# Working with Files

CHAPTER 9

# Working with files

- A web application is typically uses a database to store data
- But using plain text files is also beneficial
  - For example:
    - *storing an HTML template in a file and reading it and printing by PHP*
    - *reading and writing the CSV (commaseparated value) files*

# Reading an **Entire** Files

- To read the contents of a file into a string, use file_get_contents()

```
<html>
<head><title>{page_title}</title></head>
<body bgcolor="{color}">

<h1>Hello, {name}</h1>

</body>
</html>
```

```
// Load the template file from the previous example
$page = file_get_contents('page-template.html');

// Insert the title of the page
$page = str_replace('{page_title}', 'Welcome', $page);

// Make the page blue in the afternoon and
// green in the morning
if (date('H')>= 12 ) {
    $page = str_replace('{color}', 'blue', $page);
} else {
    $page = str_replace('{color}', 'green', $page);
}

// Take the username from a previously saved session
// variable
$page = str_replace('{name}', $_SESSION['username'], $page);

// Print the results
print $page;
```

# Writing a **Entire** File

- To write the content of a string into a file use file_put_contents()

```php
// Write the results to page.html
file_put_contents('page.html', $page);
```

- The file_get_contents() and file_put_contents() functions are fine when you want to work with an entire file at once.

# file() function

- You can use the file() function to access each line of a file.
- The file() function returns an array. The array elements are a string containg one line of the file, newline included
- Very convenient, but problematic with very large files.

*Example 9-5. people.txt for Example 9-4*

```
alice@example.com|Alice Liddell
bandersnatch@example.org|Bandersnatch Gardner
charles@milk.example.com|Charlie Tenniel
dodgson@turtle.example.com|Lewis Humbert
```

```php
foreach (file('people.txt') as $line) {
    $line = trim($line);
    $info = explode('|', $line);
    print '<li><a href="mailto:' . $info[0] . '">' . $info[1] ."</li>\n";
}
```

```
<li><a href="mailto:alice@example.com">Alice Liddell</li>
<li><a href="mailto:bandersnatch@example.org">Bandersnatch Gardner</li>
<li><a href="mailto:charles@milk.example.com">Charlie Tenniel</li>
<li><a href="mailto:dodgson@turtle.example.com">Lewis Humbert</li>
```

# Understanding File Permissions

- To read or write a file, the PHP engine must have permission from the OS to do so
- Every file is assigned some permissions
- Every user or an application has an account and is assigned some permissions
- Web servers containing PHP Engine has an account with specific permissons too
- Web server's account privileges are more limited than the users privileges
- The web server (and the PHP engine) need to be able to read all of website files, but shouldn't be able to change them.

# fopen(), feof(), and fgets() functions

- fopen() opens file or URL and binds it to a stream

```php
$fh = fopen('people.txt','rb');
while ((! feof($fh)) && ($line = fgets($fh))) {
    $line = trim($line);
    $info = explode('|', $line);
    print '<li><a href="mailto:' . $info[0] . '">' . $info[1] ."</li>\n";
}
fclose($fh);
```

- The mode parameter specifies the type of access you require to the stream. It may be any of values on the rigth:

# File modes

- Use 'b' to force binary mode, which will not translate your data.
- To use it, specify 'b' as the last character of the mode parameter, for example: **wb** or **rb**

- R: read
- W: write
- A: append
- X: exist
- C: create
- +: read+write

- PHP: fopen - Manual

| Mode | Allowable actions | Position bookmark starting point | Clear contents? | If the file doesn't exist? |
|---|---|---|---|---|
| `rb` | Reading | Beginning of file | No | Issue a warning, return `false`. |
| `rb+` | Reading, Writing | Beginning of file | No | Issue a warning, return `false`. |
| `wb` | Writing | Beginning of file | Yes | Try to create it. |
| `wb+` | Reading, writing | Beginning of file | Yes | Try to create it. |
| `ab` | Writing | End of file | No | Try to create it. |
| `ab+` | Reading, writing | End of file | No | Try to create it. |
| `xb` | Writing | Beginning of file | No | Try to create it; if the file does exist, issue a warning and return `false`. |
| `xb+` | Reading, writing | Beginning of file | No | Try to create it; if the file does exist, issue a warning and return `false`. |
| `cb` | Writing | Beginning of file | No | Try to create it. |
| `cb+` | Reading, writing | Beginning of file | No | Try to create it. |

# fopen(), feof(), fgets(), and fclose() functions

- fgets() — Gets a line or a length byte from file pointer
- fgets(resource $handle, int $length = ?): string|false
- The file pointer must be valid, and must point to a file successfully opened by fopen()

- feof() — Tests for end-of-file on a file pointer

```
feof(resource $stream): bool
```

