Introducción a Doctrine ORM

Bloque 2 – Patrón MVC con acceso a Datos, DOCTRINE 2º
DAW – Desarrollo Web en Entorno Servidor

Fernando Díaz-Alonso Dorado y Jesús Molina Hernández

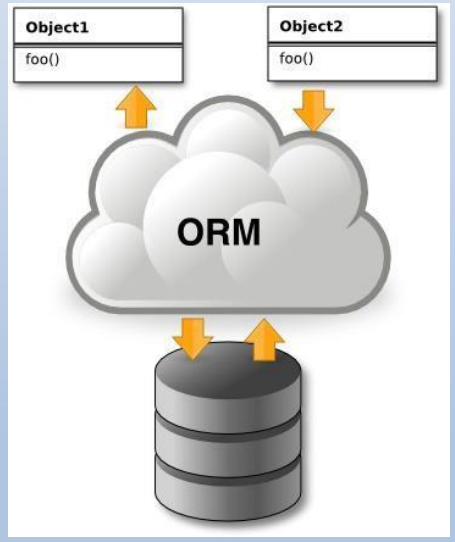
Florida Universitària

ORM

¿Qué es?

ORM → Object Relational Mapping

- Es una técnica de programación que permite **mapear tablas** de bases de datos a objetos de una aplicación.
- Las **clases** → tablas de la base de datos.
- Los **objetos** → los registros de las tablas.



Ventajas:

- Evita escribir SQL manualmente.
- Mantenimiento más fácil del código.
- Código más limpio y orientado a objetos.
- Cambios en la BD no requieren trabajo extra en el código.

Doctrine

¿Qué es?

Doctrine son un conjunto de librerias PHP enfocadas al almacenamiento de bases de datos y mapeo de objetos.

Componentes principales:

- **Doctrine ORM:** El mapeador objeto-relacional.
- **Doctrine DBAL:** Una capa de abstracción para la base de datos.
- **Doctrine Migrations:** Herramienta para gestionar cambios en la base de datos.

Doctrine

¿Qué ofrece Doctrine?

Nos permite como programadores concentrarnos en la lógica empresarial orientada a objetos y no en la persistencia, manteniendo ambas separadas.

- Simplificación del acceso a la base de datos.
- Permite cambiar de base de datos sin modificar el código.
- Evita inyecciones SQL con consultas preparadas.
- Flexibilidad para trabajar con datos complejos.
- Comunidad amplia y documentación robusta.
- Soporte para migraciones y cambios en la estructura de la base de datos.

Doctrine

¿Cómo se instala?

Necesitamos Composer para poder instalar en nuestro proyecto Doctrine.

Instalaremos la versión ^3

Podemos agregar Doctrine mediante la instrucción:

```
composer require doctrine/orm:^3 doctrine/dbal:^4 symfony/cache:^7
```

Conseguiremos en **composer.json** las siguientes líneas:

```
"require": {
    "doctrine/orm": "^3",
    "doctrine/dbal": "^4",
    "symfony/cache": "^7"
},
```

Podéis seguir el tutorial de su página web ***Primeros pasos con Doctrine***:

https://www.doctrine-project.org/projects/doctrine-orm/en/3.5/tutorials/getting-started.html

Doctrine

Uso Básico

Usaremos MVC con un par de cambios:
Se cambia la carpeta "**src/model**" por

1. "**src/Entity**" → Entidades
2. "**src/Repository**"→ Repositorios

- Necesitaremos una instancia del "**EntityManager**".

```php
//Primero debemos obtener la ruta donde se encuentran las entidades.
$path = array(__DIR__.'/../'.$_ENV['ENTITYFOLDER']);
//Definimos si estamos trabajando en desarrollo o no
$isDevMode = boolval($_ENV['DEVELOP_MODE']);
//Configuramos los parámetros de la conexión con la BB.DD.
$dbParams = array(
    'host'      => $_ENV['DBSERVER'],
    'driver'    => $_ENV['DBDRIVER'],
    'user'      => $_ENV['DBUSER'],
    'password'  => $_ENV['DBPASSWORD'],
    'dbname'    => $_ENV['DBNAME']
);
//Creamos la configuración para la conexión con un método estatico de Doctrine
$config = ORMSetup::createAttributeMetadataConfiguration($path,$isDevMode);
//Creamos una conexión para poder trabajar
$connection = DriverManager::getConnection($dbParams,$config);
//E instanciamos el entityManager definitivo con la conexión establecidad y la configuración cargada.
$this->entityManager = new \Doctrine\ORM\EntityManager($connection,$config);
```

