

Doctrine

CEFS Desenvolupament d'Aplicacions Web



ORM

Un **ORM (Object Relational Mapping)** és una tècnica de programació que converteix dades entre un llenguatge de programació orientat a objectes i una base de dades relacional.

El resultat és una base de dades orientada a objectes que pot usar-se des del llenguatge de programació utilitzat: les classes són taules de la base de dades i els objectes són registres. Així resulta més fàcil crear i manipular taules i dades.



Doctrine

Doctrine és un **conjunt de llibreries** que faciliten la persistència de dades en PHP.

Funciona com una **capa** d'abstracció entre l'aplicació i el SGBD.

Està dissenyada per ser **compatible** amb els SGBD més comuns: MySQL, MSSQL, PostgreSQL, SQLite i Oracle.

Pot donar suport a SGBD **no SQL** (mitjançant un Object Document Model)

Doctrine 1 -> 2006,

Doctrine 2 -> 2010.



Doctrine

Molt utilitzat per d'altres **frameworks** de PHP com Symfony, Zend Framework, Codeigniter ...

Doctrine fa servir les dades com objectes PHP (Entitats) de forma similar al que fa Hibernate en Java.

L'abstracció de les Entitats permet **reutilitzar** codi inclús si canviem el SGBD.



INSTITUT

THOS I CODINA

Doctrine

Un objeto Producto

id: 12
nombre: Bicicleta
precio: \$800.00
descripción: Engranaje fijo, azul...

Doctrine

Tabla: producto

id	nombre	precio	descripción
12	Bicicleta	\$800.00	Engranaje fijo, azul, rápida
13	Camiseta	\$20.99	Negro, adaptable a la mayoría
14	Casco	\$35.00	Mujer, chica, verde con blanco

Patrons

Data Mapper

En Doctrine, l'objecte que implementa aquest patró es denomina EntityManager.

- Realitza les insercions, actualitzacions i esborrat en la BD de les dades de les entitats gestionades.
- Informa (hidrata) els objectes en memòria amb dades obtinguts de la BD.

Patrons

Unity of Work

Aquest patró és l'empleat pel Entity Manager per accedir a la BD de forma transaccional.

Manté l'estat de les entitats gestionades per l'Entity Manager.

Components



ORM.

Maapejador Relacional d'Objectes. Permet l'accés a les taules de les bases de dades a través d'un API orientat a l'objecte.

DBAL

Capa d'Abstracció de Base de Dades. Proveeix d'una interfície comuna d'accés a els diferents SGBD. És similar a PDO, construïda sobre ella, i per tant, dèbilment lligada a aquesta.

COMMON

Utilitats que no són a la SPL, com ara un autoloader de classes, un parser d'anotacions, estructures avançades (p.ex: collections) i un sistema de memòria caché.



INSTITUT

THOS I CODINA

Instal.lació

Instal·lació

Doctrine es pot instal·lar fent servir **PEAR** (per tot el sistema) o com dependència del projecte, mitjançant **Composer**.

Es recomana fer servir Composer, deixant PEAR per les versions antigues.

Per instal·lar Composer

`https://getcomposer.org/download/`

L'arxiu que es descàrrega (composer.phar) pot ser reanomenat i mogut:

```
sudo mv composer.phar /usr/local/bin/composer
```



composer

Composer és un manejador de paquets per a PHP que proporciona un estàndard per a administrar, descarregar i instal·lar dependències i llibreries. Similar a NPM en Node.js i Bundler en Ruby, Composer és la solució ideal quan treballem en projectes complexos que depenen de múltiples fonts d'instal·lació. En lloc d'haver de descarregar cada dependència de manera manual, Composer fa això de manera automàtica per nosaltres.

composer

\$ composer init

Demana informació per generar el *composer.json* a partir del que es realitza la instal·lació.

\$ composer install

Processa el *composer.json*, resol dependències i les instal·la al directori */vendor* (per defecte).

\$ composer update

A partir del *composer.lock* actualitza les dependències del projecte.

Instal·lació

Per instal·lar Doctrine com dependència d'un projecte, ens haurem de situar a la carpeta arrel del projecte, i crear un fitxer ***composer.json*** on inclourem la dependència:

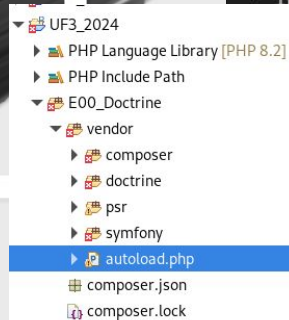
```
{  
    "require": {  
        "doctrine/orm": "*"   
    }  
}
```

La descàrrega i instal·lació es farà amb:

```
composer install
```

Instal·lació

... i si tot va bé



```
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/E00...  
- Installing symfony/polyfill-intl-normalizer (v1.28.0): Extracting archive  
- Installing symfony/polyfill-intl-grapheme (v1.28.0): Extracting archive  
- Installing symfony/polyfill-ctype (v1.28.0): Extracting archive  
- Installing symfony/string (v7.0.3): Extracting archive  
- Installing psr/container (2.0.2): Extracting archive  
- Installing symfony/service-contracts (v3.4.1): Extracting archive  
- Installing symfony/console (v7.0.3): Extracting archive  
- Installing psr/cache (3.0.0): Extracting archive  
- Installing doctrine/event-manager (2.0.0): Extracting archive  
- Installing doctrine/persistence (3.2.0): Extracting archive  
- Installing doctrine/lexer (3.0.0): Extracting archive  
- Installing doctrine/instantiator (2.0.0): Extracting archive  
- Installing doctrine/inflector (2.0.9): Extracting archive  
- Installing doctrine/deprecations (1.1.3): Extracting archive  
- Installing psr/log (3.0.0): Extracting archive  
- Installing doctrine/dbal (4.0.0): Extracting archive  
- Installing doctrine/collections (2.1.4): Extracting archive  
- Installing doctrine/orm (3.0.0): Extracting archive  
Package suggestions were added by new dependencies, use `composer suggest` to  
tails.  
ting autoload files  
ackages you are using are looking for funding.  
e `composer fund` command to find out more!  
ar@taguilar:~/Documentos/workspace/php/UF3_2024/E00_Doctrine$
```




INSTITUT

THOSI CODINA

El primer projecte

```
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine
taguilar@taguilar:~$ cd Documentos/workspace/php/UF3_2024/doctrine/
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$ composer init
Cannot load Xdebug - it was already loaded

Welcome to the Composer config generator

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

Package name (<vendor>/<name>) [taguilar/doctrine]:
Description []: El meu primer projecte
Author [n to skip]: Toni
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []:

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? y
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
[6] doctrine/cache
[7] doctrine/annotations
[8] gedmo/doctrine-extensions
[9] doctrine/doctrine-migrations-bundle
[10] doctrine/doctrine-bundle
[11] symfony/doctrine-bridge
[12] symfony/property-info
[13] doctrine/common
[14] doctrine/sql-formatter

Enter package # to add, or the complete package name if it is not listed: 0
Enter the version constraint to require (or leave blank to use the latest version): 2.4
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively [yes]? n
Add PSR-4 autoload mapping? Maps namespace "Taguilar\Doctrine" to the entered relative path. [src/, n to skip]: src/
```



