PHP
PHP – Files

Primary source: https://www.w3schools.com/php/

Internet Programming 2, Lesson 10

May 2024

PHP Include Files

The `include` (or `require`) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

require will produce a fatal error (E_COMPILE_ERROR) and stop the script
include will only produce a warning (E_WARNING) and the script will continue
So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Syntax

```
include 'filename’;    or
require 'filename';
```

PHP include Example:

Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```

To include the footer file in a page, use the `include` statement:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

PHP File Handling

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

PHP readfile() Function

The `readfile()` function reads a file and writes it to the output buffer.
Assume we have a text file called "acronyms.txt", stored on the server, that looks like this:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

The PHP code to read the file and write it to the output buffer is as follows (the `readfile()` function returns the number of bytes read on success):

```
<?php
        echo readfile("acronyms.txt");
?>
```

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

NOTE:
PHP provides a comprehensive set of built-in functions to work with files, enabling developers to perform various operations such as opening, reading, writing, appending, and closing files. The primary method of working with files in PHP involves the following steps:

1. Opening a file: The fopen() function is used to open a file in a specified mode (e.g., read, write, or append). This function returns a file pointer resource, which is then used in subsequent file operations.
2. Reading from or writing to a file: Depending on the file mode, various functions can be used to read from or write to a file. For reading, functions like fgets(), fread(), and file() are available, while for writing, functions like fwrite() and file_put_contents() are used.
3. Closing a file: After completing the desired file operations, the fclose() function is used to close the file and release the file pointer resource.

By following these steps, developers can efficiently work with text files in PHP, enabling them to build powerful and flexible applications that can store and process data across various contexts.

It may be useful to visualize the “file pointer” to understand how the system works. If we open the file acronyms.txt in Notepad, we will see the text as in the picture below. A pointer (red) is placed at the start of the file when it is opened in read or write mode.

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

Some points to note:

1. After we read the first line of text, the pointer is advanced to the next line. If we were reading a character at a time, the pointer will advance to the next character.
2. The end of each line is marked in the file by an end-of-line character which is non-printable and does not show in the text editor. We will use this fact to control scanning characters in the file one character at a time.
3. Likewise, the end of the file is marked by a special non-printable character – traditionally CNTL-Z

The end of line character in a text file is used to represent the end of a line of text and the beginning of a new one. It varies depending on the operating system. Windows uses a combination of two characters, carriage return (CR) and line feed (LF), represented as \\r\\n. Unix-based systems, such as Linux and macOS, use only LF, represented as \\n. The old MacOS (pre-OS X) used only CR, represented as \\r1.

The end-of-file (EOF) marker is a special character or characters used to denote the end of a text file. In operating systems such as CP/M and MS-DOS, where the operating system does not keep track of the file size in bytes, the end of a text file is denoted by placing one or more special characters, known as an EOF marker, as padding after the last line in a text file1.

However, in most modern operating systems, EOF is not a character but rather a condition that applies to a file stream when the end of the stream is reached.

These aspects are handled by PHP through two functions – one to detect the eol and the other to detect the eof. Discussed later in this slide deck.

PHP File Open/Read/Close

PHP Open File - fopen()
A better method to open files is with the `fopen()` function. This function gives you more options than the `readfile()` function.
We will use the text file, "webdictionary.txt", during the lessons:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

The first parameter of `fopen()` contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

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

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP Read File - fread()

The `fread()` function reads from an open file.
The first parameter of `fread()` contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

PHP Close File - fclose()

The `fclose()` function is used to close an open file.

