

Prise en main de Symfony 4.4

Doctrine

Contenu

Objectifs :

Continuer le TD Doctrine Structure : Développer une petite application qui va gérer un les navires.
Versionner le projet
Créer des contrôleurs
Créer et valider des formulaires



Environnement :

- ✓ Serveur apache : celui de wamp
- ✓ Base de données : mysql en local (wamp)
- ✓ IDE : netbeans
- ✓ virtualHost : navire.sio
- ✓ PHP 7.4.x
- ✓ Symfony 4.4
- ✓ Mysql 5.7.xx



Vous allez maintenant développer l'application.

Partie 1 : MISE A JOUR ENTITES PORT ET AISSHIPTYPE

Vous allez inverser le sens des associations entre Port et AisShipType :

```
/**
 * @ORM\ManyToMany(targetEntity=AisShipType::class, inversedBy="lesPorts" )
 * @ORM\JoinTable(
 *     name="porttypecompatible",
 *     joinColumns={@ORM\JoinColumn(name="idport", referencedColumnName="id")},
 *     inverseJoinColumns={@ORM\JoinColumn(name="idaistype", referencedColumnName="id")}
 * )
 */
private $lesTypes;
```

Entité Port

```
/**
 *
 * @ORM\ManyToMany(targetEntity=Port::class, mappedBy="lesTypes" )
 */
private $lesPorts;
```

Entité AisShipType

Puis vous mettez la base de données à jour :

```
php bin/console make:migration
php bin/console doctrine:migration:migrate
```

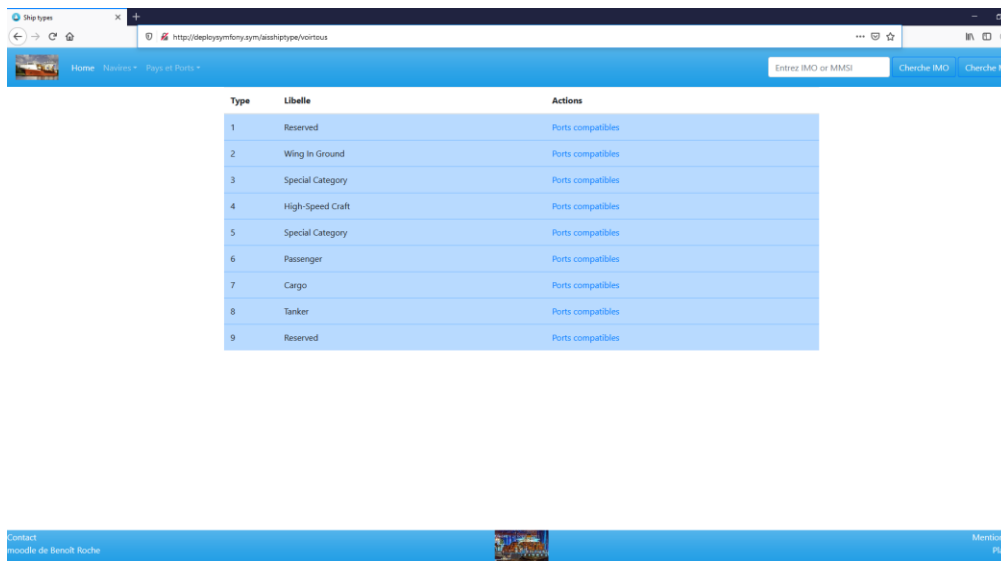
Concrètement, cela ne fera qu'inverser les colonnes de la table de jointure porttypecompatible. Mais l'Entité Port l'entité propriétaire de la relation.

Partie 2 : USER STORIES 1 ET 2

En tant que gestionnaire
je voudrais pouvoir lister les types de Navire
afin de pouvoir visualiser les ports où les navires de ce type sont susceptibles d'être accueillis

Contrôleur : AisShipTypeController
Route : aisshiptype/voirtous

Exemple :



Type	Libelle	Actions
1	Reserved	Ports compatibles
2	Wing In Ground	Ports compatibles
3	Special Category	Ports compatibles
4	High-Speed Craft	Ports compatibles
5	Special Category	Ports compatibles
6	Passenger	Ports compatibles
7	Cargo	Ports compatibles
8	Tanker	Ports compatibles
9	Reserved	Ports compatibles

Vous aurez à :

- ✓ Implémenter la méthode voirTous de la classe contrôleur AisShipTypeController
- ✓ Créer la classe formulaire AisShipTypeType
- ✓ Créer le template aisshiptype\voirtous.html.twig

1. La méthode voirTous de la classe AisShipTypeController

Elle doit s'exécuter sur la route aisshiptype/voirtous dont le nom est aisshiptype_voirtous
Elle va récupérer tous les types de navires et appeler le template aisshiptype/voirtous.html.twig

```
/**
 * @Route("/voirtous", name="voirtous")
 */
public function voirTous(AisShipTypeRepository $repo): Response {
    $types = $repo->findAll();
    return $this->render('aisshiptype/voirtous.html.twig', [
        'types' => $types,
    ]);
}
```

