

# 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 `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

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

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

## 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

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

**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	<b>Open a file for read only.</b> File pointer starts at the beginning of the file
w	<b>Open a file for write only.</b> 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	<b>Open a file for write only.</b> 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	<b>Creates a new file for write only.</b> Returns FALSE and an error if file already exists
r+	<b>Open a file for read/write.</b> File pointer starts at the beginning of the file

w+	<b>Open a file for read/write.</b> 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+	<b>Open a file for read/write.</b> 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+	<b>Creates a new file for read/write.</b> 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 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

## Database Handling

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

### Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

#### Example (MySQLi Object-Oriented)

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```
$password = "password";
```

```
// Create connection
```

```
$conn = new mysqli($servername, $username, $password);
```

```
// Check connection
```

```
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

### Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

### Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

### Example (MySQLi Object-Oriented)

```
$conn->close();
```

### Example (MySQLi Procedural)

```
mysqli_close($conn);
```

### Create a MySQL Database Using MySQLi

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

## Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

**Note:** When you create a new database, you must only specify the first three arguments to the mysqli object (servername, username and password).

**Tip:** If you have to use a specific port, add an empty string for the database-name argument, like this: `new mysqli("localhost", "username", "password", "", port)`

---

---



## Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

## Create a MySQL Table Using MySQLi

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg\_date":

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
```

```
reg_date TIMESTAMP
)
```

### Notes on the table above:

The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our [Data Types reference](#).

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO\_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

### Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

```
// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

---

---

### Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
```

```

}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

## Insert Data Into MySQL Using MySQLi

The INSERT INTO statement is used to add new records to a MySQL table:

```

INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)

```

### Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection

```

```
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

---

---

### Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');"
```

```
if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

```
mysqli_close($conn);
```

```
?>
```

## Insert Multiple Records Into MySQL Using MySQLi

Multiple SQL statements must be executed with the `mysqli_multi_query()` function.

The following examples add three new records to the "MyGuests" table:

### Example (MySQLi Object-oriented)

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```
$password = "password";
```

```
$dbname = "myDB";
```

```
// Create connection
```

```
$conn = new mysqli($servername, $username, $password, $dbname);
```

```
// Check connection
```

```
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
```

```
}
```

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
```

```
VALUES ('John', 'Doe', 'john@example.com');";
```

```
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Note that each SQL statement must be separated by a semicolon.

---

---

### Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
```

```

VALUES ('John', 'Doe', 'john@example.com');"
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');"
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com');"

if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

## Delete Data From a MySQL Table Using MySQLi

The DELETE statement is used to delete records from a table:

```

DELETE FROM table_name
WHERE some_column = some_value

```

id	firstname	lastname	Email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

The following examples delete the record with id=3 in the "MyGuests" table:

### Example (MySQLi Object-oriented)

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

```



```
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

---

---

### Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

```

}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

## Limit Data Selections From a MySQL Database

MySQL provides a LIMIT clause that is used to specify the number of records to return.

The LIMIT clause makes it easy to code multi page results or pagination with SQL, and is very useful on large tables. Returning a large number of records can impact on performance.

Assume we wish to select all records from 1 - 30 (inclusive) from a table called "Orders". The SQL query would then look like this:

```
$sql = "SELECT * FROM Orders LIMIT 30";
```

When the SQL query above is run, it will return the first 30 records.

What if we want to select records 16 - 25 (inclusive)?

MySQL also provides a way to handle this: by using OFFSET.

The SQL query below says "return only 10 records, start on record 16 (OFFSET 15)":

```
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";
```

You could also use a shorter syntax to achieve the same result:

```
$sql = "SELECT * FROM Orders LIMIT 15, 10";
```

Notice that the numbers are reversed when you use a comma.