- Parámetros de configuración en detalle en este link:

https://www.doctrine-project.org/projects/doctrine-orm/en/3.5/reference/configuration.html

Doctrine

Definiciones

- Las **Entities** son clases PHP que representan tablas en la base de datos. Cada tabla tiene su propia entidad.
- Un **Repository** es una clase que te proporciona métodos específicos para realizar consultas en una entidad particular. Es una herramienta extra para manejar queries más complejas (generalmente de consulta) con una tabla concreta.
- El **EntityManager** es el componente central de Doctrine que maneja todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para las entidades. Es como el interfaz con la BD.

Doctrine

Entidades

Las entidades son objetos / clases PHP que se pueden identificar en muchas solicitudes por medio de un identificador único o una clave primaria.

- Las clases no necesitan extender ninguna interfaz o clase abstracta.
- La clase de una entidad no debe de ser final ni contener métodos finales.
- No deben implementar ni **clonar** ni **despertar,** salvo que lo haga de forma segura.
- Una entidad contiene propiedades persistentes. Una propiedad persistente es una variable de instancia de la entidad que se guarda y recupera de la base de datos mediante las capacidades de mapeo de datos de Doctrine.

Doctrine

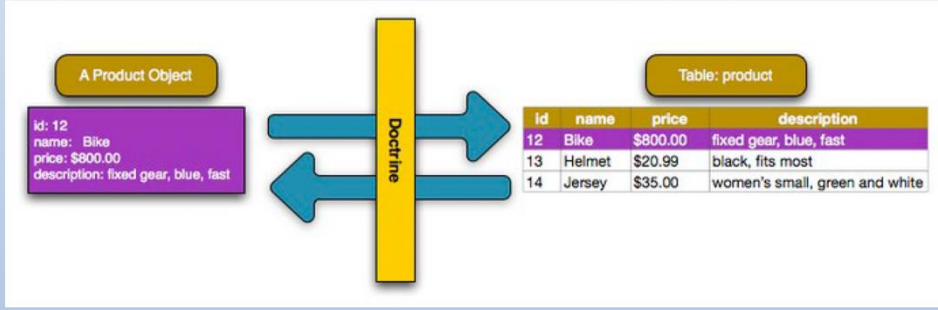
Entidades II

En definitiva, las entidades son clases que representan a tablas de la BB.DD. Concretamente una entidad por cada tabla.

Los campos se traducen en propiedades de clase, definiendo una para cada campo.

De cada campo debemos definir una serie de características:

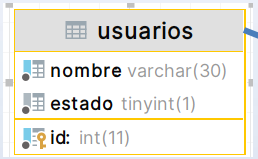
- Nombre
- Tipo
- Longitud (según corresponda)
- ¿Puede ser nulo?
- Y otras características



Para conocer las definiciones con más profundidad podemos acceder al link:
https://www.doctrine-project.org/projects/doctrine-orm/en/3.5/reference/basic-mapping.html#basic-mapping

Doctrine

Entidades III



Por ejemplo, para la tabla podemos definir:

- El campo **id:**
  - Tendrá tipo ***integer*** al ser un “int” en la BB.DD.
  - Tendrá la característica de ser ***id*** de la entidad.
  - Y además será ***GeneratedValue.***
- El campo **nombre:**
  - Tendrá tipo ***string*** al ser un “varchar” en la BB.DD.
  - Tendrá una longitud máxima de 30.
  - Definiremos que es una clave única.
- El campo **estado:**
  - Tendrá tipo ***boolean*** al ser un “tynyint” en la BB.DD.

