

Introduction

Entity repositories for PHP with file and enum sources, in-memory queries and custom repository factories

Anonymous · 20/10/2026

Lightweight Data Source Management for PHP

[github](#) [derafu/repository](#) [last commit](#) [today](#) [CI](#) [passing](#) [code size](#) [161 KiB](#) [open issues](#) [0](#)

[downloads](#) [6.09 k](#) [downloads](#) [1 k this month](#)

Derafu Repository provides an entity query contract, an in-memory repository and a manager that creates repositories through factories. Use the built-in data provider for catalogues and lookup tables, or register factories for repositories with their own infrastructure dependencies.

Installation

PHP 8.5 or later is required.

```
composer require derafu/repository
```

To read YAML files, also install the optional YAML component:

```
composer require symfony/yaml
```

Components

Component	Responsibility
<code>RepositoryInterface</code>	Entity queries, counting and the entity class name
<code>Repository</code>	In-memory implementation; also exposes container operations
<code>DataSourceInterface</code>	Read a complete dataset as a PHP array
<code>DataProvider</code>	Resolve sources, optionally cache raw data, normalize rows and retain loaded data
<code>DataProviderRepositoryFactory</code>	Construct in-memory repositories using data-provider sources and entity metadata
<code>RepositoryFactoryInterface</code>	Construct a repository using its own dependencies

Component	Responsibility
RepositoryManager	Select a factory, create a repository on demand and reuse its instance

Choosing a Data Access Path

For a small or medium-sized catalogue, use a PHP, JSON or YAML file, a backed enum, or a custom data source. The provider reads the complete dataset; `Repository` filters and sorts it in memory.

For direct database or remote-service queries, register a repository factory. Its repository can execute each query against the backend and implement `RepositoryInterface` without inheriting the in-memory container. The package supplies the extension point; it does not supply a production SQL repository or an ORM.

Reusing a repository instance does not determine how fresh its results are. The built-in in-memory repository uses loaded data; a custom repository controls its own queries and caching.

Guides

- [Basic usage](#): create a repository or use the manager.
- [Data sources](#): files, enums, normalization, caching and extension points.
- [Entities and mapping](#): entity classes, identifiers and specialized repositories.
- [Querying](#): lookup, filtering, ordering and Doctrine Criteria.
- [Repository factories](#): repositories with independent dependencies.
- [Symfony integration](#): configure sources, factories and translation resources.

Last updated on 20/10/2026