|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# PHP 5 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 Examples

### Example 1

**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:**

### Example

**<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 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.

### Example

**Assume we have a standard menu file called "menu.php":**

**<?php  
echo '<a href="/default.asp">Home</a> -  
<a href="/html/default.asp">HTML Tutorial</a> -  
<a href="/css/default.asp">CSS Tutorial</a> -  
<a href="/js/default.asp">JavaScript Tutorial</a> -  
<a href="default.asp">PHP Tutorial</a>';  
?>**

**All pages in the Web site should use this menu file. Here is how it can be done (we are using a <div> element so that the menu easily can be styled with CSS later):**

### Example

**<!DOCTYPE html>  
<html>  
<body>  
  
<div class="menu">  
<?php include 'menu.php';?>  
</div>  
  
<h1>Welcome to my home page!</h1>  
<p>Some text.</p>  
<p>Some more text.</p>  
  
</body>  
</html>**

### Example 3

**Assume we have a file called "vars.php", with some variables defined:**

**<?php  
$color='red';  
$car='BMW';  
?>**

**Then, if we include the "vars.php" file, the variables can be used in the calling file:**

### Example

**</body>  
</html>**

**<!DOCTYPE html>  
<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php include 'vars.php';  
echo "I have a $color $car.";  
?>  
  
</body>  
</html>**

## PHP include vs. require

**The require statement is also used to include a file into the PHP code.**

**However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute:**

### Example

**<!DOCTYPE html>  
<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php include 'noFileExists.php';  
echo "I have a $color $car.";  
?>  
</body>  
</html>**

**If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error:**

### Example

**<!DOCTYPE html>  
<html>  
<body>  
<h1>Welcome to my home page!</h1>  
<?php require 'noFileExists.php';  
echo "I have a $color $car.";  
?>  
</body>  
</html>**

**Use require when the file is required by the application.**

**Use include when the file is not required and application should continue when file is not found.**

# 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.**

**Be careful when manipulating files!**

**When you are manipulating files you must be very careful.**

**You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.**

## PHP readfile() Function

**The readfile() function reads a file and writes it to the output buffer.**

**Assume we have a text file called "webdictionary.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):**

### Example

**<!DOCTYPE html>  
<html>  
<body>  
<?php  
echo readfile("webdictionary.txt");  
?>  
</body>  
</html>**

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

**The next chapters will teach you more about file handling.**

# PHP 5 File Open/Read/Close

**In this chapter we will teach you how to open, read, and close a file on the server.**

## 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:**

### Example

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

**Tip: The fread() and the fclose() functions will be explained below.**

**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:**

### Example

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

**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:**

### Example

**<?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);  
?>  
</body>  
</html>**

## 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:**

### Example

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

**<!DOCTYPE html>  
<html>  
<body>  
  
<?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);  
?>  
  
</body>  
</html>**

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

# PHP 5 File Create/Write

**In this chapter we will teach you how to create and write to a file on the server.**

## 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:**

### Example

**$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":**

### Example

**<?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:**

### Example

**<?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**

## Complete PHP Filesystem Reference

**For a complete reference of filesystem functions, go to our complete**[**PHP Filesystem Reference**](https://www.w3schools.com/php/php_ref_filesystem.asp)**.**

# PHP 5 File Upload

**With PHP, it is easy to upload files to the server.**

**However, with ease comes danger, so always be careful when allowing file uploads!**

## Configure The "php.ini" File

**First, ensure that PHP is configured to allow file uploads.**

**In your "php.ini" file, search for the file\_uploads directive, and set it to On:**

**file\_uploads = On**

## Create The HTML Form

**Next, create an HTML form that allow users to choose the image file they want to upload:**

**<!DOCTYPE html>  
<html>  
<body>  
  
<form action="upload.php" method="post" enctype="multipart/form-data">  
    Select image to upload:  
    <input type="file" name="fileToUpload" id="fileToUpload">  
    <input type="submit" value="Upload Image" name="submit">  
</form>  
  