Podemos ver el ejemplo de la clase Usuarios ya con los getter y Setter de los campos.

```php
no usages  new *
#[Entity(repositoryClass: UsuariosRepository::class)]
#[Table(name:'usuarios')]
class Usuarios
{
    1 usage
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    2 usages
    #[Column(name: 'nombre', type: 'string', unique: true)]
    private string $name;

    2 usages
    #[Column(name: 'estado', type: 'boolean')]
    private bool $state;

    /**
     * @return int
     */
    new *
    public function getId(): int
    {
        return $this->id;
    }

    /**
     * @return string
     */
    new *
    public function getName(): string
    {
        return $this->name;
    }
```


Florida Universitària

Doctrine

Entidades IV: Tipos de datos

Tipo	Descripción
string	Una cadena de texto.
integer	Un número entero.
smallint	Un número entero pequeño (tamaño reducido).
bigint	Un número entero grande.
boolean	Un valor `true` o `false`.
decimal	Un número decimal (requiere especificar `precision` y `scale`).
float	Un número decimal en punto flotante.
date	Una fecha (sin hora).
datetime	Una fecha y hora combinadas.
datetimetz	Una fecha y hora con zona horaria.
time	Una hora (sin fecha).
text	Texto largo.
binary	Datos binarios.
blob	Un tipo `BLOB` de la base de datos para almacenar binarios.

Doctrine

Repositorios

Los repositorios son clases que nos proporcionan acceso a los métodos estándares de consulta de una BB.DD. (findBy, findAll, etc…)

Además, nos permiten definir métodos adicionales para crear consultas más complejas.

Para crear las consultas podremos hacerlo:

- A través de DOCTRINE QUERY LANGUAGE
- A través de QUERY BUILDER
- O directamente usando SQL
  - Con Native QUERY
  - Con PDO
    - En ese caso se devuelven datos en crudo, no objetos

Doctrine

Repositorios II

Para añadir funcionalidad a nuestro repositorio usaremos la clase abstracta de Doctrine EntityRepository.

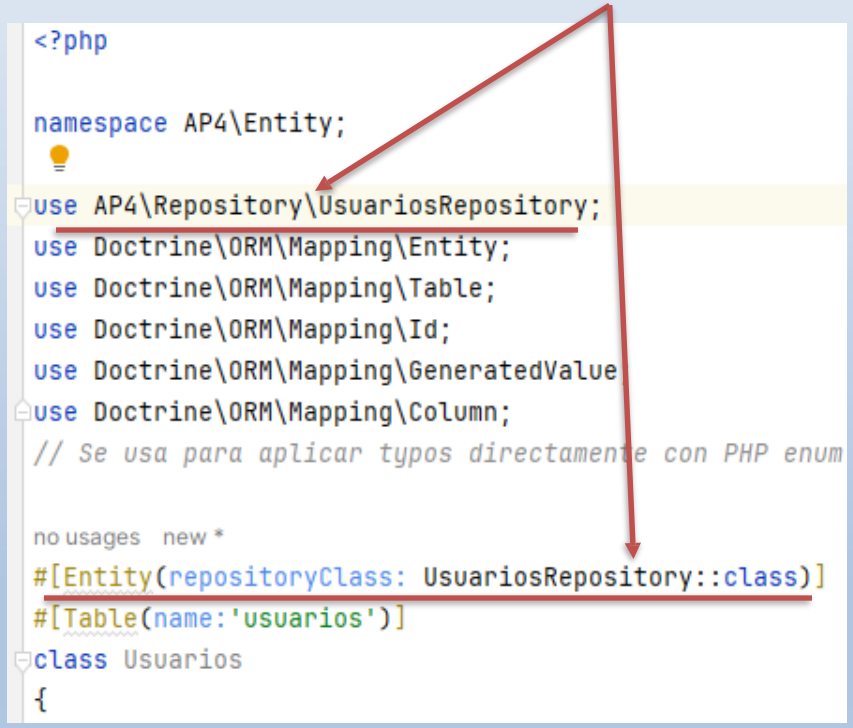
```php
<?php

namespace AP4\Repository;

use Doctrine\ORM\EntityRepository;

no usages  new *
class UsuariosRepository extends EntityRepository
{
    /*
     * Por defecto no es necesario crear ningún método, porque por defecto al extender el EntityRepository
     * nos traemos los métodos find(), findOneBy, findBy(), findAll()
     */
}
```

En la clase podremos añadir los métodos personalizados al repositorio, es decir, que estamos añadiendo esos métodos al modelo.

