Open Source Technologies

DCAP203/DCAP410

Edited by:
Sarabjit Kumar

LOVELY
PROFESSIONAL
UNIVERSITY

OPEN SOURCE TECHNOLOGIES

Edited By

Sarabjit Kumar

Printed by
FRANK BROTHERS & CO. (PUBLISHERS) LIMITED
B-41, Sector-4, NOIDA - 201301, Gautam Budh Nagar
for
Lovely Professional University
Phagwara

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 web sites using the PHP programming language.

S. No.	Description
1.	**My SQL:** 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.**

CONTENT

Unit 1: My SQL

CONTENTS

Objectives

After studying this unit, you will be able to:

- Discuss current and future versions of My SQL.
- Explain installing My SQL.
- Understand basic security guidelines in My SQL.

Introduction

My SQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout it's history. With its superior speed, reliability and ease of use, My SQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

Many of the world's largest and fastest-growing organizations use My SQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software—including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com.

The flagship My SQL offering is My SQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription.

My SQL is a key part of LAMP (Linux, Apache, My SQL, PHP/Perl/Python), the fast-growing open source enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in.

1.1 Current and Future Versions of My SQL

The installation instructions in this unit is refer to My SQL 4.0.21, which is the current production version of the software. This version number can be read as minor release number 21 of the major version 4.0 software. My SQL AB, the company responsible for creating and distributing My SQL, uses minor release numbers for updates containing security enhancements or bug fixes. Minor releases do not follow a set release schedule; when enhancements or fixes are added to the code and thoroughly tested, My SQL AB releases a new version, with a new minor version number.

It is possible that by the time you purchase this book, the minor version number will have changed, to 4.0.21 or beyond. If that is the case, you should read the list of changes at http://www.My SQL.com/doc/en/News-4.0.x.html for any changes regarding the installation or configuration process, which makes up the bulk of this unit.

Although it is unlikely that any installation instructions will change between minor version updates, you should get in the habit of always checking the changelog of software that you install and maintain. If a minor version change does occur during the time you are reading this book, but no installation changes are noted in the change log, simply make a mental note and substitute the new version number wherever it appears in the installation instructions and accompanying figures.

1.1.1 How to Get My SQL

My SQL AB, the Company that develops and maintains the My SQL database server, distributes My SQL on its Web site: http://www.My SQL.com/. Binary distributions for all platforms, installer packages for Mac OS X, and RPMs and source code files for Linux/Unix platforms can be found at the Web site. Additionally, you can purchase boxed versions of the software that is, software in a box and with a printed version of the comprehensive My SQL manualfrom the My SQL AB online store, for a very reasonable price.

Did u know? Linux distribution CDs usually contain some version or another of the open source My SQL software, although these are usually several minor versions that are out of date.

1.2 Installing My SQL

1.2.1 Installing My SQL on Linux/Unix

The process of installing My SQL on Linux/Unix is straight forward, whether you use RPMs or install the binaries. For a minimal installation from RPMs, you will need two files:

My SQL-server-VERSION.i386.rpm The My SQL server

My SQL-client-VERSION.i386.rpm The standard My SQL client libraries

To perform a minimal installation from RPMs, type the following at your prompt:

```
#> rpm -i My SQL-server-VERSION.i386.rpm My SQL-client-VERSION.i386.rpm
```

Did u know? Replace VERSION in the filename with the actual version you downloaded. For example, the current My SQL 4.0 server RPM is called My SQL-server-4.0.21-0.i386.rpm, and the client libraries RPM is called My SQL-client-4.0.21-0.i 386.rpm.

Another painless (and very common) installation method is to install My SQL from a binary distribution. This method requires the gunzip and tar utilities, to uncompress and unpack the distribution, and also requires the ability to create groups and users on the system. The first series of commands in the binary distribution installation process has you adding a group and a user and unpacking the distribution, as follows:

Did u know? Replace VERSION-OS in the filename with the actual version you downloaded. For example, the current My SQL 4.0 Linux/i386 binary is called My SQL-max-4.0.21-pclinux-i686.tar.gz.

- groupadd My SQL
- useradd -g My SQL
- cd/usr/local
- gunzip < /path/to/My SQL-VERSION-OS.tar.gz|tar xvf -

Next, the instructions tell you to create a link with a shorter name:

- ln -s My SQL-VERSION-OS My SQL
- cd My SQL

Once unpacked, the README and INSTALL files will walk you through the remainder of the installation process for the version of My SQL you've chosen. In general, the following series of commands will be used:

- scripts/My SQL_install_db
- chown -R root/usr/local/My SQL
- chown -R My SQL/usr/local/My SQL/data
- chgrp -R My SQL/usr/local/My SQL
- chown -R root/usr/local/My SQL/bin

You're now ready to start the My SQL server, so skip down to the section called "Basic Security Guidelines." If you had any issues with your installation, check the "Troubleshooting Your Installation" section.

1.2.2 Installing My SQL on Mac OS X

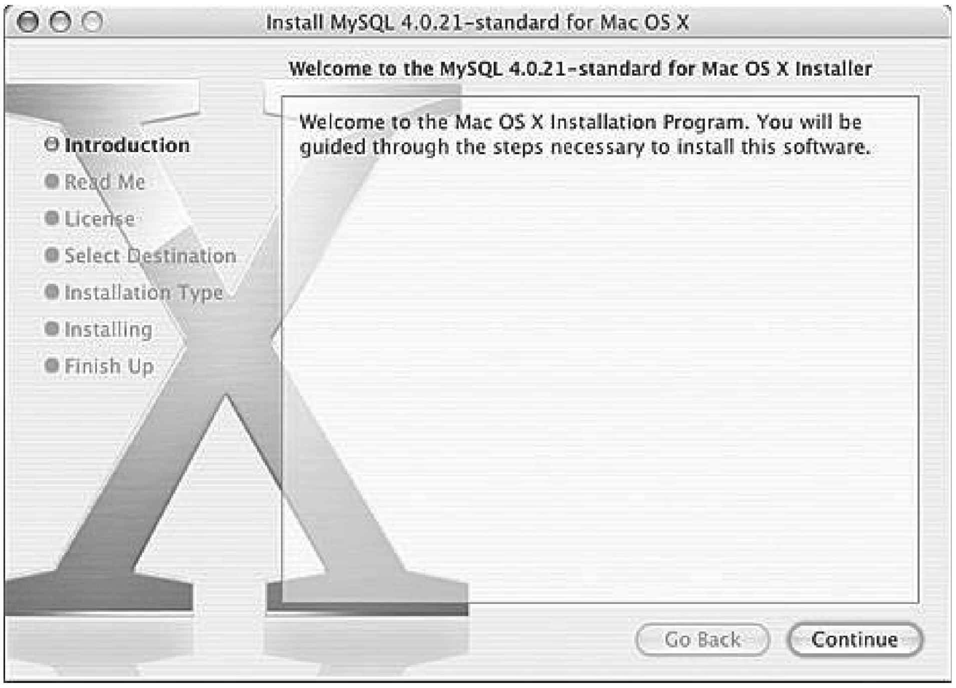
The My SQL installation process for Mac OS X is fairly straight forward the developers from My SQL AB have created an installation package for Mac OS X. Go to the My SQL downloads page at http://www.My SQL.com/downloads/My SQL-4.0.html and look for the section titled "Mac OS X downloads." Once there, select the appropriate version for your system, either.

No matter the Mac OS X version you use, download the "Standard" package for the current My SQL version. When the download is complete, unpack the installer file and double-click on the *.pkg file. Follow the installation steps below to complete the process.

1. The My SQL installer will launch automatically, as shown in Figure 1.1. Click Continue to move to the next step.

Notes

Figure 1.1: The My SQL Installer has Started

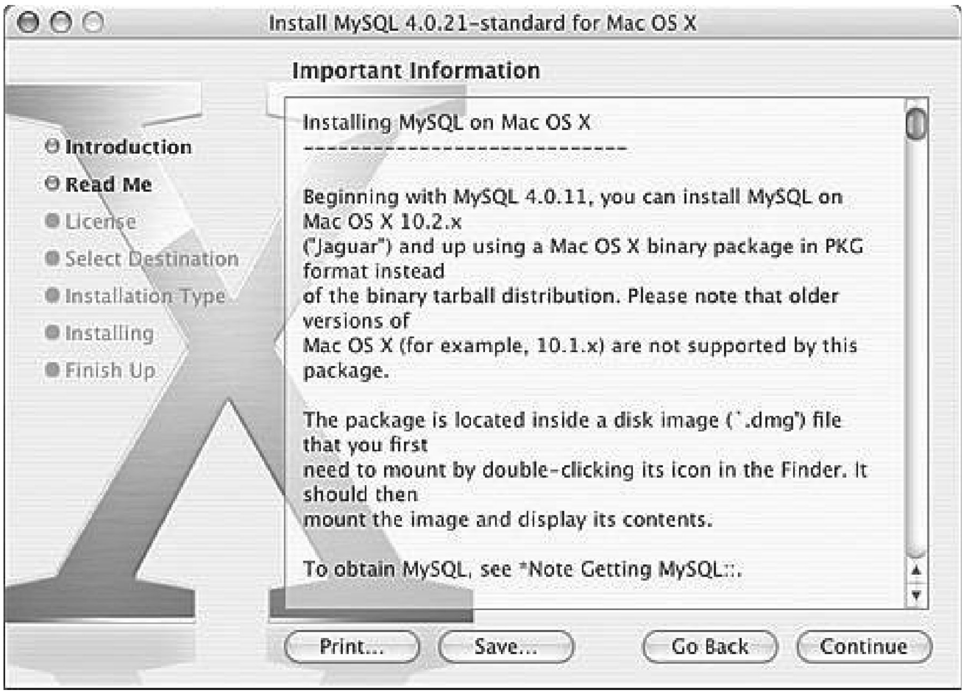


Welcome to the MySQL 4.0.21-standard for Mac OS X Installer

Welcome to the Mac OS X Installation Program. You will be guided through the steps necessary to install this software.

2. The second screen in the installation process contains information regarding installation (see Figure 1.2). Read the information and note anything relevant to your situation and then click Continue.

Figure 1.2: Step 2 of the My SQL Installation Wizard



Important Information

Installing MySQL on Mac OS X

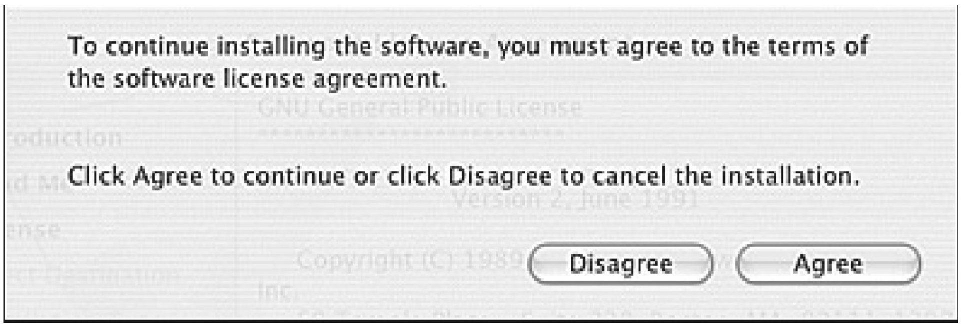
Beginning with MySQL 4.0.11, you can install MySQL on Mac OS X 10.2.x
("Jaguar") and up using a Mac OS X binary package in PKG format instead
of the binary tarball distribution. Please note that older versions of
Mac OS X (for example, 10.1.x) are not supported by this package.

The package is located inside a disk image (`.dmg') file that you first
need to mount by double-clicking its icon in the Finder. It should then
mount the image and display its contents.

To obtain MySQL, see *Note Getting MySQL::.

3. The third screen in the installation process displays the license information. Read this information and press the Continue button, and you will be prompted to agree or disagree, as shown in Figure 1.3.

Figure 1.3: Step 3 of the My SQL Installation Wizard. Read the License Information, and then Agree to its Content

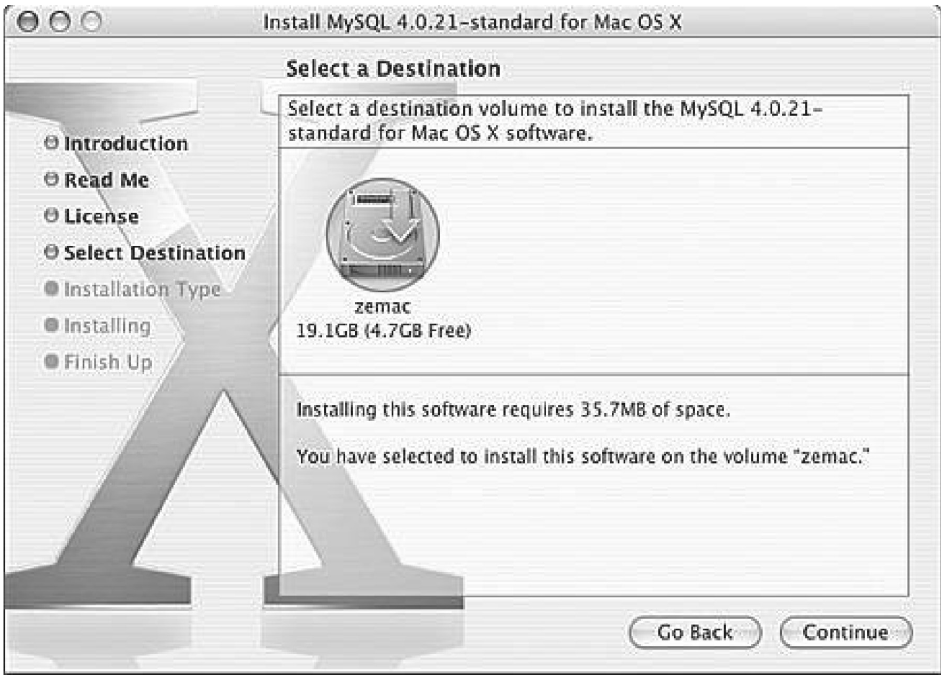


To continue installing the software, you must agree to the terms of the software license agreement.

Click Agree to continue or click Disagree to cancel the installation.

4. The fourth screen asks you to select the installation destination (see Figure 1.4). Select the appropriate drive, and then press the Continue button.

Figure 1.4: Step 4 of the My SQL Installation Wizard. Select an Installation Location



Install MySQL 4.0.21-standard for Mac OS X

Select a Destination

Select a destination volume to install the MySQL 4.0.21-standard for Mac OS X software.

zemac
19.1GB (4.7GB Free)

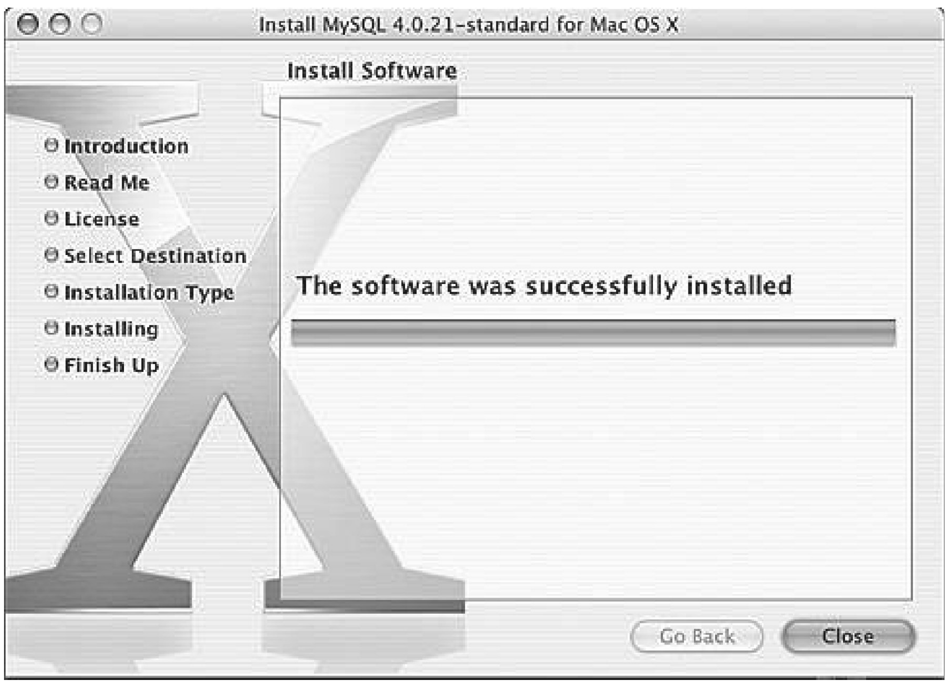
Installing this software requires 35.7MB of space.

You have selected to install this software on the volume "zemac."

5. The next screen verifies your installation location selection and requires you to press the Install button to continue. At this point, you will be prompted for the Adminstrator username and password unless you are installing as root before the installation process continues. Once it continues, let the process run its course until you see the installation is complete, as shown in Figure 1.5.

Notes

Figure 1.5: My SQL has been Installed



The software was successfully installed

You're now ready to start the My SQL server, so skip down to the section called "Basic Security Guidelines." If you had any issues with your installation, check the "Troubleshooting Your Installation" section.

1.2.3 Installing My SQL on Windows

The My SQL installation process on Windows is also quite simple the developers from My SQL AB have packaged up everything you need in one zip file with a setup program! Download the zip file, extract its contents into a temporary directory, and run the setup.exe application. After the setup.exe application installs the My SQL server and client programs, you're ready to start the My SQL server.

The following steps detail the installation of My SQL 4.0.x on Windows, when the installer is downloaded from My SQL AB. The install sequence looks similar, regardless if you have a Windows 98, Windows NT, Windows 2000, or Windows XP environment for testing and development. Many users install My SQL on personal Windows machines just to get a feel for working with the database before deploying My SQL in a production environment.

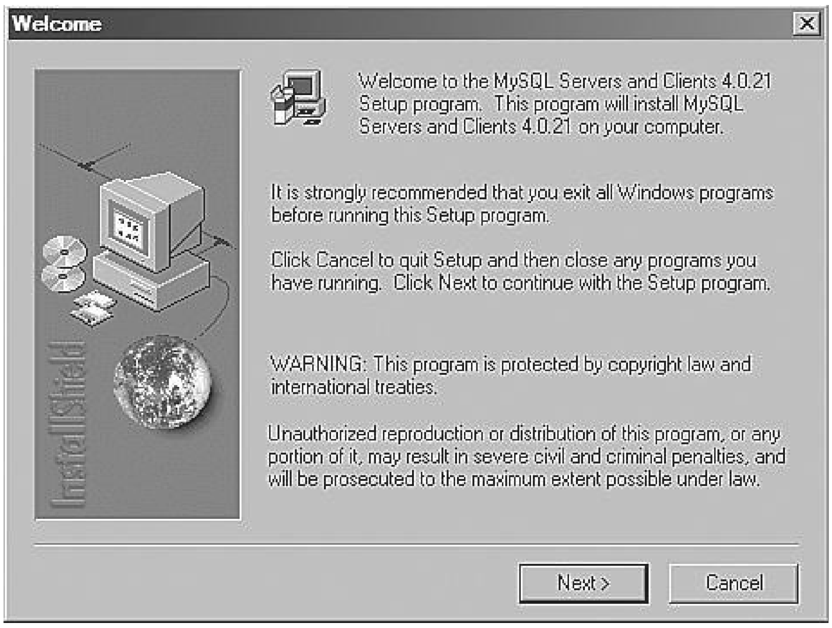
Did u know? If you have the tools and skills to compile your own Windows binary files, the Cygwin source code is also available from My SQL AB. Follow the instructions contained in the source distribution to build your own executable files.

Jumping right into the installation sequence, assuming you have downloaded the Windows installer from the My SQL AB Web site, follow these steps:

1. Extract the contents of the zip file into a temporary directory and find the setup.exe file, and then double-click it to start the installation. You will see the first screen of the installation wizard, as shown in Figure 1.6. Click Next to continue.

Notes

Figure 1.6: The First Step of the My SQL Installation Wizard



Welcome

Welcome to the MySQL Servers and Clients 4.0.21 Setup program. This program will install MySQL Servers and Clients 4.0.21 on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

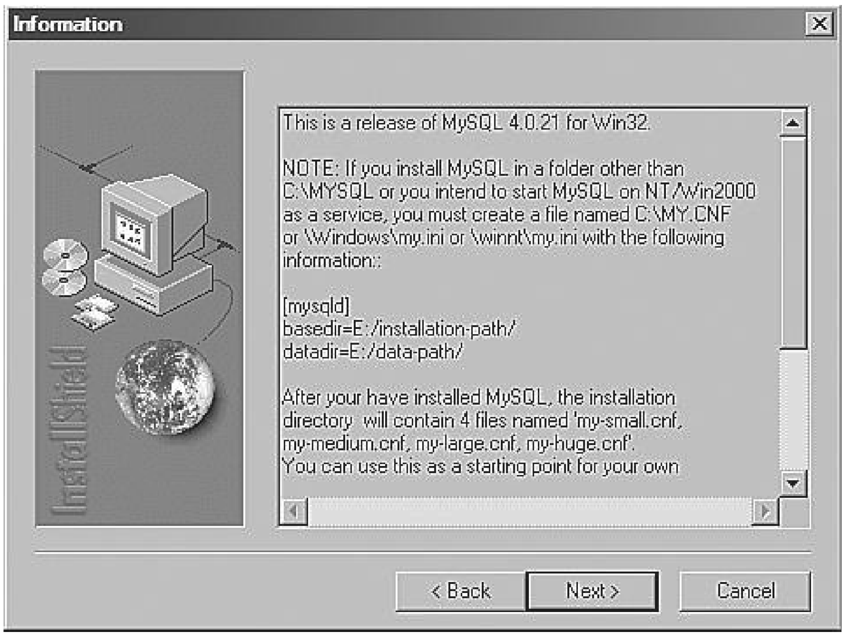
Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

2. The second screen in the installation process contains information regarding the installation location (see Figure 1.7). The default installation location is C:\My SQL. If you plan to install My SQL in a different location, this screen shows you a few changes that you will have to make on your own. The information on this screen is also important for Windows NT users who want to start My SQL as a service. Read the information and note anything relevant to your situation, and then click Next to continue.

Figure 1.7: Step 2 of the My SQL Installation Wizard. Note any Relevant Information before Continuing



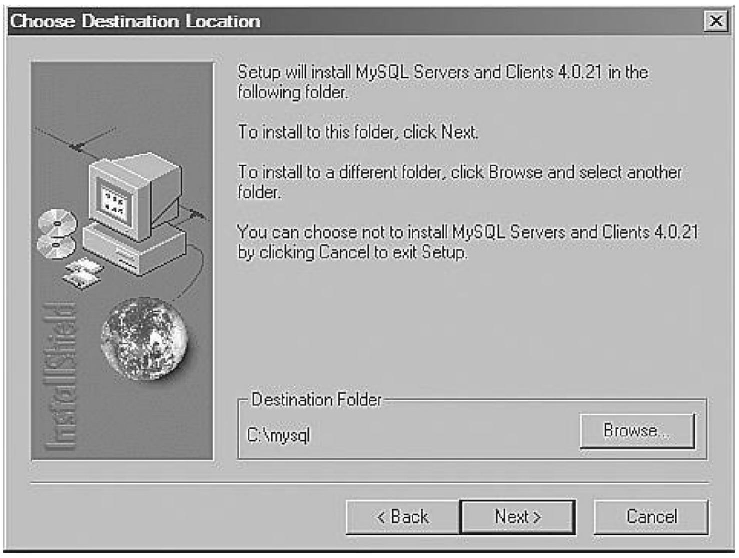
Information

```
This is a release of MySQL 4.0.21 for Win32.

NOTE: If you install MySQL in a folder other than
C:\MYSQL or you intend to start MySQL on NT/Win2000
as a service, you must create a file named C:\MY.CNF
or \Windows\my.ini or \winnt\my.ini with the following
information::

[mysqld]
basedir=E:/installation-path/
datadir=E:/data-path/

After your have installed MySQL, the installation
directory  will contain 4 files named 'my-small.cnf,
my-medium.cnf, my-large.cnf, my-huge.cnf'.
You can use this as a starting point for your own
```

Notes

3. The third screen in the installation process has you select the installation or destination location (see Figure 1.8). If you want to install My SQL in the default location, click Next to continue. Otherwise, click Browse and navigate to the location of your choice, and then click Next to continue.

Figure 1.8: Step 3 of the My SQL Installation Wizard. Select an Installation Location



Choose Destination Location

Setup will install MySQL Servers and Clients 4.0.21 in the following folder.

To install to this folder, click Next.

To install to a different folder, click Browse and select another folder.

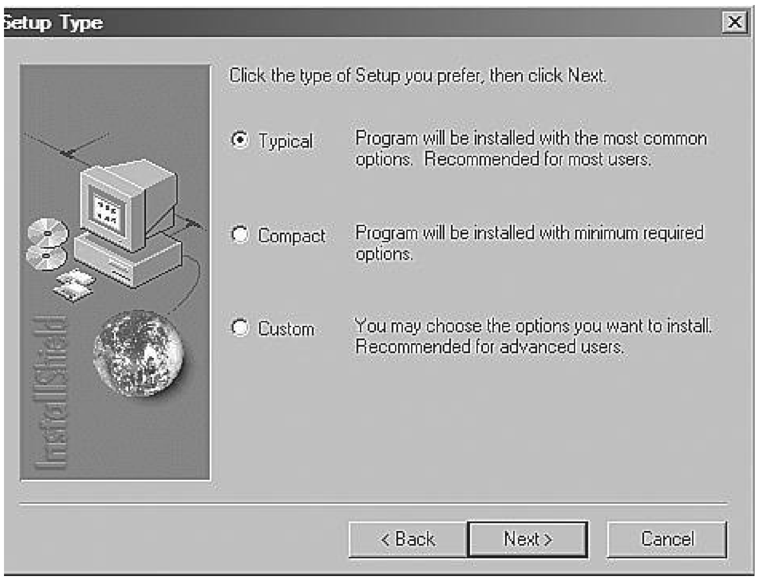
You can choose not to install MySQL Servers and Clients 4.0.21 by clicking Cancel to exit Setup.

Destination Folder

C:\mysql

4. The fourth screen asks you to select the installation method Typical, Compact, or Custom (see Figure 1.9). The Custom option allows you to select elements of My SQL to install, such as documentation and help files. Select Typical as the installation method, and click Next to continue.

Figure 1.9: Step 4 of the My SQL Installation Wizard. Select an Installation Type



Setup Type

Click the type of Setup you prefer, then click Next.

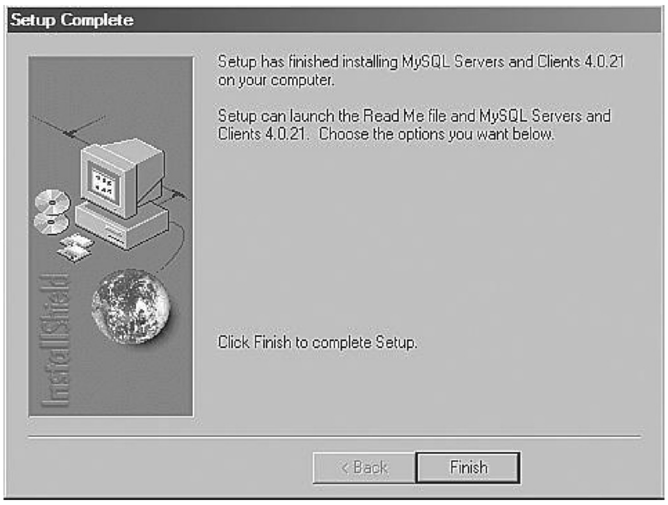
[✓ Typical] Program will be installed with the most common options. Recommended for most users.

Compact Program will be installed with minimum required options.

Custom You may choose the options you want to install. Recommended for advanced users.

5. The installation process will now take over and install files in their proper locations. When the process is finished, you will see a confirmation of completion, as in Figure 1.10. Click Finish to complete the setup process.

Figure 1.10: My SQL has been Installed


Setup Complete

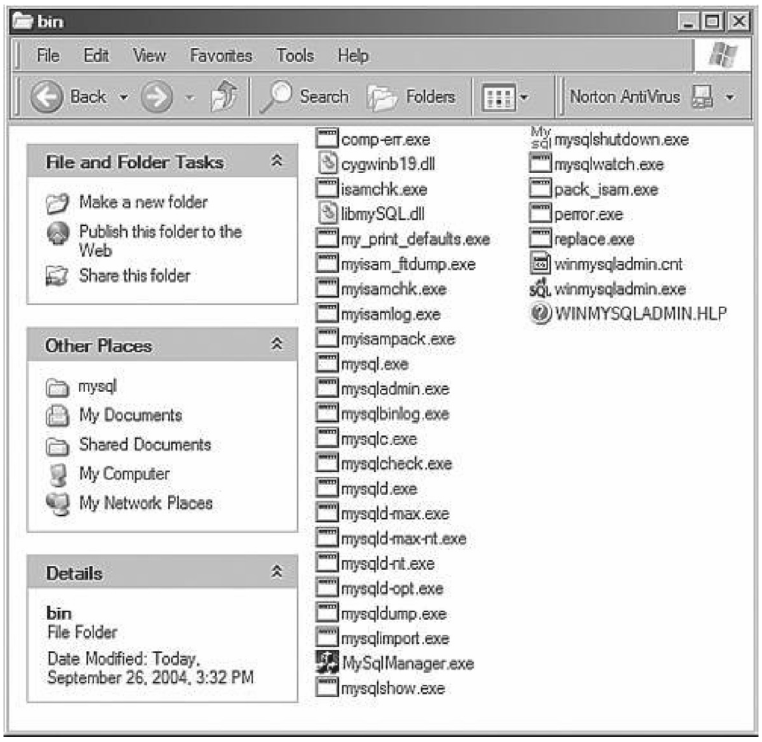
Setup has finished installing MySQL Servers and Clients 4.0.21 on your computer.

Setup can launch the Read Me file and MySQL Servers and Clients 4.0.21. Choose the options you want below.

Click Finish to complete Setup.

There are no fancy shortcuts installed in your Windows Start menu after an installation of My SQL from My SQL AB, so now you must start the process yourself. If you navigate to the My SQL applications directory (usually C:\My SQL\bin\ unless you changed your installation path), you will find numerous applications ready for action (see Figure 1.11).

Figure 1.11: A Directory Listing of My SQL Applications


```
comp-err.exe           mysqlshutdown.exe
cygwinb19.dll          mysqlwatch.exe
isamchk.exe            pack_isam.exe
libmySQL.dll           perror.exe
my_print_defaults.exe  replace.exe
myisam_ftdump.exe      winmysqladmin.cnt
myisamchk.exe          winmysqladmin.exe
myisamlog.exe          WINMYSQLADMIN.HLP
myisampack.exe
mysql.exe
mysqladmin.exe
mysqlbinlog.exe
mysqlc.exe
mysqlcheck.exe
mysqld.exe
mysqld-max.exe
mysqld-max-nt.exe
mysqld-nt.exe
mysqld-opt.exe
mysqldump.exe
mysqlimport.exe
MySqlManager.exe
mysqlshow.exe
```

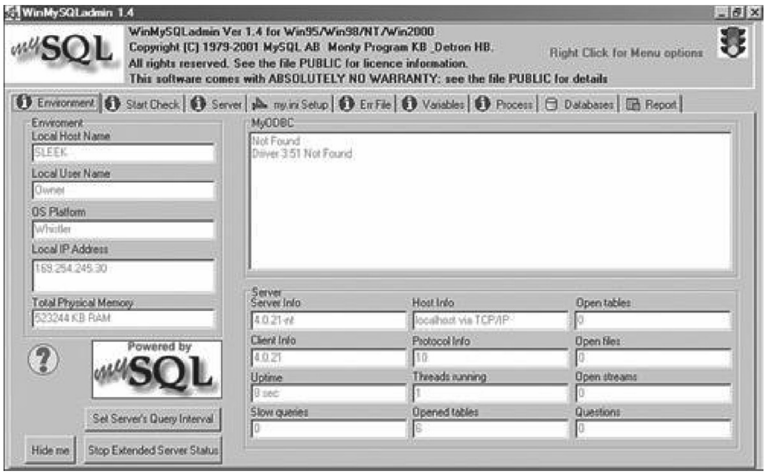
Details
bin
File Folder
Date Modified: Today, September 26, 2004, 3:32 PM

Notes

The winMy SQLadmin.exe application is a great friend to Windows users who are just getting started with My SQL. If you double-click this file, it will start the My SQL server and place a stoplight icon in your taskbar.

When the interface launches, you will see an application that provides an easy way to maintain and monitor your new server (see Figure 1.12).

Figure 1.12: WinMy SQLadmin Started and Ready for Action



WinMySQLadmin 1.4

WinMySQLadmin Ver 1.4 for Win95/Win98/NT/Win2000
Copyright (C) 1979-2001 MySQL AB Monty Program KB _Detron HB.
All rights reserved. See the file PUBLIC for licence information.
This software comes with ABSOLUTELY NO WARRANTY: see the file PUBLIC for details

Right Click for Menu options

Environment | Start Check | Server | my.ini Setup | Err File | Variables | Process | Databases | Report

Enviroment
Local Host Name: SLEEK
Local User Name: Owner
OS Platform: Whistler
Local IP Address: 169.254.245.30
Total Physical Memory: 523244 KB RAM

MyODBC
Not Found
Driver 3.51 Not Found

Server

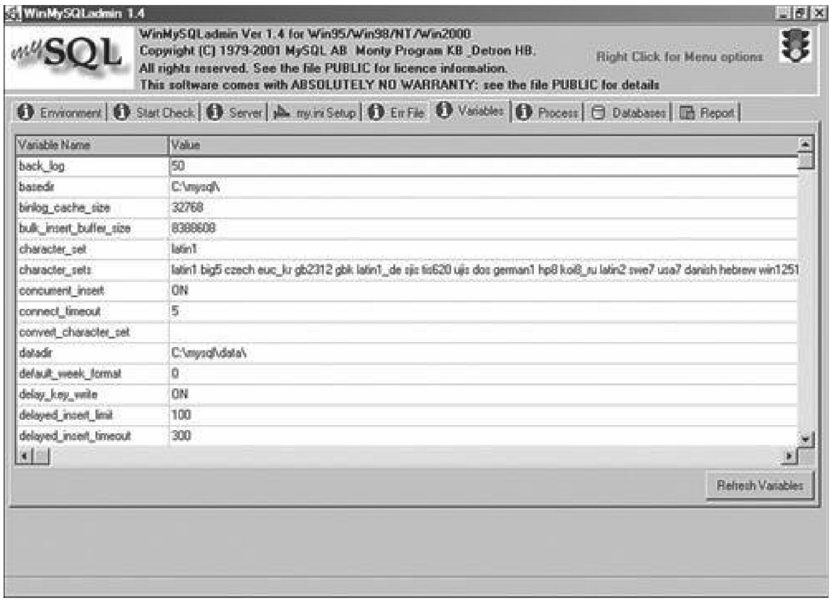
Server Info: 4.0.21-nt	Host Info: localhost via TCP/IP	Open tables: 0
Client Info: 4.0.21	Protocol Info: 10	Open files: 0
Uptime: 0 sec	Threads running: 1	Open streams: 0
Slow queries: 0	Opened tables: 6	Questions: 0

Set Server's Query Interval | Hide me | Stop Extended Server Status

WinMy SQLadmin will automatically interpret environment information, such as IP address and machine name. The tabs across the top allow you to view system information and edit My SQL configuration options.

For example, if you select the Variables tab, as shown in Figure 1.13, you can also view server configuration information. This information is similar to the output of the My SQL SHOW VARIABLES command.

Figure 1.13: Server Configuration Information



WinMySQLadmin 1.4

WinMySQLadmin Ver 1.4 for Win95/Win98/NT/Win2000
Copyright (C) 1979-2001 MySQL AB Monty Program KB _Detron HB.
All rights reserved. See the file PUBLIC for licence information.
This software comes with ABSOLUTELY NO WARRANTY: see the file PUBLIC for details

Right Click for Menu options

Environment | Start Check | Server | my.ini Setup | Err File | Variables | Process | Databases | Report

Variable Name	Value
back_log	50
basedir	C:\mysql\
binlog_cache_size	32768
bulk_insert_buffer_size	8388608
character_set	latin1
character_sets	latin1 big5 czech euc_kr gb2312 gbk latin1_de sjis tis620 ujis dos german1 hp8 koi8_ru latin2 swe7 usa7 danish hebrew win1251
concurrent_insert	ON
connect_timeout	5
convert_character_set	
datadir	C:\mysql\data\
default_week_format	0
delay_key_write	ON
delayed_insert_limit	100
delayed_insert_timeout	300

Refresh Variables

To shut down the My SQL server and/or the WinMy SQLadmin tool, right-click again on the stoplight icon in your taskbar and select the appropriate option (stop or start). As long as the My SQL server is running, you can run additional applications through a console window, such as the My SQL.

If you have any problems during the installation of My SQL, the first place you should look is the "Problems and Common Errors" unit of the My SQL manual, which is located at http://www.My SQL.com/doc/P/r/Problems.html.

The following are some common problems:

On Linux/Unix and Mac OS X, incorrect permissions do not allow you to start the My SQL daemon. If this is the case, be sure you have changed owners and groups to match those indicated in the installation instructions.

If you see the message Access denied when connecting to My SQL, be sure you are using the correct username and password.

If you see the message Can't connect to server, make sure the My SQL daemon is running.

When defining tables, if you specify a length for a field whose type does not require a length, the table will not be created. For example, you should not specify a length when defining a field as TEXT (as opposed to CHAR or VARCHAR).

If you still have trouble after reading the manual, sending email to the My SQL mailing list (see http://lists.My SQL.com/ for more information) will likely produce results. You can also purchase support contracts from My SQL AB for a very low fee.

1.3 Basic Security Guidelines

Regardless of whether you are running My SQL on Windows, Linux/Unix, or Mac OS X, and no matter whether you administer your own server or use a system provided to you by your Internet service provider, you must understand basic security guidelines. If you are accessing My SQL through your Internet service provider, there are several aspects of server security that you, as a non-root user, should not be able to modify or circumvent. Unfortunately, many Internet service providers pay no mind to security guidelines, leaving their clients exposedand for the most part, unaware of the risk.

1.3.1 Starting My SQL

Securing My SQL begins with the server startup procedure. If you are not the administrator of the server, you won't be able to change this, but you can certainly check it out and report vulnerabilities to your Internet service provider.

If your My SQL installation is on Linux/Unix or Mac OS X, your primary concern should be the owner of the My SQL daemonit should not be root. Running the daemon as a non-root user such as My SQL or database will limit the ability of malicious individuals to gain access to the server and overwrite files.

Caution Running the daemon as a non-root user such as My SQL or database will limit the ability of malicious individuals to gain access to the server and overwrite files.

You can verify the owner of the process using the ps (process status) command on your Linux/ Unix or Mac OS X system. The following output shows My SQL running as a non-root user (see the first entry on the second line):

```
# ps auxw|grep My SQLd
My SQL 153 0.0 0.6 12068 2624 ? S Nov16 0:00 /usr/local/bin/My SQL/bin/My SQLd
—defaults-extra-file=/usr/local/bin/My SQL/data/my.cnf
—basedir=/usr/local/bin/My SQL—datadir=/usr/local/bin/My SQL/data
—user=My SQL—pid-file=/usr/local/bin/My SQL/data/mike.pid—skip-locking
```

The following output shows My SQL running as the root user (see the first entry on the second line):

```
# ps auxw|grep My SQLd
root 21107 0.0 1.1 11176 1444 ? S Nov 27 0:00 /usr/local/My SQL/bin/My SQLd
—basedir=/usr/local/My SQL —datadir=/usr/local/My SQL/data —skip-locking
```

If you see that My SQL is running as root on your system, immediately contact your Internet service provider and complain. If you are the server administrator, you should start the My SQL process as a non-root user or specify the username in the startup command line:

```
My SQLd —user=non_root_user_name
```

For example, if you want to run My SQL as user My SQL, use

```
My SQLd —user=My SQL
```

However, the recommended method for starting My SQL is through the safe_My SQLd startup script in the bin directory of your My SQL installation:

```
# /usr/local/bin/My SQL/bin/safe_My SQLd &
```

1.3.2 Securing Your My SQL Connection

You can connect to the My SQL monitor or other My SQL applications in several different ways, each of which has its own security risks. If your My SQL installation is on your own workstation, you have less to worry about than users who have to use a network connection to reach their server.

If My SQL is installed on your workstation, your biggest security concern is leaving your workstation unattended with your My SQL monitor or My SQL GUI administration tool up and running. In this type of situation, anyone can walk over and delete data, insert bogus data, or shut down the server. Utilize a screen saver or lock screen mechanism with a password if you must leave your workstation unattended in a public area.

If My SQL is installed on a server outside your network, the security of the connection should be of some concern. As with any transmission of data over the Internet, it can be intercepted. If the transmission is unencrypted, the person who intercepted it can piece it together and use the information. Suppose the unencrypted transmission is your My SQL login informationa rogue individual now has access to your database, masquerading as you.

One way to prevent this from happening is to connect to My SQL through a secure connection. Instead of using Telnet to reach the remote machine, use SSH. SSH looks and acts like Telnet, but all transmissions to and from the remote machine are encrypted. Similarly, if you use a Web-based administration interface, such as phpMyAdmin (see http:/phpmyadmin.sourceforge. net for more information) or another tool used by your Internet service provider, access that tool over a secure HTTP connection.

Task Write down all the steps to install My SQL in your system.

Notes

Case Study

Success Story on Big Fish Games Triples Database

Big Fish Games is a global leader in the online games industry and distributes more games worldwide than any other online site. Within three years of its debut, BigFishGames.com rocketed into the Top 10 game portals on the Web and now serves millions of downloads every day.

Their Business Challenge

BigFishGames.com is a fast-growing website with over 25 million unique customer accounts and over 2.5 million visitors per month. In addition to the English site, Big Fish Games also offers international game portals in Japanese, German, French and Spanish. Their ever-growing user base is a huge boost to their business, but it also raises big challenges around IT capacity planning. To ensure the highest quality game experience, Big Fish Games has to accurately predict demand and increase bandwidth at the right time to keep a balance between over-utilizing the system, introducing delays and a bad user experience, and under-utilizing the system, resulting in a waste of capacity and money.

Their My SQL Solution

Big Fish Games started using My SQL as a small start-up. My SQL allowed Big Fish Games to quickly grow their business with lower cost and hardware requirements, and has scaled with the company as it has grown into an industry leader. Today, Big Fish Games deploys 40 My SQL servers to power its popular gaming website which offers thousands of games, with new games introduced every day. To achieve the scalability and reliability required by this high-trafficked website, Big Fish Games relies on My SQL Replication. Plus, DRBD is used to improve high availability. In addition to customer-facing material such as the dynamic website content, e-commerce store, game coupons and discussion forums, the My SQL database is also used for internal operations, tracking game downloads, account authentication, game activations and server logs.

My SQL Query Analyzer

In order to accommodate the growth in website traffic, the DBA team at Big Fish Games has been looking into opportunities to improve application performance. Tuning and optimizing the database is one of the options, but it won't help if the performance problem is caused by poorly-written SQL code.

To gain insights into the quality of the SQL code and execution statistics, Big Fish Games has been using the command line tools to identify target areas for potential performance improvement. However, for every problem resolution, extra effort was required to combine information from multiple sources because each command only provided a limited perspective.

Now, the My SQL Query Analyzer provides a consolidated view of query activities and execution details, and has enabled Big Fish Games to quickly identify poorly running queries and tackle the root causes directly in the SQL code. With the help of the My SQL Query Analyzer, the DBA team caught a "bad" query running 400,000 times overnight which never showed up in query logs. Furthermore, the My SQL Query Analyzer is very easy to use and doesn't require the user to be a world-class My SQL expert to fully leverage its benefits.

Since the Query Analyzer uses a Service Agent listening to application queries and performances metrics, the My SQL servers can always be live and operational when being

Contd...

analyzed. There is no need to switch the servers back and forth between on-line and off-line, which eliminates unnecessary risks to server availability and reliability.

After deploying the My SQL Query Analyzer, Big Fish Games tripled its database performance within three days, rather than weeks.

My SQL Enterprise Monitor

Big Fish Games also relies on the My SQL Enterprise Monitor and the Dashboard graphs, which show the number of queries per second, CPU load and replication status, to ensure that the website is performing well. Big Fish Games finds the My SQL Enterprise Monitor valuable because it is built for My SQL and offers more relevant and useful information than generic monitoring tools.

The My SQL Enterprise Monitor provides critical data points for Big Fish Games to analyze and determine the optimal number of slaves to serve its current website traffic and to plan for the future capacity requirements. This tool also helps the DBA team to gain insight into the system status, usage patterns and potential problems, without having to wait to be notified by the operations group.

Big Fish Games chooses to deploy My SQL Enterprise for the following reasons:

- **High Performance:** My SQL provides fast transaction speed to serve over 300,000 simultaneous users on BigFishGames.com.
- **Ease of use:** My SQL is very easy to use which allows DBAs to manage My SQL servers without a steep learning curve.
- **Low Maintenance:** Using My SQL Enterprise Monitor, Big Fish Games employs just two DBAs to monitor over 70 My SQL servers, 40 in active production and 30 in the testing environment.
- **Low TCO:** My SQL enabled Big Fish Games to launch their business, grow fast and establish themselves as the industry leader at a fraction of the cost compared to using a proprietary database.
- **Unlimited Deployment:** My SQL Enterprise Unlimited gives Big Fish Games the fixed-cost predictability to deploy additional servers without additional costs. This is especially beneficial for companies with rapidly growing data.
- **24x7 support:** My SQL offers top quality support, with long-time My SQL developers providing guaranteed 30 minutes response time for My SQL Enterprise Platinum customers. It’s invaluable for Big Fish Games to receive problem solving advice from My SQL support engineers when business-critical applications go down at midnight.
- **Support for popular Operating Systems:** My SQL is well-integrated with all major Linux, Solaris and Unix distributions, saving time for DBAs and improving administrative experience.
- **Support for C, C++, C#, PHP, Python, Ruby and Java:** My SQL supports drivers for a wide range of programming languages, including PHP, used by Big Fish Games for the front-end presentation layer, and Java, used with the Tomcat application server in the middleware layer.

Memcached

In addition to My SQL Replication, Big Fish Games further increases scalability by using Memcached, a distributed caching layer. All the web content is stored in Memcached, and

Contd...

most of the website queries are processed by this in-memory cache, which significantly improves response time as well as scalability.

Sun Fire x64 Servers

Big Fish Games utilizes a 3-tier server deployment strategy, and Sun's x64 servers have been chosen because of their excellent reputation for performance and reliability.

- Sun Fire X2100 server is best for applications which require lots of local disk space but less I/O or CPU speed.
- Sun Fire X4100 server works well for applications which demand fast processors but don't need speedy local disk I/O.
- Sun Fire X4140 server is optimal with 8 drive bays for applications where faster local disk I/O via RAID 10 and battery backed up write cache is essential.

By identifying the requirements for each application and the right server for each condition, Big Fish Games has gained 20x in performance by merely replacing an X4100 server with an X4140 machine.

Questions

1. What do you mean by My SQL Query Analyzer?
2. Explain My SQL Replication.

1.4 Summary

- My SQL is a key part of LAMP (Linux, Apache, My SQL, PHP/Perl/Python), the fast growing open source enterprise software stack.
- My SQL AB, the company responsible for creating and distributing My SQL, uses minor release numbers for updates security enhancements or bug fixes.
- The process of installing My SQL on Linux/Unix is straight forward, whether you use RPMs or install the binaries.
- If our my SQL installation is on Linux/Unix or Mac OS X, our primary concern should be the owner of My SQL daemonit should not be root.
- If My SQL is installed on our workstation, our biggest security concern is leaving our workstation unattended with our My SQL monitor or My SQL GUI administration tool up and running.

1.5 Keywords

My SQL Connection: You can connect to the My SQL monitor or other My SQL applications in several different ways, each of which has its own security risks.

My SQL Installation Process for Mac OS X: My SQL installation process for Mac OS X is fairly straightforwardthe developers from My SQL.

My SQL Privilege System: The My SQL privilege system is always on. The first time you try to connect, and for each subsequent action.

My SQL on Windows: The My SQL installation process on Windows is also quite simple the developers from My SQL AB have packaged up everything you need in one zip file with a setup program!

Notes

Securing My SQL: Securing My SQL begins with the server startup procedure. If you are not the administrator of the server, you won't be able to change this, but you can certainly check it out and report vulnerabilities to your Internet service provider.

Lab Exercise

1. Explain My SQL Query Analyzer.
2. Explain directory listing of My SQL applications.

1.6 Self Assessment Questions

1. Telnet is a perfectly acceptable method to securely connect to My SQL from a remote host.

   (*a*) True (*b*) False

2. Which three pieces of information does My SQL check each time a request is made?

   (*a*) Who you are?

   (*b*) Where you are accessing from?

   (*c*) What actions you're allowed to perform?

   (*d*) All of the above.

3. What command would you use to grant SELECT, INSERT, and UPDATE privileges to a user named bill on localhost to all tables on the BillDB database? Also, what piece of information is missing from this statement that is recommended for security purposes?