2. la classe formulaire AisShipTypeType

Vous la créez avec la commande `symfony make:form`

Elle sera très simple car susceptible d'être utilisée dans plusieurs environnements :

```
public function buildForm(FormBuilderInterface $builder, array $options) {

    $builder
        ->add('aisShipType', TextType::class)
        ->add('libelle', TextType::class)
    ;
}
```

3. le template aishiptype\voirtous.html.twig

```
{% extends 'base.html.twig' %}
{% block title %}Ship types{% endblock %}

{% block body %}

<div class="container">
    <table class="table table-hover">
        <thead>
            <tr>
                <th scope="col" style="width: 100px;">Type</th>
                <th scope="col">Libelle</th>
                <th scope="col">Actions</th>
            </tr>
        </thead>
        <tbody>
            {% for type in types %}
                <tr class="table-primary">
                    <td>{{ type.aisshiptype }}</td>
                    <td>{{ type.libelle }}</td>
                    <td><a href="{{ path('aishiptype_portscompatibles', {id: type.id}) }}">Ports compatibles</a></td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

En cliquant sur le lien :

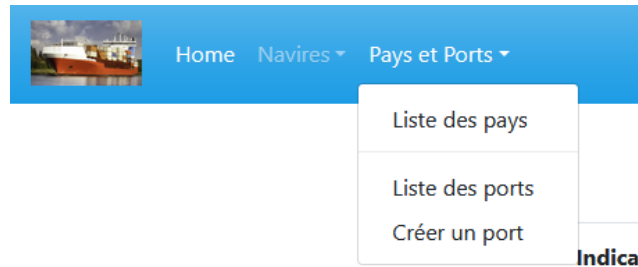
Ports compatibles
Type de navire : 7 Cargo

Indicateur	Nom	Pays	Voie
KSENP	Busan New Port	Corée du Sud	Corée du Sud
CUCCC	Cayo Coco	Cuba	Cuba
GIGIB	Gibraltar	Gibraltar	Gibraltar
TRIST	Istanbul	Turquie	Turquie
SKOP	Koper	Slovénie	Slovénie
FRLEH	Le Havre	France	France
USLDB	Long Beach	Etats-Unis	Etats-Unis
USMIA	Miami	Etats-Unis	Etats-Unis
CHNSA	Nansha	Chine	Chine
USNYC	New York	Etats-Unis	Etats-Unis
CHNGB	Ningbo	Chine	Chine
NGPHC	Port Harcourt	Nigeria	Nigeria
BONAS	Port de NASSAU	Bahamas	Bahamas
ITRAN	Ravenna	Italie	Italie



Liste triée alphabétiquement par twig

En tant que responsable
je voudrais pouvoir créer des ports
afin de pouvoir enregistrer le trafic maritime vers le nouveau port



Vous aurez à créer

- ✓ Le contrôleur PortController
- ✓ Le formulaire portType
- ✓ Le template port/edit.html.twig

L'interface pourrait ressembler à ceci :

<p>Indicatif</p> <div>Indicatif international</div>	<p>Nom du port</p> <div>Saisir le nom du port</div>
<p>Pays</p> <div>Aruba (Ile d')</div>	<p>Types de navires acceptés</p> <div> <input type="checkbox"/> Reserved <input type="checkbox"/> Wing In Ground <input type="checkbox"/> Special Category <input type="checkbox"/> High-Speed Craft <input type="checkbox"/> Special Category <input type="checkbox"/> Passenger <input type="checkbox"/> Cargo <input type="checkbox"/> Tanker <input type="checkbox"/> Petit bateau </div>

Créer



Vous n'aurez rien à faire à ce niveau-là pour saisir les escales et les navires attendus !

Voici à quoi pourrait ressembler le formulaire PortType :

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('nom', TextType::class)
        ->add('indicatif', TextType::class)
        ->add('lePays', EntityType::class, [
            'class' => Pays::class,
            'choice_label' => 'nom',
            'expanded' => false,
            'multiple' => false,
        ])
        ->add('lesTypes', EntityType::class
            , [
                'class' => AisShipType::class,
                'choice_label' => 'libelle',
                'expanded' => true,
                'multiple' => true,
            ]
        );
}
```

Et la classe portController :

```
class PortController extends AbstractController
{
    /**
     * @Route("/creer", name="creer")
     */
    public function creer(Request $request, \Doctrine\ORM\EntityManagerInterface $manager): Response
    {
        $port= new Port();
        //$Paysrepo->findAll()
        $form=$this->createForm(PortType::class, $port);
        $form->handleRequest($request);
        if($form->isSubmitted() && $form->isValid()){
            $manager->persist($port);
            $manager->flush();
            return $this->redirectToRoute('home');
        }
        return $this->render('port/edit.html.twig', [
            'form' => $form->createView(),
        ]);
    }
}
```