Doctrine

Repositorios III

En las entidades deberemos enlazar el repositorio asociado a esa entidad



Florida Universitària

Doctrine

Repositorios IV – Métodos por defecto

Al extender EntityRepository tenemos acceso a los métodos por defecto implementados por Doctrine

- **find:**

Busca un registro a través de su clave primaria → *find($id);*

- **findOneBy:**

Busca un registro por uno o más criterios

```
findOneBy([
    'campo1' => 'valor_buscado1',
    'campo2' => 'valor_buscado2',
    …
]);
```

- **findBy:**

Busca todos los registros que coinciden con los criterios y los puede ordenar.

```
findBy(
    ['campo1' => 'valor_buscado1',
    'campo2' => 'valor_buscado2'],
    ['campo_orden' => ASC|DESC]);
```

- **findAll:**

Busca todos los registros de una entidad.

```
findAll();
```

Doctrine

Obtención de datos

Para poder obtener los datos debemos instanciar la clase de Doctrine → EntityManager. (nuestro Database del MVC).

Para facilitarnos el trabajo creamos una nueva clase EntityManager que personalizamos a nuestra configuración y con el que podremos coger la entidad que necesitemos.

```php
class EntityManager
{
    2 usages
    private Doctrine\ORM\EntityManager $entityManager;

    /**
     * @return Doctrine\ORM\Doctrine
     */
    new *
    public function getEntityManager(): Doctrine\ORM\Doctrine
    {
        return $this->entityManager;
    }

    new *
    public function __construct()
    {
        //Primero debemos obtener la ruta donde se encuentran las entidades.
        $path = array(__DIR__.'/../'.$_ENV['ENTITYFOLDER']);
        //Definimos si estamos trabajando en desarrollo o no
        $isDevMode = boolval($_ENV['DEVELOP_MODE']);
        //Configuramos los parámetros de la conexión con la BB.DD.
        $dbParams = array(
            'host'      => $_ENV['DBSERVER'],
            'driver'    => $_ENV['DBDRIVER'],
            'user'      => $_ENV['DBUSER'],
            'password'  => $_ENV['DBPASSWORD'],
            'dbname'    => $_ENV['DBNAME']
        );
        //Creamos la configuración para la conexión con un método estatico de Doctrine
        $config = ORMSetup::createAttributeMetadataConfiguration($path,$isDevMode);
        //Creamos una conexión para poder trabajar
        $connection = DriverManager::getConnection($dbParams,$config);
        //E instanciamos el entityManager definitivo con la conexión establecidad y la configuración cargada.
        $this->entityManager = new \Doctrine\ORM\EntityManager($connection,$config);
    }
}
```

Doctrine

Obtención de datos II

En nuestro controlador debemos obtener el EntityManager, y a través de él llamamos al repositorio de la entidad que necesitamos.

```php
<?php

namespace AP4\Controllers;

use AP4\Core\AbstractController;
use AP4\Core\EntityManager;
use AP4\Entity\Usuarios;

no usages  new *
class MainControllers extends AbstractController
{
    new *
    public function main():void
    {
        //Obtenemos el entityManager para poder trabajar con la instancia de Doctrine
        $entityManager = (new EntityManager())->getEntityManager();
        //Ahora a partir del repositorio podemos trabajar con la entidad.
        $usuarios = $entityManager->getRepository(Usuarios::class);
        //Podemos realizar una consulta predefinida o personalizada que este contenida en el repositorio.
        $data = $usuarios->findAll();
        $this->render(
            "list.html.twig",[
                'resultados'=>$data
            ]);
    }
}
```

