

Table des matières

[1. 2](#_Toc12627973)

[2. 2](#_Toc12627974)

[3. 2](#_Toc12627975)

[4. 2](#_Toc12627976)

[5. 2](#_Toc12627977)

[6. 2](#_Toc12627978)

[7. 2](#_Toc12627979)

[8. 2](#_Toc12627980)

[9. Astuces 3](#_Toc12627981)

Tuto 3/4 , blog :

Gestion des entités et leurs relations.

1. Les entités.

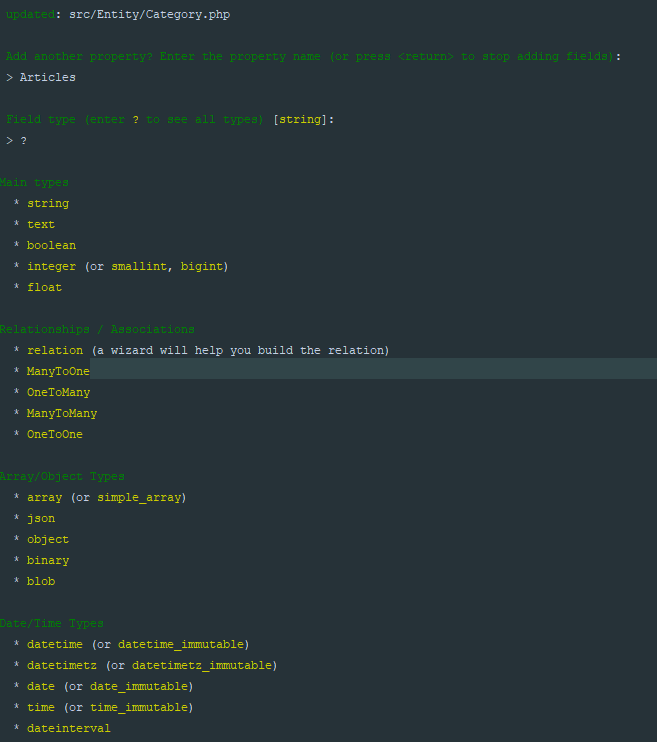
Creation d’une nouvelle entité : Category

Php bin/console make : entity

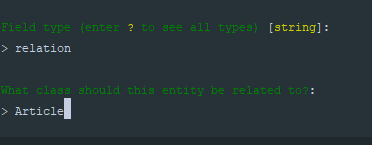
Ajout des champs : title et description

Ensuite afin de faire le lien avec article, dans la demande de création d’un champ supplémentaire on ajoute : Articles

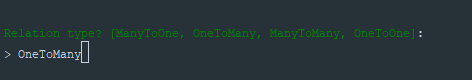
C’est un champ de type « relation ».



On le lie à la classe Article



Et on créer la relation :



Une fois les champs complétés :

Php bin/console make :migration

Ensuite pour mettre à jour la bdd :

php bin/console doctrine:migrations:migrate

Ajout de l’entité : commentaire idem que Category

Ajout des données dans la base :



composer require fzaninotto/faker

Pour utiliser faker :



*<?php  
  
namespace* App\DataFixtures;  
  
*use* App\Entity\Article;  
*use* App\Entity\Category;  
*use* App\Entity\Comment;  
*use* DateTime;  
*use* Doctrine\Bundle\FixturesBundle\Fixture;  
*use* Doctrine\Common\Persistence\ObjectManager;  
  
*class* ArticleFixtures *extends* Fixture  
{  
 *public function* load(ObjectManager $manager)  
 {  
 *//utlisation de faker pour compléter la BDD, ici on précise le lieu "fr"* $faker = \Faker\Factory::create ('fr\_FR');  
  
 *//Création de 3 catégories fakées  
 for* ($i=1;$i<=3; $i++){  
 $category = *New* Category();  
 $category ->setTitle($faker -> sentence())  
 ->setDescription($faker -> paragraph());  
 $manager ->persist($category);  
  
 *//Creer entre 4 et 6 articles  
  
 for* ($j=0; $j<=mt\_rand(4,6); $j++){  
 $article = *new* Article();  
  
 *//on place les paragraphs de faker qui sont des tableaux  
 //dans une variable* $content = '<p>';  
 *//on ajout a content avec le "."* $content.=join($faker->paragraphs(5),'</p><p>');  
 $content.='</p>';  
  
 $article ->setTitle($faker -> sentence())  
 -> setContent($content)  
 -> setImage($faker ->imageUrl())  
 -> setCreatedAt($faker->dateTimeBetween('-6 mounths'))  
 *// on ajout la catégorie à l'article* -> setCategory($category);  
  
 $manager->persist($article);  
  
 *//on donne des commentaires à l'article  
 for* ($k=1; $k <= mt\_rand(4,10);$k++){  
  
  
 $content = '<p>';  
 *//on ajout a content avec le "."* $content.=join($faker->paragraphs(2),'</p><p>');  
 $content.='</p>';  
  
 $now = *new* \DateTime();  
 $interval =$now -> diff($article-> getCreatedAt());  
 $days = $interval ->days;  
 $minimum ='-'.$days.'days'; *// -100 days* $comment =*new* Comment();  
 $comment->setAuthor ($faker ->name)  
 ->setContent($content)  
 ->setCreatedAt($faker ->dateTimeBetween($minimum))  
 ->setArticle ($article);  
 $manager->persist($comment);  
  
 }  
 }  
  
 }  
 $manager->flush();  
 }  
}

Ensuite lancer les fixtures via la console :



La BDD est complétée et bien fournie avec plusieurs exemples.

1. Affichage
2. {% *extends* 'base.html.twig' %}  
     
   {% *block* body %}  
     
    <article>  
    <h2 >{{ article.title }} </h2>  
    <div *class*="metadata"> Ecrit le {{ article.createdAt | date('d/m/Y') }} à {{ article.createdAt | date ('H:i') }} dans la catégorie  
    {{ article.category.title }}</div>  
    <div *content*>  
    <img *src*='{{ article.image }}' *alt*="">  
    {{ article.content | raw }}  
    </div>  
    </article>  
     
    <section *id*="commentaires">  
    {% *for* comment *in* article.comments %}  
    <div *class*="comment">  
    <div *class*="row">  
    <div *class*="col-3">  
    {{ comment.author }} (<small>{{ comment.createdAt | date ('d/m/Y à H:i') }}</small>)  
    </div>  
    <div *class*="col">  
    {{ comment.content | raw }}  
    </div>  
    </div>  
    {% *endfor* %}  
    </section>  
     
   {% *endblock* %}

Les commentaires sont présents et mis en page à coté des articles.

7. Astuces

-Cls : pour nettoyer l’écran du terminal.

-Alt + F12 : pour ouvrir un terminal dans php storm

-Permet d’inserer des images et de simuler leurs présences. Ex : <img *src*="http://placehold.it/350x150" *alt*="">

- le lien vers le tuto : https://www.youtube.com/watch?v=UTusmVpwJXo