It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

The `fclose()` requires the name of the file (or a variable that holds the filename) we want to close:

```
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

PHP Read Single Line - fgets()

The fgets() function is used to read a single line from a file.
The example below outputs the first line of the "webdictionary.txt" file:

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

Note: After a call to the fgets() function, the file pointer has moved to the next line.

PHP Check End-Of-File - feof()

The feof() function checks if the "end-of-file" (EOF) has been reached.
The feof() function is useful for looping through data of unknown length.
The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

PHP Read Single Character - fgetc()

The `fgetc()` function is used to read a single character from a file.
The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

Note: After a call to the `fgetc()` function, the file pointer moves to the next character.

PHP File Create/Write

PHP Create File - fopen()

The fopen() function is *also* used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).
The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

```
$myfile = fopen("testfile.txt", "w")
```

PHP File Permissions

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File - fwrite()

The `fwrite()` function is used to write to a file.
The first parameter of `fwrite()` contains the name of the file to write to and the second parameter is the string to be written.
The example below writes a couple of names into a new file called "newfile.txt":

```php
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the `fclose()` function.
If we open the "newfile.txt" file it would look like this:

```
John Doe
Jane Doe
```

PHP Overwriting

Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.
In the example below we open our existing file "newfile.txt", and write some new data into it:

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

```
Mickey Mouse
Minnie Mouse
```

PHP Append Text

You can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file, while the "w" mode overrides (and erases) the old content of the file.
In the example below we open our existing file "newfile.txt", and append some text to it:

```
<?php
$myfile = fopen("newfile.txt", "a") or die("Unable to open file!");
$txt = "Donald Duck\n";
fwrite($myfile, $txt);
$txt = "Goofy Goof\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

If we now open the "newfile.txt" file, we will see that Donald Duck and Goofy Goof is appended to the end of the file:

```
Mickey Mouse
Minnie Mouse
Donald Duck
Goofy Goof
```

Exercise:

1. Write a PHP script to generate and store lotto numbers for the Month's draws in a text file for the month of May 2022. Name the file lotto1.php
   The format of the file must be as follows:
   Draw Number, Day, Date, Lott-Nums, bonus ball
   For example, the first line in the file should read:
   1, Wed, 4 May 2022, 1,2,3,4,5,6, 20
   In the above example, the numbers were 1, 2, 3, 4, 5, 6, and the bonus ball was 20
   Call the output file lotto1.txt

2. Write a PHP script to display the contents of the file created in 1, neatly on a web page. Make use of CSS to make it an acceptable user experience.

3. Using Notepad (or other text editor) create two text files as follows:
   File1: WedLottoDraws.txt consisting of the results (which you can decide but remember to choose unique numbers within each draw) according to the format above. This file will have only the results for the Wednesday draw for the month of May 2022.
   File2: SatLottoDraws.txt consisting of the results (in same format as above) for all Saturdays in May 2022.
   Note that an easy way to create these two files will be to edit the resulting file from question 1 above(provided that you were successful in solving ques 1 above, off course!), in Notepad (or some other text editor) to form these two files for this question.
   Now, write php to merge these two files into a third file called AllMay22Draws.txt which will contain the content of the two files (file1 and file2), maintaining the date order of the original files.

4. Yes, you may have guessed it… Now write a PHP script to split-up ("unmerge" if you like!) the file called AllMay22Drawx.txt produced in question 3 above, into the two files described as File1 and File2 above. Name these output files as wedDraws1.txt and SatDraws-1.txt respectively.

Yes, there is something extraordinarily cyclic about these questions – not intended. Good Luck.

Please explore and and be able to apply the following concepts (see w3schools-site for the material) on the following sections:

- PHP File Upload
- PHP Cookies
- PHP Sessions
- PHP Filters

Class Example 2:
Examine the file called StudentMarks.txt
It contains the following records:
12345, 60, 60, 55
12346, 20, 30, 70
12347, 45, 50, 60

The data represents student test marks in the following format:
StudNum, TM1, TM2, TM3

Our tasks are as follows:
Version 1: read this file and determine and display each student's course mark in a neat table on the web page. The course mark is determined by the formula: 25% of TM1, 35% of TM2 and 40% of TM3.

Version2: as in version 1 but store the student's CM in a new file called StudentCM.txt in the following format:
StudNum, TM1, TM2, TM3, CM

Version 3: store the CM in the same file [studMarks.txt]
Problem solve this first.