Returning Values by Value**

You can return a value from a function. However, a function can only return one thing at once, although that thing can be any integer, float, array, string, etc. When the function is used and finishes executing, it sort of changes from being a function name into being a value. To capture this value you can set a variable equal to the function. Something like:

```
$myVar = somefunction();
```



*Example:*

```
<?php
function mySum($numX, $numY)
{
 $total = $numX + $numY;
 return $total;
```

```

}
$myNumber = 0;
echo "Before the function, myNumber = ". $myNumber ."
";
$myNumber = mySum(3, 4); // Store the result of mySum in $myNumber
echo "After the function, myNumber = " . $myNumber ."
";
?>

```

**Output:**

Before the function, myNumber = 0

After the function, myNumber = 7

When we first print out the value of `$myNumber` it is still set to the original value of 0. However, when we set `$myNumber` equal to the function `mySum`, `$myNumber` is set equal to `mySum`'s result. In this case, the result was  $3 + 4 = 7$ , which was successfully stored into `$myNumber` and displayed in the second echo statement.

## 7.4 Returning Values by Reference

References in PHP are a means to access the same variable content by different names. They are not like C pointers; for instance, you cannot perform pointer arithmetic using them, they are not actual memory addresses, and so on. Returning by reference is useful when you want to use a function to find to which variable a reference should be bound. Do not use return-by-reference to increase performance. The engine will automatically optimize this on its own.



Example:

```

<?php
class foo
{
 public $value = 42;
 public function &getValue()
 {
 return $this->value;
 }
}
$obj = new foo;
$myValue = &$obj->getValue(); // $myValue is a reference to $obj->value,
which is 42.
$obj->value = 2;
echo $myValue; // prints the new value of $obj->value, i.e. 2.
?>

```

In this example, the property of the object returned by the `getValue` function would be set, not the copy, as it would be without using reference syntax.

Unlike parameter passing, here you have to use `&` in both places - to indicate that you want to return by reference, not a copy, and to indicate that reference binding, rather than usual assignment, should be done for `$myValue`. If you try to return a reference from a function with

**Notes**

the syntax: `return ($this->value);` this will *not* work as you are attempting to return the result of an *expression*, and not a variable, by reference. You can only return variables by reference from a function – nothing else.

**Self Assessment**

Fill in the blanks:

- 5. You can return a value from a function with the ..... keyword.
- 6. .... in PHP are a means to access the same variable content by different names.

**7.5 Declaring Function Parameters**

To add more functionality to a function, we can add parameters. A parameter is just like a variable. Parameters are specified after the function name, inside the parentheses. They can be passed via value or reference as explained below.

**7.5.1 By Value Parameters**

By default, function arguments are passed by value. If the value of the argument within the function is changed, it does not get changed outside of the function.



*Example:*

```
<?
function pass_by_value($param) {
 push_array($param, 4, 5);
}
$ar = array(1,2,3);
pass_by_value($ar);
foreach ($ar as $elem) {
 print "
$elem";
}
?>
```

Output: 123

**7.5.2 By Reference Parameters**

To allow a function to modify its arguments, they must be passed by reference. For having an argument to a function always passed by reference, prepend an ampersand (&) to the argument name in the function definition.



*Example:*

```
<?
function pass_by_reference(&$param) {
 push_array($param, 4, 5);
```

```

}
$var = array(1,2,3);
pass_by_reference($var);
foreach ($var as $elem) {
 print "
$elem";
}
?>

```

Output: 1 2 3 4 5

The array is passed as reference, meaning that the function (`pass_by_reference`) doesn't manipulate a copy of the variable passed, but the *actual* variable itself.

## Self Assessment

Fill in the blanks:

7. By default, function arguments are passed by .....
8. To allow a function to modify its arguments, they must be passed by .....

## 7.6 Default Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it. PHP also allows the use of arrays and the special type **NULL** as default values. The default value must be a constant expression, not (for example) a variable, a class member or a function call.



*Notes* When using default arguments, any defaults should be on the right side of any non-default arguments; otherwise, things will not work as expected.



*Example:*

```

<?php
function default_f($Name = "Paul") {
 return "Hi $Name!\n";
}
default_f ();
default_f ("Paul");
default_f ("Andrew");
?>

```

Output:

Hi Paul!

Hi Paul!

Hi Andrew!

## 7.7 Static Variables

A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope.



*Example:*

```
<?php
function test()
{
 $a = 0;
 echo $a;
 $a++;
}
?>
```

This function is quite useless since every time it is called it sets  $a$  to 0 and prints 0. The  $a++$  which increments the variable serves no purpose since as soon as the function exits the  $a$  variable disappears. To make a useful counting function which will not lose track of the current count, the  $a$  variable is declared static:

```
<?php
function test()
{
 static $a = 0;
 echo $a;
 $a++;
}
?>
```

Now,  $a$  is initialized only in first call of function and every time the  $test()$  function is called it will print the value of  $a$  and increment it.

Static variables also provide one way to deal with recursive functions. A recursive function is one which calls itself. Care must be taken when writing a recursive function because it is possible to make it repeat indefinitely.



*Notes* Static declarations are resolved in compile-time.

## **Self Assessment**

State whether the following statements are true or false:

9. The default value must be a variable expression.
10. Any defaults should be on the left side of any non-default arguments.
11. A static variable exists only in a local function scope.
12. A recursive function is one which calls itself.

## 7.8 Arrays

An array is a data structure that stores one or more similar type of values in a single value. For example, if you want to store 100 numbers then instead of defining 100 variables it is easy to

define an array of 100 length. There are three different kind of arrays and each array value is accessed using an ID which is called array index. Let us now discuss these types in detail.

Notes

## Numeric Arrays

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



*Example:*

Here we have used array() function to create array.

```
<?php
$numbers = array(1, 2, 3, 4, 5); // First method to create array.
foreach($numbers as $value)
{
 echo "Value is $value
";
}
$numbers[0] = "one"; // Second method to create array.
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";
foreach($numbers as $value)
{
 echo "Value is $value
";
}
?>
```

Output:


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

## Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

**Notes**

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.



*Notes* Don't keep associative array inside double quote while printing otherwise it would not return any value.



*Example:*

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

echo "Salary of mohammad is ". $salaries['mohammad'] . "
";
echo "Salary of qadir is ". $salaries['qadir'] . "
";
echo "Salary of zara is ". $salaries['zara'] . "
";
$salaries['mohammad'] = "high"; //Second method to associate create array.
$salaries['qadir'] = "medium";
$salaries['zara'] = "low";
echo "Salary of mohammad is ". $salaries['mohammad'] . "
";
echo "Salary of qadir is ". $salaries['qadir'] . "
";
echo "Salary of zara is ". $salaries['zara'] . "
";
?>
```

**Output:**

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

**Multidimensional Arrays**

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.



Example:

```
<?php
 $marks = array(
 "mohammad" => array
 (
 "physics" => 35,
 "maths" => 30,
 "chemistry" => 39
),
 "qadir" => array
 (
 "physics" => 30,
 "maths" => 32,
 "chemistry" => 29
),
 "zara" => array
 (
 "physics" => 31,
 "maths" => 22,
 "chemistry" => 39
)
);
 /* Accessing multi-dimensional array values */
 echo "Marks for mohammad in physics : " ;
 echo $marks['mohammad']['physics'] . "
";
 echo "Marks for qadir in maths : " ;
 echo $marks['qadir']['maths'] . "
";
 echo "Marks for zara in chemistry : " ;
 echo $marks['zara']['chemistry'] . "
";
?>
```

#### Output:

Marks for mohammad in physics: 35

Marks for qadir in maths: 32

Marks for zara in chemistry: 39

### 7.8.1 Array ( ) Construct

An **array** can be created by the `array()` language-construct. It takes a certain number of comma-separated *key => value* pairs.

```
array([key =>] value
 , ...
)
```



**Notes**

The key may be an integer or string and value may be any value.



*Example:*

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"];
echo $arr[12];
?>
```

Output: bar 1

## 7.8.2 Accessing Array Elements

You can work with each value in an array by referring to its key.



*Example:* To select the 2nd value in an array, we would do this: `$arrayName[1]`. Just a reminder that numbering starts at zero, so the 2nd item actually uses the number one as its key.

```
$fruit = array("Apples", "Strawberries", "Blackberries");
echo $fruit[1];
```

## 7.8.3 Modifying/Creating Array Elements

An existing array can be modified by explicitly setting values in it. This is done by assigning values to the array, specifying the key in brackets. The key can also be omitted, resulting in an empty pair of brackets (`[]`).

```
$arr[key] = value;
$arr[] = value;
```

If `$arr` doesn't exist yet, it will be created, so this is also an alternative way to create an array

To change a certain value, assign a new value to that element using its key. To remove a key/value pair, call the `unset()` function on it.



*Example:*

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56; // This is the same as $arr[13] = 56;
// at this point of the script

$arr["x"] = 42; // This adds a new element to
// the array with key "x"

unset($arr[5]); // This removes the element from the array

unset($arr); // This deletes the whole array
?>
```

## 7.8.4 Reading Array Values

Notes

An array value can be read with the `$arrayname[key]` syntax.



*Example:*

```
$cars=array("Volvo","BMW","Toyota");
$cars[0]="Volvo";
```

## 7.8.5 Accessing Nested Arrays (or Multi-Dimensional Arrays)

A multi-dimensional array is an array containing one or more arrays. In a multi-dimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.



*Example:*

```
$families = array
(
 "Griffin"=>array
 (
 "Peter",
 "Lois",
 "Megan"
),
 "Quagmire"=>array
 (
 "Glenn"
),
 "Brown"=>array
 (
 "Cleveland",
 "Loretta",
 "Junior"
)
);
```

To access an element we write,

```
echo "Is " . $families['Griffin'][2] . " a part of the Griffin family?";
```

**Output:** Is Megan a part of the Griffin family?

## 7.8.6 Traversing Arrays Using Foreach

Foreach loops in PHP can be used to iterate over the contents of an array. They are commonly used for executing the same piece of code multiple times for different data values.



*Example:*

```
<?php

$fruitColours = array(
```

**Notes**

```
"Banana" => "Yellow",
"Apple" => "Green",
"Plum" => "Purple",
);

foreach ($fruitColours as $colour)
{
 echo "$colour
\n";
}
```

?>

Output:

Yellow  
Green  
Purple

There is another way to use foreach as shown below.

Example:

```
<?php

$fruitColours = array(
 "Banana" => "Yellow",
 "Apple" => "Green",
 "Plum" => "Purple",
);

foreach ($fruitColours as $fruit => $colour)
{
 echo "$fruit is $colour
\n";
}
```

?>

**Output:**

Banana is Yellow  
Apple is Green  
Plum is Purple

### 7.8.7 Traversing Arrays Using List() and Each()

List() assigns variables as if they were an array. Its syntax is,

```
list (mixed $varname [, mixed $...])
```

Like array(), this is not really a function, but a language construct. **list()** is used to assign a list of variables in one operation. It returns the assigned array. **list()** only works on numerical arrays and assumes the numerical indices start at 0.

Each() return the current key and value pair from an array and advance the array cursor. After **each()** has executed, the array cursor will be left on the next element of the array, or past the last element if it hits the end of the array. You have to use reset() if you want to traverse the array

again using `each`. Returns the current key and value pair from the *array*. This pair is returned in a four-element array, with the keys *0*, *1*, *key*, and *value*. Elements *0* and *key* contain the key name of the array element, and *1* and *value* contain the data. If the internal pointer for the array points past the end of the array contents, `each()` returns **FALSE**.



Example:

```
<?php
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');
reset($fruit);
while (list($key, $val) = each($fruit))
{
 echo "$key => $val\n";
}
?>
```

#### Output:

a => apple

b => banana

c => cranberry

### 7.8.8 Reset( )

`reset()` rewinds *array*'s internal pointer to the first element and returns the value of the first array element. It returns the value of the first array element, or **FALSE** if the array is empty.

### 7.8.9 Additional Methods for Traversing Arrays

We can also use `current()` and `next()` language constructs while working with arrays.

### Self Assessment

Fill in the blanks:

13. Array value is accessed using an ID which is called .....
14. By default array index starts from .....

## 7.9 Objects

An object is an individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.

### 7.9.1 Creating an Object

#### Object Initialization

Once you defined your class, then you can create as many objects as you like of that class type. Following is an example of how to create object using `new` operator.

```
$physics = new Books; $maths = new Books; $chemistry = new Books;
```

**Notes**

Here we have created three objects and these objects are independent of each other and they will have their existence separately.

**Converting to Object**

When we convert an object to another object, it is not modified. Arrays are converted to an object with properties named by keys, and corresponding values. For others, a member variable named *scalar* will contain the value.



*Example:*

```
<?php
$object_one = (object) 'abc';
echo $object_one->scalar;
?>
```

**Output:** abc

**7.9.2 Object Inheritance**

Inheritance is one of the most useful tools of *Object Oriented Programming* (OOP). **Inheritance** enables you to define a base class (also called *parent class*) and create one or more classes derived from it. A class that inherits from another is said to be a subclass of it, or child class. A child class inherits and can use all public and protected properties and methods from the parent, without having to copy any code. That way, you need to write only the additional properties and methods of the child class.



*Example:*

```
class Novel extends Books{ var publisher; function setPublisher($par){ $this->publisher = $par;
} function getPublisher(){ echo $this->publisher. "
"; }}
```

Now apart from inherited functions, class Novel keeps two additional member functions.

**Self Assessment**

Fill in the blanks:

- 15. An ..... is an individual instance of the data structure defined by a class.
- 16. A class that inherits from another is said to be a ..... class.



*Case Study*

**Recursive Functions**

**W**e're going to implement a recursive function in MIPS. Recursion is one of those programming concepts that seem to scare students. One reason is that recursive functions can be difficult to trace properly without knowing something about how a stack works.

"Classic" recursion attempts to solve a "big" problem by solving smaller versions of the problem, then using the solutions to the smaller versions of the problem to solve the big problem.

*Contd...*

“Classic” recursion is a divide-and-conquer method. For example, consider merge sorting. The idea behind merge sorting is to divide an array into two halves, and sort each half. Then, you “merge” the two sorted halves to sort the entire list.

Thus, to solve the big problem of sorting an array, solve the smaller problem of sorting part of an array, and use the solution of the smaller sorted array to solve the big problem (by merging).

How do you solve the smaller version of the problem? You break that smaller problem into even smaller problems, and solve those.

Eventually, you get to the smallest sized problem, which is called the *base case*. This can be solved without using recursion.

Recursion allows you to express solutions to a problem very compactly and elegantly. This is why people like using recursion.

However, recursion can use a lot of stack, and if you’re not careful, you can overflow the stack.

For recursion to be successful, the problem needs to have a recursive substructure.

This is a fancy term that says that to solve a big problem (say of size  $N$ ), you should be able to solve a smaller problem (of smaller size) in exactly the same way, except the problem size is smaller.

As an analogy, you might have to, say, sweep the floor. Sweeping half the floor is the same as sweeping the entire floor, except you do it on a smaller area. A problem that is solved recursively generally needs to have this property.

### *It’s the Same!*

Many folks think recursive functions are implemented in some weird and unusual way in assembly language. They can’t believe it’s implemented just like any other function.

The biggest misconception is that a function calls a smaller and smaller version, reaches the base case, then quits. For example, you might call **fact(3)**, which calls **fact(2)**, which calls **fact(1)**, which finally calls **fact(0)**, the base case.

Some programmers think the function stops there, right at the base case, and then jumps back to the initial recursive call. But you know that’s not true.

Or do you? Suppose we have function **f()**, which calls function **g()**, and **g()** calls **h()**.

What happens when we’re done with **h()**? We return back to **g()**. And what happens when we’re done with **g()**? We return back to **f()**.

Recursive functions behave that way too! Thus, **fact(3)** calls **fact(2)** which calls **fact(1)** which calls **fact(0)**, the base case. I call this phase the winding of the stack.

Once **fact(0)** is done, it goes back to **fact(1)**, just like it would for non-recursive functions. When **fact(1)** is done, it goes back to **fact(2)**. When **fact(2)** is done, it goes back to **fact(3)**. When **fact(3)** is done, it goes back to whoever called **fact(3)**. I call this the “unwinding” of the stack.

During the winding of the stack, you are making progress towards the base case. Basically, you’re trying to get to the base case, solve the base case, and slowly grow your solution back as you go through the unwinding part of the recursion. Thus, winding heads to the solution of the base case, while unwinding typically grows the solution from base case back to the original call.

*Contd...*

## Notes

*An Example*

Let's solve the following recursive function, written in C. This sums all elements of an array.

```
int sum(int arr[], int size) {
 if (size == 0)
 return 0 ;
 else
 return sum(arr, size - 1) + arr[size - 1] ;
}
```

*A MIPS Translation*

We assume **arr** is in **\$a0** and **size** is in **\$a1**.

We have to decide what to save to the stack. We have two choices. Either save **size - 1**, from which we can compute **arr[ size - 1 ]**, or save **arr[ size - 1 ]**. Let's opt to save **size - 1** on the stack.

We must also save the return address, **\$ra** since there is a function call. It's usually easy to tell whether to save the return address to the stack. If there's a function call, then save it.

```
sum: addi $sp, $sp, -8 # Adjust sp
 addi $t0, $a0, -1 # Compute size - 1
 sw $t0, 0($sp) # Save size - 1 to stack
 sw $ra, 4($sp) # Save return address
 bne $a0, $zero, ELSE # branch ! (size == 0)
 li $v0, 0 # Set return value to 0
 addi $sp, $sp, 8 # Adjust sp
 jr $ra # Return

ELSE: move $a1, $t0 # update second arg
 jal sum
 lw $t0, 0($sp) # Restore size - 1 from stack
 li $t7, 4 # t7 = 4
 mult $t0, t7 # Multiple size - 1 by 4
 mflo $t1 # Put result in t1
 add $t1, $t1, $a0 # Compute & arr[size - 1]
 lw $t2, 0($t1) # t2 = arr[size - 1]
 add $v0, $v0, $t2 # retval = $v0 + arr[size - 1]
 lw $ra, 4($sp) # restore return address from stack
 addi $sp, $sp, 8 # Adjust sp
 jr $ra # Return
```

Notice that we could have tested the if-else statement right away, and not used the stack. However, by adjusting the stack right away, it makes sure that we deal with the stack. So that's why we often have the saving of registers on the stack at the beginning even if the base case might not require any adjustment of the stack.

*Contd...*

There's nothing wrong to avoid moving the stack for the base case, however.

Notice in the ELSE, we copy **size - 1**, which is in **\$t0** to **\$a0**. We didn't retrieve it from the stack. Why not? Because up to this point, we've made no subroutine call. Thus, **\$t0** still contains **size - 1**.

#### Questions

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

Notes

Source: [http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Mips/case\\_rec.html](http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Mips/case_rec.html)

## 7.10 Summary

- A function is a block of code which can be called from any point in a script after it has been declared.
- The scope of a variable defined within a function is local to that function. If a function needs to use a variable that is defined in the main body of the program, it must reference it using the "global" keyword.
- You can return a value from a function.
- References in PHP are a means to access the same variable content by different names.
- A parameter is just like a variable inside a function.
- Pass by value means that if the value of the argument within the function is changed, it does not get changed outside of the function.
- To allow a function to modify its arguments, they must be passed by reference.
- You can set a parameter to have a default value if the function's caller doesn't pass it.
- A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope.
- An array is a data structure that stores one or more similar type of values in a single value.
- An object is an individual instance of the data structure defined by a class.

## 7.11 Keywords

**Array:** It is a data structure that stores one or more similar type of values in a single value.

**Function:** It is a block of code which can be called from any point in a script after it has been declared.

**Object:** It is an individual instance of the data structure defined by a class.

**Parameter:** It is just like a variable inside a function.

**Pass by Reference:** Used to allow a function to modify its arguments.

**Pass by Value:** It means that if the value of the argument within the function is changed, it does not get changed outside of the function.

**Reference:** They are a means to access the same variable content by different names.

**Static Variable:** It exists only in a local function scope, but it does not lose its value when program execution leaves this scope.



Notes

**7.12 Review Questions**

1. What are functions? Why are they needed?
2. What do you mean by scope?
3. Differentiate between passing by value and passing by reference.
4. What are default parameters?
5. Explain the use of static variables with examples.
6. What are arrays?
7. Explain the types of arrays.
8. Write short notes on the language constructs used in an array.
9. What are objects?
10. Explain object inheritance.

**Answer: Self Assessment**

- |            |              |
|------------|--------------|
| 1. True    | 2. False     |
| 3. False   | 4. False     |
| 5. Return  | 6. Reference |
| 7. Value   | 8. Reference |
| 9. False   | 10. False    |
| 11. True   | 12. True     |
| 13. Index  | 14. Zero     |
| 15. Object | 16. Child    |

**7.13 Further Readings**



*Books*

Adam Trachtenberg, PHP Cookbook.

David Powers, PHP Solutions: Dynamic Web Design Made Easy.

Jason Lengstorf, PHP for Absolute Beginners.

Luke Welling and Laura Thomson, PHP and MySQL Web Development.



*Online links*

<http://www.freewebmasterhelp.com/>

<http://www.w3schools.com/>

[www.tutorialspoint.com/php/php\\_functions.htm](http://www.tutorialspoint.com/php/php_functions.htm) <http://www.php.net/manual/en/ref.array.php>

## Unit 8: Working with Strings, Date and Time

Notes

### CONTENTS

Objectives

Introduction

8.1 Date Handling

8.2 Retrieving Date and Time Information

8.2.1 time()

8.2.2 microtime()

8.2.3 gettimeofday()

8.2.4 getdate()

8.2.5 localtime()

8.2.6 mktime()

8.2.7 gmmktime()

8.3 Formatting Date and Time

8.4 Parsing Date Formats

8.5 Strings with PHP

8.5.1 Single Quoted

8.5.2 Double Quoted

8.5.3 Heredoc Syntax

8.5.4 Nowdoc Syntax

8.6 PHP String Handling Functions

8.7 Accessing String Offsets

8.8 `__toString()` Method

8.9 Summary

8.10 Keywords

8.11 Review Questions

8.12 Further Readings

### Objectives

After studying this unit, you will be able to:

- Understand Data Handling
- Discuss Retrieving Date and Time Information
- Explain Formatting Date and Time
- Understand Parsing Date Formats
- Explain Strings with PHP
- Discuss Accessing String Offsets
- Understand `__toString()` Method

## Introduction

Now we will cover some very important topics that will tell us about how to work with string, date and time. Let us discuss each of them in detail.

### 8.1 Date Handling

Dates are so much part of everyday life that it becomes easy to work with them without thinking. PHP also provides powerful tools for date arithmetic that make manipulating dates easy.

### 8.2 Retrieving Date and Time Information

We have multiple functions to retrieve date and time. We will discuss some of them in detail.

#### 8.2.1 time()

PHP's `time()` function gives you all the information that you need about the current date and time. It requires no arguments but returns an integer. The integer returned by `time()` represents the number of seconds elapsed since midnight GMT on January 1, 1970. This moment is known as the UNIX epoch, and the number of seconds that have elapsed since then is referred to as a time stamp.



*Example:*

```
<?phpprint time();?>
```

Output: 948316201

This is something difficult to understand. But PHP offers excellent tools to convert a time stamp into a form that humans are comfortable with.

#### 8.2.2 microtime()

`microtime()` returns the current Unix timestamp with microseconds. This function is only available on operating systems that support the `gettimeofday()` system call. If used and set to `TRUE`, `microtime()` will return a float instead of a string. By default, `microtime()` returns a string in the form "msec sec", where *sec* is the current time measured in the number of seconds since the Unix epoch (0:00:00 January 1, 1970 GMT), and *msec* is the number of microseconds that have elapsed since *sec* expressed in seconds.

#### 8.2.3 gettimeofday()

It returns the current time. By default an array is returned. If `return_float` is set, then a float is returned.

*Array keys:*

- "sec" - seconds since the Unix Epoch
- "usec" - microseconds
- "minuteswest" - minutes west of Greenwich
- "dstime" - type of dst correction



Example:

```
<?php
print_r(gettimeofday());

echo gettimeofday(true);
?>
```

### 8.2.4 getdate()

The function `getdate()` optionally accepts a time stamp and returns an associative array containing information about the date. If you omit the time stamp, it works with the current time stamp as returned by `time()`. Following table lists the elements contained in the array returned by `getdate()`.

**Table 8.1: Elements Contained in the Array Returned by Getdate()**

Key	Description	Example
Second	Second past the inutes (0-59)	20
Minutes	Minutes past the hour (0 – 59)	29
Hours	Hours of the day (0 – 23)	22
may	Day of the month (1 – 31)	11
Wday	Day of the week (0 – 6)	4
Mon	Month of the year (1 – 12)	7
Year	Year (4 digits)	1997
Yday	Day of year (0 – 365)	19
Weekday	Day of the week	Thursday
Month	Month of the year	January
0	Timestamp	948370048

Source: [http://www.tutorialspoint.com/php/php\\_date\\_and\\_time.htm](http://www.tutorialspoint.com/php/php_date_and_time.htm)



Example:

```
<?php
$date_array = getdate();
foreach ($date_array as $key => $val)
{
 print "$key = $val
";
}
$formated_date = "Today's date: ";
$formated_date .= $date_array[mday] . "/";
$formated_date .= $date_array[mon] . "/";
$formated_date .= $date_array[year];

print $formated_date;
?>
```

**Notes****Output:**

```
seconds = 27
minutes = 25
hours = 11
mday = 12
wday = 6
mon = 5
year = 2007
yday = 131
weekday = Saturday
month = May
0 = 1178994327
Today's date: 12/5/2007
```

**8.2.5 localtime()**

The **localtime()** function returns an array identical to that of the structure returned by the C function call. Its syntax is,

