Doctrine

Premiers pas

Fil directeur de cette présentation

Création d'une « todo liste » en 4 heures

Pré-requis

Savoir installer et créer des contrôleurs Symfony

Connaître la syntaxeTwig

Maîtriser la programmation orientée Objet : instanciation, héritage, getters/setters, méthodes magiques et Composer

Code source public

https://github.com/AlexisRomeroDev/DoctrineTutoral

1. Installation de Doctrine et création BDD

Installation de Doctrine

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

```
composer require symfony/orm-pack
```

Créer la base de données

Puis renseigner la configuration à la base de données dans le fichier .env

Plusieurs drivers sont possibles :
mysql, sqlite, postgresql

Créer la base de données

```
bin/console doctrine:database:create
```

Installation du makerBundle

https://symfony.com/bundles/SymfonyMakerBundle/current/index.html

```
composer require --dev symfony/maker-bundle
```

2. Création d'une entité Doctrine

Générer une entité

```
bin/console make:entity
```

Création d'une entité Todo

Analyse des fichiers générés

```
git status
```

src/Entity/Todo.php

L'entité Todo est une classe php.
Se référer au cours sur la POO

L'entité Todo comporte des annotations :
@ORM\Entity( )
@ORM\Column()
Ces annotations définissent le modèle physique de données. Une entité correspond à une table en base de données et chaque propriété est une colonne de cette table.
Tout savoir sur les annotations :
https://www.doctrine-project.org/projects/doctrine-orm/en/2.7/reference/annotations-reference.html

Il existe des formats alternatifs aux annotations :

Par ex, les formats XML et YAML

src/Repository/Todo.php

Le repository centralise tout ce qui touche à la récupération des entités en base de données.

Migrations

Pour créer ce schéma en base de données : 2 commandes :

```
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

Modifier une entité

Ajouter un champs « nom »

Relancer les commandes :

```
bin/console make:entity
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

3. CRUD

Pour notre entité, on va vouloir faire des opérations de base : Ajouter des todo, les afficher, les modifier et les supprimer.

Installer les bundles

```
composer require form validator twig-bundle security-csrf annotations
```

Générer un Crud

```
bin/console make:crud
```

Tester dans un navigateur : http://127.0.0.1:8000/todo/

Analyse des modifications dans le contrôleur

Dans la méthode index(), qui a vocation a retourner la liste des todos :

```
$todoRepository-> findAll()
```

Cette méthode récupère tous les todos actuellement en base de données. Cette méthode est disponible par héritage de la classe lib/Doctrine/ORM/EntityRepository.php. Quelques méthodes disponibles :

- findAll()
- find($id)
- findBy()
- findOneBy()
- ...

Dans la méthode show(), qui a vocation à retourner le todo dont l'id est passé en paramètre d'url, pas de méthode « magique ». Le todo est récupéré grace au mécanisme de ParamConterter

Pour en saoir plus sur les ParamConverter :
https://symfony.com/bundles/SensioFrameworkExtraBundle/current/annotations/converters.html

Dans les méthode edit(), new() et delete() :

```
$entityManager->persist($todo);
$entityManager->remove($todo);
$entityManager->flush();
```

Ce sont les méthodes traditionnelles pour insérer, modifier et supprimer des données en base.

Noter l'utilisation de la classe TodoType src/Form/TodoType.php qui permet de créer un formulaire HTML correspondant à l'entité en Base de donnée.

1 Entité = 1 FormType

Les formulaires seront traités dans un autre Module

V5

Modifier une entité existante

On souhaite ajouter une description à chaque Todo

Modification de l'entité

La modification d'une entité, comme la création d'ailleurs, peut se faire manuellement, ou bien par la commande

```
bin/console make:entity
```

Cette commande ajoute une nouvelle propriété à la classe TODO

```
    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $description;
```

ainsi que les getters et les setters.

Modification du formulaire

Il faut, dans tous les cas, compléter la classe TodoType

```
        $builder->add('description');
```

4. Fixtures

Installation de bundles

```
composer require --dev orm-fixtures
composer require fzaninotto/faker --dev
```

Docs :

https://symfony.com/bundles/DoctrineFixturesBundle/current/index.html

https://github.com/fzaninotto/Faker

Rédaction des fixtures & chargement en base de données

Consulter la documentation

5. Trier les données

Requêtes sur les données

Méthode 1 : dans le controller

Dans le contrôleur

```
$todo = $todoRepository->findBy(
                array(),
                array('name' => 'ASC')
            )
```

Méhode rapide , mais limitée et code non-réutilisatble.

