



# DOCTRINE

## INTRODUCCIÓN

Tutorial Luis Miguel

# INICIALIZACIÓN Y DEPENDENCIAS

```
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> composer init
```

Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

Package name (<vendor>/<name>) [**luism/doctrine-presentacion**]: daw/doctrine

Description []: Proyecto presentación de doctrine ORM

Author [**LuisCoding05 <162503462+LuisCoding05@users.noreply.github.com>**], n to skip]: n

Minimum Stability []:

Package Type (e.g. library, project, metapackage, composer-plugin) []: project

License []: MIT

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? yes

Search for a package: doctrine

Found 15 packages matching doctrine

- [0] doctrine/orm
- [1] doctrine/lexer
- [2] doctrine/instantiator
- [3] doctrine/inflector
- [4] doctrine/event-manager
- [5] doctrine/dbal

PSR-4 autoloading configured. Use "namespace Daw\Doctrine;" in src/  
Include the Composer autoloader with: require 'vendor/autoload.php';

## IMPORTANTE

Lo primero que haremos será crear en nuestro directorio de htdocs la carpeta de proyecto, dentro haremos el proceso de instalación de dependencias y demás cosas, donde al finalizar nos dará un namespace del que deberemos acordarnos

Las dependencias a instalar, en este caso son

1. doctrine/orm
2. doctrine/dbal
3. symfony/cache
4. vlucas/phpdotenv

Add PSR-4 autoload mapping? Maps namespace "Daw\Doctrine" to the

```
{  
    "name": "daw/doctrine",  
    "description": "Proyecto presentación de doctrine ORM",  
    "type": "project",  
    "require": {  
        "doctrine/orm": "^3.3",  
        "doctrine/dbal": "^4.2",  
        "symfony/cache": "^7.2"  
    },  
    "license": "MIT",  
    "autoload": {  
        "psr-4": {  
            "Daw\\Doctrine\\": "src/"  
        }  
    }  
}
```

Do you confirm generation [yes]? yes  
Would you like to install dependencies now [yes]?

# BOOTSTRAP . PHP, CONFIG Y . ENV'

En esta parte vamos a crear el archivo de bootstrap el cual contiene los datos de conexión de la base de datos, estos datos lo obtenemos de un archivo .env que contiene las variables de entorno

```
.env.example
1 DB_HOST=localhost
2 DB_PORT=3306
3 DB_NAME=your_database
4 DB_USER=your_username
5 DB_PASSWORD=your_password
6 DB_CHARSET=utf8mb4
```

```
bootstrap.php > ...
1 <?php
2 use Doctrine\DBAL\DriverManager;
3 use Doctrine\ORM\EntityManager;
4 use Doctrine\ORM\ORMSetup;
5 use Dotenv\Dotenv;
6
7 require_once "vendor/autoload.php";
8
9 // Cargar variables de entorno
10 $dotenv = Dotenv::createImmutable(paths: __DIR__);
11 $dotenv->load();
12
13 // Crea la configuración de Doctrine ORM
14 $config = ORMSetup::createAttributeMetadataConfiguration(
15     paths: __DIR__ . '/src',
16     isDevMode: true,
17 );
18
19 // Configuración de conexión de MySQL
20 $connection = DriverManager::getConnection(params: [
21     'driver' => 'pdo_mysql',
22     'host' => $_ENV['DB_HOST'],
23     'port' => $_ENV['DB_PORT'],
24     'dbname' => $_ENV['DB_NAME'],
25     'user' => $_ENV['DB_USER'],
26     'password' => $_ENV['DB_PASSWORD'],
27     'charset' => $_ENV['DB_CHARSET'],
28 ], config: $config);
29
30 // obtiene el entity manager
31 $entityManager = new EntityManager(conn: $connection, config: $config);
```

```
bin > doctrine
1 <?php
2 use Doctrine\ORM\Tools\Console\ConsoleRunner;
3 use Doctrine\ORM\Tools\Console\EntityManagerProvider\SingleManagerProvider;
4
5 // Conectar con el archivo bootstrap.php
6 require __DIR__ . '/../bootstrap.php';
7
8 ConsoleRunner::run(
9     entityManagerProvider: new SingleManagerProvider(entityManager: $entityManager)
10 );
```

# INTERACCIÓN CON EL ORM

Ya podemos acceder al orm, por medio del siguiente comando:

```
php ubicacionDelArchivoConfig
```