INSTITUT

THOS I CODINA

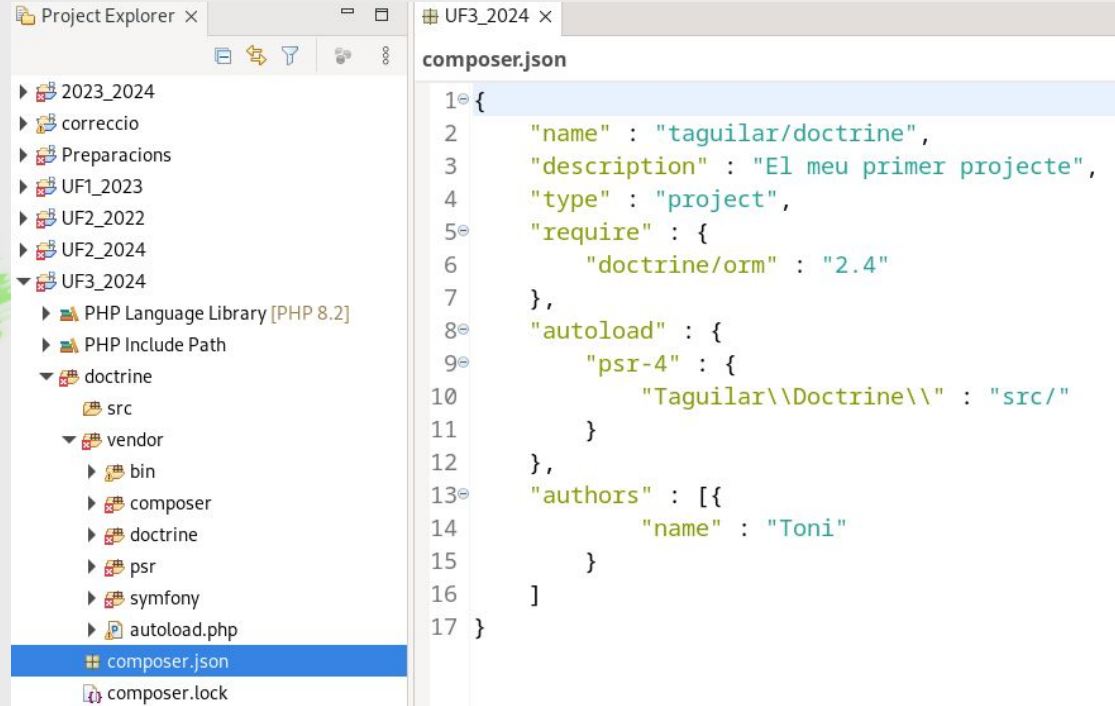
El primer projecte

```
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine
{
    "name": "Toni"
}
]
}

Do you confirm generation [yes]? y
Would you like to install dependencies now [yes]? y
Loading composer repositories with package information
Updating dependencies
Lock file operations: 10 installs, 0 updates, 0 removals
- Locking doctrine/cache (2.2.0)
- Locking doctrine/collections (1.8.0)
- Locking doctrine/dbal (2.13.9)
- Locking doctrine/deprecations (1.1.3)
- Locking doctrine/event-manager (1.2.0)
- Locking doctrine/orm (v2.4.0)
- Locking psr/log (1.1.4)
- Locking symfony/console (v2.8.52)
- Locking symfony/debug (v3.0.9)
- Locking symfony/polyfill-mbstring (v1.28.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 10 installs, 0 updates, 0 removals
- Downloading doctrine/cache (2.2.0)
- Downloading doctrine/event-manager (1.2.0)
- Downloading psr/log (1.1.4)
- Downloading symfony/debug (v3.0.9)
- Downloading symfony/console (v2.8.52)
- Downloading doctrine/dbal (2.13.9)
- Downloading doctrine/collections (1.8.0)
- Downloading doctrine/orm (v2.4.0)
- Installing doctrine/cache (2.2.0): Extracting archive
- Installing doctrine/deprecations (1.1.3): Extracting archive
- Installing doctrine/event-manager (1.2.0): Extracting archive
- Installing symfony/polyfill-mbstring (v1.28.0): Extracting archive
- Installing psr/log (1.1.4): Extracting archive
- Installing symfony/debug (v3.0.9): Extracting archive
- Installing symfony/console (v2.8.52): Extracting archive
- Installing doctrine/dbal (2.13.9): Extracting archive
- Installing doctrine/collections (1.8.0): Extracting archive
- Installing doctrine/orm (v2.4.0): Extracting archive
4 package suggestions were added by new dependencies, use 'composer suggest' to see details.
Package symfony/debug is abandoned, you should avoid using it. Use symfony/error-handler instead.
Generating autoload files
4 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
Found 1 security vulnerability advisory affecting 1 package.
Run "composer audit" for a full list of advisories.
PSR-4 autoloading configured. Use "namespace Taguilar\Doctrine;" in src/
Include the Composer autoloader with: require 'vendor/autoload.php';
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$
```

El primer projecte

... i si tot va bé



The screenshot displays an IDE interface with two main panels. The left panel, titled 'Project Explorer', shows a project structure for 'UF3_2024'. The right panel, titled 'UF3_2024 x', shows the contents of the 'composer.json' file.

Project Explorer Structure:

- 2023_2024
- correccio
- Preparacions
- UF1_2023
- UF2_2022
- UF2_2024
- UF3_2024
 - PHP Language Library [PHP 8.2]
 - PHP Include Path
 - doctrine
 - src
 - vendor
 - bin
 - composer
 - doctrine
 - psr
 - symfony
 - autoload.php
 - composer.json
 - composer.lock

composer.json Content:

```
1 {  
2     "name" : "taguilar/doctrine",  
3     "description" : "El meu primer projecte",  
4     "type" : "project",  
5     "require" : {  
6         "doctrine/orm" : "2.4"  
7     },  
8     "autoload" : {  
9         "psr-4" : {  
10            "Taguilar\\Doctrine\\" : "src/"  
11        }  
12    },  
13    "authors" : [{  
14        "name" : "Toni"  
15    }  
16 ]  
17 }
```

El primer projecte

creem una carpeta config i posarem un fitxer de configuració *config.php*

```
<?php
// Configuració de l'aplicació
// Accés a la base de dades
$dbParams = [
    'driver' => 'pdo_mysql',
    'host' => '127.0.0.1',
    'dbname' => 'test',
    'user' => 'usr_generic',
    'password' => '2024@Thos'
];
// Estem en mode desenvolupament?
$dev = true;
```

El primer projecte

Doctrine no és capaç de crear la base de dades i les credencials d'usuari proporcionades (tampoc seria recomanable) per tant abans de fer el procés ho haurem de tenir preparat.

El primer projecte

dins la carpeta config posarem un fitxer
cli-config.php

```
<?php
// Configuració del CLI de Doctrine.
// Dependència de l'objecte ConsoleRunner
use Doctrine\ORM\Tools\Console\ConsoleRunner;

// Includem el bootstrap per obtenir l' "Entity Manager"
require_once __DIR__.'../../src/bootstrap.php';

// Retornem l'objecte HelperSet de consola
return ConsoleRunner::createHelperSet($entityManager);
```


El primer projecte

a la carpeta src posarem un fitxer de
bootstrap.php

```
<?php
use Doctrine\ORM\Tools\Setup;
use Doctrine\ORM\EntityManager;

require_once __DIR__.'../../vendor/autoload.php';
require_once __DIR__.'../../config/config.php';

$entitiesPath = array(__DIR__.'./cinema/Entity');

$config = Setup::createAnnotationMetadataConfiguration
    ($entitiesPath,$dev);
$entityManager = EntityManager::create($dbParams, $config);
```

El primer projecte

Si tot ha anat correctament, ja podem invocar doctrine des de la consola a la carpeta arrel:

```
php vendor/doctrine/orm/bin/doctrine.php
```

si surten errors, s'hauran d'arreglar

```
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$ php vendor/doctrine/orm/bin/doctrine.php
Cannot load Xdebug - it was already loaded
PHP Fatal error:  Uncaught Error: Class "Doctrine\Common\Cache\ArrayCache" not found in /home/taguilar/Documentos/workspace/php/UF3_2024/doctrine/vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Setup.php:144
Stack trace:
#0 /home/taguilar/Documentos/workspace/php/UF3_2024/doctrine/vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Setup.php(70): Doctrine\ORM\Tools\Setup::createConfiguration()
#1 /home/taguilar/Documentos/workspace/php/UF3_2024/doctrine/src/bootstrap.php(10): Doctrine\ORM\Tools\Setup::createAnnotationMetadataConfiguration()
#2 /home/taguilar/Documentos/workspace/php/UF3_2024/doctrine/config/cli-config.php(8): require_once('...')
#3 /home/taguilar/Documentos/workspace/php/UF3_2024/doctrine/vendor/doctrine/orm/bin/doctrine.php(48): require('...')
#4 {main}
  thrown in /home/taguilar/Documentos/workspace/php/UF3_2024/doctrine/vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Setup.php on line 144
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$
```

El primer projecte

Hem instal·lat doctrine/cache en la v. 2.

