

Entity Manager

Create an entity manager from an array, with the options of Doctrine DBAL and ORM

Anonymous · 26/10/2026

Entity Manager

`EntityManagerFactory` creates an `EntityManager` from an array of options. It implements `Derafu\Orm\Contract\EntityManagerFactoryInterface`, so it can be a service that is injected and replaced.

```
use Derafu\Orm\Factory\EntityManagerFactory;

$entityManager = (new EntityManagerFactory())->create([
    'dbal' => [
        'url' =>
'mysql://app:secret@127.0.0.1:3306/app?serverVersion=8.0&charset=utf8mb4',
    ],
    'orm' => [
        'paths' => [__DIR__ . '/src/Entity'],
        'dev_mode' => false,
        'naming_strategy' => Doctrine\ORM\Mapping\UnderscoreNamingStrategy::class,
    ],
]);
```

The configuration has two sections, `dbal` and `orm`, with the options that Doctrine has. The entities are mapped with PHP attributes, and Doctrine's native lazy objects are always enabled.

The parameters of the connection are created by a `ConnectionParametersFactoryInterface`, which `EntityManagerFactory` receives in its constructor. `ConnectionParametersFactory` is the default one.

dbal

Every parameter of the connection that DBAL's `DriverManager` understands, with its official name (`driver`, `path`, `memory`, `host`, `port`, `user`, `password`, `dbname`, `charset`, `serverVersion`, `defaultTableOptions`, `driverOptions`...). Doctrine checks them. The section also has these options:

Option	Value
<code>url</code>	A DSN, read with DBAL's <code>DsnParser</code> . The schemes are <code>mysql</code> , <code>mysqli</code> , <code>pgsql</code> , <code>postgresql</code> , <code>postgres</code> , <code>sqlite</code> , <code>sqlserver</code> and <code>mssql</code> . The parameters of the DSN win over the other ones

Option	Value
types	Custom types, as name => class of a Doctrine\DBAL\Types\Type. The uuid type is always registered
schema_filter	A regular expression of the tables that Doctrine keeps

A SQLite connection needs a path, unless it is memory. A DSN like `sqlite:///memory:` is in memory, and a relative path in a SQLite DSN is relative to the current directory.

The DBAL type registry is global. A type that is already registered is replaced, so the same configuration can be used more than once.

orm

Option	Value
paths	List of directories with the entities
dev_mode	true or false
naming_strategy	A class, or an instance, of Doctrine\ORM\Mapping\NamingStrategy
resolve_target_entities	interface => class of the entity that stands for the interface
dql	Custom functions of DQL: string_functions, numeric_functions and datetime_functions, each as name => class of a FunctionNode
metadata_cache, query_cache, result_cache, hydration_cache	A PSR-6 pool each

Without the caches, `ORMSetup` of Doctrine chooses one: in memory in development mode, and APCu, Memcached or Redis if they are available otherwise.

Errors

The options are read with [derafu/config](#), through the typed readers of `derafu/array`, so one with another type raises their translatable error, naming the option (for example The key `"orm.paths"` must be of type `list<string>`, string given.). `DatabaseConfigurationException` is for what the package itself checks: a database that is not configured, a DSN that is not valid or has no known driver, a SQLite connection without a file, a regular expression that is not valid, a class that is not of the expected type, and a map whose keys or values are not strings.

Last updated on 26/10/2026