```
array localtime ([int $timestamp = time() [, bool $is_associative = false]])
```

**timestamp:** The optional timestamp parameter is an integer Unix timestamp that defaults to the current local time if a timestamp is not given. In other words, it defaults to the value of `time()`  
**is\_associative:** If set to `FALSE` or not supplied then the array is returned as a regular, numerically indexed array. If the argument is set to `TRUE` then **localtime()** returns an associative array containing all the different elements of the structure returned by the C function call to `localtime`. The names of the different keys of the associative array are as follows:

- "tm\_sec" - seconds, 0 to 59
- "tm\_min" - minutes, 0 to 59
- "tm\_hour" - hours, 0 to 23
- "tm\_mday" - day of the month, 1 to 31
- "tm\_mon" - month of the year, 0 (Jan) to 11 (Dec)
- "tm\_year" - years since 1900
- "tm\_wday" - day of the week, 0 (Sun) to 6 (Sat)
- "tm\_yday" - day of the year, 0 to 365
- "tm\_isdst" - is daylight savings time in effect? Positive if yes, 0 if not, negative if unknown.

**Example:**

```
<?php
$localtime = localtime();
$localtime_assoc = localtime(time(), true);
print_r($localtime);
print_r($localtime_assoc);
?>
```

**8.2.6 mktime()**

Returns the Unix timestamp corresponding to the arguments given. This timestamp is a long integer containing the number of seconds between the Unix Epoch (January 1, 1970, 00:00:00 GMT) and the time specified. Arguments may be left out in order from right to left; any arguments

thus omitted will be set to the current value according to the local date and time. It takes in the following parameters:

**hour:** The number of the hour relative to the start of the day determined by month, day and year. Negative values reference the hour before midnight of the day in question. Values greater than 23 reference the appropriate hour in the following day(s).

**minute:** The number of the minute relative to the start of the hour. Negative values reference the minute in the previous hour. Values greater than 59 reference the appropriate minute in the following hour(s).

**second:** The number of seconds relative to the start of the minute. Negative values reference the second in the previous minute. Values greater than 59 reference the appropriate second in the following minute(s).

**month:** The number of the month relative to the end of the previous year. Values 1 to 12 reference the normal calendar months of the year in question. Values less than 1 (including negative values) reference the months in the previous year in reverse order, so 0 is December, -1 is November, etc. Values greater than 12 reference the appropriate month in the following year(s).

**day:** The number of the day relative to the end of the previous month. Values 1 to 28, 29, 30 or 31 (depending upon the month) reference the normal days in the relevant month. Values less than 1 (including negative values) reference the days in the previous month, so 0 is the last day of the previous month, -1 is the day before that, etc. Values greater than the number of days in the relevant month reference the appropriate day in the following month(s).

**year:** The year

**is\_dst:** Parameters always represent a GMT date so is\_dst doesn't influence the result.

### 8.2.7 gmmktime()

Identical to mktime() except the passed parameters represents a GMT date. **gmmktime()** internally uses mktime() so only times valid in derived local time can be used. Like mktime(), arguments may be left out in order from right to left, with any omitted arguments being set to the current corresponding GMT value. The **gmmktime()** function returns a integer Unix timestamp. The parameters are same like mktime().

### Self Assessment


Fill in the blanks:

- ..... is the integer that defines the number of seconds elapsed since midnight GMT on January 1, 1970.
- Number of seconds that have elapsed since midnight GMT on January 1, 1970 is referred to as a .....
- By default, microtime() returns a string in the form .....
- gettimeofday() returns the current .....

## 8.3 Formatting Date and Time

The PHP date() function is used to format a time and/or date. It formats a timestamp to a more readable date and time.

Notes



*Notes* A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

The syntax for the `date()` function is,

```
date (format, timestamp)
```

It takes in the following two parameters,

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

The required *format* parameter in the `date()` function specifies how to format the date/time.

Here are some characters that can be used:

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)

Other characters, like `"/"`, `"."`, or `"-"` can also be inserted between the letters to add additional formatting.



*Example:*

```
<?php
echo date("Y/m/d") . "
";
echo date("Y.m.d") . "
";
echo date("Y-m-d");
?>
```

**Output:**

```
2013/05/11
2013.05.11
2013-05-11
```

The optional *timestamp* parameter in the `date()` function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used.

The `mktime()` function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1, 1970, 00:00:00 GMT) and the time specified. Its syntax is,

```
mktime(hour,minute,second,month,day,year,is_dst)
```

To go one day in the future we simply add one to the day argument of `mktime()`:



*Example:*

```
<?php
$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Tomorrow is ".date("Y/m/d", $tomorrow);
?>
```

Output: Tomorrow is 2013/05/12

Notes

Table 8.2: Date Formatting

Date Formatting		
Y	4 digit year	2005
y	2 digit year	05
F	Long month	January
M	Short month	Jan
m	Month (leading zeros)	01 to 12
n	Month	1 to 12
D	Short day name	Mon
I	Long day name	Monday
d	Day (leading zeros)	01 to 31
j	Day	1 to 31
h	12 Hour (leading zeros)	01 to 12
g	12 Hour	1 to 12
G	24 Hour	0 to 23
i	Minutes (Leading zeros)	00 to 59
s	Second (leading zeros)	00 to 59
w	Day of week <sup>1</sup>	0 to 6
z	Day of year	0 to 365
W	Week of year <sup>2</sup>	1 to 53
t	Days in month	28 to 31
a		am or pm
A		AM or PM
B	Swatch Internet Time	000 to 999
S	Ordinal Suffix	st, nd, td, th
T	Timezone of machine	GMT
Z	Timezone offset (seconds)	
O	Difference to GMT (hours)	+0200
I	Daylight saving	1 or 0
L	Leap year	1 or 0
U	Second Since Epoch <sup>3</sup>	
c	ISO 8601 (PHP 5)	
r	RFC 2822	

1. date ("w"): 0 is Sunday, 6 is Saturday.  
 2. Week that overlaps two years belongs to year that contains most days of that week. Hence week number for 1<sup>st</sup> January of a given year can return 53<sup>rd</sup> week if week belongs to previous year.  
 Date ("W", mktime (0, 0, 0, 12, 28, \$year)) always gives correct number of week in \$year  
 3. The Epoch was the 1<sup>st</sup> January 1970.

Source: <http://cheatsheets.mobi/?tag=php>



## Notes

**Self Assessment**

State whether the following statements are true or false:

5. The PHP `date()` function is used to format a date only.
6. If you do not specify a timestamp in `date()`, the current date and time will be used.

**8.4 Parsing Date Formats**

In computer science, parsing means to analyze or separate (input, for example) into more easily processed components. We can perform date parsing in PHP using the `strtotime()` function. `strtotime()` parses about any English textual date and time description into a Unix timestamp. Its syntax is,

```
int strtotime (string $time [, int $now = time()])
```

The function expects to be given a string containing an English date format and will try to parse that format into a Unix timestamp, relative to the timestamp given in *now*, or the current time if *now* is not supplied. Each parameter of this function uses the default time zone unless a time zone is specified in that parameter. Be careful not to use different time zones in each parameter unless that is intended.

Its parameters are,

*time*: A date/time string.

*now*: The timestamp which is used as a base for the calculation of relative dates.

It returns a timestamp on success, **FALSE** otherwise. Previous to PHP 5.1.0, this function would return *-1* on failure.



*Example:*

```
<?php
echo strtotime("now"), "\n";
echo strtotime("10 September 2000"), "\n";
echo strtotime("+1 day"), "\n";
echo strtotime("+1 week"), "\n";
echo strtotime("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime("next Thursday"), "\n";
echo strtotime("last Monday"), "\n";
?>
```



*Did u know?* The valid range of a timestamp is typically from Fri, 13 Dec 1901 20:45:54 UTC to Tue, 19 Jan 2038 03:14:07 UTC. (These are the dates that correspond to the minimum and maximum values for a 32-bit signed integer.) Additionally, not all platforms support negative timestamps, therefore your date range may be limited to no earlier than the Unix epoch.

**Self Assessment**

State whether the following statements are true or false:

7. `strtotime()` converts a UNIX format to an English statement.
8. `strtotime()` returns a timestamp on success.

## 8.5 Strings with PHP

A string is series of characters, where a character is the same as a byte. This means that PHP only supports a 256-character set, and hence does not offer native Unicode support.



*Notes* String can be as large as 2GB.

A string literal can be specified in four different ways that we will now discuss.

### 8.5.1 Single Quoted

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



*Example:*

```
$my_string = 'Tizag - Unlock your potential!';
echo 'Tizag - Unlock your potential!';
echo $my_string;
```

If you want to use a single-quote within the string you have to *escape* the single-quote with a backslash `\`. Like this: `'\'`.

```
echo 'Tizag - It\'s Neat!';
```



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

### 8.5.2 Double Quoted

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

**Table 8.3: Escape Sequences in PHP**

Sequence	Meaning
<code>\n</code>	linefeed (LF or 0x0A (10) in ASCII)
<code>\r</code>	carriage return (CR or 0x0D (13) in ASCII)
<code>\t</code>	horizontal tab (HT or 0x09 (9) in ASCII)
<code>\v</code>	vertical tab (VT or 0x0B (11) in ASCII) (since PHP 5.2.5)
<code>\e</code>	escape (ESC or 0x1B (27) in ASCII) (since PHP 5.4.0)
<code>\f</code>	form feed (FF or 0x0C (12) in ASCII) (since PHP 5.2.5)
<code>\\</code>	backslash
<code>\\$</code>	dollar sign
<code>\"</code>	double-quote
<code>\[0-7]{1,3}</code>	the sequence of characters matching the regular expression is a character in octal notation
<code>\x[0-9A-Fa-f]{1,2}</code>	the sequence of characters matching the regular expression is a character in hexadecimal notation

*Source:* <http://php.net/manual/en/language.types.string.php>

**Notes**

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



*Example:*

```
newline = "A newline is \n";
$return = "A carriage return is \r";
$tab = "A tab is \t";
$dollar = "A dollar sign is \$";
$doublequote = "A double-quote is \"";
```

### 8.5.3 Heredoc Syntax

In order to allow people to easily write large amounts of text from within PHP, but without the need to constantly escape things, heredoc syntax was developed. It's another way to delimit strings is the heredoc syntax: <<<. After this operator, an identifier is provided, then a newline. The string itself follows, and then the same identifier again to close the quotation. The closing identifier *must* begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore. Heredoc text behaves just like a double-quoted string, without the double quotes. This means that quotes in a heredoc do not need to be escaped, but the escape codes listed above can still be used. Variables are expanded, but the same care must be taken when expressing complex variables inside a heredoc as with strings. It is also possible to use the heredoc syntax to pass data to function arguments.



*Example:* We could use the string "EOT" (end of text) for our delimiter, meaning that we can use double quotes and single quotes freely within the body of the text - the string only ends when we type EOT.

```
<?php
$mystring = <<<EOT
 This is some PHP text.
 It is completely free
 I can use "double quotes"
 and 'single quotes',
 plus $variables too, which will
 be properly converted to their values,
 you can even type EOT, as long as it
 is not alone on a line, like this:
EOT;
?>
```

### 8.5.4 Nowdoc Syntax

Nowdocs are to single quoted strings what heredocs are to double quoted strings. A nowdoc is specified similarly to a heredoc, but *no parsing is done* inside a nowdoc. The construct is ideal for embedding PHP code or other large blocks of text without the need for escaping. A nowdoc is identified with the same <<< sequence used for heredocs, but the identifier which follows is enclosed in single quotes, e.g. <<<'EOT'. All the rules for heredoc identifiers also apply to nowdoc identifiers, especially those regarding the appearance of the closing identifier.



Example:

```
$str2 = <<<'NOWDOC_Example'
Lorem ipsum dolor sit amet,
nibh euismod tincidunt $myname .
NOWDOC_Example;

echo "
" . $str2 ;
```

## Self Assessment

Fill in the blanks:

9. PHP only supports a ..... character set.
10. A nowdoc is specified similarly to a heredoc, but *no* ..... *is done* inside a nowdoc.

## 8.6 PHP String Handling Functions

PHP supports a wide range of functions to manipulate strings. Some of them are shown in Table 8.4:

**Table 8.4: String Functions**

join()	Alias of implode()
ltrim()	Removes whitespace or other characters from the left side of a string
print()	Outputs one or more strings
printf()	Outputs a formatted string
quoted_printable_decode()	Converts a quoted-printable string to an 8-bit string
quoted_printable_encode()	Converts an 8-bit string to a quoted printable string
quotemeta()	Quotes meta characters
rtrim()	Removes whitespace or other characters from the right side of a string
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_replace()	Replaces some characters in a string (case-sensitive)
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string
strnatcasecmp()	Compares two strings using a "natural order" algorithm (case-insensitive)

Contd...

**Notes**

strnatcmp()	Compares two strings using a "natural order" algorithm (case-sensitive)
strncasecmp()	String comparison of the first n characters (case-insensitive)
strncmp()	String comparison of the first n characters (case-sensitive)
strpbrk()	Searches a string for any of a set of characters
strpos()	Returns the position of the first occurrence of a string inside another string (case-sensitive)
strrchr()	Finds the last occurrence of a string inside another string
strrev()	Reverses a string
stripos()	Finds the position of the last occurrence of a string inside another string (case-insensitive)
strrpos()	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strstr()	Finds the first occurrence of a string inside another string (case-sensitive)
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters
strtr()	Translates certain characters in a string
substr()	Returns a part of a string
substr_compare()	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
substr_count()	Counts the number of times a substring occurs in a string
substr_replace()	Replaces a part of a string with another string
trim()	Removes whitespace or other characters from both sides of a string
ucfirst()	Converts the first character of a string to uppercase
ucwords()	Converts the first character of each word in a string to uppercase
fprintf()	Writes a formatted string to a specified output stream
printf()	Outputs a formatted string
vsprintf()	Writes a formatted string to a variable
wordwrap()	Wraps a string to a given number of characters

Source: [http://www.w3schools.com/php/php\\_ref\\_string.asp](http://www.w3schools.com/php/php_ref_string.asp)

**Self Assessment**

Fill in the blanks:

11. .... removes whitespace or other characters from both sides of a string.
12. .... returns a part of a string.

**8.7 Accessing String Offsets**

We can use `$str{offset}` to extract individual characters in a string. Only valid indices can be obtained. When modifying characters, you may access offsets that don't yet exist.



Example:

```
$str_1 = "A";
$str_1{1} = "B";
$str_1{2} = "C";
print $str;
```

## 8.8 \_\_toString() Method

The `__toString()` method allows a class to decide how it will react when it is treated like a string. For example, what `echo $obj;` will print. This method must return a string, as otherwise a fatal `E_RECOVERABLE_ERROR` level error is emitted.



Example:

```
<?php
// Declare a simple class
class TestClass
{
 public $foo;

 public function __construct($foo)
 {
 $this->foo = $foo;
 }

 public function __toString()
 {
 return $this->foo;
 }
}

$class = new TestClass('Hello');
echo $class;
?>
```

**Output:** Hello



**Caution** You cannot throw an exception from within a `__toString()` method. Doing so will result in a fatal error.

## Self Assessment

State whether the following statements are true or false:

13. We can use `$str{offset}` to extract individual characters in a string.
14. The `String_to()` method allows a class to decide how it will react when it is treated like a string.
15. You can easily throw an exception from within a `__toString()` method.

## Notes



Case Study

## With or Without Web Experience BCD's WebSmart Helps IBM i Companies get to the Web with PHP

PHP has taken the Web by storm and IBM i shops are quickly getting swept up in the surge. Two companies that recently went live with IBM i PHP applications are Midwest Fuel Injection (MWFI) and American Foods Groups (AFG). Interestingly, the technicians responsible for PHP development at each company have very different programming backgrounds. MWFI's Tony Kudrys has lots of web experience but only basic IBM i knowledge, while AFG's Warren Schultz is a skilled RPG programmer with incidental exposure to web-based technologies. Ultimately, both men created professional PHP applications. What they have in common is that they both used WebSmart PHP, Business Computer Design's ([www.bcdsoftware.com](http://www.bcdsoftware.com)) rapid web application development tool.

### *PHP on IBM i?*

The use of PHP is definitely on the rise. According to Nexen.net, an organization that tracks PHP's proliferation across the web, 33% of all domains on the Internet are running PHP apps, and growth is expected to continue. Multi-platform support, ease of use, a three to four-times performance advantages over JSP, and a huge support community for this open source web programming language are a few reasons that PHP has grown so quickly. PHP is also gaining ground because it's being supported on a wider range of platforms. PHP applications running on Zend Core for IBM i gives programmers more opportunities to create innovative Web-based systems that satisfy evolving business requirements while leveraging existing investments in technology and talent. With the right tools, PHP programs can be easily integrated with existing RPG and COBOL applications and data residing in IBM DB2 for i. Many IBM i shops are considering PHP or have firm project plans that include it. One person on the front line of IBM i PHP development, Marcel Sarrasin, Technical Sales Manager from Business Computer Design (BCD), believes PHP has passed the tipping point across the IBM i landscape and notes a dramatic increase in the number of registrations for BCD's webinars on the PHP edition of WebSmart. "Over 500 people signed up for our PHP webinars in the past few months," says Sarrasin. "About 70% of those people have no prior PHP experience but they see the potential in PHP and they immediately see the value in WebSmart PHP because it generates the base PHP and HTML code for them and handles the connection to their database and to their IBM i." BCD launched WebSmart PHP in September, 2007.

### *The Diesel Store*

While many of us have plans for PHP, others such as Midwest Fuel Injection have already deployed new applications based on PHP. MWFI specializes in maintaining diesel engines and houses an extensive inventory of parts. In addition to using these parts in its repair business, MWFI also sells them through what is currently the top web store for such items, [www.thedieselstore.com](http://www.thedieselstore.com).



Roughly one year ago MWFI's president Ben Seidel authorized the construction of a dynamic website that would offer shoppers Web access to detailed product descriptions, inventory availability and pricing information. Their plans also included broad search and shopping cart capabilities, credit card transaction clearing and UPS shipment tracking.

Contd...

“We had a static website and spent a lot of time keying in telephone orders. We were quickly adding hundreds of new parts to our catalog and needed to offer customers an easy-to-use online parts store that would allow them to find what they needed very quickly,” says MWFI’s Tony Kudrys, senior system engineer and web developer. MWFI manages its entire repair and parts operation with a niche application from Computer Central in Arlington, Texas. This package runs on a server with the IBM i OS housed at MWFI’s main facility in Bolingbrook, Illinois. Kudrys and one web designer handle all system administrator duties; help desk tickets and web application development. Paradoxically, neither Kudrys nor his web designer, Steve Heinrich have a significant amount of experience with IBM i. Kudrys says they considered building their e-commerce site with Microsoft .NET ASP pages and transferring the data over to the IBM i server for processing and invoicing. This approach seemed too convoluted so they evaluated an e-commerce system proposed by Sirius Computer Solutions that required the acquisition of \$250,000 in new hardware. Then, acting on a tip from their application software vendor, Computer Central, Kudrys called BCD and asked about WebSmart PHP. “The license fees and professional services costs were well within the budget limits we set for this project and we were able to use our existing hardware,” according to Kudrys. “We also liked the simplicity of PHP and how easily it integrated with existing RPG applications and the database.” Kudrys estimates his total investment in WebSmart PHP and a couple weeks of consulting time at \$18,000. After three days of training, he and MWFI’s web designer started customizing the templates included with WebSmart PHP to match their organizations look and feel for the front end, and then started developing the catalog and shopping cart on the backend. At the same time, BCD’s professional services team built the check out process. Says Kudrys, “While WebSmart PHP gives you templates that help you create applications, we wanted to build the user interface from scratch and it allowed us to do that. We wanted it to look a certain way. BCD built the sign-in, shipping, billing and credit card submission applications. Their part does a silent order post to the credit card company using web services and once the approval response comes back they send the information to the IBM i to be processed and invoiced.” While information on pricing and availability is being pulled from the IBM i server, MWFI’s PHP applications acquire the full product description and product image from a MySQL database that resides on a separate PC. “The product descriptions on the main server are only about 20 characters long and we needed to display longer narratives for each SKU,” says Kudrys. This is a classic case of leveraging legacy information on two platforms Page two Business Computer Design, Intl, Inc. [www.bcdsoftware.com](http://www.bcdsoftware.com) transparently, one of the significant benefits of using web services and PHP strategically. Most of the vendors in the online diesel engine parts business sell the same 600 SKUs. MWFI’s intention is to offer 700,000 different SKUs and to be a single source for everything related to diesel engine repair, maintenance and modification. Adds Kudrys, “We will definitely be able to do this with the way this site is built.” BCD’s PHP development solution provided many of the tools that we needed and allowed us to use our existing computing resources and talent. And there’s a big difference between the \$18,000 price tag for this solution and the other option for \$250,000.” Says Kudrys.

#### *American Foods Group*

American Foods Group, LLC ranks among the top three privately held meat processors in the U.S. and distributes its products through retail outlets, foodservice vendors and convenience stores. Headquartered in Green Bay, Wisconsin, AFG employees over 4,000 people. AFG uses PRISM ERP running on the IBM i operating system. In AFG’s case, systems analyst Warren Schultz used WebSmart PHP to develop



*Contd...*



Notes

several web applications to simplify file maintenance. Rather than creating custom templates like MWFI's Kudrys, Schultz used the standard program templates included with WebSmart. A key objective of Schultz's is to pass current pricing information to a government server, thus fulfilling a statutory requirement. Since pricing values change weekly, the accounting staff can adjust these values in a browser and have their reports up-to-date in real time as they are automatically transmitted to the government.

*Overcoming the Differences between RPG and PHP*

With 18 years of experience on computers running IBM i OS and its predecessors, Schultz says that, like most developers with his background, his Web skills were somewhat limited. After attending a WebSmart PHP training seminar and reading other materials relating to PHP development he says, "I was able to get off and running in PHP development fairly quickly. The templates and the training were both very helpful."

Schultz also used BCD's browser-based report generator, Clover, and its portal, Nexus, to build a secure system where end users can do quick queries and view reports in a browser in a number of file formats, including Excel spreadsheets. He also developed Inquiry and Lookup screens with both Clover and WebSmart PHP that allow users to filter searches by specific values such as order number or dates.

"In terms of how these applications are accessed, I've set them up as folders or drop down menus within our Nexus Portal," says Schultz. "As long as users have a valid System i user ID they can log in to the portal and access these options based on their profile. In a few instances where users do not have a valid user profile I can grant them access within Nexus," he says.

"Over 500 people signed up for our PHP webinars in the past few months. About 70% of those people have no prior PHP experience but they see the potential in PHP and they immediately see the value in WebSmart PHP because it generates the base PHP and HTML code for them and handles the connection to their database and to their IBM i." **Marcel Sarrasin, BCD**

The Nexus portal also serves a number of other purposes. Besides providing shortcuts for these applications it's also a convenient repository for documents. Users can just click on a link and download a document rather than relying on a coworker to find it for them in a network file folder and mail it. "What I like about Nexus, among other things, is that while it was initially set up with the help of BCD's technical support staff, it's not that difficult to maintain, especially when it comes to adding new users or adding menu options, setting up groups, adding people to those groups, and using those groups to limit who has authority to what options. It's actually very easy to use."

Both Kudrys, a technician with extensive web development knowledge but only basic IBM i skills, and Schultz, a seasoned IBM i veteran with limited web development experience, were able to solve strategic and tactical business problems with the same tool, WebSmart PHP.

Kudrys and Schultz both see the benefits of these new web-based applications. Kudrys says MWFI's online shopping and order entry application was an instant success with his company's customers, and cut down considerably on personnel costs, while Schultz sees a dramatic reduction in the number of calls made to his office for current reports and compliance documents.

10,000+ installations 30,000+ products sold 40 Industry Awards 950 York Road, Hinsdale, IL 60521 (630) 986-0800 Fax: (630) 986-0926 www.BCDsoftware.com sales@bcdsoftware.com TMs mentioned are those of BCDII or of their respective owners © MMIX BCDII Page four Business Computer Design, Intl, Inc. www.bcdsoftware.com

Contd...

**Questions**

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

**Notes**

**Source:** <http://www.bcdsoftware.com/iseriess400solutions/websmart/productinfo/showcase/casestudies/thedieselstore/>

**8.9 Summary**

- PHP also provides powerful tools for date arithmetic that make manipulating dates easy.
- PHP's `time()` function gives you all the information that you need about the current date and time.
- UNIX Epoch is the integer that defines the number of seconds elapsed since midnight GMT on January 1, 1970.
- Number of seconds that have elapsed since midnight GMT on January 1, 1970 is referred to as a timestamp.
- `microtime()` returns the current Unix timestamp with microseconds.
- `gettimeofday()` returns the current time.
- The function `getdate()` optionally accepts a time stamp and returns an associative array containing information about the date.
- The PHP `date()` function is used to format a time and/or date. It formats a timestamp to a more readable date and time.
- `strtotime()` parses about any English textual date and time description into a Unix timestamp.
- A string is series of characters, where a character is the same as a byte.
- PHP supports a wide range of functions to manipulate strings.
- We can use `$str{offset}` to extract individual characters in a string.
- The `__toString()` method allows a class to decide how it will react when it is treated like a string.

**8.10 Keywords**

**`date()`:** It is used to format a time and/or date. It formats a timestamp to a more readable date and time.

**`getdate()`:** It optionally accepts a time stamp and returns an associative array containing information about the date.

**`gettimeofday()`:** It returns the current time.

**`microtime()`:** It returns the current Unix timestamp with microseconds.

**`strtotime()`:** It parses about any English textual date and time description into a Unix timestamp.

**`time()`:** It gives you all the information that you need about the current date and time.

***Timestamp*:** Number of seconds that have elapsed since midnight GMT on January 1, 1970.

***UNIX Epoch*:** It is the integer that defines the number of seconds elapsed since midnight GMT on January 1, 1970.

Notes

### 8.11 Review Questions

1. Write a short note on time() function in PHP.
2. Why do we use microtime() function in PHP?
3. Explain date() function in PHP.
4. Why do we need to format date and time?
5. What is date parsing?
6. What are strings?
7. Explain the four kinds of string handling.
8. Write a short note on all PHP String Handling Functions.
9. How do we access a string offset?
10. Explain the use of toString( ) Method in PHP.

### **Answers: Self Assessment**

#### *Fill in the blanks*

- |               |              |
|---------------|--------------|
| 1. UNIX Epoch | 2. Timestamp |
| 3. "msec,sec" | 4. Time      |
| 5. False      | 6. True      |
| 7. False      | 8. True      |
| 9. 256        | 10. Parsing  |
| 11. trim()    | 12. substr() |
| 13. True      | 14. False    |
| 15. False     |              |

### 8.12 Further Readings



#### *Books*

Adam Trachtenberg, PHP Cookbook.

David Powers, PHP Solutions: Dynamic Web Design Made Easy.

Jason Lengstorf, PHP for Absolute Beginners.

Luke Welling and Laura Thomson, PHP and MySQL Web Development.



#### *Online links*

<http://codingrecipes.com/php-date-time-handling-made-easy-a-beginners-guide-to-php-time-and-date-handling>

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

<http://www.phpfreaks.com/tutorial/>

[http://www.tutorialspoint.com/php/php\\_and\\_mysql.htm](http://www.tutorialspoint.com/php/php_and_mysql.htm)

## Unit 9: Working with Forms

Notes

### CONTENTS

Objectives

Introduction

- 9.1 Creating a Simple Input Form
- 9.2 Creating the Form
- 9.3 Accessing Form Input with User-Defined Arrays
- 9.4 Combining HTML and PHP Code on a Single Page
- 9.5 Using Hidden Fields to Save State
- 9.6 Redirecting the User
- 9.7 Sending Mail on Form Submission
  - 9.7.1 System Configuration for the Mail() Function
  - 9.7.2 Creating the Script to Send the Mail
- 9.8 Working with File Uploads
- 9.9 Creating the File Upload Form
- 9.10 Summary
- 9.11 Keywords
- 9.12 Review Questions
- 9.13 Further Readings

### Objectives

After studying this unit, you will be able to:

- Understand Input Form
- Discuss about Creating of a Form
- Elaborate upon the Use of Arrays to Access Form Input
- Discuss How to Combine HTML and PHP Code on a Single Page
- Explain the Concept of Hidden Fields
- Understand User Redirection
- Discuss Sending Mail
- Elaborate upon the Concept of File Uploads
- Explain Creation of File Upload Form

### Introduction

One of the most powerful features of PHP is the way it handles HTML forms. The basic concept that is important to understand is that any form element will automatically be available to your PHP scripts.

## 9.1 Creating a Simple Input Form

Let us first create a simple HTML form.

