PHP File System

In this tutorial you will learn how to create, access (or read) and manipulate files dynamically using the PHP's file system functions.

Working with Files in PHP

Since PHP is a server side programming language, it allows you to work with files and directories stored on the web server. In this tutorial you will learn how to create, access, and manipulate files on your web server using the PHP file system functions.

Opening a File with PHP `fopen()` Function

To work with a file you first need to open the file. The PHP `fopen()` function is used to open a file. The basic syntax of this function can be given with:

```
fopen(filename, mode)
```

The first parameter passed to `fopen()` specifies the name of the file you want to open, and the second parameter specifies in which mode the file should be opened. For example:

Example

Run this code »

```php
<?php
$handle = fopen("data.txt", "r");
?>
```

The file may be opened in one of the following modes:

Modes	What it does
`r`	Open the file for reading only.
`r+`	Open the file for reading and writing.
`w`	Open the file for writing only and clears the contents of file. If the file does not exist, PHP will attempt to create it.
`w+`	Open the file for reading and writing and clears the contents of file. If the file does not exist, PHP will attempt to create it.
`a`	Append. Opens the file for writing only. Preserves file content by writing to the end of the file. If the file does not exist, PHP will attempt to create it.

a+	Read/Append. Opens the file for reading and writing. Preserves file content by writing to the end of the file. If the file does not exist, PHP will attempt to create it.
x	Open the file for writing only. Return `FALSE` and generates an error if the file already exists. If the file does not exist, PHP will attempt to create it.
x+	Open the file for reading and writing; otherwise it has the same behavior as 'x'.

If you try to open a file that doesn't exist, PHP will generate a warning message. So, to avoid these error messages you should always implement a simple check whether a file or directory exists or not before trying to access it, with the PHP `file_exists()` function.

Example

Run this code »

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to open the file
    $handle = fopen($file, "r");
} else{
    echo "ERROR: File does not exist.";
}
?>
```

Tip: Operations on files and directories are prone to errors. So it's a good practice to implement some form of error checking so that if an error occurs your script will handle the error gracefully. See the tutorial on PHP error handling.

Closing a File with PHP `fclose()` Function

Once you've finished working with a file, it needs to be closed. The `fclose()` function is used to close the file, as shown in the following example:

Example

Run this code »

```
<?php
$file = "data.txt";
```

```
// Check the existence of file
if(file_exists($file)){
    // Open the file for reading
    $handle = fopen($file, "r") or die("ERROR: Cannot open the
file.");

    /* Some code to be executed */

    // Closing the file handle
    fclose($handle);
} else{
    echo "ERROR: File does not exist.";
}
?>
```

Note: Although PHP automatically closes all open files when script terminates, but it's a good practice to close a file after performing all the operations.

Reading from Files with PHP `fread()` Function

Now that you have understood how to open and close files. In the following section you will learn how to read data from a file. PHP has several functions for reading data from a file. You can read from just one character to the entire file with a single operation.

Reading Fixed Number of Characters

The `fread()` function can be used to read a specified number of characters from a file. The basic syntax of this function can be given with.

```
fread(file handle, length in bytes)
```

This function takes two parameter — A file handle and the number of bytes to read. The following example reads 20 bytes from the "data.txt" file including spaces. Let's suppose the file "data.txt" contains a paragraph of text "The quick brown fox jumps over the lazy dog."

Example

Run this code »

```
<?php
$file = "data.txt";

// Check the existence of file
```

```
if(file_exists($file)){
    // Open the file for reading
    $handle = fopen($file, "r") or die("ERROR: Cannot open the
file.");

    // Read fixed number of bytes from the file
    $content = fread($handle, "20");

    // Closing the file handle
    fclose($handle);

    // Display the file content
    echo $content;
} else{
    echo "ERROR: File does not exist.";
}
?>
```

The above example will produce the following output:

```
The quick brown fox
```

Reading the Entire Contents of a File

The `fread()` function can be used in conjugation with the `filesize()` function to read the entire file at once. The `filesize()` function returns the size of the file in bytes.

Example

Run this code »

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Open the file for reading
    $handle = fopen($file, "r") or die("ERROR: Cannot open the
file.");

    // Reading the entire file
    $content = fread($handle, filesize($file));

    // Closing the file handle
    fclose($handle);

    // Display the file content
    echo $content;
} else{
    echo "ERROR: File does not exist.";
}
?>
```

The above example will produce the following output:

The quick brown fox jumps over the lazy dog.