Segons github:

`This library is deprecated and will no longer receive bug fixes from the Doctrine Project. Please use a different cache library, preferably PSR-6 or PSR-16 instead.`

Per tant tenim dues opcions:

- incloure doctrine/cache: ^1.11
- fer servir altres llibreries com symfony/cache modificant el composer.json i actualitzant.

El primer projecte

Hem instal·lat doctrine/cache en la v. 2.

Segons github:

`This library is deprecated and will no longer receive bug fixes from the Doctrine Project. Please use a different cache library, preferably PSR-6 or PSR-16 instead.`

Per tant tenim dues opcions:

- incloure doctrine/cache: ^1.11
- fer servir altres llibreries com symfony/cache modificant el composer.json i actualitzant.
- posem una versió de doctrine més nova, p.e. 2.8

El primer projecte

i tindrem ...

```

taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine
Use the 'composer fund' command to find out more!
No security vulnerability advisories found.
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$ php vendor/doctrine/orm/bin/doctrine.php
Cannot load Xdebug - it was already loaded
Doctrine Command Line Interface 2.8.0@418587bc25c107f16b53ae67a6ef8589408aff6c

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display help for the given command. When no command is given display help for the list command
  -q, --quiet           Do not output any message
  -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:import      Import SQL file(s) directly to Database.
  dbal:reserved-words Checks if the current database contains identifiers that are reserved.
  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:convert-dl-schema [orm:convert:dl-schema] Converts Doctrine 1.x schema into a Doctrine 2.x schema
  orm:convert-mapping [orm:convert:mapping] Convert mapping information between supported formats
  orm:ensure-production-settings Verify that Doctrine is properly configured for a production environment
  orm:generate-entities [orm:generate:entities] Generate entity classes and method stubs from your mapping
  information
  orm:generate-proxies [orm:generate:proxies] Generates proxy classes for entity classes
  orm:generate-repositories [orm:generate:repositories] Generate repository classes from your mapping information
  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
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$

```


El primer projecte

El nostre projecte comptarà amb varies entitats relacionades (Pel·lícula, Comentari, Etiqueta) de forma que una pel·lícula pugui tenir comentaris i/o etiquetes, i que cada etiqueta pot aparèixer en una o varies pel·lícules.

Film
id titol titolOriginal director any

Coment
id text data

Tag
nom



Entity

Les Entitats de Doctrine estan definides com classes de PHP, amb algunes característiques:

- No poden ser final ni contenir mètodes final.
- Les propietats/atributs persistents han de ser `private` o `protected`.
- No poden implementar `__clone()` o `__wakeup()`.
- No es pot fer servir la funció `func_get_args()`.
- Les propietats/atributs només són accessibles des de la pròpia entitat o mitjançant getters i setters.



Entity

En crear una entitat, Doctrine **no invoca al constructor** de la classe, aquest només és invocat si creem una instància amb new. Això permet l'ús del constructor per a la inicialització d'estructures, establir valors per defecte o requerir paràmetres.

A Doctrine, una entitat és un objecte amb identitat. S'utilitza el patró **IdentityMap** per fer un seguiment de les entitats i els seus ids, de tal manera que la instància de l'entitat amb un determinat id sigui única, sense importar les variables que apuntin a ella, o el tipus de consulta usat.

Aquest patró de seguiment facilita el manteniment dels estats de l'entitat al llarg del seu cicle de vida.



Entity

NEW

Estat de la instància de l'entitat, que no té una identitat persistent i no està associada a un Entity Manager (p.ex: creada amb l'operador new).

MANAGED

Estat de la instància de l'entitat, amb entitat persistent, que està associada a un Entity Manager, i per tant, gestionada per ell.



Entity

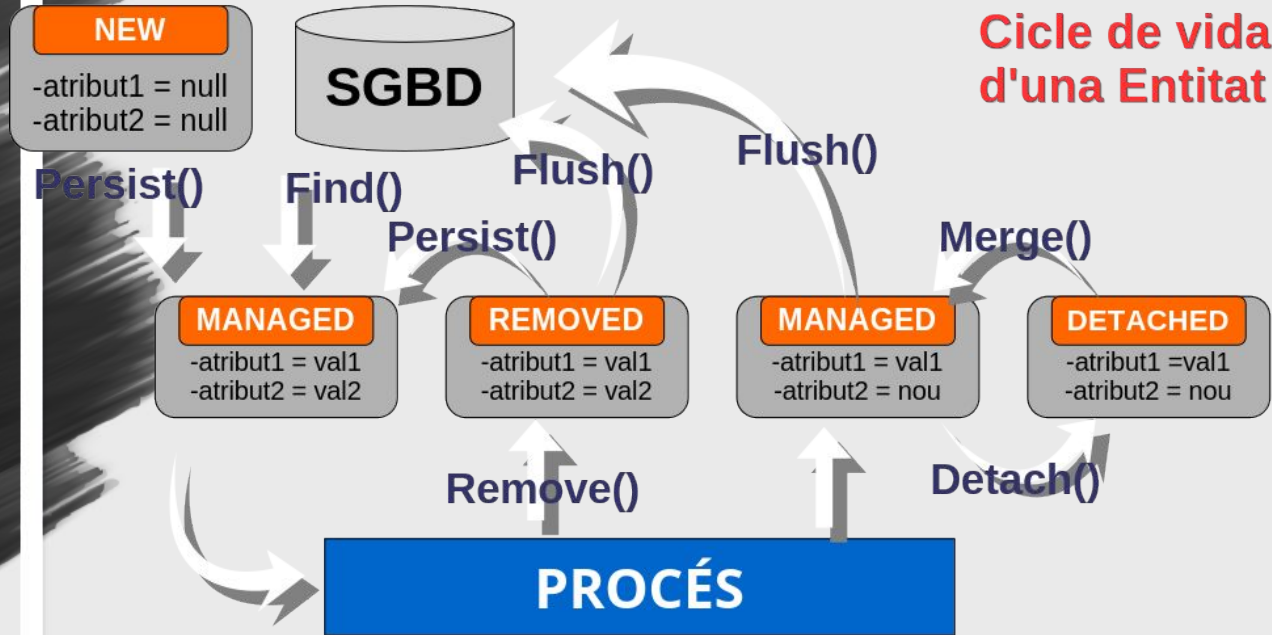
DETACHED

Estat de la instància de l'entitat, amb entitat persistent, però que NO està associada a un Entity Manager, i per tant, NO gestionada per elll.

REMOVED

Estat de la instància de l'entitat, amb entitat persistent, que està associada a un Entity Manager, que serà eliminada de la BD en la següent transacció.

Entity





Entity

L'estat persistent d'una entitat està representat per les seves variables d'instància. És a dir, com a objecte PHP, el seu estat persistent estarà definit pel valor dels seus camps / propietats.

Aquests camps o propietats poden ser de diferents tipus. Encara que Doctrine suporta un ampli conjunt de tipus de dades, això no vol dir que sigui el mateix que el dels tipus nadius de PHP, o els del SGBD utilitzat.

Mitjançant el mapejat de dades, informarem a Doctrine quins camps definiran l'estat de l'entitat i el tipus de dades de cada un.

El mapejat de dades, al costat del d'associacions, generen unes metadades que serveixen a l'ORM per gestionar les entitats i les seves relacions.