```
<html>
<body>
<form action="myform.php" method="post">
<p>Your Name: <input type="text" name="yourname" />

E-mail: <input type="text" name="email" /></p>

<p>Do you like this website?
<input type="radio" name="likeit" value="Yes" checked="checked" /> Yes
<input type="radio" name="likeit" value="No" /> No
<input type="radio" name="likeit" value="Not sure" /> Not sure</p>

<p>Your comments:

<textarea name="comments" rows="10" cols="40"></textarea></p>

<p><input type="submit" value="Send it!"></p>
</form>
</body>
</html>
```

This is a simple HTML form with two input fields, one radio box group and a text area for comments.

### **Self Assessment**

State whether the following statements are true or false:

1. The action attribute defines the target when the user clicks submit.
2. There is no difference between a textbox and textarea.

## 9.2 Creating the Form

Let's say we save the above code in a file called "test.html". When submitted data is sent to the "myform.php" file using **POST** HTTP method. All variables passed to the current script via the HTTP POST method are stored in associative array **\$\_POST**. In other words, in PHP you can access data from each field using **\$\_POST['NAME']**, where **NAME** is the actual field name. If you submit the form above you would have access to a number of **\$\_POST** array values inside the myform.php file:

Variable Holds value of

<code>\$_POST['yourname']</code>	text field "yourname"
<code>\$_POST['email']</code>	text field "email"
<code>\$_POST['likeit']</code>	selected radio box group "likeit"
<code>\$_POST['comments']</code>	textarea "comments"

With `register_globals` activated all form data is automatically stored in variable `$name` (where `name` is field name, for example `$yourname` or `$email`). Now, if you wanted to display submitted data you could simply echo all the variables as shown below:

```
<html>
<body>
Your name is: <?php echo $_POST['yourname']; ?>

Your email: <?php echo $_POST['email']; ?>

Do you like this website? <?php echo $_POST['likeit']; ?>

Comments:

<?php echo $_POST['comments']; ?>
</body>
</html>
```

If you saved this code in a file called “myform.php”, filled the fields in the test.html form and hit the Submit button, the myform.php output would look something like this:

```
Your name is: John Doe
Your email: john@doe.com
Do you like this website? Yes
Comments:
This is my comment...
```

## Self Assessment

State whether the following statements are true or false:

3. POST is used as a value in the action attribute of a form.
4. All variables passed to the current script via the HTTP POST method are stored in associative array `$_POST`

## 9.3 Accessing Form Input with User-Defined Arrays

The examples so far enable us to gather information from HTML elements that submit a single value per element name. This leaves us with a problem when working with SELECT elements. These elements make it possible for the user to choose multiple items. If we name the SELECT element with a plain name

```
<select name="products" multiple>
```

The script that receives this data will only have access to a single value corresponding to this name. We can change this behavior by renaming any elements of this kind so that its name ends with an empty set of square brackets. Given below is an HTML form with a select element.

```
1: <html>
2: <head>
3: <title> An HTML form including a SELECT element</title>
4: </head>
```

**Notes**

```
5: <body>
6: <form action="code2.php" method="POST">
7: <input type="text" name="user">
8:

9: <textarea name="address" rows="5" cols="40">
10: </textarea>
11:

12: <select name="products[]" multiple>
13: <option>Sonic Screwdriver
14: <option>Tricorder
15: <option>ORAC AI
16: <option>HAL 2000
17: </select>
18:

19: <input type="submit" value="hit it!">
20: </form>
21: </body>
22: </html>
```

In the script that processes the form input, we now find that input from the “products[]” form element created on line 12 will be available in an array called \$products. Products[] is a select element, and we offer the user multiple choices using the option elements on lines 13 to 16.

Now lets see how do we read input from the form in the code above.

```
1: <html>
2: <head>
3: <title> Reading input from the form in Listing 9.4</title>
4: </head>
5: <body>
6: <?php
7: print "Welcome $user<p>\n\n";
8: print "Your address is:<p>\n\n$address<p>\n\n";
9: print "Your product choices are:<p>\n\n";
10: if (! empty($products)) {
11: print "\n\n";
12: foreach ($products as $value) {
13: print "$value
\n";
14: }
15: print "";
16: }
17: ?>
18: </body>
19: </html>
```

## Notes

On line 7 we access the \$user variable, which is derived from the user form element. On line 10 we test for the \$products variable. If it is present we loop through it on line 12, outputting each choice to browser on line 13.

Although this technique is particularly useful with the SELECT element, it will in fact work with any form element at all. By giving a number of check boxes the same name, for example, you can allow a user to choose many values within a single field name. As long as the name you choose ends with empty square brackets, PHP compiles the user input for this field into an array. We can replace the SELECT element from lines 12-17 in Listing 9.4 with a series of check boxes to achieve exactly the same effect:

```
<input type="checkbox" name="products[]" value="Sonic Screwdriver">Sonic
Screwdriver

<input type="checkbox" name="products[]" value="Tricorder">Tricorder

<input type="checkbox" name="products[]" value="ORAC AI">ORAC AI

<input type="checkbox" name="products[]" value="HAL 2000">HAL 2000

```

In fact, we are not limited to numerically indexed arrays. We can place form input data into associative arrays, and even into multidimensional arrays. To keep our script's data neat, for example, we might wish to place all form input into an associative array called \$form. We can do this very simply by constructing our form field names as if they were elements in an associative array (once again, omitting the dollar sign).

```
<input type="text" name="form[user]">

<textarea name="form[address]" rows="5" cols="40">
</textarea>
```

Once submitted we will then be able access 'user' and 'address' as elements in the \$form array.

```
print $form[user];
```

To construct a multidimensional array, we can simply extend the associative array naming convention to include another level.

```
<input type="checkbox" name="form[products][]" value="Sonic Screwdriver">Sonic
Screwdriver

<input type="checkbox" name="form[products][]"
value="Tricorder">Tricorder

<input type="checkbox" name="form[products][]" value="ORAC AI">ORAC
AI

<input type="checkbox" name="form[products][]" value="HAL 2000">HAL
2000

```

When submitted, the \$form[products] element should contain a numerically indexed array, populated according to the checkboxes clicked by the user.

## Self Assessment

Fill in the blanks:

- ..... elements make it possible for the user to choose multiple items.
- As long as the name you choose ends with empty ....., PHP compiles the user input for this field into an array.



## 9.4 Combining HTML and PHP Code on a Single Page

When building a complex page, at some point you will be faced with the need to combine PHP and HTML to achieve your needed results. At first point, this can seem complicated, since PHP and HTML are two separate languages, but this is not the case. PHP is designed to interact with HTML and PHP scripts can be included in an HTML page without a problem. In an HTML page, PHP code is enclosed within special PHP tags. When a visitor opens the page, the server processes the PHP code and then sends the output (not the PHP code itself) to the visitor's browser. Actually it is quite simple to integrate HTML and PHP. A PHP script can be treated as an HTML page, with bits of PHP inserted here and there. Anything in a PHP script that is not contained within `<?php ?>` tags is ignored by the PHP compiler and passed directly to the web browser.



*Example:*

```
<html>
<head></head>
<body >
Hello, today is <?php echo date('l, F jS, Y'); ?>.
</body>
</html>
```

The code above is simply HTML, with just a bit of PHP that prints out today's date using the built-in date function. As mentioned above, all of the plain HTML in the code above will be ignored by the PHP compiler and passed through to the web browser untouched.

### Self Assessment

Fill in the blanks:

7. In an HTML page, PHP code is enclosed within special ..... tags.
8. Anything in a PHP script that is not contained within ..... tags is ignored by the PHP compiler.

## 9.5 Using Hidden Fields to Save State

Sometimes the script uses the form data to create another form, this form will be submitted to another PHP script, some of the first form data is needed in the final script. To do so, you can use hidden fields.



*Example:*

```
<html>
<title>Page1</title>
<body>
<form method="POST" action="process1.php">
Name: <input type="text" name="name">
<p>
Sex: <input type="radio" name="sex" value="M"> Male
<input type="radio" name="sex" value="F"> Female
<input type="submit" name="submit" value="Submit">
</form>
</body>
</html>
```

This HTML document has two form fields, the text field “name”, and the radio button “sex”, the data is entered by user, and then submitted to the PHP script “process1.php”, here is the code for “process1.php”:

```
<html>
<title>Page2</title>
<body>
<form method="POST" action="process2.php">
<input type="hidden" name="form_name" value="<?php print $_POST['name']
?>">
Gift:
<p>
<?php
if($_POST['sex'] == 'W') {
print "<input type='radio' name='gift' value='Bracelet'>Bracelet
\n
print "<input type='radio' name='gift' value='Earrings'>Earrings
\n
} else {
print "<input type='radio' name='gift' value='Watch'>Watch
\n
print "<input type='radio' name='gift' value='Tie'>Tie
\n
}
?>
<input type=" submit" name=" submit" value=" Submit">
</form>
</body>
</html>
```

“process1.php” will create another document that has two form elements, the hidden field “form\_name”, the value of it is the value of the “name” text field of the html document, the other form element is the radio button of the gifts to choose from, the code of these radio buttons depend on the value of the “sex” text field of the html file, after the PHP file is processed, sent to the user, filled by user and submitted, here is the code for “process2.php”:

```
<html>
<title>Process2.php</title>
<body>
Name: <?php print $_POST['form_name'] ?>

Gift: <?php print $_POST['gift'] ?>
</body>
</html>
```

“process2.php” printed user selections, note that the user name that’s displayed here is the name sent from the hidden field “form\_name” of “process1.php”.

## Self Assessment

Fill in the blanks:

9. Hidden fields are used to save .....
10. The hidden fields are created using the ..... value in the type attribute of input tag.

## 9.6 Redirecting the User

You can use a simple PHP script to redirect a user from the page they entered to a different web page. We can do this via the header() function. Its syntax is,

**Notes**

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

**header()** is used to send a raw HTTP header. **header()** must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP. It is a very common error to read code with include, or require, functions, or another file access function, and have spaces or empty lines that are output before **header()** is called. The same problem exists when using a single PHP/HTML file. Header() return no value.



*Example:*

```
<html>
<?php
/* This will give an error. Note the output
 * above, which is before the header() call */
header('Location: http://www.example.com/');
?>
```

Header() takes the following parameters:

- **string:** The header string.

There are two special-case header calls. The first is a header that starts with the string “HTTP/” (case is not significant), which will be used to figure out the HTTP status code to send. For example, if you have configured Apache to use a PHP script to handle requests for missing files (using the *Error Document* directive), you may want to make sure that your script generates the proper status code.

```
<?php
header("HTTP/1.0 404 Not Found");
?>
```

For FastCGI you must use the following for a 404 response:

```
<?php
header("Status: 404 Not Found");
?>
```

The second special case is the “Location:” header. Not only does it send this header back to the browser, but it also returns a *REDIRECT* (302) status code to the browser unless the 201 or a 3xx status code has already been set.

```
<?php
header("Location: http://www.example.com/"); /* Redirect browser */
/* Make sure that code below does not get executed when we redirect.
 */
exit;
?>
```

- **replace:** The optional *replace* parameter indicates whether the header should replace a previous similar header, or add a second header of the same type. By default it will replace, but if you pass in **FALSE** as the second argument you can force multiple headers of the same type. For example:

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

- **http\_response\_code:** Forces the HTTP response code to the specified value.



*Note* That this parameter only has an effect if the *string* is not empty.

Notes

The HTTP status header line will always be the first sent to the client, regardless of the actual `header()` call being the first or not. The status may be overridden by calling `header()` with a new status line at any time unless the HTTP headers have already been sent.

## Self Assessment

State whether the following statements are true or false:

11. To redirect a user we use the `header()` function.
12. `header()` must not be called before any actual output is sent.
13. The HTTP status header line will always be the first sent to the client.

## 9.7 Sending Mail on Form Submission

We can easily send mail through our web page using the `mail()` function in PHP.

### 9.7.1 System Configuration for the Mail() Function

Before using the `mail()` function we need to make sure that our system fulfills all the requirements needed. For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the `php.ini` file. The `php.ini` file is where you configure your PHP installation. This is the file you need to edit in order to configure PHP to send mail.

You need to ensure that the `php.ini` file contains details of the mail server that should be used whenever your application sends mail.

To check/change your PHP mail configuration:

1. Open your `php.ini` file (if you don't know where this is, see below)
2. Search for the line that reads `[mail function]`
3. Add/change the details of your mail server. This could be a local mail server or the mail server of your ISP.
4. Save/close the `php.ini` file
5. Restart your web server

Here's an example of what the mail settings could look like when you first open the `php.ini` file:

```
[mail function]
; For Win32 only.
SMTP = localhost
smtp_port = 25
; For Win32 only.
;sendmail_from = me@example.com
; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
;sendmail_path =
```

**Notes**

If you're using a UNIX based system, you will need to change the line that reads `;sendmail_path =`. You will also need to remove the semicolon from the start of this line (semicolons indicate that the line is a comment). For example, `sendmail_path = /usr/sbin/sendmail`.

If you're using a Windows system, you should change the line that reads `SMTP = localhost` to include your mail server (or your ISP's mail server). You could leave it at `localhost` if you're using your own local SMTP server. If you aren't using your own local SMTP server, you will need to enter a mail server that you have access to (such as your ISP's mail server). For example, `SMTP = mail.earthlink.net`.

You should also set a default "From" email address by changing the line that reads `;sendmail_from = me@example.com`. For example, `sendmail_from = you@earthlink.net`.

Mail configuration options:

**Table 9.1: Configuration Options**

Name	Default	Description	Changeable
SMTP	"localhost"	Windows only: The DNS name or IP address of the SMTP server	PHP_INI_ALL
smtp_post	"25"	Windows only: The SMTP port number. Available since PHP 4.3	PHP_INI_ALL
sendmail_from	NULL	Windows only: Specifies the "form" address to be used in email sent from PHP	PHP_INI_ALL
sendmail_path	NULL	Unix systems only: Specifies where the sendmail program can be found (usually <code>/usr/sbin/sendmail</code> or <code>/usr/lib/sendmail</code> )	PHP_INI_System

Source: [http://www.w3schools.com/php/php\\_ref\\_mail.asp](http://www.w3schools.com/php/php_ref_mail.asp)

### 9.7.2 Creating the Script to Send the Mail

The simplest way to send an email with PHP is to send a text email.

In the example below we first declare the variables (`$to`, `$subject`, `$message`, `$from`, `$headers`), then we use the variables in the `mail()` function to send an email:

```
<?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple email message.";
$from = "someone@example.com";
$headers = "From:" . $from;
mail($to,$subject,$message,$headers);
echo "Mail Sent.";
?>
```

With PHP, you can create a feedback-form on your website. The example below sends a text message to a specified email address:

```
<html>
<body>

<?php
```

```

if (isset($_REQUEST['email']))
//if "email" is filled out, send email
{
//send email
$email = $_REQUEST['email'] ;
$subject = $_REQUEST['subject'] ;
$message = $_REQUEST['message'] ;
mail("someone@example.com", $subject,
$message, "From:" . $email);
echo "Thank you for using our mail form";
}
else
//if "email" is not filled out, display the form
{
echo "<form method='post' action='mailform.php'>
Email: <input name='email' type='text'>

Subject: <input name='subject' type='text'>

Message:

<textarea name='message' rows='15' cols='40'>
</textarea>

<input type='submit'>
</form>";
}
?>

</body>
</html>

```

This is how the example above works:

- First, check if the email input field is filled out
- If it is not set (like when the page is first visited); output the HTML form
- If it is set (after the form is filled out); send the email from the form
- When submit is pressed after the form is filled out, the page reloads, sees that the email input is set, and sends the email

## Self Assessment

State whether the following statements are true or false:

14. The php.ini file is where you configure your PHP installation.
15. After changing the php.ini file we do not restart the web server.

## 9.8 Working with File Uploads

A very useful aspect of PHP is its ability to manage file uploads to your server. Allowing users to upload a file to your server opens a whole can of worms, so please be careful when enabling file uploads.

By using the global PHP `$_FILES` array you can upload files from a client computer to the remote server.

**Notes**

The first parameter is the form's input name and the second index can be either "name", "type", "size", "tmp\_name" or "error". Like this:

- `$_FILES["file"]["name"]` - the name of the uploaded file
- `$_FILES["file"]["type"]` - the type of the uploaded file
- `$_FILES["file"]["size"]` - the size in kilobytes of the uploaded file
- `$_FILES["file"]["tmp_name"]` - the name of the temporary copy of the file stored on the server
- `$_FILES["file"]["error"]` - the error code resulting from the file upload

This is a very simple way of uploading files. For security reasons, you should add restrictions on what the user is allowed to upload.

## 9.9 Creating the File Upload Form

To allow users to upload a file to the server, you first need to provide a form for them to specify which file they want to upload. Once they click the submit button of the form, the action page is called. This is the page that needs to contain the PHP code to process the uploaded file.

Before a user can upload a file, you need to provide them with an interface that allows them to select a file and initiate the upload. The following code is an example of an input form. There are a couple of important things to note about this code:

- The *action* attribute points to a .php file. This is the file that will process the uploaded file.
- There is an attribute called *enctype*, and its value is *multipart/form-data*.
- One of the *input* fields has *type="file"*.



*Example:*

```
<html>
<head>
<title>PHP File Upload Example</title>
</head>
<body>
<form enctype="multipart/form-data" method="post" action="uploadFile.php">
<input type="file" name="fileToUpload" />

<input type="submit" value="Upload File" />
</form>
</body>
</html>
```

Once the user uploads a file, the file is uploaded into a temporary directory on the server. If you don't move the file it will disappear. Therefore, your action page needs to move the file to another location where it can stay as long as you want it to. Whenever a file is uploaded, you can find out certain information about the file including its name, type, size, as well as the name of the temporary file on the server. These details are made available to you via a PHP array called `$_FILES`.



Case Study

## Custom Web Application Development

### *Client*

A leading Textile Machinery Manufacturer in India and one among the world to produce the entire range of Spinning Machinery. In the early sixties, the company was founded to provide the Indian Textile Mills with the latest Spinning Technology. It caters to the domestic market as well as exports the products to Asian and Oceanic regions.

### *Challenges*

The client wants to conduct an online photo contest as a tribute to their past chairman and managing director as well as an accomplished photographer. They wanted an application which allows the participants to upload photos online after confirming the terms and conditions set by the administrator for the contest.

### *What We Did*

ANGLER developed a user-friendly and attractive application driven towards the objective of the client. The application was built in such a way that the participants found it very user friendly. They can also view the list of photographs uploaded by them so far, edit or delete the photos. The information from the admin side also gets displayed to the users like viewed by admin, blocked by admin with reason, etc. The admin user will also have the features to download the photos, delete or block the photos.

### *Technologies Used*



### **Results**

The following results are provided from the developed web application:

- Initially the contest was not online and thus the participants were sending their photos through post. This was made easier and simpler so that everyone can participate online.
- The participants can have their own privileges to edit/delete their photos online up to the limited period of time post upload.
- They can also get the updates from the admin user regarding the status of the photographs like viewed by admin, blocked by admin, etc.
- The administrator can have the option to delete, block photos and download the photos.

### *Questions*

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: [http://www.angleritech.com/case\\_studies/custom-web-application-development/](http://www.angleritech.com/case_studies/custom-web-application-development/)



## 9.10 Summary

- One of the most powerful features of PHP is the way it handles HTML forms.
- All variables passed to the current script via the HTTP POST method are stored in associative array `$_POST`.
- To keep our script's data neat, for example, we might wish to place all form input into an associative array called `$form`.
- When building a complex page, at some point you will be faced with the need to combine PHP and HTML to achieve your needed results.
- Anything in a PHP script that is not contained within `<?php ?>` tags is ignored by the PHP compiler and passed directly to the web browser.
- Sometimes the script uses the form data to create another form, this form will be submitted to another PHP script, some of the first form data is needed in the final script. To do so, you can use hidden fields.
- `header()` must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP.
- We can easily send mail through our web page using the `mail()` function in PHP.
- For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the `php.ini` file. The `php.ini` file is where you configure your PHP installation.
- A very useful aspect of PHP is its ability to manage file uploads to your server.
- By using the global PHP `$_FILES` array you can upload files from a client computer to the remote server.

## 9.11 Keywords

**File Upload:** Element used to upload a file.

**Form:** Collection of elements to take user.

**header():** User for user redirection.

**Hidden field:** Used to Save State.

**HTML:** Hyper Text Markup Language needed for data presentation.

**mail() function:** It allows you to send emails directly from a script.

**php.ini:** The file where you configure your PHP installation.

## 9.12 Review Questions

1. Give the steps to create a simple input form.
2. How can we access form input with user-defined arrays?
3. Why do we need to combine HTML and PHP code on a single page?
4. What is the use of hidden fields?
5. How do we redirect the user?

6. Can we sending mail on form submission?
7. What System Configuration is needed for the Mail() Function?
8. Write a script to send the mail.
9. What are file uploads?
10. Give the steps to create the file upload form.

Notes

### Answers: Self Assessment

- |           |            |
|-----------|------------|
| 1. True   | 2. False   |
| 3. False  | 4. True    |
| 5. Select | 6. [ ]     |
| 7. PHP    | 8. <?php?> |
| 9. State  | 10. Hidden |
| 11. True  | 12. False  |
| 13. True  | 14. True   |
| 15. False |            |

### 9.13 Further Readings



Books

Adam Trachtenberg, PHP Cookbook.

David Powers, PHP Solutions: Dynamic Web Design Made Easy.

Jason Lengstorf, PHP for Absolute Beginners.

Luke Welling and Laura Thomson, PHP and MySQL Web Development.



Online links

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

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

<http://www.tizag.com/phpT/forms.php>

[http://www.w3schools.com/php/php\\_forms.asp](http://www.w3schools.com/php/php_forms.asp)

## Unit 10: Cookies

### **CONTENTS**

Objectives

Introduction

10.1 Cookies

10.2 Setting Cookies

10.3 Deleting Cookies with PHP

10.4 Session Function Overview

10.4.1 Starting Session

10.4.2 Working with Session Variables

10.5 Destroying Session and Unsetting Variables

10.6 Summary

10.7 Keywords

10.8 Review Questions

10.9 Further Readings

### Objectives

After studying this unit, you will be able to:

- Understand Cookies
- Explain Setting Cookies
- Discuss the Use of Cookies with PHP
- Explain Session Function
- Discuss Destroying Session and Unsetting Variables

### Introduction

A cookie, also known as an HTTP cookie, web cookie, or browser cookie, is a small piece of data sent from a website and stored in a user's web browser while a user is browsing a website. When the user browses the same website in the future, the data stored in the cookie can be retrieved by the website to notify the website of the user's previous activity. Cookies were designed to be a reliable mechanism for websites to remember the state of the website or activity the user had taken in the past. This can include clicking particular buttons, logging in, or a record of which pages were visited by the user even months or years ago.

### 10.1 Cookies

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values. The most popular uses of cookies are:

- To store username/password information so that the user doesn't have to log in every time they visit the website ("remember me" sign ins).

- To simply remember the user's name.
- To keep track of a user's progress during a specified process.
- To remember a user's theme.

## Self Assessment

State whether the following statements are true or false:

1. Cookie is a text file.
2. Cookies are stored on the server.
3. Cookies are used to identify a user.

## 10.2 Setting Cookies

When you create a cookie, using the function `setcookie`, you must specify three arguments. These arguments are `setcookie` (*name, value, expiration*):

- **name:** The name of your cookie. You will use this name to later retrieve your cookie, so don't forget it!
- **value:** The value that is stored in your cookie. Common values are username (string) and last visit (date).
- **expiration:** The date when the cookie will expire and be deleted. If you do not set this expiration date, then it will be treated as a session cookie and be removed when the browser is restarted.

Any cookies sent to you from the client will automatically be included into a `$_COOKIE` auto-global array if `variables_order` contains "C". If you wish to assign multiple values to a single cookie, just add `[]` to the cookie name. Depending on `register_globals`, regular PHP variables can be created from cookies. However it's not recommended to rely on them as this feature is often turned off for the sake of security. `$HTTP_COOKIE_VARS` is also set in earlier versions of PHP when the `track_vars` configuration variable is set.

In this example we will be creating a cookie that stores the user's last visit to measure how often people return to visit our webpage.



Example:

```
<?php
//Calculate 60 days in the future
//seconds * minutes * hours * days + current time
$inTwoMonths = 60 * 60 * 24 * 60 + time();
setcookie('lastVisit', date("G:i - m/d/Y"), $inTwoMonths);
?>
```

If your cookie hasn't expired, we can retrieve it from the user's PC using the aptly named `$_COOKIE` associative array. The name of your stored cookie is the key and will let you retrieve your stored cookie value.

Notes



Example:

```
<?php
if(isset($_COOKIE['lastVisit']))
 $visit = $_COOKIE['lastVisit'];
else
 echo "You've got some stale cookies!";

echo "Your last visit was - ". $visit;
?>
```

This handy script first uses the *isset* function to be sure that our “lastVisit” cookie still exists on the user’s PC, if it does, then the user’s last visit is displayed. If the user visited our site on February 28, 2008 it might look something like this:

**Output:**

Your last visit was - 11:48 - 02/28/13

### Self Assessment

Fill in the blank:

4. Cookie is set using ..... function.
5. `setcookie()` takes in ..... parameters.
6. We can retrieve it from the user’s PC using the aptly named ..... associative array.
7. .... parameter in `setcookie()` specifies the name of your cookie.
8. .... parameter in `setcookie()` specifies the value that is stored in your cookie.
9. .... parameter in `setcookie()` specifies the date when the cookie will expire and be deleted.

### 10.3 Deleting Cookies with PHP

Cookies can also be deleted. This is useful for situations such as when a user logs out of your site. To delete a cookie, call the `setcookie()` function again with the same name, folder and domain that you used earlier to set the cookie. However, instead of an expiry date set in the future, this time give an expiry date sometime in the past.



Example:

```
$date_of_expiry = time() - 60 ;
setcookie("userlogin", "anonymous", $date_of_expiry, "/",
 "example.com");
```

The above code simply sets the expiry date 60 seconds in the past, effectively making the cookie no longer valid.

## 10.4 Session Function Overview

Notes

The problem with cookies is that a user can block cookies or delete them at any time. If, for example, your website's shopping cart utilized cookies, and a person had their browser set to block them, then they could not shop at your website.

Sessions are not reliant on the user allowing a cookie. They work instead like a token allowing access and passing information while the user has their browser open. Sessions are stored on the server, which means clients do not have access to the information you store about them – this is particularly important if you store shopping baskets or other information you do not want your visitors to be able to edit by hand by hacking their cookies. Session data, being stored on your server, does not need to be transmitted with each page; clients just need to send an ID and the data is loaded from the local file. Finally, sessions can be any size you want because they are held on your server, whereas many web browsers have a limit on how big cookies can be to stop rogue web sites chewing up gigabytes of data with meaningless cookie information.

The problem with sessions is that when you close your browser you also lose the session. So, if you had a site requiring a login, this couldn't be saved as a session like it could as a cookie, and the user would be forced to re-login every time they visit.

### 10.4.1 Starting Session

Before you can begin storing user information in your PHP session, you must first start the session. When you start a session, it must be at the very beginning of your code, before any HTML or text is sent. Below is a simple script that you should place at the beginning of your PHP code to start up a PHP session.



*Example:*

```
<?php
session_start(); // start up your PHP session
?>
```

The function `session_start()` takes no parameters. It always returns `TRUE`, so you don't have to bother to check its return value. This tiny piece of code will register the user's session with the server, allow you to start saving user information and assign a UID (unique identification number) for that user's session.

### 10.4.2 Working with Session Variables

When you want to store user data in a session use the `$_SESSION` associative array. This is where you both store and retrieve session data. In previous versions of PHP there were other ways to perform this store operation, but it has been updated and this is the correct way to do it.



*Example:*

```
<?php
session_start();
$_SESSION['views'] = 1; // store session data
echo "Pageviews = ". $_SESSION['views']; //retrieve data
?>
```

**Notes****Output:**

Pageviews = 1

In this example we learned how to store a variable to the session associative array `$_SESSION` and also how to retrieve data from that same array.

Now that you are able to store and retrieve data from the `$_SESSION` array, we can explore some of the real functionality of sessions. When you create a variable and store it in a session, you probably want to use it in the future. However, before you use a session variable it is necessary that you check to see if it exists already. This is where PHP's *isset* function comes in handy. *isset* is a function that takes any variable you want to use and checks to see if it has been set. That is, it has already been assigned a value. With our previous example, we can create a very simple pageview counter by using *isset* to check if the pageview variable has already been created. If it has we can increment our counter. If it doesn't exist we can create a pageview counter and set it to one. Here is the code to get this job done:

**Example:**

```
<?php
session_start();
if(isset($_SESSION['views']))
 $_SESSION['views'] = $_SESSION['views']+ 1;
else
 $_SESSION['views'] = 1;

echo "views = ". $_SESSION['views'];
?>
```

The first time you run this script on a freshly opened browser the *if statement* will fail because no session variable *views* would have been stored yet. However, if you were to refresh the page the *if statement* would be true and the counter would increment by one. Each time you reran this script you would see an increase in *view* by one.

**Self Assessment**

State whether the following statements are true or false:

10. Cookies cannot be deleted by the user.
11. Sessions are stored on the client machine.
12. `session_start()` is used to start a session.
13. When you want to store user data in a session use the `$_SESSION` associative array.

**10.5 Destroying Session and Unsetting Variables**

If you wish to delete some session data, you can use the `unset()` or the `session_destroy()` function.

The `unset()` function is used to free the specified session variable:

```
<?php
session_start();
```

## Notes

```
if(isset($_SESSION['views']))
 unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the `session_destroy()` function:

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



*Note* `session_destroy()` will reset your session and you will lose all your stored session data.

## Self Assessment

State whether the following statements are true or false:

14. The `unset()` function is used to free the specified session variable.
15. You can also completely destroy the session by calling the `session_destroy()` function.



Case Study

## Cookies

**H**eaders are pieces of information sent to the browser before the main page is evaluated. When a cookie is sent, it must be accompanied by a compact privacy policy so the user's browser can look at both, see if they marry up, and decide what to do. Get this bit right, and all but the toughest setting on your user's browser won't have a problem with your cookies.

Now, we don't need to go through the details of this, because the good folks at the Privacy Council offer an automated service that creates compact policies. They'll even email the result to you. Just register with them, select from a series of multiple choice questions about what your site does and doesn't do, and you're in business again.

Now, you need to know how to implement the compact policy into your pages. Again, I'll illustrate this point with the code I used for my own site.

In pure HTML pages, insert this code into the head section of your page:

```
<meta http-equiv="P3P" content='CP="IDC DSP COR CURa ADMa
OUR IND PHY ONL COM STA"'>
```

In PHP pages, insert this as the first thing on the page after the setting of the cookie:

```
<?php header('P3P: CP="IDC DSP COR CURa ADMa OUR IND
PHY ONL COM STA"'); ?>
```

For other server-side languages, see the link below titled "Header Creation".

Of course, don't just use the code above as-is. You need to go to the URL given below at the Privacy Council, and generate your own. Don't worry, it's straightforward and non-technical.

*Contd...*



## Notes

It's important to understand that only pages that place cookies need to have a CP. Form pages don't set cookies, so they don't need a policy. Remember that if you use a piece of JavaScript code to set a cookie for popup control, the page that calls the popup and does the cookie-setting will require a compact policy.

Some sites may need more than one policy. Why? Well, a policy describes what information is collected (and why) in a specific URL location. That can be the whole site, or specific folders on your site. While most of us will probably generate one policy for the whole site, it is possible to point to a different policy location in each header, on each page. You would do this if, for example, one section of your site allowed users to subscribe to your newsletter by providing their email addresses and first names, while the other offers a members' area that uses cookies to customize the browser's view. Perhaps you also provide a shopping cart that stores user status and personal information for use in processing the order.

If you need to point to another policy that has been generated to describe a specific use of cookies like this, you'll want to put one of the following headers on the page(s) that pass cookies to the visiting browser:

Firstly, using PHP:

```
<?php Header('P3P: href="/your_2nd_policy/p3p.xml"
CP="your compact policy"'); ?>
```

Now, using HTML:

```
<meta http-equiv="P3P" href="/your_2nd_policy/p3p.xml"
content='CP="your compact policy"'>
```

If, following these guidelines, you've built your own individual files, you can test them with the policy validator.

### Questions

1. Study and analyze the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: <http://www.sitepoint.com/p3p-cookies-ie6/>

## 10.6 Summary

- A cookie, also known as an HTTP cookie, web cookie, or browser cookie, is a small piece of data sent from a website and stored in a user's web browser while a user is browsing a website.
- When you create a cookie, using the function `setcookie`, you must specify three arguments: Name, value, expiration.
- `$_COOKIE` associative array to store cookie.
- Cookies can also be deleted.
- Sessions are not reliant on the user allowing a cookie.
- Before you can begin storing user information in your PHP session, you must first start the session.
- The function `session_start()` takes no parameters.

## Notes

- When you want to store user data in a session use the `$_SESSION` associative array.
- `isset` is a function that takes any variable you want to use and checks to see if it has been set.
- If you wish to delete some session data, you can use the `unset()` or the `session_destroy()` function.
- The `unset()` function is used to free the specified session variable.

## 10.7 Keywords

**`$_COOKIE`:** An associative array to store cookie.

**`$_SESSION`:** An associative array to store a session.

***Cookie*:** It is a small piece of data sent from a website and stored in a user's web browser while a user is browsing a website.

***Isset()*:** It is a function that takes any variable you want to use and checks to see if it has been set.

***session\_destroy()*:** It is used to delete some session data.

***session\_start()*:** To start a session.

***Sessions*:** They are not reliant on the user allowing a cookie.

***setcookie()*:** To set a cookie using the function.

***unset()*:** It is a function is used to free the specified session variable.

## 10.8 Review Questions

1. What are cookies?
2. How do we set cookies?
3. Give the way we can delete a cookie.
4. What is a session?
5. Differentiate between a cookie and a session.
6. How do we start a session?
7. What are session variables?
8. How can we destroy a session?
9. What is the way to unset a session?
10. Give the advantages of using sessions over cookies.

### Answers: Self Assessment

- |               |                             |
|---------------|-----------------------------|
| 1. True       | 2. False                    |
| 3. True       | 4. <code>setcookie()</code> |
| 5. 3          | 6. <code>\$_COOKIE</code>   |
| 7. Name       | 8. Value                    |
| 9. Expiration | 10. False                   |

**Notes**

- |           |          |
|-----------|----------|
| 11. False | 12. True |
| 13. True  | 14. True |
| 15. True  |          |

## **10.9 Further Readings**



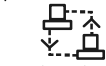
*Books*

Adam Trachtenberg, PHP Cookbook.

David Powers, PHP Solutions: Dynamic Web Design Made Easy.

Jason Lengstorf, PHP for Absolute Beginners.

Luke Welling and Laura Thomson, PHP and MySQL Web Development.



*Online links*

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

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

<http://www.tizag.com/phpT/forms.php>

[http://www.w3schools.com/php/php\\_forms.asp](http://www.w3schools.com/php/php_forms.asp)

## Unit 11: Directories and Files

Notes

### CONTENTS

Objectives

Introduction

11.1 Including Files with PHP Include( ) Function

11.2 Validating Files

11.2.1 Checking for Existence with file\_exists ()

11.2.2 A File or a Directory

11.2.3 Checking the Status of a File

11.2.4 Determining File Size with filesize()

11.2.5 Getting Date Information About a File

11.3 Creating File

11.4 Delete File

11.5 Opening a File for Reading

11.6 Writing Files

11.6.1 File Open Write

11.6.2 File Write Fwrite Function

11.6.3 File Write: Overwriting

11.7 Appending Files

11.7.1 File Read

11.7.2 File Open: Read

11.7.3 File Read fread Function

11.7.4 File Read: Gets Function

11.7.5 When and How to Include Files

11.8 Summary

11.9 Keywords

11.10 Review Questions

11.11 Further Readings

### Objectives

After studying this unit, you will be able to:

- Explain the Include Function
- Discuss File Validation
- Understand File Creation
- Explain the Process to Delete a File

**Notes**

- Elaborate upon the File Opening Process
- Discuss the Writing of Files
- Understand how to Append in Files

## **Introduction**

PHP has a complete set of directory support functions. PHP gives you a variety of functions to read and manipulate directories and directory entries. Like other file-related parts of PHP, the functions are similar to the C functions that accomplish the same tasks, with some simplifications. Directories contain files. Manipulating files is a basic necessity for serious programmers and PHP gives you a great deal of tools for creating, uploading, and editing files. Data is saved in our hard disk in the home directory which can be accessed via, echo \$HOME or using the symbol ~ /tmp is a temporary directory for the user to store files but it is washed off when the system reboots. So care should be taken while storing any permanent use data in this directory.

### **11.1 Including Files with PHP Include( ) Function**

The *include* statement includes and evaluates the specified file. Files are included based on the file path given or, if none is given, the `include_path` specified. If the file isn't found in the `include_path`, *include* will finally check in the calling script's own directory and the current working directory before failing. The *include* construct will emit a warning if it cannot find a file; this is different behavior from *require*, which will emit a fatal error. If a path is defined — whether absolute (starting with a drive letter or/ on Windows, or / on Unix/Linux systems) or relative to the current directory (starting with . or ..) — the `include_path` will be ignored altogether. For example, if a filename begins with ../, the parser will look in the parent directory to find the requested file. When a file is included, the code it contains inherits the variable scope of the line on which the include occurs. Any variables available at that line in the calling file will be available within the called file, from that point forward. However, all functions and classes defined in the included file have the global scope.



*Example:*

#### ***File1.php***

```
<?php

$color = 'green';
$fruit = 'apple';

?>
```

#### ***File2.php***

```
<?php

echo "A $color $fruit";

include 'File1.php';

echo "A $color $fruit";
?>
```

**Output:** A green apple

## Self Assessment

## Notes

Fill in the blanks:

1. Data is saved in our hard disk in the home directory which can be accessed .....
2. .... is a temporary directory for the user to store files.
3. The ..... statement includes and evaluates the specified file.

## 11.2 Validating Files

It is important to validate a file before using it. The way to do this is explained below.

### 11.2.1 Checking for Existence with `file_exists ()`

It checks whether a file or directory exists. This function returns TRUE if the file or directory exists, otherwise it returns FALSE. Its syntax is,

```
file_exists(path_to_check)
```



*Example:*

```
<?php
$filename = '/path/to/foo.txt';
if (file_exists($filename))
{
 echo "The file $filename exists";
}
else
{
 echo "The file $filename does not exist";
}
?>
```

### 11.2.2 A File or a Directory

The `is_file()` function tells whether the filename is a regular file or not. It returns **TRUE** if the filename exists and is a regular file, **FALSE** otherwise.



*Example:*

```
<?php
$file = "test.txt";
if(is_file($file))
{
 echo ("$file is a regular file");
}
else
{
 echo ("$file is not a regular file");
}
?>
```

**Notes**

**Output:** test.txt is a regular file



*Notes* Because PHP's integer type is signed and many platforms use 32-bit integers, some file system functions may return unexpected results for files which are larger than 2GB.

On the other hand, the `is_dir()` function checks whether the specified file is a directory. This function returns `TRUE` if the directory exists.



*Example:*

```
<?php
$file = "images";
if(is_dir($file))
{
 echo ("$file is a directory");
}
else
{
 echo ("$file is not a directory");
}
?>
```

**Output:** images is a directory

### 11.2.3 Checking the Status of a File

The status of any file can be readable, writable or executable. It can be found out using the following functions.

- **`is_readable()`** function tells whether a file exists and is readable. Returns `TRUE` if the file or directory specified by *filename* exists and is readable, `FALSE` otherwise.



*Example:*

```
<?php
$filename = 'test.txt';
if (is_readable($filename))
{
 echo 'The file is readable';
}
else
{
 echo 'The file is not readable';
}
?>
```

- **`is_writable ()`** function tells you whether the filename is writable. It returns `TRUE` if the *filename* exists and is writable. The filename argument may be a directory name allowing you to check if a directory is writable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.



Example:

```
<?php
$filename = 'test.txt';
if (is_writable($filename))
{
 echo 'The file is writable';
}
else
{
 echo 'The file is not writable';
}
?>
```

- *is\_executable()* function tells you whether the filename is executable. It returns **TRUE** if the filename exists and is executable, or **FALSE** on error.



Example:

```
<?php

$file = '/home/vincent/somefile.sh';

if (is_executable($file))
{
 echo $file.' is executable';
}
Else
{
 echo $file.' is not executable';
}

?>
```

### 11.2.4 Determining File Size with *filesize()*

The `filesize()` function gets the size for the given file in bytes, or **FALSE** (and generates an error of level **E\_WARNING**) in case of an error.



*Notes* Because PHP's integer type is signed and many platforms use 32-bit integers, some file system functions may return unexpected results for files which are larger than 2GB.



Example:

```
<?php
echo filesize("test.txt");
?>
```



### 11.2.5 Getting Date Information About a File

In order to know about the date information of a file, we make use of two functions - `filetime()` and `filectime()`.

`filetime()` gets last access time of file. It returns the time the file was last accessed, or **FALSE** on failure. The time is returned as a Unix timestamp.



*Example:*

```
<?php

$filename = 'somefile.txt';
if (file_exists($filename)) {
 echo "$filename was last accessed: " . date("F d Y H:i:s.",
filetime($filename));
}

?>
```

**Output:** somefile.txt was last accessed: December 29 2012 22:16:23.

The atime of a file is supposed to change whenever the data blocks of a file are being read. This can be costly performance-wise when an application regularly accesses a very large number of files or directories. Some Unix file systems can be mounted with atime updates disabled to increase the performance of such applications; USENET news spools are a common example. On such file systems this function will be useless.

`filectime()` gets inode change time of file. It returns the time the file was last changed, or **FALSE** on failure. The time is returned as a Unix timestamp.



*Example:*

```
<?php

$filename = 'somefile.txt';
if (file_exists($filename)) {
 echo "$filename was last changed: " . date("F d Y H:i:s.",
filectime($filename));
}

?>
```

**Output:** somefile.txt was last changed: December 29 2012 22:16:23.



*Notes* In some Unix texts the `ctime` of a file is referred to as being the creation time of the file. This is wrong. There is no creation time for Unix files in most Unix file systems.

### Self Assessment

Fill in the blanks:

- ..... function checks whether a file or directory exists.

5. .... function tells whether the filename is a regular file or not.
6. .... function checks whether the specified file is a directory.

Notes

### 11.3 Creating File

Before you can do anything with a file it has to exist! In this unit you will learn how to create a file using PHP. In PHP, a file is created using a command that is also used to open files. It may seem a little confusing, but we'll try to clarify this conundrum. In PHP the *fopen* function is used to open files. However, it can also **create** a file if it does not **find** the file specified in the function call. So if you use *fopen* on a file that does not exist, it will create it, given that you open the file for writing or appending. The *fopen* function needs two important pieces of information to operate correctly. First, we must supply it with the name of the file that we want it to open. Secondly, we must tell the function what we plan on doing with that file (i.e. read from the file, write information, etc.). Since we want to create a file, we must supply a file name and tell PHP that we want to write to the file.



*Notes* We have to tell PHP we are writing to the file, otherwise it will not create a new file.



*Example:*

```
$ourFileName = "testFile.txt";
$ourFileHandle = fopen($ourFileName, 'w') or die("can't open file");
fclose($ourFileHandle);
```

The file "testFile.txt" should be created in the same directory where this PHP code resides. PHP will see that "testFile.txt" does not exist and will create it after running this code. There's a lot of information in those three lines of code, let's make sure you understand it.

- `$ourFileName = "testFile.txt";`

Here we create the name of our file, "testFile.txt" and store it into a PHP String variable *\$ourFileName*.

- `$ourFileHandle = fopen($ourFileName, 'w') or die("can't open file");`

This bit of code actually has two parts. First we use the function *fopen* and give it two arguments: our file name and we inform PHP that we want to write by passing the character "w".

Second, the *fopen* function returns what is called a *file handle*, which will allow us to manipulate the file. We save the file handle into the *\$ourFileHandle* variable.

- `fclose($ourFileHandle);`

We close the file that was opened. *fclose* takes the file handle that is to be closed.

If you are trying to get this program to run and you are having errors, you might want to check that you have granted your PHP file access to write information to the hard drive. Setting permissions is most often done with the use of an FTP program to execute a command called *CHMOD*. Use *CHMOD* to allow the PHP file to write to disk, thus allowing it to create a file.

Notes

**Self Assessment**

State whether the following statements are:

- 7. *fopen* function returns what is called a *file handle*.
- 8. Setting permissions on a file uses the `perm` command.

**11.4 Delete File**

We can erase a file with `unlink()` function. It is similar to the Unix C `unlink()` function. A `E_WARNING` level error will be generated on failure.



Example:

```
<?php
$fh = fopen('test.html', 'a');
fwrite($fh, '<h1>Hello world!</h1>');
fclose($fh);

unlink('test.html');
?>
```

**Self Assessment**

State whether the following statements are true or false:

- 9. We can erase a file with `unset()` function.
- 10. `unlink()` returns a `E_WARNING` level error will be generated on failure.

**11.5 Opening a File for Reading**

The `fopen()` function opens a file or URL. If `fopen()` fails, it returns `FALSE` and an error on failure. You can hide the error output by adding an '@' in front of the function name.

Its syntax is, `fopen(filename,mode,include_path,context)`

**Table 11.1: fopen( ) Parameters**

Parameter	Description
Filename	Required. Specifies the file or URL to open
Mode	Required. Specifies the type of access you require to the file/steam. Possible values: <ul style="list-style-type: none"> <li>• "r" (Read only. Starts at the beginning of the file)</li> <li>• "r+" (Read/Write. Starts at the beginning of the file)</li> <li>• "w" (Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist).</li> <li>• "a" (Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist).</li> <li>• "a+" (Read/Write. Preserves file content by writing to the end of the file)</li> <li>• "x" (Write only. Creates a new file. Return <code>FALSE</code> and an error if the already exists)</li> </ul>

Contd...

## Notes

	<ul style="list-style-type: none"> <li>• "x+" (Read/Write. Creates a new file. Returns FALSE and an error if file already exists)</li> </ul>
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
Context	Optional. Specifies the context of the file handle. Context is a set of optional that can modify the behaviour of a stream

Source: [http://www.w3schools.com/php/func\\_filesystem\\_fopen.asp](http://www.w3schools.com/php/func_filesystem_fopen.asp)

When writing to a text file, be sure to use the correct line-ending character. Unix systems use \n, Windows systems use \r\n, and Macintosh systems use \r as the line ending character. Windows offers a translation flag ('t') which will translate \n to \r\n when working with the file. You can also use 'b' to force binary mode. To use these flags, specify either 'b' or 't' as the last character of the mode parameter.

## Self Assessment

State whether the following statements are true or false:

11. The fopen() function opens a file or URL.
12. You can hide the error output by adding an ! in front of the function name.

## 11.6 Writing Files

After opening a file we can read it or write into it. Let us see how the writing is done.

### 11.6.1 File Open Write

Open a file in the write mode with fopen( ) and you can write to the file.



Example:

```
$file_write_f = "File1.txt";
$fh = fopen($file_write_f, 'w');
```

### 11.6.2 File Write Fwrite Function

The fwrite() writes to an open file. The function will stop at the end of the file or when it reaches the specified length, whichever comes first. This function returns the number of bytes written, or FALSE on failure. Its syntax is,

```
fwrite(file, string, length)
```

Parameter	Description
file	Required. Specifies the open file to write to
string	Required. Specifies the string to write to the open file
length	Optional. Specifies the maximum number of bytes to write



*Notes* This function is binary-safe (meaning that both binary data, like images, and character data can be written with this function).

