DWES

Doctrine

Asociaciones con ORM v:^3

JESÚS MOLINA HERNÁNDEZ
jmolina@florida-uni.es

Índice

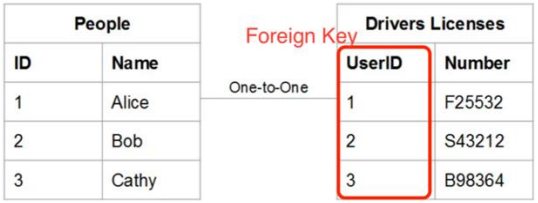
1. Introducción – qué problema queremos resolver?

Las relaciones entre tablas en BD.

Existen 3 tipos:

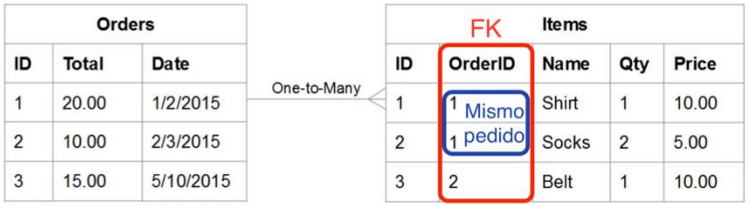
One-to-One

Un registro de la tabla B pertenece a un único registro de la tabla A. Y un registro de la tabla A pertenece a un único registro de la tabla B.



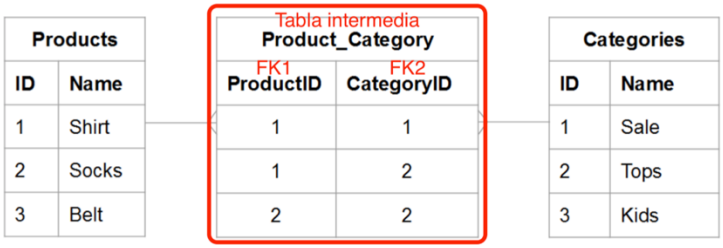
One-to-Many (Many-to-One)

Un registro de la tabla A puede pertenecer potencialmente a varios registros de la tabla B.



Many-to-Many

Un registro de la tabla B puede pertenecer potencialmente a varios registros de la tabla A. Y viceversa, un registro de la tabla A puede pertenecer potencialmente a varios registros de la tabla B.

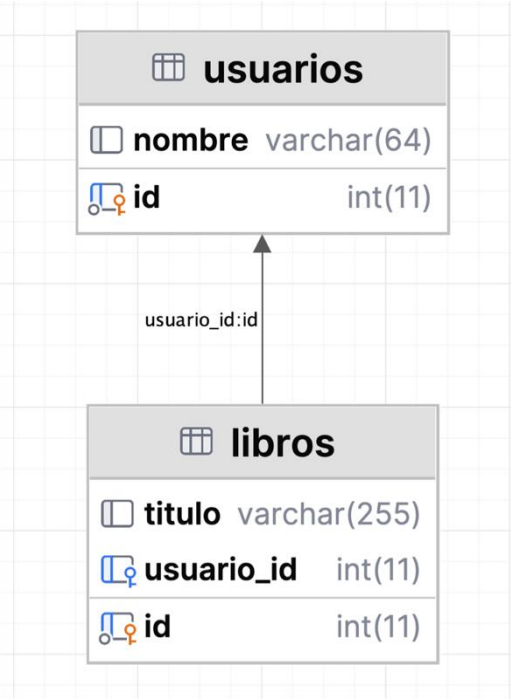


Jesús Molina Hernández – jmolina@florida-uni.es

Florida Secundària

1. Introducción

Las **asociaciones** en Doctrine reflejan las **relaciones entre entidades**, similares a las relaciones en bases de datos relacionales.



Jesús Molina Hernández – jmolina@florida-uni.es

2. Tipos de asociaciones en Doctrine

Tipos:

- **OneToOne:** una instancia de la entidad actual se refiere a una instancia de la entidad referida.
- **OneToMany:** una instancia de la entidad actual tiene muchas instancias (referencias) a la entidad referida.
- **ManyToOne:** muchas instancias de la entidad actual se refieren a una instancia de la entidad referida.
- **ManyToMany**: Muchas entidades están relacionadas con muchas entidades del otro lado.