Entity

Anotacions

Les metadades són incrustats dins de blocs de comentaris, anàlegs als utilitzats per eines com PHPDocumentor.

És possible definir nous blocs d'anotacions.

```
use Doctrine\ORM\Mapping as ORM;
/**
 * @ORM\Entity
 * @ORM\Table(name="ejemplo")
 */
class Example {
    /**
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     * @ORM\Column(type="integer")
     */
    private $id;
    /**
     * @ORM\Column(type="string")
     */
    private $name;

    public function getName() {
        return $this->name;
    }

    public function setName($name) {
        $this->name = $name;
    }
}
??>
```

Entity

XML

Utilitzar fitxers XML amb un esquema propi per al mapejat de dades amb Doctrine.

Cada entitat ha de tenir el seu propi fitxer descriptor, el nom serà el Full Qualified Name de l'entitat.

```
<?xml version="1.0" encoding="UTF-8"?>
<doctrine-mapping xmlns="http://doctrine-project.org/schemas/orm/doctrine-mapping" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://doctrine-project.org/schemas/orm/doctrine-mapping https://www.doctrine-project.org/schemas/orm/doctrine-mapping.xsd">
  <entity name="Doctrine\Tests\ORM\Mapping\User" table="cms_users">
    <indexes>
      <index name="name_idx" columns="name" />
      <index columns="user_email" />
    </indexes>
    <unique-constraints>
      <unique-constraint columns="name,user_email" name="search_idx" />
    </unique-constraints>
    <lifecycle-callbacks>
      <lifecycle-callback type="prePersist" method="doStuffOnPrePersist" />
      <lifecycle-callback type="prePersist" method="doOtherStuffOnPrePersistToo" />
      <lifecycle-callback type="postPersist" method="doStuffOnPostPersist" />
    </lifecycle-callbacks>
    <id name="id" type="integer" column="id">
      <generator strategy="AUTO" />
      <sequence-generator sequence-name="tablename_seq" allocation-size="100" initial-value="1" />
    </id>
    <field name="name" column="name" type="string" length="50" nullable="true" unique="true" />
    <field name="email" column="user_email" type="string" column-definition="CHAR(32) NOT NULL" />
    <one-to-one field="address" target-entity="Address" inverted-by="user">
      <cascade><cascade-remove /></cascade>
      <join-column name="address_id" referenced-column-name="id" on-delete="CASCADE" on-update="CASCADE" />
    </one-to-one>
    <one-to-many field="phonenumbers" target-entity="Phonenumber" mapped-by="user">
      <cascade><cascade-persist /></cascade>
      <order-by><order-by-field name="number" direction="ASC" /></order-by>
    </one-to-many>
    <many-to-many field="groups" target-entity="Group">
      <cascade><cascade-all /></cascade>
      <join-table name="cms_users_groups">
```

Entity

Yaml

El document de mapatge YAML d'una classe es carrega sota demanda la primera vegada que se sol·licita i posteriorment s'emmagatzema a la memòria cau de metadades.

```
# Doctrine\Tests\ORM\Mapping\User.dcm.yml
Doctrine\Tests\ORM\Mapping\User:
  type: entity
  repositoryClass: Doctrine\Tests\ORM\Mapping\UserRepository
  table: cms_users
  schema: schema_name # The schema the table lies in, for platforms that support schemas (Optional, >= 2.5)
  readOnly: true
  indexes:
    name_index:
      columns: [ name ]
  id:
    id:
      type: integer
      generator:
        strategy: AUTO
  fields:
    name:
      type: string
      length: 50
    email:
      type: string
      length: 32
      column: user_email
      unique: true
      options:
        fixed: true
        comment: User's email address
  loginCount:
    type: integer
    column: login_count
    nullable: false
    options:
      unsigned: true
      default: 0
  oneToOne:
    address:
```

Entity

PHP

Doctrine ORM també permet proporcionar les metadades ORM en forma de codi PHP pla utilitzant l'API ClassMetadata. Podeu escriure el codi en fitxers PHP o dins d'una funció estàtica anomenada loadMetadata(\$class) a la pròpia classe d'entitat.

```
<?php
// /path/to/php/mapping/files/Entities/User.php
$metadata->mapField(array(
    'id' => true, 'fieldName' => 'id', 'type' => 'integer' ));

$metadata->mapField(array(
    'fieldName' => 'username',
    'type' => 'string',
    'options' => array(
        'fixed' => true,
        'comment' => "User's login name"
    )
));

$metadata->mapField(array(
    'fieldName' => 'login_count',
    'type' => 'integer',
    'nullable' => false,
    'options' => array(
        'unsigned' => true,
        'default' => 0
    )
));
```

Entity

PHP vs Mysql

DOCTRINE	PHP	SQL
string	string	VARCHAR
integer	integer	INT
smallint	integer	SMALLINT
bigint	string	BIGINT
boolean	boolean	BOOLEAN
decimal	double	DECIMAL
date	DateTime	DATETIME
time	DateTime	TIME
datetime	DateTime	DATETIME

DOCTRINE	PHP	SQL
datetimez	DateTime	DATETIME
text	string	CLOB
object	Object (1)	CLOB
array	Array (1)	CLOB
float	Double (4)	FLOAT
simple array	Array (2)	CLOB
json array	object	CLOB
guid	string (3)	UUID/GUID
blob	resource	BLOB

- 1.- Fent servir serialize() i unserialize().
- 2.- Fent servir implode() i explode().
- 3.- Si el SGBD no té tipus específic fa servir VARCHAR.
- 4.- Només si locale fa servir separador decimal.

El primer projecte

En el nostre projecte, treballarem amb anotacions.

Les anotacions no estan escrites en PHP, s'analitzen mitjançant un lector d'anotacions des d'una aplicació PHP, d'aquesta forma afecta al funcionament d'una aplicació. És per això que es considera un DSL (Domain-specific language) o llenguatge específic del domini ja que permet convertir conceptes complexos en instàncies senzilles.

Les anotacions es formen dins de doc blocks, que són comentaris que comencen amb `/**` i s'utilitzen com a documentació del codi al qual precedeix (classes, propietats, funcions...).

El primer projecte

La implementació de les anotacions en Doctrine estan situades en el namespace Doctrine\Common\Annotations i formen part del paquet Common. Nosaltres podem definir les nostres pròpies anotacions igual que Doctrine 2 defineix les anotacions que s'utilitzen per a mapejar la informació objecte-relacional.

```
<?php
/**
 * @Entity
 * @Table(name="my_persistent_class")
 */
class MyPersistentClass {
    //...
}
```

El primer projecte

Per a mapejar propietats a una columna d'una taula en una base de dades relacional n'hi ha prou amb utilitzar l'anotació **@Column** i especificar el tipus de mapatge entre elles. Si no especifiquem un tipus de mapatge, es pren el tipus de mapatge string per defecte.

```
<?php
/** @Entity */
class MyPersistentClass {

    /** @Column(type="integer") */
    private $id;

    /** @Column(length=50) */
    private $name; // type defaults to string
    //...

}
```

El primer projecte

L'anotació @Column té alguns altres atributs com:

- **type:** opcional, per defecte *string*, estableix el tipus de mapatge a usar per a la columna.
- **name:** opcional, per defecte el nom de la propietat, estableix el nom de la columna en la base de dades.
- **length:** opcional, per defecte 255, la longitud de la columna en la base de dades, (només per string).
- **unique:** opcional, per defecte FALSE, estableix si la columna és una clau única.
- **nullable:** opcional, per defecte FALSE, estableix si la columna pot ser nul·la.
- **precision:** opcional, per defecte 0, estableix la precisió per a una columna decimal, només si és un camp decimal.
- **scale:** opcional, per defecte 0, estableix l'escala per a una columna decimal, només si és un camp decimal.

