PHP Parsing Directories

In this tutorial you will learn how to process directories or folders using PHP.

Working with Directories in PHP

In the previous chapter you've learned how to work with files in PHP. Similarly, PHP also allows you to work with directories on the file system, for example, you can open a directory and read its contents, create or delete a directory, list all files in the directory, and so on.

Creating a New Directory

You can create a new and empty directory by calling the PHP `mkdir()` function with the path and name of the directory to be created, as shown in the example below:

Example Run this code »

```php
<?php
// The directory path
$dir = "testdir";

// Check the existence of directory
if(!file_exists($dir)){
    // Attempt to create directory
    if(mkdir($dir)){
        echo "Directory created successfully.";
    } else{
        echo "ERROR: Directory could not be created.";
    }
} else{
    echo "ERROR: Directory already exists.";
}
?>
```

To make the `mkdir()` function work, the parent directories in the directory path parameter has to exist already, for example, if you specify the directory path as `testdir/subdir` than the `testdir` has to exist otherwise PHP will generate an error.

Copying Files from One Location to Another

You can copy a file from one location to another by calling PHP `copy()` function with the file's source and destination paths as arguments. If the destination file already exists it'll be overwritten. Here's an example which creates a copy of "example.txt" file inside backup folder.

Example Run this code »

```php
<?php
// Source file path
$file = "example.txt";

// Destination file path
$newfile = "backup/example.txt";

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

To make this example work, the target directory which is *backup* and the source file i.e. "example.txt" has to exist already; otherwise PHP will generate an error.

Listing All Files in a Directory

You can use the PHP `scandir()` function to list files and directories inside the specified path.

Now we're going to create a custom function that will recursively list all files in a directory using PHP. This script will be helpful if you're working with deeply nested directory structure.

Example Run this code »

```php
<?php
// Define a function to output files in a directory
function outputFiles($path){
    // Check directory exists or not
    if(file_exists($path) && is_dir($path)){
```

```
        // Scan the files in this directory
        $result = scandir($path);

        // Filter out the current (.) and parent (..) directories
        $files = array_diff($result, array('.', '..'));

        if(count($files) > 0){
            // Loop through retuned array
            foreach($files as $file){
                if(is_file("$path/$file")){
                    // Display filename
                    echo $file . "<br>";
                } else if(is_dir("$path/$file")){
                    // Recursively call the function if directories
found
                    outputFiles("$path/$file");
                }
            }
        } else{
            echo "ERROR: No files found in the directory.";
        }
    } else {
        echo "ERROR: The directory does not exist.";
    }
}

// Call the function
outputFiles("mydir");
?>
```

Listing All Files of a Certain Type

While working on directory and file structure, sometimes you might need to find out certain types of files within the directory, for example, listing only `.text` or `.png` files, etc. You can do this easily with the PHP `glob()` function, which matches files based on the pattern.

The PHP code in the following example will search the *documents* directory and list all the files with `.text` extension. It will not search the subdirectories.

Example	Run this code »

```php
<?php
/* Search the directory and loop through
returned array containing the matched files */
foreach(glob("documents/*.txt") as $file){
    echo basename($file) . " (size: " . filesize($file) . " bytes)" . "
<br>";
}
?>
```

The `glob()` function can also be used to find all the files within a directory or its subdirectories. The function defined in the following example will recursively list all files within a directory, just like we've done in previous example with the `scandir()` function.

Example Run this code »

```php
<?php
// Define a function to output files in a directory
function outputFiles($path){
    // Check directory exists or not
    if(file_exists($path) && is_dir($path)){
        // Search the files in this directory
        $files = glob($path ."/*");
        if(count($files) > 0){
            // Loop through retuned array
            foreach($files as $file){
                if(is_file("$file")){
                    // Display only filename
                    echo basename($file) . "<br>";
                } else if(is_dir("$file")){
                    // Recursively call the function if directories
found
                    outputFiles("$file");
                }
            }
        } else{
            echo "ERROR: No such file found in the directory.";
        }
    } else {
        echo "ERROR: The directory does not exist.";
    }
}

// Call the function
```

```
outputFiles("mydir");
?>
```