   (*a*) GRANT SELECT, INSERT, UPDATE

   ON BillDB.*

   TO bill@localcost

   (*b*) GRANT SELECT, INSERT, UPDATE

   ON BillDB.*

   TO bill@localhost

   (*c*) GRANT SELECT, INSERT, UPDATE

   ON BillDB.*

   TO bill@localhost

   (*d*) None of the above.

4. My SQL provides slow transaction speed to serve over 300,000 simultaneous users on BigFishGames.com

   (*a*) True (*b*) False

5. What are common problems in My SQL connection?

   (*a*) On Linux/Unix and Mac OS X, incorrect permissions do not allow you to start the My SQL daemon. If this is the case, be sure you have changed owners and groups to match those indicated in the installation instructions.

   (*b*) If you see the message Access denied when connecting to My SQL, be sure you are using the correct username and password.

   (*c*) If you see the message Can't connect to server, make sure the My SQL daemon is running.

   (*d*) All of the above.

6. My SQL is a key part of LAMP (Linux, Apache, My SQL, PHP/Perl/Python)
   (*a*) True (*b*) False

1.7 Review Questions

1. What are the system requirements to install My SQL?
2. Write steps to Installing My SQL on Mac OS X.
3. Explain Authentication Process.
4. What is Two-Step Authentication Process?
5. What if I tell my Internet service provider to stop running My SQL as root, and it won't?
6. What are common problems My SQL connection?
7. Explain Basic Security Guidelines.
8. Explain directory listing of My SQL applications.
9. What is the difference between SQL and SQL Server?
10. What are the advantages and disadvantages of primary key and foreign key in SQL?

Answers for Self Assessment Questions

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

1.8 Further Reading

Books *Teach Yourself PHP, My SQL & Apache*, By Meloni, Pearson Education.

Online link http://www.dev.My SQL.com/

Notes

Unit 2: Working with My SQL

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain SQL privilege.
- Discuss working with user privileges.

Introduction

Table privileges are effective for a certain table. SQL also supports privileges for an entire database, such as the privilege to create tables or views in a certain database.

Granting privileges on the database level is not supported by all SQL products. SQL supports the following database privileges:

- *SELECT.* This privilege gives the user the right to access all tables of the specified database with the SELECT statement.
- *INSERT.* This privilege gives the user the right to add rows to all tables of the specified database with the INSERT statement.
- *DELETE.* This privilege gives the user the right to remove rows from all tables of the specified database with the DELETE statement.
- *UPDATE.* This privilege gives the user the right to update values in all tables of the specified database with the UPDATE statement.
- *REFERENCES.* This privilege gives the user the right to create foreign keys that point to tables of the specified database.

- *CREATE*. This privilege gives the user the right to create new tables in the specified database with the CREATE TABLE statement.
- *ALTER*. This privilege gives the user the right to alter all tables of the specified database with the ALTER TABLE statement.
- *DROP*. This privilege gives the user the right to remove all tables of the specified database.
- *INDEX*. This privilege gives the user the right to define and remove indexes on all tables of the specified database.
- *CREATE TEMPORARY TABLES.* This privilege gives the user the right to create temporary tables in the specified database.
- *CREATE VIEW.* This privilege gives the user the right to create new views in the specified database with the CREATE VIEW statement.
- *CREATE ROUTINE.* This privilege gives the user the right to create new stored procedures and functions for the specified database.
- *ALTER ROUTINE.* This privilege gives the user the right to update and remove existing stored procedures and functions of the specified database.
- *EXECUTE ROUTINE.* This privilege gives the user the right to invoke existing stored procedures and functions of the specified database.
- *LOCK TABLES.* This privilege gives the user the right to block existing tables of the specified database.
- *ALL or ALL PRIVILEGES.* This privilege is a shortened form for all the privileges just named.

2.1 Privilege System in SQL

The My SQL privilege system is always on. The first time you try to connect, and for each subsequent action, My SQL checks the following three things:

Where you are accessing from (your host).

Who you say you are (your username and password).

What you're allowed to do (your command privileges).

All this information is stored in the database called My SQL, which is automatically created when My SQL is installed. There are several tables in the My SQL database:

columns_priv Defines user privileges for specific fields within a table.

db Defines the permissions for all databases on the server.

func Defines user-created functions.

host Defines the acceptable hosts that can connect to a specific database.

tables_priv Defines user privileges for specific tables within a database.

user Defines the command privileges for a specific user.

These tables will become more important to you later in this unit as you add a few sample users to My SQL. For now, just remember that these tables exist and must have relevant data in them in order for users to complete actions. Further discussed in working with user privileges.

The Two-Step Authentication Process

As you've learned, My SQL checks three things during the authentication process. The actions associated with these three things are performed in two steps:

1. My SQL looks at the host you are connecting from and the username and password pair that you are using. If your host is allowed to connect, your password is correct for your username, and the username matches one assigned to the host, My SQL moves to the second step.
2. For whichever SQL command you are attempting to use, My SQL verifies that you have the ability to perform that action for that database, table, and field.

If step 1 fails, you'll see an error about it and you won't be able to continue on to step 2. For example, suppose you are connecting to My SQL with a username of joe and a password of abc123 and you want to access a database called myDB. You will receive an error message if any of those connection variables is incorrect for any of the following reasons:

Your password is incorrect.

Username joe doesn't exist.

User joe can't connect from localhost.

User joe can connect from localhost but cannot use the myDB database.

You may see an error like the following:

```
#/usr/local/My SQL/bin/My SQL -h localhost -u joe -pabc123 test

Error 1045: Access denied for user: 'joe@localhost' (Using password: YES)
```

If user joe with a password of abc123 is allowed to connect from localhost to the myDB database, My SQL will check the actions that joe can perform in step 2 of the process. For our purposes, suppose that joe is allowed to select data but is not allowed to insert data. The sequence of events and errors would look like the following:

```
#/usr/local/My SQL/bin/My SQL -h localhost -u joe -pabc123 test

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Welcome to the My SQL monitor. Commands end with ; or \g.

Your My SQL connection id is 61198 to server version: 4.0.21-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

My SQL> select * from test_table;

+— —+— — — — — — —+

| id | test_field |

+— —+— — — — — — —+

+— —+— — — — — — —+
| 1  | blah       |
| 2  | blah blah  |
+— —+— — — — — —+
2 rows in set (0.0 sec)
My SQL> insert into test_table values ('', 'my text');
Error 1044: Access denied for user: 'joe@localhost' (Using password: YES)
```

Action-based permissions are common in applications with several levels of administration. For example, if you have created an application containing personal financial data, you might grant only SELECT privileges to entry-level staff members, but INSERT and DELETE privileges to executive-level staff with security clearances.

2.2 Working with User Privileges

In most cases when you are accessing My SQL through an Internet service provider, you will have only one user and one database available to you. By default, that one user will have access to all tables in that database and will be allowed to perform all commands. In this case, the responsibility is yours as the developer to create a secure application through your programming.

If you are the administrator of your own server or have the ability to add as many databases and users as you want, as well as modify the access privileges of your users, these next few sections will take you through the processes of doing so.

2.2.1 Adding Users Through My SQL

Administering your server through a third-party application might afford you a simple method for adding users, using a wizard-like process or a graphical interface. However, adding users through the My SQL monitor is not difficult, especially if you understand the security checkpoints used by My SQL, which you just learned.

The simplest method for adding new users is the GRANT command. By connecting to My SQL as the root user, you can issue one command to set up a new user. The other method is to issue INSERT statements into all the relevant tables in the My SQL database, which requires you to know all the fields in the tables used to store permissions. This method works just as well but is more complicated than the simple GRANT command.

The simple syntax of the GRANT command is:

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

The privileges you can grant are:

ALL Gives the user all the following privileges.

ALTER User can alter (modify) tables, columns, and indexes.

CREATE User can create databases and tables.

DELETE User can delete records from tables.

DROP User can drop (delete) tables and databases.

Notes

FILE User can read and write files; this is used to import or dump data.

INDEX User can add or delete indexes.

INSERT User can add records to tables.

PROCESS User can view and stop system processes; only trusted users should be able to do this.

REFERENCES Not currently used by My SQL, but a column for REFERENCES privileges exists in the user table.

RELOAD User can issue FLUSH statements; only trusted users should be able to do this.

SELECT User can select records from tables.

SHUTDOWN User can shut down the My SQL server; only trusted users should be able to do this.

UPDATE User can update (modify) records in tables.

USAGE User can connect to My SQL but has no privileges.

If, for instance, you want to create a user called john with a password of 99hjc!5, with SELECT and INSERT privileges on all tables in the database called myDB, and you want this user to be able to connect from any host, use

```
GRANT SELECT, INSERT

ON myDB.*

TO john@”%”

IDENTIFIED BY “99hjc!5”;
```

Note the use of two wildcards: * and %. These wildcards are used to replace values. In this example, * replaces the entire list of tables, and % replaces a list of all hosts in the known World a very long list indeed.

Here’s another example of adding a user using the GRANT command, this time to add a user called jane with a password of 45sdg11, with ALL privileges on a table called employees in the database called myCompany. This new user can connect only from a specific host:

```
GRANT ALL

ON myCompany.employees

TO jane@janescomputer.company.com

IDENTIFIED BY “45sdg11”;
```

If you know that janescomputer.company.com has an IP address of 63.124.45.2, you can substitute that address in the hostname portion of the command, as follows:

```
GRANT ALL

ON myCompany.employees

TO jane@’63.124.45.2’

IDENTIFIED BY “45sdg11”;
```

One note about adding users: Always use a password and make sure that the password is a good one! My SQL allows you to create users without a password, but that leaves the door wide open should someone with bad intentions guess the name of one of your users with full privileges granted to them!

If you use the GRANT command to add users, the changes will immediately take effect. To make absolutely sure of this, you can issue the FLUSH PRIVILEGES command in the My SQL monitor to reload the privilege tables.

2.2.2 Removing Privileges of My SQL

Removing Privileges is as simple as adding them; instead of a GRANT command, you use REVOKE. The REVOKE command syntax is:

REVOKE privileges.

ON databasename.tablename.

FROM username@hostname;

In the same way that you can grant permissions using INSERT commands, you can also revoke permissions by issuing DELETE commands to remove records from tables in the My SQL database. However, this requires that you be familiar with the fields and tables, and it’s just much easier and safer to use REVOKE.

To revoke the ability for user john to INSERT items in the myCompany database, you would issue this REVOKE statement:

REVOKE INSERT

ON myDB.*

FROM john@”%”;

Changes made to the data in the privilege tables happen immediately, but in order for the server to be aware of your changes, issue the FLUSH PRIVILEGES command in the My SQL monitor.

Installing My SQL on Windows and Mac OS X is a very simple process, thanks to a wizard-based installation method. My SQL AB provides a GUI-based administration tool for Windows users, called WinMy SQLadmin. Linux/Unix users do not have a wizard-based installation process, but it’s not difficult to follow a simple set of commands to unpack the My SQL client and server binaries. Linux/Unix users can also use RPMs for installation.

Security is always a priority and there are several steps you can take to ensure a safe and secure installation of My SQL. Even if you are not the administrator of the server, you should be able to recognize breaches and raise a ruckus with the server administrator!

The My SQL server should never run as the root user. Additionally, named users within My SQL should always have a password and their access privileges should be well defined.

My SQL uses the privilege tables in a two-step process for each request that is made. My SQL needs to know who you are and where you are connecting from and each of these pieces of information must match an entry in its privilege tables. Also, the user whose identity you are using must have specific permission to perform the type of request you are making.

You can add user privileges using the GRANT command, which uses a simple syntax to add entries to the user table in the My SQL database. The REVOKE command, which is equally simple, is used to remove those privileges.

```
GRANT SELECT, INSERT, UPDATE
ON BillDB.*
TO bill@localhost;
```

The important missing piece is a password for the user.

Activities

Think of situations in which you might want to restrict command access at the table level. For example, you wouldn't want the intern-level administrator to have shutdown privileges for the corporate database.

If you have administrative privileges in My SQL, issue several GRANT commands to create dummy users. It doesn't matter whether the tables and databases you name are actually present.

Use REVOKE to remove some of the privileges of the users you created.

Case Study

History of Microsoft SQL Server

SQL Server 2005 is the latest version of a database server product that has been evolving since the late 1980s. Microsoft SQL Server originated as Sybase SQL Server in 1987. In 1988, Microsoft, Sybase, and Aston-Tate ported the product to OS/2. Later, Aston-Tate dropped out of the SQL Server development picture, and Microsoft and Sybase signed a co-development agreement to port SQL Server to Windows NT. The co-development effort cumulated in the release of SQL Server 4.0 for Windows NT. After the 4.0 release, Microsoft and Sybase split on the development of SQL Server; Microsoft continued forward with future releases targeted for the Windows NT platform while Sybase moved ahead with releases targeted for the UNIX platform, which they still market today. SQL Server 6.0 was the first release of SQL Server that was developed entirely by Microsoft. In 1996, Microsoft updated SQL Server with the 6.5 release. After a two-year development cycle, Microsoft released the vastly updated SQL Server 7.0 release in 1998. SQL Server 7.0 embodied many radical changes in the underlying storage and database engine technology used in SQL Server. SQL Server 2000, the accumulation of another two-year development effort, was released in September 2000. The move from SQL Server 7.0 to SQL Server 2000 was more of an evolutionary move that didn't entail the same kinds of massive changes that were made in the move from 6.5 to 7.0. Instead, SQL Server 2000 built incrementally on the new code base that was established in the 7.0 release. Starting with SQL Server 2000, Microsoft began releasing updates to the basic release of SQL Server in the following year starting with XML for SQL Server Web Release 1, which added several XML features including the ability to receive a result set as an XML document. The next year they renamed the web release to the more succinctly titled SQLXML 2.0, which, among other things, added the ability to update the SQL Server database using XML update grams. This was quickly followed by the SQLXML 3.0 web release, which included the ability to expose stored procedures as web services. Two years later, Microsoft SQL Server release history cumulates with the release of SQL Server 2005. SQL Server 2005 uses the same basic architecture that was established with SQL Server 7 and it adds to this all the features introduced with SQL Server 2000 and its web releases in conjunction with the integration of the .NET CLR and an array of powerful new BI functions. The following timeline summarizes the development history of SQL Server:

Contd...

– **1987** Sybase releases SQL Server for UNIX.

– **1988** Microsoft, Sybase and Aston-Tate port SQL Server to OS/2.

– **1989** Microsoft, Sybase and Aston-Tate release SQL Server 1.0 for OS/2.

– **1990** SQL Server 1.1 is released with support for Windows 3.0 clients.

– Aston-Tate drops out of SQL Server development.

– **1991** Microsoft and IBM end joint development of OS/2.

- **1992** Microsoft SQL Server 4.2 for 16-bit OS/2 1.3 is released.

- **1992** Microsoft and Sybase port SQL Server to Windows NT.

- **1993** Windows NT 3.1 is released.

- **1993** Microsoft and Sybase release version 4.2 of SQL Server for Windows NT.

- **1994** Microsoft and Sybase co-development of SQL Server officially ends.

- Microsoft continues to develop the Windows version of SQL Server.

– Sybase continues to develop the UNIX version of SQL Server.

– **1995** Microsoft releases version 6.0 of SQL Server.

– **1996** Microsoft releases version 6.5 of SQL Server.

– **1998** Microsoft releases version 7.0 of SQL Server.

– **2000** Microsoft releases SQL Server 2000.

– **2001** Microsoft releases XML for SQL Server Web Release 1 (download).

– **2002** Microsoft releases SQLXML 2.0 (renamed from XML for SQL Server).

– **2002** Microsoft releases SQLXML 3.0.

– **2005** Microsoft releases SQL Server 2005 on November 7th, 2005.

Questions

1. Explain SQL Server.

2. Give the time line of all SQL server version.

2.3 Summary

- My SQL looks at the host you are connecting from and the username and password pair that you are using. If your host is allowed to connect, your password is correct for your username and the username matches one assigned to host, My SQL moves to next step.
- For whichever SQL command you are attempting to use, My SQL verifies that you have the ability to perform that action for that database, table and field.
- Administering your server through a third party application might afford you a simple method for adding users, a wizard-like process or a graphical interface.
- The simplest method for adding new users is the GRANT command.
- For removing privileges we have to use REVOKE command.

2.4 Keywords

Grant Command: This is the simplest method for adding new users.

Column: You cannot create a table without specifying any column name.

Flush: Privileges command is used to reload the privileges tables.

Revoke: Command is used to remove the privileges.

Table: A table must have atleast one column.

Lab Exercise Explain directory listing of My SQL applications.

2.5 Self Assessment Questions

1. The simplest method for adding new users is _______________ command.

   (*a*) Revoke (*b*) Insert

   (*c*) Grant (*d*) Update

2. _______________ command is used to remove the privileges.

   (*a*) Grant (*b*) Insert

   (*c*) Update (*d*) Revoke

3. Users can drop (delete) tables and databases through.

   (*a*) Delete (*b*) Drop

   (*c*) Insert (*d*) Alter

4. _______________ is used to import or dump data.

   (*a*) File (*b*) Select

   (*c*) Flush (*d*) Delete

5. The FLUSH PRIVILEGES command in the My SQL monitor to ___________ the privileges table.

   (*a*) Insert (*b*) Reload

   (*c*) Delete (*d*) Drop

2.6 Review Questions

1. How will you connect SQL to the server?
2. How will you create a database?
3. What is CREATE TABLE?
4. Explain two-step authentication process.
5. What do you mean by GRANT Command and give its syntax?

Answers for Self Assessment Questions

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

2.7 Further Reading

Books

Teach Yourself PHP, My SQL & Apache, By Meloni, Pearson Education.

Online link

http://www.dev.My SQL.com/

Unit 3: Apache Server

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand versions of apache.
- To understand different installation method for apache.

Introduction

The Web is still a very young phenomenon. Tim Berners-Lee invented the Web in late 1990 while working at CERN, the European Laboratory for Particle Physics. He developed it so that physicists working at various universities around the world could have instantaneous access to information, to enable their collaboration on a variety of projects.

Tim defined URLs, HTTP, and HTML and, with Robert Cailliau, wrote the first Web server and the first Web client software, which was later dubbed *a browser.*

Just a few years ago, it would have been necessary to explain what these concepts meant to all but the most technically aware audience. Now, there are few people (at least in developed nations) who are unaware of the WWW.

Shortly after Tim's initial work, a group at the National Center for Supercomputing Activities (NCSA) at the University of Illinois at Urbana-Champaign (UIUC) developed the NCSA HTTPd Web server and the NCSA Mosaic graphical Web browser. Mosaic wasn't the first graphical Web browser, although it's almost universally remembered as such. That honor rightfully belongs to Viola, written by Pei Wei and available before Mosaic. But Mosaic quickly stole the spotlight and most users becoming the most widely used Web browser sometime in 1992.

NCSA HTTPd was the server most used on the Web for the first several years of its existence. However, in 1994, Rob McCool, who had developed NCSA HTTPd, left NCSA, and the project fizzled. There was no longer any central organization collecting fixes, developing new features, and distributing a functional product.

Since the source code of the server was publicly available, many people using it had developed their own bug fixes and additional features that they needed for their own sites. These patches were shared rather haphazardly via Usenet, but there wasn't a centralized mechanism for collecting and distributing these patches.

Thus, Apache like the World-wide Web was put together largely by volunteers. Although the demise of the NCSA HTTPd project left developers with a product that didn't work very well at the time and no one to complain to a far superior product resulted in the long run.

3.1 Versions of Apache

3.1.1 Apache HTTP Server Version 2.0

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

3.1.2 Apache Mod

This seems to be accurate as of Apache 2.0.39. At least it will give a good starting point into what has traditionally been a sparsely documented area of Apache. This document is written from the Unix perspective, but should not need much modification to work under other platforms. In the past it was been very difficult for coders unfamiliar with the guts of Apache to get a start on making custom mods. Currently (July 2010) this is even more difficult because, though Apache 2.0 is shaping up nicely, the documentation available to the public has not caught up. Let's start by jumping into the fridged waters head first. Let's make a mod.

You can include custom mods in Apache in two ways. The first way is to build them into Apache. That is nice for a production quality mod, but when you're still developing the

Notes

mod, it is generally a pain to rebuild 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 (detailed more fully at http://httpd.apache.org/docs-2.0/install.html):

```
>./configure —prefix=PREFIX —enable-so

>make

>make install
```

Then it, for simplicity, make the user and group in httpd.conf match my own. They will look something like this:

```
User billyboebob

Group billyboebob
```

3.1.3 Hello World

Next you need a mod to run. Here is the code for a Hello World. We'll look at what it does in more detail shortly.

```
#include "httpd.h"

#include "http_config.h"

#include "http_core.h"

#include "http_log.h"

#include "http_protocol.h"

#include "ap_compat.h"

static void register_hooks(apr_pool_t *p);

static int helloworld2_handler(request_rec *r);

static void register_hooks(apr_pool_t *p)

{

    ap_hook_handler(helloworld2_handler, NULL, NULL, APR_HOOK_MIDDLE);

}

static int helloworld2_handler(request_rec *r)

{

     r->content_type = "text/html"

     ap_send_http_header(r);

     ap_rputs("<H1>Hello <i>Apache 2.0</i> World!</H1>", r);

     return OK;

}
```

Notes

```
 module AP_MODULE_DECLARE_DATA helloworld2_module =
{
STANDARD20_MODULE_STUFF, /* stuff that needs to be declared in
                              every 2.0 mod */
NULL,                    /* create per-directory config structure   */
NULL,                    /* merge per-directory config structures */
NULL,                    /* create per-server config structure      */
NULL,                    /* merge per-server config structures      */
NULL,                    /* command apr_table_t                      *
register_hooks           /* register hooks                          */
};
```

Take this c code name it 'mod_helloworld2.c' and save it. Now you can compile it using an apache tool called apxs. apxs is located in the bin Directory of the Apache 2.0.x build. The command that I used to compile this mod is 'apxs -c -i -a mod_helloworld2.c'. This will build our mod and install it.

What good is the code if you don't see it doing anything? 'cd' to the 'conf' directory. In this directory is the file 'httpd.conf'. Add these lines where it seems appropriate:

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

Task Give a mod to run, the code for a Hello World.

3.1.4 Module Definition

Every standard Apache mod needs an initial 'module' to describe it. By convention this is the last thing in the module. To define it use:

    module AP_MODULE_DECLARE_DATA foobar_module =

Replace 'foobar' with the name of the module (the file name minus any extensions).

3.1.5 Standard Module Stuff

There are 14 slots in a standard Apache 2.0 module. We're fortunate though. The macro 'STANDARD20 MODULE STUFF' predefines the first eight of these for us. The macro is defined in http_config.h if you desire to take a closer look at it. It mostly contains bookkeeping items such as the major and minor magic numbers for the Apache release.

3.1.6 Config Slots

The next four slots we do not use, so let's skip them. They have to do with host wide and directory wide configurations and access.

Commands

Special commands sent to the Apache server in the HTTP Header can be captured and handled with the function pointed to by this slot. 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 lots and lots of potential hooks. We only needed one type, the ap_hook_handler.

helloworld2_handler

Our actual handler takes the form of:

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

Where foobar is replaced with the name of the handler. 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.

3.2 Choosing Appropriate Installation Method

The Apache Web server is arguably the most popular Web server in use on the Internet today. Here are some of the reasons why Apache is so popular; you don't have to be running Windows to run Apache. It was developed on various Unix/Linux/BSD platforms, and then recently ported to Win32. Internet Information Server, a Web server made by Microsoft for the Windows NT platform, is made for use in the "Windows-only" world. While IIS has many features, it's lack of portability limits it's market share. Another reason for Apache's widespread acceptance is its overall stability. While you can slow down an Apache Web server, you can rarely, if ever, kill one. The Apache Web server service is near bulletproof.

Apache has been shown to be substantially faster, more stable, and more feature-full than many other web servers. Although certain commercial servers have claimed to surpass Apache's speed (it has not been demonstrated that any of these "benchmarks" are a good way of measuring WWW server speed at any rate). The developers of Apache feel that it is better to have a mostly fast free server than an extremely fast server that costs thousands of dollars. Apache is run on sites that get millions of hits per day, and they have experienced no performance difficulties.

3.2.1 Installation Options

You can download Apache from the Apache Software Foundation web site located at http://www.apache.org, in source and binary forms. While your downloading Apache, you may want to browse over the documentation.

3.2.2 What You Need

To install Apache, you will need the following things:

1. A computer running Linux.

2. Root access on this computer.
3. For binary and source installations, the tar and gunzip Unix utilities.

3.2.3 Binary Installation

A binary is pre-configured, which means someone else has gone to the trouble of configuring and building the software for you. There are, however, a few things you should keep in mind: 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 machine. 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/httpd/binaries/ and download the binary for your operating system (in our case, Linux). You'll most likely be presented with a directory containing multiple versions of Apache in various compressed forms. For the purposes of this guide, I'll assume you've downloaded the gzip'd form of the latest 2.0.x Apache binary (currently that's httpd-2.0.35-i686-pc-linux-rh72.tar.gz). If there is a README associated with the file you're downloading, you may want to review it for any interesting installation tidbits or possible bugs.

3.2.4 RPM Installation

Those of you running Red Hat Linux may want to take advantage of Red Hat's RPM ("RedHat Package Manager") system. Almost identical to a binary, an RPM is further customized to play nicely with other RPMs and provide a consistent interface to installing, updating, and removing binaries. For Linux newcomers or when installing a small standard component, RPMs are simple and reliable. Bear in mind that an Apache RPM may already be installed on your system depending on how Linux was originally installed on your computer. To find out, at the shell prompt, type:

rpm -qa | grep apache If you see something like apache-1.3.9xxx, an Apache RPM has already been installed. You can also type that command typing httpd instead of apache to see if it's installed.

If you don't have an Apache RPM, you must obtain one. Red Hat 7.3 ships apache-1.3.23-11.src.rpm in the RedHat/RPMS directory on the installation CD. Or, point your browser at ftp://ftp.redhat.com/pub/redhat/redhat-7.3-en/os/i386/RedHat/RPMS and download it. If you've not already done so, you'll need to become root. Navigate to the same directory as the .rpm file you obtained, and then type the following command, substituting the name of the .rpm you're using for example: apache-1.3.23-11.src.rpm.

```
rpm -ivh apache-1.3.23-11.src.rpm
```

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

Notes

3.2.5 Build from Source

Building Apache from source may seem like a daunting task 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.

Point your browser at http://www.apache.org/dist/httpd/ and download the gzip'd form of the current version of Apache (2.0.36 at the time of this writing).

Now let's uncompress that archive using gunzip and tar. You should replace the httpd-2.0.36.tar.gz below with the name of the gzip'd file you downloaded.

gunzip < httpd-2.0.36.tar.gz | tar xvf – You should end up with an httpd-2.0.x directory, x being the particular sub-version of Apache.

you downloaded. Move into the newly created directory. cd httpd-2.0.x.

Now we'll use the configure and make commands to 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

checking for chosen layout... Apache.

checking for working mkdir -p... yes.

checking build system type... i686-pc-linux-gnu.

checking host system type... i686-pc-linux-gnu.

checking target system type... i686-pc-linux-gnu.

Configuring Apache Portable Runtime library.

...

config.status: executing default commands
```

Unless errors were reported (not warnings), your Apache installation is now configured and we can move on. This is where things get a bit ugly. Make'ing Apache produces screenfulls of output.

```
make
```

Your screen should look something like:

```
# make

Making all in srclib

make[1]: Entering directory '/home/ryan/dl/apache_guide/httpd-
```

```
2.0.36/srclib'

Making all in apr

make[2]: Entering directory '/home/ryan/dl/apache_guide/httpd-

2.0.36/srclib/apr'

...

make[1]: Leaving directory '/home/ryan/dl/apache_guide/httpd-2.0.36'

#
```

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

```
# make install
```

Now Apache is installed.

3.2.6 Starting Apache

Let's take your new Apache installation out for a spin.

If you installed Apache using a binary or from scratch, as root, type:

```
/usr/local/apache/bin/apachectl start
```

If you used an RPM, as root, type:

```
/sbin/service httpd start
```

Point your browser at your brand new Web server, http://localhost/. If everything worked you should see the default home page.

3.2.7 Customize

Apache uses some rather easy to understand text files for configuration. On a Red Hat system, you'll find them in /etc/httpd/conf. Quite a few Linux distributions place them in this same place, but if you can't find such a directory, do a search for "httpd.conf". Once you find these, you've found the main config files. If you're new to Linux, and need help finding this file, here's how you can find it.

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

Now you should see where the file is located. When you move into the directory containing httpd.conf, you should see these three files:

- *httpd.conf* – This has the settings for the overall configuration for the server.
- *access.conf* – This file contains all the security settings for Apache.
- *srm.conf* – This file contains the MIME definitions and default document names for files on the server.

3.2.8 Restarting Apache

Whenever you make changes to the server configuration files, such as httpd.conf, they won't take effect until the server is restarted. In Linux, Apache can be restarted depending on how you installed it. If you installed Apache using a binary or from scratch, as root, type:

```
/usr/local/apache/bin/apachectl start
```

If you used an RPM, as root, type:

```
/sbin/service httpd start
```

After being restarted the changes will have taken effect.

Did u know? Every standard Apache mod needs an initial 'module' to describe it.

Case Study

The Apache project began in 1995 when a group of eight volunteers, seeing that web software was becoming increasingly commercialized, got together to create a supported open source web server. Apache began as an enhanced version of the public-domain NCSA server but steadily diverged from the original. Many new features have been added to Apache over the years: significant features include the ability for a single server to host multiple virtual web sites, a smorgasbord of authentication schemes, and the ability for the server to act as a caching proxy. In some cases, Apache is way ahead of the commercial vendors in the features wars. For example, at the time this book was written only the Apache web server had implemented the HTTP/1.1 Digest Authentication scheme.

Internally, the server has been completely redesigned to use a modular and extensible architecture, turning it into what the authors describe as a "web server toolkit." In fact, there's very little of the original NCSA httpd source code left within Apache. The main NCSA legacy is the configuration files, which remain backward-compatible with NCSA httpd.

Apache's success has been phenomenal. In less than three years, Apache has risen from relative obscurity to the position of market leader. Netcraft, a British market research company that monitors the growth and usage of the web, estimates that Apache servers now run on over 50 per cent of the Internet web sites, making it by far the most popular web server in the world. Microsoft, its nearest rival, holds a mere 22 per cent of the market. This is despite the fact that Apache has lacked some of the conveniences that common wisdom holds to be essential, such as a graphical user interface for configuration and administration.

Apache has been used as the code base for several commercial server products. The most successful of these, C2Net's Stronghold, adds support for secure communications with Secure Socket Layer (SSL) and a form-based configuration manager. There is also WebTen by Tenon Intersystems, a Macintosh PowerPC port, and the Red Hat Secure Server, an inexpensive SSL-supporting server from the makers of Red Hat Linux.

Another milestone was reached in November of 1997 when the Apache Group announced its port of Apache to the Windows NT and 95 operating systems (Win32). A fully multithreaded

Contd...

implementation, the Win32 port supports all the features of the Unix version and is designed with the same modular architecture as its brother. Freeware ports to OS/2 and the AmigaOS are also available.

In the summer of 1998, IBM announced its plans to join with the Apache volunteers to develop a version of Apache to use as the basis of its secure Internet commerce server system, supplanting the servers that it and Lotus Corporation had previously developed.

Why use Apache? Many web sites run Apache by accident. The server software is small, free, and well documented and can be downloaded without filling out pages of licensing agreements. The person responsible for getting his organization's web site up and running downloads and installs Apache just to get his feet wet, intending to replace Apache with a "real" server at a later date. But that date never comes. Apache does the job and does it well.

However, there are better reasons for using Apache. Like other successful open source products such as Perl, the GNU tools, and the Linux operating system, Apache has some big advantages over its commercial rivals.

Questions

1. What do you mean by Secure Socket Layer (SSL)?

2. Explain Linux operating system and GNU tools.

3.3 Summary

- Apache, like the world wide web was put together largely by volunteers. Although the demise of the NCSA HTTPd project left developers with a product that did not work very well at the time.
- There are two ways in apache mod. The first way is to build them into Apache and the second way is to include a mod is to build Apache to load mod at start up.
- Apache has been shown to be substantially faster, more stable, and more feature-full than many other web servers.

3.4 Keywords

***Apache Configuration File Structure*:** Apache keeps all of its configuration information in text files. The main file is called httpd.conf. This file contains directives and containers, which enable you to customize your Apache installation.

Apache Mod: This seems to be accurate as of Apache 2.0.39. At least it will give a good starting point into what has traditionally been a sparsely documented area of Apache.

HTTP Header: HTTP Header can be captured and handled with the function pointed by the slot.

***Module definition*:** Every standard Apache mod needs an initial 'module' to describe it. By convention this is the last thing in the module.

Register Hooks: This is a pointer to a function that details the hooks that the module handles.

Lab Exercise Give appropriate installation method of Apache.

Notes

3.5 Self Assessment Questions

1. Special commands sent to the Apache server in ______________ .
2. ______________ is a pointer to a function that details the hooks that this module handles.
3. Apache Web server is arguably the most popular Web server in use on the Internet today.

   (*a*) True (*b*) False
4. Apache Web server service is near ______________ .
5. A binary is pre-configured, which means someone else has gone to the trouble of configuring and building.

   (*a*) True (*b*) False

3.6 Review Questions

1. Define Apache.
2. Give various version of Apache.
3. Define Apache HTTP Server Version 2.0.
4. What is Apache mod?
5. Give appropriate installation method of Apache.
6. What is Binary Installation?
7. Give RPM Installation.

Answers for Self Assessment Questions

1. HTTP Header	2. Register hook	3. (*a*)	4. Bulletproof	5. (*a*)

3.7 Further Reading

Books *A Beginner's Guide by: Vaswani, Vikram,* By Tata MC-Graw Hill.

Online link http://www.gibmonts.com/C-plus/ch11level1sec10.htm/.

Unit 4: Apache Server Installation in Window

CONTENTS

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

The Apache Server combined with the power of PHP, MySQL, and phpMy Admin, creates one of the best possible development environments for a web programmer. Getting everything properly configured can take 20-30 minutes.

Installed Subversion and Apache on a Windows Server installation. It's relatively simplified, but you can do various different changes throughout to support your own environment.

4.1 Installation Method in Windows

You have several options when it comes to getting a basic Apache installation in place. Apache is open source, meaning that you can have access to the full source code of the software, which in turn enables you to build your own custom server. Additionally, pre-built Apache binary distributions are available for most modern Unix platforms. Finally, Apache comes already bundled with a variety of Linux distributions, and you can purchase commercial versions from software vendors such as Covalent Technologies and IBM. The examples in this hour will teach you how to build Apache from source if you are using Linux/Unix, and how to use the installer if you plan to run Apache on a Windows system.

4.1.1 Building from Source

Building from source gives you the greatest flexibility, as it enables you to build a custom server, remove modules you do not need and extend the server with third-party modules. Building Apache from source code enables you to easily upgrade to the latest versions and quickly apply security patches, whereas updated versions from vendors can take days or weeks to appear.

The process of building Apache from the source code is not especially difficult for simple installations, but can grow in complexity when third-party modules and libraries are involved.

4.1.2 Installing a Binary

Linux/Unix binary installations are available from vendors and can also be downloaded from the Apache Software Foundation Web site. They provide a convenient way to install Apache for users with limited system administration knowledge, or with no special configuration needs. Third party commercial vendors provide prepackaged Apache installations together with an application server, additional modules, support, and so on.

The Apache Software Foundation provides an installer for Windows systems—a platform where a compiler is not as commonly available as in Linux/Unix systems.

4.2 Apache Configuration File Structure

Apache keeps all of its configuration information in text files. The main file is called httpd. conf. This file contains directives and containers, which enable you to customize your Apache installation. Directives configure specific settings of Apache, such as authorization, performance and network parameters. Containers specify the context to which those settings refer. For example, authorization configuration can refer to the server as a whole, a directory or a single file.

4.2.1 Directives

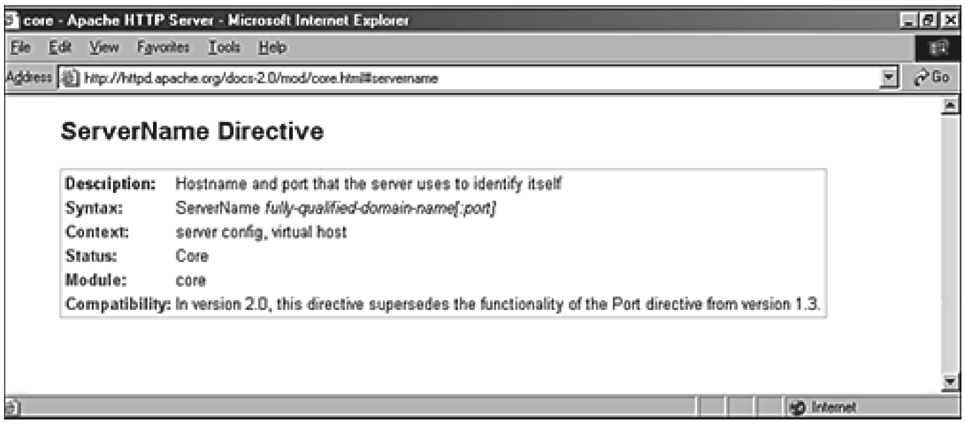
The following rules apply for Apache directive syntax:

- The directive arguments follow the directive name.
- Directive arguments are separated by spaces.
- The number and type of arguments vary from directive to directive; some have no arguments.
- A directive occupies a single line, but you can continue it on a different line by ending the previous line with a backslash character (\).
- The pound sign (£) should precede the directive, and must appear on its own line.

In the Apache server documentation, found online at http://httpd.apache.org/docs-2.0/, you can browse the directives in alphabetical order or by the module to which they belong. You'll soon learn about some of the basic directives, but you should supplement your knowledge using the online documentation.

Figure shows an entry from the documentation for the ServerName directive description. You can read this description in the online documentation at http://httpd.apache.org/docs-2.0/mod/core.html#servername.

Figure 4.1: Directive description example


core - Apache HTTP Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://httpd.apache.org/docs-2.0/mod/core.html#servername
Go
Internet

ServerName Directive

Description:	Hostname and port that the server uses to identify itself
Syntax:	ServerName *fully-qualified-domain-name[:port]*
Context:	server config, virtual host
Status:	Core
Module:	core
Compatibility:	In version 2.0, this directive supersedes the functionality of the Port directive from version 1.3.

Syntax: This entry explains the format of the directive options. Compulsory parameters appear in italics, optional parameters appear in italics and brackets.

Default: If the directive has a default value, it will appear here.

Context: This entry details the containers or sections in which the directive can appear. Containers are explained in the next section. The possible values are server config, virtual host, directory, and.htaccess.

Status: This entry indicates whether the directive is built in Apache (core), belongs to one of the bundled modules (base or extension, depending on whether they are compiled by default), is part of a Multi Processing Module (MPM), or is bundled with Apache but not ready for use in a production server (experimental).

Module: This entry indicates the module to which the directive belongs.

Compatibility: This entry contains information about which versions of Apache support the directive.

Override: Apache directives belong to different categories. The override field is used to specify which directive categories can appear in.htaccess per-directory configuration files. A brief explanation of the directive follows these entries in the documentation, and a reference to related directives or documentation may appear at the end.

4.2.2 Containers

Directive containers, also called sections, limit the scope for which directives apply. If directives are not inside a container, they belong to the default server scope (server config) and apply to the server as a whole.

These are the default Apache directive containers:

<VirtualHost>? A VirtualHost directive specifies a virtual server. Apache enables you to host different Web sites with a single Apache installation. Directives inside this container apply to a particular Web site. This directive accepts a domain name or IP address and an optional port as arguments. You will learn more about virtual hosts in Hour 22, "Apache Performance Tuning and Virtual Hosting."

<Directory>, <DirectoryMatch>? These containers allow directives to apply to a certain directory or group of directories in the file system. Directory containers take a directory or directory pattern argument. Enclosed directives apply to the specified directories and their subdirectories. The DirectoryMatch container allows regular expression patterns to be specified as an argument. For example, the following allows a match of all subdirectories of the www directory that are made up of four numbers, such as a directory named after a year and month (0902 for September 2010):

```
<DirectoryMatch "^/www/.*/[0-9]{4}">
```

<Location>, <LocationMatch>? These containers allow directives to apply to certain requested URLs or URL patterns. They are similar to their Directory counterparts. LocationMatch takes a regular expression as an argument. For example, the following matches directories containing either "/my/data" or "/your/data":

```
<LocationMatch "/(my|your)/data">
```

<Files>, <FilesMatch>? Similar to Directory and Location containers, Files sections allow directives to apply to certain files or file patterns.

Containers surround directives, as shown in Listing 1.

Listing 1 Sample Container Directives:

1. <Directory "/some/directory">
2. SomeDirective1
3. SomeDirective2
4. </Directory>
5. <Location "/downloads/*.html">

```
SomeDirective3
</Location>
<Files "\.(gif|jpg)">
SomeDirective4
</Files>
```

Sample directives *SomeDirective1* and *SomeDirective2* will apply to the directory /www/docs and its subdirectories. *SomeDirective3* will apply to URLs referring to pages with the .html extension under the/download/URL. *SomeDirective4* will apply to all files with .gif or .jpg extensions.

4.2.3 Conditional Evaluation

Apache provides support for conditional containers. Directives enclosed in these containers will be processed only if certain conditions are met.

- <IfDefine>? Directives in this container will be processed if a specific command line switch is passed to the Apache executable. The directive in Listing 2 will be processed only if the DMyModule switch was passed to the Apache binary being executed. You can pass this directly or by modifying the apachectl script, as described in the "Apache-Related Commands" section later in this hour.

  If Define containers allow the argument to be negated. That is, directives inside a <IfDefine !*MyModule*>section will be processed only if no -DMyModule parameter was passed as a command-line argument. For example, if DSSL is not passed, listening on the SSL port (usually 443) will not occur.

- <IfModule>? Directives in an IfModule section will be processed only if the module passed as an argument is present in the Web server. For example, Apache ships with a default httpd.conf configuration file that provides support for different MPMs. Only the configuration belonging to the MPM compiled in will be processed, as you can see in Listing 3. The purpose of the example is to illustrate that only one of the directive groups will be evaluated.

Listing 2 IfDefine Example

```
<IfDefine MyModule>
LoadModule my_module modules/libmymodule.so
</IfDefine>
```

Listing 3 IfModule Example

```
<IfModule prefork.c>
StartServers        5
MinSpareServers     5
MaxSpareServers     10
MaxClients          20
```

```
MaxRequestsPerChild      0
</IfModule>
}
<IfModule worker.c>
StartServers             3
MaxClients               8
MinSpareThreads          5
MaxSpareThreads          10
ThreadsPerChild          25
MaxRequestsPerChild      0
</IfModule>
```

4.2.4 ServerRoot

The ServerRoot directive takes a single argument: a directory path pointing to the directory where the server lives. All relative path references in other directives are relative to the value of ServerRoot. If you compiled Apache from source on Linux/Unix, as described earlier in this hour, the default value of ServerRoot is/usr/local/apache2. If you used the Windows installer, the ServerRoot is c:\Program Files\Apache Group.

4.2.5 Per Directory Configuration Files

Apache uses perdirectory configuration files to allow directives to exist outside the main configuration filehttpd.conf. These special files can be placed in the file system. Apache will process the content of these files if a document is requested in a directory containing one of these files or any subdirectories under it. The contents of all the applicable perdirectory configuration files are merged and processed. For example, if Apache receives a request for the/usr/local/apache2/htdocs/index.html file, it will look for per-directory configuration files in the/,/usr,/usr/local,/usr/local/apache2, and /usr/local/apache2/htdocsdirectories, in that order.

Enabling perdirectory configuration files has a performance penalty. Apache must perform expensive disk operations looking for these files in every request, even if the files do not exist.

Perdirectory configuration files are called .htaccess by default. This is for historical reasons; they were originally used to protect access to directories containing HTML files.

The directive AccessFileName enables you to change the name of the perdirectory configuration files from.htaccess to something else. It accepts a list of filenames that Apache will use when looking for perdirectory configuration files.

To determine whether a directive can be overridden in the perdirectory configuration file, check whether the Context: field of the directive syntax definition contains. htaccess.

Apache directives belong to different groups, specified in the Override: field in the directive syntax description. Possible values are:

- AuthConfig? Authorization directives.

- FileInfo? Directives controlling document types.
- Indexes? Directives controlling directory indexing.
- Limit? Directives controlling host access.
- Options? Directives controlling specific directory features.

You can control which of these directive groups can appear in perdirectory configuration files by using theAllowOverride directive. AllowOverride can also take an All or a None argument. All means that directives belonging to all groups can appear in the configuration file. None disables perdirectory files in a directory and any of its subdirectories. Listing 4 shows how to disable perdirectory configuration files for the server as a whole. This improves performance and is the default Apache configuration.

Listing 4 Disabling PerDirectory Configuration Files

```
<Directory />
AllowOverride none
</Directory>
```

4.3 Apache Log File

One of the many pieces of the Website puzzle is Web logs. Traffic analysis is central to most Websites, and the key to getting the most out of your traffic analysis revolves around how you configure your Web logs. Apache is one of the most if not the most powerful open source solutions for Website operations. You will find that Apache's Web logging features are flexible for the single Website or for managing numerous domains requiring Web log analysis.

For the single site, Apache is pretty much configured for logging in the default install. The initial httpd.conf file (found in/etc/httpd/conf/httpd.conf in most cases) should have a section on logs that looks similar to this (Apache 2.0.x), with descriptive comments for each item. Your default logs folder will be found in/etc/httpd/logs. This location can be changed when dealing with multiple Websites, as we'll see later. For now, let's review this section of log configuration.

4.3.1 Error Logs

The error log contains messages sent from Apache for errors encountered during the course of operation. This log is very useful for troubleshooting Apache issues on the server side.

Apache Log Tip: If you are monitoring errors or testing your server, you can use the command line to interactively watch log entries. Open a shell session and type "*tail –f /path/to/error_log*". This will show you the last few entries in the file and also continue to show new entries as they occur.

There are no real customization options available, other than telling Apache where to establish the file, and what level of error logging you seek to capture. First, let's look at the error log configuration code from httpd.conf.

```
ErrorLog logs/error_log
```

You may wish to store all error-related information in one error log. If so, the above is fine, even for multiple domains. However, you can specify an error log file for each individual domain you have. This is done in the <VirtualHost> container with an entry like this:

Notes

```
<VirtualHost 10.0.0.2>
DocumentRoot "/home/sites/domain1/html/"
ServerName domain1.com
ErrorLog /home/sites/domain1/logs/error.log
</VirtualHost>
```

If you are responsible for reviewing error log files as a server administrator, it is recommended that you maintain a single error log. If you're hosting for clients, and they are responsible for monitoring the error logs, it's more convenient to specify individual error logs they can access at their own convenience.

The setting that controls the level of error logging to capture follows below:
LogLevel warn

Apache's definitions for their error log levels are as follows:

Level	**Description**
Emerg	Emergencies-System is unusable
alert	Action must be taken immediately
Crit	Critical conditions
Error	Error conditions
Warn	Warning conditions
Notice	Normal but significant condition
Info	Informational
Debug	Debug-level messages

4.3.2 Apache Access Log File

Apache server records all incoming requests and all requests processed to a log file. The format of the access log is highly configurable. The location and content of the access log are controlled by the **CustomLog** directive. Default apache access log file location:

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

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

```
# grep CustomLog /usr/local/etc/apache22/httpd.conf

# grep CustomLog /etc/apache2/apache2.conf

# grep CustomLog /etc/httpd/conf/httpd.conf
```

Output

```
# a CustomLog directive.

  #CustomLog "/var/log/httpd-access.log" common

  CustomLog "/var/log/httpd-access.log" combined
```

4.3.3 Tracking Website

Notes

Often by default, Apache will generate three activity logs: access, agent and referrer. These track the accesses to your Website, the browsers being used to access the site and referring urls that your site visitors have arrived from.

It is commonplace now to utilize Apache's "combined" log format, which compiles all three of these logs into one logfile. This is very convenient when using traffic analysis software as a majority of these third-party programs are easiest to configure and schedule when only dealing with one log file per domain.

Let's break down the code in the combined log format and see what it all means.

LogFormat "%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i"" combined

LogFormat starts the line and simply tells Apache you are defining a log file type (or nickname), in this case, combined. Now let's look at the cryptic symbols that make up this log file definition.

%h	Logs the remote host
%1	Remote logname, if supplied
%u	Remote user (mostly useful if logging behind authentication)
%t	The date and time of the request
%r	The request to your web site
%s	The status of the request (201, 301, 404, 500, etc.). the > in front of the
"s"	insures only the last status is logged.
%b	Bytes sent for request (tracks http bandwidth use)
%i	Tracks items sent in the HTML header. So by adding (Referer) and (User Agent), we are capturing the referring url and the browser type in the combined log format

The most common is to identify individual log files for each domain. This is seen in the example below, again using the log directive within the <VirtualHost> container for each domain.

