**Atelier Base de données & Doctrine ORM**

1. **Installation du “support” Doctrine**

Ouvrez le terminal (console cmd, powershell, GIT bash, terminal de Visual code, **PhpStorm: View🡪Tool Windows🡪Terminal**)

composer require symfony/orm-pack

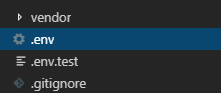
composer require --dev symfony/maker-bundle

ou si on utilise le fichier composer.phar dans le dossier “my-project”

php composer.phar require symfony/orm-pack

php composer.phar require --dev symfony/maker-bundle

Ouvrez le fichier de configuration dans votre arborescence, ".env"



Ce fichier contient le nom de la base de données qui sera crée par défaut "db\_name", vous pouvez le changer évidemment

Allez chercher cette ligne (à peu près ligne 27)

# customize this line!

DATABASE\_URL="mysql://db\_user:db\_password@127.0.0.1:3306/db\_name"

Modifier la ligne comme suit

DATABASE\_URL=mysql://root@127.0.0.1:3306/db\_name

Avec “db\_name” est le nom de la base de données qu’on va créer

Pour créer la base Tapez dans la console la commande suivante:

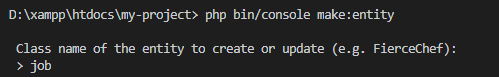
php bin/console doctrine:database:create

ouvrez PHPMyadmin pour vérifier la création de la base de données

1. **Création de l’entité :**

php bin/console make:entity

suivez les réponses suivantes



Entrez l’attribut « type » (tapez entrée pour String, 255 et nullable)

New property name (press <return> to stop adding fields):

> type

Field type (enter ? to see all types) [string]:

>

Field length [255]:

>

Can this field be null in the database (nullable) (yes/no) [no]:

>

La dernière question si vous répondez par “yes” c’est que vous permettez au champ d’avoir une valeur “NULL”, si vous appuyez sur “Entrée” (c-à-d [no]) vous jugez que ce champ est obligatoire

Ajoutez les attributs suivants :

* Company (String, 255,obligatoire)
* Description (text, obligatoire)
* Expires\_at (datetime, obligatoire)

**A la fin vous avez l’entité « job » dans le dossier  « ENTITY » ouvrez la et vérifiez le contenu**

1. **Création de la table :**

On doit installer d’abord le [DoctrineMigrationsBundle](https://github.com/doctrine/DoctrineMigrationsBundle), ce n’est pas obligatoire de le faire il existe déjà dans votre installation

Tapez d’abord dans la console:

**php bin/console make:migration**

vous aurez

Success!

Next: Review the new migration "src/Migrations/Version20190404221341.php"

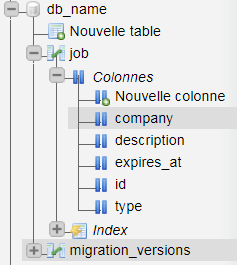
Then: Run the migration with php bin/console doctrine:migrations:migrate

See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html

Maintenant on peut créer la table, Tapez

**php bin/console doctrine:migrations:migrate**

Vérifier que la table est crée convenablement



1. **Ajout d’un nouveau champ contenant :**

On veut ajouter à l’entité “Job” déjà crée le champ “email” de type “String” de taille “255” et obligatoire

Vous pouvez l’ajouter directement dans l’entité avec les annotations nécessaires évidemment, **ou réécrire l’instruction:**

php bin/console make:entity

puis re introduisez le nom l’entité que vous avez introduit correctement “Job” si vous n’écrivez pas le nom correcte une nouvelle entité sera crée et vous ratez la modification

php bin/console make:entity

Class name of the entity to create or update

> Job

> email

Field type (enter ? to see all types) [string]:

Maintenant il faut migrer cette modification à notre base de données, il faut surtout pas le faire manuellement via SQL (alter table…) mais le système est intelligent tapez :

php bin/console make:migration

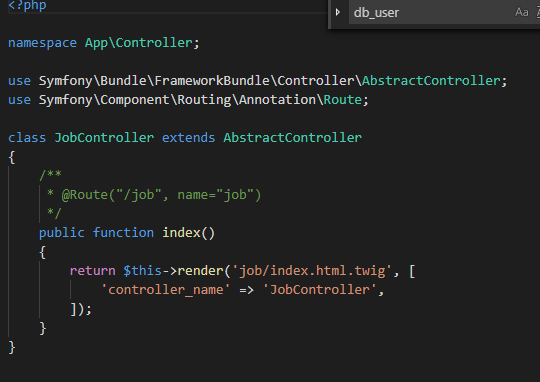
php bin/console doctrine:migrations:migrate

