Fiche de procédure : Commandes de base Symfony 7

1. Créer un projet Symfony :

```
symfony new nom_du_projet --webapp --version="7.1"
```

2. Démarrage du serveur

Démarrer le serveur local Symfony :

```
symfony server:start -d
symfony serve -d
```

Le -d lance le serveur en arrière plan pour ne pas bloquer le terminal
Cela empêche par contre de voir les logs directement

Arrêter le serveur :

```
symfony server:stop
```

3. Gestionnaire de dépendances (Composer)

Initialiser composer / créer le dossier vendor:

```
composer install
```

Installer une dépendance :

```
composer require nom_du_package
(exemple : composer require symfony/maker-bundle)
```

Mettre à jour les dépendances :

```
composer update
```

4. Génération et gestion de la base de données (avec doctrine)

Il faut tout d'abord modifier / créer le fichier .env (ou .env.local) et mettre la ligne de connexion à la base de données :

```
DATABASE_URL="postgresql://allavenavr:Btssio2024@172.20.96.1:5432/he
gresphere"

allavenavr   → nom d'utilisateur
Btssio2024   → mot de passe
@172.20.96.1 → ip du serveur où se trouve la base de données
:5432        → port
/hegresphere → nom de la base de données
```

Créer la base de données :

```
symfony doctrine:database:create
peut être abrégé en symfony do:da:cr
```

Création des migrations :

```
symfony make:migration

Cela va créer un fichier de migration avec le SQL qui sera exécuté
sur la base de données
```

Exécution des migrations :

```
symfony doctrine:migrations:migrate

Va jouer la migration
```

Vérifier la connexion à la base de données :

```
symfony doctrine:database:validate
```

5. Commandes pour créer des composants (maker)

Créer une entité :

```
symfony make:entity NomDeLEntité
<?php
```

```php
namespace App\Entity;

use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity]
class Adherent
{
    #[ORM\Id] // clé primaire
    #[ORM\Column]
    #[ORM\GeneratedValue] // auto incrémental
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $nom = null;

    #[ORM\Column(length: 255)]
    private ?string $prenom = null;

    #[ORM\Column(length: 255)]
    private ?string $email = null;

    #[ORM\Column(type: Types::DATE_MUTABLE)]
    private ?\DateTime $dateInscription = null;
// \DateTime → le "\" = fonction php

// relation one to many
// on peut la créer depuis le maker en ajoutant un attribut de type relation lors de la création de l'entité
    #[ORM\OneToMany(targetEntity: Inscription::class, mappedBy: 'adherent')]
    private Collection $inscriptions;

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

// les getter et setter (peuvent être générés par l'IDE)
    public function getId(): ?int
    {
        return $this->id;
    }
```

```php
    public function getNom(): ?string
    {
        return $this->nom;
    }

    setNom ...
    getPrenom ...
    setPrenom ...
    getEmail ...
    setEmail ...

    public function getDateInscription(): ?\DateTime
    {
        return $this->dateInscription;
    }

    public function setDateInscription(?\DateTime
$dateInscription): void
    {
        $this->dateInscription = $dateInscription;
    }

    public function getInscriptions(): Collection
    {
        return $this->inscriptions;
    }

    public function addInscription(Inscription $inscription):
static
    {
        if (!$this->inscriptions->contains($inscription)) {
            $this->inscriptions->add($inscription);
            $inscription->setAdherent($this);
        }
        return $this;
    }

    public function removeInscription(Inscription $inscription):
static
    {
        if ($this->inscriptions->removeElement($inscription)) {
            // set the owning side to null (unless already changed)
            if ($inscription->getAdherent() === $this) {
                $inscription->setAdherent(null);
            }
        }
        return $this;
    }
```

```
}
```

Créer un contrôleur :

```
symfony make:controller NomDuController
```

```php
<?php

namespace App\Controller;

// tous les imports
use App\Entity\Adherent;
use App\Form\AdherentType;
use App\Repository\AdherentRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

// format pour une route
#[Route('/adherent', name: 'app_adherent')]
class AdherentController extends AbstractController
{

// permet d'accéder aux repository
    public function __construct(
        private readonly EntityManagerInterface $entityManager,
        private readonly AdherentRepository $adherentRepository
    )
    {}
// pour accéder à cette route : /adherent/add
    #[Route('/add', name: '_add')]

// fonction pour ajouter un objet adherent dans la bdd
    public function add(Request $request): Response
    {
        $adherent = new Adherent();
        $form = $this->createForm(AdherentType::class, $adherent);
        $form->handleRequest($request);

// important !!
        if($form->isSubmitted() && $form->isValid())
        {
            $this->entityManager->persist($adherent);
            $this->entityManager->flush();
        }
```

```
        return $this->render('adherent/add.html.twig', [
            'formAdd' => $form,
        ]);
    }

    #[Route('/list', name: '_list')]
    public function list(): Response
    {
        $adherents = $this->adherentRepository->findAll();

        return $this->render('adherent/list.html.twig', [
            'adherents' => $adherents,
        ]);
    }

    #[Route('/update/{id}', name: '_update')]
    public function update(int $id, Request $request): Response
    {
        $adherent = $this->adherentRepository->find($id);
        $form = $this->createForm(AdherentType::class, $adherent);
        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid())
        {
            $this->entityManager->persist($adherent);
            $this->entityManager->flush();
        }

        return $this->render('adherent/add.html.twig', [
            'formAdd' => $form,
        ]);
    }

    #[Route('/delete/{id}', name: '_delete')]
    public function delete(int $id): Response
    {
        $adherent = $this->adherentRepository->find($id);
        $this->entityManager->remove($adherent);
        $this->entityManager->flush();

        return $this->redirectToRoute('app_adherent_list');
    }
}
```

