

Run PHP frameworks in Playground



- [What changes when you skip WordPress](#)
 - [Try a Symfony app](#)
 - [with WP\ HTML\ Processor without installing or * booting WordPress. */ function get\ first\ h1\ te](#)
-

WordPress Playground is also a browser-based PHP runtime. WordPress is the most common app it boots, but a Blueprint can skip the WordPress download, write any PHP files into the virtual filesystem, and run a framework such as Symfony.

This guide shows the shape of that setup. Use it when you want a shareable demo, a docs example, or a quick compatibility check for a PHP app that does not need a server, database, Node.js, Sass, or a local Composer install.

What changes when you skip WordPress

Set `preferredVersions.wp` to `false` in a Blueprint, or `wp="none"` on a `<php-snippet>`. Playground still downloads PHP, mounts a writable filesystem, runs Blueprint steps, and supports networking when `features.networking` is `true`. It just does not download or boot WordPress.

That makes Playground useful for generic PHP examples:

- PHP libraries that need a real filesystem.
- Framework demos that can run behind `public/index.php`.
- Documentation snippets that should execute in the browser.
- Reproducible bug reports for PHP code that is not WordPress-specific.

Try a Symfony app

The example below uses a Blueprint to download and unzip a bundled Symfony app into `/app`. Then a `<php-snippet>` boots the Symfony kernel and renders the dashboard route. The app's Composer dependencies include the WordPress HTML API, so the snippet can read the `<h1>` with `WP_HTML_Processor` without installing or booting WordPress.

```
{ "features": { "networking": true }, "steps": [ { "step": "unzip", "zipFile": { "resource": "url", "url": "https://wordpress.github.io/blueprints/blueprints/symfony-package-radar/symfony-package-radar.zip?v=wp-php-toolkit-html-2026-06-09" }, "extractToPath": "/app" } ] }
```

```
<?php require '/app/symfony-package-radar/vendor/autoload.php'; use App\Kernel; use Symfony\Component\HttpFoundation\Request; $kernel = new Kernel( 'prod', false ); $request = Request::create( '/' ); $response = $kernel->handle( $request ); $page_title = get_first_h1_text( $response->getContent() ); echo 'HTTP ' . $response->getStatusCode() . PHP_EOL; echo 'Symfony page: ' . $page_title . PHP_EOL; echo 'WordPress installed: '; echo file_exists( '/wordpress/wp-load.php' ) ? 'yes' : 'no'; $kernel->terminate( $request, $response ); \*\* The app's Composer dependencies include the WordPress HTML API, so the \* snippet can read the
```

with `WP_HTML_Processor`

without installing or *

booting WordPress. */

```
function get_first_h1_text(
```

```
string $html ): string {
```

```
$processor =
```

```
WP_HTML_Processor::creat
```

```
e\_fragment( $html ); if ( !  
$processor->next\_tag( 'H1'  
) ) { return 'unknown'; }  
$text = ''; while (   
$processor->next\_token() )  
{ if ( 'H1' === $processor-  
>get\_tag() && $processor-  
>is\_tag\_closer() ) { break;  
} if ( '#text' ===  
$processor-  
>get\_token\_type() ) { $text  
.= $processor-  
>get\_modifiable\_text(); } }
```

```
return trim( $text ); } HTTP
```

200 Symfony page: Symfony
Playground WordPress
installed: no Here is the
complete embed: ```` The
same app is also available
as a full Playground page:
[Open the Symfony Package
Radar
demo](<https://playground.wordpress.net/?blueprint-url=https%3A%2F%2Fwordpress.github.io%2Fblueprints>)

%2Fblueprints%2Fsymfony-
package-
radar%2Fblueprint.json) ##
Package the app as a ZIP For
framework demos, prefer a
ZIP that already contains
`vendor/`. That keeps the
Playground startup path
short and avoids asking
every visitor to wait for
Composer, Git, and package
registry downloads. The
Symfony demo uses that

path to bundle both Symfony and a Composer-installed copy of the WordPress HTML API; it still does not include a WordPress install. For snippets or CLI runs, a small Blueprint can install the app into `/app` with one step:

```
``` { "$schema":  
"https://playground.wordpress.net/blueprint-
schema.json",
"landingPage": "/symfony-
```

```
package-
radar/public/index.php",
"preferredVersions": {
 "php": "8.4", "wp": false },
"features": { "networking":
true }, "steps": [{ "step":
 "unzip", "zipFile": {
 "resource": "bundled",
 "path": "./symfony-package-
radar.zip" },
 "extractToPath": "/app" }] }
```` Use `bundled` resources  
when the ZIP ships next to
```

`blueprint.json`, or use a
`url` resource when the ZIP
is hosted separately. See
[Blueprint
bundles](<https://developer.wordpress.org/playground/blueprints/bundles>) for
packaging details. For a full-
page Playground website,
use a Blueprint like the
gallery demo. It adds a tiny
router at the Playground
document root so the

Symfony `public/` directory
can respond to browser
requests. ## Keep the demo
browser-friendly A
Playground-hosted
framework demo works best
when it: - Does not require a
long-running background
process. - Stores generated
files under the virtual
filesystem. - Avoids native
extensions that are not
compiled into PHP.wasm. -

Avoids frontend build steps at runtime. - Keeps network calls optional or resilient, because browsers may require CORS proxying for third-party services. Those constraints still leave plenty of room for real framework behavior: controllers, routing, dependency injection, templates, forms, HTTP clients, and plain PHP libraries all work when their

PHP dependencies are available.

[] [] #2	
[] [] []	2026-08-29 15:36:35 UTC
[] [] []	2026-08-29 15:36:35 UTC