Symfony

Alexzandre
LE HELLAY

Installation du framework

composer create-project symfony/skeleton:”7.0.*” nom_du_projet

cd nom_du_projet

composer require webapp

y

Les Différent dossier

.env
Piloter les variable d’environnement (ex: choix de la base de donnée)

public
Dossier Racine

Démarrage d’un serveur

php -S localhost:8000 -t public

Nos premières pages

php bin/console make:controller nom_du_controlle [HomeController]

Ça vas crée deux chose
- src/Controlle/Nom_du_controller.php [src/Controlle/HomeController.php]
- templates/Nom_du_controller/index.html.twig [templates/Home/index.html.twig]

src/Controller/Nom_du_controller.php [src/Controller/HomeController.php]

Supprimer la fonction par défaut :

Ajout d’un text sur une page vide

```
Fonction index (): Response {
        Return new Response(‘Bonjour les gens’);
}
```

Création d’une route

config/routes.yaml

Ajout d’une nouvelle route

```
home:
        path: /
        controller: App\Controller\Nom_du_controller::index   [App\Controller\HomeController::index]
```

Ou

[src/Controller/HomeController.php]

```
Class HomeController {
        #[Route("/", name: "home"]

        Function …
```

Connaitre tout les route

```
php bin/console debug:router
```

Exemple

Localhost 8000/?name=john

[src/Controller/HomeController.php]

```
Class HomeController extends AbstractController {
        #[Route("/", name: "home")]
        Function index (): Response {
                Return new Response('Bonjour ' . $_GET['name']);
        }
}
```

Avancer

```
        Function index (Request $request): Response {
                dd($request);
                Return new Response('Bonjour ' . $request->query->get('name', 'Inconnu'));
        }
```

Autre

```
Php bin/controle make:controller RecipeController
```

src/Controller/RecipeController.php

```
Class RecipeController extends AbstractController {
        #[Route('/recette/{slug}-{id}', name: 'recipe.show')]

        Public Function index(Request $request): Response
        {
                dd($request->attributes->get('slug'), $request->attributes->get('id'))
        }
}
```

```
Class RecipeController extends AbstractController {
        #[Route('/recette/{slug}-{id}', name: 'recipe.show', requirements: ['id' => '\d+', 'slug' => '[a-z0-9-]+'
        ])]

        Public Function index(Request $request): Response
        {
                dd($request->attributes->get('slug'), $request->attributes->getInt('id'))
        }
}
```

```
Class RecipeController extends AbstractController {
        #[Route('/recette/{slug}-{id}', name: 'recipe.show', requirements: ['id' => '\d+', 'slug' => '[a-z0-9-]+' ])]

        Public Function index(Request $request, string $slug, int $id): Response
        {
                dd($slug, $id);
}
```

src/Controller/RecipeController.php

```
Class RecipeController extends AbstractController {

        #[Route('/recette', name: 'recipe.index')]

        Public Function index(Request $request): Response
        {
                return new Response('Recettes');
        }

        #[Route('/recette/{slug}-{id}', name: 'recipe.show', requirements: ['id' => '\d+', 'slug' => '[a-z0-9-]+' ])]
        {
                Return new Response('Recette : ' . $slug);
                        Ou
                Return new JsonResponse([
                        'slug' => $slug
                ]);
                        Ou
                Return $this->json([
                        'slug' => $slug
                ]);

        }
}
```

Moteur de template Twig

template/recipe/index.html.twig

```
{% extends 'base.html.twig' %}  // la référence de tout les pages

{% block title %} blablabla {% endblock%}
                        Ou
{% block title "Toutes les recettes" %}

{% block body %}
azdazhjdoajdoiajdoazjo
{% endblock%}
```

src/Controller/RecipeController.php

```
Public …
{
        Return $this->render('recipe/index.html.twig');
```

```
}
```

Modification de la page template/base.html.twig pour modifier tout les autre page en ajoutant le css ou une nav bar par défaut.

Exemple

Creation d'un fichier dans template/recipe [show.html.twig]

Modification de Controller/RecipeController

```
        #[Route('/recette/{slug}-{id}', name: 'recipe.show', requirements: ['id' => '\d+', 'slug' => '[a-z0-9-]+' ])]
        {
                Return $this->render('recipe/show.html.twig', [
                        'slug' => $slug
                        'id' => $id

                        'person' => [
                                'firstname' => 'John',
                                'lastname' => 'Doe'
                        ]
                ]);
        }
}
```

Modification de la page show.html.twig

Ajout

```
{% block title "Recette : "  ~ slug %}

{% block body %}
        {{ slug }} {{ person.firstname ~ person.lastname}}
{% endblock%}
```

Les information sont échapper ex: sera écrit en tout lettre et pas considérer comme une balise
OU
Ajouter le filtre raw