```
<VirtualHost 10.0.0.2>

DocumentRoot "/home/sites/domain1/html/"

ServerName domain1.com

ErrorLog /home/sites/domain1/logs/error.log

CustomLog /home/sites/domain1/logs/web.log

</VirtualHost>
<VirtualHost 10.0.0.3>
DocumentRoot "/home/sites/domain2/html/"
ServerName domain2.com
ErrorLog /home/sites/domain2/logs/error.log
CustomLog /home/sites/domain2/logs/web.log
</VirtualHost>

<VirtualHost 10.0.0.4>
DocumentRoot "/home/sites/domain3/html/"
```

Notes

```
ServerName domain3.com
ErrorLog /home/sites/domain3/logs/error.log
CustomLog /home/sites/domain3/logs/web.log
</VirtualHost>
```

In the above example, we have three domains with three unique Web logs (using the combined format we defined earlier). A traffic analysis package could then be scheduled to process these logs and generate reports for each domain independently.

This method works well for most hosts. However, there may be situations where this could become unmanageable. Apache recommends a special single log file for large virtual host environments and provides a tool for generating individual logs per individual domain.

We will call this log type the cvh format, standing for "common virtual host." Simply by adding a %v (which stands for virtual host) to the beginning of the combined log format defined earlier and giving it a new nickname of cvh, we can compile all domains into one log file, then automatically split them into individual log files for processing by a traffic analysis package.

```
LogFormat "%v %h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i"" cvh
```

In this case, we do not make any CustomLog entries in the <VirtualHost> containers and simply have one log file generated by Apache. A program created by Apache called split_logfile is included in the src/support directory of your Apache sources. from your master log file will be named for each domain (virtual host) and look like: virtualhost.log.

4.3.4 Log Rotation

Finally, we want to address log rotation. High traffic sites will generate very large log files, which will quickly swallow up valuable disk space on your server. You can use log rotation to manage this process. There are many ways to handle log rotation, and various third party tools are available as well. However, we're focusing on configurations native to Apache, so we will look at a simple log rotation scheme here. I'll include links to more flexible and sophisticated log rotation options in a moment.

This example uses a rudimentary shell script to move the current Web log to an archive log, compresses the old file and keeps an archive for as long as 12 months, then restarts Apache with a pause to allow the log files to be switched out.

```
mv web11.tgz web12.tgz

mv web10.tgz web11.tgz

mv web9.tgz  web10.tgz

mv web8.tgz  web9.tgz

mv web7.tgz  web8.tgz

mv web6.tgz  web7.tgz

mv web5.tgz  web6.tgz

mv web5.tgz  web6.tgz

mv web4.tgz  web5.tgz
mv web3.tgz  web4.tgz
mv web2.tgz  web3.tgz
mv web1.tgz  web2.tgz
mv web.tgz   web1.tgz
mv web.log   web.old
/usr/sbin/apachectl
graceful
sleep 300
tar cvfz web.tgz web.old
```

This code can be copied into a file called logrotate.sh, and placed inside the folder where your web log file is stored (or whatever you name your log file, e.g. access_log, etc.). Just be sure to modify for your log file names and also chmod (change permissions on the file) to 755 so it becomes an executable.

This works fine for a single busy site. If you have more complex requirements for log rotation, be sure to see some of the following sites. In addition, many Linux distributions now come with a log rotation included. For example, Red Hat 9 comes with logrotate.d, a log rotation daemon which is highly configurable. To find out more, on your Linux system with logrotate.d installed, type man logrotate.

4.4 Starting Apache for First Time

Before you start Apache, you should verify that the minimal set of information is present in the Apache configuration file, httpd.conf. The following sections describe the basic information needed to configure Apache and how to start the server.

Using Apache With Microsoft Windows

This document explains how to install, configure and run Apache 1.3 under Microsoft Windows. Most of this document assumes that you are installing Windows from a binary distribution. If you want to compile Apache yourself (possibly to help with development, or to track down bugs).

Warning: Apache on NT has not yet been optimized for performance: Apache still performs best, and is most reliable on Unix platforms. Over time NT performance has improved, and great progress is being made in the upcoming version 2.0 of Apache for the Windows platforms. Folks doing comparative reviews of webserver performance are still asked to compare against Apache on a Unix platform such as Solaris, FreeBSD or Linux.

- Installing Apache for Windows (binary install)
- Running Apache for Windows
- Testing Apache for Windows
- Configuring Apache for Windows

Notes

Installing Apache for Windows

Run the Apache .msi file you downloaded above. This will prompt you for:

- Whether or not you want to run Apache for all users (installing Apache as a Service), or if you want it installed to run in a console window when you choose the Start Apache shortcut.
- Your Server name, Domain name and administrative email account.
- The directory to install Apache into (the default is C:\Program Files\Apache Group\Apache although you can change this to any other directory you wish)
- The installation type. The "Complete" option installs everything, including the source code if you downloaded the -src.msi package. Choose the "Custom" install if you choose not to install the documentation, or the source code from that package.

During the installation, Apache will configure the files in the conf directory for your chosen installation directory. However, if any of the files in this directory already exist they will not be overwritten. Instead the new copy of the corresponding file will be left with the extension default.conf. So, for example, if conf\httpd.conf already exists it will not be altered, but the version which would have been installed will be left in conf\httpd.default.conf. After the installation has finished you should manually check to see what in new in the .default.conf file, and if necessary update your existing configuration files.

Also, if you already have a file called htdocs\index.html then it will not be overwritten (no index.html.default file will be installed either). This should mean it is safe to install Apache over an existing installation (but you will have to stop the existing server running before doing the installation, then start the new one after the installation is finished).

After installing Apache, you should edit the configuration files in the conf directory as required. These files will be configured during the install ready for Apache to be run from the directory where it was installed, with the documents served from the subdirectory htdocs. There are lots of other options which should be set before you start really using Apache. However to get started quickly the files should work as installed.

If you eventually uninstall Apache, your configuration and log files will not be removed. You will need to delete the installation directory tree ("C:\Program Files\Apache Group" by default) yourself if you do not care to keep your configuration and other web files. Since the httpd.conf file is your accumulated effort in using Apache, you need to take the effort to remove it. The same happens for all other files you may have created, as well as any log files Apache created.

Running Apache for Windows

There are two ways you can run Apache:

- As a "service". This is the best option if you want Apache to automatically start when your machine boots, and to keep Apache running when you log-off.
- From a console window. Closing this console window will terminate the Apache server.

Complete the steps below before you attempt to start Apache as a Windows "service"!

To run Apache from a console window, select the "Start Apache as console app" option from the Start menu (in Apache 1.3.4 and earlier, this option was called "Apache Server"). This will open a console window and start Apache running inside it. The window will remain active until you stop Apache. To stop Apache running, either press select the "Shutdown Apache console app" icon option from the Start menu (this is not available in Apache 1.3.4 or earlierIn Apache 1.3.13 and above it is now quite safe to press Ctrl+C or Ctrl+Break to stop the Apache in the console window.

Testing Apache for Windows

If you have trouble starting Apache please use the following steps to isolate the problem. This applies if you started Apache using the "Start Apache as a console app" shortcut from the Start menu and the Apache console window closes immediately (or unexpectedly) or if you have trouble starting Apache as a service.

Run the "Command Prompt" from the Start Menu - Programs list. Change to the folder to which you installed Apache, type the command apache, and read the error message. Then review the error.log file for configuration mistakes. If you accepted the defaults when you installed Apache, the commands would be:

```
c:

cd "\program files\apache group\apache"

apache

Wait for Apache to exit, or press Ctrl+C

more <logs\error.log
```

After looking at the error.log you will probably have a good chance of working out what went wrong and be able to fix the problem and try again. If you are unable to work it out then please follow the guidelines for assistance at the top of this document or in the FAQ. Many users discover that the nature of the httpd.conf file is easier to manage and audit than page after page of configuration dialog boxes.

After starting Apache running (either in a console window or as a service) it will be listening to port 80 (unless you changed the Port, Listen or BindAddress directives in the configuration files). To connect to the server and access the default page, launch a browser and enter this URL:

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

This should respond with a welcome page, and a link to the Apache manual. If nothing happens or you get an error, look in the error.log file in the logs directory. If your host isn't connected to the net, you may have to use this URL:

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

Once your basic installation is working, you should configure it properly by editing the files in the conf directory.

Because Apache *CANNOT* share the same port with another TCP/IP application, you may need to stop or uninstall certain services first. These include (but are not limited to) other web servers, and firewall products such as BlackIce. If you can only start Apache with these services disabled, reconfigure either Apache or the other product so that they do not listen on the same

TCP/IP ports. You may find the Windows "netstat -an" command useful in finding out what ports are in use.

Configuring Apache for Windows

Apache is configured by files in the conf directory. These are the same as files used to configure the Unix version, but there are a few different directives for Apache on Windows Begin configuring the Apache server by reviewing httpd.conf and its directives. Although the files access.conf and srm.conf both exist, these are old files which are no longer used by most administrators, and you will find no directives there.

httpd.conf contains a great deal of documentation itself, followed by the default configuration directives recommended when starting with the Apache server. Begin by reading these comments to understand the configuration file and make small changes, starting Apache in a console window with each change. If you make a mistake, it will be easier to back up to configuration that last worked. You will have a better idea of which change caused the server to fail.

The main differences in Apache for Windows are:

- Because Apache for Windows is multithreaded, it does not use a separate process for each request, as Apache does with Unix. Instead there are usually only two Apache processes running: a parent process, and a child which handles the requests. Within the child each request is handled by a separate thread. So, "process"-management directives are different:
  - MaxRequestsPerChild - Like the Unix directive, this controls how many requests a process will serve before exiting. However, unlike Unix, a process serves all the requests at once, not just one, so if this is set, it is recommended that a very high number is used. The recommended default, MaxRequestsPerChild 0, does not cause the process to ever exit.
  - ThreadsPerChild - This directive is new, and tells the server how many threads it should use. This is the maximum number of connections the server can handle at once; be sure and set this number high enough for your site if you get a lot of hits. The recommended default is ThreadsPerChild 50.
- The directives that accept filenames as arguments now must use Windows filenames instead of Unix ones. However, because Apache uses Unix-style names internally, you must use forward slashes, not backslashes. Drive letters can be used; if omitted, the drive with the Apache executable will be assumed.
- Apache for Windows has 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 modules directory. To activate these or other modules, the new LoadModule directive must be used.
- LoadModule status_module modules/mod_status.so.
- Apache can also load ISAPI Extensions (*i.e.* Internet Server Applications), such as those used by Microsoft's IIS, and other Windows servers.

- 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 under windows, you may find it useful to change the name of this configuration file using the AccessFilename directive.

Case Study

Apache Software Foundation

The Apache Web Server from the Apache Software Foundation. It's a legendary product in many ways. The Internet as we know it now almost certainly wouldn't be here if it wasn't for Apache's web server.

It is currently, and has been for many years, the number one web server on the Internet according to most analytical reports. This is despite the extreme lengths that some rivals have gone to try and skew the figures. The widely read monthly survey by Netcraft shows that in July 2008, Apache was serving almost 50% of the worlds web sites. Apache has been the most popular web server on the Internet since April 1996.

The Apache web server became so popular for several reasons:

- Cost: It is Open Source and free.
- Cross Platform: Apache runs on almost every mainstream Operating System.
- Modular: Plugin modules enable wide support for building interactive and dynamic web sites using a multitude of backend services. Highly popular is the combination of the MySQL database and the PHP programming language.
- Performance: Consistently Apache has outperformed its rivals in terms of raw page impression performance, efficiency of memory and processor cycle usage and its ability to scale to support many websites within one running server instance.

Questions

1. Explain the Software foundation of Apache.
2. What do you mean by PHP programming language?

4.5 Summary

- Apache comes already bundled with a variety of linux distribution.
- The main file of its called httpd conf. This file contains directives and containers, which enable you to customize your apache installation.
- For the single site, apache is pretty much configured for logging in the default install.

4.6 Keywords

Building from Source: Building from source gives you the greatest flexibility, as it enables you to build a custom server, remove modules you do not need, and extend the server with third *party modules.*

Notes

Conditional Evaluation: Apache provides support for conditional containers. Directives enclosed in these containers will be processed only if certain conditions are met.

Containers: Directive containers, also called sections, limit the scope for which directives apply.

Error Logs**:** The error log contains messages sent from Apache for errors encountered during the course of operation. This log is very useful for troubleshooting Apache issues on the server side.

Installing a Binary: Linux/Unix binary installations are available from vendors and can also be downloaded from the Apache Software Foundation Web site.

Installing Apache on Windows: Apache 2.0 runs on most Windows platforms and offers increased performance and stability over the 1.3 versions for Windows. You can build Apache from source, but because not many Windows users have compilers, this section deals with the binary installer.

ServerRoot: The ServerRoot directive takes a single argument: a directory path pointing to the directory where the server lives.

Lab Exercise Give installation method of Apache in windows.

4.7 Self Assessment Questions

1. Apache keeps all of its configuration information in text files. The main file is called ________________.
2. Apache uses ________________ configuration files to allow directives to exist outside the main configuration filehttpd.conf.
3. Directive containers, also called sections.

   (*a*) True (*b*) False
4. ServerRoot directive takes a double argument.

   (*a*) True (*b*) False
5. The error log contains logs sent from Apache for errors encountered during the course of operation.

   (*a*) True (*b*) False

4.8 Review Questions

1. Give the steps of Apache Configuration File Structure.
2. What are Apache Log Files? Give their types also.
3. What are error logs?
4. Write down different rules for Apache directive syntax.
5. Explain containers.
6. What do you mean by ServerRoot?

Notes

Answers for Self Assessment Questions

1. httpd.conf.	2. per-directory
3. (*a*)	4. (*b*)
5. (*b*)	

4.9 Further Reading

Books *A Beginner's Guide by: Vaswani, Vikram*, By Tata MC-Graw Hill.

Online link http://www.gibmonts.com/C-plus/ch11level1sec10.htm/

Notes

Unit 5: PHP

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain versions of PHP.
- Discuss installation of PHP.
- Discuss PHP installation basics.
- Explain testing installation.
- Understand PHP string handling function.

Introduction

PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in stand alone graphical applications. PHP can be deployed on most web servers and as a stand alone interpreter, on almost every operating system and platform free of charge PHP is installed on more than 20 million websites and 1 million web servers.

PHP was originally created by Rasmus Lerdorf in 1995. The main implementation of PHP is now produced by The PHP Group and serves as the de facto standard for PHP as there is no formal specification. PHP is free software released under the PHP License; it is incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP While PHP originally stood for "Personal Home Page", it is now said to stand for "PHP: Hypertext Preprocessor", a recursive acronym.

5.1 Versions of PHP

Major version	Minor version	Release date	
1	1.0.0	1995-06-08	Officially called "Personal Home Page Tools (PHP Tools)". This is the first use of the name "PHP".
2	2.0.0	1997-11-01	Considered by its creator as the "fastest and simplest tool" for creating dynamic web pages.
3	3.0.0	1998-06-06	Development moves from one person to multiple developers. Zeev Suraski and Andi Gutmans rewrite the base for this version.
4	4.0.0	2000-05-22	Added more advanced two-stage parse/execute tag-parsing system called the Zend engine.
	4.1.0	2001-12-10	Introduced 'superglobals' ($_GET, $_POST, $_SESSION, etc.)
	4.2.0	2002-04-22	Disabled register_globals by default. Data received over the network is not inserted directly into the global namespace anymore, closing possible security holes in applications.
	4.3.0	2002-12-27	Introduced the CLI, in addition to the CGI.
	4.4.0	2005-07-11	Added man pages for phpize and php-config scripts.
	4.4.9	2008-08-07	Security enhancements and bug fixes. The last release of the PHP 4.4 series.
5	5.0.0	2004-07-13	Zend Engine II with a new object model.
	5.1.0	2005-11-24	Performance improvements with introduction of compiler variables in re-engineered PHP Engine.
	5.2.0	2006-11-02	Enabled the filter extension by default. Native JSON support.
	5.2.17	2011-01-06	Fix of critical vulnerability connected to floating point.

Contd...

Notes

	5.3.0	2009-06-30	Namespace support; Late static bindings, Jump label (limited goto), Native closures, Native PHP archives (phar), garbage collection for circular references, improved Windows support, sqlite3, mysqlnd as a replacement for libmysql as underlying library for the extensions that work with MySQL, fileinfo as a replacement for mime_magic for better MIME support, the Internationalization extension, and deprecation of ereg extension.
	5.3.1	2009-11-19	Over 100 bug fixes, some of which were security fixes as well.
	5.3.2	2010-03-04	Includes a large number of bug fixes.
	5.3.3	2010-07-22	Mainly bug and security fixes; FPM SAPI.
	5.3.4	2010-12-10	Mainly bug and security fixes; improvements to FPM SAPI.
	5.3.5	2011-01-06	Fix of critical vulnerability connected to floating point.
	5.3.6	2011-03-10	
php-trunk-dev	?.?	No date set	Removed items: 'register_globals', 'safe_mode', 'allow_call_time_pass_reference', session_register(), session_unregister() and session_is_registered() functions New features: traits, array dereferencing, closure $this support, JsonSerializable interface.

5.2 Installation of PHP

In this section, you learn how to install PEAR on your platform from a PHP distribution or through the go-pear.org web site.

5.2.1 Installing with UNIX / Linux PHP Distribution

This section describes PEAR installation and basic usage that is specific for UNIX or UNIX-like platforms, such as Linux and Darwin. The installation of the PEAR Installer itself is somewhat OS-dependent, and because most of what you need to know about installation is OS-specific, you find that here. Using the installer is more similar on different platforms, so that is described in the next section, with the occasional note about OS idiosyncrasies.

As of PHP 4.3.0, PEAR with all its basic prerequisites is installed by default when you install PHP.

If you build PHP from source, these

configure

options cause problems

for PEAR:

--disable-pear

.

make install

will neither install the PEAR installer or any

packages.

--disable-cli

. The PEAR Installer depends on a standalone version of

PHP installed.

--without-xml

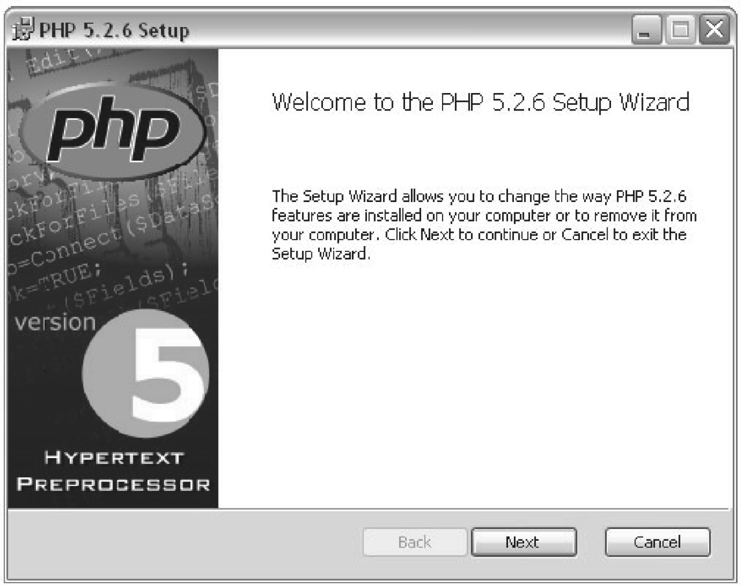
. PEAR requires the XML extension for parsing package

information files.

5.2.2 Windows

This section shows how to install PEAR on a Windows PHP installation. Start by just installing a binary distribution of PHP from http://www.php.net/downloads.php (see Figure 5.1). If you go with the defaults, your PHP install will end up in C:\PHP, which is what you will see in the forthcoming examples.

Figure 5.1: PHP Welcome Screen



PHP 5.2.6 Setup

Welcome to the PHP 5.2.6 Setup Wizard

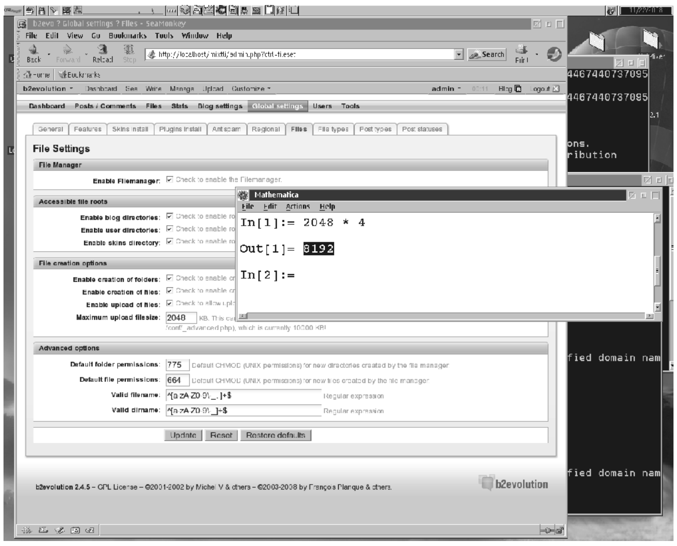
The Setup Wizard allows you to change the way PHP 5.2.6 features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Notes

5.2.3 Installing with PHP Windows Installer

When you have PHP installed, you need to make sure that your include_path PHP setting is sensible. Some versions of the Windows PHP Installer use c:\php4\pear in the default include path, but this directory (c:\php4) is different from the one created by the PHP Windows Installer. So, edit your php.ini file (in c:\winnt or c:\windows, depending on your Windows version) and change this directory to c:\php\pear (see Figure 5.2).

Figure 5.2: Example php.ini Modifications


b2evo ? Global settings ? Files - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost/mixtli/admin.php?ctrl=fileset
Search
b2evolution Dashboard See Write Manage Upload Customize
admin Blog Logout
Dashboard Posts / Comments Files Stats Blog settings Global settings Users Tools
General Features Skins install Plugins install Antispam Regional Files File types Post types Post statuses
Mathematica
File Edit Actions Help
Update Reset Restore defaults
b2evolution

File Settings

File Manager

Enable Filemanager: ☑ Check to enable the Filemanager.

Accessible file roots

Enable blog directories: ☑ Check to enable ro

Enable user directories: ☑ Check to enable ro

Enable skins directory: ☑ Check to enable ro

File creation options

Enable creation of folders: ☑ Check to enable cr

Enable creation of files: ☑ Check to enable cr

Enable upload of files: ☑ Check to allow upl

Maximum upload filesize: 2048 KB. This ca
/conf/_advanced.php), which is currently 10000 KB!

Advanced options

Default folder permissions: 775 Default CHMOD (UNIX permissions) for new directories created by the file manager

Default file permissions: 664 Default CHMOD (UNIX permissions) for new files created by the file manager

Valid filename: ^[a-zA-Z0-9\-_.]+$ Regular expression

Valid dirname: ^[a-zA-Z0-9\-_]+$ Regular expression

b2evolution 2.4.5 – GPL License – ©2001-2002 by Michel V & others – ©2003-2008 by François Planque & others.

```
In[1]:= 2048 * 4

Out[1]= 8192

In[2]:=
```

5.2.4 Go-Pear.Org

Go-Pear.Org is a web site with a single PHP script that you can download and run to install the latest stable version of the PEAR Installer and the PHP Foundation Classes (PFC). Go-Pear is cross-platform and can be run from the command line and from your web server. PHP distributions bundle a particular release of the PEAR Installer; on the other hand, go-pear gives you the newest stable PEAR releases. However, go-pear does know your directory layout, but really contorts itself to figure it out, and will try adapting your PEAR Installation to that. In this section, you learn how to use go-pear from the command line and web server, and on UNIX and Windows.

5.2.5 Prerequisites

Because go-pear is written in PHP, you need a CGI or CLI version of PHP to execute it outside the web server. By default, the CLI version is installed along with your web server PHP module. Try running php –v to see if it is available to you:

PHP 5.0.0 (cli), Copyright (c) 1997-2004 The PHP Group

Zend Engine v2.0, Copyright (c) 1998-2004 Zend Technologies

By default, the php command is installed in the /usr/local/bin directory on UNIX, or c:\php on Windows. In Windows, the CLI version of PHP may also be called php-cli ; in that case, you need to type php-cli for every example that says just php.

5.2.6 Going PEAR

Notes

If your PHP install did not include PEAR, you can use go-pear as a universal PEAR bootstrapper. All you need is a CLI or CGI version of PHP installed somewhere. You can download the go pear script and execute it, or run it all in one command, like this:

$ lynx –source http://go-pear.org | php This command simply takes the contents of http://go pear.org and sends it to PHP for execution.

If you do not have lynx available on your system, try an alternative way of executing go-pear directly:

Using GNUS wget:

```
$ wget –O- http://go-pear.org | php
```

Using fetch on FreeBSD:

```
$ fetch –o – http://go-pear.org | php
```

Using Perl LWP's GET utility:

```
$ GET http://go-pear.org | php
```

On Windows, there is no "fetch this URL" tool, but you may be able to use PHP's URL streams (make sure that url_includes is not disabled in your php.ini file):

```
C:\> php-cli –r "include('http://go-pear.org');"
```

If none of this works, open http://go-pear.org in your browser, save the contents as go-pear.php and simply run it from there:

```
C:\> php go-pear.php
```

The output will look like this:

```
Welcome to go-pear!
```

Go-pear will install the 'pear' command and all the files needed by it. This command is your tool for PEAR installation and maintenance. Go-pear also lets you download and install the PEAR packages bundled with PHP: DB, Net_Socket, Net_SMTP, Mail, XML_Parser, PHPUnit. If you wish to abort, press Control-C now, or press Enter to continue:

This greeting tells you what you are about to start. Press Enter for the first real question:

HTTP proxy (http://user:password@proxy.myhost.com:port), or Enter for none:

Go-pear checks your http_proxy environment variable and presents the value of that as the default value if http_proxy is defined. If you want to use an HTTP proxy when downloading packages, enter the address of it here, or just press Enter for "no proxy."

Now, on to the interesting part: Below is a suggested file layout for your new PEAR installation. To change individual locations, type the number in front of the directory. Type 'all' to change all of then, or simply press Enter to accept these locations:

Notes

1. Installation prefix :/usr/local
2. Binaries directory : $prefix/bin
3. PHP code directory : $prefix/share/pear
4. Documentation base directory : $php_dir/docs
5. Data base directory : $php_dir/data
6. Tests base directory : $php_dir/tests

1-6, 'all' or Enter to continue:

Each setting is internally assigned to a variable (prefix, bin_dir, php_dir, doc_dir, data_dir and test_dir, respectively). You may refer to the value of other settings by referencing these variables, as shown previously. Let's take a look at each setting:

Installation prefix: The root directory of your PEAR installation. It has no other effect than serving as a root for the next five settings, using $prefix.

Binaries directory: Where programs and PHP scripts from PEAR packages are installed. The pear executable ends up here. Remember to add this directory to your PATH.

PHP code directory: Where PHP code is installed. This directory must be in your include_path when using the packages you install.

Documentation base directory: The base directory for documentation. By default, it is $php_dir/doc, and the documentation files for each package are installed as $doc_dir/Package/file.

Database directory: Where the PEAR Installer stores data files. Data files are just a catch-all category for anything that does not fit as PHP code, documentation, and so on. As with the documentation base directory, the package name is added to the path, so the data file convert.xsl in MyPackage would be installed as $data_dir/MyPackage/convert.xsl.

Tests base directory: Where regression test scripts for the package are installed. The package name is also added to the directory. When you are satisfied with the directory layout, press Enter to proceed: The following PEAR packages are bundled with PHP: DB, Net_Socket.

Task Draw PHP Welcome screen

?Net_SMTP, Mail, XML_Parser, PHPUnit2.

Would you like to install these as well? [Y/n] :

For your convenience, go-pear requests whether you want to install the PFC packages. Just install them (press Enter):

Loading zlib: ok

Downloading package: PEAR.............ok

Downloading package: Archive_Tar......ok

Contd...

Notes

```
Downloading package: Console_Getopt..........ok
Downloading package: XML_RPC..........ok
Bootstrapping: PEAR...................(remote) ok
Bootstrapping: Archive_Tar............(remote) ok
Bootstrapping: Console_Getopt.........(remote) ok
Downloading package: DB...............ok
Downloading package: Net_Socket.......ok
Downloading package: Net_SMTP.........ok
Downloading package: Mail.............ok
Downloading package: XML_Parser.......ok
Downloading package: PHPUnit2.........ok
Extracting installer..................ok
install ok: PEAR 1.3.1
install ok: Archive_Tar 1.2
install ok: Console_Getopt 1.2
install ok: XML_RPC 1.1.0
install ok: DB 1.6.4
install ok: Net_Socket 1.0.2
install ok: Net_SMTP 1.2.6
install ok: Mail 1.1.3
install ok: XML_Parser 1.2.0
install ok: PHPUnit2 2.0.0beta2
The 'pear' command is now at your service at /usr/local/bin/pear
You have just installed PEAR!.
```

Caution PHP Welcome screen. and php.ini modifications

5.3 PHP Installation

After you have compiled or installed PHP, you can still change its behavior with the php.ini file. On Linux/Unix systems, the default location for this file is/usr/local/php/lib, or the lib subdirectory of the PHP installation location you used at configuration time. On a Windows system, this file should be in the Windows directory.

Notes

Directives in the php.ini file come in two forms: values and flags. Value directives take the form of a directive name and a value separated by an equal sign. Possible values vary from directive to directive. Flag directives take the form of a directive name and a positive or negative term separated by an equal sign. Positive terms include 1, On, Yes, and TRue. Negative terms include 0, Off, No, and False. Whitespace is ignored.

You can change your php.ini settings at any time, but after you do, you'll need to restart the server for the changes to take effect. At some point, take time to read through the php.ini file on your own to see the types of things that can be configured.

5.4 Testing Installation

Installing PHP on your development PC allows you to safely create and test a web application without affecting the data or systems on your live website. This article describes PHP installation as a module within the Windows version of Apache 2.2. Mac and Linux users will probably have it installed already.

All-in-One packages

There are some excellent all-in-one Windows distributions that contain Apache, PHP, MySQL and other applications in a single installation file, e.g. XAMPP (including a Mac version), WampServer and Web Developer. There is nothing wrong with using these packages, although manually installing Apache and PHP will help you learn more about the system and its configuration options.

The PHP Installer

Although an installer is available from php.net, I would recommend the manual installation if you already have a web server configured and running.

Manual Installation

Manual installation offers several benefits:

- backing up, reinstalling, or moving the web server can be achieved in seconds and
- you have more control over PHP and Apache configuration.

Step 1: extract the files

We will install the PHP files to C:\php, so create that folder and extract the contents of the ZIP file into it.

PHP can be installed anywhere on your system, but you will need to change the paths referenced in the following steps.

Step 2: configure php.ini

Copy C:\php\php.ini-recommended to C:\php\php.ini. There are several lines you will need to change in a text editor (use search to find the current setting).

Define the extension directory:

```
extension_dir = ”C:\php\ext”
```

Enable extensions. This will depend on the libraries you want to use, but the following extensions should be suitable for the majority of applications (remove the semi-colon comment):

1. extension=php_curl.dll
2. extension=php_gd2.dll

3. extension=php_mbstring.dll
4. extension=php_mysql.dll
5. extension=php_mysqli.dll
6. extension=php_pdo.dll
7. extension=php_pdo_mysql.dll
8. extension=php_xmlrpc.dll

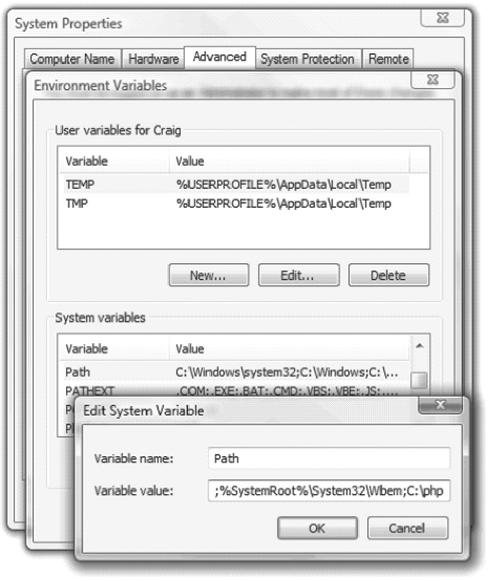
If you want to send emails using the PHP mail() function, enter the details of an SMTP server (your ISP's server should be suitable):

1. [mail function]
2. ; For Win32 only.
3. SMTP = mail.myisp.com
4. smtp_port = 25
5. ; For Win32 only.
6. sendmail_from = my@emailaddress.com

Step 3: add C:\php to the path environment variable

To ensure Windows can find PHP, you need to change the path environment variable. From the Control Panel, choose System, (then "Advanced system settings" in Vista), select the "Advanced" tab, and click the "Environment Variables" button.

Scroll down the System variables list and click on "Path" followed by the "Edit" button. Enter "**;C:\php**" to the end of the Variable value line (remember the semi-colon).



User variables for Craig

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
Path	C:\Windows\system32;C:\Windows;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

Variable name: Path

Variable value: ;%SystemRoot%\System32\Wbem;C:\php

Figure 5.3: Example Edit System Variable Value

Now OK your way out. You might need to reboot at this stage.

Notes

Step 4: configure PHP as an Apache module

Ensure Apache is not running (use "net stop Apache2.23 from the command line) and open its \conf\httpd.conf configuration file in an editor. The following lines given on next page should be changed:

Line 239, add index.php as a default file name:

1. DirectoryIndex index.php index.html

DirectoryIndex index.php index.html

At the bottom of the file, add the following lines (change the PHP file locations if necessary):

1. # PHP5 module
2. LoadModule php5_module "c:/php/php5apache2_2.dll"
3. AddType application/x-httpd-php .php
4. PHPIniDir "C:/php"

Save the configuration file and test it from the command line (Start > Run > cmd):

1. cd \Apache2\bin
2. httpd -t

Step 5: test a PHP file

Create a file named index.php in Apache's web page root (either htdocs or D:\WebPages) and add this code:

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

Ensure Apache has started successfully, open a web browser and enter the address **http://localhost/**. If all goes well, a "PHP version" page should appear showing all the configuration settings.

Case Study

Success Story of Colgate-Palmolive

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 the 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."

Contd...

Notes

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.

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 centered 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—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 talkshow-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

Contd...

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 standup 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 Year's Eve" which logged 1,255,872 views and Michelle Phan's "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.

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.

Questions

1. What kind of challenges Colgate faced during promoting their product?

2. What kind of Strategies Colgate follow during promoting their product?

5.5 Summary

- After you have compiled or installed PHP, you can still change its behavior with the php.ini file.
- PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages.
- PHP was originally created by Rasmus Lerdorf in 1995.
- Some versions of the Windows PHP Installer use c:\php4\pear in the default include path, but this directory (c:\php4) is different from the one created by the PHP Windows Installer.

- Go-pear.org is a web site with a single PHP script that you can download and run to install the latest stable version of the PEAR Installer and the PHP Foundation Classes (PFC).
- PHP is installed on more than 20 million websites and 1 million web servers in the web.

5.6 Keywords

PHP: It is a scripting language originally designed for web development to produce web pages.

Installation Prefix: It is the root directory of PEAR installation. It has no other effect than serving as a root for the next five settings, using and prefix.

Binaries directory: Where programs and PHP scripts from PEAR packages are installed. The pear executable ends up here.

Test base directory: Where regression test scripts for the package are installed.

PHP code directory: This directing exists where PHP code is installed. This directory is the include N path when using the packages you install.

Lab Exercise

1. Make the syntax table.
2. Define switch Flow.

5.7 Self Assessment Questions

1. PHP is a widely used scripting language that is especially suited for web development and can be embedded into html.

   (*a*) Open source general purpose

   (*b*) Proprietary general purpose

   (*c*) Open source special purpose

   (*d*) Proprietary special purpose

2. Which of the following variable is not a predefined variable?

   (*a*) $get (*b*) $ask

   (*c*) $request (*d*) $post

3. Installation prefix: The root directory of your PEAR installation. It has no other effect than serving as a root for the next five settings, using $prefix.

   (*a*) True (*b*) False

4. Binaries directory: Where PHP code is installed. This directory must be in your include_path when using the packages you install.

   (*a*) True (*b*) False

5. **Tests base directory:** Where regression test scripts for the package are installed. The package name is also added to the directory.

   (*a*) True (*b*) False

Notes

5.8 Review Questions

1. Explain PHP and its versions.
2. Explain installation of PHP.
3. What is PEAR? How is its installation done?
4. Name five various version of PHP with details theirin.

Answers for Self Assessment Questions

1. (*a*) 2. (*b*) 3. (*b*) 4. (*b*) 5. (*a*)

5.9 Further Reading

Books

Open source development with LAMP: Using Linux, Apache, My SQL, Perl & PHP by: James Lee, Pearson Education.

Online link

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

Unit 6: Building Blocks of PHP

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain variables in PHP and its data types.
- Discuss types of operators.
- Explain expressions.
- Discuss PHP constants.

Introduction

On the building blocks of PHP, including data types, literals, variables, and constants. Every concept is illustrated with tested code examples, screen shots showing program output, and line-by-line explanations.

The basic building blocks of the PHP language: variables and operators. You'll need to get up to speed with reading and writing PHP code and a solid grounding in the basics is very important.

6.1 Building Blocks of PHP

6.1.1 Variables

Variables in PHP are quite different from compiled languages such as C and Java. This is because their weakly typed nature, which in short means you don't need to declare variables before using them, you don't need to declare their type and, as a result, a variable can change the type of its value as much as you want. Variables in PHP are preceded with a $ sign, and similar to most modern languages, they can start with a letter (A-Za-z) or _ (underscore) and can then contain as many alphanumeric characters and underscores as you like.

Examples of legal variable names include

```
$count

$_Obj

$A123
```

Example of illegal variable names include

```
$123

$*ABC
```

As previously mentioned, you don't need to declare variables or their type before using them in PHP. The following code example uses variables:

```
$PI = 3.14;

$radius = 5;

$circumference = $PI * 2 * $radius; // Circumference = π * d
```

You can see that none of the variables are declared before they are used. Also, the fact that $PI is a floating-point number, and $radius (an integer) is not declared before they are initialized.

PHP does not support global variables like many other programming languages (except for some special pre-defined variables, which we discuss later). Variables are local to their scope, and if created in a function, they are only available for the lifetime of the function. Variables that are created in the main script (not within a function) aren't global variables; you cannot see them inside functions, but you can access them by using a special array $GLOBALS[], using the variable's name as the string offset. The previous example can be rewritten the following way:

```
$PI = 3.14;

$radius = 5;

$circumference = $GLOBALS["PI"] * 2 * $GLOBALS["radius"];

→ // Circumference = π * d
```

You might have realized that even though all this code is in the main scope (we didn't make use of functions), you are still free to use $GLOBALS[], although in this case, it gives you no advantage.

6.1.1.1 Indirect References to Variables

An extremely useful feature of PHP is that you can access variables by using indirect references, or to put it simply, you can create and access variables by name at runtime.

Consider the following example:

```
$name = ”John”;
$$name = ”Registered user”;
print $John;
```

This code results in the printing of ”Registered user.” The bold line uses an additional $ to access the variable with name specified by the value of $name (”John”) and changing its value to “Registered user”. Therefore, a variable called $John is created. You can use as many levels of indirections as you want by adding additional $ signs in front of a variable.

6.1.1.2 Managing Variables

Three language constructs are used to manage variables. They enable you to check if certain variables exist, remove variables, and check variables' truth values.

isset() isset() The Determines whether a certain variable has already been declared by PHP. It returns a boolean value true if the variable has already been set, and false otherwise, or if the variable is set to the value NULL. Consider the following script:

```
if (isset($first_name)) {
print '$first_name is set';
}
```

This code snippet checks whether the variable $first_name is defined. If $first_name is defined, isset() returns true, which will display '$first_name is set.' If it isn't, no output is generated.

isset() can also be used on array elements (discussed in a later section) and object properties. Here are examples for the relevant syntax, which you can refer to later:

Checking an array element:

```
if (isset($arr[”offset”])) {
...
}
```

Checking an object property:

```
if (isset($obj->property)) {
...
}
```

Note that in both examples, we didn't check if $arr or $obj are set (before we checked the offset or property, respectively). The isset() construct returns false automatically if they are not set.

isset() is the only one of the three language constructs that accepts an arbitrary amount of parameters. Its accurate prototype is as follows:

```
isset($var1, $var2, $var3, ...);
```

It only returns true if all the variables have been defined; otherwise, it returns false. This is useful when you want to check if the required input variables for your script have really been sent by the client, saving you a series of single isset() checks.

unset() "Undeclares" a previously set variable, and frees any memory that was used by it if no other variable references its value. A call to isset() on a variable that has been unset() returns false.

For example:

```
$name = "John Doe";
unset($name);
if (isset($name)) {
print '$name is set';
}
```

This example will not generate any output, because isset() returns false.

unset() can also be used on array elements and object properties similar to isset().

6.1.1.3 Empty

Empty() empty() may be used to check if a variable has not been declared or its value is false. This language construct is usually used to check if a form variable has not been sent or does not contain data. When checking a variable's truth value, its value is first converted to a Boolean according to the rules in the following section, and then it is checked for true/false.

For example:

```
if (empty($name)) {
print 'Error: Forgot to specify a value for $name';
}
```

This code prints an error message if $name doesn't contain a value that evaluates to true.

6.1.1.4 Superglobals

As a general rule, PHP does not support global variables (variables that can automatically be accessed from any scope). However, certain special internal variables behave like global variables similar to other languages. These variables are called superglobals and are predefined by PHP for you to use. Some examples of these superglobals are:

$_GET[]. An array that includes all the GET variables that PHP received from the client browser.

$_POST[]. An array that includes all the POST variables that PHP received from the client browser.

$_COOKIE[]. An array that includes all the cookies that PHP received from the client browser.

$_ENV[]. An array with the environment variables.

$_SERVER[]. An array with the values of the web-server variables.

"How to Write a Web Application with PHP." On a language level, it is important to know that you can access these variables anywhere in your script whether function, method, or global scope. You don't have to use the $GLOBALS[] array, which allows for accessing global variables without having to predeclare them or using the deprecated globals keyword.

6.1.2 Data Types

Eight different data types exist in PHP, five of which are scalar and each of the remaining three has its own uniqueness. The previously discussed variables can contain values of any of these data types without explicitly declaring their type. The variable "behaves" according to the data type it contains.

6.1.2.1 Integers

Integers are whole numbers and are equivalent in range as your C compiler's long value. On many common machines, such as Intel Pentiums, that means a 32-bit signed integer with a range between –2,147,483,648 to +2,147,483,647.

Integers can be written in decimal, hexadecimal (prefixed with 0x), and octal notation (prefixed with 0), and can include +/- signs.

Some examples of integers include

240000

0xABCD

007

-100

Notes As integers are signed, the right shift operator in PHP always does a signed shift.

6.1.2.2 Floating-Point Numbers

Floating-point numbers (also known as real numbers) represent real numbers and are equivalent to your platform C compiler's *double* data type. On common platforms, the data type size is 8 bytes and it has a range of approximately 2.2E–308 to 1.8E+308. Floating-point numbers include a decimal point and can include a +/- sign and an exponent value.

Notes

Examples of floating-point numbers include

3.14

+0.9e-2

-170000.5

54.6E42

6.1.2.3 Strings

Strings in PHP are a sequence of characters that are always internally nullterminated. However, unlike some other languages, such as C, PHP does not rely on the terminating null to calculate a string's length, but remembers its length internally. This allows for easy handling of binary data in PHP—for example, creating an image on-the-fly and outputting it to the browser. The maximum length of strings varies according to the platform and C compiler, but you can expect it to support at least 2GB. Don't write programs that test this limit because you're likely to first reach your memory limit.

When writing string values in your source code, you can use double quotes ("), single quotes (') or here-docs to delimit them.

Double Quotes Examples for double quotes:

"PHP: Hypertext Pre-processor"

"GET / HTTP/1.0\n"

"1234567890"

Strings can contain pretty much all characters. Some characters can't be written as is, however, and require special notation:

\n	Newline
\t	Tab.
\"	Double quote.
\\	Backslash.
\0	ASCII 0 (null).
\r	Line feed.
\$	Escape $ sign so that it is not treated as a variable but as the character $.
\{Octal #}	The character represented by the specified octal #— for example,\70 represents the letter 8.
\x{Hexadecimal#}	The character represented by the specified hexadecimal #—forexample, \0x32 represents the letter 2.

An additional feature of double-quoted strings is that certain notations of variables and expressions can be embedded directly within them. Without going into specifics, here are some examples of legal strings that embed variables. The references to variables are automatically replaced with the variables' values, and if the values aren't strings, they are converted to their corresponding string representations (for example, the integer 123 would be first converted to the string "123").

"The result is $result\n"

"The array offset $i contains $arr[$i]"

In cases, where you'd like to concatenate strings with values (such as variables and expressions) and this syntax isn't sufficient, you can use the . (dot) operator to concatenate two or more strings.

Single Quotes In addition to double quotes, single quotes may also delimit strings. However, in contrast to double quotes, single quotes do not support all the double quotes' escaping and variable substitution.

The following table includes the only two escapings supported by single quotes:

\'	Single quote.
\\	Backslash, used when wanting to represent a backslash followed by a single quote—for example, \\'.

Example: 'Hello, World'

'Today\'s the day'

6.1.2.4 Here-Docs Here-docs

Here-Docs Here-docs enable you to embed large pieces of text in your scripts, which may include lots of double quotes and single quotes, without having to constantly escape them.

The following is an example of a here-doc.

Example: <<<THE_END

PHP stands for "PHP: Hypertext Preprocessor".

The acronym "PHP" is therefore, usually referred to as a recursive acronym '→because the long form contains the acronym itself.

As this text is being written in a here-doc there is no need to escape the '→double quotes.

THE_END

The strings starts with <<<, followed by a string that you know doesn't appear in your text. It is terminated by writing that string at the beginning of a line, followed by an optional semicolon (;), and then a required newline (\n). Escaping and variable substitution in here-docs is identical to double-quoted strings except that you are not required to escape double quotes.

Accessing String Offsets Individual characters in a string can be accessed using the $str{offset} notation. You can use it to both read and write string offsets. When reading characters, this notation should be used only to access valid indices. When modifying characters, you may access offsets that don't yet exist. PHP automatically sets that offset to the said character, and if this results in a gap between the ending of the original string and the offset of the new character, the gap filled with space characters (' ').

This example creates and prints the string "Andi" (in an awkward way):

$str = "A";

$str{2} = "d";

Notes

```
$str{1} = "n";
$str = $str . "i";
print $str;
```

Tip: For many cases, PHP has string manipulation functions which use efficient algorithms. You should first look at them before you access strings directly using string offsets. They are usually prefixed with str_. For more complex needs, the regular expressions functions—most notably the pcre_ family of functions—will come in handy.

Did u know? In PHP 4, you could use [] (square brackets) to access string offsets. This support still exists in PHP 5, and you are likely to bump into it often. However, you should really use the {} notation because it differentiates string offsets from array offsets and thus, makes your code more readable.

6.1.2.5 Booleans

Booleans were introduced for the first time in PHP 4 and didn't exist in prior versions. A Boolean value can be either true or false. As previously mentioned, PHP automatically converts types when needed. Boolean is probably the type that other types are most often converted to behind the scenes. This is because, in any conditional code such as if statements, loops, and so on, types are converted to this scalar type to check if the condition is satisfied. Also, comparison operators result in a Boolean value.

Consider the following code fragment:

```
$numerator = 1;
$denominator = 5;
if ($denominator == 0) {
print "The denominator needs to be a non-zero number\n";
}
```

The result of the equal-than operator is a Boolean; in this case, it would

be false and, therefore, the if() statement would not be entered.

Now, consider the next code fragment:

```
$numerator = 1;
$denominator = 5;
if ($denominator) {
/* Perform calculation */
} else {
print "The denominator needs to be a non-zero number\n";
}
```

You can see that no comparison operator was used in this example; however, PHP automatically internally converted $denominator or, to be more accurate, the value 5 to its Boolean equivalent, true, to perform the if () statement and, therefore, enter the calculation. Although not all types have been covered yet, the following table shows truth values for their values. You can revisit this table to check for the types of Boolean value equivalents, as you learn about the remaining types.

Data Type	**False Values**	**True Values**
Integer	0	All non-zero values
Floating point		All non-zero values
Strings		All other strings
Null		Never
Array		If it contains at least one element
Object	Never	Always
Resource	Never	Always

6.1.2.6 Null

Null is a data type with only one possible value: The NULL value. It marks variables as being empty, and it's especially useful to differentiate between the empty string and null values of databases. The isset($variable) operator of PHP returns false for NULL, and true for any other data type, as long as the variable you're testing exists.

The following is an example of using NULL.

Example: $value = NULL;

6.1.2.7 Resources

Resources, a special data type, represent a PHP extension resource such as a database query, an open file, a database connection, and lots of other external types.

You will never directly touch variables of this type, but will pass them around to the relevant functions that know how to interact with the specified resource.

6.1.2.8 Arrays

An **array** in PHP is a collection of key/value pairs. This means that it maps keys (or indexes) to values. **Array indexes** can be either integers or strings whereas values can be of any type (including other arrays).

Tip: Arrays in PHP are implemented using hash tables, which means that accessing a value has an average complexity of O (1).

array() construct: Arrays can be declared using the array() language construct, which generally takes the following form (elements inside square brackets, [], are optional):

array([key =>] value, [key =>] value, ...)

The key is optional, and when it's not specified, the key is automatically assigned one more than the largest previous integer key (starting with 0). You can intermix the use with and without the

Notes

key even within the same declaration. The value itself can be of any PHP type, including an array. Arrays containing arrays give a similar result as multi-dimensional arrays in other languages.

Here are a few examples:

array(1, 2, 3) is the same as the more explicit array(0 => 1, 1 => 2, 2 ‘=> 3).

array(“name” => “John”, “age” => 28)

array(1 => “ONE”, “TWO”, “THREE”) is equivalent to array(1 => “ONE”, 2 =>

→“TWO”, 3 => “THREE”).

array() an empty array.

Here’s an example of a nested array() statement:

array(array(“name” => “John”, “age” => 28), array(“name” =>

→“Barbara”, “age” => 67))

The previous example demonstrates an array with two elements: Each one is a collection (array) of a person’s information.

Accessing Array Elements: Array elements can be accessed by using the $arr[key] notation, where key is either an integer or string expression. When using a constant string for key, make sure you don’t forget the single or double quotes, such as $arr[“key”]. This notation can be used for both reading array elements and modifying or creating new elements.

Modifying/Creating Array Elements