Ejemplos

- **OneToOne**: *Una biblioteca con libros y usuarios. Cada libro sólo lo tiene un usuario en cada momento.* Relación 1to1 de libros con usuarios.
- **OneToMany**: *Mismo ejemplo, pero un usuario tiene muchos libros asignados. Relación 1toMany de usuarios a libros.*
- ***ManyToOne**: Mismo ejemplo, pero la relación hecha a la inversa, varios libros pueden estar asignados al mismo usuario. Relación Manyto1 de libros a usuarios.*
- ***ManytoMany**: Un usuario puede tener varios libros, y un libro puede estar asignado a varios usuarios a la vez*

Jesús Molina Hernández – jmolina@florida-uni.es

3. Asociación Unidireccional vs Bidireccional

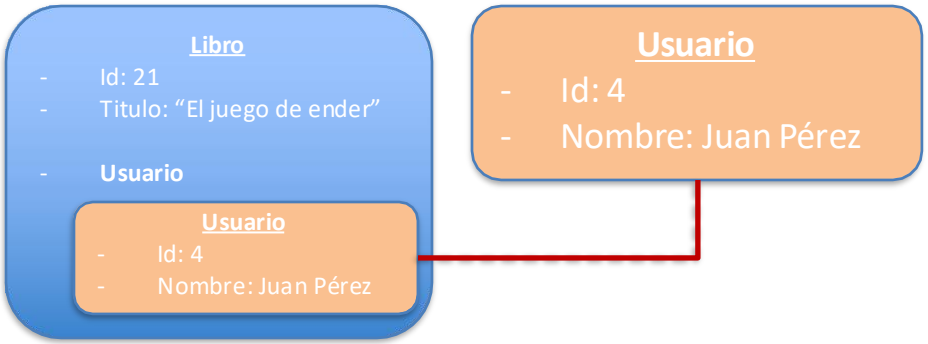
- **Unidireccional:** Sólo uno de los objetos conoce la relación.
- **Bidireccional:** Ambos objetos conocen la relación.

Esto afecta a la hora de definir la asociación en Doctrine como veremos más adelante

IMPORTANTE → Doctrine solo verifica si hay cambios en el lado propietario de la asociación. Es nuestro deber actualizar en ambos lados, ya que los cambios realizados solo en el lado inverso de una asociación se ignoran.

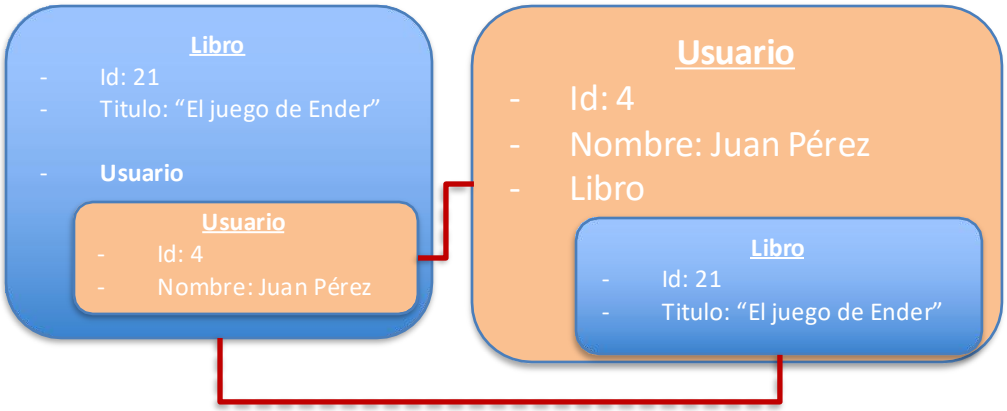
Jesús Molina Hernández – jmolina@florida-uni.es

3. Asociación Unidireccional vs Bidireccional



Unidireccional

El objeto libro de nuestro código sabe qué usuario lo tiene alquilado, pero el usuario no conoce qué libro tiene. Habría que ir al objeto libro para averiguarlo.



Bidireccional

El objeto libro sabe qué usuario lo tiene. El objeto usuario sabe qué libro tiene.

Los objetos son referencias entre sí.