Notes



Example:

```
<?php
$file = fopen("test.txt","w");
echo fwrite($file,"Hello World. Testing!");
fclose($file);
?>
```

### 11.6.3 File Write: Overwriting

It is possible to overwrite the contents of a file. To do this,

```
<?php
$filename = "justatest.txt";
$output = "HELLO";
$filehandle = fopen($filename, 'w');
fwrite($filehandle, $output);
fclose($filehandle);
?>
```

The text HELLO will be overwritten by the content of the filename specified.

### Self Assessment

Fill in the blanks:

- 13. The ..... writes to an open file.
- 14. fwrite ( ) function returns the number of ..... written.

## 11.7 Appending Files

As shown with fopen(), there are multiple ways you can open a file (r, r+, w, etc.). To append the file means that you preserve its existing contents and add the data to the end of the file. All you need to do to append the file is to open it in either append (a) or read & append (a+) mode as so:

```
$file=fopen("myfile.txt", "a")
```

Opening the file in append mode will write any data to the end of the file. If we wanted to actually write some data to the file, we can assign that file a variable and specify what data to append:

```
$file=fopen("myfile.txt","a") or die("myfile.txt does not exist!");
$info = "I have brown hair.n";
fwrite($file, $info);
$info = "I have brown eyes.n";
fwrite($file, $info);
fclose($file);
```

This bit of code would add two lines to the end of myfile.txt:

I have brown hair.

I have brown eyes.

So, if the original contents of myfile.txt before you opened it were:

I am six feet tall.

I weigh 165lbs.

After appending the new data, your myfile.txt contents would be:

Notes

I am six feet tall.

I weigh 165lbs.

I have brown hair.

I have brown eyes.

### 11.7.1 File Read

We will now discuss the ways to read from a file.

### 11.7.2 File Open: Read

We can use fopen() function in read mode to read a file.



Example:

```
$file_reading = "File1.txt";
$fh = fopen($file_reading, 'r');
```

### 11.7.3 File Read fread Function

The fread() reads from an open file. The function will stop at the end of the file or when it reaches the specified length, whichever comes first. This function returns the read string, or FALSE on failure.



Example:

```
<?php
$file = fopen("test.txt","r");
fread($file, filesize("test.txt"));
fclose($file);
?>
```

### 11.7.4 File Read: Gets Function

It gets line from file pointer. Its syntax is,

```
string fgets (resource $handle [, int $length])
```

**handle** : The file pointer must be valid, and must point to a file successfully opened by fopen() or fsockopen() (and not yet closed by fclose()).

**length**: Reading ends when length - 1 bytes have been read, or a newline (which is included in the return value), or an EOF (whichever comes first). If no length is specified, it will keep reading from the stream until it reaches the end of the line.



**Notes** Until PHP 4.3.0, omitting it would assume 1024 as the line length. If the majority of the lines in the file are all larger than 8KB, it is more resource efficient for your script to specify the maximum line length.

**Notes**

It returns a string of up to *length* - 1 bytes read from the file pointed to by *handle*. If there is no more data to read in the file pointer, then **FALSE** is returned. If an error occurs, **FALSE** is returned.



*Example:*

```
<?php
$handle = @fopen("/tmp/inputfile.txt", "r");
if ($handle) {
 while (($buffer = fgets($handle, 4096)) !== false) {
 echo $buffer;
 }
 if (!feof($handle)) {
 echo "Error: unexpected fgets() fail\n";
 }
 fclose($handle);
}
?>
```

**11.7.5 When and How to Include Files**

We should be careful while including files. There are three principles to follow while including files:

1. Only use `include_once` or `require_once` to include PEAR code.
2. Determine the correlation between classes and file names. PEAR uses the one-class-per-file principle, with the intention that it should be trivial to generate the required file name from the class name. Replace underscores with the directory separator character, `append.php`.
3. Each file should use includes to express clearly which class it depends on from other packages.

**Self Assessment**

State whether the following statements are true or false:

15. `fgets( )` gets line from file pointer.
16. `fgets ( )` returns a string of up to *length* bytes.



*Case Study*

**Scan Directories with PHP's DirectoryIterators**

One of PHP5's most interesting new features is the addition of Iterators, a collection of ready-made interfaces designed to help in navigating and processing hierarchical data structures. These Iterators significantly reduce the amount of code required to process an XML document tree or a file collection.

A number of Iterators are available, including the *ArrayIterator*, *CachingIterator*, *LimitIterator*, *RecursiveIterator*, *SimpleXMLIterator* and *DirectoryIterator*.

*Contd...*

It's this last Iterator that's the subject of this How do I... tutorial. The *DirectoryIterator* provides a quick and efficient way of processing the files in a directory; with a little creative coding, it can also be used to recursively process a nested directory tree. Both these tasks can be accomplished using just a few lines of code, representing a significant improvement over the "standard" way of doing things.

#### *Processing a single-level directory*

Let's begin with something simple: processing a single-level directory. Type (or copy) the following script (Listing A), altering the directory path to reflect your local configuration:

#### **Listing A**

```
<?php
$it = new DirectoryIterator("/tmp/mystuff");

foreach($it as $file) {
if (!$it->isDot()) {
echo $file . "\n";
}
}
?>
```

When you view the output of this script in your browser, you should see a list of the files in the named directory. How did this happen? Well, the *DirectoryIterator* class provides a pre-built interface to iterating over the contents of a directory; once instantiated with the location of the target directory, it can then be processed as though it were a standard PHP array, with each element representing a file in the directory. Note the use of the *isDot()* method to filter out the "." and ".." directories, respectively.

#### *Processing a nested directory tree*

Recursively processing a nested directory tree is almost as simple. In this case, the *DirectoryIterator* needs to check each object it encounters within the first-level directory, determine whether it is a file or directory, and, if a directory, drill one level deeper to examine the next level of contents. This sounds fairly complex, and in the past could easily add up to 15-plus lines of code.

With PHP5, though, all you need are two new Iterators: the *RecursiveDirectoryIterator* and the *RecursiveIteratorIterator*, which together incorporate all the above functionality. Take a look at Listing B:

#### **Listing B**

```
<?php
$it = new RecursiveDirectoryIterator("/tmp");
foreach(new RecursiveIteratorIterator($it) as $file) {
echo $file . "\n";
}
?>
```

In this case, the output should now include a list of all the files and directories under the starting directory. Needless to say, this kind of built-in recursive interface is very handy for situations that require you to process all the files under a particular directory level — for example, when recursively compressing a directory tree, or altering group/owner permissions on a series of nested files.

*Contd...*



## Notes

**A real-world application: Printing a directory tree**

A common application of directory recursion involves printing a graphical directory tree. With Iterators, this task is a snap, because included within the Iterator class documentation is an example class written specifically for this purpose.

The *DirectoryTreeIterator* (credit: Marcus Boerger) provides additional enhancements to the *RecursiveIterator* discussed previously, most notably ASCII markers that represent depth and location within the tree structure.

You can examine the source code for this example class on the php.net Website.

Listing C shows how the *DirectoryTreeIterator* can be used.

**Listing C**

```
<?php
$it = new DirectoryTreeIterator("/tmp/cookbook/");
foreach($it as $path) {
 echo $path . "\n";
}
?>
```

And here's a brief snippet of the output you might see:

```
| -ch01
| | |-recipe01
| | | |-example01.php
| | | \-example02.php
| | |-recipe02
| | | |-example01.php
| | | \-example02.php
| | |-recipe03
| | \-example01.php
...

```

To better understand the value-add of these various *DirectoryIterators*, try coding the three applications demonstrated in this tutorial using standard file and directory functions. Once you're done, you'll have a new appreciation for the simplicity and ease of use the *DirectoryIterators* bring to PHP5. Happy coding!

*Source:* <http://www.techrepublic.com/blog/programming-and-development/how-do-i-recursively-scan-directories-with-phps-directoryiterators/417>

**11.8 Summary**

- PHP gives you a variety of functions to read and manipulate directories and directory entries.
- Directories contain files.
- Data is saved in our hard disk in the home directory which can be accessed via `echo $HOME`.
- `/tmp` is a temporary directory for the user to store files.
- The `include` statement includes and evaluates the specified file.
- `file_exists()` : It checks whether a file or directory exists.
- The `is_file()` function tells whether the filename is a regular file or not.
- `is_dir()` function checks whether the specified file is a directory.

## Notes

- `is_readable()` function tells whether a file exists and is readable.
- `is_writable ()` function tells you whether the filename is writable.
- `is_executable()` function tells you whether the filename is executable.
- The `filesize( )` function gets the size for the given file.
- `fileatime()` gets last access time of file.
- `filectime()` gets inode change time of file.
- `fopen` function is used to open files.
- Setting permissions is most often done with the use of an FTP program to execute a command called CHMOD.
- We can erase a file with `unlink()` function.
- The `fopen()` function opens a file or URL.
- The `fwrite()` writes to an open file.
- The `fread()` reads from an open file.
- Gets Function gets line from file pointer.

### 11.9 Keywords

*/tmp*: It is a temporary directory for the user to store files.

*Directories*: A set of files.

*file\_exists ()*: It checks whether a file or directory exists.

*is\_dir()*: It checks whether the specified file is a directory.

*is\_executable()*: It tells you whether the filename is executable.

*is\_file()*: It tells whether the filename is a regular file or not.

*is\_readable()*: It tells whether a file exists and is readable.

*is\_writable ()*: It tells you whether the filename is writable.

### 11.10 Review Questions

1. What is the function of PHP Include( )?
2. Why do we use `file_exists ()` function?
3. What is the way to check for a File or a Directory?
4. Which function is used to check the Status of a File?
5. Write a short note on `filesize()` function.
6. Give the steps to create a file.
7. What is the method followed to delete a file?
8. What is the method to overwrite a file?
9. Explain the `fread ( )` function.
10. How is writing different from appending in a file?

Notes

**Answers: Self Assessment**

- |               |                   |
|---------------|-------------------|
| 1. echo home  | 2. /tmp           |
| 3. include    | 4. file_exists( ) |
| 5. is_file( ) | 6. is_dir( )      |
| 7. True       | 8. False          |
| 9. False      | 10. True          |
| 11. True      | 12. False         |
| 13. fwrite( ) | 14. bytes         |
| 15. True      | 16. False         |

**11.11 Further Readings**



*Books*

Adam Trachtenberg, PHP Cookbook.

David Powers, PHP Solutions: Dynamic Web Design Made Easy.

Jason Lengstorf, PHP for Absolute Beginners.

Luke Welling and Laura Thomson, PHP and MySQL Web Development.



*Online links*

<http://davidwalsh.name/basic-php-file-handling-create-open-read-write-append-close-delete>

[http://webcheatsheet.com/PHP/working\\_with\\_directories.php](http://webcheatsheet.com/PHP/working_with_directories.php)

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

<http://www.tizag.com/phpT/forms.php>

## Unit 12: Images

Notes

### CONTENTS

Objectives

Introduction

12.1 Image Creation Process

12.1.1 Basic PHP Knowledge

12.1.2 Your PHP Must Have Been Compiled with the GD Library

12.1.3 Free Type Must Be Compiled for True Type Font Support

12.1.4 Creating an Image from Scratch Using PHP

12.2 Necessary Modifications to PHP

12.2.1 Obtaining Additional Libraries

12.3 Drawing a New Image

12.3.1 Drawing Lines and Shapes

12.3.2 Using a Colour Fill

12.4 Modifying an Existing Image

12.4.1 Using True Type Fonts

12.4.2 Drawing to Your Image

12.5 Image Creation from User Input

12.6 Summary

12.7 Keywords

12.8 Review Questions

12.9 Further Readings

### Objectives

After studying this unit, you will be able to:

- Explain how Images are Created
- Discuss Necessary Modifications to PHP
- Elaborate the Process of Drawing a New Image.
- Modify an Existing Image
- Explain the Creation of an Image from User Input

### Introduction

PHP is not limited to creating just HTML output. It can also be used to create and manipulate image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP, and XPM. Even more convenient, PHP can output image streams directly to a browser.

## 12.1 Image Creation Process

We will be discussing in detail the entire process to create images in PHP in the sections below.

### 12.1.1 Basic PHP Knowledge

For creating and using images the basic knowledge PHP is sufficient. Else it's a better option to first learn the main pointers and then come back to this unit.

### 12.1.2 Your PHP Must Have Been Compiled with the GD Library

You will need a PHP interpreter compiled with the GD library in order to create and use the functions given in this unit. This is inbuilt if you are using the PHP provided by a commercial web host. The GD library is used for dynamic image creation. From PHP we use with the GD library to create GIF, PNG or JPG images instantly from our code. This allows us to do things such as create charts on the fly, created an anti-robot security image, create thumbnail images, or even build images from other images. If you are unsure if you have GD library, you can run `phpinfo()` to check that GD Support is enabled. If you don't have it, you can download it for free.

### 12.1.3 Free Type Must Be Compiled for True Type Font Support

PHP needs to have FreeType support compiled into it in order to enable the use of True Type Font Support.

### 12.1.4 Creating an Image from Scratch Using PHP

Let us explain this through a small code snippet.

```
<?php
 header ("Content-type: image/png");
 $handle = ImageCreate (130, 50) or die ("Cannot Create image");
 $bg_color = ImageColorAllocate ($handle, 255, 0, 0);
 $txt_color = ImageColorAllocate ($handle, 0, 0, 0);
 ImageString ($handle, 5, 5, 18, "PHP.About.com", $txt_color);
 ImagePng ($handle);
?>
```

With this code we are creating a PNG image.

- In our first line, the header, we set the content type. If we were creating a jpg or gif image, this would change accordingly.
- Next we have the image handle. The two variables in *ImageCreate()* are the width and height of our rectangle, in that order. This function Returns an image resource identifier on success, **FALSE** on errors. Our rectangle is 130 pixels wide, and 50 pixels high.
- Next we set our background color. We use *ImageColorAllocate()* and have four parameters. The first is our handle, and the next three determine the color. They are the Red, Green and Blue values (in that order) and must be an integer between 0 and 255.
- Next we choose our text color, using the same format as our background color. We have chosen black.

## Notes

- Now we enter the text we want to appear in our graphic using *ImageString()*. The first parameter is the handle. Then the font (1-5), starting X ordinate, starting Y ordinate, the text itself, and finally it's color.
- Finally *ImagePng()* actually creates the PNG image. Your image does not have to be a PNG image. You can use *imagegif()* or *imagejpeg()* to create GIF and JPG images respectively.

```
<?php
header ("Content-type: image/png");
$handle = ImageCreate (130, 50) or die ("Cannot Create image");
$bg_color = ImageColorAllocate ($handle, 255, 0, 0);
$txt_color = ImageColorAllocate ($handle, 0, 0, 0);
ImageTTFText ($handle, 20, 15, 30, 40, $txt_color, "/Fonts/Quel.ttf",
"Quel");
ImagePng ($handle);
?>
```

Although most of our code has stayed the same you will notice we are now using *ImageTTFText()* instead of *ImageString()*. This allows us to choose our font, which must be in TTF format.

The first parameter is our handle, then font size, rotation, starting X, starting Y, text color, font, and finally our text. For the font parameter, you need to include the path to font file. For our example we have placed the font Quel in a folder called Fonts. As you can see from our example, we have also set the text to print at a 15 degree angle.

If your text isn't showing, you may have the path to your font wrong. Another possibility is that your Rotation, X and Y parameters are placing the text outside of the viewable area.

```
<?php
header ("Content-type: image/png");
$handle = ImageCreate (130, 50) or die ("Cannot Create image");
$bg_color = ImageColorAllocate ($handle, 255, 0, 0);
$txt_color = ImageColorAllocate ($handle, 255, 255, 255);
$line_color = ImageColorAllocate ($handle, 0, 0, 0);
ImageLine($handle, 65, 0, 130, 50, $line_color);
ImageString ($handle, 5, 5, 18, "PHP.About.com", $txt_color);
ImagePng ($handle);
?>
```

In this code, we use *ImageLine()* to draw a line. The first parameter is our handle, followed by our starting X and Y, our ending X and Y, and finally our color.

On completion, the program releases the resources associated with the image by calling *imagecolordeallocate()* and *imagedestroy()*.

## Self Assessment

Fill in the blanks:

1. The ..... library is used for dynamic image creation.
2. You can run ..... to check that GD Support is enabled.
3. .... function is used to add colors.

## 12.2 Necessary Modifications to PHP

GD graphics library in our PHP distributions eliminates the need to download and install third-party libraries. GD graphics library needs to be activated at installation time. PHP version earlier than 4.3.0, need to download the library from <http://www.boutell.com/gd/> to use it. To use of the GD library at installation time, Linux/Unix users must follow the steps below:

—with-gd

The GD path needs to be given when you download your own GD version.

—with-gd=/path/to/gd.

Windows users need to activate `php_gd2.dll` as an extension in the `php.ini` file.

### 12.2.1 Obtaining Additional Libraries

Some additional steps need to be performed in case you wish to work with files other than PNG. For JPEG, libraries and information can be found at <ftp://ftp.uu.net/graphics/jpeg/>. For PNG, libraries and information can be found at <http://www.libpng.org/pub/png/libpng.html>. For PNG files, you should also install the zlib library, found at <http://www.gzip.org/zlib/>.

After installation of these libraries, Linux/Unix users must reconfigure and rebuild PHP again. They need to add the following to the PHP configures parameters:

—with-jpeg-dir=[path to jpeg directory]

—with-png-dir=[path to PNG directory]

—with-zlib=[path to zlib directory]

After running the PHP configure program again, you need to go through the installation process. Your libraries should then be activated and ready for use.

## 12.3 Drawing a New Image

We can draw new images using the `imagecreate()` function. It create a new palette based image. It takes in two parameters, height and width and returns an image identifier representing a blank image of specified size on success.

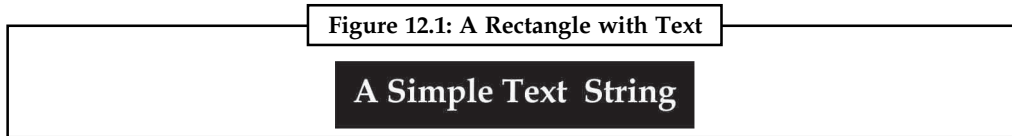


*Example:*

```
<?php
header("Content-Type: image/png");
$im = @imagecreate(110, 20) or die("Cannot Initialize new GD image
stream");
$background_color = imagecolorallocate($im, 0, 0, 0);
$text_color = imagecolorallocate($im, 233, 14, 91);
imagestring($im, 1, 5, 5, "A Simple Text String", $text_color);
imagepng($im);
imagedestroy($im);
?>
```

Output:

Notes



Source: <http://www.php.net/manual/en/function.imagecreate.php>

The function has created an image of size 110 pixels by 20 pixels. To add colours, we use the RGB system which uses decimal values from 0 to 255 for each of the red (R), green (G), and blue (B). The function used to define colors is `ImageColorAllocate()`. For example for a white color we will write,

```
$white = ImageColorAllocate ($imageOne, 255, 255, 255);
```

### 12.3.1 Drawing Lines and Shapes

There are various functions that we can use in order to create lines and shapes. Some of them are:

- `ImageRectangle()` — To draw a rectangle.
- `ImageLine()` — To draw a line.
- `ImageEllipse()` — To draw an ellipse.
- `ImageArc()` — To draw an arc
- `ImagePolygon()` — To draw a polygon.



*Example:* To create an ellipse,

```
<?php

// Create a blank image.
$image = imagecreatetruecolor(400, 300);

// Select the background color.
$bg = imagecolorallocate($image, 0, 0, 0);

// Fill the background with the color selected above.
imagefill($image, 0, 0, $bg);

// Choose a color for the ellipse.
$col_ellipse = imagecolorallocate($image, 255, 255, 255);

// Draw the ellipse.
imageellipse($image, 200, 150, 300, 200, $col_ellipse);

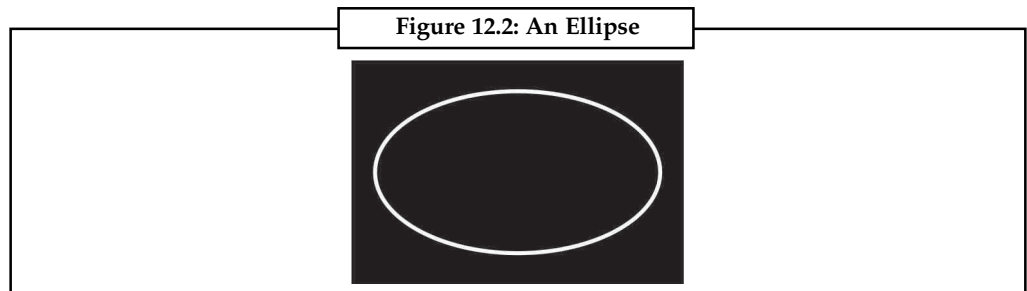
// Output the image.
header("Content-type: image/png");
imagepng($image);

?>
```



## Notes

## Output:



Source: <http://www.php.net/manual/en/function.imageellipse.php>

### 12.3.2 Using a Colour Fill

To allow adding colors to your PHP images, we have various functions to do that:

- `ImageFilledPolygon()` — To fill a polygon.
- `ImageFilledRectangle()` — To fill a rectangle.
- `ImageFilledEllipse()` — To fill an ellipse.
- `ImageFilledArc()` — To fill a partial ellipse.



*Example:* To create a polygon,

```
<?php
// set up array of points for polygon
$values = array(
 40, 50, // Point 1 (x, y)
 20, 240, // Point 2 (x, y)
 60, 60, // Point 3 (x, y)
 240, 20, // Point 4 (x, y)
 50, 40, // Point 5 (x, y)
 10, 10 // Point 6 (x, y)
);

// create image
$image = imagecreatetruecolor(250, 250);

// allocate colors
$bg = imagecolorallocate($image, 0, 0, 0);
$blue = imagecolorallocate($image, 0, 0, 255);

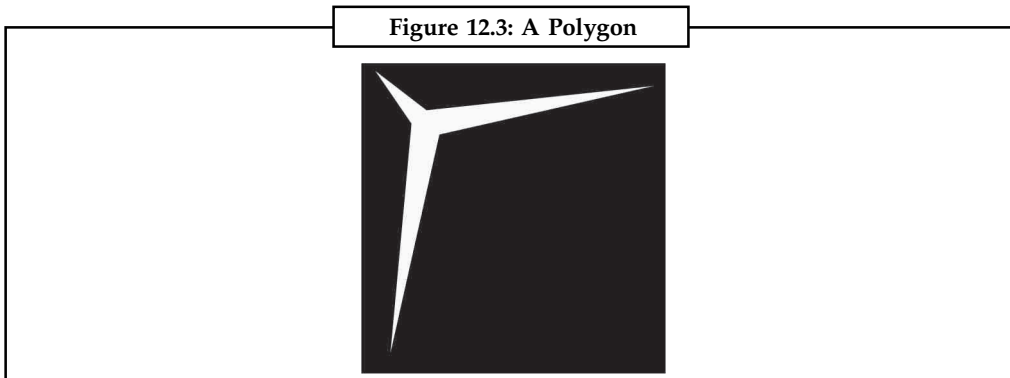
// fill the background
imagefilledrectangle($image, 0, 0, 249, 249, $bg);

// draw a polygon
imagefilledpolygon($image, $values, 6, $blue);

// flush image
header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

Output:

Notes



Source: <http://www.php.net/manual/en/function.imagefilledpolygon.php>

## Self Assessment

Fill in the blanks:

4. .... function is used to draw an arc.
5. .... function is used to fill a polygon.

## 12.4 Modifying an Existing Image

We have functions to create images in various formats. Some of them are,

`imagecreatefromgif ( string $filename )` – To create a GIF image

`imagecreatefromjpeg ( string $filename )` – To create a JPEG image

`imagecreatefrompng ( string $filename )` – To create a PNG image

These functions return `FALSE` if they fail to load the image else it returns an image identifier representing the image obtained from the given filename.

