



Atelier Framework coté serveur

Au : 2024-2025

Classe : L2DSI

Enseignante : Nidhal
Cherif



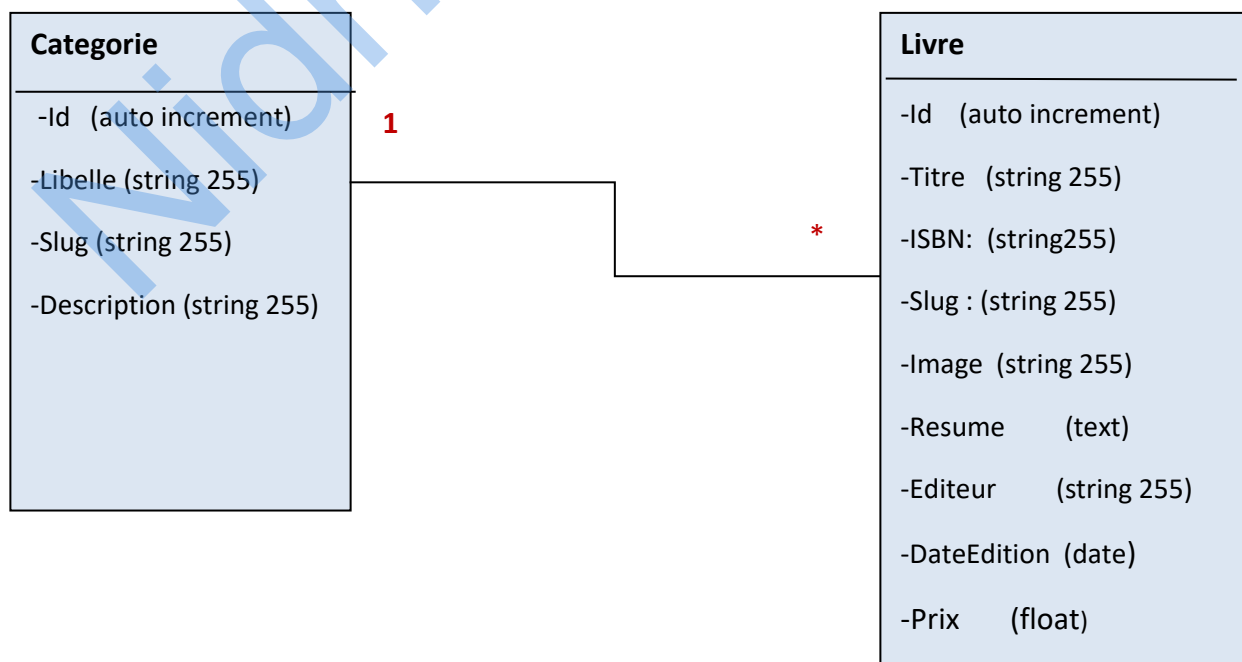
Atelier N°4

Gestion de la base de données avec Doctrine

Objectifs :

