Symfony & Composer
Les bases

Création d'un projet sur Symfony:

Pour commencer un projet il faut le créer, donc cliquez droit sur le document de votre choix où vous voulez que votre dossier soit crée , puis cliquez sur «Git bash here»..

Insérez la ligne de commande suivante dans votre terminal.
$ composer create-project symfony/website-skeleton leNomDeMonProjet "4.4.*"
**Pour le nom du projet mettez le nom de votre choix «leNomDeMonProjet» est un simple exemple*
à la suite, après quelques secondes de téléchargement..

Vous allez donc rentrer dans votre projet, en restant sur la même fenêtre de votre terminal en insérant la commande suivante.
$ cd leNomDeMonProjet

Maintenant avant tout de chose il faut faire démarrer le serveur, car sans le serveur vous ne verrez pas le résultat si cela fonctionne ou non, donc pour démarrer le serveur à la suite, vous insérez la commande ci-dessous.
$ php -S localhost:8000 -t public

Normalement maintenant, ça marche sans problème.
Ouvrez donc votre navigateur et naviguer sur l'adresse http://localhost:8000
Si cela affiche «Welcome to Symfony» sur votre navigateur, vous êtes dans le bon chemin..

Maintenant, via votre éditeur de texte ouvrez le dossier de votre projet que vous venez de tout juste créer.

Pour créer une page avec Symfony, il faut remplir ces deux conditions:
- Créer une route
- Créer un contrôleur

Pour créer la route nous avons besoin de: Son nom, son chemin,et son contrôleur.
Nous allons donc nous diriger vers le fichier qui ce trouve dans « config *» et le fichier ce nomme «* routes.yaml *» ouvrez-le.*
et insérez le code ci-dessous:

```
homepage:
        path: /
        controller: App\Controller\NomController::index
```

Nous créons une route qui s'appelle homepage dont le chemin est / et qui a comme contrôleur la méthode index qui se trouve dans la classe NomController.
**Pour le controlleur mettez le nom de votre choix «NomController» est un un simple exemple*
Mais vérifiez bien qu'il soit inscrit de la même façon partout pour les liaisons de vos pages.

Allons donc créer ce contrôleur et faire le lien avec notre route.
Pour créer un contrôleur, nous allons retourner , à la suite de notre ligne de commande ou nous avons créer notre projet, et insérez la commande suivante:
$ php bin/console make:controller NomController

À savoir: Le controlleur gère la réponse à une route

Si tout est ok, un petit carré vert inscrit Success apparaît, là, 2 fichiers ont été créés, NomController.php dans src/Controller et index.html.twig dans templates/leNomDeMonProjet

Ouvrons maintenant le fichier src/Controller/NomController.php

Symfony nous a créer le contrôleur, et a même défini une route en utilisant ce qu'on appelle des annotations.

Pour retourner une page HTML: Twig

{{ ... }} *Afficher quelque chose, une variable par exemple*
{% ... %} *Faire quelque chose, définir une variable, une boucle, structure conditionnelles*
{# ... #} *Commenter du texte*
Dans le dossier templates , contient tous les templates que nous allons développer si vous regardez bien, il doit y avoir un dossier nom (le vôtre est peut être différent, tout dépend le nom que vous avez donné à la création de votre contrôleur), à l'intérieur de ce dossier se trouve le fichier index.html.twig

Modifiez donc le fichier index.html.twig avec un simple code html comme ceci:
```
<!DOCTYPE html>
<html>
<head>
        <title>Bienvenue sur mon blog</title>
        <meta charset=»utf-8»>
</head>
<body>
        <h1>Hello World!</h1>
        <h2>Bienvenue sur mon blog!</h2>
</body>
</html>
```

Une fois que cela à été modifié.

Allons maintenant modifier le contrôleur dans le fichier NomController.php

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class BlogController extends AbstractController
{
    public function index(){
        return $this->render('nom/index.html.twig');
    }
}
```

Nous utilisons la méthode render() pour retourner la page html, en paramètre nous l'envoyons le chemin de notre template nom/index.html.twig

Enregistrons tout cela, et allons voir sur notre navigateur http://localhost:8000
Notre page s'affiche normalement.

Création de la DATABASE:

Le fichier .env qui se trouve à la racine, ce fichier contient les configurations de notre application comme les informations sur la bases de données.
Ouvrez donc ce fichier.

Modifié donc cette ligne comme ci dessous, afin qu'elle s'adapte bien à votre base de donnée, moi elle se trouve à la ligne 32 de mon fichier.

DATABASE_URL=»mysql://root@localhost:3306/nomdevotrebasededonnée»
**En exemple cette ligne est configuré sans mot de passe pour la base de données*

DATABASE_URL=»mysql://root:123@localhost:3306/nomdevotrebasededonnée»
**En exemple cette ligne est configuré avec un mot de passe*

Pour créer la database via la ligne de commandes insérez le code suivant:
$ php bin/console doctrine:database:create
créer une ENTITY
$ php bin/console make:entity

Créer la table associé à l'entité
$ php bin/console doctrine:schema:update --dump-sql
Pour exécuter la requette sql
$ php bin/console doctrine:schema:update --force

Création d'un formulaire:

La database doit être créée, l'entité ainsi que les tables associés (voir page précédente)

créer le formulaire:

```
$ php bin/console make:form
```

créer une route dans le fichier routes.yaml qui va aller sur la page du formulaire

exemple:

```
addBillet:
    path: /addBillet
    controller: App\Controller\NomController::addBillet
```

créer dans le contrôleur la méthode pointée par la route:

- Ajouter les fichiers USE de l'ENTITY et du FORM

exemple:

```
public function addBillet(Request $request): Response
 {
   // créer un billet vierge
        $billet = new Billet();

   // créer le formulaire et je l'associe avec le billet vierge
        $formulaire = $this->createForm(BilletType::class, $billet);

// j'associe au formulaire un handle sur les données POST
                    $formulaire->handleRequest($request);

// je check si le request est un type post
                    if ( $request->isMethod('POST'))
                    {
// check si le formulaire a été soumis  et si les champs sont valables
                     if ( $formulaire->isSubmitted() && $formulaire->isValid())
                     {
                        // je demande à doctrine un handle pour manager la BDD
                         $entityManager = $this->getDoctrine()->getManager();

                         // verification des champs qui contiennent de l'objet Billet
                          $entityManager->persist( $billet );

                          // je descend toutes les donnée dans la base
                          $entityManager->flush();
                      }
                    }

                    // affichage de la page par TWIG
                    return $this->render('toto/addBillet.html.twig',
                        [
                            'formulaire' => $formulaire->createView()
                        ]);
                    //return new Response( «<h2>bienvenue</h2>» );
                }
```

créer le fichier TWIG

qui se trouve dans le dossier templates et qui se nommera addbillet.html.twig et insérez ce code suivant:

```
<H1>ajouter un billet</H1>

    {{ form( formulaire ) }}
```