Ajout de filtre
https://twig.symfony.com/doc/3.x/

```
{{ slug | upper }}  //majuscule
```

Nav Url

On a le choix entre url ou path (préférence sur path)

```
<a class="navbar-brand" href="/">Accueil </a>   // nop
```

```
<a class="navbar-brand" href="{{ path('home') }}">Accueil </a>  //home est le name de ma route
```

Note : 13:01

Exemple

recipe/index.html.twig

```
{% block body %}
        <ul>
                <li>
                        <a href="{{ path('recipe.show', {id: 32, slug: 'pate-bolognaise'}) }}">pâtes bolo </a>
                </li>
        </ul>
{% endblock%}
```

base.html.twig

```
        <ul>
                <li>
                        <a class="nav-link {{ app.current_route == 'home' . 'active' : '' }}" href={{
                        path('hom') }}">Accueil</a>
                </li>
                <li>
                        <a class="nav-link {{ app.current_route starts with 'recipe.' ? 'active' : '' }}" href="{{
                        path('recipe.index') }}">Recettes</a>
                </li>
        </ul>
```

Include

L'ORM Doctrine

Base de donnée gérer au niveau .env

Téléchargement de adminer qu'on va aller le placer dnas le dossier public et le nomée (adminer.php)

https://www.adminer.org/en/#download

Downloads

- **Adminer 4.8.1** (.php, 465 kB), **English only** (.php, 310 kB)
- **Adminer 4.8.1 for MySQL** (.php, 356 kB), **English only** (.php, 208 kB)
- **Source codes** (.zip, 785 kB), **Current development version**
- Donate: **Paypal**, **Patreon**, **Revolut**
- Latest stable version (use e.g. by `wget`): `https://www.adminer.org/latest[-mysql][-en].php`
- **Change log**, **blog**
- User contributed packages: **Debian package**, **Arch Linux package**, **Wordpress plugin**, **Drupal module**, **Docker**, **Dockette**, **Joomla extension**, **Moodle plugin**, **TYPO3 extension**, **CMS Made Simple Module**, **Laravel**, **Laravel**, **Laravel**, **AMPPS**, **Electron**, **Jaxon**
- Adminer is also bundled with **Nette Framework** (which this site runs on).
- **Older versions**

Puis allez sur localhost:8000/adminer.php

Exemple
Système : MySQL
Serveur : 127.0.0.1
Username : root
Password : password
Database : main

Verification de la version :
SELECT VERSION();

Changer dans .env

DATABASE_URL = “mysql://root:password@127.0.0.1:3306/main?serverVersion=8.0.31…

Creation d’une entity

Les entité ce trouve dans src/Entity/nom de l’entity

Php bin/console make:entity
// demande le nom
> Recipe
// broadcast entity
> No
// nom du fields (champ)
> title
// type
(? pour connaitre tout les type ET mettre rien si c’est string)
// Taille du champ
>
// Est ce qu’il peut être nul
>

// Autre Champ
(appuyer sur entrer si c’est terminer)
(est si jamais c’est fait par erreur vous pouvez faire “ php bin/console make:entity Recipe

> slug
>
>
>

> content
> text
>

> createdAt
> datetime_immutable
>

> updateAt
> datetime_immutable
>

php bin/console make:migration
Crée un fichier .php dans le dossier Migrations

php bin/console doctrine:migrations:migrate
OU
php bin/console migrate
> yes
—--
> yes