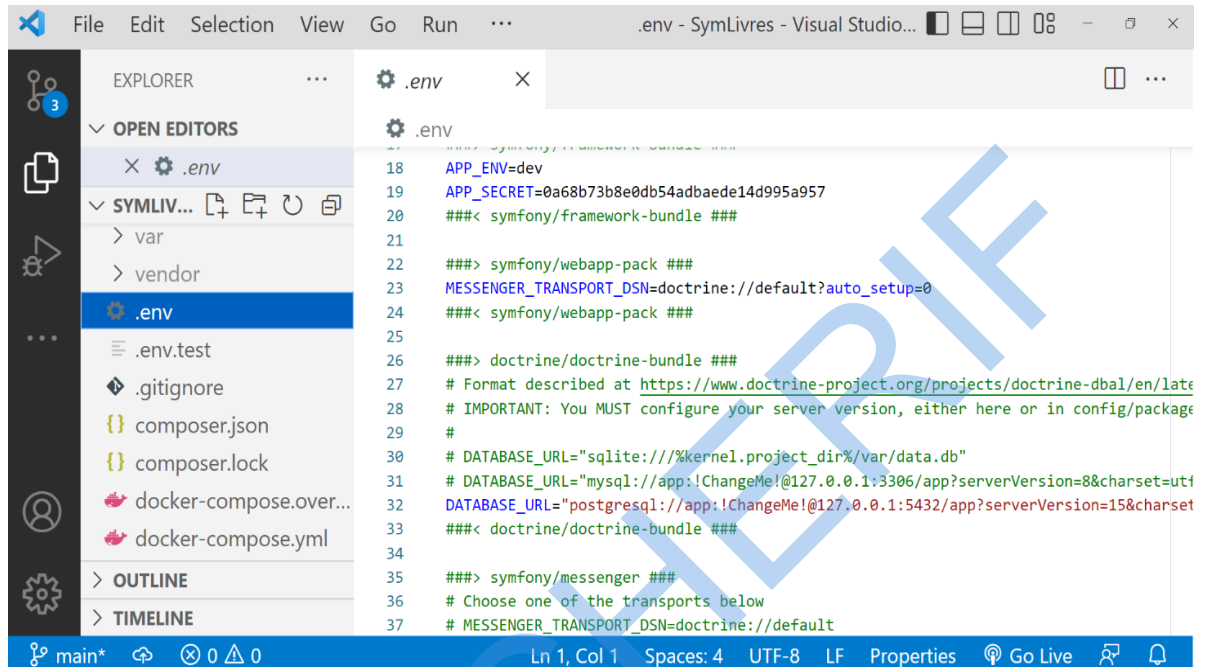
- Générer les entités
- Générer la Base de données
- Interroger la base de données avec doctrine
- Mettre à jour la base de données
- Utiliser de faker et de fixture pour remplir la BD avec de fausses données
- Gérer la relation (ManyToOne et OneToMany) entre les entités
- Mettre en place un système de pagination avec KnPaginator

- On considère l'application « **SymLivres** » permettant de gérer des livres dans une base de données . Chaque livre possède un id(identifiant numérique) un titre et une date d'Edition ,un éditeur, une image(page de garde),et un prix
- Un livre appartient à une seule catégorie (informatique, Médecine, Sciences, Histoire,..)
- Une catégorie peut concerner plusieurs livres
- Et voici le Diagramme de classe de ce Tp :



1. Les fichiers : .env et .env.local

Toutes les variables d'environnement sont définies dans un seul fichier **.env** à la racine du projet **SymLivres** :