```
$im = @imagecreatefromgif($imgname);
```

### 12.4.1 Using True Type Fonts

To write text to the image using TrueType fonts you can use the `imagettftext()` function in PHP.



*Notes* This function requires both the GD library and the FreeType library.



*Example:*

```
<?php
$image = imagecreate(400,300);
$blue = imagecolorallocate($image, 0, 0, 255);
$white = ImageColorAllocate($image, 255,255,255);

if(!isset($_GET['size'])) $_GET['size'] = 44;
if(!isset($_GET['text'])) $_GET['text'] = "Hello, world!";
```

**Notes**

```
imagefttext($image, $_GET['size'], 15, 50, 200, $white, "ARIAL",
$_GET['text']);
header("content-type: image/png");
imagepng($image);
imagedestroy($image);
?>
```

*imagefttext()* takes eight parameters in total: the image resource to draw on, font size to use, angle to draw at, X co-ordinate, Y co-ordinate, colour, font file, and the text to write. A few of those parameters are exactly the same as parameters we've used in other functions, but font size in points, angle, name of font, and the text to print are all new. The X and Y co-ordinates might fool you at first because they should be set to the position you want the lower-left corner of the first character to appear.

The angle parameter works almost in the same manner as the angle parameters used in *imagefilledarc()* with the difference being that it works in the opposite direction – the angles in *imagefilledarc()* work in a clockwise direction from 3 o'clock, whereas *imagefttext()* works anti-clockwise. That is, specifying 15 as the angle will make the text rotate fifteen degrees so that it slants upwards.

The font name parameter needs to point to the TTF file you want to use. If this filename does not begin with /, PHP will automatically add ".ttf" to the end and search locally. On Unix machines, you may find PHP searches in /usr/share/fonts/truetype. As you can see in the example, "ARIAL" is specified, so ARIAL.TTF will be loaded and used for printing the text.

The final parameter for the function is the text to print, and you should be sure to specify any new lines as \n\r, not one or the other. You may find that certain fonts do not have various special characters - in this situation you will see empty boxes drawn rather than the special characters.

### 12.4.2 Drawing to Your Image

Generally create your base image using a normal picture editor, load that image using a function like *imagecreatefromjpeg()*, and modify small details with your script. Generating everything from scratch is unnecessary for most purposes, and can drain your web server resources.

### Self Assessment

State whether the following statements are true or false:

6. Windows users need to activate *php\_gd2.dll* as an extension in the *php.ini* file.
7. We can draw new images using the *image()* function.
8. *imagecreategif ( string \$filename )* is used to create a GIF image.
9. To write text to the image using TrueType fonts you can use the *text()* function in PHP.
10. *ImageRectangle()* is used to draw a rectangle.
11. The angles in *imagefilledarc()* work in a clockwise direction.

## 12.5 Image Creation from User Input

In addition to creating images from other images, and drawing images on your own, you can also create images based on user input. There's no fundamental difference in how the scripts are

created, except for the fact that you'll be gathering values from a form instead of hard-coding them into your script.

In our example below, we create an all-in-one form and script, which asks for user input for a variety of attributes ranging from image size to text and background colors, as well as a message string. The `imagestring()` function, which is used to "write" a string onto an image.



*Example:*

```

1: <?
2: if ($_POST[op] != "do") {
3: //show form
4: echo "<HTML>
5: <HEAD>
6: <TITLE>Image Creation Form</TITLE>
7: </HEAD>
8: <BODY>
9: <H1>Create an Image</H1>
10: <FORM method=\"POST\" action=\"$_SERVER[PHP_SELF]\">
11: <P>Image Size:

12: W: <input type=\"text\" name=\"w\" size=5 maxlength=5>
13: H: <input type=\"text\" name=\"h\" size=5 maxlength=5>
14: <P>Background Color:

15: R: <input type=\"text\" name=\"b_r\" size=3 maxlength=3>
16: G: <input type=\"text\" name=\"b_g\" size=3 maxlength=3>
17: B: <input type=\"text\" name=\"b_b\" size=3 maxlength=3>
18: <P>Text Color:

19: R: <input type=\"text\" name=\"t_r\" size=3 maxlength=3>
20: G: <input type=\"text\" name=\"t_g\" size=3 maxlength=3>
21: B: <input type=\"text\" name=\"t_b\" size=3 maxlength=3>
22: <P>Text String:

23: <input type=\"text\" name=\"string\" size=35>
24: <P>Font Size:

25: <select name=\"font_size\">
26: <option value=1>1</option>
27: <option value=2>2</option>
28: <option value=3>3</option>
29: <option value=4>4</option>
30: <option value=5>5</option>
31: </select>

```

**Notes**

```
32: <P>Text Starting Position:

33: X: <input type="text" name="x" size=3 maxlength=3>
34: Y: <input type="text" name="y" size=3 maxlength=3>
35: <input type="hidden" name="op" value="do">
36: <P><input type="submit" name="submit" value="create image">
37: </FORM>
38: </BODY>
39: </HTML>";
40: } else {
41: //create image
42: //create the canvas
43: $myImage = ImageCreate($_POST[w], $_POST[h]);
44: //set up some colors
45: $background = ImageColorAllocate ($myImage, $_POST[b_r], $_POST[b_g],
 $_POST[b_b]);
46: $text = ImageColorAllocate ($myImage, $_POST[t_r], $_POST[t_g],
 $_POST[t_b]);
47: // write the string at the top left
48: ImageString($myImage, "$_POST[font_size]", "$_POST[x]", "$_POST[y]",
 "$_POST[string]", $text);
49: //output the image to the browser
50: header ("Content-type: image/png");
51: ImagePNG($myImage);
52: //clean up after yourself
53: ImageDestroy($myImage);
54: }
55: ?>
```

In this basic form, you see that several fields are used to obtain image specifications. On lines 12, 13 there are fields to define the width and the height of the image we want to draw. Next, we set up fields to obtain the RGB values for a background color (lines 15,17) and a text color (lines 19, 21).

You could create drop-down list boxes containing values 0 through 255, for the red, green, and blue values. This would ensure that the user input was within the required range.

Line 23 contains a form field for the input string. This string will be drawn onto the background of the image, in the text color specified. Lines 25–31 represent a drop-down list for the selection of the font size. There are three sizes, 1 through 5, for the default fixed-width font.

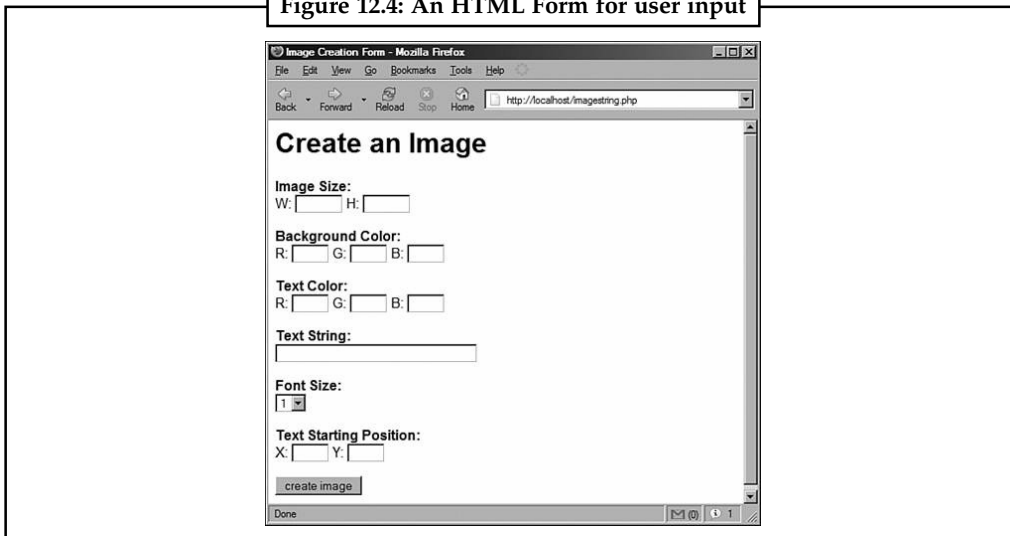
You can specify fonts using the `imagedloadfont()` and `imagefttext()` functions.

Finally, lines 33 and 34 allow you to define the text starting position. The upper left corner would be X position 0, Y position 0; 10 increments downward would be Y position 10, 10 increments to the right would be X position 10, and so forth.

Output:

Notes

Figure 12.4: An HTML Form for user input



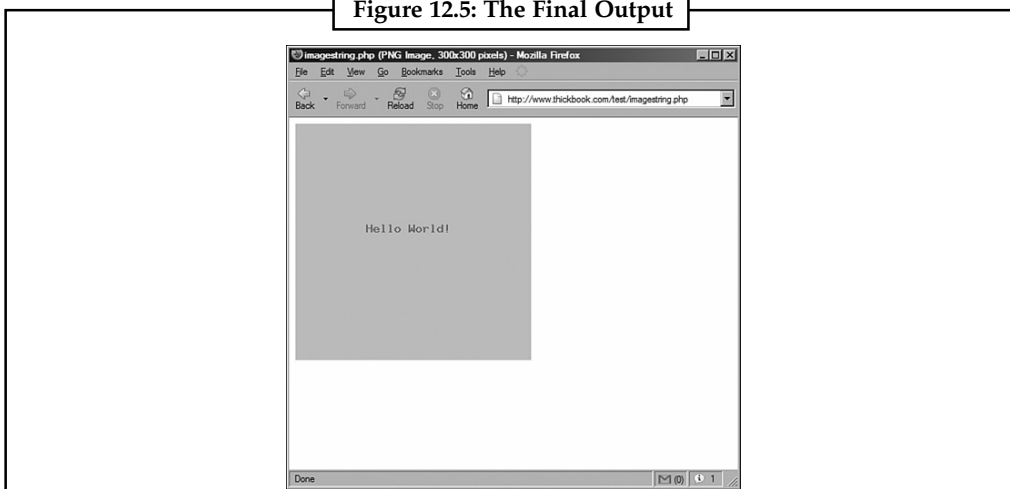
Source: [http://www.yaldex.com/php\\_tutorial\\_3/ch14lev1sec6.html](http://www.yaldex.com/php_tutorial_3/ch14lev1sec6.html)

The majority of lines 40–55 you’ve already seen before, only this time we use extracted elements from the `$_POST` superglobal to take the place of hard-coded values. In line 43 we use the width and height values from the form, to set up the initial image. Lines 45 and 54 define two colors, `$background` and `$text`, using the appropriate RGB values provided by the form. The colors weren’t given actual color names in this script, because we don’t know what the user input would create we could call the color `$red`. Instead, we simply name our colors after their purpose, not their appearance.

Line 48 represents the only new item in this script, the use of the `imagestring()` function. The six parameters for this function are the image stream (`$myImage`), the font size (`$_POST[font_size]`), the starting X and Y positions (`$_POST[x]` and `$_POST[y]`), the string to be drawn (`$_POST[string]`) and the color in which to draw it (`$text`). Lines 50, 51 output the image to the browser and line 56 destroys and cleans up the image creation process.

If we save this file as `imagecreate.php`, place it in the document root of the Web server, and fill out the form.

Figure 12.5: The Final Output



Source: [http://www.yaldex.com/php\\_tutorial\\_3/ch14lev1sec6.html](http://www.yaldex.com/php_tutorial_3/ch14lev1sec6.html)

## Notes

## Self Assessment

Fill in the blanks:

12. The ..... function, which is used to “write” a string onto an image.
13. The RGB values for color take the values ..... to .....
14. You can specify fonts using the `imagefont()` and `imagefttext()` functions.
15. `imagestring()` function takes ..... parameters.



Case Study

### The Future of PHP

PHP is already popular, used in millions of domains (according to Netcraft), supported by most ISPs and used by household-name Web companies like Yahoo! The upcoming versions of PHP aim to add to this success by introducing new features that make PHP more usable in some cases and more secure in others. Are you ready for PHP V6? If you were upgrading tomorrow, would your scripts execute just fine or would you have work to do? This article focuses on the changes for PHP V6 — some of them back-ported to versions PHP V5.x — that could require some tweaks to your current scripts.

If you’re not using PHP yet and have been thinking about it, take a look at its latest features. These features, from Unicode to core support for XML, make it even easier for you to write feature-filled PHP applications.

#### *New PHP V6 Features*

PHP V6 is currently available as a developer snapshot, so you can download and try out many of the features and changes listed in this article.

#### *Improved Unicode Support*

Much improved for PHP V6 is support for Unicode strings in many of the core functions. This new feature has a big impact because it will allow PHP to support a broader set of characters for international support. So, if you’re a developer or architect using a different language, such as the Java™ programming language, because it has better internationalization (i18n) support than PHP, it’ll be time to take another look at PHP when the support improves.

Because you can download and use a developer’s version of PHP V6 today, you will see some functions already supporting Unicode strings.

#### *Namespaces*

*Namespaces* are a way of avoiding name collisions between functions and classes without using prefixes in naming conventions that make the names of your methods and classes unreadable. So by using namespaces, you can have class names that someone else might use, but now you don’t have to worry about running into any problems. Listing 1 provides an example of a namespace in PHP.

You won’t have to update or change anything in your code because any PHP code you write that doesn’t include namespaces will run just fine. Because the namespaces feature appears to be back-ported to V5.3 of PHP, when it becomes available, you can start to introduce namespaces into your own PHP applications.

*Contd...*

**Web 2.0 Features**

Depending on how you use PHP and what your scripts look like now, the language and syntax differences in PHP V6 may or may not affect you as much as the next features, which are those that directly allow you to introduce Web 2.0 features into your PHP application.

**SOAP**

SOAP is one of the protocols that Web services “speak” and is supported in quite a few other languages, such as the Java programming language and Microsoft® .NET. Although there are other ways to consume and expose Web services, such as Representational State Transfer (REST), SOAP remains a common way of allowing different platforms to have interoperability. In addition to SOAP modules in the PHP Extension and Application Repository (PEAR) library, a SOAP extension to PHP was introduced in V5. This extension wasn’t enabled by default, so you have to enable the extension or hope your ISP did. In addition, PEAR packages are available that allow you to build SOAP clients and servers, such as the SOAP package.

Unless you change the default, the SOAP extension will be enabled for you in V6. These extensions provide an easy way to implement SOAP clients and SOAP servers, allowing you to build PHP applications that consume and provide Web services.

If SOAP extensions are on by default, that means you won’t have to configure them in PHP. If you develop PHP applications and publish them to an ISP, you may need to check with your ISP to verify that SOAP extensions will be enabled for you when they upgrade.

**XML**

As of PHP V5.1, XMLReader and XMLWriter have been part of the core of PHP, which makes it easier for you to work with XML in your PHP applications. Like the SOAP extensions, this can be good news if you use SOAP or XML because PHP V6 will be a better fit for you than V4 out of the box.

The XMLWriter and XMLReader are stream-based object-oriented classes that allow you to read and write XML without having to worry about the XML details.

**Questions**

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

**Notes**

Source: <http://www.ibm.com/developerworks/opensource/library/os-php-future/>

**12.6 Summary**

- PHP is not limited to creating just HTML output. It can also be used to create and manipulate image files in a variety of different image formats.
- The GD library is used for dynamic image creation.
- You can run `phpinfo()` to check that GD Support is enabled.
- PHP needs to have FreeType support compiled into it in order to enable the use of True Type Font Support.
- `ImageCreate()` creates an image by taking height and width as parameters.
- `ImageColorAllocate()` is used to add colors.



**Notes**

- We enter the text we want to appear in our graphic using ImageString().
- On completion, the program releases the resources associated with the image by calling imagecolordeallocate() and imagedestroy().
- Windows users need to activate php\_gd2.dll as an extension in the php.ini file.
- We can draw new images using the imagecreate() function.
- To allow adding colors to your PHP images, we have various functions to do that.
- imagecreatefromgif ( string \$filename ) – To create a GIF image.
- To write text to the image using TrueType fonts you can use the imagettftext() function in PHP.
- PHP can also create images based on user input.

### **12.7 Keywords**

*FreeType Support:* Need to be compiled into PHP in order to enable the use of True Type Font Support.

*GD Library:* It is used for dynamic image creation.

*ImageColorAllocate():* It is used to add colors.

*ImageCreate():* It creates an image by taking height and width as parameters.

*ImagePng():* It creates the PNG image.

*ImageRectangle():* It is used to draw a rectangle.

*ImageString():* It is used to enter text in our graphic.

*phpinfo():* It is used to check that GD Support is enabled.

### **12.8 Review Questions**

1. Give a code to explain image creation.
2. What are the changes needed to embed images in PHP?
3. Why do we need GD library?
4. How do we create an arc?
5. Which functions do we use for color fills?
6. How do we use true type fonts in PHP?
7. Write a short note on ImageRectangle() function.
8. How do we create a wave image in PHP?
9. Why do we need to modify an existing image?
10. How do we create an image by user input?

**Answers: Self Assessment****Notes**

1. GD
2. phpinfo()
3. imagecolorallocate()
4. imagedrawarc()
5. imagefilledpolygon()
6. True
7. False
8. False
9. False
10. True
11. True
12. imagestring()
13. 0, 255
14. imageloadfont() , imagegetttfext()
15. 6

**12.9 Further Readings***Books*

Adam Trachtenberg, PHP Cookbook.

David Powers, PHP Solutions: Dynamic Web Design Made Easy.

Jason Lengstorf, PHP for Absolute Beginners.

Luke Welling and Laura Thomson, PHP and MySQL Web Development.

*Online links*

[http://php.about.com/od/advancedphp/ss/gd\\_library\\_4.htm](http://php.about.com/od/advancedphp/ss/gd_library_4.htm)

<http://php.net/manual/en/ref.image.php>

<http://www.thesitewizard.com/php/create-image.shtml>

<http://www.tuxradar.com/practicalphp/11/2/6>

## Unit 13: Stored Procedure

### CONTENTS

Objectives

Introduction

13.1 Transactions

13.1.1 Properties of Transactions

13.1.2 COMMIT and ROLLBACK

13.1.3 Row-Level Locking

13.2 Stored Procedures

13.2.1 Implementation

13.2.2 Other Uses

13.2.3 Comparison with Dynamic SQL

13.2.4 Comparison with Functions

13.2.5 Disadvantages

13.2.6 Why Use Stored Procedures?

13.2.7 Creating a Stored Procedure

13.2.8 Calling a Stored Procedure

13.2.9 Specifying Parameters

13.2.10 Data Retrieval

13.2.11 Inserting Data Using Parameters

13.2.12 Benefits of Stored Procedures

13.3 Summary

13.4 Keywords

13.5 Review Questions

13.6 Further Readings

### Objectives

After studying this unit, you will be able to:

- Discuss about Transactions
- Explain Stored Procedures

### Introduction

A stored procedure is a subroutine available to applications that access a relational database system. A stored procedure is actually stored in the database data dictionary. Typical use for stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures can consolidate and centralize logic that was originally implemented in applications. Extensive or complex processing that requires execution of several SQL statements is moved into stored procedures, and all applications call the procedures.

One can use nested stored procedures by executing one stored procedure from within another. Stored procedures are similar to user-defined functions (UDFs). The major difference is that UDFs can be used like any other expression within SQL statements, whereas stored procedures must be invoked using the CALL statement

CALL procedure(...)

or

EXECUTE procedure(...)

Stored procedures may return result sets, *i.e.* the results of a SELECT statement. Such result sets can be processed using cursors, by other stored procedures, by associating a result set locator, or by applications. Stored procedures may also contain declared variables for processing data and cursors that allow it to loop through multiple rows in a table. Stored procedure flow control statements typically include IF, WHILE, LOOP, REPEAT, and CASE statements, and more. Stored procedures can receive variables, return results or modify variables and return them, depending on how and where the variable is declared.

### 13.1 Transactions

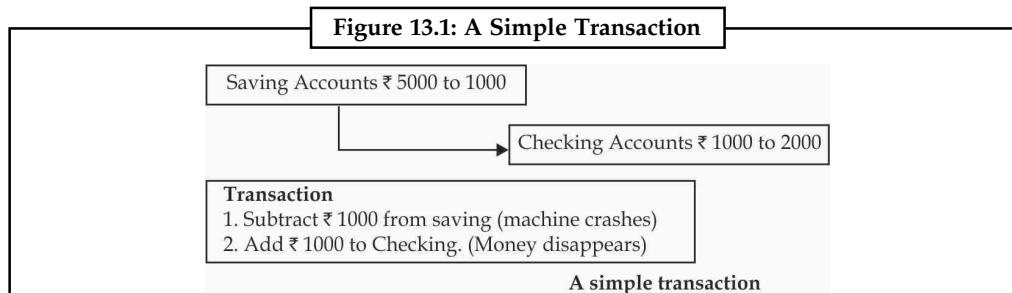
Concurrent execution of user programs is essential for good DBMS performance. Because disk accesses are frequent, and relatively slow, it is important to keep the CPU humming by working on several user programs concurrently. A user's program may carry out many operations on the data retrieved from the database, but the DBMS is only concerned about what data is read/written from/to the database. A transaction is the DBMS's abstract view of a user program: a sequence of reads and writes.



*Example:* A transfer of money from one bank account to another requires two changes to the database both must succeed or fail together. You are working on a system for a bank. A customer goes to the ATM and instructs it to transfer Rs.1000 from savings to a checking account. This simple transaction requires two steps:

- Subtracting the money from the savings account balance.
- Adding the money to the checking account balance.

The code to create this transaction will require two updates to the database. For example, there will be two SQL statements: one UPDATE command to decrease the balance in savings and a second UPDATE command to increase the balance in the checking account. You have to consider what would happen if a machine crashed between these two operations. The money has already been subtracted from the savings account will not be added to the checking account. It is lost. You might consider performing the addition to checking first, but then the customer ends up with extra money, and the bank loses. The point is that both changes must be made successfully. Thus, a transaction is defined as a set of changes that must be made together.



Source: <http://ecomputernotes.com/database-system/rdbms/transaction>

### 13.1.1 Properties of Transactions

In the context of transaction processing, the acronym *ACID* refers to the four key properties of a transaction: atomicity, consistency, isolation and durability.

- **Atomicity:** All changes to data are performed as if they are a single operation. That is, all the changes are performed, or none of them are.



*Example:* In an application that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account.

- **Consistency:** Data is in a consistent state when a transaction starts and when it ends.



*Example:* In an application that transfers funds from one account to another, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction.

- **Isolation:** The intermediate state of a transaction is invisible to other transactions. As a result, transactions that run concurrently appear to be serialized.



*Example:* In an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither.

- **Durability:** After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure.



*Example:* In an application that transfers funds from one account to another, the durability property ensures that the changes made to each account will not be reversed.

### 13.1.2 COMMIT and ROLLBACK

The transaction manager uses COMMIT and ROLLBACK operations for ensuring atomicity of transactions.

The COMMIT operation indicates successful completion of a transaction which means that the database is in a consistent state and all updates made by the transaction can now be made permanent. If a transaction successfully commits, then the system will guarantee that its updates will be permanently installed in the database even if the system crashes immediately after the COMMIT.



*Example:*

Consider Customers table is having following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Source: <http://www.tutorialspoint.com/sql/sql-transactions.htm>

Following is the example which would delete records from the table having age = 25, and then COMMIT the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
 WHERE AGE = 25;
SQL> COMMIT;
```

As a result, two rows from the table would be deleted and SELECT statement would produce following result:

**Table 13.2: Customers Table Post the Transaction**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Source: <http://www.tutorialspoint.com/sql/sql-transactions.htm>

The ROLLBACK operation on the other hand indicates that the transaction has been unsuccessful which means that all updates done by the transaction till then need to be undone to bring the database back to a consistent state. To help undoing the updates once done, a system log or journal is maintained by the transaction manager. The before- and after-images of the updated tuples are recorded in the log.



*Example:* Consider the Table 13.1.