Quant au template twig, en voici un extrait :

```
<div class="col-8" id="some-custom-id">
    {{ form_label(form.nom, 'Nom du port') }}
    {{ form_widget(form.nom, {'attr': {'placeholder': 'Saisir le nom d
</div>
</div>

<div class="row">
    <div class="col-4" id="some-custom-id">
        {{ form_label(form.lePays, 'Pays') }}
        {{ form_widget(form.lePays) }}
    </div>
    <div class="col-4" id="some-custom-id">
        {{ form_label(form.lesTypes, 'Types de navires acceptés') }}
        {{ form_widget(form.lesTypes) }}
    </div>
</div>
</br>
<button type='submit' class ='btn btn-success'>Créer</button>
{{ form_end(form) }}
```

Vous allez donc créer le port suivant :

Indicatif	Nom du port
<input type="text" value="ESBCN"/>	<input type="text" value="BARCELONA"/>
Pays	Types de navires acceptés
<input type="text" value="Espagne"/>	<input type="checkbox"/> Reserved <input type="checkbox"/> Wing In Ground <input type="checkbox"/> Special Category <input checked="" type="checkbox"/> High-Speed Craft <input type="checkbox"/> Special Category <input checked="" type="checkbox"/> Passenger <input checked="" type="checkbox"/> Cargo <input checked="" type="checkbox"/> Tanker <input type="checkbox"/> Reserved
<input type="button" value="Créer"/>	

Cliquez sur Créer et revenez sur votre home page.



Voyez maintenant la base de données

25	29	Qingdao	CNQDG
26	54	BARCELONA	ESBCN

Le port a bien été créé avec l'id 26.

Vérifiez maintenant les types de navires acceptés avec la requête suivante :

```
select nav.libelle
from naviresymfo.aisshtype nav inner join naviresymfo.porttypecompatible typenav
on nav.id = typenav.idaistype
where typenav.idport=26;
```

Et vous obtiendrez le résultat suivant :

	libelle
▶	High-Speed Craft
	Passenger
	Cargo
	Tanker



C'est exactement ce qui avait été saisi !!!
Cool !!!...



Il y a quand même un souci : vous avez remarqué que la liste des pays n'est pas triée ... C'est gênant quand on doit choisir UN pays dans autant de pays non triés

Pays
Aruba (Ile d')
Aruba (Ile d')
Angola
Anguilla (Ile d')
Albanie
Aland (Iles d')
Antilles Néerlandaises
Emirats Arabes Unis
Argentine
Kerguelen (Iles)

Pour avoir une liste triée, vous allez donc exploiter tout le potentiel de symfony

L'idée est de récupérer la liste des pays triée sur le nom du pays..... et là on est obligés de passer par une requête DQL juste pour effectuer un tri.....

Vous allez donc modifier :

- ✓ Le repository PaysRepository
- ✓ Le formulaire PortType

Le repository PaysRepository en rajoutant la méthode getPaysTrieSurNom(). On n'a pas le choix, on doit passer par l'API QueryBuider.

```
public function getPaysTrieSurNom()
{
    return $this->createQueryBuilder('p')
        ->orderBy('p.nom', 'ASC');
}
```

Le formulaire PortType : on va faire appel à la méthode getPaysTrieSurNom() de la classe PortRepository pour récupérer les pays triés sur le nom :

```
$builder
    ->add('nom', TextType::class)
    ->add('indicatif', TextType::class)
    ->add('lePays', EntityType::class, [
        'class' => Pays::class,
        'choice_label' => 'nom',
        'expanded' => false,
        'multiple' => false,
        'query_builder' => function(PaysRepository $repo) {
            $lesPaysTries= $repo->getPaysTrieSurNomV2();
            return $lesPaysTries;
        }
    ])
    ->add('lesTypes', EntityType::class
```

Partie 3 : LA BARRE DE RECHERCHE

Lors du dernier sprint, une nouvelle User Story a été créée avec une priorité très haute. Vous êtes chargé de mettre en place cette user story :

En tant que gestionnaire
je voudrais pouvoir rechercher un navire par son numéro IMO ou son indicatif MMSI
afin de pouvoir récupérer ses informations et le localiser sur la carte.

