## 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;
   }
}
```

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 %}
```

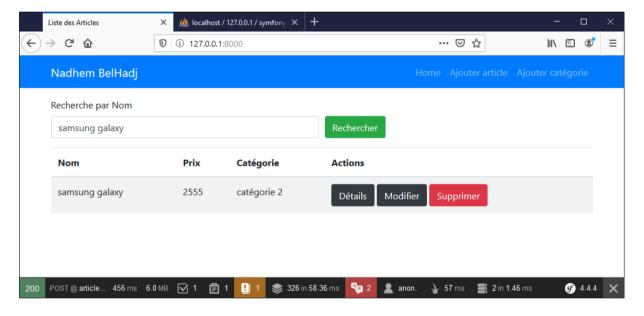
4. Modifier la fonction *home()* du controlleur IndexController comme suit :

```
/**
    *@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 n
om
    $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:



# 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;</pre>
```

```
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 controlleur 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 :

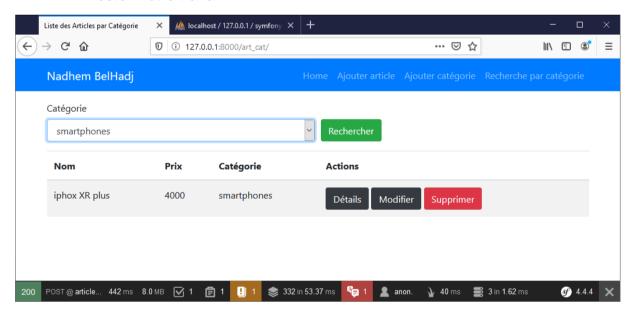
```
{% extends 'base.html.twig' %}
{% block title%} Liste des Articles par Catégorie{% endblock %}
{% block body %}
{{ form_start(form) }}
<div class="form-row align-items-end" >
<div class="col">
  {{ form_row(form.category) }}
</div>
 <div class="col">
  <div class="form-group">
     <button type="submit" class="btn btn-success">Rechercher</button>
   </div>
</div>
</div>
{{ form end(form) }}
{% if articles %}
   Nom
```

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

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

```
<a href="/" class="nav-link">Home</a>
    <a href="{{ path('new_article')}}" class="nav-link">Ajouter article</a>
    <a href="{{ path('new_category')}}" class="nav-</pre>
link">Ajouter catégorie</a>
    <a href="{{ path('article_par_cat')}}" class="nav-</pre>
link">Recherche par catégorie</a>
```

#### 11. Testez votre travail:



# 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

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 %}
   Nom
        Prix
        Catégorie
        Actions
      </thead>
     {% for article in articles %}
        {{ article.nom }}
          {{ article.prix }}
          {{ article.category.titre }}
           <a href="/article/{{ article.id }}" class="btn btn-</pre>
dark">Détails</a>
```

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

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

#### 18. Testez:

