

TP7 Formulaires de Recherche en Symfony 4

On se propose dans cet atelier de créer des formulaires de recherche des articles :

- Recherche des articles par Nom
- Recherche des articles par Catégorie
- Recherche des articles dont le prix est compris entre deux valeurs

Recherche des articles par Nom

1. Créer dans Entity, la classe PropertySearch.php (sans utiliser la commande *php bin/console make:form*)

```
<?php

namespace App\Entity;

class PropertySearch
{
    private $nom;

    public function getNom(): ?string
    {
        return $this->nom;
    }

    public function setNom(string $nom): self
    {
        $this->nom = $nom;

        return $this;
    }
}
```

2. Créer le formulaire PropertySearchType basé sur l'entité PropertySearch
php bin/console make:form

3. Ajouter le code qui suit au début du fichier index.html.twig qui permet de rechercher les articles par nom :

```
{% extends 'base.html.twig' %}
{% block title%} Liste des Articles{% endblock %}

{% block body %}
{{ form_start(form) }}

<div class="form-row align-items-end" >
  <div class="col">
    {{ form_row(form.nom) }}
  </div>

  <div class="col">
    <div class="form-group">
      <button type="submit" class="btn btn-success">Rechercher</button>
    </div>
  </div>
</div>
{{ form_end(form) }}
{% if articles %}
...

```

4. Modifier la fonction **home()** du controlleur IndexController comme suit :

```
/**
 * @Route("/", name="article_list")
 */
public function home(Request $request)
{
    $propertySearch = new PropertySearch();
    $form = $this->createForm(PropertySearchType::class, $propertySearch);
    $form->handleRequest($request);

    //initialement le tableau des articles est vide,
    //c.a.d on affiche les articles que lorsque l'utilisateur
    //clique sur le bouton rechercher
    $articles= [];

    if($form->isSubmitted() && $form->isValid()) {
        //on récupère le nom d'article tapé dans le formulaire
    }
}

```

```

    $nom = $propertySearch->getNom();
    if ($nom!="")
        //si on a fourni un nom d'article on affiche tous les articles ayant ce nom
        $articles= $this->getDoctrine()->getRepository(Article::class)->findBy(['nom' => $nom] );
    else
        //si si aucun nom n'est fourni on affiche tous les articles
        $articles= $this->getDoctrine()->getRepository(Article::class)->findAll();
    }
    return $this->render('articles/index.html.twig',[ 'form' =>$form->createView(), 'articles' => $articles]);
}

```

5. Testez votre travail :

200 POST @ article... 456 ms 6.0 MB 1 1 1 326 in 58.36 ms 2 anon. 57 ms 2 in 1.46 ms 4.4.4

Recherche des articles par Catégorie

6. Créer dans Entity, la classe CategorySearch sans utiliser la commande `make:entity` :

```
<?php
namespace App\Entity;
use Doctrine\ORM\Mapping as ORM;

class CategorySearch
{
    /**
     * @ORM\ManyToOne(targetEntity="App\Entity\Category")
     */
    private $category;

    public function getCategory(): ?Category
    {
        return $this->category;
    }

    public function setCategory(?Category $category): self
    {
        $this->category = $category;

        return $this;
    }
}
```

7. Créer le formulaire CategorySearchType basé sur l'entité CategorySearch

php bin/console make:form

taper **CategorySearchType**

puis

\App\Entity\ CategorySearch

```
<?php

namespace App\Form;
```

```

use App\Entity\CategorySearch;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use App\Entity\Category;

class CategorySearchType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('category', EntityType::class, ['class' => Category::class,
                'choice_label' => 'titre' ,
                'label' => 'Catégorie' ]);
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => CategorySearch::class,
        ]);
    }
}

```

8. Ajouter au contrôleur IndexController la fonction articlesParCategorie :

```

use App\Entity\CategorySearch;
use App\Form\CategorySearchType;

