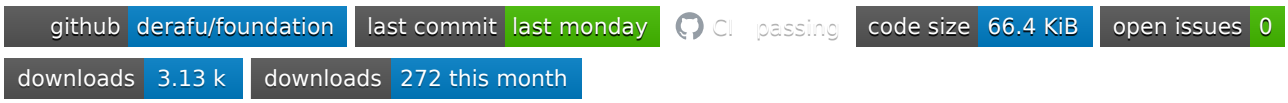
Foundation Project

Meta-package that requires the Derafu packages every web application needs, with the configuration files they share

Anonymous · 09/11/2026 · #php

Base of Dependencies for Derafu's Web Applications



Overview

Derafu Foundation has no PHP code. It declares the packages that every Derafu web application needs, so an application requires only `derafu/foundation` and gets them all, and it provides the configuration files they share.

It is a `library` and not a `metapackage` on purpose: Composer does not write anything of a metapackage in `vendor`, and the configuration files (`resources/config`) have to be there.

```
{
    "require": {
        "derafu/foundation": "dev-main"
    }
}
```

The application declares in its own `composer.json` only what is extra to the base (for example `derafu/auth`).

What it requires

`derafu/captcha`, `derafu/contact-form`, `derafu/content`, `derafu/csrf`, `derafu/data-processor`, `derafu/form`, `derafu/http`, `derafu/kernel`, `derafu/markdown`, `derafu/renderer`, `derafu/routing`, `derafu/session`, `derafu/support`, `derafu/translation`, `derafu/twig`, `derafu/twig-components`, `guzzlehttp/guzzle`, `mpdf/mpdf`, `symfony/cache`, `symfony/yaml` and `utopia-php/openapi`.

The packages whose configuration files foundation imports are declared even when another package already requires them, because if that package stopped requiring them the import would fail.

`utopia-php/openapi` is what lets the docs of `derafu/content` that have an `openapi` in their frontmatter be exported as Markdown and PDF; `derafu/content` suggests it and does not require it.

Configuration

It also provides the configuration that every site repeated, so each site imports one file for its services and one for its routes.

`config/services.yaml` of the site:

```
imports:
  - { resource: '../vendor/derafu/foundation/resources/config/foundation-
services.yaml' }
```

`foundation-services.yaml` imports the shared HTTP, Routing, Renderer, Twig, Translation, Form and content-package configurations. It contributes the site integration through their tags instead of reconstructing those services.

The HTTP handler uses this order:

Middleware	Priority
Static files	1000
Request conversion	900
Client identification	800
Routing	600
Session	500
Flash messages	450
CSRF session	400
Dispatch	200
Response normalization	100

Session persistence stays configured by `derafu/session`. Foundation contributes the existing Session, Flash and CSRF middleware services to `derafu_http.middleware`; it does not create a separate session graph. CORS and throttling remain opt-in application policies.

Routing combines declared routes and controller attributes from `%routes%` with filesystem pages in `%kernel.project_dir%/templates/pages`. The same router matches requests, generates URLs and receives contributed routes.

The shared renderer enables Twig, Markdown and PDF. Each package contributes its own templates, components and extensions. `Derafu\Foundation\Provider\SiteTemplatePathProvider` adds the

site's `templates/` directory at priority `300`, before package defaults. The site needs no provider class or extra path configuration. A template in that directory can override a package template using the same relative name.

`Derafu\Foundation\Bridge\Twig\FoundationExtensionProvider` contributes the routing functions and the `twig` filter. Translation, Form and HTTP contribute their own extensions to the same Twig environment. Form rendering uses the same Renderer service as page rendering.

Foundation also supplies the Guzzle PSR-18 HTTP client, with a ten-second timeout, and one Guzzle HTTP factory shared by the PSR-17 `RequestFactoryInterface` and `StreamFactoryInterface` IDs. The default PSR-6 application cache pool uses the kernel's cache directory.

Client identification is configured through `HTTP_TRUSTED_PROXIES` and `HTTP_CLIENT_IP_HEADERS`, as described in Client IP. By default, forwarding headers are not trusted. The site locale is `APP_LOCALE`, falling back to `app.locale.default` (English).

Captcha-protected forms require `CAPTCHA_PROVIDER`, with the provider's own credentials where applicable. For example, `altcha` uses `CAPTCHA_SECRET_KEY`; `CAPTCHA_PROVIDER=none` explicitly disables captcha. Foundation sets the reCAPTCHA v3 `CAPTCHA_MIN_SCORE` default to `0.7`. It also supplies `CONTENT_CACHE_ENABLED=true` for sites that reference that environment setting in `derafu.content.config`.

`config/routes.yaml` of the site:

```
imports:
  - { resource: '../vendor/derafu/foundation/resources/config/foundation-routes.yaml'
}
```

`resources/config/foundation-routes.yaml` imports the routes of `derafu/content` and `derafu/contact-form`, and defines `homepage` (`templates/pages/home.html.twig`).

The page of the errors is not a route, because it is not a URL to ask for: `foundation-services.yaml` sets the parameter `derafu.http.error_pages` with `templates/error.html.twig` as the page for every error. A site with other pages sets the parameter in its own `services.yaml` (see Error Handling).

Changing the defaults

Set application parameters or add tagged contributions after the import. For example, a custom middleware with priority `400` runs between Routing and Dispatch. Give ordered contributions distinct priorities when their relative position matters. To change error templates or the static directory:

```
parameters:
    derafu.http.static_files.directory: '%kernel.project_dir%/public/static'
    derafu.http.error_pages:
        default: '%kernel.project_dir%/templates/error.html.twig'
```

A deliberately replaced service uses the same interface ID; Symfony replaces its complete definition rather than merging constructor arguments. Template paths, extensions and middleware additions use their tags, so adding one does not require redefining Renderer, Twig or the request handler.