```
$arr1 = array(1, 2, 3);
$arr2[0] = 1;
$arr2[1] = 2;
$arr2[2] = 3;
print_r($arr1);
print_r($arr2);
```

The print_r() function has not been covered yet in this book, but when it is passed an array, it prints out the array’s contents in a readable way. You can use this function when debugging your scripts.

The previous example prints

```
Array
(
[0] => 1
Gutmans_ch02 Page 24 Thursday, September 23, 2004 2:37 PM
[1] => 2
[2] => 3
)
Array
(
[0] => 1
[1] => 2
[2] => 3
)
```

So, you can see that you can use both the array() construct and the $arr[key] notation to create arrays. Usually, array() is used to declare arrays whose elements are known at compile-time, and the $arr[key] notation is used when the elements are only computed at runtime. PHP also supports a special notation, $arr[], where the key is not specified. When creating new array offsets using this notation (fo example, using it as the l-value), the key is automatically assigned as one more than the largest previous integer key.

Therefore, the previous example can be rewritten as follows:

```
$arr1 = array(1, 2, 3);
$arr2[] = 1;
$arr2[] = 2;
$arr2[] = 3;
```

The result is the same as in the previous example.

The same holds true for arrays with string keys:

```
$arr1 = array("name" => "John", "age" => 28);
$arr2["name"] = "John";
$arr2["age"] = 28;
if ($arr1 == $arr2) {
print '$arr1 and $arr2 are the same' . "\n";
}
```

The message confirming the equality of both arrays is printed.

Reading Array Values

You can use the $arr[key] notation to read array values. The next few examples build on top of the previous example:

```
print $arr2["name"];
if ($arr2["age"] < 35) {
```

print " is quite young\n";

This example prints

Rati is Quite Young

Notes As previously mentioned, using the $arr[] syntax is not supported when reading array indexes, but only when writing them.

Accessing Nested Arrays (or Multi-Dimensional Arrays) When accessing nested arrays, you can just add as many square brackets as required to reach the relevant value. The following is an example of how you can declare nested arrays:

$arr = array(1 => array("name" => "John", "age" => 28), array("name"

→=> "Barbara", "age" => 67))

You could achieve the same result with the following statements:

$arr[1]["name"] = "John";

$arr[1]["age"] = 28;

$arr[2]["name"] = "Barbara";

$arr[2]["age"] = 67;

Reading a nested array value is trivial using the same notation. For example, if you want to print John's age, the following statement does the trick:

print $arr[1]["age"];

Traversing Arrays Using foreach There are a few different ways of iterating over an array. The most elegant way is the foreach() loop construct.

The general syntax of this loop is

foreach($array as [$key =>] [&] $value)

...

$key is optional, and when specified, it contains the currently iterated value's key, which can be either an integer or a string value, depending on the key's type.

Specifying and for the value is also optional, and it has to be done if you are planning to modify $value and want it to propagate to $array. In most cases, you won't want to modify the $value when iterating over an array and will, therefore, not need to specify it.

Here's a short example of the foreach() loop:

$players = array("John", "Barbara", "Bill", "Nancy");

print "The players are:\n";

foreach ($players as $key => $value) {

print "#$key = $value\n";

```
}
```

The output of this example is

```
The players are:
#0 = John
#1 = Barbara
#2 = Bill
#3 = Nancy
```

Here's a more complicated example that iterates over an array of people and marks which person is considered old and which one is considered young:

```
$people = array(1 => array("name" => "John", "age" => 28),
→array("name" => "Barbara", "age" => 67));
foreach ($people as and $person) {
if ($person["age"] >= 35) {
$person["age group"] = "Old";
} else {
$person["age group"] = "Young";
}
}
print_r($people);
```

Again, this code makes use of the print_r() function.

The output of the previous code is the following:

```
Array
(
[1] => Array
(
[name] => John
[age] => 28
[age group] => Young
)
[2] => Array
(
[name] => Barbara
[age] => 67
[age group] => Old
)
)
```

You can see that both the John and Barbara arrays inside the $people array were added an additional value with their respective age group.

Traversing Arrays Using list() and each() Although foreach() is the nicer way of iterating over an array, an additional way of traversing an array is by using a combination of the list() construct and the each() function:

```
$players = array("John", "Barbara", "Bill", "Nancy");
reset($players);
while (list($key, $val) = each($players)) {
print "#$key = $val\n";
}
```

The output of this example is

```
#0 = John
#1 = Barbara
#2 = Bill
#3 = Nancy
```

reset() Iteration in PHP is done by using an internal array pointer that keeps record of the current position of the traversal. Unlike with foreach(), when you want to use each() to iterate over an array, you must reset() the array before you start to iterate over it. In general, it is best for you to always use foreach() and not deal with this subtle nuisance of each() traversal.

each() The each() function returns the current key/value pair and advances the internal pointer to the next element. When it reaches the end of of the array, it returns a booloean value of false. The key/value pair is returned as an array with four elements: the elements 0 and "key", which have the value of the key, and elements 1 and "value", which have the value of the value. The reason for duplication is that, if you're accessing these elements individually, you'll probably want to use the names such as:

$elem["key"] and $elem["value"]:

```
$ages = array("John" => 28, "Barbara" => 67);
reset($ages);
$person = each($ages);
print $person["key"];
print " is of age ";
print $person["value"];
```

This prints

```
John is of age 28
```

When we explain how the list() construct works, you will understand why offsets 0 and 1 also exist.

list() The list() construct is a way of assigning multiple array offsets to multiple variables in one statement:

```
list($var1, $var2, ...) = $array;
```

The first variable in the list is assigned the array value at offset 0, the second is assigned offset 1, and so on. Therefore, the list() construct translates into the following series of PHP statements:

```
$var1 = $array[0];
$var2 = $array[1];
...
```

As previously mentioned, the indexes 0 and 1 returned by each() are used by the list() construct. You can probably already guess how the combination of list() and each() work. Consider the highlighted line from the previous $players traversal example:

```
$players = array("John", "Barbara", "Bill", "Nancy");
reset($players);
while (list($key, $val) = each($players)) {
print "#$key = $val\n";
}
```

What happens in the boldfaced line is that during every loop iteration, each() returns the current position's key/value pair array, which, when examined with print_r(), is the following array:

```
Array
(
[1] => John
[value] => John
[0] => 0
[key] => 0
)
```

Then, the list() construct assigns the array's offset 0 to $key and offset 1

to $val.

Notes

Additional Methods for Traversing Arrays

You can use other functions to iterate over arrays including current() and next(). You shouldn't use them because they are confusing and are legacy functions. In addition, some standard functions allow all sorts of elegant ways of dealing with arrays such as array_walk(), which is covered in a later unit.

6.1.3 Operators

PHP contains three types of operators: unary operators, binary operators, and one ternary operator.

Binary operators are used on two operands:

2 + 3

14 * 3.1415

$i – 1

These examples are also simple examples of expressions. 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, according to the following rules (unless stated differently, such as in the concatenation operator).

Type of one of the Operands	Type of the other Operand	Conversion Performed
Integer	Floating point	The integer operand is converted to a floating point number.
Integer	String	The string is converted to a number. If the converted string's type is real, the integer operand is converted to a real as well.
Real	String	The string is converted to a real.

Booleans, nulls, and resources behave like integers, and they convert in the following manner:

- Boolean: False = 0, True = 1
- Null = 0
- Resource = The resource's # (id)

6.1.3.1 Binary Operators

Numeric Operators All the binary operators (except for the concatenation operator) work only on numeric operands. If one or both of the operands are strings, Booleans, nulls, or resources, they are automatically converted to their numeric equivalents before the calculation is performed.

Operator	Name	Value
+	Addition	The sum of the two operands.
–	Subtraction	The difference between the two operands.
*	Multiplication	The product of the two operands.

Contd...

/	Division	The quotient of the two operands.
%	Modulus	Both operands are converted to integers. The result is the remainder of the division of the first operand by the second operand.

Concatenation Operator (.) The **concatenation operator** concatenates two strings. This operator works only on strings; thus, any non-string operand is first converted to one.

The following example would print out "The year is 2000":

```
$year = 2000;

print "The year is " . $year;
```

The integer $year is internally converted to the string "2000" before it is concatenated with the string's prefix, "The year is".

6.1.3.2 Assignment Operators

Assignment operators enable you to write a value to a variable. The first operand (the one on the left of the assignment operator or l value) must be a variable. The value of an assignment is the final value assigned to the variable; for example, the expression $var = 5 has the value 5 (and assigns 5 to $var).

In addition to the regular assignment operator =, several other assignment operators are composites of an operator followed by an equal sign. These composite operators apply the operator taking the variable on the left as the first operand and the value on the right (the r value) as the second operand, and assign the result of the operation to the variable on the left.

For example:

```
$counter += 2; // This is identical to $counter = $counter + 2;

$offset *= $counter;// This is identical to $offset = $offset *

→ $counter;
```

The following list show the valid composite assignment operators:

+=, -=, *=, /=, %=, ^=, .=, &=, |=, <<=, >>=

By-Reference Assignment Operator PHP enables you to create variables as aliases for other variables. You can achieve this by using the by-reference assignment operator =&. After a variable aliases another variable, changes to either one of them affects the other.

For example:

```
$name = "Judy";

$name_alias =& $name;

$name_alias = "Jonathan";

print $name;
```

Notes

The result of this example is

Jonathan

When returning a variable by-reference from a function (covered later in this book), you also need to use the assign by-reference operator to assign the returned variable to a variable:

$retval =& func_that_returns_by_reference();

6.1.3.3 Comparison Operators

Comparison operators enable you to determine the relationship between two operands. When both operands are strings, the comparison is performed lexicographically.

The comparison results in a Boolean value. For the following comparison operators, automatic type conversions are performed, if necessary

Operator	Name	Value
==	Equal to	Checks for equality between two arguments performing type conversion when necessary: 1 == "1" results in true 1 == 1 results in true
!=	Not equal to	Inverse of ==.
>	Greater than	Checks if first operand is greater than second
<	Smaller than	Checks if first operand is smaller than second
>=	Greater than or equal to	Checks if first operand is greater or equal to second
<=	Smaller than or equal to	Checks if first operand is smaller or equal to second

For the following two operators, automatic type conversions are *not* performed and, therefore, both the types and the values are compared.

Operator	Name	Value
===	Identical to	Same as == but the types of the operands have tomatch. No automatic type conversions are performed:1 === "1" results in false.1 === 1 results in true.
!==	Not identical to	The inverse of ===.

6.1.3.4 Logical Operators

Logical operators first convert their operands to boolean values and then perform the respective comparison.

Notes

Operator	Name	Value
&&, and	Logical AND	The result of the logical AND operation between the two operands
||, or	Logical OR	The result of the logical OR operation between the two operands
xor	Logical XOR	The result of the logical XOR operation between the two operands

Short-Circuit Evaluation: When evaluating the logical and/or operators, you can often know the result without having to evaluate both operands. For example, when PHP evaluates 0 && 1, it can tell the result will be false by looking only at the left operand, and it won't continue to evaluate the right one. This might not seem useful right now, but later on, we'll see how we can use it to execute an operation only if a certain condition is met.

6.1.3.5 Bitwise Operators

Bitwise operators: Perform an operation on the bitwise representation of their arguments. Unless the arguments are strings, they are converted to their corresponding integer representation, and the operation is then performed. In case both arguments are strings, the operation is performed between corresponding character offsets of the two strings (each character is treated as an integer).

Operator	Name	Value
&	Bitwise AND	Unless both operands are strings, the integer value of the bitwise AND operation between the two operands. If both operands are strings, a string in which each character is the result of a bitwise AND operation between the two corresponding characters in the operands. In case the two operand strings are different lengths, the result string is truncated to the length of the *shorter* operand.
|	Bitwise OR	Unless both operands are strings, the integer value of the bitwise OR operation between the two operands. If both operands are strings, a string in which each bitwise character is the result of a bitwise OR operation between the two corresponding characters in

Contd...

Notes

		the operands. In case the two operand strings are of different lengths, the result string has the length of the *longer* operand; the missing characters in the shorter operand are assumed to be zeroes.
^	Bitwise XOR(exclusive or)	Unless both operands are strings, the integer value of the bitwise XOR operation between the two operands. If both operands are strings, astring in which each character is the result of a bitwise XOR operation between the two corresponding characters in the operands. In case the two operand strings are of different lengths, the result string is truncated to the length of the *shorter* operand.

6.1.3.6 Unary Operators

Unary operators act on one operand.

6.1.3.7 Negation Operators

Negation operators appear before their operand—for example, !$var (! is the operator, $var is the operand).

Operator	**Name**	**Value**
!	Logical Negation	True if the operand evaluates to false. False if the operand evaluates to true.
~	Bitwise Negation	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.

6.1.3.8 Increment/Decrement Operators

Increment/decrement operators are unique in the sense that they operate only on variables and not on any value. The reason for this is that in addition to calculating the result value, the value of the variable itself changes as well.

Operator	Name	Effect on $var	Value of the Expression
$var++	Post-increment	$var is incremented by 1.	The previous value of $var.
++$var	Pre-increment	$var is incremented by 1.	The new value of $var (incremented by 1).
$var––	Post-decrement	$var is decremented by 1.	The previous value of $var.
––$var	Pre-decrement	$var is decremented by 1.	The new value of $var (decremented by 1).

As you can see from the previous table, there's a difference in the value of post and pre increment. However, in both cases, $var is incremented by 1. The only difference is in the value to which the increment expression evaluates.

Example: 1:

```
$num1 = 5;
$num2 = $num1++;// post-increment, $num2 is assigned $num1's original value
print $num1; // this will print the value of $num1, which is now 6
print $num2; // this will print the value of $num2, which is the
→original value of $num1, thus, 5
```

Example: 2:

```
$num1 = 5;
$num2 = ++$num1;// pre-increment, $num2 is assigned $num1's
→incremented value
print $num1; // this will print the value of $num1, which is now 6
print $num2; // this will print the value of $num2, which is the
→same as the value of $num1, thus, 6
```

The same rules apply to pre- and post-decrement.

Incrementing Strings Strings (when not numeric) are incremented in a similar way to Perl. If the last letter is alphanumeric, it is incremented by 1. If it was 'z', 'Z', or '9', it is incremented to 'a', 'A', or '0' respectively, and the next alphanumeric is also incremented in the same way. If there is no next alphanumeric, one is added to the beginning of the string as 'a', 'A', and '1,' respectively. If this gives you a headache, just try and play around with it.

Notes

You'll get the hang of it pretty quickly.

Did u know? Non-numeric strings cannot be decremented.

6.1.3.9 The Cast Operators

PHP provides a C-like way to force a type conversion of a value by using the **cast operators**. The operand appears on the right side of the cast operator, and its result is the converted type according to the following table.

Operator	Changes Type To
(int), (integer)	Integer
(float), (real), (double)	Floating point String
(string)	String
(bool), (boolean)	Boolean
(array)	Array
(object)	Object

The casting operators change the type of a value and not the type of a variable. For example:

```
$str = "5";

$num = (int) $str;
```

This results in $num being assigned the integer value of $str (5), but $str remains of type string.

6.1.3.10 The Silence Operator

The operator @ silences error messages during the evaluation process of an expression. It is discussed in more detail in unit 7.

6.1.3.11 The One and Only Ternary Operator

One of the most elegant operators is the ?: (question mark) operator. Its format

Is truth_expr ? expr1 : expr2

The operator evaluates truth_expr and checks whether it is true. If it is, the value of the expression evaluates to the value of expr1 (expr2 is not evaluated). If it is false, the value of the expression evaluates to the value of expr2

(expr1 is not evaluated).

For example, the following code snippet checks whether $a is set (using isset()) and displays a message accordingly:

```
$a = 99;

$message = isset($a) ? '$a is set' : '$a is not set';

print $message;
```

This example prints the following:

```
$a is set
```

6.1.4 Expressions

Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

The most basic forms of expressions are constants and variables. When you type "$a = 5", you're assigning '5' into $a. '5', obviously, has the value 5, or in other words '5' is an expression with the value of 5 (in this case, '5' is an integer constant).

After this assignment, you'd expect $a's value to be 5 as well, so if you wrote $b = $a, you'd expect it to behave just as if you wrote $b = 5. In other words, $a is an expression with the value of 5 as well. If everything works right, this is exactly what will happen.

Slightly more complex examples for expressions are functions. For instance, consider the following function:

```
< ?php

function foo ()

{

   return 5;

}

? >
```

Assuming you're familiar with the concept of functions (if you're not, take a look at the unit about functions), you'd assume that typing $c = foo() is essentially just like writing $c = 5, and you're right. Functions are expressions with the value of their return value. Since foo() returns 5, the value of the expression 'foo()' is 5. Usually functions don't just return a static value but compute something.

Of course, values in PHP don't have to be integers, and very often they aren't. PHP supports four scalar value types: integer values, floating point values (float), string values and boolean values (scalar values are values that you can't 'break' into smaller pieces, unlike arrays, for instance). PHP also supports two composite (non-scalar) types: arrays and objects. Each of these value types can be assigned into variables or returned from functions.

PHP takes expressions much further, in the same way many other languages do. PHP is an expression-oriented language, in the sense that almost everything is an expression. Consider the example we've already dealt with, '$a = 5'. It's easy to see that there are two values involved here, the value of the integer constant '5', and the value of $a which is being updated to 5 as well. But the truth is that there's one additional value involved here, and that's the value of the assignment itself. The assignment itself evaluates to the assigned value, in this case 5. In practice, it means that '$a = 5', regardless of what it does, is an expression with the value 5. Thus, writing something like '$b = ($a = 5)' is like writing '$a = 5; $b = 5;' (a semicolon marks the end of a statement). Since assignments are parsed in a right to left order, you can also write '$b = $a = 5'.

Another good example of expression orientation is pre- and post-increment and decrement. Users of PHP and many other languages may be familiar with the notation of variable++ and

Notes

variable—These are increment and decrement operators. In PHP/FI 2, the statement '$a++' has no value (is not an expression), and thus you can't assign it or use it in any way. PHP enhances the increment/decrement capabilities by making these expressions as well, like in C. In PHP, like in C, there are two types of increment-pre-increment and post-increment. Both pre-increment and post-increment essentially increment—the variable, and the effect on the variable is identical. The difference is with the value of the increment expression. Pre-increment, which is written '++$variable', evaluates to the incremented value (PHP increments the variable before reading its value, thus the name 'pre-increment'). Post-increment, which is written '$variable++' evaluates to the original value of $variable, before it was incremented (PHP increments the variable after reading its value, thus the name 'post-increment').

A very common type of expressions are comparison expressions. These expressions evaluate to either FALSE or TRUE. PHP supports > (bigger than), >= (bigger than or equal to), == (equal), != (not equal), < (smaller than) and <= (smaller than or equal to). The language also supports a set of strict equivalence operators: === (equal to and same type) and !== (not equal to or not same type). These expressions are most commonly used inside conditional execution, such as if statements.

The last example of expressions we'll deal with here is combined operator-assignment expressions. You already know that if you want to increment $a by 1, you can simply write '$a++' or '++$a'. But what if you want to add more than one to it, for instance 3? You could write '$a++' multiple times, but this is obviously not a very efficient or comfortable way. A much more common practice is to write '$a = $a + 3'. '$a + 3' evaluates to the value of $a plus 3, and is assigned back into $a, which results in incrementing $a by 3. In PHP, as in several other languages like C, you can write this in a shorter way, which with time would become clearer and quicker to understand as well. Adding 3 to the current value of $a can be written '$a += 3'. This means exactly "take the value of $a, add 3 to it, and assign it back into $a". In addition to being shorter and clearer, this also results in faster execution. The value of '$a += 3', like the value of a regular assignment, is the assigned value. Notice that it is NOT 3, but the combined value of $a plus 3 (this is the value that's assigned into $a). Any two-place operator can be used in this operator-assignment mode, for example '$a -= 5' (subtract 5 from the value of $a), '$b *= 7' (multiply the value of $b by 7), etc.

There is one more expression that may seem odd if you haven't seen it in other languages, the ternary conditional operator:

```
<?php

$first ? $second : $third

?>
```

If the value of the first subexpression is TRUE (non-zero), then the second subexpression is evaluated, and that is the result of the conditional expression. Otherwise, the third subexpression is evaluated and that is the value.

The following example should help you understand pre- and post-increment and expressions in general a bit better:

```
<?php

function double($i)
```

```
{
   return $i*2;
}
$b = $a = 5;   /* assign the value five into the variable $a and $b */
$c = $a++;     /* post-increment, assign original value of $a
               (5) to $c */
$e = $d = ++$b; /* pre-increment, assign the incremented value of
               $b (6) to $d and $e */
               /* at this point, both $d and $e are equal to 6 */
$f = double($d++); /* assign twice the value of $d before
               the increment, 2*6 = 12 to $f */
$g = double(++$e); /* assign twice the value of $e after
               the increment, 2*7 = 14 to $g */
$h = $g += 10; /* first, $g is incremented by 10 and ends with the
               value of 24. the value of the assignment (24) is
               then assigned into $h, and $h ends with the value
               of 24 as well. */
?>
```

Some expressions can be considered as statements. In this case, a statement has the form of 'expr' ';' that is, an expression followed by a semicolon. In '$b=$a=5;', $a=5 is a valid expression, but it's not a statement by itself. '$b=$a=5;' however is a valid statement.

One last thing worth mentioning is the truth value of expressions. In many events, mainly in conditional execution and loops, you're not interested in the specific value of the expression, but only care about whether it means TRUE or FALSE. The constants TRUE and FALSE (case-insensitive) are the two possible boolean values. When necessary, an expression is automatically converted to boolean. See the section about type-casting for details about how.

PHP provides a full and powerful implementation of expressions, and documenting it entirely goes beyond the scope of this manual. The above examples should give you a good idea about what expressions are and how you can construct useful expressions. Throughout the rest of this manual we'll write expr to indicate any valid PHP expression.

6.1.5 PHP Constants

Constants and variables is that constant value can not be changed in the process of running program. It can be mathematic constants, passwords, paths to files, etc. By using a constant you "lock in" the value which prevents you from accidentally changing it. If you want to run a program several times using a different value each time, you do not need to search throughout the entire program and change the value at each instance. You only need to change it at the beginning of the program where you set the initial value for the constant.

Have a look at the example where we use the define function to set the initial value of a constant:

```
php

// first we define a constant PASSWORD

define("PASSWORD","admin");

echo (PASSWORD); // will display value of PASSWORD constant,  i.e. admin

echo constant("PASSWORD");  // will also display admin

echo "PASSWORD";          // will display PASSWORD

?>
```

PHP also provides a number of built-in constants for you. "__FILE__", for example, returns the name of the file currently being read by the interpreter. "__LINE__" returns the line number of the file. These constants are useful for generating error messages. You can also find out which version of PHP is interpreting the script using the "PHP_VERSION" constant.

In PHP, you can define names, called **constants**, for simple values. As the name implies, you cannot change these constants once they represent a certain value. The names for constants have the same rules as PHP variables except that they don't have the leading dollar sign. It is common practice in many programming languages—including PHP—to use uppercase letters for constant names, although you don't have to. If you wish, which we do not recommend, you may define your constants as case-insensitive, thus not requiring code to use the correct casing when referring to your constants.

Notes Only use case-sensitive constants both to be consistent with accepted coding standards and because it is unclear if case-insensitive constants will continued to be supported in future versions of PHP. Unlike variables, constants, once defined, are globally accessible. You don't have to (and can't) redeclare them in each new function and PHP file. To define a constant, use the following function:

Define ("CONSTANT_NAME", value [, case_sensitivity])

Where:

"CONSTANT_NAME" is a string.

value is any valid PHP expression excluding arrays and objects.

case_sensitivity is a Boolean (true/false) and is optional. The default is true.

An example for a built-in constant is the Boolean value true, which is registered as case-insensitive.

Here's a simple example for defining and using a constant:

define("MY_OK", 0);

define("MY_ERROR", 1);

...

if ($error_code == MY_ERROR) {

print("There was an error\n");

6.1.6 Switch Flow

Statement.	**Statement List**
switch (*expr*){	switch (*expr*):
	case *expr*:
	case *expr*:
	statement list
statement list	case *expr*:
	statement list
	default:
	statement list
	endswitch;
case *expr*:	
statement list	
...	
default:	
statement list	

You can use the switch construct to elegantly replace certain lengthy if/elseif constructs. It is given an expression and compares it to all possible case expressions listed in its body. When there's a successful match, the following code is executed, ignoring any further case lines (execution does not stop when the next case is reached). The match is done internally using the regular equality operator (==), not the identical operator (===). You can use the break statement to end execution and skip to the code following the switch construct:

}

Usually, break statements appear at the end of a case statement list, although it is not mandatory. If no case expression is met and the switch construct contains default, the default statement list is executed. Note that the default case must appear last in the list of cases or not appear at all:

```
switch ($answer) {

case 'y':

case 'Y':

print "The answer was yes\n";
```

Notes

```
break;
case 'n':
case 'N':
print "The answer was no\n";
break;
default:
print "Error: $answer is not a valid answer\n";
break;
}
```

Did u know? It is important to include a break statement at the end of any code that will be executed as part of a case statement. Without a break statement, the program flow will continue to the next case statement and ultimately to the default statement. In most cases, this will result in unexpected behavior, likely incorrect!

6.1.7 Loop Control Structures

Loop control structures are used for repeating certain tasks in your program, such as iterating over a database query result set.

6.1.7.1 While Loops

Statement	**Statement List**
while (*expr*) statement	while (*expr*) :statement list endwhile;

while **loops** are the simplest kind of loops. In the beginning of each iteration, the while's truth expression is evaluated. If it evaluates to true, the loop keeps on running and the statements inside it are executed. If it evaluates to false, the loop ends and the statement(s) inside the loop is skipped. For example, here's one possible implementation of factorial, using a while loop (assuming $n contains the number for which we want to calculate the factorial):

```
$result = 1;
while ($n > 0) {
$result *= $n—;
}
print "The result is $result";
```

Loop Control: break and continue

break;

break *expr;*

continue;

continue *expr;*

Sometimes, you want to terminate the execution of a loop in the middle of an iteration. For this purpose, PHP provides the break statement. If break appears alone, as in break; the innermost loop is stopped. break accepts an optional argument of the amount of nesting levels to break out of,

break *n*;

which will break from the *n* innermost loops (break 1; is identical to break;). *n* can be any valid expression. In other cases, you may want to stop the execution of a specific loop iteration and begin executing the next one. Complimentary to break, continue provides this functionality. continue alone stops the execution of the innermost loop iteration and continues executing the next iteration of that loop. Continue *n* can be used to stop execution of the *n* innermost loop iterations. PHP goes on executing the next iteration of the outermost loop. As the switch statement also supports break, it is counted as a loop when you want to break out of a series of loops with break *n*.

6.1.7.2 do...while Loops

```
do

statement

while (expr);
```

The do...while loop is similar to the previous while loop, except that the truth expression is checked at the end of each iteration instead of at the beginning. This means that the loop always runs at least once. do...while loops are often used as an elegant solution for easily breaking out of a code block if a certain condition is met. Consider the following example:

```
do {

statement list

if ($error) {

break;

}

} while (false);
```

Because do...while loops always iterate at least one time, the statements inside the loop are executed once, and only once. The truth expression is always false. However, inside the loop body, you can use the break statement to stop the execution of the statements at any point, which is convenient. Of course, do...while loops are also often used for regular iterating purposes.

6.1.7.3 For Loops

Statement	**Statement List**

for (expr; expr; expr) statement	for (expr, expr, …; expr, expr, …;
	expr, expr, …):
	statement list
	endfor;

Notes

PHP provides C-style for loops. The for loop accepts three arguments: for (start_expressions; truth_expressions; increment_expressions). Most commonly, for loops are used with only one expression for each of the start, truth, and increment expressions, which would make the previous syntax table look slightly more familiar.

Statement	Statement List
for (expr; expr; expr) statement	for (expr; expr; expr): statement list endfor;

The start expression is evaluated only once when the loop is reached. Usually it is used to initialize the loop control variable. The truth expression I evaluated in the beginning of every loop iteration. If true, the statements inside the loop will be executed; if false, the loop ends. The increment expression is evaluated at the end of every iteration before the truth expression is evaluated. Usually, it is used to increment the loop control variable, but it can be used for any other purpose as well. Both break and continue behave the same way as they do with while loops. continue causes evaluation of the increment expression before it re-evaluates the truth expression.

Example:
```
for ($i = 0; $i < 10; $i++) {
print "The square of $i is " . $i*$i . "\n";
}
```
The result of running this code is
```
The square of 0 is 0
The square of 1 is 1
...
The square of 9 is 81
```

Like in C, it is possible to supply more than one expression for each of the three arguments by using commas to delimit them. The value of each argument is the value of the rightmost expression. Alternatively, it is also possible not to supply an expression with one or more of the arguments. The value of such an empty argument will be true. For example, the following is an infinite loop:

```
for (; ;) {
print "I'm infinite\n";
}
```

Did u know? Infinite loops are, as the name suggests, loops that run without bounds. If your loop is running infinitely, your script is running for an infinite amount of time. This is very stressful on your Web server, and renders the Web page unusable.

Task Make the syntax table.

6.1.8 Browser Output

The PHP output buffering functions provide a handy way of intercepting the contents of the buffer before it is sent to the browser. The output is whatever is sent to the browser whenever you print something off. PHP allows you to capture this output in a buffer before it is sent to the browser.

The Output Control functions allow you to control when output is sent from the script. This can be useful in several different situations, especially if you need to send headers to the browser after your script has began outputting data. The Output Control functions do not affect headers sent using **header()** or **setcookie()**, only functions such as **echo()** and data between blocks of PHP code.

Example #1 Output Control example

```
<?php

ob_start();

echo "Hello\n";

setcookie("cookiename", "cookiedata");

ob_end_flush();

?>
```

In the above example, the output from **echo()** would be stored in the output buffer until **ob_ end_flush()** was called. In the mean time, the call to **setcookie()** successfully stored a cookie without causing an error. (You can not normally send headers to the browser after data has already been sent.)

Browser Output Control Functions

- **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_handler**s — 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

6.2 Summary

- Variables overall in PHP are proceeded with a $ sign and similar to most modern languages.
- Eight different data types exist in PHP, five of which are scalar and each of the remaining three has its own uniqueness.
- Strings in PHP are a sequence of characters that are always internally nullterminated.
- An array in PHP is a collection of key/value pairs.
- PHP contains three types of operators: unary operators, binary operators, and one binary operator.

6.3 Keywords

Comparison Operators: Comparison operators enable you to determine the relationship between two operands. When both operands are strings, the comparison is performed lexicographically.

Comparison Expressions: Comparison expressions.evaluate to either **FALSE** or **TRUE**.

Concatenation Operator: The **concatenation operator** concatenates two strings. This operator works only on strings; thus, any non-string operand is first converted to one.

Debugger: The Code::Blocks debugger has full breakpoint support. It also allows the user to debug their program by having access and using the local function symbol.

Expressions: Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

Switch Statement: The switch statement is an alternative way of changing flow, based on the evaluation of an expression.

Type Casting: Type casting is the name of the process in which a specific data type is transformed into another (e.g., switching from a string data type to an integer data type).

Type Juggling: It means a variable is not restricted to just one data type. This means that you're allowed to switch any variable between all the available data types without having to do anything.

Lab Exercise

1. Make the syntax table.
2. Define switch flow.

6.4 Self Assessment Questions

1. You can define a constant by using the define() function. Once a constant is defined
   (*a*) It can never be changed or undefined.
   (*b*) It can never be changed but can be undefined.

(*c*) It can never be changed but can not be undefined.

(*d*) It can never be changed and can be undefined.

2. **Integers** and ___________ are equivalent in range as your C compiler's long value.

   (*a*) Numbers (*b*) Whole numbers

   (*c*) Binary Numbers (*d*) All of the above.

6.5 Review Questions

1. What is the advantage of using System.Text.StringBuilder over System.String?
2. What are variables and its various types?
3. What are integers?
4. What are floating-point numbers?
5. What are strings?
6. What are Booleans?
7. Define Data type.
8. Define Binary operators and its type.
9. What are Logical operators?
10. What are Bitwise operators?
11. What are Negation operators?
12. What are Expressions?
13. Define Constants.
14. What are Loops?

Answers for Self Assessment Questions

1. (*a*) 2. (*d*)

6.6 Further Reading

Books

Open source development with LAMP: Using Linux, Apache, My SQL, Perl & PHP, by James Lee, Pearson Education.

Online link

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

Unit 7: Functions

CONTENTS

Objectives

After studying this unit, you will be able to:

- Discuss user defined functions.
- Explain function scope.
- Understand returning values by value.
- Declaring function parameters.
- Explain default parameters.
- Discuss static variables.
- Explain arrays.
- Discuss objects.

Introduction

A function in PHP can be built-in or user-defined; however, they are both called the same way. The general form of a function call is func(arg1,arg2,…). The number of arguments varies from one function to another. Each argument be any valid expression, include other function calls.

Here is a simple example of a predefined function:

Example: $length = strlen(“Peter”);

strlen is a standard PHP function that returns the length of a string. Therefore, $length is assigned the length of the string “Peter”: four.

Here’s an example of a function call being used as a function argument:

Example: $length = strlen(strlen(“Peter”));

You probably already guessed the result of this example. First, the innerstrlen(“Peter”) is executed, which results in the integer 4. So, the code simpli-fies to

$length = strlen(4);

Strlen ( ) expects a string and, therefore, (due to PHP’s magical auto conversion between types) converts the integer 4 to the string “4”, and thus, the resulting value of $length is 1, the length of “4”.

7.1 User-Defined Functions

The general way of defining a function is

```
function function_name (arg1, arg2, arg3, …)
          statement list
          }
```

To return a value from a function, you need to make a call to return expr inside your function. This stops execution of the function and returns expr as the function’s value.

Notes

The following example function accepts one argument, $x, and returns its square:

Example:
```
function square ($x)
{
        return $x*$x;
    }
```

After defining this function, it can be used as an expression wherever you desire.

For example:

print 'The square of 5 is ' . square(5);

7.2 Function Scope

Every function has its own set of variables. Any variables used outside the function's definition are not accessible from within the function by default. When a function starts, its function parameters are defined. When you use new variables inside a function, they are defined within the function only and don't hang around after the function call ends. In the following example, the Variable $var is not changed by the function call:

Example:
```
            function func ( )
          {
      svar = 2;
  }
      $var = 1;
            func( );
      print $var;
```

When the function func is called, the variable $var, which is assigned 2, is only in the scope of the function and thus does not change $var outside the function. The code snippet prints out 1.

Now hat if you actually do want to access and/or change $var on the outside? As mentioned in the "Variables" section, you can use the built-in $ GLOBALS[] array to access variables in the global scope of the script

Rewrite the previous script the following way:

```
  Function  func ()
  {
   $ GLOBALS["var"] = 2;
}
   $var = 1;
   func ( );
   print $var;
```

It prints the value 2.

A global keyword also enables you to declare what global variables you want to access, causing them to be imported into the function's scope. However, using this keyword is not recommended for various reasons, such as misbehaving with assigning values by reference, not supporting unset ( ), and soon.

Here's a short description of it—but please, don't use it!

The syntax is

global $var1, $var2, ...;

Adding a global line for the previous example results in the following:

Example:

```
function func()
{
        global $var;
         $var = 2;
 }
$var = 1;
func( );
print $var;
```

This way of writing the example also prints the number 2.

7.3 Returning Values By Value

You can tell from the previous example that the return statement is used to return values from functions. The return statement returns values **by value**, which means that a copy of the value is created and is returned to the caller of the function.

Example:

```
function get_global_variable_value($name)
{
      return $GLOBALS[$name];
}
 $num = 10;
$value = get_global_variable_value("num");
print $value;
```

This code prints the number 10. However, making changes to $value before the print statement only affects $value and not the global variable $num. This is because its value was returned by the get_global_variable_value ( ) by value and not by reference.

7.4 Returning Values By Reference

PHP also allows you to return variables **by reference**. This means that you're not returning a copy to the variable, but you're returning the address of your variable instead, which enables

Notes

you to change it from the calling scope. To return a variable by-reference, you need to define the function as such by placing an and sign in front of the function's name and in the caller's code, assigning the return value by reference to $value:

Example:

```
function and get_global_variable($name)
{
        return $GLOBALS[$name];
}
$num = 10;
$value =& get_global_variable("num");
print $value . "\n";
$value = 20;
print $num;
```

The previous code prints as

10

20

You can see that $num was successfully modified by modifying $value, because it is a reference to the global variable $num.

You won't need to use this returning method often. When you do, use it with care, because forgetting to assign by reference the by-reference returned value can lead to bugs that are difficult to track down.

7.5 Declaring Function Parameters

As previously mentioned, you can pass an arbitrary amount of arguments to a function. There are two different ways of passing these arguments. The first is the most common, which is called **passing by value**, and the second is called **passing by reference**. Which kind of argument passing you would like is specified in the function definition itself and not during the function call.

7.5.1 By-Value Parameters

Here, the argument can be any valid expression, the expression is evaluated, and its value is assigned to the corresponding variable in the function. For example, here, $x is assigned the value 8 and $y is assigned the value of $c:

Example:

```
Function pow($x, $y)
{
...
}
pow(2*4, $c);
```

7.5.2 By-Reference Parameters

Passing by-reference requires the argument to be a variable. Instead of the variable's value being passed, the corresponding variable in the function directly refers to the passed variable whenever used. Thus, if you change it inside the function, it affects the sent variable in the outer scope as well:

Example:

```
function square(&$n)
{
        $n = $n*$n;
}
$number = 4;
square($number);
print $number;
```

The & sign that proceeds $n in the function parameters tells PHP to pass it by-reference, and the result of the function call is $number squared; thus, this code would print 16.

7.6 Default Parameters

Default parameters like C++ are supported by PHP. **Default P( )arameters** enable you to specify a default value for function parameters that aren't passed to the function during the function call. The default values you specify must be a constant value, such as a scalar, array with scalar values, or constant.

The following is an example for using default parameters:

Example:

```
function increment(&$num, $increment = 1)
{
        $num += $increment;
}
$num = 4;
increment($num);
increment($num, 3);
```

This code results in $num being incremented to 8. First, it is incremented by 1 by the first call to increment, where the default increment size of 1 is used, and second, it is incremented by 3, altogether by 4.

Did u know? When you a call a function with default arguments, after you omit a default function argument, you must emit any following arguments. This also means that following a default argument in the function's definition, all other arguments must also be declared as default arguments.

7.7 Static Variables

Like C, PHP supports declaring local function variables as static. These kind of variables remain in tact in between function calls, but are still only accessible from within the function they are declared. Static variables can be initialized, and this initialization only takes place the first time the static declaration is reached.

Notes

Here's an example for the use of static that runs initialization code the first time (and only the first time) the function is run:

Example:

```
function do _something()
{
     static first _time = true;
     if (first _time) {
       // Execute this code only the first time the
          function is
        →called
       ...
     }
    // Execute the function's main logic every time
      the function is
     →called
      ...
      }
```

7.8 Arrays

A variable is a storage area holding a number or text. The problem is, a variable will hold only one value. An array is a special variable, which can store multiple values in one single variable.

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.

For now, let's take a look at the general syntax of the array() statement:

Array array([key]=>[value], [index2]=>[value], ...);

In PHP, there are three kinds of arrays:

- Numeric array - An array with a numeric index.
- Associative array - An array where each ID key is associated with a value.
- Multidimensional array - An array containing one or more arrays.

Numeric Arrays

A numeric array stores each array element with a numeric index. There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

   ```
   $cars=array("Saab","Volvo","BMW","Toyota");
   ```

2. In the following example we assign the index manually:

   ```
   $cars[0]="Saab";
   ```

```
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Example: In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

The code above will output:

Saab and Volvo are Swedish cars.

Associative Arrays

An associative array, each ID key is associated with a value. When storing data about specific named values, a numerical array is not always the best way to do it. With associative arrays we can use the values as keys and assign values to them.

Example: 1. In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Example: 2. This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
The ID keys can be used in a script:
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

Notes

Multidimensional 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: 1. In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
   (
   "Griffin"=>array
   (
   "Peter",
   "Lois",
   "Megan"
   ),
   "Quagmire"=>array
   (
   "Glenn"
   ),
   "Brown"=>array
   (
   "Cleveland",
   "Loretta",
   "Junior"
   )
   );
```

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
    (
      [0] => Peter
      [1] => Lois
      [2] => Megan
      )
[Quagmire] => Array
      (
      [0] => Glenn
      )
[Brown] => Array
      (
    [0] => Cleveland
    [1] => Loretta
    [2] => Junior
    )
)
```

Example: 2. Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
The code above will output:
Is Megan a part of the Griffin family?
```

Did u know? Arrays in PHP are implemented using hash tables, which means that accessing a value has an average complexity of O(1).

7.8.1 Array ( ) Construct

Array( ) is one of the methods PHP provides to create and populate an array with data. It is a PHP statement that takes your input and returns an array variable containing that input. Although the general format never changes.

Syntax

An array can be created by the array( ) language construct. It takes as parameters any number of comma-separated key => value pairs.

```
array(  key =>  value
     , ...
     )
// key may only be an integer or string
// value may be any value of any type
```

Now that you have an idea of what an array( ) statement looks like, let's look at how your earlier syntax of an array can be stored as an array:

```
<?php
  $val1 = "car";

  $mycar = array(0=>"cycle",
                 1=>"byke",
                 2=>$val1);
?>
```

Notes

7.8.2 Accessing Array Elements

Array elements can be accessed by using the $arr[key] notation, where key is either an integer or string expression. When using a constant string for key, make sure you don’t forget the single or double quotes, such as $arr[“key”]. This notation can be used for both reading array elements and modifying or creating new elements.

Example: `<?php echo $val1[2]; ?>`

7.8.3 Modifying/Creating Array Elements

```
$arr1 = array(1, 2, 3);

$arr2[0] = 1;

$arr2[1] = 2;

$arr2[2] = 3;

print_r($arr1);

print_r($arr2);
```

The print_r() function has not been covered yet in this book, but when it is passed an array, it prints out the array’s contents in a readable way. You can use this function when debugging your scripts.

The previous example prints

```
Array

(

[0] => 1

[1] => 2

[2] => 3

)

Array

(

[0] => 1

[1] => 2

[2] => 3

)
```

So, you can see that you can use both the array() construct and the $arr[key] notation to create arrays. Usually, array() is used to declare arrays whose elements are known at compile-time, and the $arr[key] notation is used when the elements are only computed at runtime. PHP also supports a special notation, $arr[], where the key is not specified. When creating new array offsets using this notation (for example, using it as the l-value), the key is automatically assigned as one more than the largest previous integer key.

Notes

Therefore, the previous example can be rewritten as follows:

```
$arr1 = array(1, 2, 3);
$arr2[] = 1;
$arr2[] = 2;
$arr2[] = 3;
```

The result is the same as in the previous example.

The same holds true for arrays with string keys:

```
$arr1 = array("name" => "Peter", "age" => 28);
$arr2["name"] = "Peter";
$arr2["age"] = 28;
if ($arr1 == $arr2) {
print '$arr1 and $arr2 are the same' . "\n";
}
```

The message confirming the equality of both arrays is printed.

7.8.4 Reading Array Values

You can use the $arr[key] notation to read array values. The next few examples build on top of the previous example:

```
print $arr2["name"];
if ($arr2["age"] < 35) {
print "is quite young\n";
```

This example prints

```
Peter is quite young
```

Caution As previously mentioned, using the $arr[] syntax is not supported when reading array indexes, but only when writing them.

7.8.5 Accessing Nested Arrays (or Multi-Dimensional Arrays)

When accessing nested arrays, you can just add as many square brackets as required to reach the relevant value. The following is an example of how you can declare nested arrays:

```
$arr = array(1 => array("name" => "Peter", "age" => 28), array("name"
|→ => "Barbara", "age" => 67))
```

You could achieve the same result with the following statements:

```
$arr[1]["name"] = "Peter";
$arr[1]["age"] = 28;
```

```
$arr[2][“name”] = “Barbara”;

$arr[2][“age”] = 67;
```

Reading a nested array value is trivial using the same notation. For example, if you want to print Peter’s age, the following statement does the trick:

```
print $arr[1][“age”];
```

7.8.6 Traversing Arrays Using Foreach

There are a few different ways of iterating over an array. The most elegant way is the foreach() loop construct.

The general syntax of this loop is

```
foreach($array as [$key =>] [&] $value)

...
```

$key is optional, and when specified, it contains the currently iterated value’s key, which can be either an integer or a string value, depending on the key’s type.

Specifying and for the value is also optional, and it has to be done if you are planning to modify $value and want it to propagate to $array. In most cases, you won’t want to modify the $value when iterating over an array and will, therefore, not need to specify it.

Here’s a short example of the foreach() loop:

```
$players = array(“Peter”, “Barbara”, “Bill”, “Nancy”);

print “The players are:\n”;

foreach ($players as $key => $value) {

print “#$key = $value\n”;

}
```

The output of this example is

```
The players are:

#0 = Peter

#1 = Barbara

#2 = Bill

#3 = Nancy
```

Here’s a more complicated example that iterates over an array of people and marks which person is considered old and which one is considered young:

```
$people = array(1 => array(“name” => “Peter”, “age” => 28),

|→array(“name” => “Barbara”, “age” => 67));

foreach ($people as &$person) {

if ($person[“age”] >= 35) {
```

```
$person["age group"] = "Old";
} else {
$person["age group"] = "Young";
}
}
print_r($people);
```

Again, this code makes use of the print_r() function.

The output of the previous code is the following:

```
Array
(
[1] => Array
(
[name] => Peter
[age] => 28
[age group] => Young
)
[2] => Array
(
[name] => Barbara
[age] => 67
[age group] => Old
        )
)
```

You can see that both the Peter and Barbara arrays inside the $people array were added an additional value with their respective age group.

7.8.7 Traversing Arrays Using List( ) and Each( )

Although foreach() is the nicer way of iterating over an array, an additional way of traversing an array is by using a combination of the list( ) construct and the each( ) function:

```
$players = array("Peter", "Barbara", "Bill", "Nancy");
reset($players);
while (list($key, $val) = each($players)) {
print "#$key = $val\n";
}
```

The output of this example is

```
#0 = Peter
#1 = Barbara
#2 = Bill
#3 = Nancy
```

7.8.8 Reset( )

Iteration in PHP is done by using an internal array pointer that keeps record of the current position of the traversal. Unlike with foreach(), when you want to use each() to iterate over an array, you must reset() the array before you start to iterate over it. In general, it is best for you to always use foreach() and not deal with this subtle nuisance of each() traversal.

7.8.9 Each( )

The each( ) function returns the current key/value pair and advances the internal pointer to the next element. When it reaches the end of of the array, it returns a booloean value of false. The key/value pair is returned as an array with four elements: the elements 0 and "key", which have the value of the key, and elements 1 and "value", which have the value of the value. The reason for duplication is that, if you're accessing these elements individually, you'll probably want to use the names such as

$elem["key"] and $elem["value"]:

```
$ages = array("Peter" => 28, "Barbara" => 67);
reset($ages);
$person = each($ages);
print $person["key"];
print " is of age ";
print $person["value"];
```

This prints

```
Peter is of age 28
```

When we explain how the list() construct works, you will understand why offsets 0 and 1 also exist.

7.8.10 List( )

The list( ) construct is a way of assigning multiple array offsets to multiple variables in one statement:

```
list($var1, $var2, ...) = $array;
```

The first variable in the list is assigned the array value at offset 0, the second is assigned offset 1, and so on. Therefore, the list() construct translates into the following series of PHP statements:

```
$var1 = $array[0];
```

```
$var2 = $array[1];
...
```

As previously mentioned, the indexes 0 and 1 returned by each() are used by the list() construct. You can probably already guess how the combination of list() and each() work. Consider the highlighted line from the previous $players traversal example:

```
$players = array("Peter", "Barbara", "Bill", "Nancy");
reset($players);
while (list($key, $val) = each($players)) {
print "#$key = $val\n";
}
```

What happens in the boldfaced line is that during every loop iteration, each() returns the current position's key/value pair array, which, when examined with print_r(), is the following array:

```
Array
(
[1] => Peter
[value] => Peter
[0] => 0
[key] => 0
)
```

Then, the list() construct assigns the array's offset 0 to $key and offset 1

to $val.

7.8.11 Additional Methods for Traversing Arrays

You can use other functions to iterate over arrays including current() and next(). You shouldn't use them because they are confusing and are legacy functions. In addition, some standard functions allow all sorts of elegant ways of dealing with arrays such as array_walk(), which is covered in a later unit.

7.9 Objects

The main difference in OOP as opposed to functional programming is that the data and code are bundled together into one entity, which is known as an object. Object-oriented applications are usually split up into a number of objects that interact with each other. Each object is usually an entity of the problem, which is self-contained and has a bunch of properties and methods. The properties are the object's data which basically means the variables that belong to the object. The methods if you are coming from a functional background are basically the functions that the object supports. Going one step further, the functionality that is intended for other objects to be accessed and used during interaction is called an object' **sinter face.**

7.9.1 Creating an Object

Object Initialization: To create a new object, use the new statement to instantiate a class:

```
<?php
class foo
{
       function do_foo()
       {
              echo "Doing foo.";
       }
}

$bar = new foo;
$bar->do_foo();
?>
```

Converting to object: If an object is converted to an object, it is not modified. If a value of any other type is converted to an object, a new instance of the stdClass built-in class is created. If the value was NULL, the new instance will be empty. Arrays convert to an object with properties named by keys, and corresponding values. For any other value, a member variable named scalar will contain the value.