1. **Persisting d’objets dans la base de données**

Création d’un nouveau contrôleur

php bin/console make:controller JobController

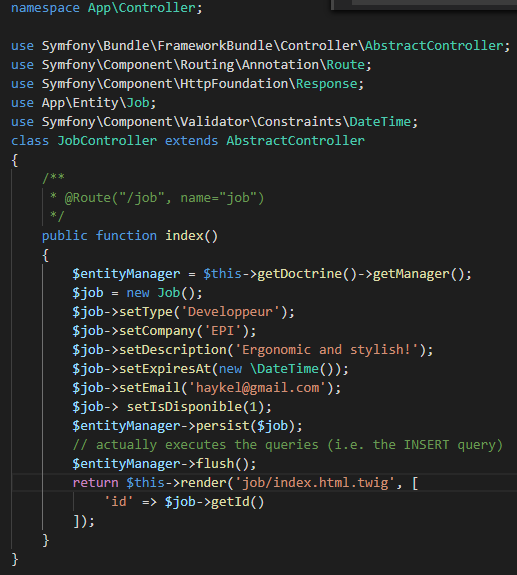
ouvrez le nouveau fichier



Modifier le contrôleur comme suit:

Remarque : implémentez juste les champs que vous avez déterminé dans l’entité « job » par exemple si vous n’avez pas fait le champ « is\_disponible » alors ne mettez pas la ligne

  $job-> setIsDisponible(1);



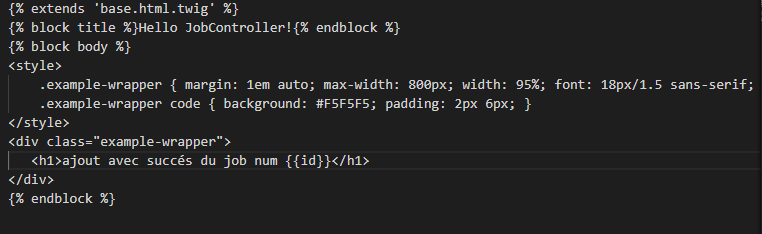
N’écrivez pas cette ligne

Le principe de l’ajout est le fait de créer un objet vide puis affecter les valeurs, dans ce cas, à travers les setters puis dans un autre Atelier va utiliser un formulaire pour remplir les champs. Enfin persister cet objet avec l’objet “entityManager”.

**persist($candidat)**n’insère pas l’objet immédiatement c’est **flush()qui le confirme**

En créant le contrôleur via le terminal, un dossier est crée automatiquement sous “Templates” portant le même nom, ici “job”

Ouvrez le fichier “index.html.twig” et modifier le comme suit



**Le service EntityManager** : C'est le service EntityManager qui permet de dire à Doctrine « Persiste cet objet », c'est lui qui va exécuter les requêtes SQL.

**< ?php**

**$em=$this->getDoctrine()->getManager();**

Le seul problème c’est qu’il ne peut pas récupérer facilement les entités depuis la base de données. Pour faciliter l'accès aux objets, on va utiliser un Repository

**Les repositories**: On parle des repositories au pluriel car **il en existe un par entité**.

Quand on parle d'un repository en particulier, il faut donc toujours préciser le repository de quelle entité, afin de bien savoir de quoi on parle

< ?php

$em=$this->getDoctrine()->getManager();

$jobRepository=$em->getRepository(Job::class)

**L'EntityManager** : sert à manipuler les entités

**Les repositories** : servent à récupérer les entités

1. **Fetching d’un objet à partir de la base de données (find)**

Ajouter une nouvelle action dans **JobController**

*/\*\**

*\* @Route("/job/{id}", name="job\_show")*

*\*/*

public function show($id)

{

$job = $this->getDoctrine()

->getRepository(Job::class)

->find($id);

if (!$job) {

throw $this->createNotFoundException(

'No job found for id '.$id

);

}

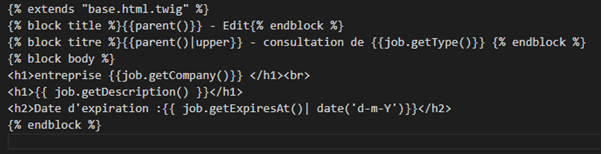
return $this->render('job/show.html.twig', [

'job' =>$job

]);