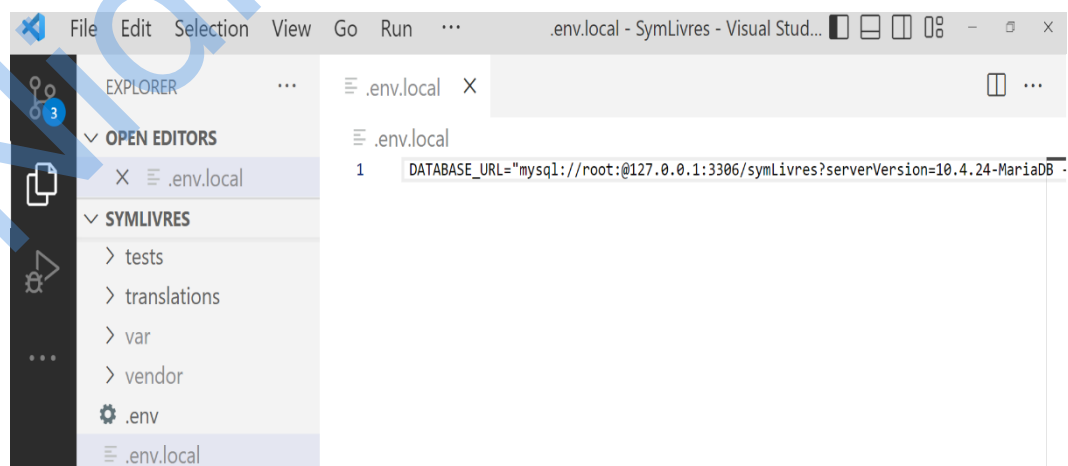


The screenshot shows the Visual Studio Code interface with the **.env** file open in the editor. The Explorer sidebar on the left shows the project structure with **SYMLIVRES** expanded, and **.env** selected under the **SYMLIVRES** folder. The editor displays the following content:

```
.env
18 APP_ENV=dev
19 APP_SECRET=0a68b73b8e0db54adbaede14d995a957
20 ###< symfony/framework-bundle ###
21
22 ###> symfony/webapp-pack ###
23 MESSENGER_TRANSPORT_DSN=doctrine://default?auto_setup=0
24 ###< symfony/webapp-pack ###
25
26 ###> doctrine/doctrine-bundle ###
27 # Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest
28 # IMPORTANT: You MUST configure your server version, either here or in config/packages
29 #
30 # DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
31 # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8&charset=utf8mb4"
32 DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
33 ###< doctrine/doctrine-bundle ###
34
35 ###> symfony/messenger ###
36 # Choose one of the transports below
37 # MESSENGER_TRANSPORT_DSN=doctrine://default
```

Ne mettez pas des informations confidentielles sur le fichier **.env** mais plutôt créez un autre fichier **.env.local** qui sera enregistré en local sur votre machine. et là vous pouvez mettre les informations sensibles comme le mot de passe de serveur Base de données utilisé.

- Copier la ligne 31 de fichier **.env** et coller la dans le fichier **.env.local**
- Décommenter cette ligne et indiquer les informations suivantes : login, mot de passe de votre serveur MySQL, le nom de la base de données de l'application SymLivres et la version de serveur BD.



The screenshot shows the Visual Studio Code interface with the **.env.local** file open in the editor. The Explorer sidebar on the left shows the project structure with **SYMLIVRES** expanded, and **.env.local** selected under the **SYMLIVRES** folder. The editor displays the following content:

```
.env.local
1 DATABASE_URL="mysql://root:@127.0.0.1:3306/symLivres?serverVersion=10.4.24-MariaDB"
```

2. Création de la base de données

- Ouvrir le Terminal et écrire la commande suivante permettant de créer la base de données SymLivres configurée dans le fichier .env.local

```
SymLivres>php bin/console doctrine:database:create
```

3. Création de l'entité Livre

- Ouvrir la console et taper la commande :

```
SymLivres >php bin/console make:entity
```

- Indiquer le nom de l'entité :Livres et répondre aux différentes questions comme suit :

```
New property name (press <return> to stop adding fields):
> titre
Field type (enter ? to see all types) [string]:
>
Field length [255]:
>
Can this field be null in the database (nullable) (yes/no) [no]:
>
Add another property? Enter the property name (or press <return> to stop adding fields):
> image
Field type (enter ? to see all types) [string]:
>
Field length [255]:
>
Can this field be null in the database (nullable) (yes/no) [no]:
>
Add another property? Enter the property name (or press <return> to stop adding fields):
> resume
Field type (enter ? to see all types) [string]:
> text
Can this field be null in the database (nullable) (yes/no) [no]:
> yes
Add another property? Enter the property name (or press <return> to stop adding fields):
> editeur
Field type (enter ? to see all types) [string]:
>
Field length [255]:
>
Can this field be null in the database (nullable) (yes/no) [no]:
>
Add another property? Enter the property name (or press <return> to stop adding fields):
> dateEdition
Field type (enter ? to see all types) [string]:
> date
Can this field be null in the database (nullable) (yes/no) [no]:
```

```

>
Add another property? Enter the property name (or press <return> to stop adding fields):
> prix
Field type (enter ? to see all types) [string]:
> float
Can this field be null in the database (nullable) (yes/no) [no]:
>
Add another property? Enter the property name (or press <return> to stop adding fields):
>

```

Si tout s'est bien passé, vous devriez avoir une classe « Livres.php » sous le dossier Entity avec la structure suivante

⚠ *Attention ! Les commentaires du code sont très importants, ce sont eux qui indiquent à Doctrine2 quel est le type des éléments*

Chaque attribut doit être précédé de l'annotation `@ORM\Column` indiquant son type, son nom et sa taille maximale pour la base de données.

⚠ *Nous pouvons également créer un constructeur pour définir des valeurs par défaut*

4. Générer la table Livres dans la base de données

Nous allons maintenant créer la table correspondante à la classe livre dans la base de données. Pour cela on doit créer un fichier migration versionné sous le répertoire **Migration**. C'est un fichier de différenciation entre les classes existantes dans notre application Symfony et les tables existantes dans la BD. Et ce ci afin que la base de données reflète ce qui existe réellement dans l'application

- Générer le fichier migration , avec la commande console suivante :

```
SymLivres>php bin/console make:migration
```

- Écrire la commande suivante pour faire tourner toutes les migrations existantes

```
SymLivres>php bin/console doctrine:migrations:migrate
```

- Aller voir le résultat dans phpmyadmin, que remarquez vous?

