ORM : Objet relationnel mapping

Repository : pour chaque entité on va créer un repository.

Sites references  doc symfony : doctrine symfony et twig

Via site internet : <https://secuweb.aesisoft.fr/?symfony/GetStarted>

symfony new -full ProjectName

On fait après ca :

\* Bin/console make:entity produit // produit étant le nom

On mettra le property name : designation

On laissera string avec 255

On mettra description puis text puis yes // Si on ne sait pas on met « ? » il nous aidera

\* poids puis float puis no(peut pas etre null)

\*make :entity categorie

Puis on mettra designation - string –

🡺ON aura ainsi dans le categorie.php la liste des public function

-> Il est possible de modifier en retapant la commande (make :entity produit)

En validant on pourra rajouter ce qu’on à oublié.

Entité produit/catégorie produit/le string est relation

\* On va avec l’assistant voir si l’on souhaite ManyToOne (relationtype), on dit s’il peut etre null et on add.

Une fois cela fait, on aura dans categorie la class produits et les functions

\*Important : on configure ensuite la BDD. Pour cela, on va dans le fichier env et la l32 : on change le root/mdp

\*On va ensuite sur le mémo des commandes utiles : on créer la bdd avec les 4 commandes php/bin …

// On va créer la bdd

php bin/console doctrine:database:create

// Il à créer les tables de migration dans version…

php bin/console make:migration

// On va créer nos tables

php bin/console doctrine:migrations:migrate

Ensuite on va lancer le var-dumper puis le data fixture

composer require --dev symfony/var-dumper

composer require --dev doctrine/doctrine-fixtures-bundle

Il a créé un DataFixtures que l’on va utiliser

On va devoir faire dans le public function load(Object Manager $manager)

{

$categorie = new Categorie() ;

$cateforie->setNom(‘Boisson’) ;

$produit = new Produit() ;

$produit->setDesignation(‘Vodka’) ;

$produit->set Prix(15) ;

$produit->setCategorie($categorie) ;

$manager->persist($produit) ;

$manager->flush() ; // Si on a souligné le produit : on fait click droit import class (dl sinon l’extension le php NameSpace Resolver)

}

\* On va ensuite les ajouter dans la bdd :

composer require --dev doctrine/doctrine-fixtures-bundle

🡪Si cela marche dans produit on a vodka, on aura les tables vidées avec la purge.

CRUD

\*On va créer le controller, le form et les templates

php bin/console make:crud NameEntity

Dans le produit controller on aura la route /produit.

\*symfony serve : on lance le serveur avec le localhost/produit

On aura la liste des produits.

\* on recupere dans creation d’une entité :

public function \_\_toString(): string

{

return $this->nom;

}

On l’ajoute à categorie.php après la function remove

La méthode \_\_toString() ; afficher sous forme de chaine l’objet

-> On aura donc un’ bel’ affichage ☺

\* Dans base.html.twig on fait le lien du css et le css on créer un dossier assets dans public puis des dossiers dedans css et img et js avec un index.php

Dans le base ca sera après le title et avant block : <link rel= « stylesheet » href= »{{ asset(‘assets/css/style.css’) }} » />

\* On va pour faire le lien entre block body et block javascript :

On fera ainsi le lien du script d’image : <script src = « {{ asset (‘assets/js/app.js’’) }} » />

\* On va créer un controller :

php bin/console make:controller Home

\* On va voir la liste de toutes les routes :

php bin/console debug:router

Aparthé : On va travailler sur le diagramme de classe commun et le back-office d’un prestashop pour voir le bo d’un site d’e-commerce : easy admin

A faire

- Créer la vue/affichage de la boutique

- Back office easyadmin : je prends

- Gestion du panier

- Validation du paiement

- Création API pour avoir une interface front/end

-