Este comando nos mostrará que comandos podemos utilizar para este archivo y que hace cada uno

```
Doctrine Command Line Interface 3.3.1.0

Usage:
  command [options] [arguments]

Options:
  -h, --help          Display help for the given command. When no command is given display help for the list command
  --silent           Do not output any message
  -q, --quiet          Only errors are displayed. All other output is suppressed
  -V, --version         Display this application version
  --ansi|--no-ansi    Force (or disable --no-ansi) ANSI output
  -n, --no-interaction Do not ask any interactive question
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  completion          Dump the shell completion script
  help                Display help for a command
  list               List commands
  dbal
  dbal:run-sql        Executes arbitrary SQL directly from the command line.

  orm
    orm:clear-cache:metadata      Clear all metadata cache of the various cache drivers
    orm:clear-cache:query         Clear all query cache of the various cache drivers
    orm:clear-cache:region:collection  Clear a second-level cache collection region
    orm:clear-cache:region:entity   Clear a second-level cache entity region
    orm:clear-cache:region:query   Clear a second-level cache query region
    orm:clear-cache:result        Clear all result cache of the various cache drivers
    orm:generate-proxies        [orm:generate:proxies] Generates proxy classes for entity classes
    orm:info                  Show basic information about all mapped entities
    orm:mapping:describe       Display information about mapped objects
    orm:run-dql                Executes arbitrary DQL directly from the command line
    orm:schema-tool:create     Processes the schema and either create it directly on EntityManager Storage Connection or generate the SQL output
    orm:schema-tool:drop        Drop the complete database schema of EntityManager Storage Connection or generate the corresponding SQL output
    orm:schema-tool:update     Executes (or dumps) the SQL needed to update the database schema to match the current mapping metadata
    orm:validate-schema        Validate the mapping files
```

Ahora ejecutaremos el comando:

```
php cli-config orm:schema-tool:create
```

No habrá ningún error, pero como no tenemos entidades nos dirá que no hay clases de metadatos para procesar

```
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php cli-config orm:schema-tool:create
[OK] No Metadata Classes to process.
```

Procederemos a continuación a crear una entidad usuario que concuerde con una tabla de nuestra base de datos

# BASE DE DATOS

- He creado una tabla User con usuarios, tiene 5 columnas, teniendo un id de autoincremento, nombre, apellido, email y clave, pues ahora vamos a crear esta entidad en el código. Debemos acordarnos del namespace porque lo vamos a usar

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	<b>id</b> 	int(11)			No	Ninguna		AUTO_INCREMENT
2	<b>nombre</b>	varchar(100)	utf8mb4_general_ci		No	Ninguna		
3	<b>apellido</b>	varchar(100)	utf8mb4_general_ci		No	Ninguna		
4	<b>email</b>	varchar(200)	utf8mb4_general_ci		No	Ninguna		
5	<b>clave</b>	varchar(100)	utf8mb4_general_ci		No	Ninguna		

Creamos la carpeta entity dentro de src y creamos la entidad conforme a como está en la base de datos

```
src > Entity > User.php > PHP Intelephense > User
1  <?php
2  // file path: /C:/xampp/htdocs/Cositas/doctrineProject/src/Entity/User.php
3  namespace Daw\Doctrine\Entity;
4  use Doctrine\ORM\Mapping as ORM;
5
6  #[ORM\Entity]
7  #[ORM\Table(name: 'user')]
8  4 references | 0 implementations
9  class User
10 {
11     #[ORM\Id]
12     #[ORM\Column(type: 'integer', length: 11)]
13     #[ORM\GeneratedValue]
14     1 reference
15     private ?int $id = null;
16
17     #[ORM\Column(type: 'string', length: 100, nullable: false, options: ['collation' => 'utf8mb4_general_ci'])]
18     3 references
19     private string $nombre;
20
21     #[ORM\Column(type: 'string', length: 100, nullable: false, options: ['collation' => 'utf8mb4_general_ci'])]
22     3 references
23     private string $apellido;
24     #[ORM\Column(type: 'string', length: 200, nullable: false, options: ['collation' => 'utf8mb4_general_ci'])]
25     3 references
26     private string $email;
27
28     #[ORM\Column(type: 'string', length: 100, nullable: false, options: ['collation' => 'utf8mb4_general_ci'])]
29     3 references
30     private string $clave;
31
32     // Constructor
33     0 references | 0 overrides
34     public function __construct(string $nombre, string $apellido, string $email, string $clave) {
35         $this->nombre = $nombre;
36         $this->apellido = $apellido;
37         $this->email = $email;
38         $this->clave = $clave;
39     }
40
41     // Getters y Setters
```

# FUSIÓN DE ESQUEMAS

Con [orm:schema-tool:update](#) actualizamos el esquema

```
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php cli-config orm:schema-tool:update
```

```
[OK] Nothing to update - your database is already in sync with the current entity metadata.
```

```
PS C:\xampp\htdocs\Cositas\doctrinePresentacion>
```

Con [orm:validate-schema](#) validamos que todos las entidades mapeadas estén correctas y sincronizadas con la base de datos, el sistema nos informa de esto

```
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php cli-config orm:validate-schema
```

```
Mapping
```

```
Mapping
```

```
-----
```

```
[OK] The mapping files are correct.
```

```
Database
```

```
-----
```

```
[OK] The database schema is in sync with the mapping files.
```

# CREACIÓN DE PRODUCTOS

Ahora creamos la entidad producto que después introduciremos en la BBDD a través del ORM y

```
src > OptionsProducto.php > PHP Intelephense > 🌐 crearProducto
1  <?php
2  require_once 'vendor/autoload.php';
3  require_once 'bootstrap.php';
4
5  use Daw\Doctrine\Entity\Producto;
6
7  function crearProducto($nombre, $precio, $stock = null, $descripcion = null): Producto|null {
8      global $entityManager;
9
10     $producto = new Producto(nombre: $nombre, precio: $precio, stock: $stock, descripcion: $descripcion);
11
12     try {
13         $entityManager->persist(object: $producto);
14         $entityManager->flush();
15         echo "Producto creado exitosamente\n";
16         return $producto;
17     } catch (Exception $e) {
18         echo "Error al crear producto: " . $e->getMessage() . "\n";
19         return null;
20     }
21 }
```

## Código de la entidad producto

```
src > Entity > 📄 Producto.php > PHP Intelephense > 🏷 Producto
1  <?php
2  namespace Daw\Doctrine\Entity;
3
4  use Doctrine\ORM\Mapping as ORM;
5
6  #[ORM\Entity]
7  #[ORM\Table(name: 'productos')]
6 references | 0 implementations
8  class Producto
9  [
10     #[ORM\Id]
11     #[ORM\Column(type: 'integer')]
12     #[ORM\GeneratedValue]
1 reference
13     private ?int $id = null;
14
15     #[ORM\Column(type: 'string', length: 150, nullable: false)]
3 references
16     private string $nombre;
17
18     #[ORM\Column(type: 'float', nullable: false)]
3 references
19     private float $precio;
20
21     #[ORM\Column(type: 'integer', nullable: true)]
3 references
22     private ?int $stock = null;
23
24     #[ORM\Column(type: 'text', nullable: true)]
3 references
25     private ?string $descripcion = null;
26
27     #[ORM\Column(type: 'datetime', nullable: true)]
2 references
28     private ?\DateTimeInterface $fechaCreacion = null;
29
30     // Constructor
1 reference | 0 overrides
31     public function __construct($nombre = '', $precio = 0.0, $stock = null, $descripcion = null) {
32         $this->nombre = $nombre;
33         $this->precio = $precio;
34         $this->stock = $stock;
35         $this->descripcion = $descripcion;
36         $this->fechaCreacion = new \DateTime();
37     }
38 }
```

# FUSIÓN DE ESQUEMAS

Ahora como hemos creado una nueva entidad que es producto podemos ver que el mapeo está bien pero no está sincronizado con la base de datos

```
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php cli-config orm:validate-schema

Mapping
-----
[OK] The mapping files are correct.

Database
-----
[ERROR] The database schema is not in sync with the current mapping file.
```

Con [orm:schema-tool:create](#) creamos el esquema de las entidades que tengamos, ahora lo que haremos será crear una nueva entidad desde php para probar este comando, este comando genera conflicto si tenemos otras entidades ya creadas así que podemos usar [orm:schema-tool:update --force](#) para que cree solo lo necesario

```
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php cli-config orm:schema-tool:update --force
Updating database schema...

1 query was executed

[OK] Database schema updated successfully!

PS C:\xampp\htdocs\Cositas\doctrinePresentacion>
```

# RESULTADO Y COMANDO DE INTERÉS

Con `php_cli-config_orm:info` podemos ver información básica de las entidades mapeadas

```
● PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php cli-config orm:info
  Found 2 mapped entities:
    [OK] Daw\Doctrine\Entity\Producto
    [OK] Daw\Doctrine\Entity\User
```

## Resultado:

<input type="checkbox"/>	Perfilando	[ Editar en línea ]	[ Editar ]	[ Explicar SQL ]	[ Crear ]	
	<b>id</b>	<b>nombre</b>	<b>precio</b>	<b>stock</b>	<b>descripcion</b>	<b>fechaCreacion</b>

Con `php cli-config orm:mapping:describe` Daw\Doctrine\Entity\Producto podemos ver información detallada de una entidad

# RESULTADO

Ahora ejecutaremos un script en main.php para crear un nuevo producto

```
23 | $nuevoProducto = crearProducto(nombre: 'Laptop Gaming', precio: 1299.99, stock: 10, descripció
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE COMMENTS powershell + ▾
default Null
type datetime
fieldName fechaCreacion
columnName fechaCreacion
-----
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php cli-config orm:info
Found 2 mapped entities:
[OK] Daw\Doctrine\Entity\Producto
[OK] Daw\Doctrine\Entity\User
PS C:\xampp\htdocs\Cositas\doctrinePresentacion> php main.php
Producto creado exitosamente
```

Y observamos que podemos ver la tabla con éxito

SELECT * FROM `productos`						
<input type="checkbox"/> Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]						
<input type="checkbox"/> Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla						
Opciones extra						
←→	▼	id	nombre	precio	stock	descripcion
<input type="checkbox"/>	<a href="#">Editar</a>	<a href="#">Copiar</a>	<a href="#">Borrar</a>	1	Laptop Gaming	1299.99
	<input type="checkbox"/>	<a href="#">Seleccionar todo</a>	Para los elementos que están marcados:			
	<input type="checkbox"/>	<a href="#">Editar</a>	<a href="#">Copiar</a>	<a href="#">Borrar</a>	<a href="#">Exportar</a>	

Código del proyecto:  
[github.com/LuisCoding05/doctrineTest](https://github.com/LuisCoding05/doctrineTest)

# PREGUNTAS

# GRACIAS