Usamos el método por defecto de doctrine

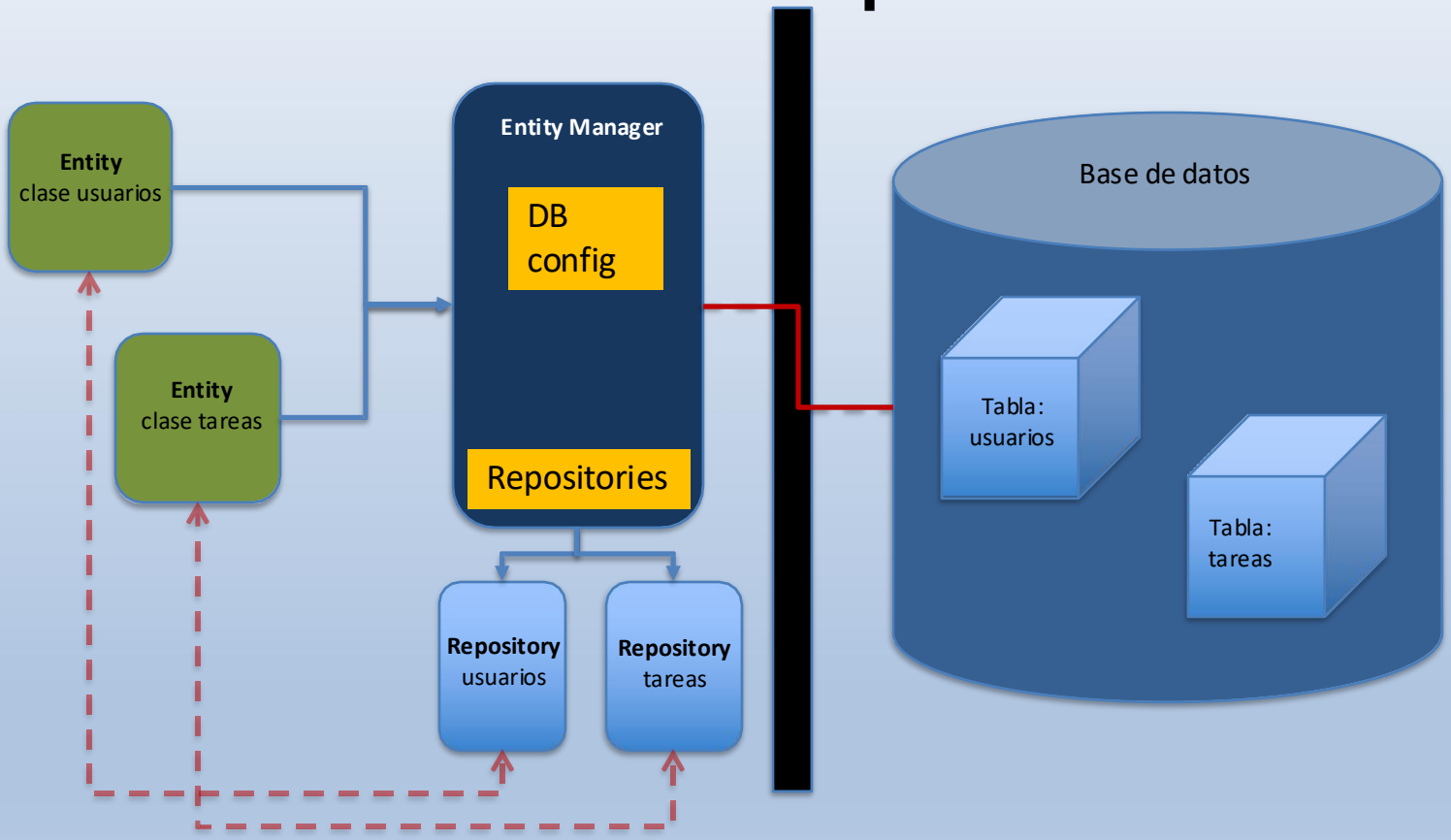
Doctrine

EntityManager >> Funciones (Tema 5)

- **Guardar datos**
  - persist($entity)
    - Marca una entidad para ser insertada o actualizada en la base de datos.
  - flush()
    - Ejecuta todas las operaciones pendientes que han sido marcadas con persist. Es el comando que sincroniza el estado del EntityManager con la base de datos.
    - Utilizado para **aplicar cambios** pendientes en la base de datos.
- **Buscar entidades**
  - find($entityName, $id)
    - Busca y devuelve una entidad específica por su **ID**. Es una operación de búsqueda simple por clave primaria.
  - getRepository($entityName):
    - Devuelve el repository asociado a una entidad específica. El repository es una clase especializada para hacer consultas más complejas a la base de datos.
    - Aunque puedes realizar consultas directamente desde el EntityManager, utilizar un repository suele ser más cómodo y organizado.
- **Eliminar entidades**
  - remove($entity):
    - Marca una entidad para ser eliminada de la base de datos.

Doctrine

Relación entre los 3 conceptos



Florida Universitària

Doctrine

Flujo de trabajo básico

Definir el EntityManager: Configuración de conexión a la base de datos, mapeo y otros parámetros generales. Solo necesitas **un EntityManager** para toda la base de datos.

Crear las Entities: Son las clases PHP que representan las tablas de la BD. Deben incluir la definición de las tablas de BD.

Repositories: Se crea automáticamente uno para cada Entity. No necesitas modificarlos a menos que necesites lógica de consulta personalizada. Accedes a ellos a través del EntityManager.

Acceder a la BD:

- Usa el EntityManager para manejar las entidades.
- Usa los Repositories para consultas específicas.

Doctrine

Bibliografía

- https://www.doctrine-project.org/index.html