

Files and CSV

File and Csv of Derafu Support

Anonymous · 08/10/2026

Files and CSV

File

File operations. Every failure is a `RuntimeException` that says which file and what failed.

Method	What it does	Needs
<code>File::write(\$targetFile, \$content, \$permissions = null)</code>	Writes a file atomically.	
<code>File::rmdir(\$dir)</code>	Removes a directory and everything in it.	<code>symfony/filesystem</code>
<code>File::mimetype(\$file)</code>	The MIME type of a file, or <code>false</code> .	<code>symfony/mime</code>
<code>File::zip(\$source, \$destination)</code>	Makes a ZIP of a file or a directory.	<code>maennchen/zipstream-php</code>
<code>File::compress(\$source, \$download = false, \$delete = false)</code>	<code>zip()</code> next to the source, optionally sending it or removing the source.	<code>maennchen/zipstream-php</code>
<code>File::unzip(\$zipFile, \$destination, \$overwrite = false)</code>	Extracts a ZIP.	<code>ext-zip</code>
<code>File::send(\$file, \$delete = false, \$sendHeaders = true)</code>	Sends a file as a download.	<code>symfony/mime</code>

Writing

```
use Derafu\Support\File;

File::write('/var/app/data/config.json', $json);
File::write('/var/app/data/secret.key', $key, 0600);
```

The content goes to a temporary file in the same directory, and then it **replaces** the target with a rename, so another process sees the old file or the new one, never a half-written file, and if the process is interrupted the old one is still there. The directory is created if it does not exist. The

permissions are `0666` minus the `umask` unless they are given.

MIME types, directories and ZIP

```
File::mimetype('document.pdf');           // "application/pdf" (false if the file does
not exist or is not known).

File::rmdir('/tmp/work');                  // Does not follow symbolic links: it removes
the link, not what it points to.

File::zip('/path/to/dir', '/tmp/dir.zip'); // The files keep their path inside the
directory.
                                           // The same if the directory is reached
through a symbolic link or with a slash at the end.
File::zip('/path/to/file.txt', '/tmp/file.zip');
File::unzip('/tmp/dir.zip', '/path/to/out');
```

`zip()` writes the archive to the file and sends no HTTP headers: it can be called from code that is going to answer a request.

`unzip()` does not overwrite: if **any** file of the archive already exists in the destination it throws before extracting anything, unless `$overwrite` is true.

`compress()` makes `<source>.zip` next to the source (`/path/to/dir` is `/path/to/dir.zip`). With `$download` it sends the ZIP and removes it after sending; with `$delete` it removes the source after compressing.

```
File::compress('/path/to/dir');           // Makes /path/to/dir.zip.
File::compress('/path/to/dir', download: true); // Sends it and removes the ZIP.
```

Sending

```
File::send('/path/to/report.pdf');           // As a download, with its MIME
type.
File::send('/tmp/export.zip', delete: true); // And removes it after sending.
```

`send()` writes the headers (Content-Type from the MIME type, Content-Disposition: attachment with the name of the file, Content-Length and no cache) and then the content, and it does **not** end the script. If the headers were already sent it throws; with `$sendHeaders = false` it only writes the content (when the framework or the test that calls it takes care of the headers). A file that does not exist or can not be read is an error.

Csv

Simplified reading and writing of CSV with [League/csv](#). The defaults are the ones that spreadsheets in

Spanish use: separator `;`, enclosure `"` and escape `\`.

Method	What it does
<code>Csv::read(\$file, \$separator = ';', \$enclosure = '"', \$escape = '\\', \$encoding = 'UTF-8')</code>	The rows of a file.
<code>Csv::load(\$content, ...)</code>	The rows of a text (same parameters).
<code>Csv::write(\$data, \$file, \$separator, \$enclosure, \$escape)</code>	Writes records to a file.
<code>Csv::generate(\$data, \$separator, \$enclosure, \$escape, \$encoding = 'UTF-8')</code>	The CSV text of some records.
<code>Csv::send(\$data, \$filename, \$separator, \$enclosure, \$escape, \$sendHeaders = true)</code>	Sends the CSV as a download.
<code>Csv::validate(\$file, \$separator, \$enclosure, \$escape, \$sampleSize = 5)</code>	Whether a file looks like a CSV.

Reading

`read()` and `load()` give **a list of rows**, each one a list of texts. The first row is the header like any other, so take it off if the file has one:

```
use Derafu\Support\Csv;

$rows = Csv::read('file.csv');           // [['a', 'b'], ['1', '2'], ['3', '4']]
$rows = Csv::load("x,y\n1,2", ',');      // [['x', 'y'], ['1', '2']]

$header = array_shift($rows);
$records = array_map(fn (array $row) => array_combine($header, $row), $rows);
```

`$encoding` is the encoding of what is read (`'ISO-8859-1'` for a file from an old system): the rows come out in UTF-8.

Writing

`write()`, `generate()` and `send()` take **records**: a list of associative arrays. The first line is the header, with the keys of the first record.

```
$data = [['a' => 1, 'b' => 2], ['a' => 3, 'b' => 4]];

Csv::generate($data);           // "a;b\n1;2\n3;4\n"
Csv::write($data, 'export.csv'); // The same, in a file.
Csv::send($data, 'export.csv'); // As a download.

Csv::generate($data, encoding: 'ISO-8859-1'); // For the system that wants it.
```

A value with the separator inside is put in the enclosure (a;b is written as "a;b"). `generate()` can write in another encoding; `send()` always sends UTF-8 (Content-Type: text/csv; charset=UTF-8, as an attachment, with no cache) and, like `File::send()`, it throws if the headers were already sent and does not end the script; `$sendHeaders = false` only writes the content.

Errors

The failures are `RuntimeExceptions` (a `LogicException` when `write()` can not write the file), with the file and the reason in the message.

Validating

`validate()` says whether the file exists and can be read, whether its first `$sampleSize` rows can be read, and whether they all have the same number of columns. It does not say that the content is right, only that it looks like a CSV with that separator.

Last updated on 08/10/2026