```

```

/**
 * @Route("/art_cat/", name="article_par_cat")
 * Method({"GET", "POST"})
 */
public function articlesParCategorie(Request $request) {
    $categorySearch = new CategorySearch();
    $form = $this->createForm(CategorySearchType::class, $categorySearch);
    $form->handleRequest($request);

    $articles= [];
}

```

```

        if($form->isSubmitted() && $form->isValid()) {
            $category = $categorySearch->getCategory();

            if ($category!="")
                $articles= $category->getArticles();
            else
                $articles= $this->getDoctrine()->getRepository(Article::class)->findAll();
        }

        return $this->
>render('articles/articlesParCategorie.html.twig',['form' => $form-
>createView(),'articles' => $articles]);
    }

```

9. Créer le fichier articles/articlesParCategorie.html.twig son contenu est le suivant :

```

{% extends 'base.html.twig' %}
{% block title%} Liste des Articles par Catégorie{% endblock %}

{% block body %}
    {{ form_start(form) }}

    <div class="form-row align-items-end" >
        <div class="col">
            {{ form_row(form.category) }}
        </div>

        <div class="col">
            <div class="form-group">
                <button type="submit" class="btn btn-success">Rechercher</button>
            </div>
        </div>
    </div>
    {{ form_end(form) }}

    {% if articles %}
        <table id="articles" class="table table-striped">
            <thead>
                <tr>
                    <th>Nom</th>

```

```
<th>Prix</th>
<th>Catégorie</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
{% for article in articles %}
    <tr>
        <td>{{ article.nom }}</td>
        <td>{{ article.prix }}</td>
        <td>{{ article.category.titre }}</td>
        <td>
            <a href="/article/{{ article.id }}" class="btn btn-
dark">Détails</a>
            <a href="/article/edit/{{ article.id }}" class="btn btn-
dark">Modifier</a>
            <a href="/article/delete/{{ article.id }}" class="btn btn-
danger"
                onclick="return confirm('Etes-
vous sûr de supprimer cet article?');">Supprimer</a>
        </td>
    </tr>
{% endfor %}
</tbody>
</table>
{% else %}
    <p>Aucun articles</p>
{% endif %}
{% endblock %}
```

10. Ajouter au menu (navbar) un lien pour afficher la page `artilcesParCategorie.html.twig` :

```
<ul class="navbar-nav ml-auto">
  <li class="nav-item">
    <a href="/" class="nav-link">Home</a>
  </li>
  <li class="nav-item">
    <a href="{{ path('new_article') }}" class="nav-link">Ajouter article</a>
  </li>
  <li class="nav-item">
    <a href="{{ path('new_category') }}" class="nav-link">Ajouter catégorie</a>
  </li>
  <li class="nav-item">
    <a href="{{ path('article_par_cat') }}" class="nav-link">Recherche par catégorie</a>
  </li>
</ul>
```

11. Testez votre travail :

Localhost / 127.0.0.1 / symfony

127.0.0.1:8000/art_cat/

Nadhem BelHadj

Home Ajouter article Ajouter catégorie Recherche par catégorie

Catégorie

smartphones

Rechercher

Nom	Prix	Catégorie	Actions
iphox XR plus	4000	smartphones	Détails Modifier Supprimer

200 POST @ article... 442 ms 8.0 MB 1 332 in 53.37 ms 1 anon. 40 ms 3 in 1.62 ms 4.4.4

Recherche des articles dont le prix est compris entre deux valeurs

12. Créer dans Entity, la classe PriceSearch sans utiliser la commande *make :entity* :

```
<?php
namespace App\Entity;
class PriceSearch
{
    /**
     * @var int|null
     */
    private $minPrice;

    /**
     * @var int|null
     */
    private $maxPrice;

    public function getMinPrice(): ?int
    {
        return $this->minPrice;
    }

    public function setMinPrice(int $minPrice): self
    {
        $this->minPrice = $minPrice;
        return $this;
    }

    public function getMaxPrice(): ?int
    {
        return $this->maxPrice;
    }

    public function setMaxPrice(int $maxPrice): self
    {
        $this->maxPrice = $maxPrice;
        return $this;
    }
}
```