5. Manipulation des entités

Les entités ne sont que de simples classes liées à la BD via l'ORM Doctrine. Pour créer une nouvelle entrée, il suffit de créer un objet de l'entité et de le "persister".

Pour manipuler les entités (les enregistrer (persister) et les récupérer), nous aurons besoin des services : *Doctrine* , *EntityManager (em)* et *Repository*.

5.1. Récupérer les données de la base de données :

Le fichier **LivresRepository.php** associé à l'entité **Livres**, va nous permettre de stocker et séparer les appels en BD, la création des requêtes dbal/dql. Toutes les méthodes présentes dans la classe **LivresRepository**

Ouvrir le fichier LivreRepository.php vous verrez qu'elle est dotée de plusieurs méthodes de base:

- **find(\$id):** Pour récupérer une entité par sa clé primaire / id

- **findAll():** Pour récupérer toutes les entités sans distinction
- **findBy(\$criteria, \$orderBy, \$limit, \$offset):** Pour récupérer les entités suivant une liste de critères
- **findOneBy(\$criteria):** Récupérer l'entité qui répond aux critères donnés

- Insérer dans votre table Livres les 3 enregistrements suivants (via phpMyadmin)

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Options supplémentaires

		id	titre	image	resume	editeur	date_edition	prix
<input type="checkbox"/>	Éditer Copier Supprimer	1	Titre1	https://via.placeholder.com/150	Resumé de livre Titre1	Editeur1	2022-01-30	100
<input type="checkbox"/>	Éditer Copier Supprimer	2	Titre2	https://via.placeholder.com/150	Resumé de livre Titre3	Editeur 3	2023-01-01	65
<input type="checkbox"/>	Éditer Copier Supprimer	3	Titre3	https://via.placeholder.com/150	Resumé de livre Titre3	Editeur 3	2021-01-03	125

- Créer un contrôleur : LivreController.php

5.2. Injection de dépendances dans Symfony :

5.2.1. Qu'est-ce qu'une dépendance ?

Pour lister les livres, symfony a besoin d'un Repository des livres pour pouvoir fonctionner, Quand une fonction a besoin de quelque chose pour fonctionner on appelle ça une dépendance. Symfony peut dans ce cas nous fournir cette dépendance.

Pour lister les livres , On demande à Symfony de nous injecter une instance de la classe **LivreRepository** dans l'entête de la fonction et on aura cette solution avec l'injection de dépendance :

```
public function listAll(LivresRepository $rep)
{
    $livres=$rep->findAll();
    return $this->render('livres/listAll.html.twig',['livres'=>$livres]);
}
```

- Créer cette action **listAll()** afin qu'elle récupère tous livres existants avec une route : **admin/livres** et qui retourne la réponse au template **listAll.html.twig**
- Créer le template listAll.html.twig afin qu'il affiche tous les livres existants dans la BD sous forme d'une table HTML en appliquant bootstrap et avec les liens : Voir détail, supprimer et Edit et qui sont inactifs en ce moment

La liste des livres

Identifiant	Titre	Prix	Edition	Date Edition	Action
1	Titre1	100	Editeur1	30/01/2022	Voir détail... Supprimer Edit
2	Titre2	65	Editeur 3	01/01/2023	Voir détail... Supprimer Edit
3	Titre3	125	Editeur 3	03/01/2021	Voir détail... Supprimer Edit

- Créer une action Rechercher(\$id) permettant de retourner les données d'un Livre dont l'id est passé en paramètre et ceci lorsque la requête : localhost :8000/livres/find/{id} est lancée
- Créer le template rechercher.html.twig afin qu'il affiche le résultat suivant :

← → ↺ ⌂ <http://127.0.0.1:8000/livres/find/1>

SymLivres Accueil Nos Livres Contactez nous

Détail du livre : Titre1



Prix: 100DT
Résumé:Resumé de livre Titre1
Editeur :Editeur1
Date Edition :30/01/2022

5.2.2. Le ParamConverter

On peut aller plus loin avec symfony et le système d'injection dans la fonction **rechercher (\$id)** .