```
<?php
$obj = (object) 'ciao';
echo $obj->scalar;  // outputs 'ciao'
?>
```

7.9.2 Object Inheritance

Inheritance is a well-established programming principle, and PHP makes use of this principle in its object model. This principle will affect the way many classes and objects relate to one another.

For example, when you extend a class, the subclass inherits all of the public and protected methods from the parent class. Unless a class overrides those methods, they will retain their original functionality.

This is useful for defining and abstracting functionality, and permits the implementation of additional functionality in similar objects without the need to remployment all of the shared functionality.

Example:

```
<?php
class foo
```

```
{
    public function printItem($string)
      {
          echo 'Foo: ' . $string . PHP_EOL;
       }
           public function printPHP()
      {
          echo 'PHP is great.' . PHP_EOL;
       }
  }

class bar extends foo
{
     public function printItem($string)
    {
        echo 'Bar: ' . $string . PHP_EOL;
      }
}

$foo = new foo();
$bar = new bar();
$foo->printItem('baz'); // Output: 'Foo: baz'
$foo->printPHP(); // Output: 'PHP is great'
$bar->printItem('baz'); // Output: 'Bar: baz'
$bar->printPHP(); // Output: 'PHP is great'
?>
```

7.10 Summary

- The general way of defining a function is

  Function function_name (arg1, arg2, arg3,………).
- Every function has its own set of variables. Any variable used outside the function's definition are not accessible from within the function by default.
- The return statement returns value by value, which means that a copy of the value is created and is returned to the caller of the function.
- Default parameters enable you to specify a default value for function parameters that are not passed to the function during the function call.

Notes

- An array in PHP is actually an ordered map. A map is a type that associates values to keys.
- The main difference in OOP as opposed to functional programming is that the data and code are bundled together into one entity, which is known as an object.

7.11 Keywords

Multi-Dimensional Arrays**:** When accessing nested arrays, you can just add as many square brackets as required to reach the relevant value. The following is an example of how you can declare nested arrays.

Arrays: An **array** in PHP is a collection of key/value pairs. This means that it maps keys (or indexes) to values.

Default Parameters: Default parameters like C++ are supported by PHP. **Default Parameters** enable you to specify a default value for function parameters that aren't passed to the function during the function call.

Returning Values By Reference: PHP also allows you to return variables **by reference.**

Declaring Function Parameter: There are two different ways of passing these arguments. The first is the most common, which is called **passing by value**, and the second is called **passing by reference**. Which kind of argument passing you would like is specified in the function definition itself and not during the function call.

By-Value Parameters: Here, the argument can be any valid expression, the expression is evaluated, and its value is assigned to the corresponding variable in the function.

By-Reference Parameters: Passing by-reference requires the argument to be a variable. Instead of the variable's value being passed, the corresponding variable in the function directly refers to the passed variable whenever used.

Static Variables: Static variables can be initialized, and this initialization only takes place the first time the static declaration is reached.

Accessing Array Elements: Array elements can be accessed by using the $arr[key] notation, where key is either an integer or string expression.

Objects: The main difference in OOP as opposed to functional programming is that the data and code are bundled together into one entity, which is known as an object.

Case Study

We'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...

Notes

"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 is generally needs to have this property.

7.12 Self Assessment Questions

1. ______________ in PHP is a collection of key/value pairs.
2. ______________ is like C++ are supported by PHP.
3. Array elements can be accessed by using the $arr[key] notation.

   (*a*) True (*b*) False
4. Arrays in PHP are implemented using hash tables.

   (*a*) True (*b*) False
5. PHP also allows you to return variables ______________ .
6. Array elements can be accessed by using the ______________ .

   (*a*) $arr[key] notation (*b*) Z. arr[key] notation

   (*c*) P arr[key] notation (*d*) all of the above.

7.13 Review Questions

1. Define the Function scope.
2. What are user defined functions?
3. What is function scope?
4. Define the returning value.
5. What is default parameters?
6. Define static variables.

Notes

7. What is arrays? Describe all parts of arrays.
8. Define array elements.
9. How can you declare nested arrays?
10. Give an example for using default parameters.
11. How many ways of defining a function are there?
12. Define objects.

Answer for Self Assessment Questions

1.	An Array	2.	Default parameters.
3.	(*a*)	4.	(*a*)
5.	by reference	6.	(*a*)

7.14 Further Reading

Books *Open Source Development with LAMP,* by Meloni, Pearson Education.

Online link htttp://www.HTMLcenter.com/blog/teach-yourself-php-in-24hours.

Unit 8: Working with Strings, Date and Time

Notes

CONTENTS

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.

8.1 Date Handling

PHP has a range of functions that handle date and time. Some of these functions work with a so-called **UNIX timestamp**, which is the number of seconds since January 1, 1970 at 00:00:00 GMT, the beginning of the UNIX epoch. Because PHP only handles unsigned 32-bit integers

Notes

and most operating systems don't support negative timestamps, the range in which most of the PHP date functions operate is January 1, 1970 to January 19, 2038. The PEAR::Date package handles dates outside this range and also in a platform-independent way.

8.2 Retrieving Date and Time Information

The easiest way of obtaining the current time is with the time( ) function. It accepts no parameters and simply returns the current timestamp:

```
<?php
      echo time( ); // Outputs something similar to "1077913162"
?>
```

The resolution is 1 second. If you want some more accuracy, you have two options: microtime() and gettimeofday(). The microtime() function has one annoying peculiarity: The return value is a floating-point number containing the decimal part of the timestamp and the number of seconds since the epoch, concatenated with a space. This makes it, of course, a bit hard to use for a timestamp with sub-second resolution:

```
<?php
      // Outputs something similar to "0.87395100 1078006447"
      echo microtime();
      $time = preg_replace('@^(.*)\s+(.*)$@e', '\\2 + \\1',
      'microtime());
      echo $time; // Outputs 1078006447.8741
?>
```

In putting the two parts back together, you lose some of the precision. The gettimeofday() function has a nicer interface. It returns an array with elements representing the timestamp and additional microseconds. Two more elements are included in this array, but you cannot really rely on them because the underlying system functionality—at least in Linux—is not working correctly:

```
<?php
      print_r(gettimeofday());
?>
returns
Array
  (
      [sec] => 1078006910
      [usec] => 339699
      [minuteswest] => -60
      [dsttime] => 0
 )
```

Localtime() and getdate() both return an array. The elements contain information belonging to the (optional) timestamp passed to the function. The returned arrays are not exactly the same. Table shows what the elements in the arrays mean.

Table 1: Elements in Arrays Returned by localtime( ) and getdate( )

Meaning	**Index (*localtime()*)**	**Index (*localtime()*)**	**Index (*localtime()*)**
Hours	tm_hour	hours	
Day of month	tm_mday	mday	
Month	tm_mon	mon	For localtime: January=0; for getdate:January=1
Year	tm_year	Year	
Day of week	tm_wday	wday	With 0 being Sunday and 6 being Saturday
Day of year	tm_yday	yday	With 0 being January 1st and 366 being December 32nd
DST in effect		tm_isdst	Set to true if Daylight Savings Time is in effect
Textual day of		weekday	English name of week the weekday
Textual month		month	English name of the month
Timestamp			Number of seconds since 01-01-1970

The tm_isdst element of localtime() is especially interesting. It's the only way in PHP to see whether the server is in DST. Also, note that the month number in the return array of localtime() starts with 0, not with 1, which makes December month 11. The first parameter for both functions is a time stamp, allowing the functions to return date information based on the time you pass them, rather than just on the current time. localtime() normally returns an array with numerical indices, rather than the indices as described in the previous table. To signal the function to return an associative array, you need to pass true as the second parameter. If you want to return this associative array with information about the current time, you need to pass the time() function as first parameter:

```
<?php
      print_r(localtime(time(), true));
?>
```

Two more date functions are available: gmmktime() and mktime(). Both functions create a timestamp based on parameters passed when the function is called. The difference between the two functions is that gmmktime() treats the date/time parameters passed as a Greenwich Mean Time (GMT), while parameters passed to mktime() are treated as local time. The order of parameters is not very user friendly, as you can see in the prototype of the following function:

timestamp mktime ([$hour [, $minute [, $second [, $month [, $day [,

'$year [, $is_dst]]]]]]])

Caution The particularly weird order of the parameters.

All parameters are optional. If any parameter is not included, the "current" value is used, depending on the current date and time. The last parameter, is_dst, controls whether the date and time parameters that are passed to the function are DST-enabled or not. The default value for the parameter is -1, which signals PHP to determine for itself whether the date falls into the range when DST is observed. Here is an example:

Example:
```
<?php
/* mktime with a date outside the DST range */
echo date("Ymd H:i:s", mktime(15, 16, 17, 1, 17, 2004)). "\n";
echo date("Ymd H:i:s", mktime(15, 16, 17, 1, 17, 2004, 0)). "\n";
echo date("Ymd H:i:s", mktime(15, 16, 17, 1, 17, 2004, 1)). "\n";
/* mktime with a date inside the DST range */
echo date("Ymd H:i:s", mktime(15, 16, 17, 6, 17, 2004)). "\n";
echo date("Ymd H:i:s", mktime(15, 16, 17, 6, 17, 2004, 0)). "\n";
echo date("Ymd H:i:s", mktime(15, 16, 17, 6, 17, 2004, 1)).
"\n\n";
 ?>
```

The first three calls "make" a timestamp for January 17, in which no DST is observed. Therefore, setting the $is_dst parameter to 0 has no effect on the returned timestamp. If it's set to 1, though, the timestamp will be one hour earlier, as the mktime() function converts the DST time (which is always one hour ahead of non-DST). For the second set of mktime() calls, we use June 17 in which DST is observed. Setting the $is_dst parameter to 0 now makes the function convert the time from non-DST to DST and, thus, the returned timestamp will be one hour ahead of the result of the first and third calls. The output is

```
20040217 15:16:17
20040217 15:16:17
20040217 14:16:17
20040617 15:16:17
20040617 16:16:17
20040617 15:16:17
```

It's best not to touch the $is_dst parameter, because PHP usually interprets the date and time correctly.

If we replace all calls to mktime() by gmmktime(), the parameters passed to the function are treated as GMT time, with no time zones taken into account. With mktime(), the time zone that the server has configured is taken into account. For instance, if you are on Central European Time (CET), passing the same parameters as shown previously to gmmktime output times that are one hour "later." Because the date function *does* take into account time zones, the generated GMT timestamp is treated as a CET time zone, resulting in times that are one hour for non DST times and two hours for DST times (CEST is CET+1).

8.3 Formatting Date and Time

Making a GMT date with gmmktime() and then showing it in the current time zone with the date() function doesn't make much sense. Thus, we also have two functions for formatting date/time: date() to format a local date/time, and gmdate() to format a GMT date/time. Both functions accept exactly the same parameters. The first parameter is a format string (more about that in a bit), and the second is an optional timestamp. If the timestamp parameter is not included, the current time is used in formatting the output. gmdate() and date() always format the date in English, not in the current "locale" that is set on your system. Two functions are provided to format local time/date according to locale settings: strftime() for local time and gmstrftime() for GMT times. Table describes formatting string characters for both functions. Note that the (gm) strftime() prefix to the formatting string options with a %.

Table 2: Date Formatting Modifiers

Description	**Date/ gmdate**	**Strftime/ gmstrime**	**Remarks**
AM/PM	A		
am/pm	a	%p	Either am or pm for the English locale. Other Locates Migh have replacements (for example, n1_ NL has an empty atring here).
Century, numeric two digits		%C	Returne the century number 20 for 2004, and so on.
Character, literal%		%%	Use this to place a literal character % inside the formatting string.
Character, newline		%n	Use this to place a newline character inside the formatting string.
Character, tab		%t	Use this to place a tab character inside the formatting string.
Day connt in month	t		Number of days in the month defined by the timestamp.
Day of month, leading spaces		%e	Current day in this month defined by the timestamp. A space is prepended when the day number is less than 10.

Contd...

Notes

Day of month, leading zeroes	d	%D	Current day in this month defined by the timestamp. A zero is prepended when the day number is less than 10.
Day of month, without leading zeroes	j		Current day in the month defined by the timestamp.

Formatted, locale preferred date		%x	The date in preferred Locale format
			`<?Phy`
			`setlocale (LC-All, ‘1w_IL’),`
			`echo strft1me (‘%X\N”);//`
			`→ shows 29/02/04 ?>`

Formatted, locale Preferred date and time		%c	The date and time in preferred locale format format.
			`<? php`
			`setlocale 9LC_All, n1_NL’);`
			`// shows zo 29 feb 2004`
			`→ 23: 56:12 CET`
			`echo strft1me (“%C/n”);`
			`?>`

Formatted, locale preferred time		%x	The date in preferred locale format.
			`<? php`
			`setlocale (LC_ALL, ‘n1_NL’);`
			`echo strftime (‘%x\n”); //`
			`→ shows 29-02-04`
			`?>`

Hour 12-hour format, leading zeroes	h	%1	
Hour 12-hour format, no leading zeroes	g		
Hour, 24-hour format, leading zeroes	H	%H	
Hour, 24-hour format, no leading zeroes	G		

Contd...

Notes

Internet time	B		The swath Internet time in which a day is
			divided into 1,000 units:
			`<? php`
			`echo date ('B'). "\n"//shows`
			`→ 00s`
			`?>`

ISO 8601	c		Shows the date in ISO 8601 format: 2004-03-01Too: 08:37+01:00
Leap Year	L		Returns 1 if the year represented by the timestamp is a leap year, or 0 otherwise.
Minutes, leading Zeroes	1	%M	

Month, full textual	p	%B	For (gm) strftime (), the month name is the name in the language of the current locale.
			`<?php`
			`set locale (LC_ALL, '1w_IL')'`
			`echo strftime 9"%B\n");//`
			`shows`
			`?>`

Month, numeric with leading zeroes	M	%m	
Month, numeric without leading zeroes	N		
Month, short textual	M	%b, %h	
RFC 2822	R		Returns a RFC 2822 (mail) formatted text (Mon, 1 Mar, 2004 00:13:34 + 0100).
Seconds since UNIX epoch	U		
Seconds, numberic with leading zeroes	s	%S	

Suffix for day of month, English ordinal	S		Returns an English ordinal suffix for use with the J formatting option.
			`<?php`
			`echo date ("js\n"); // returns`
			`→ 1st`
			`?>`

Contd...

Notes

Time zone, numeric (in seconds)	Z		Returns the offest to GMT in seconds. For CET, this is 3600; for EST, this is –18000, seconds) for example.
Time zone, numeric formatted	O		Returns a formatted offset to GMT. For GET, this is +0100; for EST, this is –0500, for example.
Time zone, textual	T	%Z	Returns the current time zone name: CET, EST, and so on.
Week number, ISO 8601	W	%V	In ISO 8601, week # 1 is the first week in the year having four or more days. The range is 01 to 53, and you can use this in combination with %g or %G for the accompanying year:

Week number, the first Monday in a year is the start of week 1		%W	`<? php`
			`// shows 01`
			`echo strftime ("%w"'`
			`strtot1me ("2001-01-0-`
			`→ 01")), "\m",`
			`// shows 53`
			`echo strft1me ("%W",`
			`strtot1me ("2005-12`
			`→ 31")),"\n",`
			`?>`

Week number, the first Sunday in a year is the start of week 1		%U	`<? php`
			`// shows 00`
			`echo strftime ("%U",`
			`strtotime ("2001-01-)`
			`01")), "\n";`
			`// shows 52`
			`echo strftime ("%U",`
			`strtotime ("2001-12-`
			`31"))."\n";`
			`?>`

Year, numeric two digits with	Y	%Y	
Year, numeric two digits; year component for %w		%g	This number might differ from the "real year", as in ISO 8601; January 1 might still belong to week 53 of the year before. In that case, the year returned with this formatting option will be the one of the previous year, too.

Contd...

Year, numeric four digits	Y	%Y	
Year, numeric four digits; year component for %w		%G	This number might differ from the "real year'" as in ISO 8601; January 1 might still belong to week 53 of the year before. In that case, the year returned with this formatting option will be the one of the previous year, too.
Day of week, full textual	1	%A	For strftime (), the day is shown according to the names of the current locale. `<?php` `setlocale (LC_ALL, 'C');` `echo strftime ('%A');` `setlocale (LC_ALL, 'no_NO');` `echo strftime ('%A');` `?>` shows Monday Mandag
Day of week, numeric (0=Sunday)	w	%W	The range is 0-6 with 0 being Sunday and 6 being Saturday.
Day of week, numeric (1=Monday)		%u	The range is 1-7 with I being Monday and 7 being Sunday.
Day of week, short textual	D	%a	For the (gm) strftime ( ) function, the name is shown according to the locale; for (gm) date( ) it is the normal three letter abbreviation: Sun, Sat, Wed, and so on.
Day of year, numeric with leading zeroes		%j	The day number in a year, starting with 001 for January 1 to 365 or 366.
Day of year, numeric without leading zeroes	z		The day number in a year, starting with 0 for January 1 to 365 or 365
DST active	I		Returns 1 if DST is active and 0 if DST is not active for the given timestamp.
Formatted, %d/%m/%y		%D	Gives the same result as using %d/%m/%y.
Formatted, %H:%M:%S		%T	Gives the same result as using &H;%M:%S.

Contd...

Notes

Formatted, in 24-hour notation		%R	The time in 24-hour notation withour seconds.
			`<?php`
			`echo strftime ("%R\n");//`
			`shows`
			`23:53`
			`?>`

Formatted, in am/ p.m. notation		%r	The time in 12-hour notation including seconds.
			`<?php`
			`echo strftime ("%r\n");//`
			shows
			11:53:47
			?>

Task **ISO 8601 Week Numbers:** That the ISO 8601 year format option (%V) might differ from the normal year format option (%Y) if a year has less than four days:

```
<?php
     for ($i = 27; $i <= 31; $i++) {
     echo gmstrftime(
     "%Y-%m-%d (%V %G, %A)\n",
     gmmktime(0, 0, 0, 12, $i, 2004)
);
}
for ($i = 1; $i <= 6; $i++) {
echo gmstrftime(
     "%Y-%m-%d (%V %G, %A)\n",
     gmmktime(0, 0, 0, 1, $i, 2005)
);
}
?>
```

The script outputs

```
2004-12-27 (53 2004, Monday)
2004-12-28 (53 2004, Tuesday)
2004-12-29 (53 2004, Wednesday)
```

Contd...

2004-12-30 (53 2004, Thursday)

2004-12-31 (53 2004, Friday)

2005-01-01 (53 2004, Saturday)

2005-01-02 (53 2004, Sunday)

2005-01-03 (01 2005, Monday)

2005-01-04 (01 2005, Tuesday)

2005-01-05 (01 2005, Wednesday)

2005-01-06 (01 2005, Thursday)

As you can see, the ISO year is different for January 1 and 2, 2005, because the first week (Monday to Sunday) only has two days.

Questions

1. Briefly explain how the ISO 8601 year format option.
2. Give the output of script.

Lab Exercise **DST Issues:** Every year around October, at least 10–25 bugs are reported when a day is listed twice in somebody's overview. Actually, the day listed twice is the date on which DST ends, as you can see in the following example.

Example:

```
<?php
    /* Start date for the loop is October 31th, 2004 */
    $ts = mktime(0, 0, 0, 10, 31, 2004);
    /* We loop for 4 days */
    for ($i = 0; $i < 4; $i++) {
    echo date ("Y-m-d (H:i:s)\n", $ts);
    $ts += (24 * 60 * 60); /* 24 hours */
}
?>
```

When this script is run, you see the following output:

```
    2004-10-31 (00:00:00)
    2004-10-31 (23:00:00)
    2004-11-01 (23:00:00)
    2004-11-02 (23:00:00)
```

The 31st is listed twice because there are actually 25 hours between midnight, October 31 and November 1, not the 24 hours that were added in our loop. You can solve the problem in one of two ways. If you pick a different time of day, such as noon, the script will always have the correct date:

Example:

```
<?php
    /* Start date for the loop is October 29th, 2004 */
    $ts = mktime(12, 0, 0, 10, 29, 2004);
    /* We loop for 4 days */
    for ($i = 0; $i < 4; $i++) {
```

Notes

```
        echo date ("Y-m-d (H:i:s)\n", $ts);
        $ts += (24 * 60 * 60);
    }
    ?>
```

Output is

```
    2004-10-29 (12:00:00)
    2004-10-30 (12:00:00)
    2004-10-31 (11:00:00)
    2004-11-01 (11:00:00)
```

Caution However, there is still a difference in the time. A better solution is to abuse the mktime() function a little.

```
<?php
    /* We loop for 6 days */
    for ($i = 0; $i < 6; $i++) {
    $ts = mktime(0, 0, 0, 10, 30 + $i, 2004);
    echo date ("Y-m-d (H:i:s) T\n", $ts);
}
?>
```

Output is

```
    2004-10-30 (00:00:00) CEST
    2004-10-31 (00:00:00) CEST
    2004-11-01 (00:00:00) CET
    2004-11-02 (00:00:00) CET
    2004-11-03 (00:00:00) CET
    2004-11-04 (00:00:00) CET
```

We add the day offset to the mktime() parameter that describes the day of month. mktime() then correctly wraps into the next months and years and takes care of the DST hours, as you can see in the previous output.

Did u know? Sometimes, you want to show a formatted time in the current time zone and in other time zones as well. The following script shows a full textual date representation for the U.S., Norway, the Netherlands, and Israel:

```
<?php
    echo strftime("%c\n");
    echo "\nEST in en_US:\n";
```

```
        setlocale(LC_ALL, "en_US");
        putenv("TZ=EST");
        echo strftime("%c\n");
        echo "\nMET in nl_NL:\n";
        setlocale(LC_ALL, "nl_NL");
        putenv("TZ=MET");
        echo strftime("%c\n");
        echo "\nMET in no_NO:\n";
        setlocale(LC_ALL, "no_NO");
        putenv("TZ=MET");
        echo strftime("%c\n");
        echo "\nIST in iw_IL:\n";
        setlocale(LC_ALL, "iw_IL");
        putenv("TZ=IST");
        echo strftime("%c\n");
?>
```

Output. is

```
MON Mar     1       20:19:20        2004
EST in en_us:
MON Mar     1       14:19:20        2004
MET in nl_Nil:
Na 01 nrt 2004      20:19:20        MET
IST in iw_IL:
IST                 21:13:20        2004
```

March 1 in different locales.

Caution You need to have the locales and time-zone settings installed on your system before this will work. It is a system-dependent setting and not everything is always available on your system. If you're a Mac OS X user, have a look at http://www.macmax.org/locales/index_en.html to install locales.

8.4 Parsing Date Formats

The opposite of formatting text is parsing a textual description of a date into a timestamp. The strtotime() function handles a many different formats. Table contains a list of the most useful formats.

Notes

Table 3: Date/Time Formats as Understood by strtotime()

Date String	GMT Formatted Date	Remarks
11970-0917	1970-09-16 23:00:00	ISO 8601 preferred date.
9/17/72	1972-09-16 23:00:00	Common U.S. way (d\m\yy)
24 September 1972	1972-09-23 23:00:00	Without any specified time, 0:00 is used. Because the time zone is set to MET (GMT+1), the GMT formatted date is in the previous day.
24 Sep 1972	1972-09-23 23:00:00	
Sep 24, 1972	1972-09-23 23:00:00	
20: 02: 00	2004-03-01 19:02:00	Without any date specified, the current date is used.
20:02	2004-03-01 19:02:00	
8: 02pm	2004-03-01 19:02:00	
20:02-0500	2004-03-02 01:02:00	–0500 is the time zone (EST)
20:02 EST	2004-03-03 01:02:00	
Thursday	2004-03-03 23:00:00	A day name advances to the first available day with this name. In case the current day has this name, the current day is used.
1 Thursday		
this Thursday		
2 Thursday 19:00	2004-03-11 18:00:00	2 is the second Thursday from now.
Next Thursday 7 pm	2004-03-11 18:00:00	Next means the next available day with this name after the first available day, and thus is the same as 2.
last Thursday 19:34	2004-02-26 18:34:00	The Thursday before the current day. If the name of the day is the same as the current day, the timestamp of the previous day is used.
1 Year 2 days ago	2003-02-27 21:25:44	The current time is used to calculate the relative displacement with The-Sign is needed before every displacement unit; ifit's not used, + is assument. If "ago" is postfixed, the meaning of + and – is reversed. Other possible units are second, minute, hour, week, Month, and fortnight 914 days).
–1 Year –2 days	2003-02027 21:25:44	
–1 year 2 days	2003-03-03 21:25:44	
1 year –2 days	2005-03-27 21:25:44	
tomorrow	2004-03-02 21:25:44	
Yesterday	2004-03-29 21:25:44	
20040301 T00:00:00_1900	2004-02-29 05:00:00	Used for WDDX parsing.

Contd...

Notes

2004W021	2004-01-04 23:00:00	Midnight of the first day of ISO week 21 in 2004.
2004122 0915	2004-12-22 08:15:00	Only numbers in the form yyyy mmdd hhmm.

Using the strtotime() function is easy. It accepts two parameters: the string to parse to a timestamp and an optional timestamp. If the timestamp is included, the time is converted relative to the timestamp; if it's not included, the current time is used. The relative calculations are only written with yesterday, tomorrow, and the 1 year 2 days (ago) format strings. strtotime() parsing is always done with the current time zone, unless a different time zone is specified in the string that is parsed:

```
<?php
        echo date("H:i T\n", strtotime("09:22")); // shows 09:22 CET
        echo date("H:i T\n\n", strtotime("09:22 GMT")); // shows 10:22 CET
        echo gmdate("H:i T\n", strtotime("09:22")); // shows 08:22 GMT
        echo gmdate("H:i T\n", strtotime("09:22 GMT")); // shows 09:22 GMT
?>
```

8.5 Strings with PHP

Strings in PHP are a sequence of characters that are always internally nullterminated. However, unlike some other languages, such as C, PHP does not rely on the terminating null to calculate a string's length, but remembers its length internally. This allows for easy handling of binary data in PHP—for example, creating an image on-the-fly and outputting it to the browser. The maximum length of strings varies according to the platform and C compiler, but you can expect it to support at least 2GB. Don't write programs that test this limit because you're likely to first reach your memory limit. When writing string values in your source code, you can use double quotes ("), single quotes (') or here-docs to delimit them. Each method is explained in this section.

Double Quotes Examples for double quotes:

"PHP: Hypertext Pre-processor"

"GET / HTTP/1.0\n"

"1234567890"

Strings can contain pretty much all characters. Some characters can't be written as is, however, and require special notation:

\n	Newline.
\t	Tab.
\"	Double quote.
\\	Backslash.
\0	ASCII 0 (null).

Contd...

Notes

\r	Line feed.
\$	Escape $ sign so that it is not treated as a variable but as the character $.
\ {Octal #}	The character represented by the specified octal #—for example.\70 represents the letter 8.
\X{Hexadecima #}	The character represented by the specified hexadecimal #—for example, \0×32 represents the letter 2.

An additional feature of double-quoted strings is that certain notations of variables and expressions can be embedded directly within them. Without going into specifics, here are some examples of legal strings that embed variables. The references to variables are automatically replaced with the variables' values, and if the values aren't strings, they are converted to their corresponding string representations (for example, the integer 123 would be first converted to the string "123").

"The result is $result\n"

"The array offset $i contains $arr[$i]"

In cases, where you'd like to concatenate strings with values (such as variables and expressions) and this syntax isn't sufficient, you can use the . (dot) operator to concatenate two or more strings. This operator is covered in a later section.

Single Quotes in addition to double quotes, single quotes may also delimit strings. However, in contrast to double quotes, single quotes do not support all the double quotes' escaping and variable substitution.

The following table includes the only two escapings supported by single quotes:

\'	Single quote.
\\	Backslash, used when wanting to represent a backslash followed by a single quote—for example.\\'.

Example: 'Hello, World'

'Today\'s the day'

Here-Docs Here-docs enable you to embed large pieces of text in your scripts, which may include lots of double quotes and single quotes, without having to constantly escape them. The following is an example of a here-doc:

```
<<<THE_END

PHP stands for "PHP: Hypertext Preprocessor".

The acronym "PHP" is therefore, usually referred to as a recursive acronym

↳because the long form contains the acronym itself.

As this text is being written in a here-doc there is no need to escape the

↳double quotes.

THE_END
```

Notes

The strings starts with <<<, followed by a string that you know doesn't appear in your text. It is terminated by writing that string at the beginning of a line, followed by an optional semicolon (;), and then a required newline (\n).

Escaping and variable substitution in here-docs is identical to double-quoted strings except that you are not required to escape double quotes.

8.6 PHP String Handling Functions

PHP is equally powerful than any other server side scripting languages in handling strings. We will experiment with some of the string functions of PHP with some examples. There are powerful functions like regular expressions to manage complex string handling requirements. Here are some of them.

Function	Action
chr	Returns the character corresponding to an ASCII code.
crypt	One way encoding of a string (called hashing).
explode	Converts a character delimited string into an arraylike the VB Split function.
implode	Converts an array into a character delimited string.
ltrim	Remove characters (default is remove blanks) from the beginning of a string.
ord	Return the ASCII code of a character.
parse_str	Extract the variables from a query string passed via a URL.
print	Display a string. Always return 1.
printf	Display a formatted string. Returns the length of the formatted string.
rtrim	Remove characters (default is remove blanks) from the endof a string.
sprintf	Return a formatted string.
str_replacestr_ireplace	Replace all occurrences of a substring within a string.&nsp;
str_split	Split a string into an array equal length chunks (defaultlength 1).
strcmp	Binary-safe string comparison.
strlen	Returns the length of a string.
strposstripos	Returns the position of first occurrence of a substringin a string.
strrev	Returns a string with characters reversed.
strtolower	Returns a string with all characters converted to lowercase.

Contd...

Notes

strtoupper	Returns a string with all characters converted to uppercase.
substr	Extracts a substring from a string.
trim	Remove characters (default is remove blanks) from the beginning and end of a string.

8.7 Accessing String Offsets

Individual characters in a string can be accessed using the $str{offset} notation. You can use it to both read and write string offsets. When reading characters, this notation should be used only to access valid indices. When modifying characters, you may access offsets that don’t yet exist. PHP automatically sets that offset to the said character, and if this results in a gap between the ending of the original string and the offset of the new character, the gap filled with space characters (‘ ’). This example creates and prints the string “Andi” (in an awkward way):

```
$str = “A”;

$str{2} = “d”;

$str{1} = “n”;

$str = $str . “i”;

print $str;
```

Caution For many cases, PHP has string manipulation functions which use efficient algorithms. You should first look at them before you access strings directly using string offsets. They are usually prefixed with str_. For more complex needs, the regular expressions functions—most notably the pcre_family of functions—will come in handy.

Did u know? In PHP 4, you could use [] (square brackets) to access string offsets. This support still exists in PHP 5, and you are likely to bump into it often. However, you should really use the {} notation because it differentiates string offsets from array offsets and thus, makes your code more readable.

8.8 __toString( ) Method

Consider the following code:

```
class Person {
function __construct($name)
        {
$this->name = $name;
```

```
        }
private $name;
        }
$obj = new Person("Andi Gutmans");
print $obj;
```

It prints the following:

```
Object id #1
```

Unlike most other data types, printing the object's id will usually not be interesting to you. Also, objects often refer to data that should have print semantics—for example, it might make sense that when you print an object of a class representing a person, the person's information would be printed out. For this purpose, PHP enables you to implement a function called __toString(), which should return the string representation of the object, and when defined, the print command will call it and print the returned string. By using __toString(), the previous example can be modified to its more useful form:

```
class Person {
function __construct($name)
{
$this->name = $name;
}
function __toString()
{
return $this->name;
}
private $name;
}
$obj = new Person ("Andi Gutmans");
print $obj;
```

It prints the following:

```
Andi Gutmans
```

The __toString( ) method is currently only called by the print and echo language constructs. In the future, they will probably also be called by common string operations, such as string concatenation and explicit casting to string.

Notes

Case Study

Imagine we are working at a school district and need to create a webpage for the students' parents. The webpage has an introduction string that we need to customize depending on if the student is male or female. With *str_replace* this is mighty easy.

PHP Code:

```
//string that needs to be customized
$rawstring = "Welcome Birmingham parents. Your replaceme is a pleasure to have!";
//male string
$malestr = str_replace("replaceme", "son", $rawstring);
//female string
$femalestr = str_replace("replaceme", "daughter", $rawstring);
echo "Son: ". $malestr . "<br />";
echo "Daughter: ". $femalestr;
```

Output:

Son: Welcome Birmingham parents. Your son is a pleasure to have!

Daughter: Welcome Birmingham parents. Your daughter is a pleasure to have!

In the last example we only needed to replace one word *replacement* in our string, but what if we wanted to replace many words? We could just use the function multiple times to get the job done, or we could create an array of *placeholders* and a second array of *replace values* to get it all done in one function call.

The key thing to understand with this technique is that you are creating two arrays that will be used to swap values. The first item in *placeholders* will be replaced by the first item in the *replace values*, the second item of *placeholders* replaced with the second in *replace values* and so on and so forth.

Let's extend our simple example to be a complete form letter addressed to a student's parents.

PHP Code:

```
//string that needs to be customized
$rawstring = "Welcome Birmingham parent! <br />
        Your offspring is a pleasure to have!
        We believe pronoun is learning a lot.<br />
        The faculty simple adores pronoun2 and you can often hear
        them say \"Attah sex!\" "<br />";
//placeholders array
```

Contd...

```
$placeholders = array('offspring', 'pronoun', 'pronoun2', 'sex');
//male replace values array
$malevals = array('son', 'he', 'him', 'boy');
//female replace values array
$malevals = array('son', 'he', 'him', 'boy');
//female replace values array
$femalevals = array('daughter', 'she', 'her', 'girl');

//male string
$malestr = str_replace($placeholders, $malevals, $rawstring);

//female string
$femalestr = str_replace($placeholders, $femalevals, $rawstring);

echo "Son: ". $malestr . "<br />";
echo "Daughter:". $femalestr;
```

Output:

```
Son: Welcome Birmingham parent!
Your son is a pleasure to have! We believe he is learning a lot.
The faculty simple adores he2 and you can often hear them say "Attah boy!"
Daughter: Welcome Birmingham parent!
Your daughter is a pleasure to have! We believe she is learning a lot.
The faculty simple adores she2 and you can often hear them say "Attah girl!"
```

8.9 Summary

- PHP has a range of functions that handle date and time. Some of these functions work with a so-called Unix timestamp, which is the number of seconds since January 1, 1970 at 00:00:00 GMT.
- The easiest way of obtaining the current time is with the time( ) function.
- The opposite of formatting text is parsing a textual description of a date into a timestamp. The strtotime() function handles many different formats.
- Strings in PHP are a sequence of characters that are always internally null terminated.
- Individual characters in a string can be accessed using the $str{offset} notation.

8.10 Keywords

DST Issues: Every year around October, at least 10-25 bugs are reported when a day is listed twice in somebody's overview. Actually, the day listed twice is the date on which DST.

Gmmktime(): Functions create a timestamp based on parameters passed when the function is called.

GMT: Greenwich Mean Time (GMT) is a term originally referring to mean solar time at the Royal Observatory in Greenwich, London.

Here-Docs: Here-docs enable you to embed large pieces of text in your scripts, which may include lots of double quotes and single quotes.

ISO 8601: ISO 8601 Data elements and interchange formats–Information interchange–Representation of dates and times is an international standard covering the exchange of date and time-related data.

Parsing: Parsing a sentence means to computer the structural description (descriptions) of the sentence assigned by a grammar, assuming of course, that the sentence is well-formed. Mathematical work on parsing.

Strings: A string is series of characters, therefore, a character is the same as a byte. That is, there are exactly 256 different characters possible. This also implies that PHP has no native support of Unicode.

UNIX timestamp: PHP has a range of functions that handle date and time. Some of these functions work with a so-called UNIX timestamp.

8.11 Self Assessment Questions

Fill in the blanks:

1. PHP has a range of functions that handle date and time. Some of these functions work with a so-called ____________________ .
2. The opposite of ____________________ is parsing a textual description of a date into a timestamp.
3. Single Quotes in addition to double quotes, single quotes may also delimit ____________________.
4. ____________________ automatically sets that offset to the said character.

State True or False:

1. The easiest way of obtaining the current time is with the time() function.

   (*a*) True (*b*) False

2. Strings in PHP are a sequence of characters that are always internally null-terminated.

   (*a*) True (*b*) False

3. Here-docs enable you to embed large pieces of text in your scripts.

   (*a*) True (*b*) False

4. The __toString( ) method is currently only called by the print and echo language constructs.

   (*a*) False (*b*) True

8.12 Review Questions

1. What is data handling?
2. How do we retrieving date and time information?
3. How we format date and time?
4. Define parsing date formats?
5. What are the strings with PHP?
6. Define accessing string offsets.

Answers for Self Assessment Questions

Fill in the blanks

1. UNIX timestamp	2. formatting text
3. strings	4. PHP

True or False

1. (*a*)	2. (*a*)
3. (*a*)	4. (*b*)

8.13 Further Reading

Books *PHP: A Beginner's Guide by: Vaswani, Vikram*, By Tata Mc-Graw Hill.

Online link http://www.gibmonks.com/c_plus/ch11lev1sec10.html/

Unit 9: Working with Forms

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain creating simple input form.
- Understand creating the form.
- Discuss accessing form input with user-defined arrays.
- Explain combining HTML and PHP code on a single page.
- Discuss hidden fields to save state.
- Understand redirecting the user.
- Discuss sending mail on form submission.
- Explain working with file uploads.
- Explain creating file upload form.

Introduction

Until now, the PHP examples in this book have been missing a crucial dimension. Sure, you know the basics, can set variables and arrays, create and call functions, and connect to MySQL to do great things with a database. But that’s all meaningless if users can’t reach into a language’s environment to offer it information. In this, you look at strategies for acquiring and working with user input. On the World Wide Web, HTML forms are the principal means by which substantial amounts of information pass from the user to the server. In this you will learn:

- How to access information from form fields?
- How to work with form elements that allow multiple selections?
- How to create a single document that contains both an HTML form and the PHP code that handles its submission?
- How to save state with hidden fields?
- How to redirect the user to a new page?
- How to build HTML forms and PHP code that send mail?
- How to build HTML forms that upload files and how to write the PHP code to handle them?

9.1 Creating a Simple Input Form

For now, let’s keep our HTML separate from our PHP code. Listing 1 builds a simple HTML form.

Listing 1 A Simple HTML Form

```
1.  <html>
2.  <head>
3.  <title>Listing 1 A simple HTML form</title>
4.  </head>
5.  <body>
6.  <form action=”listing2.php” method=”POST”>
7.  Name: <br>
8.  <input type=”text” name=”user”>
9.  <br>
10. Address: <br>
11. <textarea name=”address” rows=”5” cols=”40”></textarea>
12. <br>
13. <input type=”submit” value=”hit it!”>
14. </form>
15. </body>
16. </html>
```

Put these lines into a text file called listing 1, and place that file in your Web server document root. This listing defines a form that contains a text field with the name “user” on line 8, a text area with the name “address” on line 11, and a submit button on line 13. The FORM element’s

Notes

ACTION argument points to a file called listing 2.php, which processes the form information. The method of this form is POST, so the variables are stored in the $_POST superglobal.

Listing 2 creates the code that receives our users' input.

Listing 2 Reading Input from the Form in Listing 1

```
<html>
<head>
<title>Listing 2 Reading input from the form in Listing  1</title>
</head>
<body>
<?php
print "Welcome <b>$_POST[user]</b><P>\n\n";
print "Your address is:<P>\n\n<b>$_POST[address]</b>";
?>
</body>
</html>
```

Put these lines into a text file called listing 2.php, and place that file in your Web server document root.

9.2 Creating the Form

In Listing 3, you see the basic HTML for creating a simple feedback form. This form has an action of listing12.php, which we create in the next section. The fields are very simple: Line 7 contains a name field, line 8 contains the return email address field, and line 10 contains the text area for the user's message.

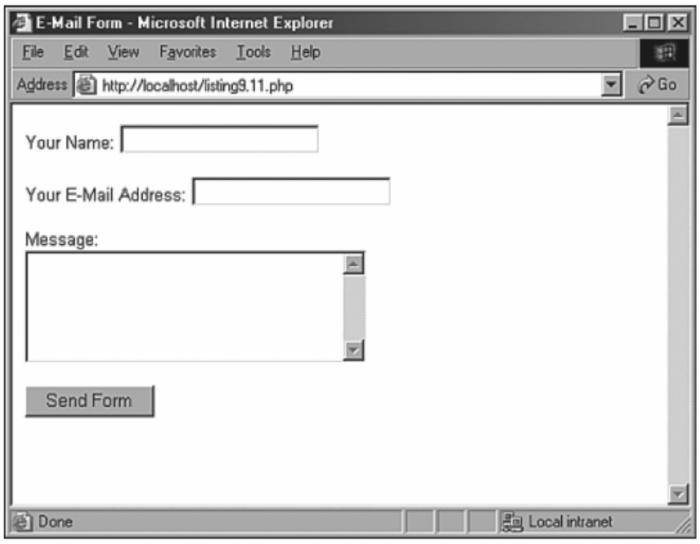
Listing 3 Creating a Simple Feedback Form

```
1.  <HTML>
2.  <HEAD>
3.  <TITLE>E-Mail Form</TITLE>
4.  </HEAD>
5.  <BODY>
6.  <FORM action="listing12.php" method="POST">
7.  Your Name: <INPUT type="text" name="name"><br><br>
8.  Your E-Mail Address: <INPUT type="text" name="email"><br><br>
9.  Message:<br>
10. <textarea name="message" cols=30 rows=5></textarea><br><br>
11. <INPUT type="submit" value="Send Form">
```

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

Put these lines into a text file called listing3.php, and place this file in your Web server document root. Now access the script with your Web browser, and you should see something like Figure 9.1.

Figure 9.1: Form Created in Listing 3


E-Mail Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/listing9.11.php
Go
Your Name:
Your E-Mail Address:
Message:
Send Form
Done
Local intranet

In the next section, you create the script that sends this form to a recipient.

Task Create simple feedback form.

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, like so <select name="products" multiple> the script that receives this data has access to only a single value corresponding to this name. We can change this behavior by renaming an element of this kind so that its name ends with an empty set of square brackets. We do this in Listing 4.

Listing 4 An HTML Form Including a SELECT Element

```
<html>
<head>
<title>Listing 4 An HTML form including a SELECT element</title>
</head>
<body>
<form action="listing5.php" method="POST">
```

Notes

```
7. Name: <br>
8. <input type=”text” name=”user”>
9. <br>
10. Address: <br>
11. <textarea name=”address” rows=”5” cols=”40”></textarea>
12. <br>
13. Pick Products: <br>
14. <select name=”products[]” multiple>
15. <option>Sonic Screwdriver</option>
16. <option>Tricorder</option>
17. <option>ORAC AI</option>
18. <option>HAL 2000</option>
19. </select>
20. <br><br>
21. <input type=”submit” value=”hit it!”>
22. </form>
23. </body>
24. </html>
```

Put these lines into a text file called listing4.php, and place that file in your Web server document root. Next, in the script that processes the form input, we find that input from the “products[]” form element created on line 14 is available in an array called $_POST[products]. Because products[] is a SELECT element, we offer the user multiple choices using the option elements on lines 15 through 18. We demonstrate that the user’s choices are made available in an array in Listing 5.

Listing 5 Reading Input from the Form in Listing 4

```
1. <html>
2. <head>
3. <title>Listing 5 Reading input from the form in Listing 6</title>
4. </head>
5. <body>
6. <?php
7. print “Welcome <b>$_POST[user]</b><p>\n\n”;
8. print “Your address is:<p>\n\n<b>$_POST[address]</b><p>\n\n”;
9. print “Your product choices are:<p>\n\n”;
10. if (!empty($_POST[products])) {
```

```
print "<ul>\n\n";
?>
</body>
</html>
```

9.4 Combining HTML and PHP Code on a Single Page

In some circumstances, you might want to include form-parsing code on the same page as a hard- coded HTML form. Such a combination can be useful if you need to present the same form to the user more than once. You would have more flexibility if you were to write the entire page dynamically, of course, but you would miss out on one of the great strengths of PHP. The more standard HTML you can leave in your pages, the easier they are for designers and page builders to amend without reference to you. You should avoid scattering substantial chunks of PHP code throughout your documents, however. Doing so makes them hard to read and maintain. Where possible, you should create functions that can be called from within your HTML code and can be reused in other projects.

For the following examples, imagine that we're creating a site that teaches basic math to preschool children, and have been asked to create a script that takes a number from form input and tells the user whether it's larger or smaller than a predefined integer.

Listing 6 creates the HTML. For this example, we need only a single text field, but even so, we'll include a little PHP.

Listing 6 An HTML Form That Calls Itself

```
<html>
<head>
<title>Listing 6 An HTML form that calls itself</title>
</head>
<body>
<form action="< ?php print $_SERVER[PHP_SELF] ? >" method="POST">
Type your guess here: <input type="text" name="guess">
</form>
</body>
</html>
```

The action of this script is $_SERVER[PHP_SELF]. This variable is the equivalent of the name of the current script. In other words, the action tells the script to reload itself.

The script in Listing 6 doesn't produce any output. In Listing 7, we begin to build up the PHP element of the page. First, we must define the number that the user guesses. In a fully working version, we'd probably randomly generate this number, but for now, we keep it simple. We assign 42 to the $num_to_guess variable on line 2. Next, we must determine whether the form has been submitted; otherwise, we'd attempt to assess variables that aren't yet made available. We can test for submission by testing for the existence of the variable $_POST[guess], which is made available if your script has been sent a "guess" parameter. If $_POST[guess] isn't present, we can safely assume that the user arrived at the page without submitting a form. If the value is present, we can test the value it contains. The test for the presence of the $_POST[guess] variable takes place on line 4.

Listing 7 A PHP Number-Guessing Script

```
1. < ?php
2. $num_to_guess = 42;
3. $message = "";
4. if (!isset($_POST[guess])) {
5. $message = "Welcome to the guessing machine!";
6. } elseif ($_POST[guess] > $num_to_guess) {
7. $message = "$_POST[guess] is too big! Try a smaller number";
8. } elseif ($_POST[guess] < $num_to_guess) {
9. $message = "$_POST[guess] is too small! Try a larger number";
10. } else { // must be equivalent
11. ?>
```

9.5 Using Hidden Fields to Save State

The script in Listing 7 has no way of knowing how many guesses a user has made, but we can use a hidden field to keep track of this. A hidden field behaves exactly the same as a text field, except that the user cannot see it unless he views the HTML source of the document that contains it. Listing 8 adds a hidden field to the number-guessing script and some PHP to work with it.

Listing 8 Saving State with a Hidden Field

```
1. < ?php
2. $num_to_guess = 42;
3. $num_tries = (isset($_POST[num_tries])) ? $num_tries + 1 : 0;
4. $message = "";
5. if (!isset($_POST[guess])) {
6. $message = "Welcome to the guessing machine!";
7. } elseif ($_POST[guess] > $num_to_guess) {
8. $message = "$_POST[guess] is too big! Try a smaller number";
9. } elseif ($_POST[guess] < $num_to_guess) {
10. $message = "$_POST[guess] is too small! Try a larger number";
11. } else { // must be equivalent
12. $message = "Well done!";
```

Notes

```
13. }
14. $guess = $_POST[guess];
15. ?>
16. <html>
17. <head>
18. <title>Listing 8 Saving state with a hidden field</title>
19. </head>
20. <body>
21. <h1>
22. < ?php print $message ? >
23. </h1>
24. Guess number: < ?php print $num_tries? >
25. <form action="< ?php print $_SERVER[PHP_SELF] ?>" method="POST" >
26. Type your guess here:
27. <input type="text" name="guess" value="< ?php print $guess? >">
28. <input type="hidden" name="num_tries" value="< ?php print $num_tries? >">
29. </form>
30. </body>
31. </html>
```

The hidden field on line 28 is given the name "num_tries". We also use PHP to write its value. While we're at it, we do the same for the "guess" field on line 27 so that the user can always see his last guess. This technique is useful for scripts that parse user input. If we reject a form submission for some reason, we can at least allow our user to edit his previous query.

Caution Be sure that absolutely no output has been sent to the browser. The first time content is sent to the browser, PHP sends out headers and it's too late for you to send your own. Any output from your document, even a line break or a space outside of your script tags, causes headers to be sent. If you intend to use the header() function in a script, you must make certain that nothing precedes the PHP code that contains the function call. You should also check any libraries that you might be using.

Listing 9 shows typical headers sent to the browser by PHP, beginning with line 3, in response to the request in line.

9.6 Redirecting the User

Our simple script still has one major drawback. The form is rewritten whether or not the user guesses correctly. The fact that the HTML is hard-coded makes it difficult to avoid writing the entire page. We can, however, redirect the user to a congratulations page, thereby sidestepping the issue altogether. When a server script communicates with a client, it must first send some headers that provide information about the document to follow. PHP usually handles this for you automatically, but you can choose to send your own header lines with PHP's header() function.

To call the header() function, you must be sure that absolutely no output has been sent to the browser. The first time content is sent to the browser, PHP sends out headers and it's too late for you to send your own. Any output from your document, even a line break or a space outside of your script tags, causes headers to be sent. If you intend to use the header() function in a script, you must make certain that nothing precedes the PHP code that contains the function call. You should also check any libraries that you might be using. Listing 9 shows typical headers sent to the browser by PHP, beginning with line 3, in response to the request in line.

Listing 9 Typical Server Headers Sent from a PHP Script