Exemple

Ajout d'un nouveaux champs dans l'entité Recipe puis faire une migration

php bin/console make:entity Recipe
> duration
> integer
> yes

Vas ajouter le nouvelle champ

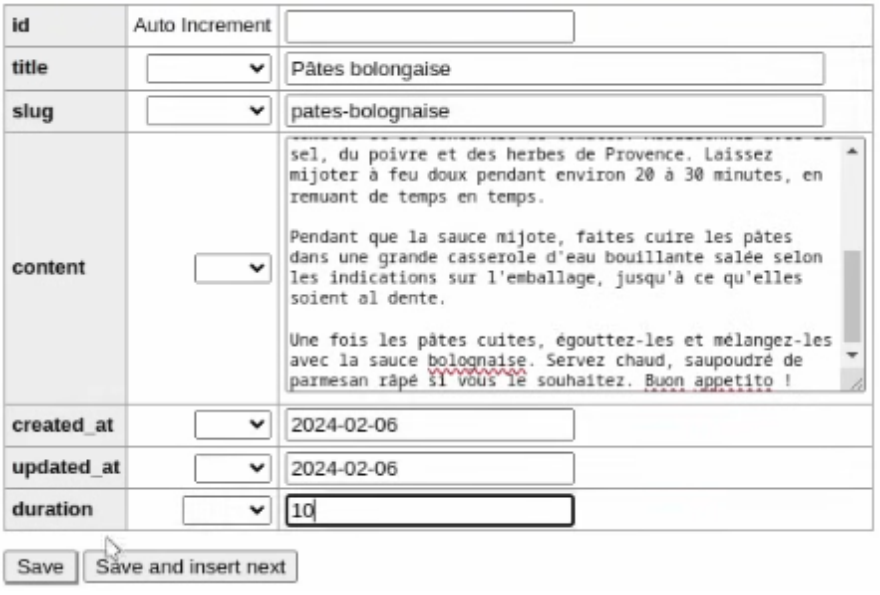
php bin/console make:migration

Crée un nouveau .php dnas /migration

php bin/console doctrine:migrations:migrate
> yes

Base de donnée

Exemple :



src/controller/RecipeController.php

```
Public function index(request $request, RecipeRepository $repository): Response .
{
        $recipe = $repository -> findAll();
        Return $this -> render ('recipe/index.html.twig' [
                'Recipes' => $recipes
        )
}
```

recipe/index.html.twig

```
{% for recipe in recipes %}
<li>
        <a href="{{ path('/recettes/{slug}-{id}', {id: 32, slug: "pate-bolognaise"}) }}">  {{ recipe.getTitle()
        }}</a>
</li>
{% endfor %}
OU

{% for recipe in recipes %}
<li>
        <a href="{{ path('/recettes/{slug}-{id}', {id: recipe.id, slug:recipe.slug}) }}">  {{ recipe.title }}</a>
</li>
{% endfor %}
```

src/controller/RecipeController.php

```
        #[Route('/recette/{slug}-{id}', name: 'recipe.show', requirements: ['id' => '\d+', 'slug' => '[a-z0-9-]+'
        ])]
        Public function show(Request $request, string $slug, int $id, RecipeRepository $repository ):
        Response
        {
                $recipe = $ repository -> find($id);
                OU
                $recipe = $ repository -> findOneBy(['slug' => $slug])

                dd($recipe);

                +if ($recipe->getSlug() != $slug) {
                        return $this->redirectToRoute('recipe.show', ['slug' => $recipe->getSlug(), 'id' =>
                        $recipe->getId()]);
                }

                Return $this->render('recipe/show.html.twig', [
                        +'recipe' => $recipe
                ]);
        }
```

show.html.twig

```
{% extends 'base' %}

{% block title "Recette : " ~ recipe.title %}

{% block body %}
        <h1> {{ recipe.title }} </h1>
        <p>
                {{ recipe.content | nl2br }}
```

```
        </p>
{% endblock %}
```

RecipeRepository.php

```
Public function findWithDurationLowerThan(int $duration): array
{
        Return $this->createQueryBuilder('r')
                ->where('r.duration <= :duration')
                ->orderBy('r.duration', 'ASC')
                ->setMaxResults(10)
                ->setParameter('duration', $duration)
                ->getQuery()
                ->getResult();
}

Autre Exemple

Public function findTotalDuration():int
{
        Return $this->createQueryBuilder('r')
                ->select('SUM(r.duration) as total')
                ->getQuery()
                ->getSingleScalarResult();
}
```

Methode

```
Where()
Order
setMaxResults    //limite
leftjoin
…
```

Erreur sur un entity

RecipeController.php

```
#[Route('/recettes', name: 'recipe.index')]
Public function index(Request $request, RecipeRepository $repository, EntityManagerInterface $em):
Response
{
        $recipes = $repository->findWithDurationLowerThan(20);
        $recipes[0]->setTitle('Pâtes bolognaise');
        $em->flush();
        27:40   composer remove symfony/ux-turbo
        Return $this->render('recipe/index.html.twig', [
                'Recipes' => $recipes
        ]);
}
```

Nouvelle object avec controller

RecipeController.php

```
#[Route('/recettes', name: 'recipe.index')]
Public function index(Request $request, RecipeRepository $repository, EntityManagerInterface $em):
Response
{
	$recipes = $repository->findWithDurationLowerThan(20);

	$recipe = new Recipe();
	$recipe->setTitle('Barbe à papa')
		->setSlug('barbe-papa')
		->setContent('Mettez du sucre')
		->setDuration(2)
		->setCreatedAt(new \DateTimeImmutable() )
		->setUpdatedAt(new \DateTimeImmutable() );

	$em->persist($recipe);
	$em->flush();

	Return $this->render('recipe/index.html.twig', [
		'Recipes' => $recipes
	]);
}
```

Supprimer un objet

RecipeController.php

```
#[Route('/recettes', name: 'recipe.index')]
Public function index(Request $request, RecipeRepository $repository, EntityManagerInterface $em):
Response
{
	$recipes = $repository->findWithDurationLowerThan(20);

	$em->remove($recipes[0]);
	$em->flush();

	Return $this->render('recipe/index.html.twig', [
		'Recipes' => $recipes
	]);
}
```