Au lieu de ce code :

```
#[Route('/livres/find/{id}', name: 'app_livres_find')]
public function rechercher($id, LivreRepository $rep) : Response
{
    $livre=$rep->find($id);
    return $this->render('livre/rechercher.html.twig',['livre'=>$livre]);
}
```

On peut écrire :

```
#[Route('/livres/find/{id}', name: 'app_livres_find')]
public function rechercher(Livres $livre): Response
{
    return $this->render('livres/rechercher.html.twig', [
        'livre' => $livre,
    ]);
}
```

grâce à une brique logicielle existante **ParamConverter**, qui converti un paramètre de la requête en un objet. Lorsque Symfony trouve dans la route un identifiant id et dans le paramètre de la fonction, un objet de la classe, il fait une recherche par id et retourne l'objet en question

- Tester ce code

- Activer le lien Voir détail... de la table HTML, liste des produits afin qu'il génère une route vers l'action rechercher

5.3. Insertion d'un livre dans la base de données

L'entityManager : est l'objet permettant d'effectuer des opérations liées à l'altération des données, à savoir les requêtes de type INSERT, UPDATE et DELETE

- Créer dans le contrôleur, une action « Ajouter » qui permet d'ajouter un livre dans la base de données,

```
#[Route('/admin/livres/add', name: 'admin_livres_add')]
public function ajouter(EntityManagerInterface $em): Response
{
    $livre = new Livres();
    $livre->setTitre('Titre du livre 4');
    $livre->setSlug('Titre-livre-4');
    $livre->setISBN('978-4545-4668-667-5');
    $livre->setImage('https://via.placeholder.com/150');
    $livre->setResume('Résumé du livre Titre4');
    $livre->setPrix(20);
    $livre->setEditeur('Editeur4');
    $dateEdition = new \DateTime('2023-01-01');
    $livre->setDateEdition($dateEdition);
    $em->persist($livre);
    $em->flush();

    return new Response("Le livre est enregistré avec succès");
}
```

✎ Dans la dernière ligne le contrôleur est chargé de retourner une réponse HTTP sous la forme d'un objet *Symfony\Component\HttpFoundation\Response* (ici on a pas besoin de twig puisque il s'agit d'un message simple)

- Modifier la réponse du code précédent, en faisant une redirection avec le fonction `redirectToRoute()`, vers page qui liste tous les livres.

5.4. Suppression d'un livre de la base de données

- Créer dans le contrôleur **LivreController**, une action « Supprimer » qui à partir de son identifiant supprime un livre dans la base de données.
- Faire une redirection vers la page affichant les livres

```
#[Route('/admin/livres/delete/{id}', name: 'admin_livres_delete')]
public function supprimer(Livres $livre, EntityManagerInterface $em):
Response
{
    $em->remove($livre);
    $em->flush();
    return $this->redirectToRoute('admin_livres');
}
```

- Activer le lien : supprimer de table HTML : liste des produits afin qu'il génère une route vers l'action Supprimer

5.5. Modification des données d'un livre

- Créer dans le contrôleur **LivreController**, une action « Editier » permettant de son d'augmenter le prix de 10 dt .
- Faire une redirection vers la page affichant les livres.
- Activer le lien Editier de table HTML : liste des produits afin qu'il génère une route vers l'action Editier

5.6. Utiliser les fixtures

Une fixture c'est un script qui nous permet d'insérer un jeu de fausses données dans la base

- Installer le composant fixtures
- ```
>composer require orm-fixtures --dev
```
- Créer la fixture LivreFixtures.php qui nous permet de créer de faux livres
- ```
>php bin/console make:fixtures LivresFixtures
```
- Ouvrir le fichier **Datafixtures/LivreFixtures.php** et Ajouter 50 livres comme suit :

```
class LivresFixtures extends Fixture
{
    public function load(ObjectManager $manager): void
    {
        for ($i = 1; $i <= 50; $i++) {
            $livre = new Livres();
            $livre->setTitre('Titre ' . $i);
            $livre->setImage('https://picsum.photos/200');
            $livre->setResume('Résumé du livre Titre ' . $i);
            $livre->setPrix(mt_rand(10, 200));
            $livre->setEditeur('Editeur ' . $i);
            $dateEdition = new \DateTime('2023-01-01');
            $livre->setDateEdition($dateEdition);
            $manager->persist($livre);
        }
        $manager->flush();
    }
}
```