Following is the example which would delete records from the table having age = 25, and then ROLLBACK the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
 WHERE AGE = 25;
SQL> ROLLBACK;
```

As a result, delete operation would not impact the table and SELECT statement would produce following result:

**Table 13.3 : Result of Rollback Operation**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Source: <http://www.tutorialspoint.com/sql/sql-transactions.htm>

### 13.1.3 Row-Level Locking

Using ROWLOCK politely asks SQL Server to only use row-level locks. You can use this in SELECT, UPDATE, and DELETE statements, but I only use it in UPDATE and DELETE statements. You'd think that an UPDATE in which you specify the primary key would always cause a row lock, but when SQL Server gets a batch with a bunch of these, and some of them happen to be in the same page (depending on this situation, this can be quite likely, e.g. updating all files in a folder, files which were created at pretty much the same time), you'll see page locks, and bad things will happen. And if you don't specify a primary key for an UPDATE or DELETE, there's no reason the database wouldn't assume that a lot won't be affected, so it probably goes right to page locks, and bad things happen. By specifically requesting row-level locks, these problems are avoided.



*Caution* If you are wrong and lots of rows are affected.

### Self Assessment

State whether the following statements are true or false:

1. Atomicity means that all changes to data are performed on different machines.
2. Consistency implies that data is in a consistent state when a transaction starts and when it ends.
3. Isolation implies the use of tools to separate the concurrent transactions.
4. Durability doesn't imply saving data after a system failure.
5. The COMMIT operation indicates successful completion of a transaction.

## 13.2 Stored Procedures

A **stored procedure** or in simple a **proc** is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage. It may or may not return any value.

We can pass parameters to procedures in three ways:

1. IN-parameters
2. OUT-parameters
3. IN OUT-parameters

The general syntax for a procedure is,

```
CREATE [OR REPLACE] PROCEDURE proc_name [list of parameters]
IS
 Declaration section
BEGIN
 Execution section
EXCEPTION
 Exception section
END;
```

### 13.2.1 Implementation

Stored procedure implementation depends on the database used. It is supported by almost all major database vendors. As per the database used, stored procedures can be implemented in a variety of programming languages, for example SQL, Java, C, or C++.

**Table 13.4: Databases and languages that support Stored Procedures**

Database system	Implementation language
CUBRID	Java
DB2	SQL PL (close to the SQL/PSM standrad) or Java
Firebird	PSQL (Fyracle also supports portions of Oracle's PL/SQL)
Informix	SPL or Java
Microsoft SQL Server	Transact-SQL and various .NET Framework languages
MySQL	Own stored produres, closely adhering to SQL/PSM standard.
Oracle	PL/SQL or Java
PostgreSQL	PL/pgSQL, can also use own function languages such as pl/perl or pl/php
Sybase ASE	Transact-SQL

Source: [http://en.wikipedia.org/wiki/Stored\\_procedure](http://en.wikipedia.org/wiki/Stored_procedure)

### 13.2.2 Other Uses

Stored procedures can also be used to control transaction management, can be invoked from a database trigger or a condition handler.

### 13.2.3 Comparison with Dynamic SQL

For improved performance compared to dynamic SQL, application designers can use stored procedures as an alternative where the application requirements allow it.

There are similarities between dynamic SQL and stored procedures:

- Creating a stored procedure is analogous to preparing a dynamic SQL statement.
- A stored procedure's input parameters serve the same purpose as dynamic parameter markers.
- Executing a stored procedure is equivalent to executing a prepared statement.

Stored procedures and dynamic SQL prepared statements offer identical functionality, with the following exceptions:

- Dynamic SQL allows retrieval of a prepared statement's parameter formats, while stored procedures do not.
- The format for stored procedure results cannot easily be determined programmatically without executing the procedure. Dynamic SQL allows retrieval of a prepared statement's result column formats without executing the statement.
- User-created stored procedures are permanent database objects, while prepared statements are automatically de-allocated when the user disconnects from the server.

A dynamic SQL statement can be replaced by a stored procedure that returns the same results.



**Notes**

*Example:* The following dynamic SQL statement queries the *pubs2..titles* table for books of a certain type in a certain price range:

```
select * from pubs2..titles
where type = ?
and price between ? and ?
```

Here, the dynamic SQL statement has dynamic parameter markers (?) for a *type* value and two *price* values.

You can create an equivalent stored procedure as follows:

```
create proc titles_type_pricerange
@type char(12),
@price1 money,
@price2 money
as
select * from titles
where
type = @type
and price between @price1 and @price2
```

When executed with the same input parameter values, the prepared statement and the stored procedure return the same rows. In addition, the stored procedure returns a return status result.

### 13.2.4 Comparison with Functions

The major differences between stored procedures and functions are summarized below:

- Procedure can return zero or n values whereas function can return one value which is mandatory.
- Procedures can have input/output parameters for it whereas functions can have only input parameters.
- Procedure allows select as well as DML statement in it whereas function allows only select statement in it.
- Functions can be called from procedure whereas procedures cannot be called from function.
- Exception can be handled by try-catch block in a procedure whereas try-catch block cannot be used in a function.
- We can go for transaction management in procedure whereas we can't go in function.
- Procedures cannot be utilized in a select statement whereas function can be embedded in a select statement.
- UDF (User Defined Functions) can be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section where as Stored procedures cannot be.
- UDFs that return tables can be treated as another rowset. This can be used in JOINS with other tables.
- In-line UDF's can be thought of as views that take parameters and can be used in JOINS and other Rowset operations.

### 13.2.5 Disadvantages

The problems with stored procedures are:

- Stored procedures make the database server high load in both memory for and processors. Instead of being focused on the storing and retrieving data, you could be asking the database server to perform a number of logical operations or a complex of business logic which is not the well designed in database server.
- Stored procedure only contains declarative SQL so it is very difficult to write a procedure with complexity of business logic like other languages in application layer such as Java, C#, C++...
- You cannot debug stored procedure in almost RDBMSs and in MySQL also. There are some workarounds on this problem but it still not good enough to do so.

### 13.2.6 Why Use Stored Procedures?

The need to use stored procedures lies in the fact that it increases the performance of SQL queries. When we write a query before execution the SQL Server compiler runs three steps. In first step compiler checks the syntax of query then in the second step compiler finds the best way to execute the query and finally in third step query is executed. So when we write in line queries these three step are always run , but for Stored Procedure the below three step run only once and first two step results are cached for future. So when you execute the stored procedure next time it just executes the last step thus increasing performance.

### 13.2.7 Creating a Stored Procedure

To create a stored procedure using an Enterprise Manager, follow the steps below:

1. Expand a server group, and then expand a server.
2. Expand Databases, and then expand the database in which to create the procedure.
3. Right-click Stored Procedures, and then click New Stored Procedure.
4. Enter the text of the stored procedure. Press TAB to indent the text of a stored procedure. Press CTRL+TAB to exit the text box, or click an appropriate button.
5. To check the syntax, click Check Syntax.
6. To set the permissions, click Permissions.



*Notes* Validate all user input. Do not concatenate user input before validating it. Never execute a command constructed from invalidated user input.

### 13.2.8 Calling a Stored Procedure

ADO.NET allows calling a stored procedure just like a standard SQL statement. Just follow the steps below:

1. Create a new C# Windows Application project.
2. From the Toolbox, drag and drop a DataGrid onto the Form.

**Notes**

3. Double-click on the Form to generate the Form\_Load event handler.
4. Add "using System.Data.SqlClient" at the top of the file.
5. Add the code as shown in the example given below:



*Example:* SelectEmployee() method to execute SP using C#.NET

```
public void SelectEmployee()
{
 SqlConnection con = null;
 SqlDataReader rd = null;

 try
 {
 // Create and Open the SQL server connection object
 con = new SqlConnection("Database Connection string");
 con.Open();

 // Create a command object and specify the Stored Procedure name and
 connection as well
 SqlCommand cmd = new SqlCommand("ProcEmployeeDetails", con);

 // Set the command object
 cmd.CommandType = CommandType.StoredProcedure;

 // Add parameter and value
 cmd.Parameters.Add(new SqlParameter("@EmployeeID", 550));

 // Execute the command
 rd = cmd.ExecuteReader();
 while (rd.Read())
 {
 Response.Write(rd["Name"].ToString(), rd["Age"].ToString(), rd["Designation"].ToString());
 }
 }
}
```

### 13.2.9 Specifying Parameters

Using parameters for stored procedures is the same as using parameters for query string commands.



*Example:*

```
// 1. create a command object identifying the stored procedure
SqlCommand cmd = new SqlCommand("CustOrderHist", conn);

// 2. set the command object so it knows to execute a stored procedure
cmd.CommandType = CommandType.StoredProcedure;
```

```
// 3. add parameter to command, which will be passed to the stored
procedure
 cmd.Parameters.Add(new SqlParameter("@CustomerID", custId));
```

The SqlCommand constructor above specifies the name of a stored procedure, *CustOrderHist*, as its first parameter. This particular stored procedure takes a single parameter, named *@CustomerID*. Therefore, we must populate this parameter using a SqlParameter object. The name of the parameter passed as the first parameter to the SqlParameter constructor must be spelled exactly the same as the stored procedure parameter. Then execute the command the same as you would with any other SqlCommand object.

### 13.2.10 Data Retrieval

We can retrieve data from a stored procedure through a data reader or a data adapter in ADO.net.



*Example:*

```
public DataTable GetData(SqlCommand cmd)
{
 DataTable dt = new DataTable();
 String strConnString
 =System.Configuration.ConfigurationManager.ConnectionStrings["conString"].
 ConnectionString;
 SqlConnection con = new SqlConnection(strConnString);
 cmd.Connection = con;
 SqlDataAdapter sda = new SqlDataAdapter();
 cmd.CommandType = CommandType.StoredProcedure;
 try
 {
 con.Open();
 sda.SelectCommand = cmd;
 sda.Fill(dt);
 }
 catch
 {
 return dt;
 }
 finally
 {
 con.Close();
 cmd.Dispose();
 sda.Dispose();
 }
 return dt;
}
```

### 13.2.11 Inserting Data Using Parameters

Stored procedures can accept data as input parameters and can return data as output parameters, result sets, or return values. The example below illustrates how ADO.NET sends and receives input parameters, output parameters, and return values. The example inserts a new record into a table where the primary key column is an identity column in a SQL Server database.



*Notes* If you are using SQL Server stored procedures to edit or delete data using a `SqlDataAdapter`, make sure that you do not use `SET NOCOUNT ON` in the stored procedure definition. This causes the rows affected count returned to be zero, which the `DataAdapter` interprets as a concurrency conflict. In this event, a `DBConcurrencyException` will be thrown.



*Example:*

The sample uses the following stored procedure to insert a new category into the `NorthwindCategories` table. The stored procedure takes the value in the `CategoryName` column as an input parameter and uses the `SCOPE_IDENTITY()` function to retrieve the new value of the identity field, `CategoryID`, and return it in an output parameter. The `RETURN` statement uses the `@@ROWCOUNT` function to return the number of rows inserted.

```
CREATE PROCEDURE dbo.InsertCategory
 @CategoryName nvarchar(15),
 @Identity int OUT
AS
INSERT INTO Categories (CategoryName) VALUES(@CategoryName)
SET @Identity = SCOPE_IDENTITY()
RETURN @@ROWCOUNT
```

The following code example uses the `InsertCategory` stored procedure shown above as the source for the `InsertCommand` of the `SqlDataAdapter`. The `@Identity` output parameter will be reflected in the `DataSet` after the record has been inserted into the database when the `Update` method of the `SqlDataAdapter` is called. The code also retrieves the return value.



*Notes* When using the `OleDbDataAdapter`, you must specify parameters with a `ParameterDirection` of `ReturnValue` before the other parameters.

```
private static void ReturnIdentity(string connectionString)
{
 using (SqlConnection connection =
 new SqlConnection(connectionString))
 {
 // Create a SqlDataAdapter based on a SELECT query.
 SqlDataAdapter adapter = new SqlDataAdapter(
 "SELECT CategoryID, CategoryName FROM dbo.Categories",
```

## Notes

```
connection);

// Create a SqlCommand to execute the stored procedure.
adapter.InsertCommand = new SqlCommand("InsertCategory", connection);
adapter.InsertCommand.CommandType = CommandType.StoredProcedure;

// Create a parameter for the ReturnValue.
SqlParameter parameter = adapter.InsertCommand.Parameters.Add(
"@RowCount", SqlDbType.Int);
parameter.Direction = ParameterDirection.ReturnValue;

// Create an input parameter for the CategoryName.
// You do not need to specify direction for input parameters.
adapter.InsertCommand.Parameters.Add(
"@CategoryName", SqlDbType.NChar, 15, "CategoryName");

// Create an output parameter for the new identity value.
parameter = adapter.InsertCommand.Parameters.Add(
"@Identity", SqlDbType.Int, 0, "CategoryID");
parameter.Direction = ParameterDirection.Output;

// Create a DataTable and fill it.
DataTable categories = new DataTable();
adapter.Fill(categories);

// Add a new row.
DataRow categoryRow = categories.NewRow();
categoryRow["CategoryName"] = "New Beverages";
categories.Rows.Add(categoryRow);
// Update the database.
adapter.Update(categories);

// Retrieve the ReturnValue.
Int32 rowCount =
(Int32)adapter.InsertCommand.Parameters["@RowCount"].Value;

Console.WriteLine("ReturnValue: {0}", rowCount.ToString());
Console.WriteLine("All Rows:");
foreach (DataRow row in categories.Rows)
{
{
Console.WriteLine(" {0}: {1}", row[0], row[1]);
```

Notes }  
}  
}  
}

### 13.2.12 Benefits of Stored Procedures

Stored procedures offer several distinct advantages over embedding queries in your Graphical User Interface (GUI).

- Stored procedures are modular. This is a good thing from a maintenance standpoint. When query trouble arises in your application, you would likely agree that it is much easier to troubleshoot a stored procedure than an embedded query buried within many lines of GUI code.
- Stored procedures are tunable. By having procedures that handle the database work for your interface, you eliminate the need to modify the GUI source code to improve a query's performance. Changes can be made to the stored procedures—in terms of join methods, differing tables, etc.—that are transparent to the front-end interface.
- Stored procedures abstract or separate server-side functions from the client-side. It is much easier to code a GUI application to call a procedure than to build a query through the GUI code.
- Stored procedures are usually written by database developers/administrators. Persons holding these roles are usually more experienced in writing efficient queries and SQL statements. This frees the GUI application developers to utilize their skills on the functional and graphical presentation pieces of the application.

### Self Assessment

Fill in the blanks:

6. A ..... is a named PL/SQL block which performs one or more specific task.
7. We can pass parameters to procedures in three ways: ....., ..... and .....
8. .... allows retrieval of a prepared statement's parameter formats, while stored procedures do not.
9. User-created stored procedures are ..... database objects.
10. Procedure can return zero or n values whereas function can return ..... value which is mandatory.
11. Procedures can have input/output parameters for it whereas functions can have only ..... parameters.
12. Procedure allows select as well as DML statement in it whereas function allows only ..... statement in it.
13. Exception can be handled by try-catch block in a procedure whereas try-catch block cannot be used in a .....
14. Procedures cannot be utilized in a ..... statement whereas function can be embedded in a select statement.
15. Press ..... to indent the text of a stored procedure.



Case Study

## Load Balancing High Transaction Volume Databases

Recently, while working on a SQL server optimization project, we had the opportunity to look into one interesting problem. We had a huge database (to the tune of 800GB) which was being hammered with approximately 30,000 transactions per second. Database load was expected to grow by a factor of 100 in coming days and the idea was to devise a solution which could handle that load. This was a SQL server 2005 enterprise edition database hosted on an 8 processor fifth Generation server. We wouldn't say this server was on its knees with this load but yes there were wait times longer than expected and to add to that there were times when data traffic suddenly went up significantly and in those times DB was not able to keep up.

Though this is not a very common scenario in many of the modern day applications out there but this definitely is a hallmark of databases handling loads from specific industries like banking. Applications intended for these industries normally have huge volume of small database transactions. In this case study, we present one of the approaches you can take to handle a scenario such as this.

### *Introducing Broker Hub*

To demonstrate the problem which has these type of database requirements, let's use the example of a Broker Hub – a stock broking hub. Stock broking applications have very high volume of small database transactions and also there are spurts in database activity depending on market conditions. For simplicity sake, let's assume that we were at a point when database design and usage pattern for Broker hub database was in the most optimal state.

### *First Choice – Scale up*

So to optimize Broker Hub further, we had a number of ideas and first choice was obviously to increase the hardware capacity. Increasing the hardware capacity did help the case. We could handle upwards of 50,000 transactions a second by moving to a better system with 16 processors and a SAN array of high speed disks. But above 50,000 in our load environments, we could still see the database to be the bottleneck.

Next obvious idea was to try SQL Server 2008 which has support for performance optimization features like advanced compression (reducing the overall disk IO) and support for virtually unlimited number of objects (2,147,483,647) and database size (524,272 TB). Again we could see the difference. Without enabling features like compression we could achieve a bigger number of around 70,000 transactions per second.

### *Would it Work? Probably not long-term*

Probably enabling compression and using other features to optimize performance would have resulted in a higher figure but the problem here was that there was a limit to this. Adding hardware or moving to newer version of a database (or even to a different database) wouldn't have given us the virtually unlimited (100 times 30,000 transactions per minute) capacity we were looking at. Obviously we needed some way to deploy more than one server to split the load and thus increase the capacity to handle very high transaction loads. In simpler terms we needed a scale out solution instead of scale up for our database. Looking around for out of the box solutions in the market didn't help, simply because there aren't many scale-out solutions available in the market to load balance SQL server.

Contd...

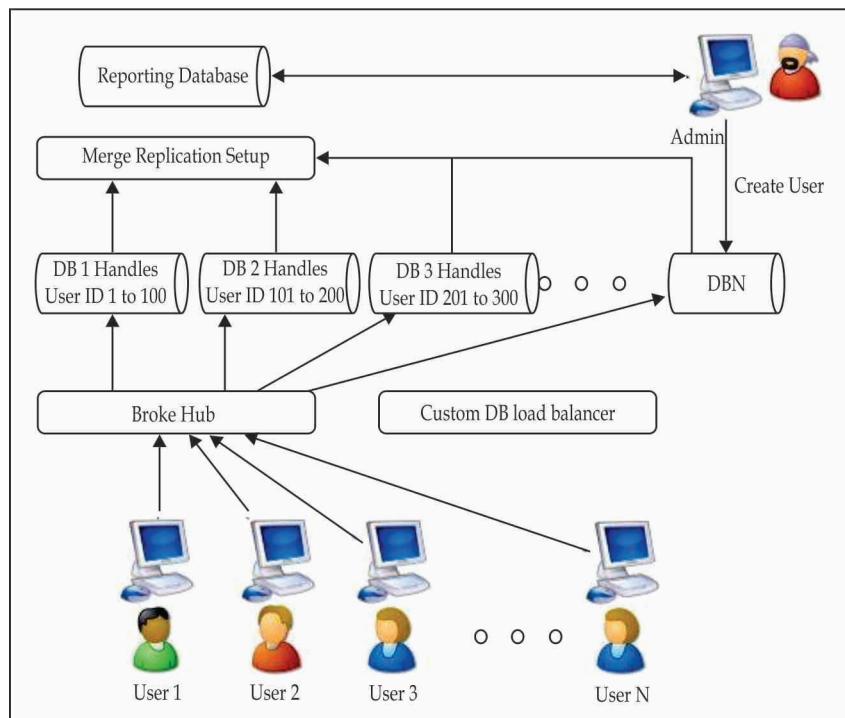
Notes



Notes

Oracle has launched such a solution but even that requires syncing between different servers in cluster which takes up a lot of network bandwidth thus reducing the overall effectiveness of the solution.

**Scale out – Approach:** After a lot of brainstorming sessions, it was decided that it was time to create our own scale out solution. The idea was to create a design which could help us scale out as our user base grew but at the same time being able to handle sudden increases in transaction volumes. But as it happens all the time, we didn't have too much money to be spent on this. We looked at various existing products such as SharePoint to see how they stored their data and came up with a very simple first draft of the scale out solution. It looked very similar to way SharePoint does load balancing for its data stores. We had a cluster of database servers connected to the Broker Hub. Each database was configured to handle a set of users with specific user IDs and thus held data only for those users. Merge replication was used to replicate data from all databases to a central database which was used for all reporting. Here's how it looked like:



Broker Hub front end had to be modified a bit in terms of process flow. In this case, each user logging into Broker Hub had to be connected to a specific database based on his user id. The task of identifying which database has the information related to the user trying to log in was handled by a new module added to the application called DB Load balancer. The process flow in this approach looked like this:

- User foo with User ID (n) logs in.
- Application calls Custom DB load balancer to find out that all data specific to user foo is on database server DB 2.
- Application creates a connection to DB 2 on behalf of this user.
- All user transactions from User 1 are directed to DB 2 thereafter.
- Periodically user data is synchronized to the master database.

Contd...

**Design Decisions for this Approach:** This approach was pretty simple to implement and could achieve the results we were looking for. But there were a few practical issues:

- We have to introduce concept of ID buckets. So every transaction table on each server was assigned unique bucket from which it could allocate Ids. With this we overcame the problem of how to maintain unique IDs.
- All the masters would be replicated using transaction with update option. So a change on one gets replicated all over.
- Adding new users was not as simple as before. Every database was configured for a specific set of users and this set increased sequentially. This meant that we always had to add the new user in the last server available and if last server was full then we had to deploy a new database even for a single user (and probably even before the last server was at its full capacity). This was accepted as an acceptable fact as this would happen once a while.
- Any admin report had to combine the data from all the servers to be useful which meant an additional job to aggregate all the data. This was accepted as a design reality.

**Conclusion:** So we could see very clearly that we were achieving the desired performance figures for this approach. Our aim was to find out how best we could achieve load balancing capabilities for the specific database load our application was creating and that was achieved. It actually became a life saver for us in more than one situation.

#### Questions

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

Source: <http://www.nagarro.com/net/case-study-load-balancing-high-transaction-volume-databases/>

### 13.3 Summary

- A stored procedure is a subroutine available to applications that access a relational database system.
- A transaction is the DBMS's abstract view of a user program: a sequence of reads and writes.
- ACID refers to the four key properties of a transaction: atomicity, consistency, isolation, and durability.
- The COMMIT operation indicates successful completion of a transaction which means that the database is in a consistent state and all updates made by the transaction can now be made permanent.
- The ROLLBACK operation on the other hand indicates that the transaction has been unsuccessful which means that all updates done by the transaction till then need to be undone to bring the database back to a consistent state.
- A stored procedure or in simple a proc is a named PL/SQL block which performs one or more specific task.
- We can pass parameters to procedures in three ways. IN-parameters, OUT-parameters and IN OUT-parameters.

**Notes**

- Dynamic SQL allows retrieval of a prepared statement's parameter formats, while stored procedures do not.
- ADO.NET allows calling a stored procedure just like a standard SQL statement.

### **13.4 Keywords**

*Atomicity:* All changes to data are performed as if they are a single operation.

*COMMIT:* It operation indicates successful completion of a transaction.

*Consistency:* Data is in a consistent state when a transaction starts and when it ends.

*Durability:* After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure.

*Isolation:* The intermediate state of a transaction is invisible to other transactions.

*ROLLBACK:* It indicates that the transaction has been unsuccessful.

*Stored Procedure:* It is a subroutine available to applications that access a relational database system.

*Transaction:* It is the DBMS's abstract view of a user program: a sequence of reads and writes.

### **13.5 Review Questions**

1. What is a transaction?
2. Why do we use COMMIT?
3. What do you mean by ROLLBACK?
4. Differentiate between stored procedure and Dynamic SQL.
5. How are functions different from stored procedures?
6. Give the importance of using stored procedures.
7. How can we create a stored procedure?
8. What is the way to call a stored procedure?
9. How can you retrieve data from a stored procedure?
10. Give the advantages and disadvantages of a stored procedure.

### **Answers: Self Assessment**

- |                    |                     |
|--------------------|---------------------|
| 1. False           | 2. True             |
| 3. False           | 4. False            |
| 5. True            | 6. Stored Procedure |
| 7. IN, OUT, IN-OUT | 8. Dynamic SQL      |
| 9. Permanent       | 10. One             |
| 11. Input          | 12. Select          |
| 13. Function       | 14. Select          |
| 15. TAB            |                     |

## 13.6 Further Readings

Notes

*Books*

Ben Forta, Sams, Sams Teach Yourself SQL in 10 Minutes, Fourth Edition.

Dusan Petkovic, Microsoft® SQL Server® 2012: A Beginner's Guide, Fifth Edition, McGraw-Hill.

Jeffrey Garbus, Microsoft Transact-SQL: The Definitive Guide, Jones & Bartlett Learning.

Kalen Delaney, Inside Microsoft® SQL Server™ 2005, Fourth Edition, Microsoft Press.

*Online links*

<http://ecomputernotes.com/database-system/rdbms/transaction>

[http://www.razorsql.com/docs/sql\\_commit\\_rollback.html](http://www.razorsql.com/docs/sql_commit_rollback.html)

[http://www.sommarskog.se/dynamic\\_sql.html](http://www.sommarskog.se/dynamic_sql.html)

<http://www.tutorialspoint.com/sql/sql-transactions.htm>

<http://www.wiziq.com/tutorial/225006-Transaction-Processing-System-in-DBMS>

## Unit 14: Connecting to MySQL with PHP

### CONTENTS

Objectives

Introduction

14.1 Connecting to MySQL with PHP

14.1.1 Creating the Connection

14.1.2 Closing the Connection

14.2 Working with MySQL Data

14.2.1 Connecting to the Server

14.2.2 Connecting to the MySQL Server

14.2.3 Issuing Queries

14.2.4 Creating a Database

14.2.5 Removing a Database

14.2.6 Creating Tables

14.2.7 Inserting Data into the Table

14.2.8 Retrieving Information from a Table

14.2.9 Editing and Deleting Records

14.2.10 Altering the Structure of Tables

14.2.11 Dropping a Table

14.2.12 Working with NULL Value

14.2.13 Backing up a Database

14.3 Summary

14.4 Keywords

14.5 Review Questions

14.6 Further Readings

### Objectives

After studying this unit, you will be able to:

- Explain how to Create a Connection
- Understand Closing the Connection
- Elaborate upon the Concept of MySQL Data

### Introduction

MySQL and PHP are two technologies used by many web developers. Connecting to MySQL from PHP is easy and when we consider that both PHP and MySQL are free technologies, the pair sounds like winning choice. Connect to MSSQ can be made by using PHP's Sybase-ct support features. For this installation of MSSQL Client is required.

## 14.1 Connecting to MySQL with PHP

Notes

Before any operations are to be made on the data, database connection should be established. Two general types of connection exist – normal connections and persistent connections. There is no commonly used term for mentioning normal (non-persistent) connections. Thus, when connection is mentioned to be “normal”, or simply not mentioned to be persistent, then it is likely non-persistent connection. Within simple PHP scripts, single-time MySQL connections are created, and closed once script execution is completed. This is good for rare connections, when PHP page isn’t requested too frequently. The basic idea of persistent connections is to keep connection open for some particular time, and if the page loads multiple times, PHP code will reclaim the same connection. For high traffic websites, it would be reasonable to use persistent connections. However, if web site suffers from a very high traffic (million visits per day), for used PHP and MySQL versions the practice showed that performance degrades. For such websites, I would not recommend to use persistent connections. PHP developers reported, that under really heavy load, persistent connections reclaim too much of web server resources, and thus performance degrades, as the result of continuous memory swapping. However, if you are not planning to run such heavy traffic sites (e.g., like world immigration center, or Microsoft), persistent connection will act with noticeable performance boost, comparing to normal non-persistent connections. As for scripting, persistent connection doesn’t differ from usual connection, except that different function is used for connection establishment. Trying to close persistent connection will do no effect, however it’s often useful to keep closing function calls in PHP code (e.g., for compatibility purposes).

### 14.1.1 Creating the Connection

To establish simple connection, function *mysql\_connect* should be used.