</body>  
</html>**

**Some rules to follow for the HTML form above:**

* **Make sure that the form uses method="post"**
* **The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form**

**Without the requirements above, the file upload will not work.**

**Other things to notice:**

* **The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control**

**The form above sends data to a file called "upload.php", which we will create next.**

## Create The Upload File PHP Script

**The "upload.php" file contains the code for uploading a file:**

**<?php  
$target\_dir = "uploads/";  
$target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);  
$uploadOk = 1;  
$imageFileType = strtolower(pathinfo($target\_file,PATHINFO\_EXTENSION));  
// Check if image file is a actual image or fake image  
if(isset($\_POST["submit"])) {  
    $check = getimagesize($\_FILES["fileToUpload"]["tmp\_name"]);  
    if($check !== false) {  
        echo "File is an image - " . $check["mime"] . ".";  
        $uploadOk = 1;  
    } else {  
        echo "File is not an image.";  
        $uploadOk = 0;  
    }  
}  
?>**

**PHP script explained:**

* **$target\_dir = "uploads/" - specifies the directory where the file is going to be placed**
* **$target\_file specifies the path of the file to be uploaded**
* **$uploadOk=1 is not used yet (will be used later)**
* **$imageFileType holds the file extension of the file (in lower case)**
* **Next, check if the image file is an actual image or a fake image**

**Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.**

## Check if File Already Exists

**Now we can add some restrictions.**

**First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:**

**// Check if file already exists  
if (file\_exists($target\_file)) {  
    echo "Sorry, file already exists.";  
    $uploadOk = 0;  
}**

## Limit File Size

**The file input field in our HTML form above is named "fileToUpload".**

**Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:**

**// Check file size  
if ($\_FILES["fileToUpload"]["size"] > 500000) {  
    echo "Sorry, your file is too large.";  
    $uploadOk = 0;  
}**

## Limit File Type

**The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:**

**// Allow certain file formats  
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"  
&& $imageFileType != "gif" ) {  
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";  
    $uploadOk = 0;  
}**

## Complete Upload File PHP Script

**The complete "upload.php" file now looks like this:**

**<?php  
$target\_dir = "uploads/";  
$target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);  
$uploadOk = 1;  
$imageFileType = strtolower(pathinfo($target\_file,PATHINFO\_EXTENSION));  
// Check if image file is a actual image or fake image  
if(isset($\_POST["submit"])) {  
    $check = getimagesize($\_FILES["fileToUpload"]["tmp\_name"]);  
    if($check !== false) {  
        echo "File is an image - " . $check["mime"] . ".";  
        $uploadOk = 1;  
    } else {  
        echo "File is not an image.";  
        $uploadOk = 0;  
    }  
}  
// Check if file already exists  
if (file\_exists($target\_file)) {  
    echo "Sorry, file already exists.";  
    $uploadOk = 0;  
}  
// Check file size  
if ($\_FILES["fileToUpload"]["size"] > 500000) {  
    echo "Sorry, your file is too large.";  
    $uploadOk = 0;  
}  
// Allow certain file formats  
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"  
&& $imageFileType != "gif" ) {  
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";  
    $uploadOk = 0;  
}  
// Check if $uploadOk is set to 0 by an error  
if ($uploadOk == 0) {  
    echo "Sorry, your file was not uploaded.";  
// if everything is ok, try to upload file  
} else {  
    if (move\_uploaded\_file($\_FILES["fileToUpload"]["tmp\_name"], $target\_file)) {  
        echo "The file ". basename( $\_FILES["fileToUpload"]["name"]). " has been uploaded.";  
    } else {  
        echo "Sorry, there was an error uploading your file.";  
    }  
}  
?>**

## Complete PHP Filesystem Reference

**For a complete reference of filesystem functions, go to our complete**[**PHP Filesystem Reference**](https://www.w3schools.com/php/php_ref_filesystem.asp)**.**