Working with Files

The data storage destination of choice for a web application is a database. That doesn't mean that you're completely off the hook from dealing with regular old files, though. Plain text files are still a handy, universal way to exchange some kinds of information. You can do some easy customization of your web site by storing HTML templates in text files. When it's time to generate a specialized page, load the text file, substitute real data for the template elements, and print it. Files are also good for importing or exporting tabular data between your program and a spreadsheet. In your PHP programs, you can easily read and write the CSV ("comma separated value") files with which spreadsheet programs work. Working with files in PHP also means working with remote web pages. A great thing about file handling in PHP is you can open a remote file on another computer as easily as you can open a file that sits on your web server. Most file-handling functions in PHP understand URLs as well as local filenames. However, for this feature to work, the allow_url_fopen configuration directive must be enabled. It is enabled by default, but if you're having problems loading a remote file, check this setting.

Understanding File Permissions

To read or write a file with any of the functions you'll learn about in this chapter, the PHP interpreter must have permission from the operating system to do so. Every program that runs on a computer, including the PHP interpreter, runs with the privileges of a particular user account. Most of the user accounts correspond to people. When you log in to your computer and start up your word processor, that word processor runs with the privileges that correspond to your account: it can read files that you are allowed to see and write files that you are allowed to change. Some user accounts on a computer, however, aren't for people, but for system processes such as web servers. When the PHP interpreter runs inside of a web server, it has the privileges that the web server's "account" has. So, if the web server is allowed to read a certain file or directory, then the PHP interpreter (and therefore, your PHP program) can read that file or directory. If the web server is allowed to change a certain file or write new files in a particular directory, then so can the PHP interpreter and your PHP program. Usually, the privileges extended to a web server's account are more limited than the privileges that go along with a real person's account. The web server (and the PHP interpreter) needs to be able to read all of the PHP program files that make up your web site, but they shouldn't be able to change them. If a bug in the web server or an insecure PHP program lets an attacker break in, the PHP program files should be protected against being changed by that attacker.

In practice, what this means is that your PHP programs shouldn't have too much trouble reading most files that you need to read. (Of course, if you try to read another user's private files, you may run into a problem but that's as it should be!) However, the files that your PHP program can change and the directories into which your program can write new files are limited.

Table: File modes for fopen()

Mode	Description
'r'	Open for reading only; place the file pointer at the beginning of the file.
'r+'	Open for reading and writing; place the file pointer at the beginning of the file.
'w'	Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'w+'	Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'a'	Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
'a+'	Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

Reading and Writing Entire Files

This section shows you how to work with an entire file at once, as opposed to manipulating just a few lines of a file. PHP provides special functions for reading or writing a whole file in a single step.

Reading a File

To read the contents of a file into a string, use **file_get_contents()**. Pass it a filename, and it returns a string containing everything in the file.

Writing a File

The counterpart to reading the contents of a file into a string is writing a string to a file. And the counterpart to **file_get_contents()** is **file_put_contents()**.

Note: For reading and writing a file examples refer to the code folders.

Inspecting File Permissions

As mentioned at the beginning of the chapter, your programs can only read and write files when the PHP interpreter has permission to do so. You don't have to cast about blindly and rely on error messages to figure out what those permissions are, however. PHP gives you functions with which you can determine what your program is allowed to do.

To check whether a file or directory exists, use file_exists( ). This function is used to report whether a directory's index file has been created.

Note: To determine whether your program has permission to read or write a particular file, use is_readable( ) or is_writeable( ).

Checking for Errors

Checking error in our file helps keeping our code shorter, so you can focus on the file manipulation functions such as file_get_contents( ), fopen( ), and fgetcsv( ). It also makes them somewhat incomplete. Just like talking to a database program, working with files means interacting with resources external to your program. This means you have to worry about all sorts of things that can cause problems, such as operating system file permissions or a disk running out of free space.

In practice, to write robust file-handling code, you should check the return value of each file related function. They each generate a warning message and return false if there is a problem.

Note: For example, refer to the folder checking-for-folders.