

# Installing and configuring MySQL

MIS 4530

Dr. Garrett

# Installing and Configuring MySQL on UNIX and Linux

1. Go to <http://dev.mysql.com/downloads/> and download the latest version of MySQL
2. Create a separate group and user named for running MySQL:  

```
groupadd mysql  
useradd -g mysql mysql
```
3. Run the `gunzip mysql-4.1.9.tar.gz` command
4. Run the `tar xvf mysql-4.1.9.tar` command
5. Change to the `mysql-4.1.9` directory

# Installing and Configuring MySQL on UNIX and Linux (continued)

6. Run the `./configure` command
7. Compile the MySQL code with the `make` command
8. Run the `make install` command
9. Change to the scripts directory
10. Run the `mysql_install_db --user=mysql` script
11. Run the ownership commands:  

```
chown -R root /usr/local/mysql  
chown -R mysql /usr/local/mysql/var  
chgrp -R mysql /usr/local/mysql
```

# Installing and Configuring MySQL on Windows

1. Go to **<http://dev.mysql.com/downloads/>**
2. Open Windows Explorer or My Computer and start the MySQL installation
3. In the Welcome screen, click **Next** to start the installation
4. Accept the default setup type **Typical**, click **Next**
5. Click **Back** to make changes or click **Install** to continue

# Installing and Configuring MySQL on Windows (continued)

6. Create a new account or skip sign-in, click **Next**
7. In the Wizard Completed screen, click **Finish**
8. In the first screen of the MySQL Server Instance Configuration Wizard, click **Next**
9. In the Configuration Type screen, select **Standard Configuration**, click **Next**

# Installing and Configuring MySQL on Windows (continued)

10. In the Windows Options screen, accept the default values (***do not select*** the Include Bin Directory in Windows PATH check box), click **Next**
11. In the Security Options screen, deselect the **Modify Security Settings** check box, click **Next**
12. Click **Back** to change any of the configuration operations or **Execute** to finish

# Testing the MySQL Server

1. Check to see if MySQL is running

- For UNIX/Linux systems:

```
/usr/local/mysql/bin/mysqld_safe --user=mysql &
```

- For Windows, use the Services window

2. Run the `mysqladmin version` command

- For UNIX/Linux systems:

```
/usr/local/mysql/bin/mysqladmin version
```

- For Windows, change to the C:\Program Files\MySQL\MySQL Server 4.1\bin\ directory and run:  
`mysqladmin version`

# Creating Basic PHP Scripts

- **Embedded scripting languages** (JavaScript or PHP) refer to code that is embedded within a Web page (either an HTML or XHTML document)
- This code is typed directly into a Web page as a separate section
- A Web page document containing PHP code must have an extension of `.php`
- PHP code is never sent to a client's Web browser

# Creating Basic PHP Scripts (continued)

- The Web page generated from the PHP code, and HTML or XHTML elements found within the PHP file, is returned to the client
- A PHP file that does not contain any PHP code should have an **.html** extension
- **.php** is the default extension that most Web servers use to process PHP scripts

# Creating PHP Code Blocks

- **Code declaration blocks** are separate sections within a Web page that are interpreted by the scripting engine
- There are four types of code declaration blocks:
  - Standard PHP script delimiters
  - The `<script>` element
  - Short PHP script delimiters
  - ASP-style script delimiters

# Standard PHP Script Delimiters

- A **delimiter** is a character or sequence of characters used to mark the beginning and end of a code segment
- The standard method of writing PHP code declaration blocks is to use the `<?php` and `?>` script delimiters
- The individual lines of code that make up a PHP script are called **statements**

# The `<script>` Element

- The **`<script>`** element identifies a script section in a Web page document
- For client-side scripting, the `type` attribute of the `<script>` element indicates which scripting language and version is being used
- When the `<script>` element is used with PHP, you do not include the `type` attribute

# Short PHP Script Delimiters

- The syntax for the short PHP script delimiters is  
`<? statements; ?>`
- Short delimiters can be disabled in a Web server's php.ini configuration file
- PHP scripts will not work if your Web site ISP does not support short PHP script delimiters
- Short delimiters can be used in XHTML documents, but not in XML documents

# ASP-Style Script Delimiters

- The syntax for the ASP-style script delimiters is  
`<% statements; %>`
- ASP-style script delimiters can be used in XHTML documents, but not in XML documents
- ASP-style script delimiters can be enabled or disabled in the `php.ini` configuration file
- To enable or disable ASP-style script delimiters, assign a value of “On” or “Off” to the `asp_tags` directive in the `php.ini` configuration file