El primer projecte

Cada classe entitat necessita una clau primària que identifiqui unívocament a l'entitat. Podem designar la propietat que s'usa com a identificador amb l'anotació **@Id**.

Si no fem res més, s'assumeix que l'identificador s'assignarà de manera manual. Si volem que sigui un valor generat automàticament podem utilitzar l'anotació **@GeneratedValue**.

Es pot definir l'estratègia (per defecte AUTO) però també SEQUENCE (Oracle, PostgreSQL), IDENTITY (Auto_increment en MySQL o SERIAL en PostgreSQL), NONE (identificador manual).

El primer projecte

Definirem Film:

```
<?php
namespace cinema\Entity;
/**
 * Pelicula
 *
 * @Entity
 * @Table( name="movie", indexes={ @Index(name="year_idx", columns="year") }
 *
 */
class Film {
    /**
     * @var int
     *
     * @Id
     * @GeneratedValue
     * @Column(type="integer")
     */
    private $id;
    /**
     * @var string
     *
     * @Column(type="string", length=100, name="spanish_title")
     */
    private $titulo;
    /**
     * @var string
     *
     * @Column(type="string", length=100, name="original_title",
     nullable=false)
     */
    private $tituloOriginal;
    /**
     * @var string
     *
     * @Column(type="string", length=100)
     */
    private $director;
    /**
     * @var int
     *
     * @Column(type="integer", name="year", nullable=false, unique=false,
     options={"unsigned":true, "default":0})
     */
    private $anyo;
```

```
/**
 * Get id
 * @return integer
 */
public function getId() {
    return $this->id;
}

/**
 * Set titulo
 * @param string $titulo
 * @return Film
 */
public function setTitulo($titulo) {
    $this-> titulo = $titulo;
    return $this;
}

/**
 * Get titulo
 * @return string
 */
public function getTitulo() {
    return $this->titulo;
}

/**
 * Set tituloOriginal
 * @param string $tituloOriginal
 * @return Film
 */
public function setTituloOriginal($tituloOriginal) {
    $this-> tituloOriginal = $tituloOriginal;
    return $this;
}

/**
 * Get tituloOriginal
 * @return string
 */
public function getTituloOriginal() {
    return $this->tituloOriginal;
}

/**
 * Set director
 * @param string $director
 * @return Film
 */
public function setDirector($director) {
    $this-> director = $director;
    return $this;
}

/**
 * Get director
 * @return string
 */
```



INSTITUT

THOS I CODINA

El primer projecte

Definirem Coment

```
namespace cinema\Entity;

/**
 * Coment
 *
 * @Entity
 * @Table( name="comment" )
 *
 */
class Coment {

    /**
     * @var int
     * @Id
     * @GeneratedValue
     * @Column(type="integer")
     */
    private $id;

    /**
     * @var string
     * @Column(type="string", length=100, name="texto")
     */
    private $texto;

    /**
     * @var date
     * @Column(type="date", name="date", nullable=false,
     unique=false )
     */
    private $fecha;
```

```
/**
 * Get id
 * @return integer
 */
public function getId() {
    return $this->id;
}

/**
 * Set texto
 * @param string $texto
 * @return Coment
 */
public function setTexto($texto) {
    $this->texto = $texto;
    return $this;
}

/**
 * Get texto
 * @return string
 */
public function getTexto() {
    return $this->texto;
}

/**
 * Set fecha
 * @param \DateTime $fecha
 * @return Coment
 */
public function setFecha($fecha) {
    $this->fecha = $fecha;
    return $this;
}

/**
 * Get fecha
 * @return \DateTime
 */
public function getFecha() {
    return $this->fecha;
}
}
```




INSTITUT

THOS I CODINA

El primer proyecto

i Tag

```
<?php
namespace cinema\Entity;
/**
 * Tag
 *
 * @Entity
 * @Table( name="tag" )
 *
 */
class Tag {
    /**
     * @var string
     * @Id
     * @Column(type="string", name="name", nullable=false, unique=true)
     */
    private $texto;

    /**
     * Set texto
     * @param string $texto
     * @return Tag
     */
    public function setTexto($texto) {
        $this->texto = $texto;
        return $this;
    }

    /**
     * Get texto
     * @return string
     */
    public function getTexto() {
        return $this->texto;
    }
}
```

El primer projecte

Un cop creades les classes de les entitats, Doctrine permet crear de automàticament els getters i els setters per a cadascuna de les classes mitjançant línia d'ordres. Des del directori arrel:

```
php vendor/doctrine/orm/bin/doctrine.php orm:generate:entities src/
```

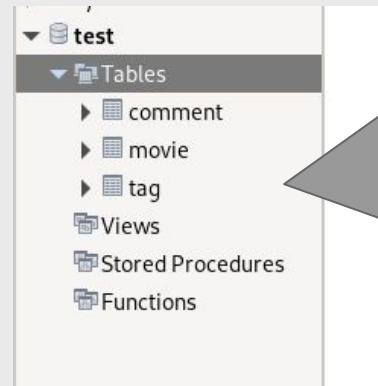
i Doctrine és capaç de crear de forma automàtica, l'esquema de DB que correspon a aquestes entitats:

```
php vendor/doctrine/orm/bin/doctrine.php orm:schema-tool:create
```

Això hauria d'haver creat l'estructura de taules a la BD, amb els seus camps definits tal com especifiquem en la informació definida en les anotacions.

El primer projecte

```
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/do...  
taguilar@taguilar:~$ cd Documentos/workspace/php/UF3_2024/doctrine/  
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$ php vendor/doctr  
ine/orm/bin/doctrine.php orm:schema-tool:create  
Cannot load Xdebug - it was already loaded  
  
! [CAUTION] This operation should not be executed in a production environment!  
!  
  
Creating database schema...  
  
[OK] Database schema created successfully!  
  
taguilar@taguilar:~/Documentos/workspace/php/UF3_2024/doctrine$
```



Relacions

UNIDIRECCIONAL:

Les entitats relacionades poden ser obtingudes des de les entitats principals. Només tenen banda propietari (owner).

BIDIRECCIONAL:

Les entitats relacionades poden ser obtingudes des de les principals, i al seu torn, les principals poden ser obtingudes des de les relacionades. Tenen una banda propietaria (**owner**) i una banda inversa (**inverse**).

Relacions

CARDINALITAT:

- **1:1 (Un a un) @OneToOne:**

Cada entitat principal només pot tenir una associada.

- **1:N (Un a molts) @OneToMany:**

Cada entitat principal pot tenir varies associades

- **N:1 (Molts a un) @ManyToOne:**

Varies entitats tenen una mateixa entitat associada.
Només és disponible per associacions bidireccionals com inversa a 1:N.

- **N:N (Molts a molts) @ManyToMany:**

Varies entitats tenen associades un conjunt d'entitats.

Relacions

HIDRATACIÓ:

En Doctrine, la hidratació (hydration) és el nom que rep el procés d'obtenir un resultat final d'una consulta a la BD i mapejar-ho a un ResultSet.

Els tipus de resultats que retorna el procés poden ser:

- Entitats
- Arrays estructurats
- Arrays escalars
- Variables simples

En general, la hidratació és un procés que consumeix molts recursos, de manera que recuperar només les dades que anem a necessitar suposarà una millora del rendiment i / o consum d'aquests recursos.

Relacions

HIDRATACIÓ:

En Doctrine, la hidratació (hydration) és el nom que rep el procés d'obtenir un resultat final d'una consulta a la BD i mapejar-ho a un ResultSet.

Els tipus de resultats que retorna el procés poden ser:

- Entitats
- Arrays estructurats
- Arrays escalars
- Variables simples

En general, la hidratació és un procés que consumeix molts recursos, de manera que recuperar només les dades que anem a necessitar suposarà una millora del rendiment i / o consum d'aquests recursos.

Relacions

Doctrine només gestiona la part propietària (owner) d'una associació. Això vol dir que sempre haurem d'identificar aquest costat de la mateixa.

