Manipulate files in PHP **2**
Open a file 3
Open a file with fopen () 4
Read the contents of a file with fgets () 5
Check if a file exists with file_exists () 6
Writing to a file with fwrite () 7
Copy a file with copy () 9
Deleting a file with unlink () 9
Text file and table 10
Browse Folders in PHP **11**
How to Open and Browse Folders in PHP 11

Manipulate files in PHP

Why manipulate files in PHP?

In programming, knowing how to open and manipulate files (text or CSV files) is an essential asset. Sometimes a text file can contain important data for your program.

In this section, we will see the different ways to manipulate files with PHP. We will start by opening a text file.

For this part of the course, we will use a text file called 'movies.txt' (available on the NAS or by email).
This text contains a movie list, to the left of the symbol equals (=) the name of the movie, on the right the name of the director. We will see how to open and manipulate this file.

The 'movies.txt' file must be in the same folder as your PHP script.

Open a file

To open a file, there are several methods.

readfile ()

We will start with the readfile () function.
As the name suggests, it reads the contents of a file for you.

Try this piece of code:

```
<? Php
   $ file_content = readfile ("movies.txt");
   echo $ file_content;
?>
```

With this piece of code, you get a web page with text (the contents of the file) without separation, without lineetc ...

breakFor the syntax, you have to write 'readfile', open parentheses and write the name of the file to read between these parentheses.

You can write the file name directly or use a variable:

```
      $ file_to_read = "movies.txt";
$ file_content = readfile ($ file_to_read);
```

The readfile () function is useful if all you want to do is open a file and read its contents.

file_get_contents ()

Another function that only reads the contents of a file is file_get_contents ().

The syntax is the same:

```
<? Php
   $ file_to_read = "movies.txt";
   echo file_get_contents ($ file_to_read);
?>
```

Open a file with fopen ()

A better way to open files is to use fopen ().
With fopen (), you have more control and options, such as whether the file is read, write, and some additional options.

The syntax is:

```
<? Php
   fopen ($ file_name, $ mode);
?>
```

- We always start with fopen and open the parentheses
- $ file_name is the name of the file to open.
- $ mode is the opening mode of the file.

Try this piece of code now:

```
<? Php
   $ file_handle = fopen ("movies.txt", "r");
   echo $ file_handle;
   fclose ($ file_handle);
?>
```

You will get a result like this:

```
Resource id # 2
```

In fact, unlike the readfile () and file_get_contents () functions, fopen () does not read the contents of a file.
All it does is set a pointer to the file we want to open. It then returns what is called a file handle.

All we do is tell PHP to remember the location of the file.

The 'r' at the end means 'read only' and thus 'open this file read only'.
Now

that we have told PHP to remember the location of the file, how to read its contents?

Read the contents of a file with fgets ()

For that, we will use fgets ().

This function will read a specific number of characters on a single line of text. It is usually used to loop and read each line of text.

When we use fgets (), we must also check if the end of the file has been reached. For this, we use the function feof () - End Of File.

Try this piece of code:

```
<? Php
   $ file_handle = fopen ("movies.txt", "r");

   while (! feof ($ file_handle)) {

      $ line_of_text = fgets ($ file_handle);
      echo $ line_of_text. "<br>";

   }

   fclose ($ file_handle);
?>
```

The result is the list of my movies displayed on several lines.

Let's take the code:

- *$ file_handle = fopen ("movies.txt", "r");*
  PHP is told to open a file and remember its location.

- *while (! feof ($ file_handle))*
  We want to continue looping until the end of the file has been reached

- *$ line_of_text = fgets ($ file_handle);*
  We use the fgets () function to retrieve a line from the file.
  Then we display this line with an echo.

- *fclose ($ file_handle);*
  This function closes the file, opened with fopen ().
  ALWAYS close the file.

Check if a file exists with file_exists ()

Sometimes it's a good idea to check if a file exists before you want to manipulate it.
To do this, use the function file_exists ().

For example :

```
<? php
   if (file_exists ("moviiiiies.txt")) {
      echo "File exists";
   }
   else {
      echo "File does not exist";
   }
?>
```

- This function is useful when we want to know if a file exists or not before handling it.
- This is also useful when creating a new file, if we want to verify that the file does not already exist on the server.

Writing to a file with fwrite ()

If we need to write to a file, there are several ways to do it.
The one we are going to see uses the function fwrite ().

Here is the syntax:

```
<? Php
        fwrite ($ handle, $ string, $ length);
?>
```

- We start by writing fwrite and we open the parentheses
- $ handle is the file pointer (we must use fopen)
- $ string is the data to write in the file
- $ length is optional, used to specify the size file maximum

Try this piece of code:

```
<? php
  $ file_handle = fopen ("test_file.txt", "w");
  $ file_content = "Hello World! How are you?";

  fwrite ($ file_handle, $ file_content);
  fclose ($ file_handle);

  echo "First file created!";
?>
```

- $ file_handle = fopen ("test_file.txt", "w");
First, PHP is asked to open the file and create a file handle (pointer).
If no file with this name is found, it creates it. We made it clear "w" to write.

- fwrite ($ file_handle, $ file_content);
We use fwrite to write our content in our file.

If you change the "w" to "a":

$ file_handle = fopen ("test_file.txt", "a");

Launch the script multiple times.
Open the text file, what do you notice?

> You notice that the content has been added several times at the end of the file.
This is because mode 'a' means 'append' and therefore allows to add content at the end of a file.

If you change the "a" to "r":

$ file_handle = fopen ("test_file.txt", "r");

Restart the script and open the file.

What happened ?

> Nothing happened! This is because the 'r' mode only reads a file.

Copy a file with copy ()

In PHP, we can copy a file easily with the copy () function

Here is the syntax:

```
copy ($ file, $ copied_file);
```

- $ file is the file to copy.
- $ copied_file is the name of the path / file to copy.

Try this piece of code:

```
<? php
copy ('movies.txt', 'movies_copy.txt');
echo "File successfully copied";
?>
```

Deleting a file with unlink ()

To delete a file, in PHP, use the **unlink () function.**

For example:

```
<? Php
        echo "File successfully deleted";

if (! unlink ('movies_copy.txt')) {
    echo "Could not delete file";
}
else {
    echo "File successfully deleted";
}
?>
```

Text file and table

You can use fgets () to save each row of a file in a table. For example:

```
<? Php
$ my_array = array ();
$ i = 0;

$ file_handle = fopen ("movies.txt", "r");

while (! feof ($ file_handle)) {
   $ line_of_text = fgets ($ file_handle);
   $ my_array [$ i] = $ line_of_text;

                $ i ++;
}

fclose ($ file_handle);
?>
```

Browse Folders in PHP

How to Open and Browse Folders in PHP

To open and browse a folder in PHP, you will need to use three (3) functions:

- **opendir ()**
- **readdir ()**
- **closedir ()**

Try this piece of code:

```
<? php
        $ handle = opendir ('.'));

        while ($ entry = readdir ($ handle)) {
                if ($ entry! = "." && $ entry! = "..") {
                        echo $ entry. "<br>";
          }
        }

        closedir ($ handle);
?>
```

Take the code:

- ***$ handle = opendir ('.'));***
PHP is told to open the current folder and remember its location.

- **while ($ entry = readdir ($ handle))**
We want to continue looping as long as there are items in the folder

- **if ($ entry! = "." && $ entry! = "..")**
The item is displayed if the current item is not the current folder or the parent folder.

- **closedir ($ handle);**
This function closes the folder, opened with opendir ().
ALWAYS close the open folder.