-Taper la commande suivante pour sauvgarder les 50 livres dans la BD

```
>php bin/console doctrine:fixtures:load
```

Ou

```
>php bin/console d:f:l
```


5.7. Créer des jeux de fausses données avec les fixtures

Faker est une bibliothèque PHP qui génère de fausses données un peu plus convaincantes,

- Installer la bibliothèque Faker avec la commande suivante :

```
> composer require fakerphp/faker
```

✎ Pour plus de détail, visitez le site : <https://packagist.org/packages/fakerphp/faker>

- Maintenant modifier la fonction load de fichier LivresFixtures.php comme suit :

```
use Faker\Factory ;

.....

class LivresFixtures extends Fixture
{
    public function load(ObjectManager $manager): void
    {
        $faker = Factory::create('fr_FR');

        for ($i = 1; $i <= 50; $i++) {
            $livre = new Livres();
            $livre->setTitre($faker->name);
            $livre->setImage('https://picsum.photos/200');
            $livre->setResume($faker->text);
            $livre->setPrix($faker->numberBetween(10, 200));
            $livre->setEditeur($faker->company);
            $livre->setDateEdition(new \DateTime($faker->date()));
            $manager->persist($livre);
        }
        $manager->flush();
    }
}
```

- Implémenter dans la base ce jeu de données

6. Ajout d'une Entity Categorie et création d'une relation ManyToOne

- Modifier l'entité **Livres** en ajoutant une propriété nommée : categorie de type relation
- Indiquer une relation avec l'entité **categorie** de Type : ManyToOne
- La propriété Catégorie peut être NULL
- Accepter la création d'une collection nommé **Livres** dans l'entité categories afin de faciliter les traitements
- Générer le fichier migration (php bin/console make:migration)
- Faire tourner toutes les migration existantes.
- Remplir les deux tables Livres et categorie par des fausses données en utilisant les fixtures et le faker

```

public function load(ObjectManager $manager): void
{
    $faker = Factory::create('fr_FR');
    for ($j = 1; $j <= 3; $j++) {
        $cat = new Categorie();
        $cat->setLibelle($faker->name);
        $cat->setDescription($faker->text) ;
        $manager->persist($cat);
        for ($i = 1; $i <= 15; $i++) {
            $livre = new Livres();
            $livre->setTitre($faker->name);
            $livre->setImage('https://picsum.photos/200');
            $livre->setResume($faker->text);
            $livre->setPrix($faker->numberBetween(10, 200));
            $livre->setEditeur($faker->company);
            $livre->setDateEdition(new \DateTime($faker->date()));
            $livre->setCategorie($cat);
            $manager->persist($livre);
        }
    }
    $manager->flush();
}

```

- Modifier le template **listAll.html.twig** afin d'obtenir une nouvelle Catégorie dans la table HTML
- On veut rendre la table affichant tous les livres dynamique avec une pagination et des filtres en appliquant le **datatable**

7. Mettre en place un système de pagination

Nous souhaitons ajouter une pagination au niveau de l'affichage des livres ,pour cela on va intégrer le **bundle KnpPaginator** :

SymLivres Accueil Nos Livres Contactez nous								
La liste des livres								
Identifiant	Titre	Prix	Edition	Date Edition	Catégorie	Action		
200	Isaac Marion	194	Bonneau S.A.S.	11/06/1990	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
201	Michel Mace	122	Turpin	23/06/2002	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
202	Marcel Langlois	32	Salmon SA	04/08/2016	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
203	Olivia Imbert	142	Regnier et Fils	19/07/2005	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
204	Olivier Da Silva	24	Mathieu S.A.R.L.	13/08/1985	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
205	Pauline Lelievre	103	Guyon	16/06/2018	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
206	Eugène Sauvage	187	Gonzalez	06/07/1990	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
207	Audrey Le Rodriguez	19	Duval Millet SAS	01/12/2022	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
208	Sophie Robin	200	Perret et Fils	02/05/1972	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
209	Colette Le Girard	89	Legendre S.A.R.L.	28/01/2001	Nath Lambert-Chauvet	Voir détail...	Supprimer	Edit
« Previous 1 2 3 4 5 Next »								