Vous allez donc développer le code de la zone de recherche de la navbar :



Pour cela, vous allez :

- ✓ Créer le contrôleur SearchController
- ✓ Créer le template éléments\searchbar.html.twig
- ✓ L'intégrer dans le template éléments\navbar.html.twig

4. Le contrôleur

Il aura 2 méthodes :

- ✓ La méthode `searchBar()` qui va construire le formulaire de recherche
- ✓ La méthode `handleSearch()` qui va exploiter le retour du formulaire quand l'utilisateur aura cliqué sur l'un des 2 boutons.

```
public function searchBar() {
    $form = $this->createFormBuilder()
        ->setAction($this->generateUrl("search_handlesearch"))
        ->add('cherche', TextType::class)
        ->add('envoiimo', SubmitType::class)
        ->add('envoimmsi', SubmitType::class)
        ->getForm();

    return $this->render('elements/searchbar.html.twig', [
        'formSearch' => $form->createView()
    ]);
}
```

.. il s'agit seulement ici de générer le formulaire et d'appeler le template `éléments\searchbar.html.twig`

La méthode `handleSearch()` sera un contrôleur qui s'exécutera sur la route `/search/handlesearch`

En voici une première version :

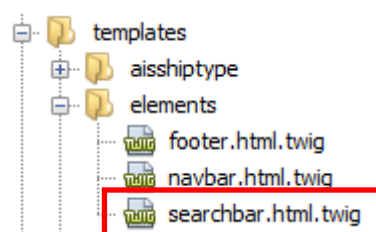
```
/**
 *
 * @Route("/search/handlesearch", name="search_handlesearch")
 */
public function handleSearch(Request $request, NavireRepository $repo): Response {
    $valeur = $request->request->get('form')['cherche'];
    if (isset($request->request->get('form')['envoiimo'])) {

        $critere = "imo Recherché : " . $valeur;
    } else {

        $critere = "mmsi recherché " . $valeur;
    }
    return new Response("<h1> $critere </h1>");
}
```

... rien d'extraordinaire pour vous Vous la complèterez plus tard. Vous y mettrez également du commentaire.

5. le template



Il s'agit d'un simple formulaire :

```
<div class="form-inline my-2">
    {{ form_start(formSearch) }}

    {{ form_widget(formSearch.cherche, {'attr': {'placeholder': 'Entrez IMO or MMSI'}}) }}
    {{ form_widget(formSearch.envoimo, { 'label': 'Cherche IMO' }) }}
    {{ form_widget(formSearch.envoimmsi, { 'label': 'Cherche MMSI' }) }}

    {{ form_end(formSearch) }}
</div>
```

6. L'intégration du formulaire de recherche dans la navbar :

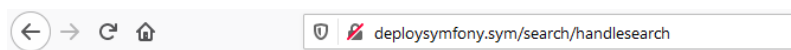
Vous utiliserez la fonction render de twig pour appeler le template de recherche. Vous ajouterez donc ce code à la fin du code de la navbar :

```
</ul>
    {{ render(controller(
        'App\\Controller\\SearchController::searchBar'
    )) }}
</div>
```

Et vous pouvez tester :



Résultat :

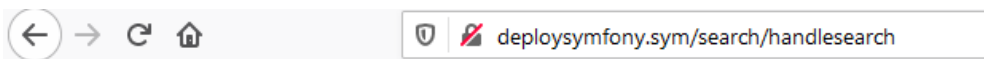


imo Recherché : 9241061

Et sur l'indicatif MMSI :



Et le résultat :



mmsi recherché 310627000

Modifier le code pour afficher le navire trouvé :

Vous serez amené à utiliser les méthodes `findByImo` ou `findByMMSI` pour arriver au résultat.



Il faudra prévoir le cas où le navire n'existe pas !!!

7. Suite des user stories

Les user stories suivantes seront à faire en autonomie, l'interface est libre.

En tant que gestionnaire
je voudrais pouvoir lister les navires
afin de pouvoir modifier certaines caractéristiques.

En tant que gestionnaire,
je voudrais pouvoir visualiser les types de navires susceptibles d'être accueillis dans un port
afin de vérifier les itinéraires des navires

En tant que gestionnaire
Je voudrais connaître la prochaine escale d'un navire
Afin d'informer le port de destination de sa prochaine arrivée.

En tant que gestionnaire

je voudrais connaître l'historique des escales d'un navire
afin de pouvoir effectuer des statistiques sur ses destinations et garder l'historique de ses routes.

Et les nouvelles :

En tant que responsable
Je voudrais connaître le nombre de ports par pays
Afin d'étudier la faisabilité d'une liaison

En tant que responsable
Je voudrais connaître le nombre de ports par pays susceptible d'accueillir un type de navire donné
Afin d'étudier la faisabilité d'une liaison

En tant que responsable
Je voudrais connaître la distance en miles entre 2 ports
Afin de planifier les liaisons maritimes

Partie 4 : SYMFONY C'EST COOL



Un conseil
Voir la commande

make:crud

Vous aurez juste à charger une dépendance et c'est parti !!!

... elle devrait aussi vous rendre des services !!!



Bravo pour votre travail,