Relaciones con Doctrine en

Symfony

Fernando Díaz-Alonso Dorado

Florida Universitària

Entidades en Symfony

Crear Entidades

Para crear una nueva entidad:

php bin/console make:entity

Sin necesidad de evaluar como va a ser la BB.DD. Sabemos que vamos a necesitar un objeto con el nombre de la entidad. Si vamos a mostrar muchos productos en nuestra aplicación necesitaremos un objeto: Product.

Tras indicar el nombre de la entidad el mismo comando se encargará de ir pidiéndonos los datos para definir cada campo:

- Nombre
- Tipo
- Longitud si corresponde
- ¿Nulo?

```
Class name of the entity to create or update (e.g. BraveElephant):
> Product

created: src/Entity/Product.php
created: src/Repository/ProductRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.
```

Relaciones en Symfony

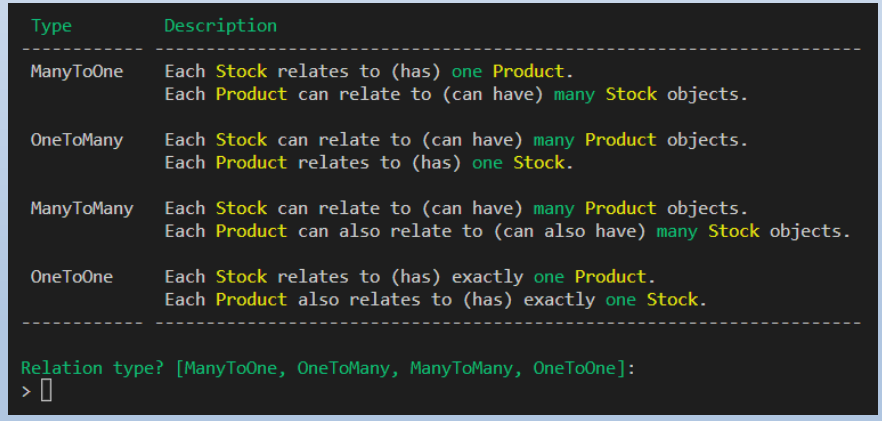
Crear Relaciones

Para crear una relación necesitamos dos entidades. Por lo tanto, vamos a crear una nueva entidad Stock que contendrá una columna asociada al productos (Product).

Para ello, actualizaremos la entidad para añadir la relación. Y lo haremos indicando que el tipo de dato es: ***relation***.

Entonces nos pedirá:

- El nombre del campo en la entidad
- El tipo de asociación
- Si puede ser nula la asociación.
- Si quieres que se unidireccional o bidireccional.



Relaciones en Symfony

Crear Relaciones II

```
 ManyToOne    Each Stock relates to (has) one Product.
              Each Product can relate to (can have) many Stock objects.

 OneToMany    Each Stock can relate to (can have) many Product objects.
              Each Product relates to (has) one Stock.

 ManyToMany   Each Stock can relate to (can have) many Product objects.
              Each Product can also relate to (can also have) many Stock objects.

 OneToOne     Each Stock relates to (has) exactly one Product.
              Each Product also relates to (has) exactly one Stock.
------------ ---------------------------------------------------------------------

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
> ManyToOne

Is the Stock.Product property allowed to be null (nullable)? (yes/no) [yes]:
> no

Do you want to add a new property to Product so that you can access/update Stock objects from it - e.g. $product->getStocks()? (yes/no) [yes]:
> yes

A new property will also be added to the Product class so that you can access the related Stock objects from it.

New field name inside Product [stocks]:
> Stocks

Do you want to activate orphanRemoval on your relationship?
A Stock is "orphaned" when it is removed from its related Product.
e.g. $product->removeStock($stock)

NOTE: If a Stock may *change* from one Product to another, answer "no".

Do you want to automatically delete orphaned App\Entity\Stock objects (orphanRemoval)? (yes/no) [no]:
> no

updated: src/Entity/Stock.php
updated: src/Entity/Product.php
```