```
1. HEAD/listing9.php
2. HTTP/1.0
3. HTTP/1.1 200 OK
4. Date: Sun, 15 Sep 2002 12 : 32 : 28 GMT
5. Server: Apache/2.0.43 (Unix) PHP/4.2.3 mod_ssl/2.8.9 OpenSSL/0.6
6. X-Powered-By: PHP/4.2.3
7. Connection: close
8. Content-Type: text/html
```

By sending a "Location" header instead of PHP's default, you can cause the browser to be redirected to a new page:

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

Assuming that we've created a suitably upbeat page called "congrats.html", we can amend our number-guessing script to redirect the user if she guesses correctly, as shown in Listing 10.

Listing 10 Using header() to Send Raw Headers

```
1. <?php
2. $num_to_guess = 42;
3. $num_tries = (isset($_POST[num_tries])) ? $num_tries + 1: 0;
4. $message = "";
5. if (!isset($_POST[guess])) {
6. $message = "Welcome to the guessing machine!";
7. } elseif ($_POST[guess] > $num_to_guess) {
8. $message = "$_POST[guess] is too big! Try a smaller number";
9. } elseif ($_POST[guess] < $num_to_guess) {
10. $message = "$_POST[guess] is too small! Try a larger number";
```

Notes

```
} else { // must be equivalent
header("Location: congrats.html");
exit;
}
?>
```

9.7 Sending Mail on Form Submission

You've already seen how to take form responses and print the results to the screen. You're only one step away from sending those responses in an email message, as you'll soon see. Before learning about sending mail, however, read through the next section to make sure that your system is properly configured.

9.7.1 System Configuration for the Mail() Function

Before you can use the mail() function to send mail, a few directives must be set up in the php.ini file so that the function works properly. Open php.ini with a text editor and look for these lines:

```
[mail function]
; For Win32 only.
SMTP = localhost
; For Win32 only.
sendmail_from = me@localhost.com
; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
;sendmail_path =
```

If you're using Windows as your Web server platform, the first two directives apply to you. For the mail() function to send mail, it must be able to access a valid outgoing mail server. If you plan to use the outgoing mail server of your ISP (in the following example, we use EarthLink), the entry in php.ini should look like this:

```
SMTP = mail.earthlink.net
```

The second configuration directive is sendmail_from, which is the email address used in the From header of the outgoing email. It can be overwritten in the mail script itself, but normally operates as the default value. For example:

```
sendmail_from = youraddress@yourdomain.com
```

A good rule of thumb for Windows users is that whatever outgoing mail server you've set up in your email client on that machine, you should also use as the value of SMTP in php.ini. If your Web server is running on a Linux/Unix platform, you use the sendmail functionality of that particular machine.

In this case, only the last directive applies to you: sendmail_path. The default is sendmail -t -i, but if sendmail is in an odd place or if you need to specify different arguments, feel free to do so, as in the following example:

```
sendmail_path = /opt/sendmail -odd -arguments
```

After making any changes to php.ini on any platform, you must restart the Web server process for the changes to take effect.

9.7.2 Creating the Script to Send the Mail

This script is only slightly different in concept than the script in Listing 5, which simply printed form responses to the screen. In this script, in addition to printing the responses to the screen, you send them to an email address as well.

Listing 12 Sending the Simple Feedback Form

```
1. <html>
2. <head>
3. <title>Listing 12 Sending mail from the form in Listing 11</title>
4. </head>
5. <body>
6. <?php
7. print “Thank you, <b>$_POST[name]</b>, for your message!<br><br>\n\n”;
8. print “Your e-mail address is: <b>$_POST[email]</b><br><br>\n\n”;
9. print “Your message was:<br><br>\n\n”;
10. print “$_POST[message] <br><br>”;
11. //start building the mail string
12. $msg = “Name: $_POST[name]\n”;
13. $msg .= “E-Mail: $_POST[email]\n”;
14. $msg .= “Message: $_POST[message]\n”;
15. //set up the mail
16. $recipient = “you@yourdomain.com”;
17. $subject = “Form Submission Results”;
18. $mailheaders = “From: My Web Site <defaultaddress@yourdomain.com> \n”;
19. $mailheaders .= “Reply-To: $_POST[email]”;
20. //send the mail
21. mail($recipient, $subject, $msg, $mailheaders);
22. ?>
23. </body>
24. </html>
```

The variables you use in lines 7-9 are $_POST[name], $_POST[email], and $_POST[message]—the names of the fields in the form, as part of the $_POST superglobal. That’s all well and good for printing the information to the screen, but in this script, you also want to create a string that’s sent in email. For this task, you essentially build the email by concatenating strings to form one long message string, using the newline (\n) character to add line breaks where appropriate.

Lines 12 through 14 create the $msg string, which contains the values typed by the user in the form fields. This string is the one sent in the email. Note the use of the concatenation operator (.=) when adding to the variable $msg, in lines 13 and 14.

Lines 16 and 17 are hard-coded variables for the email recipient and the subject of the email message. Replace you@yourdomain.com with your own email address, obviously. If you want to change the subject, feel free! Lines 18 and 19 set up some mail headers, namely From: and Reply-to: headers. You could put any value in the From: header; this is the information that displays in the From or Sender column of your email application when you receive this mail.

The mail() function takes four parameters: the recipient, the subject, the message, and any additional mail headers.

The order of these parameters is shown in line 21, and your script is complete after you close up your PHP block and your HTML elements in lines 22-24.

9.8 Working with File Uploads

So far, we've looked at simple form input. However, all popular Web browsers support file uploads, and so, of course, does PHP. In this section, you examine the features that PHP makes available to deal with this kind of input. Information about the uploaded file becomes available to you in the $_FILES superglobal, which is indexed by the name of the upload field (or fields) in the form. The corresponding value for each of these keys is an associative array. These fields are described in Table 9.1, using fileupload as the name of the form field used for the upload.

Table 9.1: File Upload Global Variables

Element	Contains	Example
$_FILES['fileupload'] ['name']	Original name of uploaded file	test.gif
$_FILES['fileupload'] ['tmp_name']	Path to temporary file	/tmp/phprDfZvN
$_FILES['fileupload'] ['size']	Size (in bytes) of uploaded file	6835
$_FILES['fileupload'] ['type']	MIME type of uploaded file (where given by client)	image/gif

Caution Keep these elements in the back of your mind for a moment, while we create the upload form in the next section.

9.9 Creating the File Upload Form

First, we must create the HTML form to handle the upload. HTML forms that include file upload fields must include an ENCTYPE argument:

ENCTYPE="multipart/form-data"

PHP also works with an optional hidden field that can be inserted before the file upload field. This field must be called MAX_FILE_SIZE and should have a value representing the maximum size in bytes of the file that you're willing to accept. This size cannot override the maximum size set in the upload_max_filesize field in your php.ini file that defaults to 2MB. The MAX_FILE_SIZE field is obeyed at the browser's discretion, so you should rely on the php.ini setting to cap unreasonable uploads. After the MAX_FILE_SIZE field has been entered, you're ready to add the upload field itself. This is simply an INPUT element with a TYPE argument of "file". You can give it any name you want. Listing 13 brings all this together into an HTML upload form.

Listing 13 A Simple File Upload Form

```
<?php

$uploaddir = '/var/www/uploads/';

$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);

echo "<p>";

if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile))

{

  echo "File is valid, and was successfully uploaded.\n";

}

else

{

  echo "Upload failed";

}

echo "</p>";

echo '<pre>';

echo 'Here is some more debugging info:';

print_r($_FILES);

print "</pre>";

?>
```

Lab Exercise

1. Create email to send script.
2. Create the File Upload Form

9.10 Summary

- The more standard HTML you can leave in your pages, the easier they are for designers and page builders to amend without reference to you.
- A hidden field behaves exactly the same as a text field, except that the user cannot see it unless he views the HTML source of the document that contains it.
- After making any changes to php.ini on any platform, you must restart the web server process for the changes to take effect.

- PHP also works with an optional hidden field that can be inserted before the file upload field.
- The mail() function takes four parameters: The recipient, the subject, the message, and any additional mail headers.

9.11 Keywords

***FORM Elements*:** FORM elements The FORM element's ACTION argument points to a file called listing2.php, which processes the form information. The method of this form is POST, so the variables are stored in the $_POST superglobal.

***HTML and PHP Code on a Single Page*:** In some circumstances, you might want to include form parsing code on the same page as a hard-coded HTML form. Such a combination can be useful if you need to present the same form to the user more than once.

***Hidden Fields to Save State*:** A hidden field behaves exactly the same as a text field, except that the user cannot see it unless he views the HTML source of the document that contains it.

***Sending Mail on Form Submission*:** This is the form that the submitter fills in; submitting this form causes **form-submit** to email a message based on the information filled in here. This form asks for the submitter's name, email address, and some comments.

System Configuration for the mail() Function: The mail() function allows you to send emails directly from a script.

Working with File Uploads: Information about the uploaded file becomes available to you in the $_FILES superglobal, which is indexed by the name of the upload field (or fields) in the form.

File Upload Form: This field must be called MAX_FILE_SIZE and should have a value representing the maximum size in bytes of the file that you're willing to accept.

9.12 Self Assessment Questions

State True or False:

1. The Form elements action argument points to a file called listing 2.php.

   (*a*) True (*b*) False

2. Form-passing code can be useful if you need to present the same form to the user more than once.

   (*a*) False (*b*) True

3. Functions can be called from within your HTML code and can be reused in other projects.

   (*a*) True (*b*) False

4. Making no changes to php.ini on any platform, you must restart the web server process.

   (*a*) True (*b*) False

Fill in the blanks:

1. Impart from the "products[] from elements created on line 14 is available in an array called ________________.

Notes

2. HTML is hard-coded makes it difficult to avoid ________________.
3. By sending a ________________ header instead of PHP's default, you can cause the browser to be redirected to a new page.
4. If the Webserver is running on a ________________ platform, you use the sendmail functionality of that particular machine.

9.13 Review Questions

1. Define single input form. Give steps how to create it.
2. Give steps to create form.
3. What are user defined arrays?
4. What are HTML and PHP code? How do we combine on single phase?
5. How do we use hidden files to save state?
6. What is redirection of user?
7. What is Sending Mail on Form Submission?
8. Give proper system configuration for mail function().
9. How to create script to send mail?
10. What are file uploads? Give its working also.
11. Explain how to create file upload form.

Answers for Self Assessment Questions

True or False:

1. (*a*)	2. (*b*)
3. (*a*)	4. (*b*)

Fill in the blanks:

1. $_Post [product]	2. Writing the entire page
3. Location	4. Linus/Unix

9.14 Further Reading

Books *Open source development with LAMP*, By Meloni, Pearson education.

Online link http://www.htmlcenter.com/blog/teach-yourself-php-in-24-hours/

Unit 10: Cookies

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain cookies.
- Discuss setting cookies.
- Discuss cookies with PHP.
- Understand session function overview.
- Explain destroying session and unsetting variables.

Introduction

A cookie is an information that a Web site puts on your hard disk so that it can remember something about you at a later time. (More technically, it is an information for future use that is stored by the server on the client side of a client/server communication.) Typically, a cookie records your preferences when using a particular site. Using the Web's Hypertext Transfer Protocol (HTTP), each request for a Web page is independent of all other requests. For this reason, the Web page server has no memory of what pages it has sent to a user previously or

Notes

anything about your previous visits. A cookie is a mechanism that allows the server to store its own information about a user on the user's own computer. You can view the cookies that have been stored on your hard disk (although the content stored in each cookie may not make much sense to you). The location of the cookies depends on the browser. Internet Explorer stores each cookie as a separate file under a Windows subdirectory. Netscape stores all cookies in a single cookies.txt. Opera stores them in a single cookies.dat file.

10.1 Cookies

Cookies are commonly used to rotate the banner ads that a site sends so that it doesn't keep sending the same ad as it sends you a succession of requested pages. They can also be used to customize pages for you based on your browser type or other information you may have provided the Web site. Web users must agree to let cookies be saved for them, but, in general, it helps Web sites to serve users better. A server can set as many as 20 cookies, and each of these cookies can be up to 4 KB in size.

The simple registration we used earlier in this chapter does not make data persistent across requests. If you go to the next page (such as by clicking a link or by entering a different URL in your browser's address bar), the posted data is gone. One simple way to maintain data between the different pages in a web application is with cookies. Cookies are sent by PHP through the web servermn with the setcookie() function and are stored in the browser. If a time-out is set for the cookie, the browser will even remember the cookie when you reset your computer; without the time-out set, the browser forgets the cookie as soon as the browser closes. You can also set a cookie to be valid only for a specific subdomain, rather than having the cookie sent by the browser to the script whenever the domain of the script is the same as the domain where the cookie was set (the default). In the next example, we set a cookie when a user has successfully logged in with the login form

```
<?php

ob_start();

?>

<html>

<head><title>Login</title></head>

<body>

<?php

if (isset ($_POST['login']) && ($_POST['login'] == 'Log in') &&

($uid = check_auth($_POST['email'], $_POST['password'])))

{

/* User successfully logged in, setting cookie */

setcookie('uid', $uid, time() + 14400, '/');

header('Location: http://kossu/crap/0x-examples/index.php');

exit();
} else {
?>
<h1>Log-in</h1>
<form method="post" action="login.php">
<table>
<tr><td>E-mail address:</td>
<td><input type='text' name='email'/></td></tr>
<tr><td>Password:</td>
<td><input type='password' name='password'/></td></tr>
<tr><td colspan='2'>
<input type='submit' name='login' value='Log in'/></td>
</tr>
</table>
</form>
<?php
}
?>
</body>
```

The check_auth() function checks whether the username and password match with the stored data and returns either the user id that belongs to the\ user or 0 when an error occurred. The setcookie('uid', $uid, time() 14400, '/'); line tells the web server to add a cookie header to send to the browser. uid is the name of cookie to be set and $uid has the value of the uid cookie. The expression time() + 14400 sets the expiry time of the cookie to the current time plus 14,400 seconds, which is 4 hours. The time on the server must be correct because the time() function is the base for calculating the expiry time. Notice that the ob_start() function is the first line of the script. ob_start() turns on output buffering, which is needed to send cookies (or other headers) after you output data. Without this call to ob_start(), the output to the browser would have started at the <html> line of the script, making it impossible to send any headers, and resulting in the following error when trying to add another header (with setcookie() or header()): Instead of using output buffering (which is memory-intensive), you can, of course, change your script so that data is not output until after you set any headers. Cookies are sent by the script/web server to the browser. The browser is then responsible for sending the cookie, via HTTP request headers, to all successive pages that belong to your web application. With the third and fourth parameters of the setcookie() function, you can control which sections of your web site receive the specific cookie headers. The third parameter is /, which means that all pages in the domain (the root and all subdirectories) should receive the cookie data. The fourth parameter controls which domains receive the cookie header. For instance, if you use .example.com, the cookie is available to all subdomains of example.com. Or, you could use admin.example.com, restricting the cookies to the admin part of your application. In this case, we did not specify a domain, so all pages in the web application receive the cookie. After the line with the setcookie() call, a line issues a redirect header to the browser. This header requires the full path to the destination page. After the header line, we terminate the script with exit() so that no headers can be set from later parts of the code. The browser redirects to the given URL by requesting the new page and discarding the content of the current one. On any web page requested after the script that called set_cookie(), the cookie data is available in your script in a manner similar to the GET and POST data. The superglobal to read cookies is $_COOKIE. The following index.php script shows the use of cookies to authenticate a user. The first line of the page checks whether the cookie with the user id is set. If it's set, we display our index.php page, echoing the user id set in the cookie. If it's not set, we redirect to the login page:

```
<?php
if (isset ($_COOKIE['uid']) && $_COOKIE['uid']) {
?>
<html>
<head><title>Index page</title></head>
<body>
Logged in with UID: <?php echo $_COOKIE['uid']; ?><br />
<a href='logout.php'>Log out</a>.
</body>
</html>
<?php
} else {
/* If no UID is in the cookie, we redirect to the login
→page */
header('Location: http://kossu/examples/login.php');
}
?>
```

Using this user id for important items, such as remembering authentication data (as we do in this script), is not wise, because it's easy to fake cookies. (For most browsers, it is enough to edit a simple text field.) A better solution—

using PHP sessions—follows in a bit.

Deleting a cookie is almost the same as setting one. To delete it, you use the same parameters that you used when you set the cookie, except for the value, which needs to be an empty string, and the expiry date, which needs to be set in the past. On our logout page, we delete the cookie this way:

```
<?php
setcookie('uid', '', time() - 86400, '/');
```

```
header('Location: http://kossu/examples/login.php');

?>
```

The time() - 86400 is exactly one day ago, which is sufficiently in the past for our browser to forget the cookie data. As previously mentioned, putting authentication data into cookies (as we did in the previous examples) is not secure because cookies are so easily faked. PHP has, of course, a better solution: sessions.

10.2 Setting Cookies

To create and modify a cookie, use the PHP functionsetcookie(). setcookie() takes up to six arguments, depending upon how much control you want over the cookie and who can read its value.

The simplest way of setting a cookie is:

setcookie('name', 'bret');

Then, for every further page on your site viewed by this browser (without the user quitting) you'll have the value of 'bret' stored in the variable $name for easy access in PHP. This type of cookie is known as a session cookie, since it lasts for the length of a user's session.

If you want the cookie to persist after the person exits his or her browser, you must passsetcookie() through a third parameter, the date you want the cookie to expire. Since PHP's background springs fully formed from the head of UNIX, you represent this time as the number of seconds since January 1, 1970. If you're a UNIX programmer, this makes complete sense. But, if you're from a Windows or a Macintosh background, you're just shaking your head wondering if you'll ever understand those wacky UNIX folk.

The main difference between a cookie and a session is that a cookie is stored on your computer, and a session is not. Although cookies have been around for many years and most people do have them enabled, there are some who do not. Cookies can also be removed by the user at any time, so don't use them to store anything too important.

A cookie is set with the following code: **setcookie(name, value, expiration)**

10.3 Deleting Cookies with PHP

PHP, or Hypertext Preprocessor, is an open-source scripting language primarily used for web programming. PHP code can be embedded into normal HTML code. A cookie is a web file that is used by a server to identify a user of that server. PHP is able to create cookies, retrieve cookie values and delete cookies.

Deleting a cookie is almost the same as setting one. To delete it, you use the same parameters that you used when you set the cookie, except for the value, which needs to be an empty string, and the expiry date, which needs to be set in the past. On our logout page, we delete the cookie this way:

```
<?php

       setcookie('uid', '', time() - 86400, '/');

       header('Location: http://kossu/examples/login.php');

?>
```

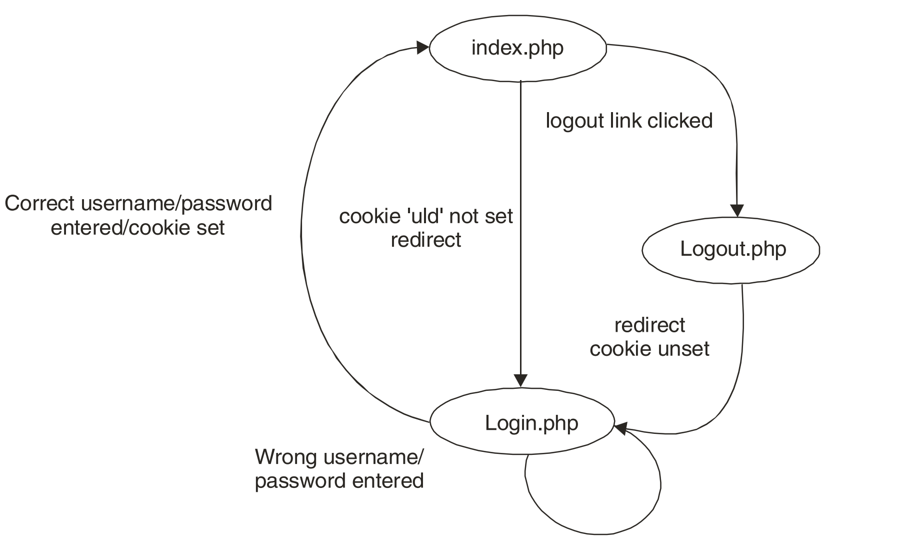
Notes

The time() - 86400 is exactly one day ago, which is sufficiently in the past for our browser to forget the cookie data.

Figure 10.1 shows the way our scripts can be tied together. As mentioned, putting authentication data into cookies (as we did in the previous examples) is not secure because cookies are so easily faked.

PHP has, of course, a better solution: sessions.

Figure 10.1: Scripts Tied Together.


index.php
logout link clicked
Correct username/password entered/cookie set
cookie 'uld' not set redirect
Logout.php
redirect cookie unset
Login.php
Wrong username/ password entered

10.4 Session Function Overview

Session functions provide a unique identifier to a user, which can then be used to store and acquire information linked to that ID. When a visitor accesses a session-enabled page, she is either allocated a new identifier or re-associated with one that was already established in a previous access. Any variables that have been associated with the session will become available to your code through the $_SESSION superglobal.

When you use sessions, cookies are used by default to store the session identifier, but you can ensure success for all clients by encoding the session ID into all links in your session-enabled pages.

Session state is usually stored in a temporary file, although you can implement database storage using a function called session_set_save_handler().

10.4.1 Starting Session

To work with a session, you need to explicitly start or resume that session unless you have changed your php.ini configuration file. By default, sessions do not start automatically. If you want to start a session this way, you will have to find the following line in your php.ini file and change the value from 0 to 1 (and restart the Web server):

```
session.auto_start = 0
```

Notes

By changing the value of `session.auto_start` to `1`, you ensure that a session is initiated for every PHP document. If you don't change this setting, you need to call the `session_start()` function in each script.

After a session is started, you instantly have access to the user's session ID via the `session_id()` function. `session_id()` allows you to either set or get a session ID. Listing 10.1 starts a session and prints the session ID to the browser.

Example: 1. Starting or Resuming a Session

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>Listing 10.1 Starting or resuming a session</title>
 7: </head>
 8: <body>
 9: <?php
10: echo "<p>Your session ID is ".session_id()."</p>";
11: ?>
12: </body>
13: </html>
```

When this script is run for the first time from a browser, a session ID is generated by the `session_start()` function call on line 2. If the page is later reloaded or revisited, the same session ID is allocated to the user. This action assumes that the user has cookies enabled. For example, when I run this script the first time, the output is

```
Your session ID is fa963e3e49186764b0218e82d050de7b
```

When I reload the page, the output is still

```
Your session ID is fa963e3e49186764b0218e82d050de7b
```

because I have cookies enabled and the session ID still exists.

Because `start_session()` attempts to set a cookie when initiating a session for the first time, it is imperative that you call this function before you output anything else at all to the browser. If you do not follow this rule, your session will not be set, and you will likely see warnings on your page.

Sessions remain current as long as the Web browser is active. When the user restarts the browser, the cookie is no longer stored. You can change this behavior by altering the `session.cookie_lifetime` setting in your `php.ini file`. The default value is 0, but you can set an expiry period in seconds.

Accessing a unique session identifier in each of your PHP documents is only the start of session functionality. When a session is started, you can store any number of variables in the `$_SESSION` superglobal and then access them on any session-enabled page.

Notes

10.4.2 Working with Session Variables

If you are using a pre-4.1.x version of PHP, the `$_SESSION` superglobal is not present, and session functionality is much different.

Example: 2: Add two variables into the `$_SESSION` superglobal: `product1` and `product2` (lines 10 and 11).

Storing Variables in a Session

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>Listing 10.2 Storing variables in a session</ title>
 7: </head>
 8: <body>
 9: <?php
10: $_SESSION[product1] = "Sonic Screwdriver";
11: $_SESSION[product2] = "HAL 2000";
12: echo "The products have been registered.";
13: ?>
14: </body>
15: </html>
```

Creates a separate PHP script that accesses the variables stored in the `$_SESSION` superglobal in example2.

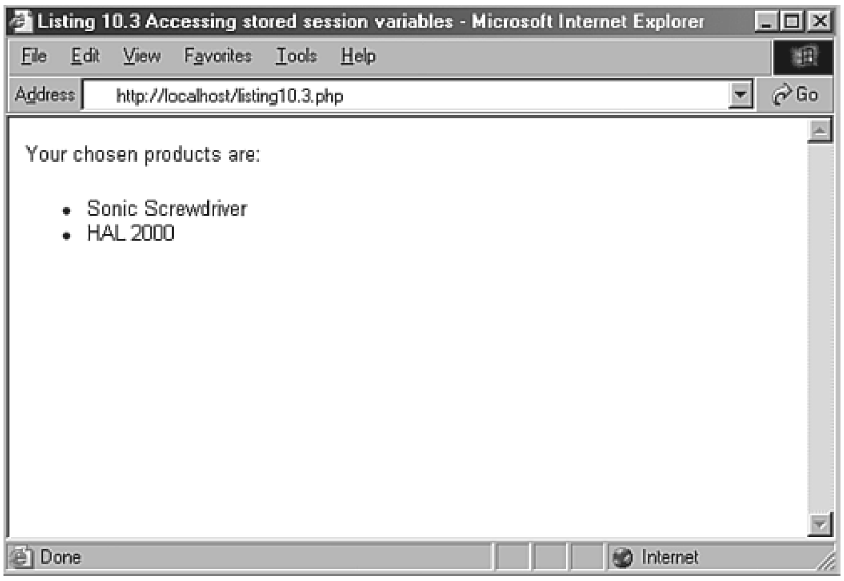
Example: 3: Accessing Stored Session Variables

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>Listing 10.3 Accessing stored session
   variables</title>
7: </head>
8: <body>
9: <?php
10: echo "Your chosen products are:";
11: echo "<ul><li>$_SESSION[product1] <li>$_SESSION
    [product2]\n</ul>\n";
```

```
?>
</body>
</html>
```

Figure 10.2 shows the output from example:3. As you can see, we have access to the `$_SESSION[product1]` and `$_SESSION[product2]` variables in an entirely new page.

Figure 10.2: Accessing Stored Session Variables


Listing 10.3 Accessing stored session variables - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/listing10.3.php
Go
Done
Internet

Your chosen products are:

- Sonic Screwdriver
- HAL 2000

Although not a terribly interesting or useful example, the script does show how to access stored session variables. Behind the scenes, PHP writes information to a temporary file. You can find out where this file is being written on your system by using the session_save_path() function. This function optionally accepts a path to a directory and then writes all session files to it. If you pass it no arguments, it returns a string representing the current directory to which session files are saved. On my system,

```
echo session_save_path();
```

prints /tmp. A glance at my /tmp directory reveals a number of files with names like the following:

```
sess_fa963e3e49186764b0218e82d050de7b

sess_76cae8ac1231b11afa2c69935c11dd95

sess_bb50771a769c605ab77424d59c784ea0
```

Opening the file that matches the session ID I was allocated when I first ran example1, I can see how the registered variables have been stored:

```
product1|s:17:"Sonic Screwdriver";product2|s:8:"HAL 2000";
```

When a value is placed in the `$_SESSION` superglobal, PHP writes the variable name and value to a file. This information can be read and the variables resurrected later—as you have already seen. After you add a variable to the `$_SESSION` superglobal, you can still change its value at any time during the execution of your script, but the altered value won't be reflected in the global setting until you reassign the variable to the `$_SESSION` superglobal.

In example 2: demonstrates the process of adding variables to the `$_SESSION` superglobal. This example is not very flexible, however. Ideally, you should be able to register a varying number of values. You might want to let users pick products from a list, for example. In this case, you can use the serialize() function to store an array in your session.

Example: 4: Creates a form that allows a user to choose multiple products. You should then be able to use session variables to create a rudimentary shopping cart. Adding an Array Variable to a Session Variable

```
<?php
session_start();
?>
<html>
<head>
<title>Listing 10.4 Storing an array with a session</title>
</head>
<body>
<h1>Product Choice Page</h1>
<?php
if (isset($_POST[form_products])) {
 if (!empty($_SESSION[products])) {
   $products = array_unique(
   array_merge(unserialize($_SESSION[products]),
   $_POST[form_products]));
$_SESSION[products] = serialize($products);
} else {
$_SESSION[products] = serialize($_POST[form_products]);
}
echo "<p>Your products have been registered!</p>";
}
?>
<form method="POST" action="<?php $_SERVER[PHP_SELF] ?>">
<P><strong>Select some products:</strong><br>
<select name="form_products[]" multiple size=3>
<option value="Sonic Screwdriver">Sonic Screwdriver</option>
```

Notes

```
27: <option value=”Hal 2000”>Hal 2000</option>
28: <option value=”Tardis”>Tardis</option>
29: <option value=”ORAC”>ORAC</option>
30: <option value=”Transporter bracelet”>Transporter
    bracelet</option>
31: </select>
32:
33: <P><input type=”submit” value=”choose”></p>
34: </form>
35:
36: <p><a href=”listing10.5.php”>go to content page</a></p>
37: </body>
38: </html>
```

We start or resume a session by calling `session_start()` on line 2. This should give us access to any previously set session variables. We begin an HTML form on line 23 and, on line 25, create a `SELECT` element named `form_products[]`, which contains OPTION elements for a number of products. Remember that HTML form elements that allow multiple selections should have square brackets appended to the value of their NAME arguments. This makes the user’s choices available in an array.

Within the block of PHP code beginning on line 10, we test for the presence of the `$_POST[form_products]` array (line 11). If the variable is present, we can assume that the form has been submitted and information has already been stored in the `$_SESSION` superglobal. We then test for an array called `$_SESSION[products]` on line 12. If the array exists, it was populated on a previous visit to this script, so we merge it with the `$_POST[form_products]` array, extract the unique elements, and assign the result back to the `$products` array (lines 13–15). We then add the `$products` array to the `$_SESSION` superglobal on line 16.

Line 36 contains a link to another script, which we will use to demonstrate our access to the products the user has chosen. We create this new script in example.

Example: 5: Accessing Session Variables

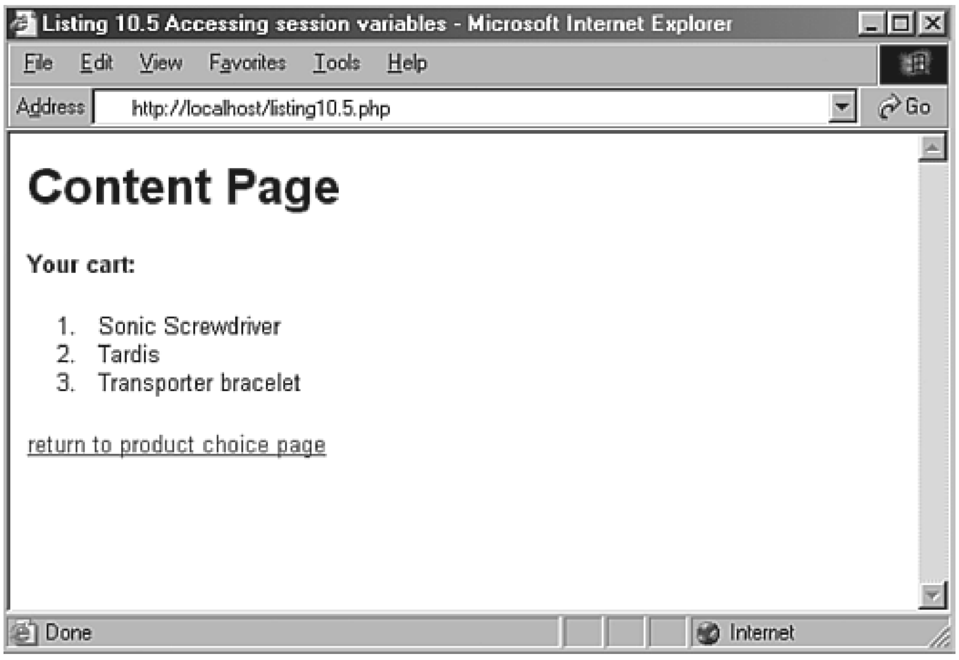
```
<?php
session_start();
?>
<html>
<head>
<title>Listing 10.5 Accessing session variables</title>
</head>
<body>
<h1> Content Page</h1>
```

Notes

```
10: <?php
11: if (isset($_SESSION[products])) {
12: echo "<strong>Your cart:</strong><ol>";
13: foreach (unserialize($_SESSION[products]) as $p) {
14:    echo "<li>$p";
15: }
16:  echo "</ol>";
17: }
18: ?>
19: <p><a href="listing10.4.php">return to product
    choice page</a></p>
20: </body>
21: </html>
```

Once again, we use `session_start()` to resume the session on line 2. We test for the presence of the `$_SESSION[products]` variable on line 11. If it exists, we unserialize it and loop through it on lines 13–15, printing each of the user's chosen items to the browser. An example is shown in Figure 10.3.

Figure 10.3: Accessing an Array of Session Variables


Listing 10.5 Accessing session variables - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Address
http://localhost/listing10.5.php
Go
Done
Internet

Content Page

Your cart:

1. Sonic Screwdriver
2. Tardis
3. Transporter bracelet

return to product choice page

For a real shopping cart program, of course, you would keep product details in a database and test user input, rather than blindly store and present it, but example 4 and 5 demonstrate the ease with which you can use session functions to access array variables set in other pages.

10.5 Destroying Session and Unsetting Variables

You can use `session_destroy()` to end a session, erasing all session variables. The `session_destroy()` function requires no arguments. You should have an established session for this function to work as expected. The following code fragment resumes a session and abruptly destroys it:

```
session_start();

session_destroy();
```

When you move on to other pages that work with a session, the session you have destroyed will not be available to them, forcing them to initiate new sessions of their own. Any registered variables will be lost.

The `session_destroy()` function does not instantly destroy registered variables, however. They remain accessible to the script in which `session_destroy()` is called (until it is reloaded). The following code fragment resumes or initiates a session and registers a variable called test, which we set to 5. Destroying the session does not destroy the registered variable.

```
session_start();

$_SESSION[test] = 5;

session_destroy();

print $_SESSION[test]; // prints 5
```

To remove all registered variables from a session, you simply unset the variable:

```
session_start();

$_SESSION[test] = 5;

session_destroy();

unset($_SESSION[test]);

print $_SESSION[test]; // prints nothing.
```

Case Study

Headers 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.

Contd...

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.

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"'>
```

10.6 Summary

- Cookies can be used for authentication, storing site preferences, shopping cart contents, the identifier for a server-based session, or anything else that can be accomplished through storing text data.
- Cookies can also be removed by the user at any time, so do not use them to stored anything too important.

- Session state is usually stored in a temporary file, although you can implement database storage using a function called session set save handler ().
- Destroying the session does not destroy the registered variable.

10.7 Keywords

***Changing the Session Function Streams*:** The following figure summarizes the session functions and their attributes. You cannot specify an input stream for the message (MSG) function.

Cookies*:* These are commonly used to rotate the banner ads that a site sends so that it doesn't keep sending the same ad as it sends you a succession of requested pages.

Deleting Cookies with PHP: PHP, or Hypertext Preprocessor, is an open-source scripting language primarily used for web programming. Deleting a cookie is almost the same as setting one. To delete it, you use the same parameters that you used when you set the cookie, except for the value, which needs to be an empty string, and the expiry date, which needs to be set in the past. On our logout page, we delete the cookie its different way.

Destroying Session and Unsetting Variables: Although a session's data is temporary and does not require that you explicitly clean after yourself, you may wish to delete some data for your various tasks.

Intensity (INT): It is refers to the brightness at which the information is displayed in a stream.

Session Function Overview: Session functions implement a concept that you have already seen; that is, the provision to users of a unique identifier, which can then be used from access to access to acquire information linked to that ID.

SESSIONS: Session Manager also has an input and output stream.

***Starting Session*:** A session is a combination of a server-side file containing all the data you wish to store, and a client-side cookie containing a reference to the server data. The file and the client- side cookie are created using the function *session_start()* - it has no parameters, but informs the server that sessions are going to be used.

Working with Session Variables: Session variables are similar to cookies, that is they are used to store information for a particular period of time. The values in session variables exist only till the session exists.

Lab Exercise

1. Write a program. If a variable that is PASSED BY REFERENCE is **unset()** inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before **unset()** was called.
2. Write a program. If a globalized variable is **unset()** inside of a function, only the local variable is destroyed.
3. Draw figure that shows the way our scripts can be tied together.

10.8 Self Assessment Questions

1. A cookie is a _______________ that allows the server to store its own information about an user on the user's own computer.

2. Cookies are commonly used to rotate the _______________ ads.
3. The check_auth() function checks whether the username and password match with the stored data and returns either the user id that belongs to the\ user or 0 when an error occurred.

   (*a*) True (*b*) False
4. The simplest way of setting a cookie is: setcookie('name', 'bret');

   (*a*) True (*b*) False
5. A cookie is a _______________ that is used by a server to identify a user of that server.

   (*a*) web file (*b*) .datfile

   (*c*) PHP file (*d*) .exe file
6. _______________ are used by default to store the session identifier.

   (*a*) PHP (*b*) session function()

   (*c*) cookies (*d*) identifier function
7. TSOIN is the input stream for the TSO/E function.

   (*a*) True (*b*) False
8. _______________ allows an application to store information for the current "session," which can be defined as one user being logged in to your application.

   (*a*) string function() (*b*) .exe function()

   (*c*) cookies (*d*) PHP session

10.9 Review Questions

1. Define cookies.
2. How we set cookies explain with the help of program?
3. Give program of deleting cookies with PHP.
4. What are the session functions? Give their types also.
5. What is session manager function()?
6. Define starting session.
7. How we destroy session and unsetting variable? Briefly explain with the help of program.
8. What is check_auth() function?
9. What is setting of cookies in PHP?
10. Explain deleting of cookies with PHP.

Answers for Self Assessment Questions

1.	mechanism	2.	banner
3.	(*a*)	4.	(*a*)
5.	(*a*)	6.	(*c*)
7.	(*a*)	8.	(*d*)

10.10 Further Reading

Books *PHP: A Beginner's Guide by: Vaswani, Vikram*, By Tata Mc-Graw Hill.

Online link http://www.w3schools.com/PHP/php_cookies.asp/

Unit 11: Directories and Files

CONTENTS

Objectives

After studying this unit, you will be able to:

- Discuss including files in include function.
- Explain the validating files.

Notes

- Understand the creating file.
- Discuss the deleting file.
- Explain the opening a file for reading.
- Discuss the writing files.
- Understand the appending files.

Introduction

The presentation of the file lessons will begin with how to create, open, and close a file. After establishing those basics, we will then cover other important file tasks, such as: create, open, read, write and append, files with PHP.

Linux stores data and programs in files. These are organized in directories. In a simple way, a directory is just a file that contains other files (or directories).

The part of the hard disk where you are authorized to save data is called your home directory. Normally all the data you want will be saved in files and directories in your home directory. To find your home directory (if you need), type:

echo $HOME

The symbol ~ can also be used for your home directory.

There is a general directory called/tmp where every user can write files. But files in/tmp usually get removed (erased) when the system boots or periodically, so you should not store in/tmp data that you want to keep permanently.

A file can be fully and uniquely identified by its full name, including all directories to which it belongs. The system starts at the root directory, with name/it "splits" into (sub) directories, and these split further, and so on, until you get to a file. For example, a home directory could be/usr15/pablo, on which there is a directory called programming, with a directory inside called include, on which there is a file called time.h, the full path of this last file will be/usr15/pablo/programming/include/time.h

11.1 Including Files with PHP Include( ) Function

The include() function enables you to incorporate file into another php file and the included file will run as it's part of that file. For example, if you want display standard or same information more than one page. It might be useful to create separate file and incorporate with any other file. It makes easier to maintain the information by changing one file instead of going through all the files.

Here is how you would use php include function.

```
<?php include("topMenu.php") ?>
```

or you can include it this way if the file is located in different directory

```
<?php include("/root/topMenu.php"); ?>
```

Following code is an example of simple php code utilizing include() function

```
<html>
```

Notes

```
<body>
<?php
if($_SESSION)
{
  $login=$_SESSION['login'] ;
}
else
{
  $login="";
}
if($login !="")
{
  include("member.php");
}
else
{
  include("notmember.php");
}
?>
</body>
</html>
```

Did u know? Linux stores data and programs in files. These are organized in directories.

11.2 Validating Files

Before you work with a file or directory within your code, it's often a good idea to learn more about it, and whether it actually exists is a pretty good start PHE provides many functions to help you to discover information about files on your system. This section briefly covers some of the useful functions.

11.2.1 Checking for Existence with file_exists ()

You can test for the existence of a file with the file_exists() function. This function requires a string representation of an absolute or relative path to a file, which might or might not be present. If the file if found, the file_exists() function return true; otherwise, it returns false.

```
if (file_exists (*test.text*)) {
       echo "The file exists";
       }
```

This is all well and good, but what if you're unsure whether something is a file or a directory, and you really need to know? Read on!

11.2.2 A File or a Directory?

You can confirm that the entity you're testing is a file, as opposed to a directorty, using the is_file() function. The *is_file ()* function requires the file path and returns a Boolean value.

```
if (is_file(*test.text*)) {

       Echo *test.txt is file!*;

}
```

Conversely, you might want to check that the entity you're testing is a directory. You can do this with the *is_dir()* function. *is_dir ()* requires the path to the directory and return a Boolean value.

```
if (is_dir("/tmp")) {

       echo  "/tmp is a directory";

}
```

After you know a file or directory exists, you might need to test its permissions.

You'll learn about this in the next section.

11.2.3 Checking the Status of a File

When you know that a particular entity exists, and it's what you expect it to be (either a directory or a file), you'll need to know what you can do with it. Typically, you might want to read, writer to, or execute a file. PHP can help you determine whether you can perform these operations.

The **is_readable()** function tells you whether you can read a file. On UNIX systems, you might be able to see a file but still be barred from reading its contents because of its user permissions. The **is_readable()** function accepts the file path as a string and returns a Boolean value.

```
if (is)_readable("test.txt")) {

       echo "test.txt is readable";

}
```

The *is_writable ()* function tells you whether you have the proper permission to write to a file. As with *is_readable()*, the *is_writable ()* function requires to file path and return a Boolean value.

```
if (is_writable ("test.txt")) {

       echo "test.txt is writable";

}
```

The *is_executable()* function tells you whether you can execute the given file, relying on either the file's permissions or its extension, depending on your platform.

The function accepts the file path and returns a Boolean value.

```
if (is_executable ("test.txt")) {

       Echo "test.txt is executable";

}
```

Permission-related information is not all you might need to know about a file. The next section shows how to determine the file size.

11.2.4 Determining File Size with *filesize()*

Given the path to a file, the filesize() function attempts to determine and return its size in bytes. It returns false if it encounters problems.

```
echo "The size of test.txt is ".filesize ("test.txt");
```

Finding the specific file size is important is situations where you want a attach a file to an email or stream a file to the user—you'll need to know the size so as to properly crate the headers (in the case of the email) ro0 known when to stop sending bytes to the user (in the case of the stream). For more general purposes, you might want to get the file size so that you can display it to the user before she attempts to download some monstrous application or high-resolution photograph from your site.

11.2.5 Getting Date Information About a File

Sometimes you need to know when a file was last written to or accessed. PHP provides several function that can provide this information.

You can find out the last-accussed time of a file using the *fileatime()* function.

This function requires the file path and returns the date that the file was last accessed. To access a file means either to read or write to it. Dates are returned from all the date information functions in time stamp—that is, the number of seconds since January 1, 1970. The examples in this book use the date() function to translate this value into human-readable form.

```
$atime = fileatime("test.txt");

Echo "test.txt was last accessed on *.date("D d M Y g:I A", Satime);
```

//Sam ple output: test.txt was last accessed on Sun 13 Jan 2008 5:33 AM.

You can discover the modification date of a file with the function filemtime(),

Which requires the file path and returns the date in UNIX epoch format. To modify a file means to change its contents in some way.

```
$atime = filemtime ("test.txt");

echo "test.txt was last modified on ".date("D d M Y g:I A", $mtime);
```

// Sample output; test.txt was last modified on Sun 13 Jan 2008 5:45 AM.

PHP also enables you to test the change time of a document with the filectime() function. On UNIX systems, the change time is set when a file's contents are modified or when changes are made to its permissions or ownership. On other platforms, the filectime() returns the creation date.

```
$ctime= filectime ("text.txt");

echo "test.txt was last changed on *.date ("D d M Y g:I A", $ctime);
```

// Sample output; test.txt was last changed on Sun 13 Jan 2008 5:45.

11.3 Creating File

File creation is a snap with PHP, but to make things easier we are going to make a directory that has permissions that allow the web server to write. Create a directory called test/ and make sure the permissions on it are set as explained above.

Now that you have the right permissions you can go ahead and make your first file. Create a PHP script named file.php inside the test/ folder and include the following code:

```
<?php

$file = “./test.txt”;
$fp = fopen (“$file”, “wb”);

$content = “Hello, I am a File!”;
fwrite($fp, $content);
fclose($fp);

echo “Wrote to ($file):<pre>”;
readfile($file);
echo “</pre>”;
?>
```

So what does all that do?

First off we have $file which just sets the name of the file you are creating with the script, in this case file.txt.

Next we have $fp or “file pointer” this is used with the fopen() function to first look for a file (specified by $file) and if not found create it. fopen() will be described in more detail in the next section.

The contents of the file will be determined in this case by the variable $content, and will be written to the file by the function fwrite(). You should now close your file, as you do not need to write anymore data to it via fclose(). For this example, we will print out what has been written to the file:

readfile($file) .

If the script has run correctly, you will see something like this:

Wrote to (./test.txt):

Hello, I am a File!

More on fopen();

Now let’s look a little more closely at what is going on here.

fopen() is the command that opens the file for reading or writing. It accepts a filename and one of six arguments: r, r+, w, w+, a, a+ as well as b for binary safe writing of files (Windows needs this for non-text files, for Unix it doesn’t matter. The safest way is thus to always use the “b” when working with binary, non-text files.)

In our case we used wb to open a binary file for writing only. This will truncate the file length to zero so that the file is blank and then place the content at the beginning of the file. If the file does not exist, it will attempt to create it.

Here is a breakdown of the modes:

w: This is the write mode, used to open a file for writing, Whenever this is used the file associated with the command will be recreated. w+ enables read/write mode.

r: This mode is used to open a file for reading. r+ enables read/write mode.

a: This is the append mode, your content will be added to the file therefore keeping any data already in the file. Change to a+ to enable read/write mode.

11.4 Delete File

Deleting a file using PHP is done by using the unlink function. Such a simple function with so much power! You need to use extreme caution when using the unlink function, as you don't want to accidentally delete a crucial file that is needed elsewhere for other functions. You can not delete a file that is already open using the fopen() function, as it is already being used. However, once you close a file you have the ability to delete it.

If the file is already opened, we need to use the fclose function to close the file. Remember how it is done:

fclose($file);

This is assuming the file that is opened is assigned the variable $file previously. (See fopen()) Now that the file has been closed, you have the ability to delete the file:

$file = "myfile.txt";

unlink($file);

First, you define which file you wish to delete. In this case, the file "myfile.txt" is defined using the variable "$file". Once you have defined the file, you then delete it using the unlink function. After executing the unlink function, the file will then be removed.

11.5 Opening a File for Reading

The fopen() function is used to open files in PHP. The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened:

```
<html>

<body>

<?php

$file=fopen("welcome.txt","r");

?>

</body>

</html>
```

The file may be opened in one of the following modes:

Modes	Description
r	Read only. Starts at the beginning of the file.
r+	Read/Write. Starts at the beginning of the file.
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist.
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist.
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist.
a+	Read/Append. Preserves file content by writing to the end of the file.
x	Write only. Creates a new file. Returns FALSE and an error if file already exists.
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists.

Notes If the fopen() function is unable to open the specified file, it returns 0 (false).

Example: The following example generates a message if the fopen() function is unable to open the specified file:

```
<html>

<body>

<?php

$file=fopen("welcome.txt","r"); exit (unable to open file!");

?>

</body>

</html>
```

11.6 Writing Files

Now that you know how to open and close a file, let's get on to the most useful part of file manipulation, writing! There is really only one main function that is used to write and it's logically called fwrite.

11.6.1 File Open Write

Before we can write information to our test file we have to use the function fopen to open the file for writing.

PHP Code

```
$myFile = “testFile.txt”;

$fh = fopen($myFile, ‘w’);
```

11.6.2 File Write Fwrite Function

We can use PHP to write to a text file. The fwrite function allows data to be written to any type of file. Fwrite’s first parameter is the file handle and its second parameter is the string of data that is to be written. Just give the function those two bits of information and you’re good to go!

Below we are writing a couple of names into our test file testFile.txt and separating them with a carriage return.

PHP Code

```
$myFile = “testFile.txt”;

$fh = fopen($myFile, ‘w’) or die(“can’t open file”);

$stringData = “Bobby Bopper\n”;

fwrite($fh, $stringData);

$stringData = “Tracy Tanner\n”;

fwrite($fh, $stringData);

fclose($fh);
```

The $fh variable contains the file handle for testFile.txt. The file handle knows the current file pointer, which for writing, starts out at the beginning of the file.

We wrote to the file testFile.txt twice. Each time we wrote to the file we sent the string $stringData that first contained Bobby Bopper and second contained Tracy Tanner. After we finished writing we closed the file using the fclose function.

If you were to open the testFile.txt file in NOTEPAD it would look like this:

Contents of the TestFile.txt File

```
Bobby Bopper
Tracy Tanner
```

We can write to a file by using fwrite() function PHP. Please note that we have to open the file in write mode and if write permission is there then only we can open it in write mode. If the file does not exist then one new file will be created. We can change the permission of the file also. However we can check the presence of a file by using file_exists function. You can read the content of a file by using fopen() function in PHP. This is the way to write entries to a guestbook, counter and many other scripts if you are not using any database for storing data. Here we will see how to write to a file.