7.1. Installation de KnpPaginator

Pour installer ce bundle, consultez ce site : <https://github.com/KnpLabs/KnpPaginatorBundle>

- Installer le KnpPaginator avec composer

```
composer require knplabs/knp-paginator-bundle
```

7.2. Activation de pagination dans le contrôleur

Pour activer la pagination dans votre contrôleur, vous devez comprendre le concept de KNP Paginator. Vous devez d'abord obtenir toutes les données que vous souhaitez paginer. Dans cet exemple, ce sont tous les livres. KNP Paginator "filtre" cette liste et ne conserve que les données correspondant à la page en cours. Utilisez la méthode "paginate" qui gère tout cela. Cette méthode prend trois paramètres, comme illustré dans l'exemple ci-dessous :

```
use Knp\Component\Pager\PaginatorInterface;
use Symfony\Component\HttpFoundation\Request;
.....
#[Route('/admin/livres', name: 'admin_livres')]
public function listAll(LivresRepository $rep, PaginatorInterface
$paginator, Request $request): Response
{
    $livres = $paginator->paginate(
        $rep->findAll(),
        $request->query->getInt('page', 1), // Numéro de la
        //page en cours, passé dans l'URL, 1 si aucune page
        10 // 3eme param 10, c'est le Nombre de résultats par page 10
    );
    return $this->render('livres/listAll.html.twig', [
        'livres' => $livres,
    ]);
}
```

7.3. Activation de pagination dans le template

Une fois les données envoyées à la vue, vous pouvez afficher la pagination en ajoutant la de la méthode **knp_pagination_render** dans le fichier **listAll.html.twig** comme suit :

```
{% extends 'base.html.twig' %}
{% block title %}Hello LivresController!
{% endblock %}
{% block body %}
    <table class=" table table-active">
        .....
        <h1>La liste des livres</h1>
        {% for book in livres %}
            <tr>
                <td>{{book.id}}</td>
                .....
            </tr>
        {% endfor %}
    </table>
    <div class="navigation">
        {{ knp_pagination_render(livres) }}
    </div>
{% endblock %}
```

On obtient :

La liste des livres

Identifiant	Titre	Prix	Edition	Date Edition	Catégorie	Action
155	Thomas-Roland Laurent	195	Aubert Guillou SA	31/10/1981	Georges du Carpentier	Voir détail... Supprimer Edit
156	Jules-Tristan Dijoux	92	Duhamel Meunier S.A.	13/10/1978	Georges du Carpentier	Voir détail... Supprimer Edit
157	Zoé Meyer-Normand	129	Masse	11/07/1987	Georges du Carpentier	Voir détail... Supprimer Edit
158	Marianne Le Gall	172	Bailly	06/01/1988	Georges du Carpentier	Voir détail... Supprimer Edit
159	Jeannine Roussel	199	Navarro	01/05/2001	Georges du Carpentier	Voir détail... Supprimer Edit
160	Andrée Andre	163	Lecomte	11/10/2010	Georges du Carpentier	Voir détail... Supprimer Edit
161	Sophie de Wagner	176	Barbe	12/02/1971	Georges du Carpentier	Voir détail... Supprimer Edit
162	Alphonse Pichon	29	De Sousa Jourdan S.A.R.L.	07/02/1972	Georges du Carpentier	Voir détail... Supprimer Edit
163	Éric de la Pierre	86	Riviere SARL	20/01/2013	Georges du Carpentier	Voir détail... Supprimer Edit
164	Bertrand Rousseau	17	Georges Rolland SAS	26/12/2007	Georges du Carpentier	Voir détail... Supprimer Edit

12345>>>

- Changer le thème bootstrap de l'affichage de pagination pour l'améliorer. Pour ce faire, créer un fichier de configuration **knp_paginator.yaml** sous le dossier **config/packages** de votre application .
- Ouvrir le site : <https://github.com/KnpLabs/KnpPaginatorBundle> de guide d'installation de bundle knp_paginator et copier/coller la configuration yaml ci-dessous dans le fichier knp_paginator.yaml

