Working with PHP

Working With PHP · https://www.exeoutput.com/help/work-with-php

ExeOutput for PHP™ integrates the entire PHP runtime into your application, allowing for complete customization.

About PHP Implementation in ExeOutput for PHP

Since version 2, ExeOutput for PHP has utilized PHP CGI as an external process. This approach offers several benefits:

- **Enhanced Stability and Responsiveness:** Isolating PHP from the UI process makes your application more stable and responsive.
- **Automatic Recovery:** If the PHP process crashes, it automatically restarts when the webpage is refreshed, ensuring your application continues to run smoothly.
- **Flexible PHP Version Support:** ExeOutput for PHP supports a wide range of PHP versions, from PHP 5.6 to the latest 8.x releases. You can select your preferred PHP version.
- **Seamless Chromium Integration:** Enjoy full integration between PHP and the Chromium rendering engine, with support for AJAX, cookies, file upload/download dialogs, custom HTTP request and response headers, and Developer Tools.

Note

For console applications, the PHP CLI is used instead of PHP CGI.

Including PHP Extensions in Your Application

If your PHP application requires **extensions**, you must first configure them on the PHP Extensions page.

Tip

ExeOutput for PHP can embed PHP extensions directly into the EXE file, eliminating the need for manual deployment.

Where Are PHP Files Located at Runtime?

ExeOutput for PHP uses virtualization: all PHP files are stored in memory (and can be optionally encoded to prevent memory dumps), while the PHP runtime is led to believe they are located on the hard disk.

Essentially, your PHP application uses a `Data` subfolder to store its PHP files. By default, this "Data" virtual subfolder is in the same directory as the EXE file. For example, if your EXE is at `E:\my folder\myprogram.exe`, the path to the `Data` subfolder will be `E:\my folder\data`. You can even specify a custom virtual folder that does not physically exist on the disk.

You can also place physical files into this `Data` subfolder. For instance, if you choose to keep some files outside your EXE (as External Files), your PHP application can still access them.

If the EXE is in a writable directory (such as **My Documents** or the user's local **AppData** folder), you can write files to the `Data` subfolder.

To retrieve the path to the `Data` subfolder, use `$_SERVER['DOCUMENT_ROOT']`.

For example, you can use the following code to read a file in the `Data` subfolder:

```
<?php
$cont = file_get_contents($_SERVER['DOCUMENT_ROOT'] . '\\include1.txt', FILE_USE_INCLUDE_PATH);
print($cont);
?>
```

Custom PHP.INI

You can adjust PHP parameters directly with the built-in PHP.INI editor.

Warning

Do not place a custom `php.ini` file in the root folder; use the ExeOutput for PHP editor instead.

Advice for Getting Started with PHP Applications in ExeOutput for PHP

How Compiled PHP Applications Work