```
<?
$body_content="This is my content"; // Store some text to enter inside the file
$file_name="test_file.txt"; // file name
$fp = fopen ($file_name, "w"); // Open the file in write mode, if file does not
fwrite ($fp,$body_content); // entering data to the file
fclose ($fp);            // closing the file pointer
chmod($file_name,0777);    // changing the file permission.
?>
```

11.6.3 File Write: Overwriting

Now that testFile.txt contains some data we can demonstrate what happens when you open an existing file for writing. All the data contained in the file is wiped clean and you start with an empty file. In this example we open our existing filetestFile.txt and write some new data into it.

PHP Code

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'w') or die("can't open file");
$stringData = "Floppy Jalopy\n";
fwrite($fh, $stringData);
$stringData = "Pointy Pinto\n";
fwrite($fh, $stringData);
fclose($fh);
```

If you now open the testFile.txt file you will see that Bobby and Tracy have both vanished, as we expected, and only the data we just wrote is present.

Contents of the TestFile.txt File

```
Floppy Jalopy
Pointy Pinto
```

11.7 Appending Files

So far we have learned how to open, close, read, and write to a file. However, the ways in which we have written to a file so far have caused the data that was stored in the file to be deleted. If you want to append to a file, that is, add on to the existing data, then you need to open the file in append mode.

Again, we attempt to open the file. This time we're passing mode 'a'. This is what we use to append the text to the end of the file. If you wish to add data to the end of the file and keep from overwriting the previous data, this is the mode you want to use.

So again we set a $line variable and write it to the file. This time, the line is added to the end of the file. You can open the file and take a look, and you'll see that the text is written on the next line.

But wait! Take a look at that '\n' newline character. That adds a line break in the file. If you hadn't put that in, the file would just contain.

This has been written to the file. This has been appended to the file.

If we want to add on to a file we need to open it up in append mode. The code below does just that.

PHP Code

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'a');
```

If we were to write to the file it would begin writing data at the end of the file. Using the *testFile.txt* file we created in the *File Write lesson* , we are going to append on some more data.

PHP Code

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'a') or die("can't open file");
$stringData = "New Stuff 1\n";
fwrite($fh, $stringData);
$stringData = "New Stuff 2\n";
fwrite($fh, $stringData);
fclose($fh);
```

The contents of the file *testFile.txt* would now look like this:

```
Floppy Jalopy
Pointy Pinto
New Stuff 1
New Stuff 2
```

The above example may not seem very useful, but appending data onto a file is actually used every day. Almost all web servers have a *log* of some sort. These various logs keep track of all kinds of information, such as: errors, visitors, and even files that are installed on the machine.

A log is basically used to document events that occur over a period of time, rather than all at once. Logs: a perfect use for append!

11.7.1 File Read

In this lesson, we will teach you how to read data from a file using various PHP functions.

11.7.2 File Open: Read

Before we can read information from a file we have to use the function fopen to open the file for reading. Here's the code to read-open the file we created in the PHP File Write lessons.

PHP Code

```
$myFile = "testFile.txt";

$fh = fopen($myFile, 'r');
```

The file we created in the last lesson was named "testFile.txt". Your PHP script that you are writing should reside in the same directory as "text.txt". Here are the contents of our file from File Write.

Did u know? We can open a file or a URL to read by using fopen() function of PHP.

testFile.txt Contents

```
Floppy Jalopy
Pointy Pinto
```

Now that the file is open, with read permissions enabled, we can get started!

11.7.3 File Read Fread Function

The fread function is the staple for getting data out of a file. The function requires a file handle, which we have, and an integer to tell the function how much data, in bytes, it is supposed to read.

One character is equal to one byte. If you wanted to read the first five characters then you would use five as the integer.

PHP Code

```
$myFile = "testFile.txt";

$fh = fopen($myFile, 'r');

$theData = fread($fh, 5);

fclose($fh);

echo $theData;
```

Display

Flopp

The first five characters from the testFile.txt file are now stored inside $theData. You could echo this string, $theData, or write it to another file.

If you wanted to read all the data from the file, then you need to get the size of the file. The filesize function returns the length of a file, in bytes, which is just what we need! The filesize function requires the name of the file that is to be sized up.

PHP Code

```
$myFile = "testFile.txt";

$fh = fopen($myFile, 'r');

$theData = fread($fh, filesize($myFile));

fclose($fh);

echo $theData;
```

Display

Floppy Jalopy Pointy Pinto

Notes It is all on one line because our "testFile.txt" file did not have a <br/> tag to create an HTML line break. Now the entire contents of the testFile.txt file is stored in the string variable $theData.

11.7.4 File Read: Gets Function

PHP also lets you read a line of data at a time from a file with the fgets function. This can or cannot be useful to you, the programmer. If you had separated your data with new lines then you could read in one segment of data at a time with the gets function.

Lucky for us our "testFile.txt" file is separated by new lines and we can utilize this function.

```
PHP Code

$myFile = "testFile.txt";

$fh = fopen($myFile, 'r');

$theData = fgets($fh);

fclose($fh);

echo $theData;
```

testFile.txt Contents

```
Floppy Jalopy
```

The fgets function searches for the first occurrence of "\n" the newline character. If you did not write newline characters to your file as we have done in File Write, then this function might not work the way you expect it to.

Did u know? If you want to append to a file, i.e., add on to the existing data, then you need to open the file in append mode.

11.7.5 When and How to Include Files

You can save yourself from some potential trouble by including files wisely.

You should follow three principles on including files:

1. Only use include_once or require_once. Rule number one is to always use require_once or include_once to include PEAR code. If you use require, your script will likely die because of redefinition errors (or it will die sometime in the future).

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, and you're finished. Here are some examples:

   Class Name File Name

   PEAR PEAR.php

   XML_Parser XML/Parser.php

   HTML_Quickform_textarea HTML/QuickForm/textarea.php

   Case is significant here because UNIX file systems are case-sensitive.

   Gutmans_ch12 Page 410 Thursday, September 23, 2004 2:53 PM

   12.6 Building Packages 411

3. Encapsulate includes. Each file should use includes to express clearly which class it depends on from other packages.

Case Study

Scan Directories with PHP's Directory Iterators

One of php5's most interesting new features is the addition of iterators, a collection of readmade interfaces designed to help in navigating and processing hierarchical data structures. These Iterators signficantly reduce the amount of code required to process an XML document tree or a file collection. A number of Iterators are available, including the Array Iteratior, CachingIterator, Limit Iterator, Racursive Iterator, Simple XMLIterator and Directory Iterator.

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-> is Dot ())  {
```

Contd...

Notes

```
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 Directory Iteratr class provieds 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 is Dot 90 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 Directory Iterator 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 iterators: the Recursive Directory Iterator and the Recursivelterator Iterator, which together incorporate all the above functionality. Take a look at.

Listing A

Questions

1. How Scan directories and PHP interrelated?

2. What is processing a nested directory tree?

11.8 Summary

- Validating the file helps to limit the file size.
- Fopen( ) is the command that opens the file for reading or writing. It accepts a file name and one of six arguments =r,rt, wt, a, at for unity of files.
- Deleting a file using PHP is done by using the unlink function.
- If you want to append to a file, i.e. add on to the existing data, them you need to open the file in append mode.

11.9 Keywords

CHMOD*:* It is a Unix command that lets you tell the system how much (or little) access it should permit to a file.

File Validator: The file validator helps validating files and even image file types.

Fopen*:* Opens file or URL, **fopen()** binds a named resource, specified by filename, to a stream.

Home Directory: The part of the hard disk where you are authorized to save data is called home directory.

$lines: contents of YourFile.txt into the array called $lines.

/tmp: There is a general directory called/tmp where every user can write files.

Lab Exercise

1. Write all PHP Code. For File Write: Overwriting.
2. Write all PHP Code. For File Write: Fwrite Function.

11.10 Self Assessment Questions

1. File () is similar to fgets in that it reads the data one line at a time, however it returns it all at once into an array.

   (*a*) True (*b*) False

2. The file validator helps validating files and even image file types.

   (*a*) True (*b*) False

3. Manipulating files is a basic necessity for

   (*a*) serious programmers (*b*) normal programmers

   (*c*) programmers (*d*) none of the above.

4. $our FileName = "testFile.txt";

   (*a*) True (*b*) False

5. We will be using function to read the content by using a file pointer.

   (*a*) Thread() (*b*) Fread()

   (*c*) Fread_() (*d*) All of the above.

6. Write a way to look at the content of a file

   (*a*) with an editor (*b*) within editor

   (*c*) without an editor (*d*) all of the above.

11.11 Review Questions

1. What do you mean by truncating a file?
2. What are the three basic ways to open a file and the corresponding character that PHP uses?
3. How to write data to a Text File?
4. What is the procedure to include files?
5. What precautions should be taken while manipulating files?

Notes

6. What are the advantages of validating a file?
7. What is the basic use of command CHMOD?
8. What are the different ways to open a file?

Answers for Self Assessment Questions

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

11.12 Further Reading

Books *Teach Yourself PHP, MySQL & Apache*, By: Meloni, Pearson Education.

Online link linux.math.tifr.res.in/linux-manual/f...

Manish Kumar, Lovely Professional University

Unit 12: Images

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain image creation process.
- Discuss necessary modifications to PHP.
- Explain how to draw a new image.
- Explain how to modify an existing image.
- Discuss how to create an image from user input.

Introduction

PHP makes it very easy to do many things needed on a website, among which is to create an image. The ability to generate an image in PHP can be useful if you want to do things like create CAPTCHA images, or even design a banner or logo on the fly the way some free blogging software do.

By the end of this tutorial, you will be able to create or modify an existing image using PHP, set the colours of its background, the text and the lines you draw, write text to that image, draw a line and set its thickness. You will also be able to either send that image to a web browser or to save it as a file. Armed with this basic foundation, you will also be able to knowledgeably explore further and perform more complex operations on images using PHP if you wish.

12.1 Image Creation Process

12.1.1 Basic PHP Knowledge

Assume that you already have some basic knowledge of how to write PHP programs. If not, you may wish to check out my basic PHP tutorial, How to Program in PHP.

12.1.2 Your PHP Must Have Been Compiled with the GD Library

For any of the functions listed here to be available, your PHP interpreter must have been compiled with the GD library. If you are using the PHP provided by a commercial web host, this is probably already provided as a standard feature.

12.1.3 Free Type Must Be Compiled for True Type Font Support

If you want True Type font support, your PHP must have FreeType support compiled into it as well.

12.1.4 Creating an Image from Scratch Using PHP

The easiest way to understand how to create an image is by looking at some sample code.

```
<?php
     $my_img = imagecreate(200, 80);
     $background = imagecolorallocate($my_img, 0, 0, 255);
     $text_colour = imagecolorallocate($my_img, 255, 255, 0);
     $line_colour = imagecolorallocate($my_img, 128, 255, 0);
     imagestring($my_img, 4, 30, 25, “thesitewizard.com”,
     $text_colour);
     imagesetthickness ($my_img, 5);
     imageline($my_img, 30, 45, 165, 45, $line_colour);
     header(“Content-type: image/png”);
     imagepng($my_img);
```

```
        imagecolordeallocate($line_color);
        imagecolordeallocate($text_color);
        imagecolordeallocate($background);
        imagedestroy($my_img);
  ?>
```

The above code creates a 200x80 PNG image with a blue background and yellow text. It can be called from within your web page simply by referencing the php file. For example, if the PHP file that contains the above code is called myimage.php, then the HTML code to invoke it can simply be:

```
<img src="myimpage.php" alt="Image created by a PHP script" width="200" height="80">
```

Explanation of the Code

- **Creating the Image**

  The first thing the code does is to call the imagecreate() function with the dimensions of the image, namely its width and height in that order. This function returns a resource identifier for the image which we save in$my_img. The identifier is needed for all our operations on the image.

  Caution If the function fails for some reason, it will return FALSE. If you want your code to be robust, you should test for this.

- **Using Colours in PHP**

  Before you can use any sort of colours in your image at all, you will need to allocate the colour. Colours are represented by three digits, known as the RGB value. The first digit denotes the red component, the second the green and the third blue, hence RGB, for Red-Green-Blue. These are the same colour values that you use for your web page as well as numerous other computer applications.

  Colours are allocated using the imagecolorallocate() function. This function will automatically fill the background of the image with the colour the first time you call it, as well as return an identifier for that particular colour. Subsequent calls to imagecolorallocate() will simply create a colour identifier for your colour, without affecting your image background.

  As you can see from the above code, my script allocates a blue identifier for the image, and in so doing, causes imagecolorallocate() to set the background to blue automatically. It also allocates a colour identifier for yellow and one for a shade of green. The latter two identifiers will be used later to write text and draw a line.

  imagecolorallocate() returns FALSE if the function fails for any reason.

- **Writing Text to the Image**

  To write text to your image, you will need to use the imagestring() function. This function uses a set of built-in fonts to do the writing. The fonts have various sizes, ranging from 1 to 5, where 1 is the smallest font size and 5 the largest. The size of the font is specified in the second parameter to the function. The third and fourth parameters to imagestring() specify the x,y coordinate for the top left hand corner of the text. In the case of the example above, my text will begin 25 pixels from the top edge of the image, and 30 pixels from the left.

Notes

The fifth parameter is for the text to print, and the final parameter the colour of the text. This is the same colour that was allocated earlier using imagecolorallocate().

- **Drawing a Line and Setting the Thickness of the Brush**

  The imageline() function can be used to draw a line to the image. To set the thickness of the brush used to draw the line, you may want to call the imagesetthickness() function as I did in my example. The numeric parameter to imagesetthickness() is the thickness of the brush in pixels. If you don't call imagesetthickness(), the line will be 1 pixel thick.

  The imageline() function is called with the start and end coordinates of the line, in x,y format. The line starts from 30,45 and ends on 165,45. That is, it will be a horizontal line 45 pixels from the top, starting 30 pixels from the left edge and ending 165 pixels from that same edge. Since $line_colour was set to a shade of green earlier, the line will be green.

- **How to Output the Image**

  Since the output of example script is the image itself, send an "image/png" content type header to the browser telling it that what follows are the bytes of a PNG image. The function imagepng() is then called to generate the necessary image from $my_img image identifer. If you prefer to save your image, don't call the header() function to output the header, and call imagepng() with the filename of the image for its second parameter, like the following:

  imagepng($my_img, "my_new_image.png");

  Your image does not have to be a PNG image. You can use imagegif() or imagejpeg() to create GIF and JPG images respectively. You should of course send the correct content type header for the type of image you are creating. For example, a jpeg image should have a content type of "image/jpeg" while a gif image "image/gif". Note though that GIF support may or may not necessarily be compiled into the version of the GD library your web host is using, so if you're not sure, use one of the other file formats.

- **Freeing Resources**

  On completion, the program releases the resources associated with the image by callingimagecolordeallocate() and imagedestroy().

Did u know? Colours are represented by three digits, known as the RGB value. The first digit denotes the red component, the second the green and the third blue. These colours are used for web pages as well as numerous other computer applications.

The fonts have various sizes, ranging from 1 to 5, where 1 is the smallest font size and 5 the largest.

12.2 Necessary Modifications to PHP

Current versions of the PHP distribution include a bundled version of GD graphics library. The inclusion of this library eliminates the need to download and install several third-party libraries, but this library will need to be activated at installation time.

If you cannot install PHP, "Installing and Configuring PHP," and are stuck with a version of PHP earlier than 4.3.0, you will have to go to http://www.boutell.com/gd/ and download the source of the GD library. Follow the instructions included with that software, and consult its manual for difficulties with installation.

To enable the use of the GD library at installation time, Linux/Unix users must add the following to the configure parameters when preparing to build PHP:

—with-gd

If you download your own version of GD, you must specify the path, as in

—with-gd=/path/to/gd.

After running the PHP configure program again, you must go through the make and make install process. Windows users who want to enable GD simply have to activate php_gd2.dll as an extension in the php.ini file.

When using the GD library, you are limited to working with files in GIF format. However, by installing additional libraries, you can work with JPEG and PNG files as well.

12.2.1 Obtaining Additional Libraries

Working with GIF files might suit your needs perfectly, but if you want to create JPEG or PNG files, you will need to download and install a few libraries, and make some modifications to your PHP installation.

JPEG libraries and information can be found at ftp://ftp.uu.net/graphics/jpeg/.

PNG libraries and information can be found at http://www.libpng.org/pub/png/libpng.html.

If you are working with PNG files, you should also install the zlib library, found at http://www.gzip.org/zlib/.

Follow the instructions at these sites to install the libraries. After installation, Linux/Unix users must again reconfigure and rebuild PHP by first adding the following to the PHP configures parameters (assuming that you want to use all three, if not, just add the applicable ones):

—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 make and make install process. Your libraries should then be activated and ready for use.

12.3 Drawing a New Image

Drawing shapes and lines with PHP is nothing like drawing with image editing program. Actually, when drawing with PHP you become the editing program. You use individual PHP functions to define colours, draw and fill shapes, re-size and save the image. These functions are part of the GD graphics library that was bundled beginning with PHP version 4.3.0.

The basic function used to create a new image is called ImageCreate(). This function creates the canvas area for your new image. For example to create an image that is 300px wide and 300px high you would use following code:

1. $imageOne = ImageCreate(300, 300);

Now that you have canvis you need to define colours you want to use in it. Colours are defined using RGB color system. Using decimal values from 0 to 255 for each of the red (R), green (G), and blue(B) you can define a specific color. The function used to define colors is ImageColorAllocate().

The first color you allocate is used as the background color of the image.

1. $white = ImageColorAllocate($imageOne, 255, 255, 255);
2. $red = ImageColorAllocate($imageOne, 255, 0, 0);
3. $blue = ImageColorAllocate($imageOne, 0, 0, 255);
4. $green = ImageColorAllocate($imageOne, 0, 255, 0);

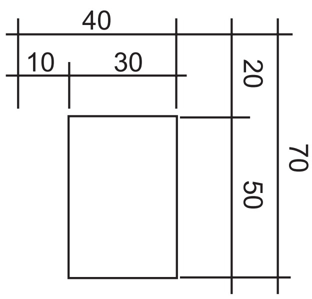
12.3.1 Drawing Lines and Shapes

There are several PHP functions to assist you in drawing lines and shapes. And as you can see below the function names are very descriptive.

- **ImageEllipse()** — To draw an ellipse.
- **ImageArc()** — To draw arc (partial ellipse).
- **ImagePolygon()** — To draw a polygon.
- **ImageRectangle()** — To draw a rectangle.
- **ImageLine()** — To draw a line.

Each of these functions use x-axis and y-axis coordinates as indicators of where to start and stop the drawing on the canvas. Here is sample of green rectangle that is 30px wide, 50px high and 10px from the left edge and 20px away from the top of the canvas.

1. Image Rectangle ($image One, 10, 20, 40, 70, $green);



ImageRectangle($imageOne, 10, 20, 40, 70, $green);

As you can see drawing with PHP requires some planning ahead.

12.3.2 Using a Colour Fill

PHP can also fill the shapes with solid colour. Functions to do that are:

- **ImageFilledEllipse()** — To fill an ellipse.

Notes

- **ImageFilledArc()** — To fill a partial ellipse.
- **ImageFilledPlygon()** — To fill a polygon.
- **ImageFilledRectangle()** — To fill a rectangle.

These functions are used just like nonfill drawing functions explained above.

1. Image Filled Rectangle ($image One, 10, 20, 40, 70, $green);

PHP comes with several built-in styles that are used in the display. For example IMG_ARC_PIE says to create a rounded edge.

12.4 Modifying an Existing Image

In most cases, creating an image from scratch is overkill. For most web purposes, you can usually design the basic background of your image using a normal image_editor like Photoshop and only add any additional text or graphical elements that need to be dynamically drawn using PHP. This allows you to speed up your scripts and reduce the resource consumption on your web server. It also lets you create your picture using professional picture designing tools.

To use an existing GIF, JPEG or PNG image as a canvas on which you add additional elements, use one of the following functions instead of imagecreate().

imagecreatefromgif (string $filename)

imagecreatefromjpeg (string $filename)

imagecreatefrompng (string $filename)

For example, if you created a GIF file called "mytemplate.gif", the function can be called as follows:

$myimage = imagecreatefromgif ("mytemplate.gif");

Like the basic imagecreate() function, these functions return FALSE if they fail to load the image for any reason.

12.4.1 Using True Type Fonts

If you want to use a True Type font, you will need to use imagettftext() instead. For details on how to use this function, please consult the function's manual page on php.net.

You should note a few things, though, before using this function:

- Check that your web host has compiled FreeType support into PHP before you rely on this function.
- Find out whether your web host's PHP is using GD version 1 or 2. This affects a number of things, including the meaning of the font size parameter to the function (whether it means pixel size or point size).
- Note that the coordinates for the text has a different starting point from imagestring(). That is, a coordinate like (say) 10, 20 has a different meaning in the two functions.
- Make sure the font you want exists on your web server. Remember that your web server is not the same as your computer. Just because a font like Arial exists on your computer

Notes

does not mean that it exists on your server. If you have fonts that you created yourself, you may want to upload those to your web directory and use those instead. At least, that way, you have a guarantee that the font needed by your script exists.

- The path setting to the font file is tricky, and depends on the version of the GD library your web server is using. One way around it is to specify the full path to the file on your server.

12.4.2 Drawing to Your Image

Besides the line drawing function used above, PHP has other functions that you can use. To whet your appetite, functions include those that allow you to draw ellipses, arcs and polygons, change the style of your lines (to say dashed lines), and so on.

However, unless you have special reasons why you might want to dynamically draw complex pictures onto an image, you should consider creating your base image using a normal picture editor, load that image using a function like imagecreatefrompng(), and then only modifying small details with your script. Generating everything from scratch is unnecessary for most purposes, and can drain your web server resources.

12.5 Image Creation from User Input

You understood that having created an image with php it was possible to load it using the img tag and it certainly works where the comment is hard coded.

HTML Code:

```
<html>

<head>

<title>myscripthtml</title>

<script type=”text/javascript”>

       function getComments()

       {

              return true;

       }

</script>

</head>

<body>

Provide comments

<form name=”cmt” action=”myscript2.php” method=”GET” onSubmit=”return
getComments()”/>

<input type=”text” name=”cmment”/>

<input type=”submit” value=”submit comments”/>

</form>
```

```
</body>
</html>
```

The following 'myscript2.php' file creates an image BUT I can't get that image to load using the img tag:

HTML Code:

```
<img src="myscript2.php"/>
```

PHP Code:

```
<?php
// Path to our font file
$font = 'arial.ttf';
$fontsize = 10;
$quotes = $_GET['cmment'];
// get the quote and word wrap it
$quote = wordwrap($quotes,20,"<br />\n");
// Create image
$image = imagecreatefrompng('baseimage.png');
// pick color for the background
$bgcolor = imagecolorallocate($image, 100, 100, 100);
// pick color for the text
$fontcolor = imagecolorallocate($image, 255, 255, 255);
// fill in the background with the background color
imagefilledrectangle($image, 200, 100, 400, 200, $bgcolor);
// x,y coords for imagettftext defines the baseline of the text: the
lower-left corner
// so the x coord can stay as 0 but you have to add the font size
to the y to simulate
// top left boundary so we can write the text within the boundary
of the image
$x = 0;
$y = $fontsize;
imagettftext($image, $fontsize, 0, $x, $y, $fontcolor, $font,
$quote);
// tell the browser that the content is an image
header('Content-type: image/png');
// output image to the browser
imagepng($image);
// delete the image resource
imagedestroy($image);
?>
```

Case Study **Enterprise Solutions Using Java-PHP**

Summary: Production use of Resin® and Quercus™ to deploy combined Java-PHP solution for Emergency Preparedness and Response for Healthcare.

Solution: Combined Java-PHP architecture.

Contd...

Product: Quercus, a feature of Resin application server.

Industry: Medical

Engineering 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 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

Contd...

Notes

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:

3/4 Ability to use PHP libraries

3/4 Use of PHP5 object model for page templating

3/4 PHP as Servlet better works than Zend solution or JSR223

3/4 Multithreading

3/4 Background tasks

3/4 Event scheduling

3/4 Java persistence

3/4 Ability to use third party Java or PHP libraries to solve problems Result:

LiveProcess has chosen Quercus and Resin as their Platform of choice.

Questions

1. Define solution for Emergency Preparedness and Response for Healthcare.

2. Define Java-PHP bridge or Quercus.

12.6 Summary

- The ability to generate an image in PHP can be useful if you want to do things like create CAPTCHA image.
- Image create function returns a resource identifier for the image which are same insmying.
- Colour are allocated using the image colour allocate () function.
- Working with files in GIF format with using GD library is limited.
- Other facilities are also available in PHP to design and modify your image.

12.7 Keywords

CAPTCHA Image*_**A ***CAPTCHA or ***Captcha:*** It is a type of challenge-response test used in computing as an attempt to ensure that the response is not generated by a computer.

Free Type: It is a software library written in C that implements a font rasterization engine. It is used to rasterize characters into bitmaps and provides support for other font-related operations.

Image Creator: The tool used to create Mee Go images is something called “MIC2” (for distinguishing from obsolete MIC - Moblin Image Creator). MIC is composed of a series of tools to create images, convert images and do some development work on MeeGo. MIC2 is based primarily on Fedora livecd-tools and appliance-tools.

Notes

True Type: It is an **outline font standard** originally developed by **Apple Computer** in the late 1980s as a competitor to **Adobe's Type 1 fonts** used in **Post Script.**

Lab Exercise

Create an Image from Scratch Using PHP

Draw an Image

12.8 Self Assessment Questions

1. Colours are allocated using the imagecolorallocate() function.
   (*a*) True (*b*) False
2. To write text to your image, you will need not to use the imagestring() function.
   (*a*) True (*b*) False
3. Colours are represented by three digits, known as
   (*a*) RCB value (*b*) RTB value
   (*c*) RGB value (*d*) None of the above.
4. Current versions of the PHP distribution include a bundled version.
   (*a*) GD graphics library (*b*) CD graphics library
   (*c*) GE graphics library (*d*) All of the above.
5. What basic function used to create a new image is called?
   (*a*) Imagecreate() (*b*) Image Create()
   (*c*) Image Create() (*d*) None of the above.
6. In addition to creating images from other images, and drawing images on your own, you can not create images based on user input.
   (*a*) True (*b*) False

12.9 Review Questions

1. What RGB values would you use for pure black and pure white?
2. How do you create a new, blank canvas that is 300 pixels wide and 200 pixels tall?
3. What functions is used to draw a polygon and a filled polygon?
4. How to create an Image in PHP?
5. Create an Image from Scratch Using PHP.
6. Explain Image Creation Process.
7. Give explanation of the ICP Code.
8. What does Writing Text to the Image mean?
9. Explain how to output the Image.

10. What are Necessary Modifications to PHP in Images?
11. How will you draw a new Image?
12. Explain Modifying an Existing Image.

Answers for Self Assessment Questions

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

12.10 Further Reading

Books

Open Source Development with LAMP: Using Linux, Apache, MySQL, Perl & PHP By James Lee, Pearson Education.

Online link

http://www.softpanorama.org/Tools/dd.shtml

Unit 13: Stored Procedure

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain transactions.
- Understand stored procedures.

Introduction

In a Database Management System (DBMS), a stored procedure is a set of Structured Query Language (SQL) statements with an assigned name that's stored in the database in compiled form so that it can be shared by a number of programs. The use of stored procedures can be helpful in controlling access to data (end-users may enter or change data but do not write procedures), preserving data integrity (information is entered in a consistent manner), and improving productivity (statements in a stored procedure only need to be written one time).

A stored procedure is a set of SQL commands that has been compiled and stored on the database server. Once the stored procedure has been "stored", client applications can execute the stored procedure over and over again without sending it to the database server again and without compiling it again. Stored procedures improve performance by reducing network traffic and CPU load.

13.1 Transactions

A transaction is a sequential group of database manipulation operations, which is performed as if it were one single work unit. In other words, a transaction will never be complete unless each individual operation within the group is successful. If any operation within the transaction fails, the entire transaction will fail.

A good example would be a banking transaction, specifically a transfer of $100 between two accounts. In order to deposit money into one account, you must first take money from another account. Without using transactions, you would have to write SQL statements that do the following:

1. Check that the balance of the first account is greater than $100.
2. Deduct $100 from the first account.
3. Add $100 to the second account.

Additionally, you would have to write your own error-checking routines within your program, specifically to stop the sequence of events should the first account not have more than $100 or should the deduction statement fail. This all changes with transactions, for if any part of the operation fails, the entire transaction is rolled back. This means that the tables and the data inside them revert to their previous state.

13.1.1 Properties of Transactions

Transactions have the following four standard properties, usually referred to by the acronym ACID:

- Atomicity ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure, and previous operations are rolled back to their former state.
- Consistency ensures that the database properly changes states upon a successfully committed transaction.
- Isolation enables transactions to operate independently of and transparent to each other.
- Durability ensures that the result or effect of a committed transaction persists in case of a system failure.

In MySQL, transactions begin with the statement BEGIN WORK and end with either a COMMIT or a ROLLBACK statement. The SQL commands between the beginning and ending statements form the bulk of the transaction.

13.1.2 COMMIT and ROLLBACK

When a successful transaction is completed, the COMMIT command should be issued so that the changes to all involved tables will take effect. If a failure occurs, a ROLLBACK command should be issued to return every table referenced in the transaction to its previous state.

Did u know?

In MySQL as well as NuSphere's Enhanced MySQL, you can set the value of a session variable called AUTOCOMMIT. If AUTOCOMMIT is set to 1 (the default), then each SQL statement (within a transaction or not) is considered a complete transaction, committed by default when it finishes. When AUTOCOMMIT is set to 0, by issuing the SET AUTOCOMMIT=0 command, the subsequent series of statements acts like a transaction, and no activities are committed until an explicit COMMIT statement is issued.

If transactions were not used in application development, a large amount of programming time would be spent on intricate error checking. For example, suppose your application handles customer order information, with tables holding general order information as well as line items for that order. To insert an order into the system, the process would be something like the following:

1. Insert a master record into the master order table.
2. Retrieve the ID from the master order record you just entered.
3. Insert records into the line items table for each item ordered.

If you are not in a transactional environment, you will be left with some straggly data floating around your tables; if the addition of the record into the master order table succeeds, but steps 2 or 3 fail, you are left with a master order without any line items. The responsibility then falls on you to use programming logic and check that all relevant records in multiple tables have been added or go back and delete all the records that have been added and offer error messages to the user. This is extremely time-consuming, both in man-hours as well as in program-execution time.

In a transactional environment, you'd never get to the point of childless rows, as a transaction either fails completely or is completely successful.

13.1.3 Row-Level Locking

Transactional table types support row-level locking, which differs from the table-level locking that is enforced in MyISAM and other nontransactional table types. With tables that support row-level locking, only the row touched by an INSERT, UPDATE, or DELETE statement is inaccessible until a COMMIT is issued.

Rows affected by a SELECT query will have shared locks, unless otherwise specified by the programmer. A shared lock allows for multiple concurrent SELECT queries of the data. However, if you hold an exclusive lock on a row, you are the only one who can read or modify that row until the lock is released. Locks are released when transactions end through a COMMIT or ROLLBACK statement.

Setting an exclusive lock requires you to add the FOR UPDATE clause to your query. In the sequence below, you can see how locks are used to check available inventory in a product catalog before processing an order. This example builds on the previous example by adding more condition-checking.

This sequence of events is independent of the programming language used; the logical path can be created in whichever language you use to create your application.

1. Begin transaction.

   BEGIN WORK;

2. Check available inventory for a product with a specific ID, using a table called inventory and a field called qty.

   SELECT qty FROM inventory WHERE id = 'ABC-001' FOR UPDATE;

3. If the result is less than the amount ordered, rollback the transaction to release the lock.

   ROLLBACK;

4. If the result is greater than the amount ordered, continue issuing a statement that reserves the required amount for the order.

   UPDATE inventory SET qty = qty - [amount ordered] WHERE id = 'ABC-001';

5. Insert a master record into the master order table.
6. Retrieve the ID from the master order record you just entered.
7. Insert records into the line items table for each item ordered.
8. If steps 5 through 7 are successful, commit the transaction and release the lock.

   COMMIT;

While the transaction remains uncommitted and the lock remains in effect, no other users can access the record in the inventory table for the product with the ID of ABC-001. If a user requests the current quantity for the item with the ID of ABC-002, that row still operates under the shared lock rules and can be read.

13.2 Stored Procedures

A stored procedure is a subroutine available to applications accessing a relational database system. Stored procedures (sometimes called a proc, sproc, StoPro, StoredProc, or SP) are actually stored in the database data dictionary.

Typical uses for stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures are used to consolidate and centralize logic that was originally implemented in applications. Extensive or complex processing that requires the 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.

Did u know? 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 languages 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.

13.2.1 Implementation

The exact and correct implementation of stored procedure varies from one database system to another. Most major database vendors support them in some form. Depending on the database ystem, stored procedures can be implemented in a variety of programming languages, for example SQL, Java, C, or C++. Stored procedures written in non-SQL programming languages may or may not execute SQL statements themselves.

The increasing adoption of stored procedures led to the introduction of procedural elements to the SQL language in the SQL:1999 and SQL:2003 standards in the part SQL/PSM. That made SQL an imperative programming language. Most database systems offer proprietary and vendor-specific extensions, exceeding SQL/PSM.

Database system	**Implementation language**
• Microsoft SQLServer	Transact-SQL and various .NET Framework languages
• Oracle	PL/SQL or Java
• DB2	SQL/PL or Java
• Informix	SPL
• Postgre SQL	PL/pg SQL, can also use own function languages such as pl/perl or pl/php
• Fire bird	PSQL (Fyracle also supports portions of Oracle's PL/SQL)
• My SQL	Own stored procedures, closely adhering to SQL: 2003 standard.

13.2.2 Other Uses

In some systems stored procedures can be used to control transaction management; in others, stored procedures run inside a transaction such that transactions are effectively transparent to them. Stored procedures can also be invoked from a database trigger or a condition handler. For example, a stored procedure may be triggered by an insert on a specific table, or update of a specific field in a table, and the code inside the stored procedure would be executed. Writing stored procedures as condition handlers also allows database administrators to track errors in the system with greater detail by using stored procedures to catch the errors and record some audit information in the database or an external resource like a file.

13.2.3 Comparison with Dynamic SQL

- **Overhead:** Because stored procedure statements are stored directly in the database, they *may* remove all or part of the compilation overhead that is typically required in situations where software applications send inline (dynamic) SQL queries to a database. (However, most database systems implement "statement caches" and other mechanisms to avoid repetitive compilation of dynamic SQL statements.) In addition, while they avoid some overhead, pre-compiled SQL statements add to the complexity of creating an optimal execution plan because not all arguments of the SQL statement are supplied at compile time. Depending on the specific database implementation and configuration, mixed performance results will be seen from stored procedures versus generic queries or user defined functions.
- **Avoidance of Network Traffic:** A major advantage with stored procedures is that they can run directly within the database engine. In a production system, this typically means that the procedures run entirely on a specialized database server, which has direct access to the data being accessed. The benefit here is that network communication costs can be avoided completely. This becomes particularly important for complex series of SQL statements.
- **Encapsulation of Business Logic:** Stored procedures allow for business logic to be embedded as an API in the database, which can simplify data management and reduce the need to encode the logic else where in client programs. This may result in a lesser likelihood of data becoming corrupted through the use of faulty client programs. The database system can ensure data integrity and consistency with the help of stored procedures.
- **Delegation of Access-Rights:** In many systems, stored-procedures can be granted access rights to the database which the users who will execute those procedures do not directly have.
- **Some Protection from SQL Injection Attacks:** Stored procedures can be used to protect against injection attacks. Stored procedure parameters will be treated as data even if an attacker inserts SQL commands. Also, some DBMSs will check the parameter's type.

13.2.4 Comparison with Functions

- A function is a subprogram written to perform certain computations and return a single value.
- Functions must return a value (using the RETURN keyword), but for stored procedures this is not compulsory.
- Stored procedures can use RETURN keyword but without any value being passed.
- Functions could be used in SELECT statements, provided they don't do any data manipulation. However, procedures cannot be included in SELECT statements.

- A function can have only IN parameters, while stored procedures may have OUT or INOUT parameters.
- A stored procedure can return multiple values using the OUT parameter or return no value at all.

13.2.5 Disadvantages

Stored procedure languages are quite often vendor-specific. If you want to switch to using another vendor's database, then you have to rewrite your stored procedures. Stored procedure languages from different vendors have different levels of sophistication; for example, Oracle's PL/SQL has more languages features and built-in features (via packages such as DBMS_ and UTL_ and others) than Microsoft's T-SQL. Tool support for writing and debugging stored procedures are often not as good as for other programming languages; but again, this differs between vendors and languages (for example, both PL/SQL and T-SQL have dedicated IDEs and debuggers).

13.2.6 Why Use Stored Procedures?

There are several advantages of using stored procedures instead of standard SQL. First, stored procedures allow a lot more flexibility offering capabilities such as conditional logic. Second, because stored procedures are stored within the DBMS, bandwidth and execution time are reduced. This is because a single stored procedure can execute a complex set of SQL statements. Third, SQL Server pre-compiles stored procedures such that they execute optimally. Fourth, client developers are abstracted from complex designs. They would simply need to know the stored procedure's name and the type of data it returns.

13.2.7 Creating a Stored Procedure

Enterprise Manager provides an easy way to create stored procedures. First, select the database to create the stored procedure on. Expand the database node, right-click on "Stored Procedures" and select "New Stored Procedure...". You should see the following:

Create procedure [owner].[procedure name] as

Substitute OWNER with "dbo" (database owner) and PROCEDURE NAME with the name of the procedure. For example:

Create procedure [dbo].[getproducts] as

So far, we are telling SQL Server to create a new stored procedure with the name GetProducts. We specify the body of the procedure after the AS clause:

CREATE PROCEDURE [dbo].[GetProducts] ASSELECT ProductID, ProductName FROM Products Click on the Check Syntax button in order to confirm that the stored procedure is syntactically correct. Please note that the GetProducts example above will work on the Northwind sample database that comes with SQL Server. Modify it as necessary to suite the database you are using.

Now that we have created a stored procedure, we will examine how to call it from within a C# application.

13.2.8 Calling a Stored Procedure

A very nice aspect of ADO.NET is that it allows the developer to call a stored procedure in almost the exact same way as a standard SQL statement.

1. Create a new C# Windows Application project.
2. From the Toolbox, drag and drop a DataGrid onto the Form. Resize it as necessary.
3. Double-click on the Form to generate the Form_Load event handler. Before entering any code, add "using System.Data.SqlClient" at the top of the file.

Enter the Following Code

```
private void Form1_Load(object sender, System.EventArgs e)

   {

        SqlConnection conn = new SqlConnection("Data

        Source=localhost;Database=Northwind;Integrated Security=SSPI");

        SqlCommand command = new SqlCommand("GetProducts", conn);

        SqlDataAdapter adapter = new SqlDataAdapter(command);

        DataSet ds = new DataSet();

        adapter.Fill(ds, "Products");

        this.dataGrid1.DataSource = ds;

        this.dataGrid1.DataMember = "Products";

   }
```

As you can see, calling a stored procedure in this example is exactly like how you would use SQL statements, only that instead of specifying the SQL statement, you specify the name of the stored procedure. Aside from that, you can treat it exactly the same as you would an ordinary SQL statement call with all the advantages of a stored procedure.

13.2.9 Specifying Parameters

Most of the time, especially when using non-queries, values must be supplied to the stored procedure at runtime. For instance, a @CategoryID parameter can be added to our GetProducts procedure in order to specify to retrieve only products of a certain category. In SQL Server, parameters are specified after the procedure name and before the AS clause.

Create procedure [dbo].[getproducts] (@categoryid int) as select productid, productname from products where categoryid = @categoryid

Parameters are enclosed within parenthesis with the parameter name first followed by the data type. If more than one parameter is accepted, they are separated by commas:

```
CREATE PROCEDURE [dbo].[SomeProcedure] (

        @Param1 int,

        @Param2 varchar(50),
```

Notes

```
    @Param3 varchar(50)
) AS
...
```

For our GetProducts example, if @CategoryID was supplied with the value 1, the query would equate to:

```
Select productid, productname from products where categoryid = 1
```

Which would select all the products that belong to CategoryID 1 or the Beverages category. To call the stored procedure, use Query Analyzer to execute:

```
exec GetProducts X
```

where X is the @CategoryID parameter passed to the stored procedure. To call the stored procedure from within a C# application using 1 as the @CategoryID parameter value, use the following code:

```
SqlConnection conn = new SqlConnection("Data
Source=localhost;Database=Northwind;Integrated Security=SSPI");
SqlCommand command = new SqlCommand("GetProducts", conn);
command.CommandType = CommandType.StoredProcedure;
command.Parameters.Add("@CategoryID", SqlDbType.Int).Value = 1;
SqlDataAdapter adapter = new SqlDataAdapter(command);
DataSet ds = new DataSet();
adapter.Fill(ds, "Products");
this.dataGrid1.DataSource = ds;
this.dataGrid1.DataMember = "Products";
```

Note that you must now specify the CommandType property of the SqlCommand object. The reason we did not do this in the first example was that it is not required if the stored procedure does not accept parameters. Of course, specifying the CommandType property even if it is not needed may improve readability. The next line actually combines two lines in one:

```
command.Parameters.Add("@CategoryID",
SqlDbType.Int);command.Parameters["@CategoryID"].Value = 1;
```

The first line of this segment specifies that the command object (which calls the GetProducts stored procedure) accepts a parameter named @CategoryID which is of type SqlDbType.Int. The type must be the same as the data type specified by the stored procedure. The second line of this code segment gives the parameter the value 1. For simplicity, especially when using more than one parameter, I prefer to combine to two lines into a single line:

```
command.Parameters.Add("@CategoryID", SqlDbType.Int).Value = 1;
```

The rest of the code is the same as in the previous example without parameters. As illustrated in the previous examples, ADO.NET takes a lot of pain out of database programming. Calling a stored procedure uses virtually the same code as using standard SQL and specifying parameters is a painless process.

13.2.10 Data Retrieval

Data Retrieval with stored procedures is the same (surprise!) as if using standard SQL. You can wrap a DataAdapter around the Command object or you can use a DataReader to fetch the data one row at a time. The previous examples have already illustrated how to use a DataAdapter and fill a DataSet. The following example shows usage of the DataReader:

```
SqlConnection conn = new SqlConnection("Data

Source=localhost;Database=Northwind;Integrated Security=SSPI");

SqlCommand command = new SqlCommand("GetProducts", conn);

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add("@CategoryID", SqlDbType.Int).Value = 1;

conn.Open();SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

     Console.WriteLine(reader["ProductName"]);

}

conn.Close();
```

Again, using either a DataAdapter or a DataReader against a query from a stored procedure is the same as specifying the SQL from within the code.

13.2.11 Inserting Data Using Parameters

Using other SQL statements such as INSERT, UPDATE or DELETE follow the same procedure. First, create a stored procedure that may or may not accept parameters, and then call the stored procedure from within the code supply the necessary values if parameters are needed. The following example illustrates how to insert a new user in a users table that has a username and password field.

```
CREATE PROCEDURE [dbo].[InsertUser] (

  @Username varchar(50),

 @Password varchar(50)

) AS

INSERT INTO Users VALUES(@Username, @Password)

string username = ... // get username from user

string password = ... // get password from user
```

Notes

```
SqlConnection conn = new SqlConnection("Data
Source=localhost; Database=MyDB;Integrated Security=SSPI");
SqlCommand command = new SqlCommand("InsertUser", conn);
command.CommandType = CommandType.StoredProcedure;
command.Parameters.Add("@Username", SqlDbType.VarChar).Value = username;
command.Parameters.Add("@Password", SqlDbType.VarChar).Value = password;
conn.Open();
int rows = command.ExecuteNonQuery();
conn.Close();
```

First, we retrieve the username and password information from the user. This information may be entered onto a form, through a message dialog or through some other method. The point is, the user specifies the username and password and the applicaton inserts the data into the database. Also notice that we called the ExecuteNonQuery() method of the Connection object. We call this method to indicate that the stored procedure does not return results for a query but rather an integer indicating how many rows were affected by the executed statement. ExecuteNonQuery() is used for DML statements such as INSERT, UPDATE and DELETE.

Caution We can test the value of rows to check if the stored procedure inserted the data successfully.

```
if (rows == 1)
{
   MessageBox.Show("Create new user SUCCESS!");
}
else
{
    MessageBox.Show("Create new user FAILED!");
}
```

We check the value of rows to see if it is equal to one. Since our stored procedure only did one insert operation and if it is successful, the ExecuteNonQuery() method should return 1 to indicate the one row that was inserted. For other SQL statements, especially UPDATE and DELETE statements that affect more than one row, the stored procedure will return the number of rows affected by the statement.

Delete from products where productid > 50

This will delete all products whose product ID is greater than 50 and will return the number of rows deleted.

Stored procedures offer developers a lot of flexibility with many features not available using standard SQL. ADO.NET allows us to use stored procedures in our applications seamlessly. The combination of these two allows us to create very powerful appliations rapidly.

13.2.12 Benefits of Stored Procedures

Why should you use stored procedures? Let's take a look at the key benefits of this technology:

- **Precompiled execution**. SQL Server compiles each stored procedure once and then reutilizes the execution plan. This results in tremendous performance boosts when stored procedures are called repeatedly.
- **Reduced client/server traffi**c. If network bandwidth is a concern in your environment, you'll be happy to learn that stored procedures can reduce long SQL queries to a single line that is transmitted over the wire.
- **Efficient reuse of code and programming abstraction.** Stored procedures can be used by multiple users and client programs. If you utilize them in a planned manner, you'll find the development cycle takes less time.
- **Enhanced security controls.** You can grant users permission to execute a stored procedure independently of underlying table permissions.

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 30000 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 article, 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 50000 transactions a second by moving to a better system with 16 processors and a SAN array of high speed disks. But above 50000 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

Contd...

Notes

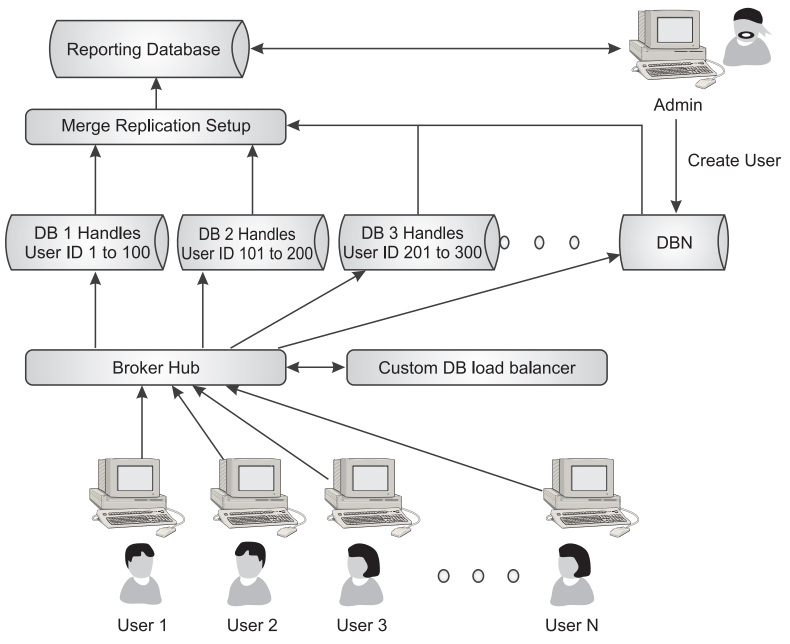
unlimited number of objects (2,147,483,647) and database size (524,272 TB). Again we could see the difference. Without enabling features like compression we could achieve a bigger number of around 70000 transactions per second.

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 30000 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. 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:



Contd...

Notes

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.

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.

Questions

1. Explain briefly broker hub.
2. What do you mean by DB Load balancer?

13.3 Summary

- A transaction is a sequence of operations performed as a single logical unit.
- A stored procedure is a set of SQL commands that has been compiled and stored on the database server.
- Stored procedures improve performance by reducing network traffic and CPU load.

