

Docs » Base for Applications Category » Website Skeleton » Project Structure

Project Structure

Project Structure

Anonymous · 08/10/2026

Project Structure

`derafu/website-skeleton` provides a standardized directory structure for your sites (see [Foundation](#) for how it relates to the other packages). This document explains the purpose of each directory and key files.

Root Directories

`.github/`

Contains GitHub-specific configurations:

- `workflows/ci.yml`: Continuous Integration workflow that runs tests and code quality checks.
- `workflows/cd.yml`: Continuous Deployment workflow for automatically deploying. By default, it is configured to deploy to GitHub Pages. Deployment is disabled by default, just remove the `false &&` in the file to enable it.

`app/`

Application bootstrap files:

- `bootstrap.php`: The main bootstrap file that initializes the application runtime.

`assets/`

Frontend assets organized by type:

- `css/`: CSS stylesheets.
- `js/`: JavaScript files.
- `img/`: Images and other media files.

`config/`

Configuration files:

- `routes.yaml`: Route definitions for your application. It imports the routes of [derafu/foundation](#) and the site adds its own.

- `services.yaml`: Service container configuration (dependency injection). It imports the services of `derafu/foundation` and the site declares its controllers, its translation provider and whatever it replaces.

`public/`

Web server document root:

- `index.php`: Application entry point with Front Controller.
- `static/`: Compiled assets (generated by the build process).

Note: If your application doesn't need a Front Controller, by default, it's used only for documentation.

`src/`

Application source code. Organize your PHP classes here according to their namespace. By convention, but not enforced out of Derafu ORG, use:

- `Abstract/`: Abstract classes, with `Abstract` prefix.
- `Contract/`: Interfaces for your application, with `Interface` suffix.
- `Controller/`: Controller classes, with `Controller` suffix.
- `Service/`: Service classes.

`templates/`

Template files for rendering HTML:

- `components/`: Reusable template components (header, footer, etc.).
- `layouts/`: Layout templates that define page structure.
- `pages/`: Page-specific templates.
- `base.html.twig`: Base template that defines the HTML structure for all layouts.
- `error.html.twig`: Error page template.
- `html.html.twig`: HTML wrapper template, used when rendering a markdown o php template.

`tests/`

Test files:

- `fixtures/`: Test fixtures and sample data.
- `src/`: Tests for your source code. You can organize your tests by tests suits, mirroring the structure of the `src/` directory, by features or any other criteria (but not enforced out of Derafu ORG).

`var/`

Temporary files and caches:

- `cache/`: Cache files.
- `logs/`: Log files.
- `tmp/`: Temporary files.

Key Files

Configuration Files

- `.gitignore`: Specifies files that Git should ignore.
- `LICENSE`: MIT license file by default.
- `package.json`: NPM configuration for frontend assets.
- `php-cs-fixer.php`: PHP CS Fixer configuration.
- `phpstan.neon`: PHPStan configuration.
- `phpunit.xml`: PHPUnit configuration.
- `vite.config.js`: Vite build configuration.

Application Files

- `public/index.php`: Main entry point that bootstraps the application.
- `app/bootstrap.php`: Initializes the runtime environment.

Everything Is in the Repository

All the files of the structure are in the repository of `derafu/website-skeleton`: there is no installer that copies files during `composer install` or `composer update`. When you start a site, the files are yours, and to get a change of the template you compare with it (or with `git`) and apply what you want.

Extending the Structure

When creating a new site based on `derafu/website-skeleton`, you should:

1. Keep the existing directory structure.
2. Add your own directories as needed.
3. Follow PSR-4 autoloading standards for your PHP classes.
4. Place tests in the corresponding structure within the `tests/` directory.

The structure is designed to be flexible while providing a consistent organization pattern across projects.

Last updated on 08/10/2026