En una associació bidireccional, la part propietària sempre tindrà un atribut `inversedBy`, i la part inversa tindrà l'atribut `mappedBy`.

Per defecte, les associacions `@OneToOne` i `@ManyToOne` són persistides en SQL utilitzant una columna amb l'id i una clau forània. Les associacions `@ManyToMany` utilitzen una taula intermitja.

Els noms d'aquestes taules i columnes són generats de forma automàtica per Doctrine, però es poden canviar fent servir les anotacions `@JoinColumn` i `@JoinTable`.

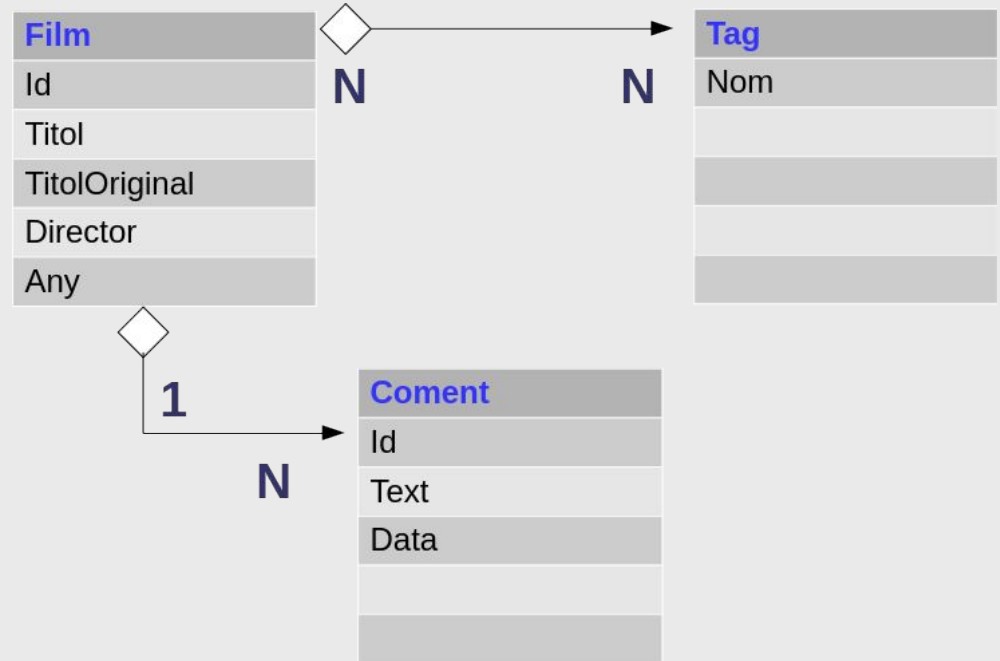


Relacions

Una pel·lícula pot tenir comentaris i/o etiquetes, i que cada etiqueta pot aparèixer en una o varies pel·lícules.

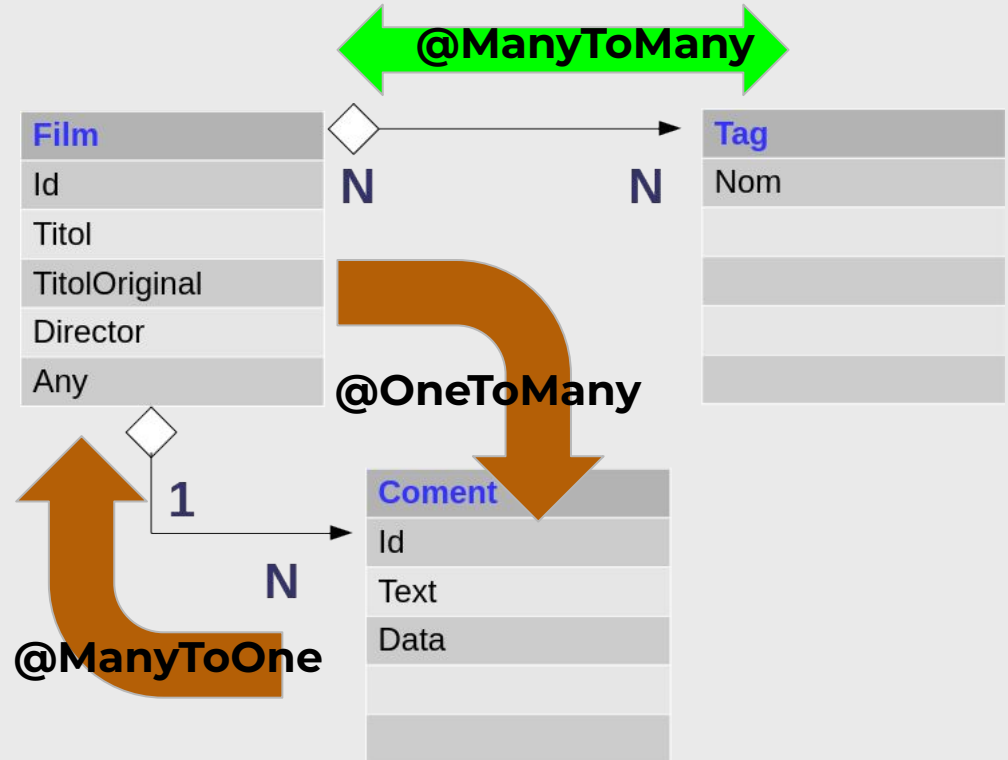
Relacions

Una pel·lícula pot tenir comentaris i/o etiquetes, i que cada etiqueta pot aparèixer en una o varies pel·lícules.



Relacions

Una pel·lícula pot tenir comentaris i/o etiquetes, i que cada etiqueta pot aparèixer en una o varies pel·lícules.



El primer projecte

Definirem les dues relacions de forma bidireccional.

Comencem per la relació Film-Coment....

A Coment afegirem \$pelicula: Això generarà un nou camp en la taula, amb un identificador de la pel·lícula a la que correspon el comentari. Per defecte, pelicula_id.

```
/**  
 * @var Film  
 * @ManyToOne(targetEntity="Film",  
   invertedBy="comentaris")  
 */  
private $pelicula;
```


El primer projecte

Definirem les dues relacions de forma bidireccional.

Comencem per la relació Film-Coment....

A Film definim comentaris que serà un Array de Coments.

```
/**  
 * @var Coment[]  
 * @OneToMany(targetEntity="Coment", mappedBy="pelicula")  
 */  
private $comentaris;
```

El primer projecte

Com és un array, hem dinicialitzar-ho al constructor, i haurem de crear un mètode per afegir comentaris.

```
/**
 * Inicialitzem coleccions
 */
public function __construct(){
    $this->comentarios = new ArrayCollection();
}

/**
 * Add Coment
 * @param Coment $comentari
 * @return Film
 */
public function addComentari(Coment $comentari){
    $this->comentarios[] = $comentari;
    $comentarios->setPelícula($this);
    return $this;
}
```

El primer projecte

La relació bidireccional entre Film-Tag serà:

A Film afegim l'atribut

```
/**
 * @var Tag[]
 * @ManyToMany(targetEntity="Tag", inversedBy="pelicules",
 *      fetch="EAGER", cascade={"persist"}, orphanRemoval=true
 * )
 * @JoinTable(
 *      name="movie_tag",
 *      inverseJoinColumns={
 *          @JoinColumn(name="tag_name",
 *              referencedColumnName="name") }
 * )
 */
private $etiquetes;
```

El primer projecte

modificarem el constructor.

```
/**  
 * Inicialitzem coleccions  
 */  
public function __construct() {  
    $this->comentaris = new ArrayCollection();  
    $this->etiquetes = new ArrayCollection();  
}
```

El primer projecte

i a Tag

```
/**
 * @var Film[]
 * @ManyToMany(targetEntity="Film", mappedBy="etiquetes" )
 */
private $pelicules;

[...]

/**
 * Add pelicules
 * @param Film $pelicula
 * @return Tag
 */
public function addPelicula(Film $pelicula) {
    $this->pelicules[] = $pelicula;
    $pelicula->addEtiqueta($this);
    return $this;
}

// Cast de l'objete com string
public function __toString() {
    return $this->getTexto();
}
```

El primer projecte

i podrem esborrar manualment les taules crades anteriorment i llençar el

```
php vendor/doctrine/orm/bin/doctrine.php orm:schema-tool:create
```

o fer-ho amb

```
php vendor/doctrine/orm/bin/doctrine.php orm:schema-tool:drop --force
```

```
php vendor/doctrine/orm/bin/doctrine.php orm:schema-tool:create
```

o amb

```
php vendor/doctrine/orm/bin/doctrine.php orm:schema-tool:update
```


Consultes

**DOCTRINE
QUERY
LANGUAGE**

DQL Llenguatge de consulta específic del domini Doctrine.

