Créer manuellement une class ResourceSearch.php

- Ajouter les attributs qu'on veut rendre rechercheable :

```
<?php
namespace App\Entity;
use phpDocumentor\Reflection\DocBlock\Tags\Return_;
class ResourceSearch
{
   /**
    * @var string|null
    */
   private $type;
   /**
    * @var string|null
    */
   private $titre;
   /**
    * @var string|null
    */
   private $person;
   /**
    * @return ResourceSearch
    */
   public function getType() : ResourceSearch
   {
      return $this->type;
   }
   /**
    * @return ResourceSearch
    */
   public function getTitre() : ResourceSearch
   {
      return $this->titre;
   }
   /**
    * @return ResourceSearch
    */
   public function getPerson() : ResourceSearch
   {
      return $this->person;
   }
   /**
    * @param string|NULL $type
    */
   public function setType(string $type)
   {
      $this->type = $type;
   }
   /**
    * @param string|NULL $titre
    */
   public function setTitre(string $titre)
```

```
    {
        $this->titre = $titre;
    }
    /**
     * @param string|NULL $person
     */
    public function setPerson(string $person)
    {
        $this->person = $person;
    }
}
```

Générer le formulaire correspondant

Commande

console/bin make:form

- Il faut rentrer le namespace de l'entity :
  \App\Entity\ResourceSearch
- **Résultat** :

```
class ResourceSearchType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('type', IntegerType::class, [
                'required' => false,
                'label' => false,
                'attr' => [
                    'placeHolder' => 'Type de document'
                ]
            ])
            ->add('titre', IntegerType::class, [
                'required' => false,
                'label' => false,
                'attr' => [
                    'placeHolder' => 'Titre'
                ]
            ])
            ->add('person', IntegerType::class, [
                'required' => false,
                'label' => false,
                'attr' => [
                    'placeHolder' => 'Personne',
                ]
            ])
        ;
    }
    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => ResourceSearch::class,
            // Important : changer la method en 'get' et désactiver le csrf token
```

```
            'method' => 'get',
            'csrf_protection' => 'false'
        ]);
    }
}
```

Traitement et affichage du formulaire

Traitement dans controller ResourceController::index

```
    /**
     * @return Response
     * @route("/ressources" , name="resource.index", options={"utf8": true})
     */

    public function index(PaginatorInterface $paginator, Request $request) : Response
    {
        // Créer une nouvelle
        $search = new ResourceSearch();
        // 1. Création du formulaire
        // 1.1. On crée une nouvelle recherche
        // 1.2. on crée le formulaire
        // 1.2.1. en passant en second paramétre de la méthode createForm l'entity
ResourceSearch
        $form = $this->createForm(ResourceSearchType::class, $search);
        // 1.2.2. On passe en paramètre de la méthode handleRequest  la Request
        $form->handleRequest($request);
        2. Gestion de la requête
            2.1. Executer la requête (en paramètre l'objet ResourceSearch
            $query = $this->repository->findAllVisibleQuery($search);
            2.2. Passer la requête en paramètre de la méthode paginate()
        $resources = $paginator->paginate(
                    $query,
                    $request->query->getInt('page', 1)/*page number*/,
                    12/*limit per page*/
                );

        return $this->render('resource/index.html.twig', [
            'current_page' => 'resources',
            'resources' => $resources,
            // 1.4. On envoie le formulaire à la vue
            'form' => $form->createView()
        ]);
    }
```

Affichage du formulaire dans la vue

```
<div class="jumbotron">
            <div class="container">
                {{ form_start(form) }}
                <div class="form-row align-items-end">
                    <div class="col">
                        {{ form_row(form.titre) }}
                    </div>
                    <div class="col">
```

```
                    {{ form_row(form.auteur) }}
                </div>
                <div class="col">
                    {{ form_row(form.type) }}
                </div>
                <div class="col">
                    <!--  class form-group aligne le bouton  -->
                    <div class="form-group">
                    <button class="btn btn-
                    primary">Rechercher</button>
                    </div>
                </div>
            </div>
        {{ form_end(form) }}
        </div>
    </div>
```

Gestion des paramètres : changer les prefix utilisés dans l'url et les rendre propres

Ajouter la méthode suivante à la classe ResourceSearchType

```
public function getBlockPrefix() {
    return '';
}
```

Modifier la fonction findAllVisibleQuery (passer en paramètre la classe ResourceSearch)

```
findAllVisibleQuery(ResourceSearch $search):
```

Gestion de la validation des champs de recherche

Voir
https://symfony.com/doc/current/reference/constraints