# Using PHP Frameworks

Getting Started · https://www.exeoutput.com/help/phpframeworks

ExeOutput for PHP is designed to work seamlessly with PHP frameworks. Since these frameworks typically involve numerous PHP `include` statements, ExeOutput for PHP intercepts all file requests from the PHP runtime and makes the files available (see Accessing Files in PHP).

However, this interception process can slow down your application. To address this, ExeOutput for PHP offers the External Files feature, which allows you to keep the framework's PHP files outside the application (usually in a `vendor` subdirectory). This ensures your application remains responsive, even if it requires tens of thousands of files.

If your preferred PHP framework does not function correctly, please refer to the Advice for Getting Started with PHP Applications section. For example, if PHP displays "missing file" errors or warnings like "**php failed to open stream**", you can try marking files in the File Manager (select your files, click File Properties, and enable the **Unpack the file(s) to virtual memory at startup** option) or, preferably, keep them external.

## Redirecting Directory Requests to the Router Script

Some PHP frameworks support pretty URLs, similar to Apache's `mod_rewrite` in certain cases.

Note

Ensure that you enable the option to redirect all non-file requests to your framework's `index.php` router script.

## Using Custom Redirection Rules

If you prefer to define your own redirection rules, you can use the Redirection Rules feature.

## Setting the Initial Page / Base URL

If your `index.php` homepage should not be the first URL displayed, you can specify a custom URL for the application bootstrap. Refer to the Application Startup URL settings.

Base URLs should start with `https://heserver/` for the CEF engine, or `https://heserver.example/` for the WebView2 engine (see Accessing Files in Compiled PHP Applications).

## Compatible PHP Frameworks

The following PHP frameworks have been successfully tested with ExeOutput for PHP:

### Laravel

Tip

Check out our Laravel sample with step-by-step instructions.

To use Laravel, you must configure the `storage` and `bootstrap/cache` folders to remain outside the EXE so that files can be written to them. We also recommend keeping the `laravel` (and `vendor`, if applicable) subdirectories external.

In ExeOutput for PHP, navigate to the File Manager, select each of the folders mentioned above, click Properties, and enable the option: **Keep the selected file(s) external and copy them to the "Data" subfolder**.

For more details, refer to the External Files topic.

### CodeIgniter

Tip

Check out our CodeIgniter sample with step-by-step instructions.

To use CodeIgniter, you need to configure the `application/cache` and `application/logs` folders to remain outside the EXE so that files can be written to them.

In ExeOutput for PHP, navigate to the File Manager, select each of the folders, click Properties, and enable the option: **Keep the selected file(s) external and copy them to the "Data" subfolder**.

## CakePHP

To use CakePHP, you need to configure the `logs` and `tmp` folders to remain outside the EXE so that files can be written to them.

In ExeOutput for PHP, navigate to the File Manager, select each of the folders, click Properties, and enable the option: **Keep the selected file(s) external and copy them to the "Data" subfolder**.

Finally, edit `www/app/Config/core.php` and add the following line:

```
Configure::write('App.baseUrl', env('SCRIPT_NAME'));
```

## Fat-Free Framework (F3)

No specific configuration is required.

## Other Frameworks

This list is not exhaustive. The absence of a framework does not mean that a PHP application based on it will not work when compiled with ExeOutput for PHP.

Accessing Files in Compiled PHP Applications

How Compiled PHP Applications Work