**QUERY
BUILDER**

Helper Class per la construcció de consultes mitjançant un API.

SQL Nadiu

Us de l'SQL propi del SGBD mitjançant NativeQuery o Query

DQL

```
use Doctrine\Common\Cache\Psr6\DoctrineProvider;
$users = DoctrineProvider::getTable('User')->findAll();
foreach($users as $user) {
    echo $user->username . " has phonenumbers: ";

    foreach($user->Phonenumbers as $phonenummer) {
        echo $phonenummer->phonenummer . "\n";
    }
}
```

Aquesta forma no és recomanable donat que usa diferents sentències SQL per carregar objectes. Seria equivalent a....

```
$q = Doctrine_Query::create()
    ->from('User u')
    ->leftJoin('u.Phonenumbers p');
echo $q->getSqlQuery();
```

DQL

Això genera una sentència SQL com

```
$q = Doctrine_Query::create()
    ->from('User u')
    ->leftJoin('u.Phonenumber p');
$users = $q->execute();
```

```
foreach($users as $user) {
    echo $user->username . " has phonenumber: ";

    foreach($user->Phonenumber as $phonenumber) {
        echo $phonenumber->phonenumber . "\n";
    }
}
```

```
SELECT
u.id AS u_id,
u.is_active AS u_is_active,
u.is_super_admin AS u_is_super_admin,
u.first_name AS u_first_name,
u.last_name AS u_last_name,
u.username AS u_username,
u.password AS u_password,
u.type AS u_type,
u.created_at AS u_created_at,
u.updated_at AS u_updated_at,
p.id AS p_id,
p.user_id AS p_user_id,
p.phonenumber AS p_phonenumber
FROM user u
LEFT JOIN phonenumber p
ON u.id = p.user_id
```

DQL

A FAVOR

És similar a SQL, però té característiques pròpies (inspirat en HQL).

Gestiona objectes i propietats en lloc de taules i camps.

Permet simplificar algunes construccions de les consultes gràcies a l'ús de metadades (p.ex: Clàusules ON en els JOINS).

Es Independent del SGBD utilitzat.

EN CONTRA

Hi ha diferències de sintaxi respecte a l'SQL estàndard.

No posseeix tota la funcionalitat i optimitzacions del SQL específic de cada SGBD.

Té limitacions a l'hora d'implementar certes consultes (P.ex: subconsultes).

Query Builder

És una classe creada per ajudar a construir consultes SQL mitjançant l'ús d'una interfície, a través d'un API.

El QueryBuilder es crea mitjançant el mètode `createQueryBuilder()` heretat del repositori base de l'entitat o des del EntityManager.

En la creació d'una instància de QueryBuilder des del repositori de l'entitat hem d'especificar un paràmetre (string), que és l'àlies de l'entitat principal.

```
$qb=$entityRepo->createQueryBuilder('u');
```

Que equival a la creació des de l'EntityManager:

```
$qb = $entityManager->createQueryBuilder();
```

```
$qb->select('u');
```

Query Builder

Es pot obtenir la consulta DQL generada pel QueryBuilder mitjançant el mètode getDQL().

```
$qb = $entityManager->createQueryBuilder();  
  
$qb->select('p')->from('Cine\Entity\Pelicula','p');  
  
$queryDQL = $qb->getDQL();
```

D'una forma similar, abans de l'obtenció de l'objecte Query associat al QueryBuilder, podem accedir a l'SQL resultant mitjançant el mètode getSQL().

```
query = $qb->getQuery();  
  
$querySQL = $query->getSQL();
```


Query Builder

Les consultes SQL fan ús intern dels **Prepared Statements** de SQL, per motius de seguretat (p.ex: SQL injections) i rendiment (unitat transaccional).

Per defecte, i llevat que s'especifiqui una altra cosa, una consulta DQL obté totes les entitats relacionades amb la principal. Això pot provocar problemes de recursos o de rendiment si no es gestiona bé.

La naturalesa de les relacions entre entitats és coneguda per Doctrine gràcies a les metadades de les seves associacions, per això no cal especificar clàusules ON o USING en els JOIN.

Les classes QueryBuilder i Query gestionen un cau de les consultes. El comportament i naturalesa d'aquest cau és diferent segons estiguem en mode desenvolupament o producció..

SQL nadiu

Native Query

Els resultats es mapejen a entitats Doctrine mitjançant ResultSetMapBuilder

Es fa servir Doctrine ORM

Només suporten consultes SELECT.

Query

Els resultats no es mapejen a entitats Doctrine.

Es fa servir Doctrine DBAL.

SQL nadiu

```
<?php
```

```
use Doctrine\ORM\Query\ResultSetMappingBuilder;  
$rsmb = new ResultSetMappingBuilder($entityManager);  
$rsmb->addRootEntityFromClassMetadata('cinema\Entity\Film', 'f');  
$rsmb->addJoinedEntityFromClassMetadata(  
    'cinema\Entity\Coment',  
    'c',  
    'f',  
    'comentaris',  
    [  
        'id' => 'comment_id'  
    ]  
);  
$sql = <<<SQL  
SELECT movie.id, movie.original_title, comment.id as comment_id,  
comment.texto,  
comment.fecha  
FROM movie INNER JOIN comment ON movie.id = comment.pelicula_id  
WHERE movie.year >= 1988  
ORDER BY movie.id, comment.fecha  
SQL;  
$query = $entityManager->createNativeQuery($sql, $rsmb);  
$result = $query->getResult();
```

SQL nadiu

```
<?php
```

```
use Doctrine\ORM\Query\ResultSetMappingBuilder;
$rsmb = new ResultSetMappingBuilder($entityManager);
$rsmb->addRootEntityFromClassMetadata(cinema\Entity\Film, 'f');
$rsmb->addJoinedEntityFromClassMetadata(
    'cinema\Entity\Coment',
    'c',
    'f',
    'comentaris',
    [
        'id' => 'comment_id'
    ]
);
$sql = <<<SQL
SELECT movie.id, movie.original_title, comment.id as comment_id,
comment.texto,
comment.fecha
FROM movie INNER JOIN comment ON movie.id = comment.pelicula_id
WHERE movie.year >= 1988
ORDER BY movie.id, comment.fecha
SQL;
$query = $entityManager->createNativeQuery($sql, $rsmb);
$result = $query->getResult();
```

SQL nadiu

```
$sql = <<<SQL
SELECT spanish_title AS titulo, COUNT(comment.id) AS comentarios
FROM movie JOIN comment ON comment.pelicula_id = movie.id
GROUP BY movie.id
ORDER BY comentarios DESC, spanish_title ASC
LIMIT 5;
SQL;
$query = $entityManager->getConnection()->query($sql);
$result = $query->fetchAll();
```

Repositoris

Quan generem una entitat, Doctrine ens proporciona un repositori base per defecte per a aquesta entitat, amb una sèrie de mètodes comuns per a operar:

find (id): Retorna l'entitat amb l'identificador id, o null si no existeix.

findAll (): Retorna un array amb totes les entitats del repositori.

findBy (array (criteris) [, array (a ordenació)]): Retorna una matriu amb les entitats que compleixin els criteris especificats en el primer paràmetre, i ordenats pels del segon paràmetre (opcional).

findOneBy (array (criteris)) : Similar a findBy() però retornant només un element, o null.