Créer un formulaire :

```
symfony make:form NomDuFormulaire
```

```php
<?php

namespace App\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\DateType;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class AdherentType extends AbstractType
{
   public function buildForm(FormBuilderInterface $builder, array
$options): void
   {
       $builder
           ->add('nom', TextType::class)
           ->add('prenom', TextType::class)
           ->add('email', EmailType::class)
           ->add('dateInscription', DateType::class)
           ->add('enregistrer', SubmitType::class)
       ;
   }

   public function configureOptions(OptionsResolver $resolver):
void
   {
       $resolver->setDefaults([
           // Configure your form options here
       ]);
   }
}
```

Créer tout en même temps CRUD :

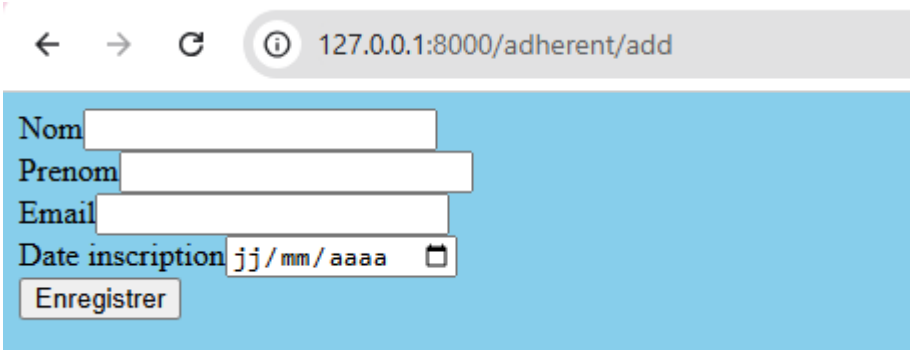
```
symfony make:CRUD
```

Pour l'affichage on utilise des twig qui vont utiliser les données que l'on va leur envoyer pour afficher de l'HTML :

```
{% extends 'base.html.twig' %}

{% block title %}Hello AdherentController!{% endblock %}

{% block body %}
    {{ form(formAdd) }} // affiche tout le formulaire
{% endblock %}
```



8. Autres commandes utiles

Effacer le cache :

```
symfony cache:clear
```

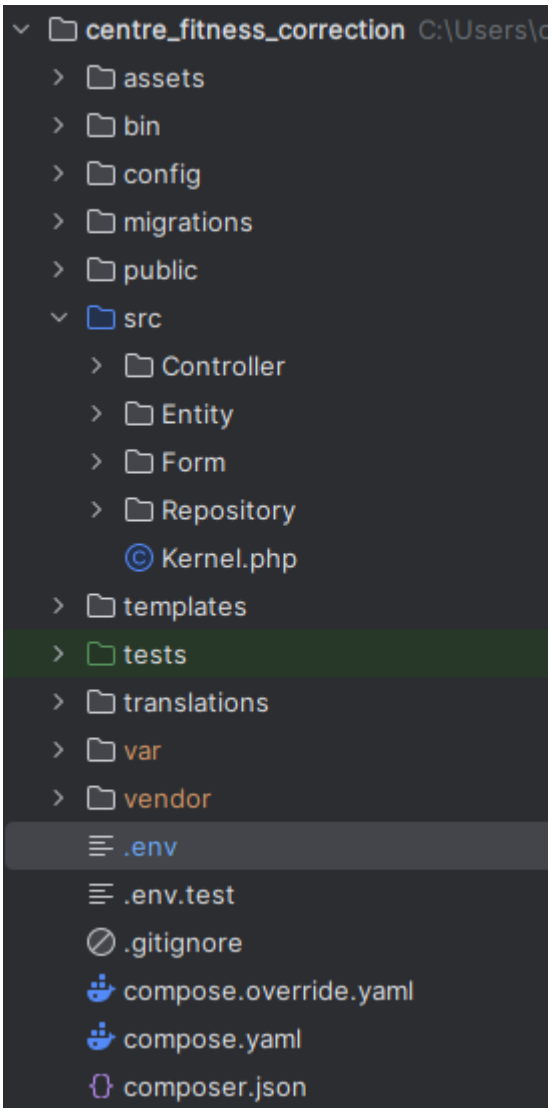
Consulter les routes :

```
symfony debug:router
```

Lister les services :

```
symfony debug:container
```

Voici l'arborescence d'un projet symfony :



C'est dans le dossier **src** que la plupart des fichiers utilisés seront rangés.
On retrouve bien **les contrôleurs, les entités, les formulaires et les repository.**

Les vues twig sont dans le dossier **templates**

Les différents types pour les données dans un formulaire :

Texte	Choix	Date et temps	Divers	Boutons	Caché
TextType TextareaType EmailType IntegerType MoneyType NumberType PasswordType PercentType SearchType UrlTypeRange	ChoiceType EntityType CountryType LanguageType LocaleType TimezoneType CurrencyType	DateType DatetimeType TimeType BirthdayType	CheckboxType FileType RadioType CollectionType RepeatedType	SubmitType ButtonType ResetType	HiddenType CsrfType

Les fonctions implémentés par le repository permettant de faire de recherches :

find($id) > retourne un objet suivant son id
findAll() > retourne un tableau de tous les d'objets
findOneBy($critère, $tri) > retourne le premier objet obtenu
findBy($critère, $tri, $limite, $décalage) > retourne un tableau d'objets