Jesús Molina Hernández – jmolina@florida-uni.es

4. Definición de asociaciones con Doctrine

La asociación se define en la **Entity**

Debemos crear un nuevo atributo para la clase indicando dos cosas:

1. Tipo de relación:
   - **#[OneToOne()] | #[OneToMany()] | #[ManyToOne()] | #[ManyToMany()]**
2. Columna referenciada (o tabla en caso de existir tabla de unión):
   - **#[JoinColumn()] | #[JoinTable()]**

Jesús Molina Hernández – jmolina@florida-uni.es

4. Definición de asociaciones con Doctrine – Tipo asociación

Estas funciones (OneToOne, …), tienen como argumentos:

- **targetEntity:** hace referencia a la clase (Entity) referenciada.
- **inversedBy:** hace referencia al atributo que se usa en el lado inverso de la foreign key.
- **mappedBy:** hace referencia al atributo que se usa en el lado propietario de la foreign key.

Sólo se usan en relaciones **bidireccionales**

Jesús Molina Hernández – jmolina@florida-uni.es

4. Definición de asociaciones con Doctrine – JoinColumn

#[JoinColumn()]

- Se utiliza para especificar los detalles de la foreign key que conecta dos tablas relacionadas.

Los argumentos son:

- **name:** hace referencia al nombre de la foreign key en la BB.DD.
- **referencedColumnName:** hace referencia al nombre que tiene esa foreign key en su tabla origen

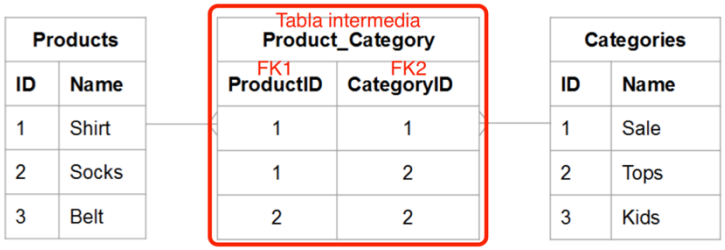
Si no uso JoinColumn() funcionará, pero con los valores por defecto de Doctrine que son:

- Al campo en BD le llamará 'nombreObjeto'_id
- Apuntará al campo 'id' del objeto

Jesús Molina Hernández – jmolina@florida-uni.es

4. Definición de asociaciones con Doctrine – JoinTable

JoinTable crea una tabla adicional (intermedia) que conecta ambas entidades.



Anatomía de `JoinTable`

1. `name`: Nombre de la tabla intermedia.
   - Ejemplo: `"usuarios_libros"`.
2. `joinColumns`:
   - Representa la columna que conecta con la entidad actual (`Usuario`).
   - Ejemplo:

```php
new JoinColumn(name: "usuario_id", referencedColumnName: "id")
```

3. `inverseJoinColumns`:
   - Representa la columna que conecta con la otra entidad (`Libro`).
   - Ejemplo:

```php
new JoinColumn(name: "libro_id", referencedColumnName: "id")
```


Jesús Molina Hernández – jmolina@florida-uni.es

5. Asociaciones OneToOne

Ejemplo: *OneToOne* ***unidireccional*** *de libros a usuarios.*

Clase referenciada

```php
#[Entity]
#[Table(name: "usuarios")]
class Usuario
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 64)]
    private string $nombre;
```

Clase que tiene la relación

```php
#[Entity]
#[Table(name: "libros")]
class Libro
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 255)]
    private string $titulo;

    #[OneToOne(targetEntity: Usuario::class)]
    #[JoinColumn(name: "usuario_id", referencedColumnName: "id", nullable: false)]
    private Usuario $usuario;
```


Jesús Molina Hernández – jmolina@florida-uni.es

5. Asociaciones OneToOne

Ejemplo: *OneToOne* ***bidireccional****.*
Libros es el principal.

Clase referenciada

```php
#[Entity]
#[Table(name: "usuarios")]
class Usuario
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 64)]
    private string $nombre;

    #[OneToOne(mappedBy: "usuario", targetEntity: Libro::class)]
    private ?Libro $libro = null;
```

Clase que tiene la relación

```php
#[Entity]
#[Table(name: "libros")]
class Libro
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 255)]
    private string $titulo;

    #[OneToOne(inversedBy: "libro", targetEntity: Usuario::class)]
    #[JoinColumn(name: "usuario_id", referencedColumnName: "id", nullable: false)]
    private ?Usuario $usuario = null;
```

En la BD esta es la tabla que tiene la FK

Jesús Molina Hernández – jmolina@florida-uni.es

6. Asociaciones OneToMany (ManyToOne)

Ejemplo: *OneToMany/ManyToOne* ***unidireccional****. Un usuario puede tener muchos libros*

Clase referenciada

```php
#[Entity]
#[Table(name: "usuarios")]
class Usuario
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 64)]
    private string $nombre;
```