Repositoris

Alguns mètodes del repositori base es poden fer servir de forma abreujada, permetent no especificar com paràmetre el nom de la propietat. P.ex.:

```
findByTitulo('valor'), findOneByTitulo('valor')
```

equival a

```
findBy('Titulo'=>'valor'), findOneBy('Titulo'=>'valor')
```

Això passa per la utilització del mètode màgic `__call()` de PHP que fa Doctrine al localitzar el nom del mètode de la classe.

Repositoris

El repositori base d'una entitat pot ser estès a fi de proporcionar mètodes específics per a consultes d'usuari.

Aquest repositori personalitzat permet optimitzar l'obtenció de resultats en les consultes, obtenint més control en la hidratació de les dades (p.ex: seleccionar parcialment entitats i / o especificar operacions a la consulta que no es farien de forma automàtica).

Doctrine permet especificar la classe que farem servir com a repositori de l'entitat a l'etiqueta @Entity, dins de les anotacions d'aquesta entitat.

```
@Entity(repositoryClass="FilmRepository")
```

La classe (buida) serà generada mitjançant les eines de consola (CLI) de Doctrine, executant:

```
php vendor/bin/doctrine.php orm:generate-repositories /src
```

El primer projecte

Una optimització bàsica del nostre exemple seria la construcció d'un repositori personalitzat per a l'entitat Film.

Aquest constarà de mètodes que ens permetin recuperar només les dades que necessitem mostrar en cada moment. Per això crearem consultes DQL de manera que ...

Es seleccionin només els camps específics de les entitats associades en lloc de recuperar tots ells (comportament per defecte).

Es faci en una sola operació el que d'una altra forma requeriria més d'una (p.ex: Recuperar dades d'una entitat associada amb `fetch = 'lazy'`).

El primer projecte

```
public function findConNumComentaris() {
    return $this
        ->createQueryBuilder( 'f' )
        ->leftJoin( 'f.comentaris', 'c' )
        ->addSelect( 'COUNT(c.id)' )
        ->groupBy( 'f.id' )
        ->getQuery()
        ->getResult();
}
```

Películas

Título	Título Original	Etiquetas		
Un puente lejano	A bridge too far	Etiqueta1 3 comentario(s).	Editar	Borrar
Matrix	Matrix	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 1 comentario(s).	Editar	Borrar
Nivel 13	The Thirteenth Floor	Etiqueta1 Etiqueta2 Etiqueta3 5 comentario(s).	Editar	Borrar
Blade Runner	Blade Runner	Etiqueta1 Etiqueta2 Etiqueta3 1 comentario(s).	Editar	Borrar
Moon	Moon	Etiqueta1 Etiqueta2 Etiqueta3 2 comentario(s).	Editar	Borrar
Brazil	Brazil	Etiqueta1 Etiqueta10 4 comentario(s).	Editar	Borrar
El crepúsculo de los dioses	Sunset Boulevard	Etiqueta1 2 comentario(s).	Editar	Borrar
La cosa	The thing	Etiqueta1 Etiqueta2 4 comentario(s).	Editar	Borrar
Alien, el octavo pasajero	Alien	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 1 comentario(s).	Editar	Borrar
La guerra de las galaxias	Star Wars	Etiqueta1 Etiqueta2 Etiqueta3 1 comentario(s).	Editar	Borrar
Tiburón	Jaws	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 Etiqueta5 5 comentario(s).	Editar	Borrar
El golpe	The sting	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 4 comentario(s).	Editar	Borrar
Primera Plana	The front page	Etiqueta1 1 comentario(s).	Editar	Borrar
Todos los hombres del presidente	All the president's men	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 Etiqueta5 1 comentario(s).	Editar	Borrar
Existenz	Existenz	Etiqueta1 Etiqueta2 5 comentario(s).	Editar	Borrar
El copón con ruedas	The wheeled copon	Etiqueta1 Etiqueta2 Etiqueta5 No hay comentarios	Editar	Borrar

Nueva película

El primer projecte

```
public function findTenitEtiquetes(array $etiquetes) {
    return $queryBuilder = $this
        ->createQueryBuilder('f')
        ->addSelect('t.nombre
            ->addSelect('COUNT(c.id)')
            ->join('f.etiquetes', 'e')
            ->leftJoin('f.comentarios', 'c')
            ->where('e.nombre IN (:etiquetes)')
            ->groupBy('f.id')
            ->having('COUNT(e.nombre) >= :numEtiquetes')
            ->setParameter('etiquetes', $etiquetes)
            ->setParameter('numEtiquetes', count($etiquetes))
            ->getQuery()
            ->getResult();
}
```

Películas

Titulo	Titulo Original	Etiquetas		
Nivel 13	The Thirteenth Floor	Etiqueta1 Etiqueta2 Etiqueta3 5 comentario(s).	Editar	Borrar
Moon	Moon	Etiqueta1 Etiqueta2 Etiqueta3 2 comentario(s).	Editar	Borrar
Tiburón	Jaws	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 Etiqueta5 10 comentario(s).	Editar	Borrar
El golpe	The Sting	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 4 comentario(s).	Editar	Borrar
Todos los hombres del presidente	All the President's Men	Etiqueta1 Etiqueta2 Etiqueta3 Etiqueta4 Etiqueta5 2 comentario(s).	Editar	Borrar

Nueva película

El primer projecte

```
public function findAmbComentaris($id) {
    return $this
        ->createQueryBuilder( 'f' )
        ->addSelect( 'c' )
        ->leftJoin( 'f.comentarios', 'c' )
        ->where( 'f.id = :id' )
        ->orderBy( 'c.fecha', 'ASC' )
        ->setParameter( 'id', $id )
        ->getQuery()
        ->getOneOrNullResult();
}
```

Un puente lejano

Título Original: **A bridge too far**

Director: **Richard Attenborough**

Año: **1977**

"Rusce id elementum mi. Phasellus suscipit nunc sit amet cursus lacinia. Sed luctus sed diam sed auctor. Quisque fringilla ante id sapien feugiat, quis laoreet enim consectetur."

2014-12-29 21:20:46

[Borrar comentario](#)

"Quisque sapien enim, congue vitae luctus a, scelerisque sed leo. Integer tempor sagittis neque, at trincidunt turpis posuere ac."

2015-01-05 21:20:46

[Borrar comentario](#)

"Sed porta in velit quis interdum. Nunc faucibus egestas est ut sagittis. Integer maximus eu dui eu consequat. Curabitur consectetur magna ut arcu ultricies rhoncus."

2015-01-12 21:20:46

[Borrar comentario](#)

Añada un nuevo Comentario

Comentario

[Volver al inicio](#)



Documentació

RECURSOS

Doctrine Project Site
<http://www.doctrine-project.org>

Notes on Doctrine 2
<http://www.krueckeberg.org/notes/d2.html>

Mastering Symfony2 performance
<http://labs.octivi.com/mastering-symfony2-performance-doctrine/>

Bibliografía
Persistence in PHP with Doctrine
ORM
(Packt Publishing)

JRLaguardia
<https://github.com/gonfert/cine>

PROJECTES SIMILARS

Propel
<http://propelorm.org>

Red Bean PHP4
<http://redbeanphp.com>

Spot ORM
<http://phpdatamapper.com>

<http://gnoma.es/blog/introduccion-a-doctrine-2/>

<http://gitnacho.github.io/symfony-doctrine-es/book/doctrine.html>

<http://web.ontuts.com/tutoriales/utiizando-doctrine-como-orm-en-php/>