← → ↻ ⌂ https://github.com/KnpLabs/KnpPaginatorBundle

🔍 ⭐

☰ README.md

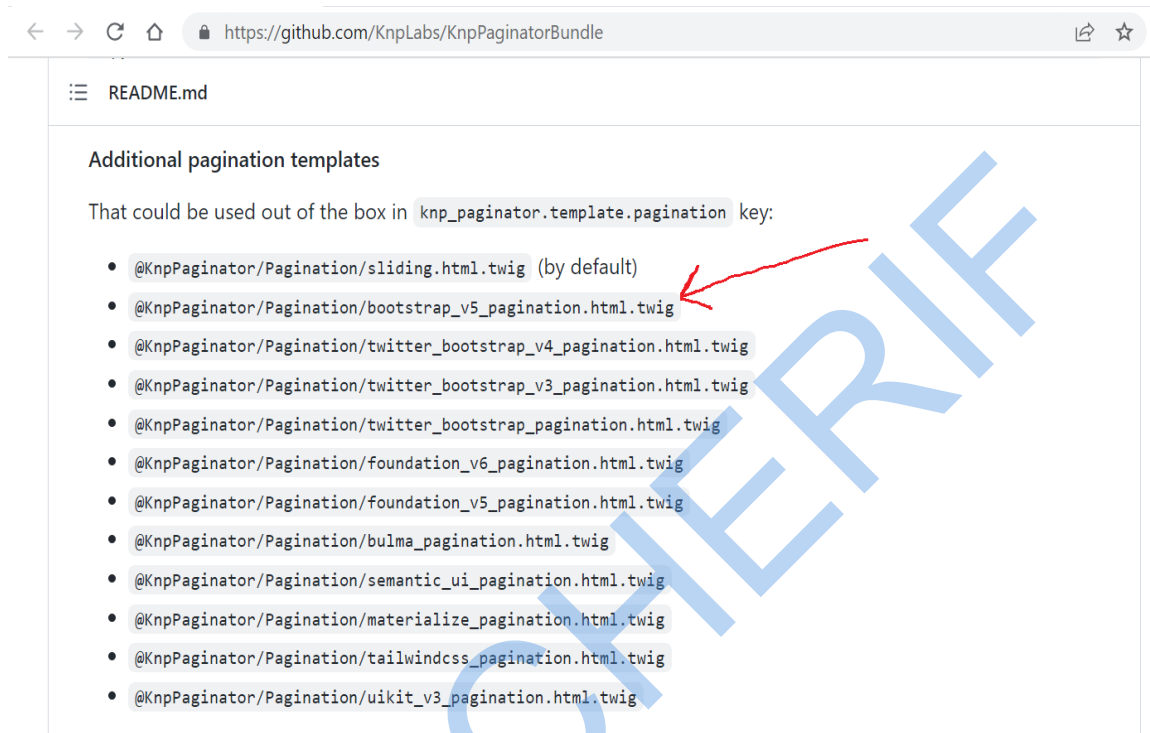
Configuration example

You can configure default query parameter names and templates

YAML:

```
knp_paginator:
  page_range: 5 # number of links shown in the pagination menu (e.g: you have 10 pa
  default_options:
    page_name: page # page query parameter name
    sort_field_name: sort # sort field query parameter name
    sort_direction_name: direction # sort direction query parameter name
    distinct: true # ensure distinct results, useful when ORM queries are using GROUP
    filter_field_name: filterField # filter field query parameter name
    filter_value_name: filterValue # filter value query parameter name
  template:
    pagination: '@KnpPaginator/Pagination/sliding.html.twig' # sliding pagination controls template
    sortable: '@KnpPaginator/Pagination/sortable_link.html.twig' # sort link template
    filtration: '@KnpPaginator/Pagination/filtration.html.twig' # filters template
```

- Choisir un thème bootstrap de cette page (guide d'installation et configuration knp_paginator) et remplacer la valeur par défaut de pagination du fichier knp_paginator.yaml par ce thème :
Exemple de thème choisi :
@KnpPaginator/Pagination/bootstrap_v5_pagination.html.twig



- Faire le test

```
$slug = strtolower(trim(preg_replace('/[^\A-Za-z0-9-]+/', '-', $article->getTitle()), '-'));
```