}

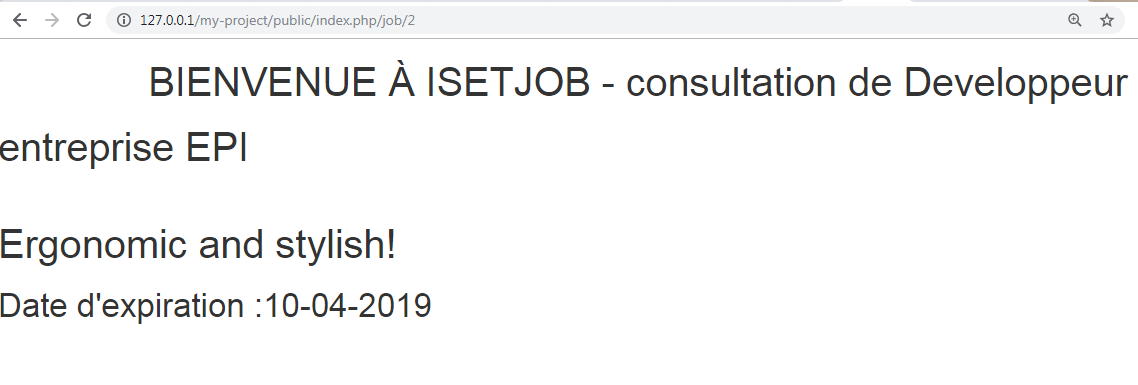
Créer dans le dossier job le fichier « show.html.twig » contenant



'job' =>$job

‘job’ est wrapper (une enveloppe) contenant l’objet $job qui est transmis au fichier twig, ce dernier récupère cette enveloppe et la décortique à travers les getters qui permettent d’afficher les détails de cet objet exemple **{{ job.getExpiresAt() | date (‘d-m-Y’) }} ici on utilise un filtre pour formater la date.**

Essayer avec <http://127.0.0.1/my-project/public/index.php/job/2>



1. **Relations unidirectionnelles entre les entités**

****

Création de l’entité « image »

php bin/console make:entity

nom de l’entité : Image

* champ : url( String 255, nullable=no)
* champ alt( String 255, nullable=yes)

Ouvrez le fichier « image.php » dans le dossier « Entity » et vérifiez le

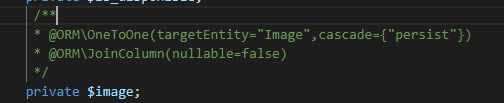
Enregistrer la nouvelle entité en Base de données :

php bin/console make:migration

php bin/console doctrine:migrations:migrate

Définir la relation entres les 2 entités :

Ajouter ceci à « Job.php » (après la déclaration du champ $email)



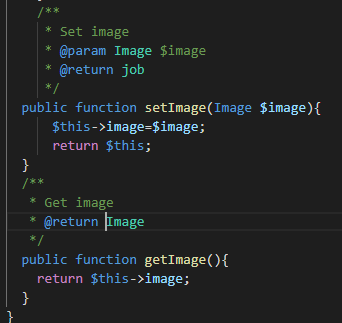
N’oubliez pas d’ajouter dans la partie des « use »

use App\Entity\Image;

**Ajouter aussi le getter et setter du nouveau champ de relation comme suit (à la fin de la partie contenant les getters et setters également) :**

**php bin/console make:entity --regenerate App**

**Cette commande va générer automatique les getters et les setters manquants**

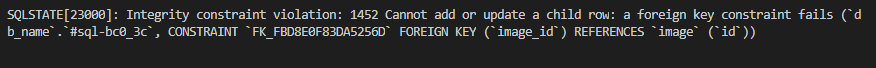


Mettre à jour la base de données

php bin/console make:migration

php bin/console doctrine:migrations:migrate

vous aurez ce message d’erreur puisque les lignes déjà insérées dans la table job n’ont pas de valeurs pour cette nouvelle clé étrangère



Ouvrez la table job



Vous pouvez rectifier cette erreur après avoir ajouter des images dans la table “image” et affecter la clé primaire de ces images (comme clés étrangères) dans le champ “image\_id”

**Modifier « JobController.php »**

Ajouter

use App\Entity\Image;

Puis modifier la fonction d’ajout comme suit

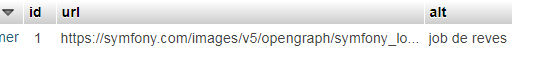


Remarque utilisez ce lien comme source d’image et non pas celui qui se trouve dans le code ci-dessus :

*https://cdn.pixabay.com/photo/2015/10/30/10/03/gold-1013618\_960\_720.jpg*

Enregistrez et lancer « <http://127.0.0.1/my-project/public/index.php/job> »

Puis consulter la base :



Modifier le code du  « show.html.twig » pour afficher l’image :