```
$handle = mysql_connect (host [, username[, password]])
```

Working with persistent connection differs with only one feature – connection function name is *mysql\_pconnect*

All three parameters are of type string, and connection handle is returned once function is executed.

For example, if connection to server “cassy” is desired, the line from PHP script may look like this:

```
$link = mysql_connect ("cassy", "george", "greatpassword1105");
```

If name and password are not specified, the PHP process owner’s user ID is taken, and empty password is assumed. If host name is not specified, “localhost” will apply.

Normally, if server data is specified properly and MySQL is accessible, such connection can always be established, however sometimes (e.g., due to server name misspelled, wrong configuration, heavy load, flaw or software bug) such connection will fail. If this happens, in most cases further PHP script execution is useless. You can terminate PHP script with diagnostic message by using *die* function, like this:

```
$link = mysql_connect (null, "george", "greatpassword1105");
if (!$link) die("Can't connect to database server");
```

After a successful connection, *\$link* variable will contain connection handle. This handle is to be used further in all MySQL API function calls.

**Notes***Example:*

```
<?php
// Connect to database server
$hd = mysql_connect("myhost", "username", "password")
 or die ("Unable to connect");
// Select database
mysql_select_db ("database", $hd)
 or die ("Unable to select database");
// Execute sample query
$res = mysql_query("SELECT * FROM customer", $hd)
 or die ("Unable to run query");
// Query number of rows in rowset
$rows = mysql_num_rows($res);
// Output
echo "The query returned $rows row(s):\n\n";
// Iteration loop, for each row in rowset
while ($row = mysql_fetch_assoc($res))
{
 // Assigning variables from cell values
 $data1 = $row["title"];
 $data2 = $row["fname"];
 $data3 = $row["lname"];
 $data4 = $row["phone"];
 // Outputting data to browser
 echo "ROW# $nr : $data1 $data2 $data3 $data4\n";
}
mysql_close($hd);
?>
```

**14.1.2 Closing the Connection**

Finally, after connection is not needed anymore, *mysql\_close* call should be used to close connection.

```
mysql_close (connection_handle)
```

Unclosed orphan connections are kept in memory, and are closed only in some time (after timeout expires). This is, of course, a resource black hole for web sites under heavy load. Hence it is important to close the connections.

**Self Assessment**

Fill in the blanks:

1. Two general types of connection exist – ..... connections and ..... connections.
2. To establish simple connection, function ..... should be used.
3. Working with persistent connection uses ..... function.
4. If host name is not specified, ..... will apply.
5. You can terminate PHP script with diagnostic message by using ..... function.
6. After connection is not needed anymore ..... call should be used to close connection.

## 14.2 Working with MySQL Data

Now let us see how do we connect to SQL server and integrate PHP in it.

### 14.2.1 Connecting to the Server

Telnet allows the user to log in to a remote computer. When you use telnet, it is as if your keyboard and screen are connected to the remote computer.

Telnet is important for two reasons: First, you can use it to access a remote computer on which you have an account. Second, the Internet has many public resources which you access via telnet.

To use telnet, you run a telnet client program on your computer. This program uses the Internet to connect to the remote computer. The telnet client acts as an intermediary between you and the other computer. Everything you type on your keyboard is passed on to the other computer. Everything the other computer displays is sent to your computer where it appears on your screen. As a result, your keyboard and screen seem to be connected directly to the remote computer.



*Did u know?* Telnet is actually the oldest of the various Internet services. The first service provided on the Arpanet was “remote login”, that is, “telnet.”

Using a telnet client program is relatively simple. All you do is start the program, tell it the name of the remote host to which you want to connect. Once the connection with the host is established, you will see whatever message that computer displays as part of the starting procedure. If you connect to a UNIX computer which expects you to log in, it will start by displaying the line:

*login:* Type in the appropriate user name and press ENTER. You will then see the line.

*password:* Type in the password and press ENTER.

You will now be logged in. Once your telnet client starts, you can start your work.

### 14.2.2 Connecting to the MySQL Server

We need to give the correct privileges to users in order to allow them to connect to MySQL database. These privileges can be any combination of select, insert, update, delete, references, alter, and index. Below is an explanation of what each privilege means.

**Table 14.1: Privileges in SQL**

Privilege	Description
Select	Ability to query the table with a select statement.
Insert	Ability to add new rows to the table with the insert statement.
Update	Ability to update rows in the table with the update statement.
Delete	Ability to delete rows from the table with the delete statement.
References	Ability to create a constraint that refers to the table.
Alter	Ability to change the table definition with the alter table statement.
Index	Ability to create index on the table with the create index statement.

Source: [http://www.techonthenet.com/oracle/grant\\_revoke.php](http://www.techonthenet.com/oracle/grant_revoke.php)



**Notes**

For this we make use of GRANT and REVOKE commands. The syntax for granting privileges on a table is:

```
grant privileges on object to user;
```



*Example:* If you wanted to grant select, insert, update, and delete privileges on a table called suppliers to a user name abc, you would execute the following statement:

```
grant select, insert, update, delete on suppliers to abc;
```

You can also use the all keyword to indicate that you wish all permissions to be granted. For example:

```
grant all on suppliers to abc;
```

If you wanted to grant select access on your table to all users, you could grant the privileges to the public keyword. For example:

```
grant select on suppliers to public;
```

Once you have granted privileges, you may need to revoke some or all of these privileges. To do this, you can execute a revoke command. You can revoke any combination of select, insert, update, delete, references, alter, and index.

The syntax for revoking privileges on a table is:

```
revoke privileges on object from user;
```



*Example:* If you wanted to revoke delete privileges on a table called suppliers from a user named xyz, you would execute the following statement:

```
revoke delete on suppliers from xyz;
```

If you wanted to revoke all privileges on a table, you could use the all keyword.



*Example:*

```
revoke all on suppliers from xyz;
```

If you had granted privileges to public (all users) and you wanted to revoke these privileges, you could execute the following statement:

```
revoke all on suppliers from public;
```

If you give a grant for a user that doesn't exist, that user is created. After getting the user\_name, password you can connect to the database server. To connect to the server invoke the MySQL program from your shell prompt.

Syntax of the command is:

```
% mysql <options>
```

Where % indicates the shell prompts, mysql is the client program and <options> include the following:

```
-h host_name -u user_name -p password
```

```
-u user_name -p (if host is localhost)
```

Let us assume,

- host : localhost
- user\_name : aabb

- password : xxyy
- database : db\_1

Given the above, to connect to the database server use the command:

```
[anand soft@localhost anandsoft]$ mysql -u aabb -p
```

Then enter your password *xxyy* at the password prompt.

Now you are connected to the database server.

The connection can be terminated by giving EXIT at the *mysql* prompt.

```
mysql>EXIT
```

## Self Assessment

State whether the following statements are true or false:

7. Telnet allows the user to log in to a local computer.
8. Telnet needs username and password to allow user to log in.
9. GRANT is used to take back the privileges given.
10. REVOKE is used to give privileges to a user.

### 14.2.3 Issuing Queries

We can issue queries after connecting to the server. MySQL allows any case letters for keywords and functions. The database name and table name are however case sensitive.

To enter a query in MySQL, just type it, end it with a semicolon(;) and press enter. The semicolon marks the end of the query. You can also use '\g' to terminate queries.

### 14.2.4 Creating a Database

You would need special privilege to create or to delete a MySQL database. So assuming you have access to root user, you can create any database.



*Example:*

```
[root@host]# mysqladmin -u root -p create TUTORIALS
Enter password:*****
```

This will create a MySQL database TUTORIALS.

### 14.2.5 Removing a Database

You would need special privilege to create or to delete a MySQL database. So assuming you have access to root user, you can create any database using MySQL. Be careful while deleting any database because it will lose your all the data available in your database.



*Example:*

```
[root@host]# mysqladmin -u root -p drop TUTORIALS
Enter password:*****
```

**Notes**

This will give you a warning and it will confirm if you really want to delete this database or not.

### 14.2.6 Creating Tables

The table creation command requires:

- Name of the table
- Names of fields
- Definitions for each field

The syntax to create a table is,

```
CREATE TABLE table_name (column_name column_type);
```

To create a MySQL table from mysql> prompt, you will use SQL command CREATE TABLE.



*Example:*

Here is an example which creates tutorials\_tbl.

```
root@host# mysql -u root -p
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> CREATE TABLE tutorials_tbl(
 -> tutorial_id INT NOT NULL AUTO_INCREMENT,
 -> tutorial_title VARCHAR(100) NOT NULL,
 -> tutorial_author VARCHAR(40) NOT NULL,
 -> submission_date DATE,
 -> PRIMARY KEY (tutorial_id)
 ->);
Query OK, 0 rows affected (0.16 sec)
mysql>
```



*Notes* MySQL does not terminate a command until you give a semi colon (;) at the end of SQL command.

### 14.2.7 Inserting Data into the Table

To insert data into MySQL table you would need to use SQL INSERT INTO command. You can insert data into MySQL table by using mysql> prompt or by using any script like PHP.

Command prompt uses the SQL INSERT INTO command to insert data into MySQL table tutorials\_tbl



*Example:* Following example will create 3 records into tutorials\_tbl table:

```
root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> INSERT INTO tutorials_tbl
```

## Notes

```

->(tutorial_title, tutorial_author, submission_date)
->VALUES
->("Learn PHP", "John Poul", NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO tutorials_tbl
->(tutorial_title, tutorial_author, submission_date)
->VALUES
->("Learn MySQL", "Abdul S", NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO tutorials_tbl
->(tutorial_title, tutorial_author, submission_date)
->VALUES
->("JAVA Tutorial", "Sanjay", '2007-05-06');
Query OK, 1 row affected (0.01 sec)
mysql>

```



*Notes* Please note that all the arrow signs (->) are not part of SQL command they are indicating a new line and they are created automatically by MySQL prompt while pressing enter key without giving a semi colon at the end of each line of the command.

In the above example we have not provided tutorial\_id because at the time of table create we had given AUTO\_INCREMENT option for this field. So MySQL takes care of inserting these IDs automatically. Here NOW() is a MySQL function which returns current date and time.

### 14.2.8 Retrieving Information from a Table

Use SQL SELECT command to fetch data from MySQL table tutorials\_tbl through the command prompt.



*Example:*

Following example will return all the records from tutorials\_tbl table:

```

root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
+-----+-----+-----+-----+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-----+-----+-----+-----+
| 1 | Learn PHP | John Poul | 2007-05-21 |
| 2 | Learn MySQL | Abdul S | 2007-05-21 |
| 3 | JAVA Tutorial | Sanjay | 2007-05-21 |
+-----+-----+-----+-----+
mysql>

```

## 14.2.9 Editing and Deleting Records

There may be a requirement where existing data in a MySQL table need to be modified. Use SQL UPDATE command with WHERE clause to update selected data into MySQL table tutorials\_tbl



*Example:*

Following example will update tutorial\_title field for a record having tutorial\_id as 3.

```
root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> UPDATE tutorials_tbl
 -> SET tutorial_title='Learning JAVA'
 -> WHERE tutorial_id=3;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql>
```

If you want to delete a record from any MySQL table then use the SQL DELETE command with WHERE clause to delete selected data into MySQL table tutorials\_tbl



*Example:*

Following example will delete a record into tutorial\_tbl whose tutorial\_id is 3.

```
root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> DELETE FROM tutorials_tbl WHERE tutorial_id=3;
Query OK, 1 row affected (0.23 sec)

mysql>
```

## 14.2.10 Altering the Structure of Tables

MySQL ALTER command is very useful when you want to change a name of your table, any table field or if you want to add or delete an existing column in a table.

Let us begin with creation of a table called **testalter\_tbl**

```
root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
```

```
mysql> create table testalter_tbl
-> (
-> i INT,
-> c CHAR(1)
->);
```

Query OK, 0 rows affected (0.05 sec)

```
mysql> SHOW COLUMNS FROM testalter_tbl;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| i | int(11) | YES | | NULL | |
| c | char(1) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
```

2 rows in set (0.00 sec)

Suppose you want to drop an existing column **i** from above MySQL table then you will use **DROP** clause along with **ALTER** command as follows

```
mysql> ALTER TABLE testalter_tbl DROP i;
```

A **DROP** will not work if the column is the only one left in the table.

To add a column, use **ADD** and specify the column definition. The following statement restores the **i** column to **testalter\_tbl**

```
mysql> ALTER TABLE testalter_tbl ADD i INT;
```

After issuing this statement, **testalter** will contain the same two columns that it had when you first created the table, but will not have quite the same structure. That's because new columns are added to the end of the table by default. So even though **i** originally was the first column in **mytbl**, now it is the last one:

```
mysql> SHOW COLUMNS FROM testalter_tbl;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| c | char(1) | YES | | NULL | |
| i | int(11) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
```

2 rows in set (0.00 sec)

To indicate that you want a column at a specific position within the table, either use **FIRST** to make it the first column, or **AFTER col\_name** to indicate that the new column should be placed after **col\_name**. Try the following **ALTER TABLE** statements, using **SHOW COLUMNS** after each one to see what effect each one has:

```
ALTER TABLE testalter_tbl DROP i;
```

```
ALTER TABLE testalter_tbl ADD i INT FIRST;
```

```
ALTER TABLE testalter_tbl DROP i;
```

```
ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

**Notes**

The **FIRST** and **AFTER** specifiers work only with the **ADD** clause. This means that if you want to reposition an existing column within a table, you first must **DROP** it and then **ADD** it at the new position.

To change a column's definition, use **MODIFY** or **CHANGE** clause along with **ALTER** command. For example, to change column **c** from **CHAR(1)** to **CHAR(10)**, do this:

```
mysql> ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```

With **CHANGE**, the syntax is a bit different. After the **CHANGE** keyword, you name the column you want to change, then specify the new definition, which includes the new name. Try out following example:

```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

If you now use **CHANGE** to convert **j** from **BIGINT** back to **INT** without changing the column name, the statement be as expected:

```
mysql> ALTER TABLE testalter_tbl CHANGE j j INT;
```

### 14.2.11 Dropping a Table

It is very easy to drop an existing MySQL table. But you need to be very careful while deleting any existing table because data lost will not be recovered after deleting a table. This needs just to execute **DROP TABLE SQL** command at **mysql>** prompt.



*Example:*

Here is an example which deletes **tutorials\_tbl**:

```
root@host# mysql -u root -p
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> DROP TABLE tutorials_tbl
Query OK, 0 rows affected (0.8 sec)
mysql>
```

### 14.2.12 Working with NULL Value

When we try to give a condition which compare field or column value to **NULL** it does not work properly. To handle such situation MySQL provides three operators:

- **IS NULL**: operator returns true of column value is **NULL**.
- **IS NOT NULL**: operator returns true of column value is not **NULL**.
- **<=>** operator compare values, which (unlike the **=** operator) is true even for two **NULL** values

Conditions involving **NULL** are special. You cannot use **= NULL** or **!= NULL** to look for **NULL** values in columns. Such comparisons always fail because it's impossible to tell whether or not they are true. Even **NULL = NULL** fails. To look for columns that are or are not **NULL**, use **IS NULL** or **IS NOT NULL**.

Suppose a table **tcount\_tbl** in **TUTORIALS** database and it contains two columns **tutorial\_author** and **tutorial\_count**, where a **NULL tutorial\_count** indicates that the value is unknown:



Example:

```

root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> create table tcount_tbl
-> (
-> tutorial_author varchar(40) NOT NULL,
-> tutorial_count INT
->);
Query OK, 0 rows affected (0.05 sec)
mysql> INSERT INTO tcount_tbl
-> (tutorial_author, tutorial_count) values ('mahran', 20);
mysql> INSERT INTO tcount_tbl
-> (tutorial_author, tutorial_count) values ('mahnaz', NULL);
mysql> INSERT INTO tcount_tbl
-> (tutorial_author, tutorial_count) values ('Jen', NULL);
mysql> INSERT INTO tcount_tbl
-> (tutorial_author, tutorial_count) values ('Gill', 20);

mysql> SELECT * from tcount_tbl;
+-----+-----+
| tutorial_author | tutorial_count |
+-----+-----+
| mahran | 20 |
| mahnaz | NULL |
| Jen | NULL |
| Gill | 20 |
+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

You can see that = and != do not work with NULL values as follows:

```

mysql> SELECT * FROM tcount_tbl WHERE tutorial_count = NULL;
Empty set (0.00 sec)

mysql> SELECT * FROM tcount_tbl WHERE tutorial_count != NULL;
Empty set (0.01 sec)

```

### 14.2.13 Backing up a Database

The **mysqldump** can be used to dump a database or a collection of databases for backup or transfer to another SQL server (not necessarily a mysql server). The dump typically contains



**Notes**

SQL statements to create the table, populate it, or both. However, **mysqldump** can also be used to generate files in CSV, other delimited text, or XML format. It requires at least the **SELECT** privilege for dumped tables, **SHOW VIEW** for dumped views, **TRIGGER** for dumped triggers, and **LOCK TABLES** if the **—single-transaction** option is not used. Certain options might require other privileges as noted in the option descriptions. To reload a dump file, you must have the same privileges needed to create each of the dumped objects by issuing **CREATE** statements manually.

**Mysqldump** output can include **ALTER DATABASE** statements that change the database collation. These may be used when dumping stored programs to preserve their character encodings. To reload a dump file containing such statements, the **ALTER** privilege for the affected database is required. If you are doing a backup on the server and your tables all are **myisam** tables, consider using the **mysqlhotcopy** instead because it can accomplish faster backups and faster restores.

There are three general ways to invoke **mysqldump**:


```
Shell> mysqldump [options] db_name [tbl_name ...]
Shell> mysqldump [options] --databases db_name ...
Shell> mysqldump [options] --all-databases
```

If you do not name any tables following **db\_name** or if you use the **—databases** or **—all-databases** option, entire databases are dumped.

**Self Assessment**

Fill in the blanks:

11. The table creation command requires: ..... of the table, Names of ..... and Definitions for each field.
12. Use SQL ..... command to fetch data from MySQL table.
13. MySQL ..... command is very useful when you want to change an aspect of your table.
14. MySQL provides three operators to work with null values ..... , ..... and .....
15. The ..... can be used to dump a database or a collection of databases for backup or transfer to another SQL server.



*Case Study*

**History of PHP**

**E**very once in a while a person faces a particular problem or requirement to which there appears to be no existing solution. Faced with this problem the person decides to create a solution to provide the needed functionality. Having developed the solution to their problem it then occurs to them that others may need to solve the same problem, and they decide to make their solution freely available to others who, in turn, can use and improve on it. Within a short period of time many people adopt the technology and work on it, adding new features they feel will be useful. The solution soon grows beyond expectations in terms of features and is adopted by more people than the original creator could ever have imagined. The history of PHP is just such a story.

*Contd...*

***The Creation of PHP***

The first version of what came to be known as PHP was created in 1995 by a man named Rasmus Lerdorf. Rasmus, now an engineer at Yahoo!, needed something to make it easier to create content on his website, something that would work well with HTML, yet give him power and flexibility beyond what HTML could offer him. Essentially, what he needed was an easy way to write scripts that would run on his web server both to create content, and handle data being passed back to the server from the web browser. Using the Perl language, he created some technology that gave him what he needed and decided to call this technology “Personal Home Page/Forms Interpreter”. The technology provided a convenient way to process web forms and create content.

The name “Personal Home Page/Forms Interpreter” was later shortened to PHP/FI and eventually renamed to represent “PHP: Hypertext Preprocessor”. The name is said to be recursive because the full name also includes the acronym “PHP” - an odd geeky joke that is common in technology circles when people have trouble naming things. GNU is another recursive name that represents “GNU’s Not Unix”.

PHP/FI version 1.0 was never really used outside of Rasmus’ own web site. With the introduction of PHP/FI 2.0 this began to change. When PHP 3 was released in 1997, adoption of PHP exploded beyond all belief.

***PHP 3 Hits the Big Time***

By the time 1997 arrived the number of websites on the internet was growing exponentially and most of these websites were being implemented using the Apache web server. It was around this time that Andy Gutmans and Zeev Suraski launched the PHP 3 project, a project designed to take PHP to the next level. One of the key achievements of the PHP 3 project was to implement PHP as a robust Apache Module.

PHP 3 was implemented using a modular approach that made it easy for others to extend functionality, and also introduced the first elements of object-orientation that would continue to evolve through subsequent releases.

The combination of PHP 3 and Apache quickly lead to the widespread adoption of PHP, and it is commonly estimated that, at its peak adoption level, PHP3 was used to power over 10% of all websites on the internet.

***PHP 4 Optimization, Scalability and More***

With PHP 4 Andi Gutmans and Zeev Suraski once again re-architected PHP from the ground up. PHP 4 was built upon a piece of technology called the Zend Engine. The move to the Zend Engine brought about a number of key improvements in PHP:

- Support for other web servers (Microsoft’s Internet Information Server (IIS) being of particular significance).
- Improved memory handling to avoid memory leaks (one of the most difficult types of problems to isolate in a program).
- Improved efficiency and performance to support large scale, complex, mission critical enterprise application development using PHP.

In addition PHP 4 also built on the earlier Object-Oriented Programming features of PHP 3 with the introduction of classes.

***PHP 5 Object-Orientation, Error Handling and XML***

The main, though far from only, feature of PHP 5 is the improved support for Object Oriented Programming (OOP). In addition, PHP 5 introduced some features common in other languages such as Java like try/catch error and exception handling.

*Contd...*

**Notes**

PHP 5 also introduced new extensions aimed at easing the storage and manipulation of data. Significant new features include SimpleXML for handling XML documents, and SQLite, an embedded basic and easy to use database interface.

***How Popular is PHP?***

A quick review of some statistics gives a very clear indication of the phenomenally widespread use of PHP. A company called Netcraft specializes in recording data about the types of web servers and web server modules that are used on the internet. As of April 2007 Netcraft reported that PHP was used on over 20,000,000 distinct web domains.

A web survey by SecuritySpace also lists PHP as the most widely deployed Apache module. It is safe to say that PHP has taken the internet by storm.

As if that wasn't enough one of the world's most popular websites, Wikipedia, is build primarily using PHP.

***Questions:***

1. Study and analyse the case.
2. Write down the case facts.
3. What do you infer from the case?

*Source:* [http://www.techotopia.com/index.php/The\\_History\\_of\\_PHP](http://www.techotopia.com/index.php/The_History_of_PHP)

### **14.3 Summary**

- Database connection should be established to connect PHP and MySQL.
- Two general types of connection exist – normal connections and persistent connections.
- The basic idea of persistent connections is to keep connection open for some particular time, and if the page loads multiple times, PHP code will reclaim the same connection.
- To establish simple connection, function `mysql_connect` should be used.
- Working with persistent connection differs with only one feature – connection function name is `mysql_pconnect`.
- If name and password are not specified, the PHP process owner's user ID is taken, and empty password is assumed.
- If host name is not specified, "localhost" will apply.
- You can terminate PHP script with diagnostic message by using `die` function.
- After connection is not needed anymore, `mysql_close` call should be used to close connection.
- Telnet allows the user to log in to a remote computer.
- The table creation command requires: Name of the table, Names of fields and Definitions for each field.
- Command prompt uses the SQL `INSERT INTO` command.
- Use SQL `SELECT` command to fetch data from MySQL table.
- Use SQL `UPDATE` command with `WHERE` clause to update selected data into MySQL table.
- MySQL `ALTER` command is very useful when you want to change an aspect of your table.

- MySQL provides three operators to work with null values- IS NULL, IS NOT NULL and <=> operator.
- The mysqldump can be used to dump a database or a collection of databases for backup or transfer to another SQL server.

## 14.4 Keywords

*die()*: To terminate PHP script with diagnostic message.

*MySQL ALTER Command*: Is very useful when you want to change an aspect of your table.

*mysql\_close()*: To close connection.

*mysqldump()*: It can be used to dump a database or a collection of databases for backup or transfer to another SQL server.

*SQL INSERT INTO*: Used by command prompt for insertion.

*SQL SELECT Command*: Used to fetch data from MySQL table.

*SQL UPDATE Command*: Used with WHERE clause to update selected data into MySQL table.

*Telnet*: It allows the user to log in to a remote computer.

## 14.5 Review Questions

1. Give the steps to connect PHP and MySQL server.
2. How do you close a connection?
3. What is telnet?
4. Explain the use of GRANT command.
5. What do you mean by REVOKE?
6. How to create a database through command prompt?
7. Give an example to create a table in MySQL via telnet.
8. Why do we need the ALTER command?
9. Can we retrieve a table that we have dropped?
10. What is mysqldump()?

## **Answers: Self Assessment**

- |                             |                             |
|-----------------------------|-----------------------------|
| 1. Normal, Persistent       | 2. mysql_connect()          |
| 3. mysql_pconnect()         | 4. localhost                |
| 5. die                      | 6. mysql_close()            |
| 7. False                    | 8. True                     |
| 9. False                    | 10. False                   |
| 11. Name, Field, Definition | 12. Select                  |
| 13. Alter                   | 14. IS NULL, IS NOT NULL, 6 |
| 15. Mysql_dump()            |                             |

Notes

## 14.6 Further Readings



*Books*

Chris Lea, PHP MySQL Website Programming: Problem - Design - Solution.  
David Lane, Web Database Applications with PHP & MySQL, 2nd Edition.  
Elizabeth Naramor, Beginning PHP5, Apache, and MySQL Web Development.  
W. Jason Gilmore, Beginning PHP 5 and MySQL: From Novice to Professional.



*Online links*

[http://docs.oracle.com/cd/E17781\\_01/server.112/e18804/connecting.htm#CHDHGDJG](http://docs.oracle.com/cd/E17781_01/server.112/e18804/connecting.htm#CHDHGDJG)  
<http://www.tutorialspoint.com/mysql/mysql-database-export.htm>  
<http://www.tutorialsworld.com/sql/working-with-mysql.htm>  
[http://www.w3schools.com/php/php\\_mysql\\_intro.asp](http://www.w3schools.com/php/php_mysql_intro.asp)  
<http://www.yohng.com/phpm/phpm.htm>

**LOVELY PROFESSIONAL UNIVERSITY**

Jalandhar-Delhi G.T. Road (NH-1)

Phagwara, Punjab (India)-144411

For Enquiry: +91-1824-300360

Fax.: +91-1824-506111

Email: [odl@lpu.co.in](mailto:odl@lpu.co.in)