MODULES

Week 8
Week 8 Tasks

1. Assignment: Read Topic: Working with Files and Directories
2. Read "Additional Reading" materials (please scroll down the page).
3. Lab Work: For instructions on what to submit for lab work, please refer to "Tasks, Tests and Surveys", "Lab 2."

Please gather all of Lab 2 work and submit it on or before Saturday of this week.

Please post all your questions, answers, and helpful tips in the Discussion area.

If you know the answers to your classmate's questions in the Discussion area, please feel free to post your answers.

Additional Reading

Please note: Web sites can change over time, and as a result some of the links below may be inactive since the publication of this course. If this is the case, try searching for a similar Web site using Yahoo!, Google, or another Internet search engine.

http://www.softwareprojects.org/php-files-15.htm

http://www.onlamp.com/pub/a/php/2002/10/03/php_foundations.html

http://www.informit.com/articles/article.asp?p=29664&seqNum=2

http://www.devarticles.com/c/a/PHP/Working-With-Text-Files-in-PHP/1

Lecture Notes:

In this chapter, you will learn how to store files on your local computer. Keep in mind that your ability to access files on a local computer or network depends on the security permissions you have been granted to the files and to the directories where they are stored. Check with your operating system's documentation or IT staff member for information on how to set permissions for resources such as files and directories.

Opening and Closing File Streams

A stream is a channel that is used for accessing a resource that you can read from write to. For example you might use a stream to access a file. The input stream reads data from a resource (such as a file), whereas the output stream writes data to a resource.

Using a file stream involves the following steps:

1. Open the file stream with the fopen() function.
2. Write data to or read data from the file stream.
3. Close the file stream with the fclose() function.

Opening a File Stream

A handle is a special type of variable that PHP uses to represent a resource such as a file. You use the fopen() function to open a handle to a file stream. The syntax for the fopen() function is open_file = ("text file" , "mode"); . The open_file variable is the handle that you can use to read and write data from and to the file. The mode argument can be one of several values that determine what you can with the file after you open it.

A file pointer is a special type of variable that refers to the currently selected line or character in a file. The file pointer is a way of keeping track of where you are in a file.

Closing a File Stream

When you are finished working with a file stream, you use the statement fclose($handle); to ensure that the file doesn't keep taking up space in your computer's memory.

Writing Data to Files

PHP supports two basic functions for writing data to text files: the file_put_contents() function and the fwrite() function.

Writing an Entire File

The file_put_contents() function writes or appends a text string to a file. The syntax for the file_put_contents() function is file_put_contents(filename, string[,options]). If the specified filename does not exist, it is created. However, if the specified filename does exist, any data it contains is overwritten. With the file_put_contents() function, you do not need to use the fopen() and fclose() functions. Instead, you simply call the file_put_contents() function and pass to it the name of the file to which you want to write data along with a text string containing the data you want to write.

Handling Magic Quotes

Because the data a user submits to a PHP script may contains single or double quotes, you should use escape sequences for any user data your script receives, especially before you write it to a data source, such as a text file or database. PHP includes a feature called magic quotes, which automatically adds a backslash (\) to any single quote (‘), double quote (*), or NULL character contained in data that a user submits to a PHP script. Magic quotes are enabled within your php.ini configuration file. By default, the magic_quotes_gpc directive is the only magic quote directive that is enabled in your php.ini configuration file when you first install PHP.

Writing Data Incrementally

The file_put_contents() function is useful if you want to quickly replace the contents of a file or append data to the end of an existing file. You can also use the fwrite() function to incrementally write data to a text file. To write data to the text file with the fwrite() function, you use the following syntax: fwrite($handle, data[,length]);. As with the file_put_contents() function, the fwrite() function returns the number of bytes that were written to file. If no data was written to the file, the function returns a value of 0.

Locking Files

When your program opens a text file via the fopen() method, there is a chance that another program will attempt to open the same file. If both programs are simply reading data from a file, there should be no problem. However, if more than one program attempts to write data to a text file at the same time, data corruption could occur. To prevent multiple users from modifying a file simultaneously, you need to use the flock() function. The basic syntax for the flock() function is flock($handle, operation).

Reading Data from Files

PHP includes a number of different functions for reading data from text files. These functions can be generally classified as functions that read an entire file or functions that read the contents of a file incrementally.

Reading an Entire File

The file_get_contents() function reads the entire contents of a file into a string. The file(filename[,use_include_path]) function reads the contents of a file into an indexed array. The fread($handle, length) function reads the contents of a file into a string up to a maximum number of bytes. The readfile(filename[, use_include_path]) function prints the contents of a file.

Reading Data Incrementally

For larger text files reading the entire contents of the file into PHP can take up a lot of memory on your server and affect the performance of your script. The function fgetc($handle) returns a single character and moves the file pointer to the next character. The function fgetcsv($handle, length[,delimiter, string_enclosure]) returns a line, parses the line for CSV fields, and then moves the file pointer to the next line. The fgets($handle[, length] function returns a line and moves the file pointer to the next line. The fgetss($handle, length[, allowed_tags]) returns a line, strips any HTML tags the line contains, and then moves the file pointer to the next line. The stream_get_line($handle, length, delimiter) function returns a line that ends with a specified delimiter and moves the file pointer to the next line.

Managing Files and Directories

To iterate through the entries in a directory, you open a handle to the directory with the opendir() function. You can then use the readdir() function to return the file and directory names from the open directory. Each time you call the readdir() function, it moves a directory pointer to the next entry in the directory. After the directory pointer reaches the end of the directory, the readdir() function returns a value of false.

You can use the mkdir() function to create a new directory. To create a new directory within the current directory, you pass just the name of the directory you want to create to the mkdir() function.

Obtaining File and Directory Information

To successfully work with files and directories, you need to be able to obtain information about them. The file_exists(filename) function determines whether a file or directory exists. The is_dir(filename) function determines whether a filename is a directory. The is_executable(filename) function determines whether a file is executable. The is_file(filename) function determines whether a file is a regular file. The is_readable(filename) function determines whether a file is readable. The is_writeable(filename) function determines whether a file is writeable. Each of these functions returns a value of true in the file exists and if the given status is true.

-

Copying and Moving Files

You can use the copy() function to copy a file with PHP. The function returns a value of true if it is successful or false if it is not. The syntax for the copy() function is copy(source, destination).

Renaming Files and Directories

You can use the rename() function to rename a file or directory with PHP. As with the copy() function, the rename() function returns a value of true if it is successful or false if it is not. The syntax for the rename() function is rename(old_name, new_name).

Removing Files and Directories

You use the unlink() function to delete files and the rmdir() function to delete directories. You pass the name of a file to the unlink() function and the name of a directory to the rmdir() function. Both functions return a value of true if successful or false if not.

****

file_put_contents:

When you use file_put_contents, there is an extra line at the end that shows up as a blank line when doing a sort. What is this blank line is all about? How do we get it not to be a part of the sort?

Answer:

A similar problem can happen with 'explode' statement as well. Having the separator at the end of each element added to the file was causing a blank to be inserted into end when the string was exploded to an array. It seems that the algorithm finds the separator and then considers everything until the next instance of the separator as an element. So a separator at the end of the string causes the additional element to be created.

Place the separator at the front of an element added to the file, rather than at the end. For example, "\n".<element> rather than <element>."\n". However, this must be tested for when the first element is added to the file, so a "\n" is not placed at

(c) 2008, 2009 Etudes, Inc. All rights reserved.
ETUDES - 2.3.x - Sakai 2.3.x - Server "athena69"