# Understanding Functions

- A **function** refers to a procedure that performs a specific task
- To execute a function, you must invoke, or **call**, it from somewhere in the script
- A **function call** is the function name followed by any data that the function needs
- The data (in parentheses following the function name) are called **arguments** or **actual parameters**
- Sending data to a called function is called **passing arguments**

# Displaying Script Results

- To return to the client the results of any processing that occurs within a PHP code block, you must use an `echo ()` statement or the `print ()` statement
- The **`echo ()`** and **`print ()`** statements create new text on a Web page that is returned as a response to a client

# Displaying Script Results (continued)



The screenshot shows a Mozilla Firefox browser window titled "phpinfo() - Mozilla Firefox". The address bar displays "http://localhost/PHP\_Projects/Chapter.01/Chapter/PHPTest". The page content includes the PHP logo and the text "PHP Version 5.0.3". Below this is a table of system and configuration information.

System	Windows NT DGOSSELI-LAP 5.1 build 2600
Build Date	Dec 15 2004 08:06:41
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS\php.ini
PHP API	20031224
PHP Extension	20041030
Zend Extension	220040412
Debug Build	no
Thread Safety	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, http, ftp, compress, zlib
Registered Stream Socket Transports	tcp, udp

At the bottom of the page, it states: "This program makes use of the Zend Scripting Language Engine: Zend Engine v2.0.3, Copyright (c) 1998-2004 Zend Technologies". To the right of this text is the "Powered By Zend Engine 2" logo.

Figure 2-13 PHP Diagnostic Information Web page

# Displaying Script Results (continued)

- The `echo()` and `print()` statements are language constructs of the PHP programming language
- A **programming language construct** refers to a built-in feature of a programming language
- The `echo()` and `print()` statements are virtually identical except:
  - The `print()` statement returns a value of 1 if it is successful
  - It returns a value of 0 if it is not successful

## Displaying Script Results (continued)

- Use the `echo ()` and `print ()` statements to return the results of a PHP script within a Web page that is returned to a client
- A **text string**, or **literal string**, is text that is contained within double or single quotation marks
- To pass multiple arguments to the `echo ()` and `print ()` statements, separate them with commas like arguments passed to a function

# Creating Multiple Code Declaration Blocks

- For multiple script sections in a document, include a separate code declaration block for each section

```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<?php echo "<p>Output from the first script section.</p>";
?>
<h2>Second Script Section</h2>
<?php echo "<p>Output from the second script section.</p>"
;?>
</body>
</html>
```

# Creating Multiple Code Declaration Blocks (continued)

- PHP code declaration blocks execute on a Web server before a Web page is sent to a client

...

```
</head>
```

```
<body>
```

```
<h1>Multiple Script Sections</h1>
```

```
<h2>First Script Section</h2>
```

```
<p>Output from the first script section.</p>
```

```
<h2>Second Script Section</h2>
```

```
<p>Output from the second script section.</p>
```

```
</body>
```

```
</html>
```

# Creating Multiple Code Declaration Blocks (continued)



**Figure 2-17** Output of a document with two PHP script sections

# Case Sensitivity in PHP

- Programming language constructs in PHP are mostly case **insensitive**

```
<?php
```

```
echo "<p>Explore <strong>Africa</strong>, <br />";
```

```
Echo "<strong>South America</strong>, <br />";
```

```
ECHO " and <strong>Australia</strong>!</p>";
```

```
?>
```

# Adding Comments to a PHP Script

- **Comments** are nonprinting lines placed in code such as:
  - The name of the script
  - Your name and the date you created the program
  - Notes to yourself
  - Instructions to future programmers who might need to modify your work

# Adding Comments to a PHP Script (continued)

- **Line comments** hide a single line of code
  - Add // or # before the text
- **Block comments** hide multiple lines of code
  - Add /\* to the first line of code
  - And \*/ after the last character in the code

# Adding Comments to a PHP Script (continued)

```
<?php
/*
This line is part of the block comment.
This line is also part of the block comment.
*/
echo "<h1>Comments Example</h1>"; // Line comments can follow
code statements
// This line comment takes up an entire line.
# This is another way of creating a line comment.
/* This is another way of creating
a block comment. */
?>
```