Méhode 2 : avec le queryBuilder dans le repo

Dans le TodoRepository

```
    public function findAllOrdered()
    {
        return $this->createQueryBuilder('t')
            ->orderBy('t.name', 'ASC')
            ->getQuery()
            ->getResult()
        ;
    }
```

Méthode permettant de faire des requêtes poussées, notamment des jointures et code réutilisable
Résultats sous forme d'objets

Méthode 3 : en SQL dans le repo

https://symfony.com/doc/current/doctrine.html#querying-with-sql
Méthode permettant de faire des requêtes poussées, notamment des jointures et code réutilisable
Résultats sous forme de tableaux

Le QueryBuilder

Méthodes du QueryBuilder

orderBy()

Pour trier les données

where() / andWhere()

pour filtrer les données

setMaxResults()

pour limiter le nombre de résutats

setParameter() setParameters()

pour passer un/des paramètre(s)

Liste exhaustive :
https://www.doctrine-project.org/projects/doctrine-orm/en/current/reference/query-builder.html

Méthodes de la Query

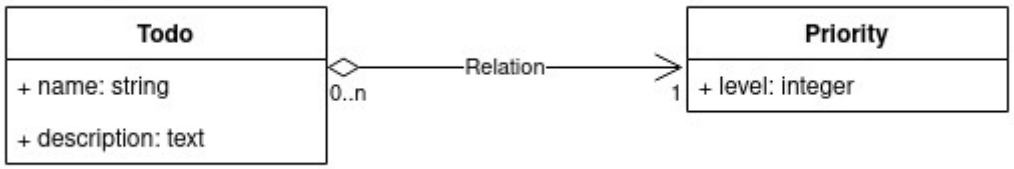
getResullt()

getOneOrNullResult()

V8

6. Associer deux entités

CAHIER DES CHARGES : On souhaite définir un niveau de priorité dans les TODO



Mise à jour du Schéma

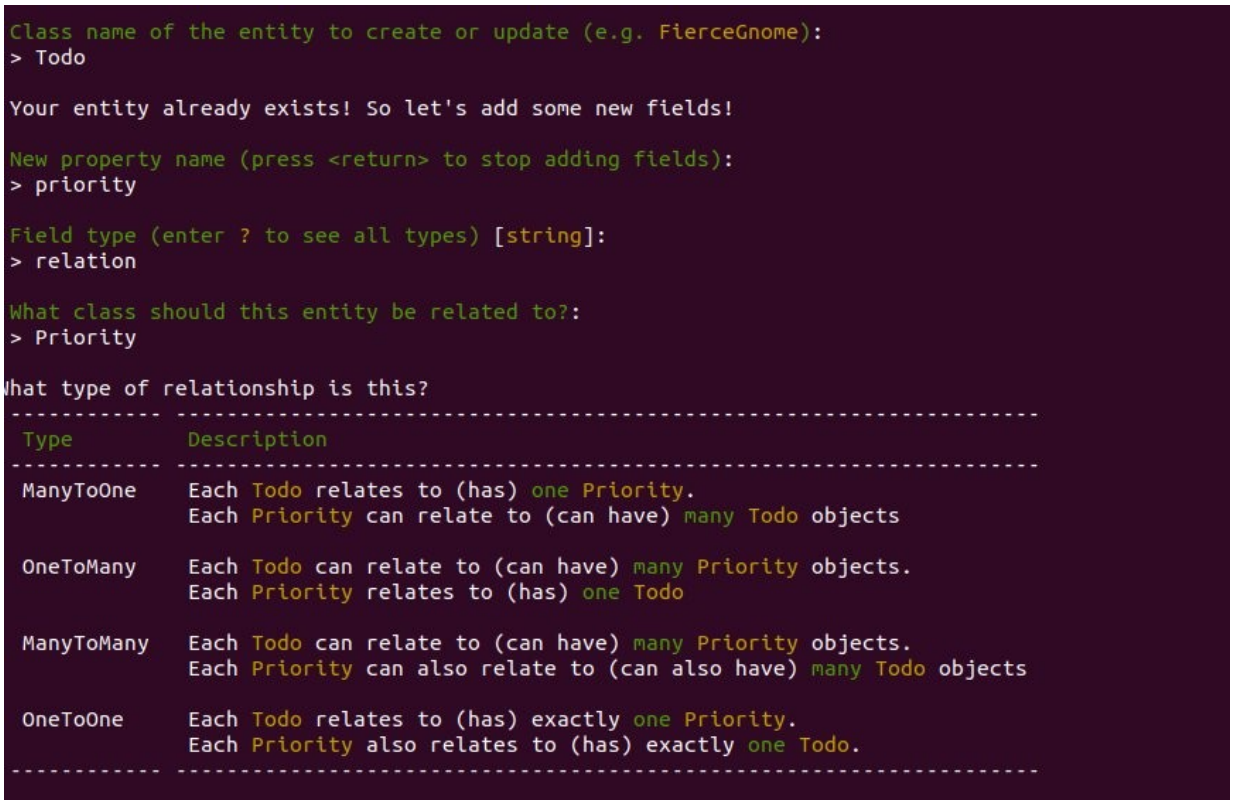
Création de l'entité Priority

```
bin/console make:entity
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

Modification de l'entité Todo

```
bin/console make:entity
```

Le type de champ est ManyToOne



Documentation :

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

```
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

Création de quelques priorités

```
INSERT INTO `priority` (`level`) VALUES (1),(2),(3),(4);
```

Mise à jour du formulaire d'édition/création

Ajout du champ dans TodoType

```
        $builder
            ->add('name')
            ->add('description')
            ->add('priority')
```

```
        ;
```

Méthode __toString de la classe Priority

```
    public function __toString()
    {
        return (string) $this->level;
    }
```

cf cours POO

Conclusion

Après cette découverte de Doctrine, les étudiants sont invités à reproduire cet exemple.

Les enseignements à suivre :

- Persister en cascade
- Unidirectionnalité / Bidirectionnalité
- Gérer le problème N+1
- Les extensions de Doctrine
- Implémenter l’héritage
- Tirer parti du cycle de vie Doctrine