- Returns true if the file pointer is at EOF or an error occurs; otherwise returns false.

```php
$fh = fopen('people.txt','rb');
while ((! feof($fh)) && ($line = fgets($fh))) {
    $line = trim($line);
    $info = explode('|', $line);
    print '<li><a href="mailto:' . $info[0] . '">' . $info[1] ."</li>\n";
}
fclose($fh);
```

# fwrite()

- Once you've opened a file in a mode that allows writing, use the fwrite() function to write something to the file.
- The fwrite() function doesn't automatically add a newline to the end of the string you write.

```php
<?php
$filename = 'test.txt';
$somecontent = "Add this to the file\n";

// Let's make sure the file exists and is writable first.
if (is_writable($filename)) {

    // In our example we're opening $filename in append mode.
    // The file pointer is at the bottom of the file hence
    // that's where $somecontent will go when we fwrite() it.
    if (!$handle = fopen($filename, 'a')) {
         echo "Cannot open file ($filename)";
         exit;
    }

    // Write $somecontent to our opened file.
    if (fwrite($handle, $somecontent) === FALSE) {
        echo "Cannot write to file ($filename)";
        exit;
    }

    echo "Success, wrote ($somecontent) to file ($filename)";

    fclose($handle);

} else {
    echo "The file $filename is not writable";
}
?>
```

# Working with CSV Files

- To read a line of a CSV file, use fgetcsv()
- It reads a line from the CSV file and returns an array containing each field in the line.

```
fgetcsv(
    resource $stream,
    int $length = 0,
    string $separator = ",",
    string $enclosure = '"',
    string $escape = "\\"
): array
```

```php
<?php
$row = 1;
if (($handle = fopen("test.csv", "r")) !== FALSE) {
    while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
        $num = count($data);
        echo "<p> $num fields in line $row: <br /></p>\n";
        $row++;
        for ($c=0; $c < $num; $c++) {
            echo $data[$c] . "<br />\n";
        }
    }
    fclose($handle);
}
?>
```

- The fputcsv() function takes a file handle and an array of values as arguments and writes those values, formatted as proper CSV, to the file.

# Checking for Errors

- To check whether a file or directory exists, use file_exists()

```
if (file_exists('/usr/local/htdocs/index.html')) {
    print "Index file is there.";
} else {
    print "No index file in /usr/local/htdocs.";
}
```

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

```
$template_file = 'page-template.html';
if (is_readable($template_file)) {
    $template = file_get_contents($template_file);
} else {
    print "Can't read template file.";
}
```

# Checking for Errors

To write robust file-handling code, you should check the return value of each file-related function.

```php
$fh = fopen('people.txt','rb');
if (! $fh) {
    print "Error opening people.txt: $php_errormsg";
} else {
    while (! feof($fh)) {
        $line = fgets($fh);
        if ($line !== false) {
            $line = trim($line);
            $info = explode('|', $line);
            print '<li><a href="mailto:' . $info[0] . '">' . $info[1] ."</li>\n";
        }
    }
    if (! fclose($fh)) {
        print "Error closing people.txt: $php_errormsg";
    }
}
```

# Excercises

1. Outside of the PHP engine, create a new template file in the style of Example 9-1. Write a program that uses `file_get_contents()` and `file_put_contents()` to read the HTML template file, substitute values for the template variables, and save the new page to a separate file.
2. Outside of the PHP engine, create a file that contains some email addresses, one per line. Make sure a few of the addresses appear more than once in the file. Call that file *addresses.txt*. Then, write a PHP program that reads each line in *addresses.txt* and counts how many times each address appears. For each distinct address in *addresses.txt*, your program should write a line to another file, *addresses-count.txt*. Each line in *addresses-count.txt* should consist of the number of times an address appears in *addresses.txt*, a comma, and the email address. Write the lines to *addresses-count.txt* in sorted order from the address that occurs the most times in *addresses.txt* to the address that occurs the fewest times in *addresses.txt*.

# Excercises

3. Display a CSV file as an HTML table. If you don't have a CSV file (or spreadsheet program) handy, use the data from Example 9-9.
4. Write a PHP program that displays a form that asks a user for the name of a file underneath the web server's document root directory. If that file exists on the server, is readable, and is underneath the web server's document root directory, then display the contents of the file. For example, if the user enters *article.html*, display the file *article.html* in the document root directory. If the user enters *catalog/show.php*, display the file *show.php* in the directory *catalog* under the document root directory. Table 7-1 tells you how to find the web server's document root directory.
5. Modify your solution to the previous exercise so that the program displays only files whose names end in *.html*. Letting users look at the PHP source code of any page on your site can be dangerous if those pages have sensitive information in them such as database usernames and passwords.