Recupérer un repository

RecipeController.php

```
#[Route('/recettes', name: 'recipe.index')]
Public function index(Request $request, RecipeRepository $repository, EntityManagerInterface $em):
Response
{
	dd($em->getRepository(Recipe::class));
	$recipes = $repository->findWithDurationLowerThan(20);

	Return $this->render('recipe/index.html.twig', [
		'Recipes' => $recipes
	]);
```

```
}
```

OU

```

#[Route('/recettes', name: 'recipe.index')]
Public function index(Request $request, EntityManagerInterface $em): Response
{
	$recipes = $em->getRepository(Recipe::class)->findWithDurationLowerThan(20);

	Return $this->render('recipe/index.html.twig', [
		'Recipes' => $recipes
	]);
}
```

Les formulaires

```
php bin/console make:form
> RecipeType
> Recipe
```

RecipeController.php

```
#[route('/recettes/{id}/edit', name: 'recipe.edit',
Public function edit(int $id) {
	dd($recipe);
	Return $this->render('recipe/edit.html.twig', [
		'recipe' => $recipe
	]);
}
```

```
Public function edit(Recipe $recipe)
3:02
```

```
Creation d'un fichier dans template/recipe/show.html.twig
```

RecipeController.php

```
#[route('/recettes/{id}/edit', name: 'recipe.edit',
Public function edit(int $id) {
	dd($recipe);
	+$form = $this->createForm(RecipeType::class, $recipe)
	Return $this->render('recipe/edit.html.twig', [
		'recipe' => $recipe
		+'form' => $form
	]);
}
```

edit.html.twig

```
{% extends 'base' %}

{% block title "Recette : " ~ recipe.title %}
```

```
{% block body %}
        <h1> {{ recipe.title }} </h1>

        {{ form(form) }}

{% endblock %}
```

```
                OU
{% block body %}
        <h1> {{ recipe.title }} </h1>

        {{ form_start(form) }}
                <div class="d-flex">
                        {{ form_row(form.title) }}
                        {{ form_row(form.duration) }}
                </div>
        {{ form_rest(form) }}     // bonus
        {{ form_end(form) }}
{% endblock %}
```

config/packages/twig.yaml

```
twig:
        file_name_pattern: '*.twig'
        +form_themes:  ['bootstrap_5_layout.html.twig']

when@test:
        twig:
                strict_variables: true
```

RecipeType.php

```
Class …
{
        Public function …
        {
                $builder
                        ->add('title')
                        ->add('slug')
                        ->add('content')
                        ->add('createdAt')
                        ->add('updatedAt')
                        ->add('duration')
                        + ->add('save', SubmitType::class)
                                OU
                        + ->add('save', SubmitType::class, [
                                'label' => 'Envoyer'
                        ])
        }
}
```

Docs
symfony.com/doc/current/reference/forms/types.html

Pour enregistrer la requête

RecipeController.php

```
#[route('/recettes/{id}/edit', name: 'recipe.edit',
Public function edit(Recipe $recipe, Request $request, EntityManagerInterface $em) {
        dd($recipe);
        $form = $this->createForm(RecipeType::class, $recipe)
        +$form->handleRequest($request);
        If ($form->isSubmitted() && $form->isValid()) {
                $em->flush();
                $this->addFlash('success', 'La recette a bien été modifiée');
                Return $this->redirectToRoute('recipe.index');
        }
        Return $this->render('recipe/edit.html.twig', [
                'recipe' => $recipe
                +'form' => $form
        ]);
}
```

base.html.twig

On peut ajouter
{{ include 'partials/flash.html.twig' %}
{{ dump(app.flashes) }}

Creation du dossier templates/partials
Puis un fichier templates/partials/flash.html.twig

flash.html.twig

```
{% for type, messages in app.flashes %}
<div class="alert alert-{{ type }}">
        {{ message | join('. ') }}
</div>
{% endfor %}
```

index.html.twig

```
{% block title "Toutes les recettes" %}

{% block body %}
<p>
  blabla
</p>
<table class="table">
    <thead>
        <tr>
            <th>Titre</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        {% for recipe in recipes %}
        <tr>
            <td>
```

```
            <a href="{{ path('/recettes/{id}-{slug}', {id: recipe.id, slug: recipe.slug}) }}">{{ recipe.title }}</a>
        </td>
        <td>
            <a class=”btn btn-primary” href="{{ path('recipe.edit', {id: recipe.id}) }}">Editer</a>
        </td>
    </tr>
    {% endfor %}
  </tbody>
</table>
{% endblock %}
```

19:06

Valider les données

Comprendre les services

TP : Formulaire de contact