For a site in Spanish, set `APP_LOCALE=es` or override the default locale:

```
parameters:
    app.locale.default: es
```

Foundation imports Translation's complete `translation-services.yaml` and configures its shared `TranslatorInterface` through `derafu.translation.default_locale: '%app.locale%'` and `derafu.translation.fallback_locales: [en]`. The site does not need a translator service definition. To choose another fallback list, override the corresponding `derafu.translation.fallback_locales` parameter.

The routes of `derafu/auth` and the ones of the site are not in foundation: each site declares them in its `routes.yaml`.

Testing a site

`Derafu\Foundation\Testing\SiteTestCase` is a base class for the smoke tests of a site: it boots the kernel with the real `config/` of the site and asks for pages. A site that extends it has the tests of what every site must do, without writing them:

```
#[CoversNothing]
final class WebsiteTest extends SiteTestCase
{
}
```

Test	What it checks
The pages of the site answer	Each path of `paths()` answers with a 200 and without "Internal Server Error". By default `/`, `/contact` and `/contact/success`.
A page that does not exist is not found	An unknown path is a 404, not a 500.
The error of a page that does not exist is the page of the site	The 404 is answered, to a browser, with the HTML of `templates/error.html.twig` and not with the Markdown text that is the last resort for a serious failure.
The error page is not a page of the site	`/error` and `/error404` are a 404 like any other path: an error page is the answer to an error, not something to ask for.

Test	What it checks
The code in `src/` is measured by coverage	If `src/` has PHP files, the `phpunit.xml` of the site has an active `<coverage>` with a `<report>` (one that is commented out does not count). It is the other half of a rule: a site **without** code must have that block commented out, because PHPUnit refuses to make a report of nothing and fails before running any test, with the warning “Configured source filter (include-path: …/src) does not match any files, code coverage will not be processed”. That one can not be a test (none runs), so the `phpunit.xml` says it in a comment next to the block. The day the site has code, this test fails until the block is uncommented.
No route of the site fails in the server	Every route that a GET can ask without parameters answers with less than a 500 (a redirect, a 401 or a 403 is fine). It finds the routes that a package registers but that the site has not configured.
Every route name used in the templates is defined	The `templates/` of the site only refer, with `path()`, `url()` or `is_active_path()`, to routes that the router of the site has. It is checked by reading the templates, with the Twig environment that renders them, so a link to a route that does not exist fails here and not the day someone renders that page. A name that is built at runtime can not be checked by reading, and it fails too.

A site without a contact page overrides `paths()`; a route that needs a service that is not there when testing (a game server, a payment gateway) is listed in `excludedPaths()` so the test of the routes does not ask for it, with a comment that says why. The tests that are only about the site use `get($path, $locale)`, which returns the status and the body:

```
protected function paths(): array
{
    return ['/'];
}
```

```
protected function excludedPaths(): array
{
    return [
        // It asks the Minecraft server through RCON, which is not there when testing.
        '/minecraft/map',
    ];
}
```

Kernel instances are cached separately by project directory and language. The default-language kernel boots in `setUpBeforeClass()`, before PHPUnit takes its photo of the global state: it reads the `.env` of the site and writes its variables to `$_ENV` and `$_SERVER`, and the services that are created later (when a page needs them) still read them. A test class that overrides `setUpBeforeClass()` has to call the parent.

By default, the site directory is the root package installed by Composer. Override the protected static `projectDir()` method when the tests target another project or a fixture site. For a test class in

`tests/src/WebsiteTest.php`:

```
protected static function projectDir(): string
{
    return dirname(__DIR__) . '/fixtures/site';
}
```

The selected directory supplies the site's configuration, templates, content and coverage configuration. Different directories keep independent kernels, even when the tests use the same language.

The tests need `CAPTCHA_PROVIDER=none` in the `phpunit.xml` (`<env name="CAPTCHA_PROVIDER" value="none"/>`) so the forms do not ask for a captcha, and `FORM_CONTACT_ENABLED=false` (`<env name="FORM_CONTACT_ENABLED" value="false"/>`) so the contact page answers with its page of a form that is not available, a 200, and not with a 503 for the webhook that a test does not have. They also need `phpunit/phpunit`, that this package only suggests. A site that does not want the contact form in production sets `FORM_CONTACT_ENABLED=false` in its `.env` too; one that wants it sets the webhook (see Contact form).

How it relates to `derafu/website-skeleton` and `derafu/library-skeleton`

There are three packages, and each has one job:

Package	Is	Use it to
`derafu/foundation`	The dependencies and the base configuration of every web application.	Be required by the sites.
`derafu/website-skeleton`	The template of a site: everything is in its repository (templates, configuration, `public/`, Vite, translations, tests). It requires `derafu/foundation`.	Start a new site.
`derafu/library-skeleton`	The template of a library: the tools, the CI, an example class with its test and the pattern of the translations. It does not depend on `derafu/foundation`.	Start a new library.

A site starts from `derafu/website-skeleton` and depends on `derafu/foundation`. A library starts from `derafu/library-skeleton` and depends on nothing of the web.

Where to go next

- To start a site, see Website Skeleton: the structure of a site, its assets, its tests and its GitHub Actions.
- To start a library, see Library Skeleton: the tools, the strict tests and the pattern of the translations.

Both create the repository with `composer create-project`:

```
composer create-project derafu/website-skeleton www.example.com --stability=dev
composer create-project derafu/library-skeleton my-library --stability=dev
```

Last updated on 09/11/2026