ZIP Archives

Create ZIP archives from contents, streams or files of any size, and extract them, with Zip

Anonymous · 05/11/2026

ZIP Archives

```
use Derafu\File\Zip;
```

Method	What it does
`Zip::fromContents($files)`	A ZIP archive, as a string, from small contents that are in memory.
`Zip::toStream($entries, $maxMemory)`	A ZIP archive, as a stream, from strings, streams or files of any size.
`Zip::fromPath($source, $destination)`	A ZIP archive file from a file or a directory.
`Zip::extract($zipFile, $destination, $overwrite = false)`	Extracts an archive.

No method sends HTTP headers: they can be called from code that is going to answer a request.

From contents

```
$archive = Zip::fromContents([
    'RecepcionDTE.xml' => $reception,
    'reports/summary.txt' => $summary,
]);

return new Response(200, ['Content-Type' => 'application/zip'], $archive);
```

The key of each entry is its name inside the archive, which can have directories; the value is its content. They go in the order of the array. The archive is made in memory: **no temporary file** is created, and the contents, the archive and the string that is returned are all in memory, so it is for small data. For anything that can be large use `toStream()`. An array without entries gives a valid empty archive.

A name that the archive would change or that would take the file out of its directory is refused, as is an empty name: a null character, a backslash, a leading slash or a `..` part. A name such as `..hidden` or `a..b/c.txt` is valid. A name or a content that is not valid is a

`TranslatableInvalidArgumentException` (a content must be a string). An archive that can not be created, for instance with a name of more than 65535 bytes, is a `FileException`.

To a stream

```
$archive = Zip::toStream([
    'summary.xml' => $xml,                              // A string: the content.
    'reports/big.csv' => new SplFileInfo($path),    // A file: read in pieces from its
path.
    'upload.bin' => $resource,                          // A stream resource, from where
it is.
    'body.json' => $psrStream,                          // A PSR-7 stream, from where it
is.
]);
```

It gives the archive as a readable and seekable stream resource, ready to be read from the start. The entries go in the order of the array, and the names are checked as in `fromContents()`. A source that is not a string, a stream (resource or PSR-7), or an `SplFileInfo` is a `TranslatableInvalidArgumentException` that names the type received, as is a closed resource.

The size of the entries is not a limit. A file or a stream is read in pieces, so the memory that making the archive takes **does not grow with the size of the entries**: the writer of the archive works in blocks of 16 MiB, so compressing a large entry takes some tens of megabytes at its peak, the same for 40 MiB as for 160 MiB. A PSR-7 stream is copied in pieces to a stream of PHP first, because that is what the writer reads.

The archive is written to a `php://temp` stream: it stays in memory up to `$maxMemory` bytes (1 MiB by default, enough for the archive of a few documents) and from then on it goes to a temporary file. That file belongs to PHP and has no path: **it is deleted when the stream is closed or freed**, so there is nothing for the caller to delete, not even when the archive fails halfway. A `$maxMemory` below zero is a `TranslatableInvalidArgumentException`.

An archive that can not be created, as one with a file that does not exist or can not be read, or a name of more than 65535 bytes, is a `FileException` and nothing is left behind.

From a path

```
Zip::fromPath('/path/to/dir', '/tmp/dir.zip');        // The files keep their path
inside the directory.
Zip::fromPath('/path/to/file.txt', '/tmp/file.zip');  // The file keeps its name.
```

A directory goes with everything inside, and each entry is named by its path inside the directory **as it was given**, so a directory reached through a symbolic link or with a slash at the end gives the same names.

A source that does not exist is a `FileNotFoundException` and nothing is created. A destination that can not be opened, or a file that can not be read, is a `FileException`, and the archive is not left half made.

Extract

```
Zip::extract('/tmp/dir.zip', '/path/to/out');
Zip::extract('/tmp/dir.zip', '/path/to/out', overwrite: true);
```

The destination is created if it does not exist. It does not overwrite: if **any** file of the archive already exists in the destination it throws before extracting anything, unless `$overwrite` is `true`. A directory that already exists is not a file that exists, so an archive can be extracted into a directory that has the folders of the archive.

It needs `ext-zip`. An archive that does not exist is a `FileNotFoundException`; an extension that is not available, a file that is not an archive, a file that exists, or an extraction that fails is a `FileException`.

Last updated on 05/11/2026