The easiest way to read the entire contents of a file in PHP is with the `readfile()` function. This function allows you to read the contents of a file without needing to open it. The following example will generate the same output as above example:

Example

Run this code »

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Reads and outputs the entire file
    readfile($file) or die("ERROR: Cannot open the file.");
} else{
    echo "ERROR: File does not exist.";
}
?>
```

The above example will produce the following output:

The quick brown fox jumps over the lazy dog.

Another way to read the whole contents of a file without needing to open it is with the `file_get_contents()` function. This function accepts the name and path to a file, and reads the entire file into a string variable. Here's an example:

Example

Run this code »

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Reading the entire file into a string
    $content = file_get_contents($file) or die("ERROR: Cannot open
the file.");

    // Display the file content
    echo $content;
} else{
    echo "ERROR: File does not exist.";
}
?>
```

One more method of reading the whole data from a file is the PHP's `file()` function. It does a similar job to `file_get_contents()` function, but it returns the file contents as an array of lines, rather than a single string. Each element of the returned array corresponds to a line in the file.

To process the file data, you need to iterate over the array using a foreach loop. Here's an example, which reads a file into an array and then displays it using the loop:

Example

Run this code »

```
<?php
$file = "data.txt";

// Check the existence of file
if(file_exists($file)){
    // Reading the entire file into an array
    $arr = file($file) or die("ERROR: Cannot open the file.");
    foreach($arr as $line){
        echo $line;
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

Writing the Files Using PHP `fwrite()` Function

Similarly, you can write data to a file or append to an existing file using the PHP `fwrite()` function. The basic syntax of this function can be given with:

```
fwrite(file handle, string)
```

The `fwrite()` function takes two parameter — A file handle and the string of data that is to be written, as demonstrated in the following example:

Example

Run this code »

```
<?php
```

```
$file = "note.txt";

// String of data to be written
$data = "The quick brown fox jumps over the lazy dog.";

// Open the file for writing
$handle = fopen($file, "w") or die("ERROR: Cannot open the file.");

// Write data to the file
fwrite($handle, $data) or die ("ERROR: Cannot write the file.");

// Closing the file handle
fclose($handle);

echo "Data written to the file successfully.";
?>
```

In the above example, if the "note.txt" file doesn't exist PHP will automatically create it and write the data. But, if the "note.txt" file already exist, PHP will erase the contents of this file, if it has any, before writing the new data, however if you just want to append the file and preserve existing contents just use the mode `a` instead of `w` in the above example.

An alternative way is using the `file_put_contents()` function. It is counterpart of `file_get_contents()` function and provides an easy method of writing the data to a file without needing to open it. This function accepts the name and path to a file together with the data to be written to the file. Here's an example:

Example

Run this code »

```
<?php
$file = "note.txt";

// String of data to be written
$data = "The quick brown fox jumps over the lazy dog.";

// Write data to the file
file_put_contents($file, $data) or die("ERROR: Cannot write the
file.");

echo "Data written to the file successfully.";
?>
```

If the file specified in the `file_put_contents()` function already exists, PHP will overwrite it by default. If you would like to preserve the file's contents you can pass the special `FILE_APPEND` flag as a third parameter to the `file_put_contents()` function. It will simply append the new data to the file instead of overwitting it. Here's an example:

Example

Run this code »

```
<?php
$file = "note.txt";

// String of data to be written
$data = "The quick brown fox jumps over the lazy dog.";

// Write data to the file
file_put_contents($file, $data, FILE_APPEND) or die("ERROR: Cannot
write the file.");

echo "Data written to the file successfully.";
?>
```

Renaming Files with PHP `rename()` Function

You can rename a file or directory using the PHP's `rename()` function, like this:

Example

Run this code »

```
<?php
$file = "file.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to rename the file
    if(rename($file, "newfile.txt")){
        echo "File renamed successfully.";
    } else{
        echo "ERROR: File cannot be renamed.";
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

Removing Files with PHP `unlink()` Function

You can delete files or directories using the PHP's `unlink()` function, like this:

Example

Run this code »

```php
<?php
$file = "note.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to delete the file
    if(unlink($file)){
        echo "File removed successfully.";
    } else{
        echo "ERROR: File cannot be removed.";
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

In the next chapter we will learn more about parsing directories or folders in PHP.

PHP Filesystem Functions

The following table provides the overview of some other useful PHP filesystem functions that can be used for reading and writing the files dynamically.

Function	Description
`fgetc()`	Reads a single character at a time.
`fgets()`	Reads a single line at a time.
`fgetcsv()`	Reads a line of comma-separated values.
`filetype()`	Returns the type of the file.
`feof()`	Checks whether the end of the file has been reached.
`is_file()`	Checks whether the file is a regular file.
`is_dir()`	Checks whether the file is a directory.
`is_executable()`	Checks whether the file is executable.

`realpath()`	Returns canonicalized absolute pathname.
`rmdir()`	Removes an empty directory.

Please check out the PHP filesystem reference for other useful PHP filesystem functions.