13. Créer le formulaire PriceSearchType basé sur l'entité PriceSearch

php bin/console make:form

taper **PriceSearchType**

puis

\App\Entity\PriceSearch

```
<?php
namespace App\Form;

use App\Entity\PriceSearch;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class PriceSearchType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('minPrice')
            ->add('maxPrice')
        ;
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => PriceSearch::class,
        ]);
    }
}
```

14. Ajouter la fonction *findByPriceRange* à la classe Repository/ *ArticleRepository* :

```
public function findByPriceRange($minValue,$maxValue)
{
    return $this->createQueryBuilder('a')
        ->andWhere('a.prix >= :minVal')
        ->setParameter('minVal', $minValue)
        ->andWhere('a.prix <= :maxVal')
        ->setParameter('maxVal', $maxValue)
}
```

```
        ->orderBy('a.id', 'ASC')
        ->setMaxResults(10)
        ->getQuery()
        ->getResult()

    ;
}
```

15. Ajouter la fonction articlesParPrix au controlleur IndexController

```
/**
 * @Route("/art_prix/", name="article_par_prix")
 * Method({"GET"})
 */
public function articlesParPrix(Request $request)
{
    $priceSearch = new PriceSearch();
    $form = $this->createForm(PriceSearchType::class,$priceSearch);
    $form->handleRequest($request);

    $articles= [];

    if($form->isSubmitted() && $form->isValid()) {
        $minPrice = $priceSearch->getMinPrice();
        $maxPrice = $priceSearch->getMaxPrice();

        $articles= $this->getDoctrine()->
getRepository(Article::class)->findByPriceRange($minPrice,$maxPrice);
    }

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

16. Créer le fichier `articles/articlesParPrix.html.twig`, son contenu est le suivant :

```
{% extends 'base.html.twig' %}
{% block title%} Liste des Articles par prix{% endblock %}

{% block body %}
{{ form_start(form) }}
<div class="form-row align-items-end" >
  <div class="col">
    {{ form_row(form.minPrice) }}
  </div>
  <div class="col">
    {{ form_row(form.maxPrice) }}
  </div>
  <div class="col">
    <div class="form-group">
      <button type="submit" class="btn btn-success">Rechercher</button>
    </div>
  </div>
</div>
{{ form_end(form) }}

{% if articles %}
<table id="articles" class="table table-striped">
  <thead>
    <tr>
      <th>Nom</th>
      <th>Prix</th>
      <th>Catégorie</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    {% for article in articles %}
      <tr>
        <td>{{ article.nom }}</td>
        <td>{{ article.prix }}</td>
        <td>{{ article.category.titre }}</td>
        <td>
          <a href="/article/{{ article.id }}" class="btn btn-dark">Détails</a>
        </td>
      </tr>
    {% endfor %}
  </tbody>
</table>
{% endif %}
```

```

        <a href="/article/edit/{{ article.id }}" class="btn btn-
dark">Modifier</a>
        <a href="/article/delete/{{ article.id }}" class="btn btn-
danger"
        onclick="return confirm('Etes-
vous sûr de supprimer cet article?');">Supprimer</a>
    </td>
</tr>
{% endfor %}
</tbody>
</table>
{% else %}
    <p>Aucun articles</p>
{% endif %}
{% endblock %}

```

17. Ajouter au menu (navbar) un lien pour afficher la page articlesParPrix.html.twig

```

<li class="nav-item">
    <a href="{{path('article_par_prix')}}" class="nav-link">Recherche par prix</a>
</li>

```

18. Testez :

Min price: 1000, Max price: 4000, Rechercher

Nom	Prix	Catégorie	Actions
samsung galaxy	2555	catégorie 2	Détails Modifier Supprimer
iphox XR plus	4000	smartphones	Détails Modifier Supprimer