Notes

13.4 Keywords

COMMIT: COMMIT command should be issued so that the changes to all involved tables will take effect.

Function: A function is a subprogram written to perform certain computations and return a single value.

ROLLBACK: ROLLBACK command should be issued to return every table referenced in the transaction to its previous state.

Stored Procedure: A stored procedure is a subroutine available to applications accessing a relational database system.

Transaction: A transaction is a sequential group of database manipulation operations, which is performed as if it were one single work unit.

Lab Exercise

1. Using SQL statements, create stored parameter.
2. Create a database of hospital management.

13.5 Self Assessment Questions

1. _______________ is a sequential group of database manipulation operations, which is performed as if it were one single work unit.

   (*a*) Stored procedure (*b*) Transaction

   (*c*) Linux (*d*) None of these.

2. COMMIT command should be issued to return every table referenced in the transaction to its previous state.

   (*a*) True (*b*) False

3. A _______________ is a subroutine available to applications accessing a relational database system.

   (*a*) Stored procedures (*b*) Transaction

   (*c*) SQL (*d*) None of these.

4. Functions must return a value.

   (*a*) True (*b*) False

5. _______________ are enclosed within parenthesis with the parameter name.

   (*a*) Functions (*b*) Statements

   (*c*) Parameters (*d*) None of these.

13.6 Review Questions

1. What do you mean by transactions?
2. Explain the following commands:

   (*a*) COMMIT

   (*b*) ROLLBACK

3. Define stored procedures.
4. What are the advantages and disadvantages of stored procedures?
5. Write steps how to create and call stored procedures.

Answers for Self Assessment Questions

1. (*b*) 2. (*b*) 3. (*a*) 4. (*a*) 5. (*c*)

13.7 Further Reading

Books

Open Source Development with LAMP: Using Linux, Apache, MySQL, Perl & PHP
By: James Lee, Pearson Education.

Online link

http://www.opensourcetutorials.com/

Unit 14: Connecting to MYSQL with PHP

CONTENTS

Objectives

After studying this unit, you will be able to:

- Discuss creating the connection.
- Discuss closing the connection.
- Understand with MySQL Data.

Introduction

PHP is becoming more and more popular in web programmers, mainly because it can be configured to connect to various databases like Oracle, MySQL, Solid and so on. But for a MS SQL server, the problem is different. Though you can use PHP's Sybase-ct support features to directly connect to MSSQ.

The extension requires the MSSQL Client Tools to be installed on the system where PHP is installed. The Client Tools can be installed from the MSSQL Server CD or by copying ntwdblib.dll from\winnt\system32 on the server to\winnt\system32 on the PHP box.

14.1 Connecting to MySQL with PHP

14.1.1 Creating the Connection

Opening a connection to MySQL database from PHP is easy. Just use the mysql_connect() function like this

```
 <?php
      $dbhost = 'localhost';
      $dbuser = 'root';
      $dbpass = 'password';
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die
('Error connecting to mysql');
$dbname = 'petstore';
mysql_select_db($dbname);
?>
```

$dbhost is the name of MySQL server. When your webserver is on the same machine with the MySQL server you can use localhost or 127.0.0.1 as the value of $dbhost. The $dbuser and $dbpass are valid MySQL user name and password.

Don't forget to select a database using my sql_select_db() after connecting to mysql. If no database selected your query to select or update a table will not work.

Sometimes a web host will require you to specify the MySQL server name and port number. For example if the MySQL server name is db.php-mysql-tutorial.com and the port number is 3306 (the default port number for MySQL) then you you can modify the above code to:

```
 <?php

      $dbhost = 'db.php-mysql-tutorial.com:3306';

      $dbuser = 'root';

      $dbpass = 'password';

      $conn = mysql_connect($dbhost, $dbuser, $dbpass) or die  ('Error
              connecting to mysql');

      $dbname = 'petstore';

      mysql_select_db($dbname);

  ?>
```

It's a common practice to place the routine of opening a database connection in a separate file. Then everytime you want to open a connection just include the file. Usually the host, user, password and database name are also separated in a configuration file.

An example of config.php that stores the connection configuration and opendb.php that opens the connection are:

Source code:

```
config.phps , opendb.phps

 <?php

      // This is an example of config.php

      $dbhost = 'localhost';

      $dbuser = 'root';

      $dbpass = 'password';

      $dbname = 'phpcake';

      ?>

  <?php

      // This is an example opendb.php

$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to mysql');

      mysql_select_db($dbname);

  ?>
```

So now you can open a connection to mysql like this:

```
 <?php

      include 'config.php';

      include 'opendb.php';

      // ... do something like insert or select, etc

 ?>
```

14.1.2 Closing the Connection

The connection opened in a script will be closed as soon as the execution of the script ends. But it's better if you close it explicitly by calling mysql_close() function. You could also put this function call in a file named closedb.php.

Source code: `closedb.phps`

```
 <?php
      // an example of closedb.php
      // it does nothing but closing
      // a mysql database connection

      mysql_close($conn);
 ?>
```

Now that you have put the database configuration, opening and closing routines in separate files your PHP script that uses mysql would look something like this :

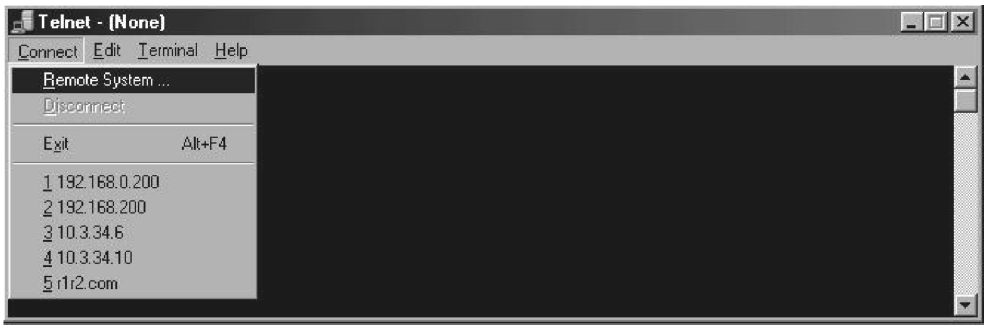
```
<?php
     include 'config.php';
     include 'opendb.php';

     // ... do something like insert or select, etc

     include 'closedb.php';
?>
```

14.2 Working with MySQL Data

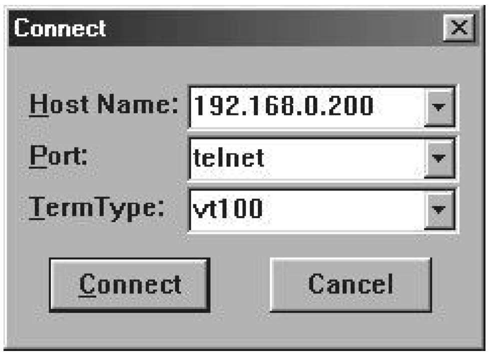
14.2.1 Connecting to the Server

As mentioned earlier, MySQL operates in client/server architecture. The client application needs to connect to database server, before manipulating the data.

For our examples, we will be using Telnet application to connect to the database server, and manipulate the database. In the examples provided, we are using Windows OS with Telnet client application. However, please note that there are other ways to connect to the database server. Initially, you need to call the telnet application by issuing "telnet" command from your DOS prompt. Now the telnet window pops up as shown below.



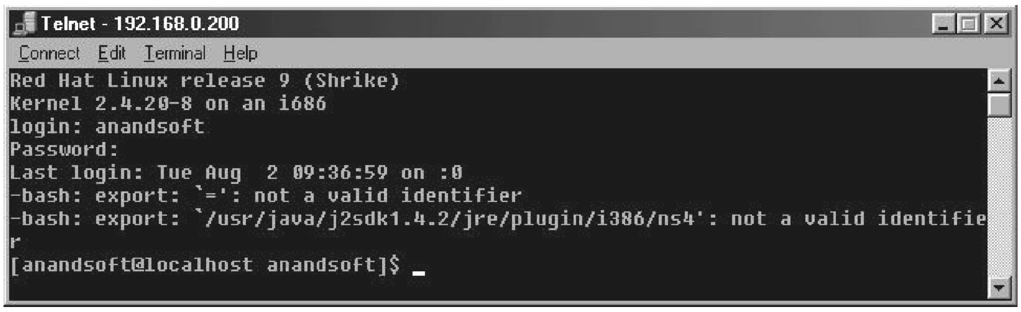
After clicking on the "Remote System", the following window will come where you can give the IP address of the server for connection



Host Name: 192.168.0.200
Port: telnet
TermType: vt100

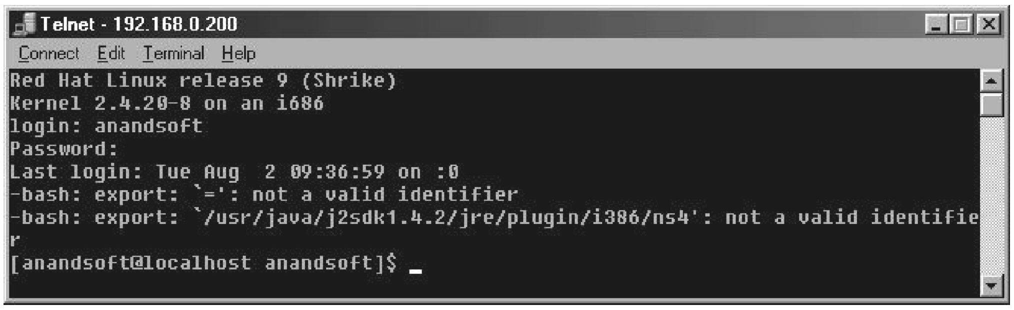
Then click the Connect button which will show the following window.

Notes



```
Red Hat Linux release 9 (Shrike)
Kernel 2.4.20-8 on an i686
login: anandsoft
Password:
Last login: Tue Aug  2 09:36:59 on :0
-bash: export: `=': not a valid identifier
-bash: export: `/usr/java/j2sdk1.4.2/jre/plugin/i386/ns4': not a valid identifie
r
[anandsoft@localhost anandsoft]$ _
```

In the above window enter your user name, and then enter the password when prompted. The following window will be shown with Linux shell prompt (if the server is a Linux system) if both user name and password are correct. Please note that the prompt depends on the username, and will vary from user to user.



```
Red Hat Linux release 9 (Shrike)
Kernel 2.4.20-8 on an i686
login: anandsoft
Password:
Last login: Tue Aug  2 09:36:59 on :0
-bash: export: `=': not a valid identifier
-bash: export: `/usr/java/j2sdk1.4.2/jre/plugin/i386/ns4': not a valid identifie
r
[anandsoft@localhost anandsoft]$ _
```

Now you are connected with the Linux server.

14.2.2 Connecting to the MySQL Server

Before you work with MySQL, ensure that you have a user name and password with appropriate permissions for connecting to and accessing the MySQL database.

The GRANT and REVOKE commands allow system administrators to create users and grant and revoke rights to MySQL users at four privilege levels:

Global level: The global privileges apply to all databases on a given server. These privileges are stored in the my sql.user table. REVOKE ALL ON *.* will revoke only global privileges.

Database level: Database privileges apply to all tables in a given database. These privileges are stored in the mysql.db and mysql.host tables. REVOKE ALL ON db.* will revoke only database privileges.

Table level: Table privileges apply to all columns in a given table. These privileges are stored in the my sql.tables_priv table. REVOKE ALL ON db.table will revoke only table privileges.

Column level: Column privileges apply to single columns in a given table. These privileges are stored in the my sql.columns_priv table. When using REVOKE you must specify the same columns that were granted.

Example: *mysql> GRANT ALL PRIVILEGES ON *.* TO name1@localhost IDENTIFIED BY 'pass' WITH GRANT OPTION;*

*mysql> GRANT ALL PRIVILEGES ON *.* TO name1@"%" IDENTIFIED BY 'pass' WITH GRANT OPTION;*

The above commands will provide the user "name1" with superuser permissions. The user can connect from anywhere.

If you give a grant for a users that doesn't exists, 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>

% indicates the shell prompts

mysql is the client program

<options> include the following:

-h host_name -u user_name -p password

-u user_name -p (if host is localhost)

In our example, we have the following information:

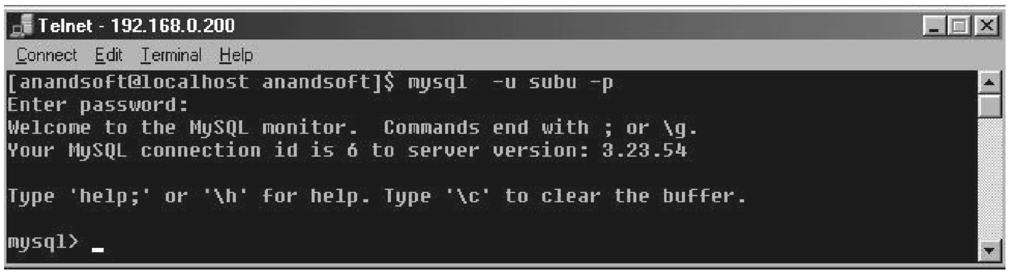
- host :localhost
- user_name :subu
- password :subu
- database :sample_db

Given the above, to connect to the database server use the command:

[anand soft@localhost anandsoft]$ mysql -u subu -p

Then enter your password *subu* at the password prompt.

The following screenshot depicts the above example:



```
[anandsoft@localhost anandsoft]$ mysql  -u subu -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 3.23.54

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> _
```

Now you are connected to the database server.

The connection can be terminated by giving QUIT at the *mysql* prompt.

mysql>QUIT

Bye

Notes

14.2.3 Issuing Queries

After you are connected to the server you are ready to issue queries.

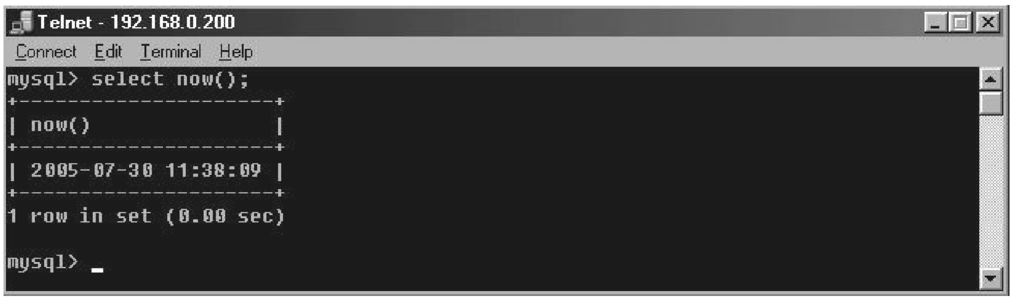
In MySQL the keywords and functions can be in uppercase or lowercase.

But the database name and table name must be in proper case as in Unix system the files and directives are case sensitive.

To enter a query in mysql, just type it, end it with a semicolon(;) and press enter. The semicolon tells mysql that the query is complete. You can also use '\g' to terminate queries.

Examples and results of some simple query is given below:

mysql>SELECT NOW();



```
mysql> select now();
+---------------------+
| now()               |
+---------------------+
| 2005-07-30 11:38:09 |
+---------------------+
1 row in set (0.00 sec)

mysql> _
```

Caution When you invoke a function in query, there must be no space between function name and following parenthesis.

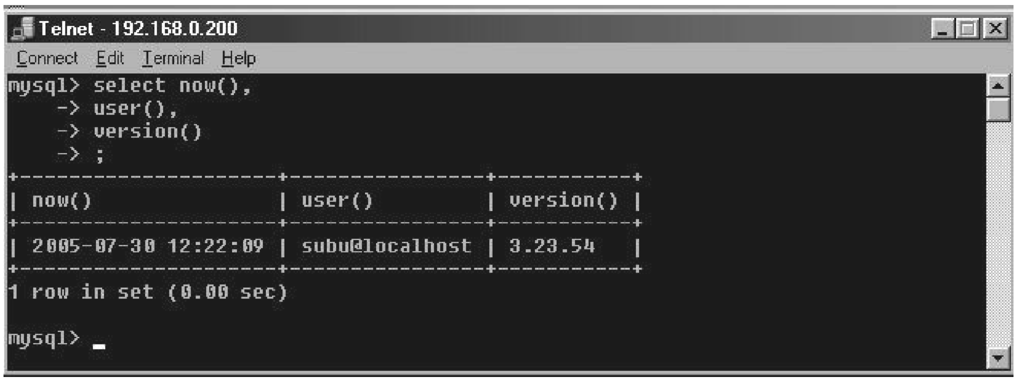
As my sql waits for the semicolon before sending the query to the server, you don't need to enter it on a single line. You can spread a query over several lines if you want:

mysql>SELECT NOW(),

->USER(),

->VERSION()

->;



```
mysql> select now(),
    -> user(),
    -> version()
    -> ;
+---------------------+----------------+-----------+
| now()               | user()         | version() |
+---------------------+----------------+-----------+
| 2005-07-30 12:22:09 | subu@localhost | 3.23.54   |
+---------------------+----------------+-----------+
1 row in set (0.00 sec)

mysql> _
```

Notes

Notes The prompt changes from 'mysql' to '->'after you enter the first line of query.

If you have begun typing in a multiple-line query and decide you don't want to execute it, type '\c' to clear(cancel) it.

```
mysql>SELECT NOW(),
->USER(),
->\c
mysql>
```

Notes The prompt changes back to mysql> to indicate that mysql is ready for new query.'c' is always lowercase.

14.2.4 Creating a Database

The first step in database management, is to create a database. The following steps are demonstrated using a database sample_db:

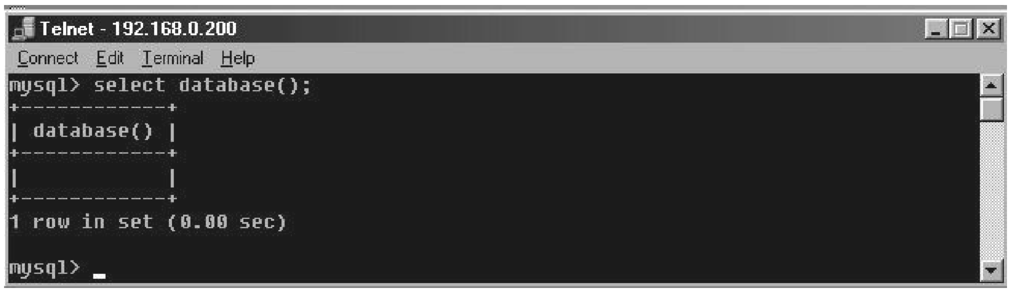
1. Creating (initializing) the database.
2. Creating the tables within the database.
3. Interacting with the tables by inserting, retrieving, modifying, or deleting data.

After connection to the server issue the following query to create database by name sample_db

```
mysql>CREATE DATABASE sample_db;
```

Now, a database by name sample_db is created, but still not in use. You need to issue USE <database-name> command to perform any operations on the database. SELECT DATABASE() command can be used to view the database in use as shown below:

```
mysql> SELECT DATABASE();
```



```
mysql> select database();
+------------+
| database() |
+------------+
|            |
+------------+
1 row in set (0.00 sec)

mysql> _
```

To make the sample_db as the current database in use, issue the command:

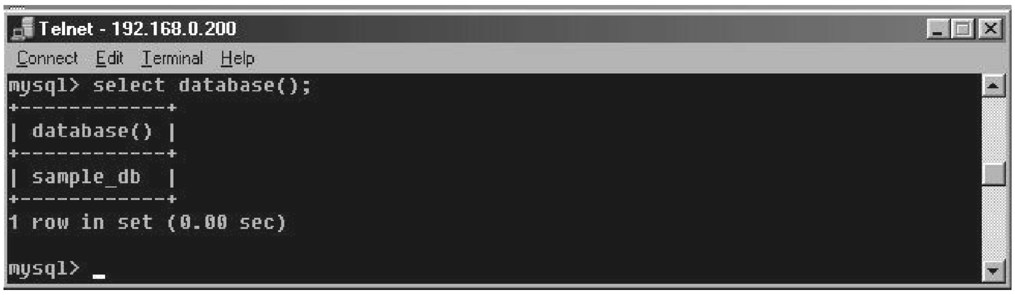
```
mysql>USE sample_db
```

Notes

Did u know? Use is one of the few statements that require no terminating semicolon, although you can give if you want.

After you issue the use statement, sample_db is the default database:

mysql>SELECT DATABASE();



```
mysql> select database();
+------------+
| database() |
+------------+
| sample_db  |
+------------+
1 row in set (0.00 sec)

mysql> _
```

The other way to make a database current is to name it on command line during connection to the server as follows:

% mysql -u subu -p sample_db

The available databases could be viewed by issuing the command:

mysql>SHOW DATABASES;

14.2.5 Removing a Database

You can remove it by the following query:

mysql>drop database sample_db;

The command will permanently remove the database.

14.2.6 Creating Tables

The CREATE TABLE statement allows you to create a table within the current database.

Syntax for creating table:

mysql>CREATE TABLE table_name(column_specs);

- table_name indicates the name you want to give the table.
- column_specs provides the specifications for the columns in the table, as well as indexes (if you have any).

Each column specification in the create table statement consists of the column name, the type (like varchar, int, date, etc.) and possibly some column attributes.

Did u know? A table must have at least one column. You cannot create a table without specifying any column name.

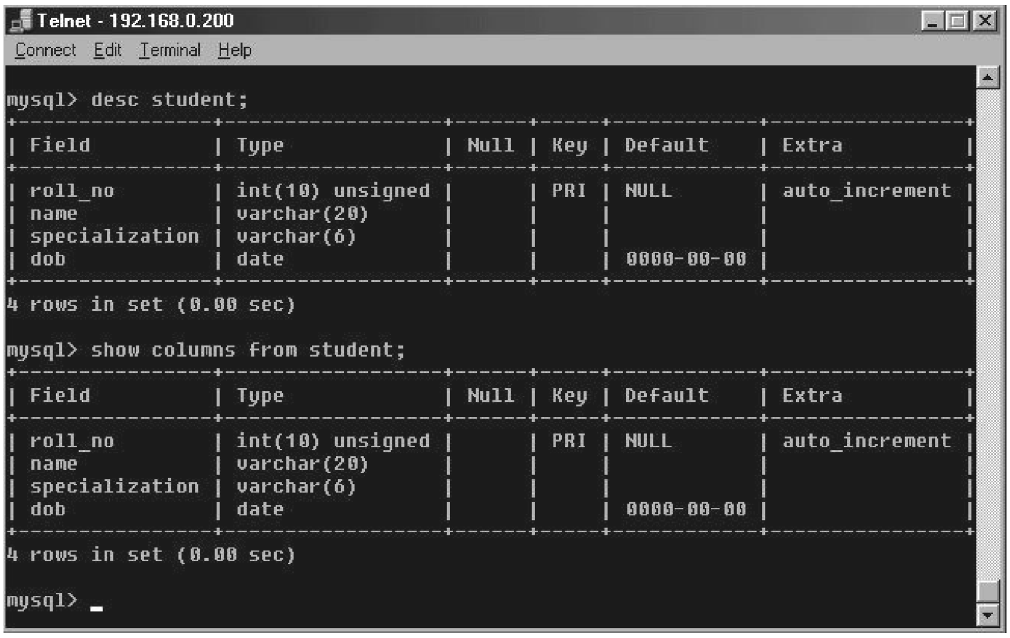
Now we can create a table having name student and four fields having name as roll_no, name, specialization, dob(date of birth).

The CREATE TABLE statement for the student table look like this

```
mysql>CREATE TABLE student
        (
        roll_no  INT  UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
        name VARCHAR(20) NOT NULL,
        specialization VARCHAR(6) NOT NULL,
        dob DATE NOT NULL);
```

In the above insert statement:

- INT signifies that the column holds integers (value with no fractional part).
- UNSIGNED disallows negative numbers.
- NOT NULL means that the column value must be filled in. (No student can be without a roll number).
- AUTO_INCREMENT works like this: If the value for the roll_no column is missing (or NULL) when you create a new student table record, MySQL automatically generates a unique number that is one greater than the maximum value currently in the column.
- PRIMARY KEY means each value in the column must be unique. This prevents us for using the roll number twice by mistake, which is desirable property for student roll number. (Not only that, but MySQL requires every AUTO_INCREMENT column have a unique index).
- VARCHAR(n) means the column contains variable-length character values, with a maximum length of n characters.
- Column type DATE holds the value in the format "YYYY-MM-DD"(for example," 1983-10-24")



```
mysql> desc student;
+----------------+------------------+------+-----+------------+----------------+
| Field          | Type             | Null | Key | Default    | Extra          |
+----------------+------------------+------+-----+------------+----------------+
| roll_no        | int(10) unsigned |      | PRI | NULL       | auto_increment |
| name           | varchar(20)      |      |     |            |                |
| specialization | varchar(6)       |      |     |            |                |
| dob            | date             |      |     | 0000-00-00 |                |
+----------------+------------------+------+-----+------------+----------------+
4 rows in set (0.00 sec)

mysql> show columns from student;
+----------------+------------------+------+-----+------------+----------------+
| Field          | Type             | Null | Key | Default    | Extra          |
+----------------+------------------+------+-----+------------+----------------+
| roll_no        | int(10) unsigned |      | PRI | NULL       | auto_increment |
| name           | varchar(20)      |      |     |            |                |
| specialization | varchar(6)       |      |     |            |                |
| dob            | date             |      |     | 0000-00-00 |                |
+----------------+------------------+------+-----+------------+----------------+
4 rows in set (0.00 sec)

mysql> _
```

Notes

After creating a table you can see the structure of that table by DESC statement or SHOW COLUMNS FROM table_name

i.e.

mysql>DESC student;

or

mysql>SHOW COLUMNS FROM student;

Did u know? If you happen to forget the name of any tables inside your database, you can see it by giving the following query:
mysql>SHOW TABLES;

Notes You can create primary key by combining two or more fields during table creation by the using the following query:

CREATE TABLE table_name (col1_name type NOT NULL, col2_name type NOT NULL,....., primary key (col1, col2)).

The two fields combining which you want to make a primary key cannot be NULL.

Here type signifies data type of the field.

14.2.7 Inserting Data into the Table

The INSERT INTO statement allows you to insert data into a table.

Syntax for insertion is:

mysql>INSERT INTO table_name values(value1,value2,....);

>table_name indicates the name of the table.

>value1,value2.... are the number of values same as the number of columns in the table _name specified

If you want to insert values into few fields instead of whole record, you can achieve this by the following query:

mysql>INSERT INTO table_name(col1,col2,col3) values(value1,value2,value3);

or

mysql>insert into table_name set col1=value1,col2=value2,col3=value3...

Did u know? Any column not named in the set clause is assigned a default value.

Another method of loading records into a table is to read the data values directly from a file. You can load records using load data statement.

The load data statement acts as a bulk loader that reads data from a file.

Syntax is:

mysql>LOAD DATA LOCAL INFILE filename INTO TABLE table_name;

Notes By default, the load data statement assumes that column values are separated by tabs and that lines end with new lines. It also assumes that the values are present in the that columns are stored in the table. "filename" should present in the user home directory.

Now you can insert some data into the student table using the above described INSERT statement.

mysql>INSERT INTO student VALUES('11','Subhransu Patra','cse','1983-6-3');

mysql>INSERT INTO student(roll_no,name,specialization) VALUES('12','Sudhansu Patra','etc');

mysql>INSERT INTO student SET name='Suvransu',specialization='ee';

mysql>INSERT INTO student VALUES('14','Jonny','etc','1982-6-2');

mysql>INSERT INTO student VALUES('15','Missy','ee','1981-5-4');

mysql>INSERT INTO student VALUES('16','Jenny','cse','1982-5-7');

mysql>INSERT INTO student VALUES('17','Billy','etc','1984-5-4');

mysql>INSERT INTO student VALUES('18','Kyle','cse','1983-7-6');

mysql>INSERT INTO student VALUES('19','Nathan','ee','1982-2-5');

mysql>INSERT INTO student VALUES('20','Abby','cse','1984-9-8');

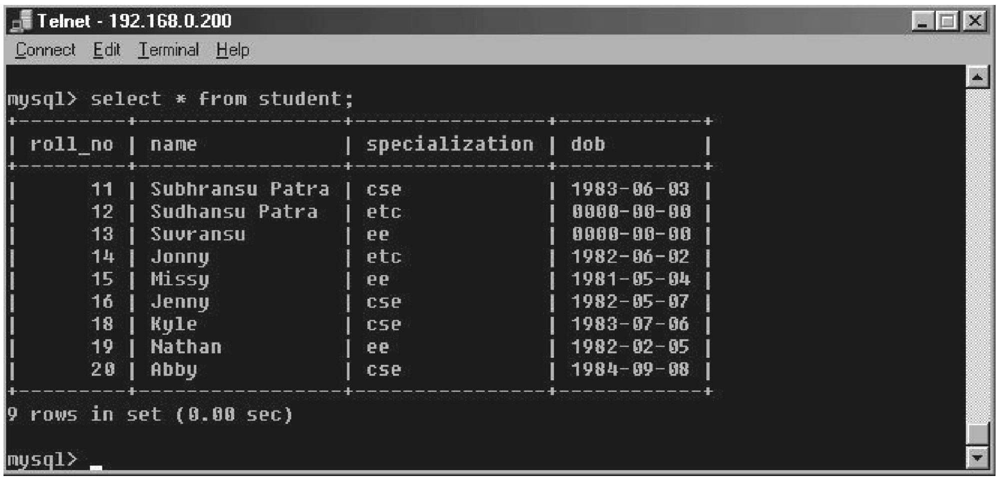
14.2.8 Retrieving Information from a Table

The SELECT statement allows you to retrieve and display information from your table.

The general form of SELECT is:

mysql>SELECT <fields-to-select>

FROM <table or tables>



```
mysql> select * from student;
+---------+-----------------+----------------+------------+
| roll_no | name            | specialization | dob        |
+---------+-----------------+----------------+------------+
|      11 | Subhransu Patra | cse            | 1983-06-03 |
|      12 | Sudhansu Patra  | etc            | 0000-00-00 |
|      13 | Suvransu        | ee             | 0000-00-00 |
|      14 | Jonny           | etc            | 1982-06-02 |
|      15 | Missy           | ee             | 1981-05-04 |
|      16 | Jenny           | cse            | 1982-05-07 |
|      18 | Kyle            | cse            | 1983-07-06 |
|      19 | Nathan          | ee             | 1982-02-05 |
|      20 | Abby            | cse            | 1984-09-08 |
+---------+-----------------+----------------+------------+
9 rows in set (0.00 sec)

mysql> _
```

Notes

WHERE <conditions that data must satisfy>;

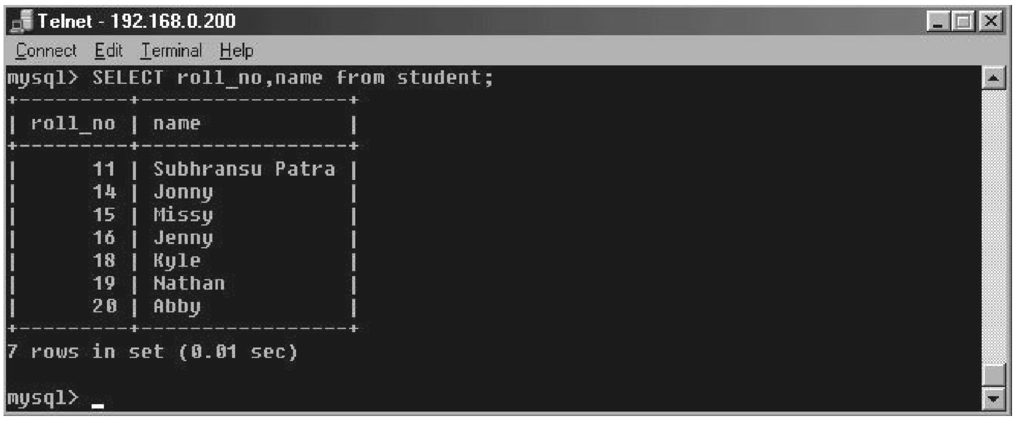
You can see the contents of student table as shown below by the following query:

mysql>SELECT * FROM student;

Here * signifies all. You can also retrieve specific field those you want.

Suppose you want to see only roll number and the name of students. The following query does this

mysql>SELECT roll_no,name from student;



```
mysql> SELECT roll_no,name from student;
+---------+-----------------+
| roll_no | name            |
+---------+-----------------+
|      11 | Subhransu Patra |
|      14 | Jonny           |
|      15 | Missy           |
|      16 | Jenny           |
|      18 | Kyle            |
|      19 | Nathan          |
|      20 | Abby            |
+---------+-----------------+
7 rows in set (0.01 sec)

mysql> _
```

14.2.9 Editing and Deleting Records

Changing some of the field values, or even deleting some records is part of any database maintenance. Two frequently used commands for doing the same are UPDATE and DELETE statements (respectively).

The DELETE statement has this form:

DELETE FROM <table_name> WHERE <records to delete>

The WHERE clause specifies which records to be deleted. It's optional but if you leave it out, all records are deleted from the table specified.

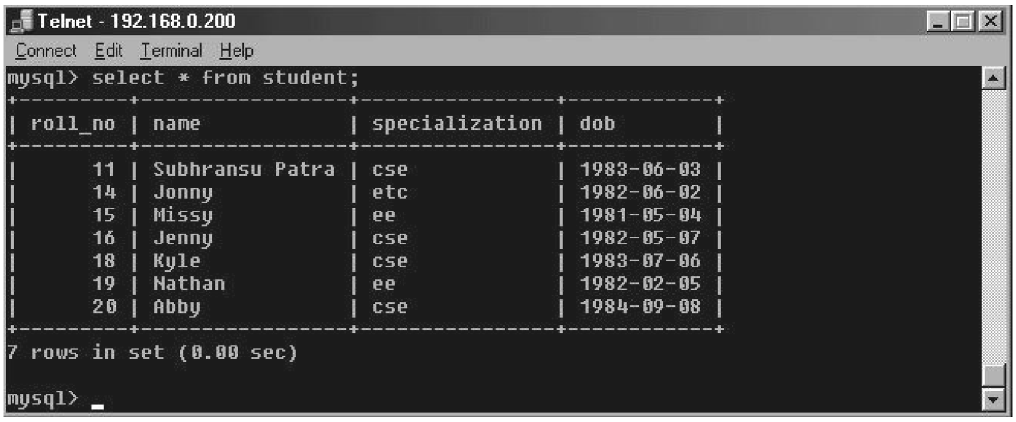
i.e. "DELETE FROM <table_name>" will delete all records from table table_name.

Now, suppose you want to delete records of those student who don't have date of birth, then you can issue the following command:

mysql>DELETE FROM student WHERE dob="0000-00-00";

After the execution of above DELETE statement you can see the contents by giving the above SELECT statement as below:

mysql>SELECT * FROM student;



```
mysql> select * from student;
+---------+-----------------+----------------+------------+
| roll_no | name            | specialization | dob        |
+---------+-----------------+----------------+------------+
|      11 | Subhransu Patra | cse            | 1983-06-03 |
|      14 | Jonny           | etc            | 1982-06-02 |
|      15 | Missy           | ee             | 1981-05-04 |
|      16 | Jenny           | cse            | 1982-05-07 |
|      18 | Kyle            | cse            | 1983-07-06 |
|      19 | Nathan          | ee             | 1982-02-05 |
|      20 | Abby            | cse            | 1984-09-08 |
+---------+-----------------+----------------+------------+
7 rows in set (0.00 sec)

mysql> _
```

TO modify existing records, use UPDATE which has this form:

UPDATE table_name SET which columns to change WHERE which records to update.

Here also the WHERE clause is optional, if you don’t specify one, every records in the table is updated.

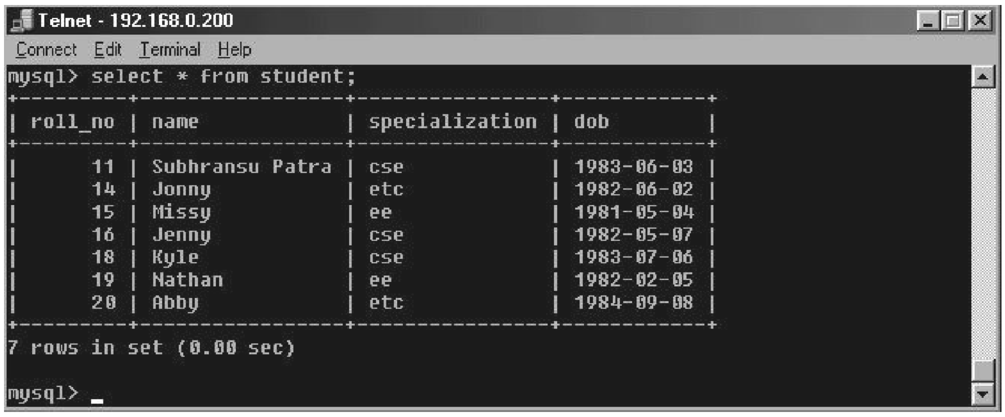
i.e., UPDATE table_name SET which columns to change

for example you can change the specialization of a student whose roll number is 20, to etc. from cse.

The following query fulfills the above change:

mysql>UPDATE student SET specialization=”etc” where roll_no=”20”;

After the execution of above query the contents of the table becomes:



```
mysql> select * from student;
+---------+-----------------+----------------+------------+
| roll_no | name            | specialization | dob        |
+---------+-----------------+----------------+------------+
|      11 | Subhransu Patra | cse            | 1983-06-03 |
|      14 | Jonny           | etc            | 1982-06-02 |
|      15 | Missy           | ee             | 1981-05-04 |
|      16 | Jenny           | cse            | 1982-05-07 |
|      18 | Kyle            | cse            | 1983-07-06 |
|      19 | Nathan          | ee             | 1982-02-05 |
|      20 | Abby            | etc            | 1984-09-08 |
+---------+-----------------+----------------+------------+
7 rows in set (0.00 sec)

mysql> _
```

14.2.10 Altering the Structure of Tables

Using ALTER statement you can add fields to a existing table.

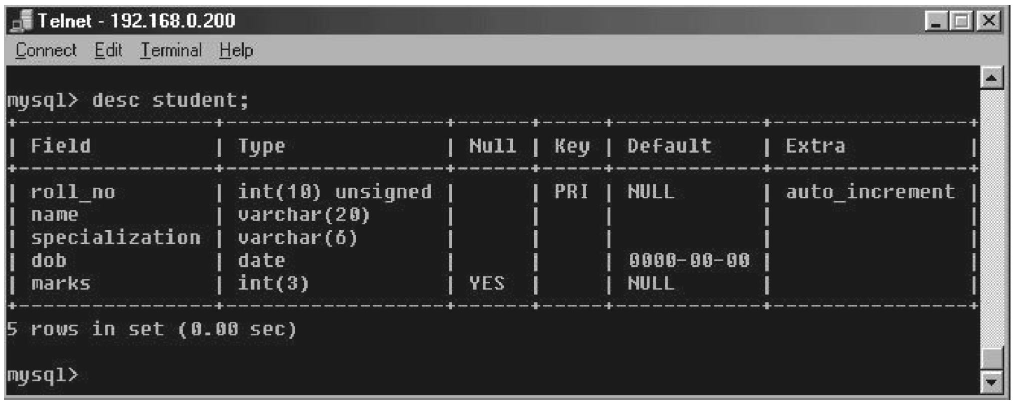
The general form of ALTER statement is:

ALTER TABLE table_name ADD (column specs);

Suppose you want to add another field as marks to the student table for storing students mark. Then the query becomes

Notes mysql>ALTER TABLE student add marks int(3);

Then the table structure becomes:



```
mysql> desc student;
+----------------+------------------+------+-----+------------+----------------+
| Field          | Type             | Null | Key | Default    | Extra          |
+----------------+------------------+------+-----+------------+----------------+
| roll_no        | int(10) unsigned |      | PRI | NULL       | auto_increment |
| name           | varchar(20)      |      |     |            |                |
| specialization | varchar(6)       |      |     |            |                |
| dob            | date             |      |     | 0000-00-00 |                |
| marks          | int(3)           | YES  |     | NULL       |                |
+----------------+------------------+------+-----+------------+----------------+
5 rows in set (0.00 sec)

mysql>
```

Using ALTER statement you can change the data type of a column and the name of an existing table.

Syntax for changing the data types of a column.

ALTER TABLE table_name MODIFY column_name type.

or

ALTER TABLE table_name CHANGE column_name new column_name type.

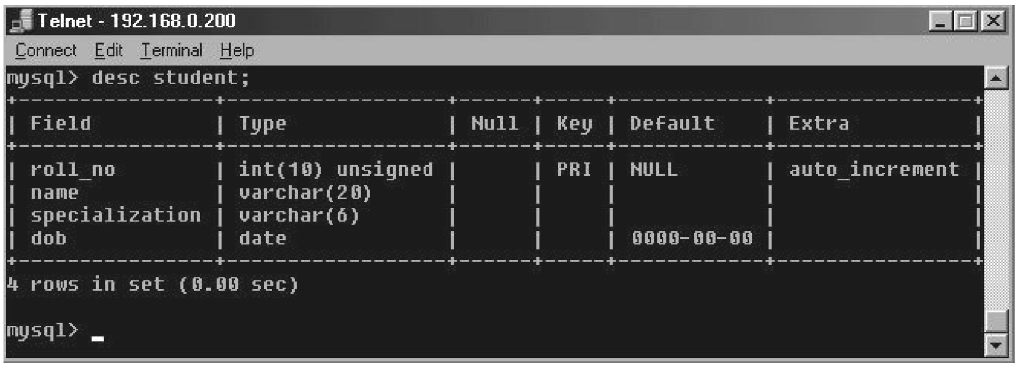
Notes The difference between MODIFY and CHANGE is that, in case of CHANGE you can change name of column which is not possible by using MODIFY that's why change takes two names?

Syntax for changing the table name:

ALTER TABLE table_name RENAME AS new_table_name

Using ALTER statement you can remove a column from a table:

Syntax is:



```
mysql> desc student;
+----------------+------------------+------+-----+------------+----------------+
| Field          | Type             | Null | Key | Default    | Extra          |
+----------------+------------------+------+-----+------------+----------------+
| roll_no        | int(10) unsigned |      | PRI | NULL       | auto_increment |
| name           | varchar(20)      |      |     |            |                |
| specialization | varchar(6)       |      |     |            |                |
| dob            | date             |      |     | 0000-00-00 |                |
+----------------+------------------+------+-----+------------+----------------+
4 rows in set (0.00 sec)

mysql> _
```

ALTER TABLE table_name DROP COLUMN col_name;

Suppose you want to drop field marks, then you can give the following query:

mysql>ALTER TABLE student DROP COLUMN marks;

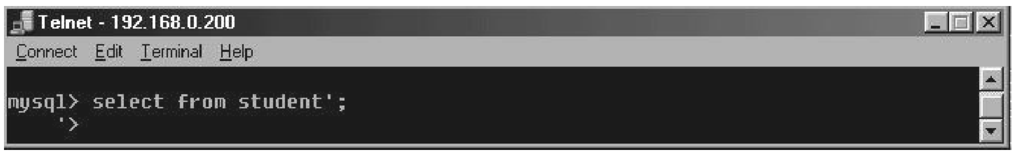
14.2.11 Dropping a Table

The difference between DROP and DELETE table is that, after executing DELETE statement the contents of table are removed but the structure remains same, but in case of DROP statement both the contents and structure are removed.

Syntax for DROP statement is:

mysql>DROP TABLE table_name;

During issuing query if you put a single quote(') or double quote(") inside a query you must have to end somewhere with single quote or double quote otherwise an error will be thrown (as shown below) because mysql will think as receiving a string until the quote ends with another quote. **Anything** inside that two quote is treated as string.



```
mysql> select from student';
    '>
```

14.2.12 Working with NULL Value

When the value of a field is NULL you cannot compare in the same way as doing for NOT NULL value, if you do so you will not get the desired result.

For NULL value comparison you may follow the following procedure:

mysql>SELECT * FROM table_name WHERE field_name is NULL;

14.2.13 Backing up a Database

You can take a backup of your database in a text file by using the mysqldump command from shell prompt as given below

[anand soft@localhost anandsoft]$mysqldump -u subu -psubu sample_db>sample.sql

After the execution of the above command sample.sql file will contain the structure as well as the data insertion statements done on sample_db database.

Notes Sample.sql file is stored in the user home directory, i.e. the user name under which you logged in to the server(not MySQL server). For example if your username is anandsoft, in Linux system the file sample.sql will be stored in the directory/home/anandsoft/

you can only take the structure of the tables by giving the following command:

[anand soft@localhost anandsoft]$mysqldump -d -u subu -psubu sample_db>sample1.sql

You can take back up of any specific table from a database by following way:

Notes

[anand soft@localhost anandsoft]$mysqldump -u subu -psubu database_name table_name>file_name

The mysqldump command is a very useful, and frequently used for taking a backup of an existing database. Another use of this command is when you want to transfer the database from a local server to a remote server effortlessly. For example, you have created a database on your local server and tested the program. Now you want to upload the same to an Internet server. Take an sql dump of the local database by using the *mysqldump* command and paste the file contents on the remote server. (Alternatively, you can also recreate the database by specifying the dump file name).

Task Create a table in a database and insert, retrieve, edit and delete records in it.

Case Study

The History of PHP

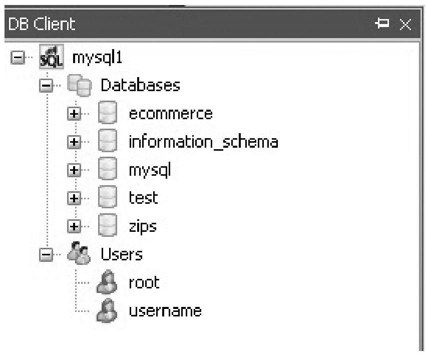
PHP is an "HTML-embedded scripting language" primarily used for dynamic Web applications. The first part of this definition means that PHP code can be interspersed with HTML, making it simple to generate dynamic pieces of Web pages on the fly. As a scripting language, PHP code requires the presence of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

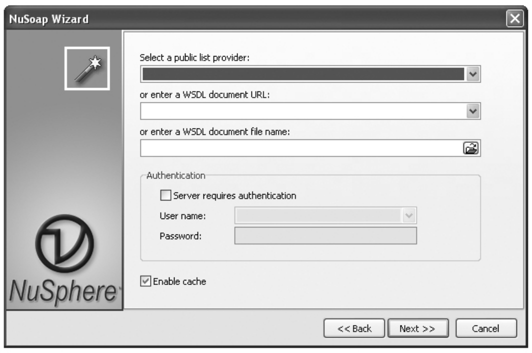
PHP 5 was released in July 2004, with the updated Zend Engine 2.0.

Among the many new features in PHP 5 are:

- Improved object-oriented programming
- Embedded SQLite
- Support for new MySQL features (see the image at right)

Contd...





Select a public list provider:

or enter a WSDL document URL:

or enter a WSDL document file name:

Authentication

☐ Server requires authentication

User name:

Password:

☑ Enable cache

- Exception handling using a try..catch structure
- Integrated SOAP support (see the image at right)
- The Filter library (in PHP 5.1)
- Better XML tools
- Iterators

and much, much more.

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

Questions

1. Give all version name of PHP.

2. Give the year of funding PHP 6.

14.3 Summary

- The client application needs to connect to database server, before manipulating the data.
- The load data statement acts as a bulk loader that reads data from a file.
- The mysql dump command is used for taking a backup of an existing database.

Notes

14.4 Keywords

Column Level: Column privileges apply to single columns in a given table.

Database Level: Database privileges apply to all tables in a given database.

Global Level: The global privileges apply to all databases on a given server.

Table Level: Table privileges apply to all columns in a given table.

$ dhost: $ dhost is the name of MySQL server. When your webserver is on the same machine with the MySQL server you can use localhost or 127.0.0.1 as the values of $ dhost.

Lab Exercise

1. Using SQL statements, create stored parameter.
2. Create a database of hospital management.

14.5 Self Assessment Questions

1. Data Reader command is used to fetch the data one row at a time.

   (*a*) True (*b*) False

2. _____________ operates in client/server architecture.

   (*a*) MySQL (*b*) Telnet

   (*c*) PHP (*d*) DOS

3. The _____________ privileges apply to all databases on a given server.

   (*a*) database (*b*) GRANT

   (*c*) global (*d*) table

4. The _____________ statement allows you to create a table within the current database.

   (*a*) edit table (*b*) create table

   (*c*) open table (*d*) none of these.

5. A table must have at least _____________ column.

   (*a*) five (*b*) two

   (*c*) four (*d*) one

14.6 Review Questions

1. Write steps to opening and closing a connection to MySQL database.
2. Explain Linux system.
3. What are the steps to create database?
4. What is the difference between DROP and DELETE table?
5. What do you mean by Backing up a Database?

Answers for Self Assessment Questions

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

14.7 Further Reading

Books *Open Source Development with LAMP: Using Linux, Apache, MySQL, Perl & PHP*
By: James Lee, Pearson Education.

Online link http://www.opensourcetutorials.com/

LOVELY PROFESSIONAL UNIVERSITY
Jalandhar-Delhi G.T. Road (NH-1)
Phagwara, Punjab (India)-144411
For Enquiry: +91-1824-521360
Fax.: +91-1824-506111
Email: odl@lpu.co.in

978-93-90164-72-1

9 789390 164721