Clase que tiene la relación

```php
#[Entity]
#[Table(name: "libros")]
class Libro
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 255)]
    private string $titulo;

    #[ManyToOne(targetEntity: Usuario::class, inversedBy: "libros")]
    #[JoinColumn(name: "usuario_id", referencedColumnName: "id", nullable: false)]
    private ?Usuario $usuario = null;
```


Jesús Molina Hernández – jmolina@florida-uni.es

6. Asociaciones OneToMany (ManyToOne)

Ejemplo: *OneToMany/ManyToOne* ***bidireccional****. Un usuario puede tener muchos libros*

Clase referenciada

```php
#[Entity]
#[Table(name: "usuarios")]
class Usuario
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 64)]
    private string $nombre;

    #[OneToMany(mappedBy: "usuario", targetEntity: Libro::class, cascade: ["persist", "remove"])]
    private Collection $libros;
```

Clase que tiene la relación

```php
#[Entity]
#[Table(name: "libros")]
class Libro
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 255)]
    private string $titulo;

    #[ManyToOne(targetEntity: Usuario::class, inversedBy: "libros")]
    #[JoinColumn(name: "usuario_id", referencedColumnName: "id", nullable: false, onDelete: "CASCADE")]
    private ?Usuario $usuario = null;
```


Jesús Molina Hernández – jmolina@florida-uni.es

7. Asociaciones ManyToMany

Ejemplo: *ManyToMany* ***unidireccional***

```php
#[Entity]
#[Table(name: "usuarios")]
class Usuario
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 64)]
    private string $nombre;

    #[ManyToMany(targetEntity: Libro::class, cascade: ["persist", "remove"])]
    #[JoinTable(
        name: "usuarios_libros",
        joinColumns: [new JoinColumn(name: "usuario_id", referencedColumnName: "id")],
        inverseJoinColumns: [new JoinColumn(name: "libro_id", referencedColumnName: "id")]
    )]
    private Collection $libros;
```

```php
#[Entity]
#[Table(name: "libros")]
class Libro
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 255)]
    private string $titulo;
}
```


Jesús Molina Hernández – jmolina@florida-uni.es

7. Asociaciones ManyToMany

Ejemplo:** ManyToMany **bidireccional

```php
#[Entity]
#[Table(name: "usuarios")]
class Usuario
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 64)]
    private string $nombre;

    #[ManyToMany(targetEntity: Libro::class, inversedBy: "usuarios")]
    #[JoinTable(
        name: "usuarios_libros",
        joinColumns: [new JoinColumn(name: "usuario_id", referencedColumnName: "id")],
        inverseJoinColumns: [new JoinColumn(name: "libro_id", referencedColumnName: "id")]
    )]
    private Collection $libros;
```

```php
#[Entity]
#[Table(name: "libros")]
class Libro
{
    #[Id]
    #[GeneratedValue]
    #[Column(type: "integer")]
    private int $id;

    #[Column(type: "string", length: 255)]
    private string $titulo;

    #[ManyToMany(targetEntity: Usuario::class, mappedBy: "libros")]
    private Collection $usuarios;
```

JoinTable() podría estar en el otro lado.

Jesús Molina Hernández – jmolina@florida-uni.es

8. Collections

Por la propia definición de los arrays en PHP, son excelentes para el uso del ORM de las asociaciones de muchos valores. Para ello, lo haremos mediante la interfaz ***Collection*** y lo implementaremos por defecto mediante ***ArrayCollection*** usando el namespace ***Doctrine\Common\Collections***.

Además, **Collection** implementa las interfaces de PHP **ArrayAccess**, **Traversable** y **Countable**.

Siempre debe inicializarse las Collections

de los *#[OneToMany()]* y *#[ManyToMany()]*

IMPORTANTE Inicializar

```php
<?php
use Doctrine\Common\Collections\Collection;
use Doctrine\Common\Collections\ArrayCollection;

#[Entity]
class User
{
    /** Many Users have Many Groups. */
    #[ManyToMany(targetEntity: Group::class)]
    private Collection $groups;

    public function __construct()
    {
        $this->groups = new ArrayCollection();
    }

    public function getGroups(): Collection
    {
        return $this->groups;
    }
}
```

Ejemplo de uso:

```php
<?php
$group = new Group();
$user = new User();
$user->getGroups()->add($group);
```


Jesús Molina Hernández – jmolina@florida-uni.es