Relaciones en Symfony

Crear Relaciones III

Product.php | Stock.php

05 - SYMFONY > ejemplo_app > src > Entity > Product.php > PHP Intelephense > Product

```
#[ORM\OneToMany(mappedBy: 'Product', targetEntity: Stock::class)]
private Collection $Stocks;

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

```
public function addStock(Stock $stock): self
{
    if (!$this->Stocks->contains($stock)) {
        $this->Stocks->add($stock);
        $stock->setProduct($this);
    }

    return $this;
}

public function removeStock(Stock $stock): self
{
    if ($this->Stocks->removeElement($stock)) {
        // set the owning side to null (unless already changed)
        if ($stock->getProduct() === $this) {
            $stock->setProduct(null);
        }
    }

    return $this;
}
```

Stock.php

...Y > ejemplo_app > src > Entity > Stock.php > PHP I...

```
private ?string $stock_cant = null;

#[ORM\ManyToOne(inversedBy: 'Stocks')]
#[ORM\JoinColumn(nullable: false)]
private ?Product $Product = null;
```

```
public function getProduct(): ?Product
{
    return $this->Product;
}

public function setProduct(?Product $Product): self
{
    $this->Product = $Product;

    return $this;
}
```

Relaciones en Symfony

Crear Relaciones IV

IMPORTANTE: Las relaciones OneToMany se han de crear a mano.

Únicamente se crean las relaciones ManyToOne con este tipo de comando porque al crearlas nos pregunta si queremos crear el otro lado.

> `ManyToOne` / `OneToMany`
> The most common relationship, mapped in the database with a foreign key column (e.g. a `category_id` column on the `product` table). This is actually only *one* association type, but seen from the two different *sides* of the relation.

https://symfony.com/doc/current/doctrine/associations.html

Crear BB.DD. Symfony

Migrar desde Doctrine a la BB.DD.

Una vez configuradas completamente las entidades desde el lado de Doctrine tendremos que guardarlo en la tabla correspondiente en la base de datos, porque aún no existirá en ella.

Para ello, vamos a usar DoctrineMigrationsBundle que ya tenemos instalado.

De forma que ahora ejecutamos el comando:

php bin/console make:migration

Que nos informará si todo ha salido correctamente y veremos que ha creado un archivo PHP dentro de la carpeta migrations.

Crear BB.DD. Symfony

Migrar desde Doctrine a la BB.DD. II

El archivo generado tras la migración contiene los comandos SQL necesarios para actualizar la BB.DD.

Ahora, nos quedará únicamente terminar la migración mediante el comando:

php bin/console doctrine:migrations:migrate

Que se encarga de ejecutar el archivo que hemos generado anteriormente.

https://symfony.com/doc/current/doctrine.html#migrations-creating-the-database-tables-schema

Crear BB.DD. Symfony

Migrar desde Doctrine a la BB.DD. III

Si tenemos que volver a modificar la entidad tras migrar.

Simplemente podemos volver a hacer *make:entity* o modificar el código de la entidad y posteriormente lanzar los comandos de la migración para volver a sincronizar nuestra aplicación con la BB.DD.

php bin/console make:migration

php bin/console doctrine:migrations:migrate

https://symfony.com/doc/current/doctrine.html#migrations-adding-more-fields

Crear BB.DD. Symfony

Generar las entidades desde una BB.DD.

Para generar las entidades desde una BB.DD. ya existente deberemos de utilizar los siguientes comandos:

php bin/console doctrine:mapping:import "App\Entity" annotation – path=src/Entity

Posteriormente podemos crear los getters y setters:

php bin/console make:entity –regenerate App

En este caso no es necesario realizar una migración, si no realizamos ningún tipo de cambio sobre la entidad.

IMPORTANTE: A partir de 2019, Doctrine ha indicado como obsoleto el método mapping:import y recomienda usar make:entity.

https://symfony.com/doc/current/doctrine/reverse_engineering.html