# Bucket-List : formulaire

## Fiche TP du module 6

## Solution

- Générer la classe de formulaire WishType avec `make:form`
    - Associer le formulaire avec l'entité Wish
    - Ajouter les 3 champs

```
...
class WishType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('title', TextType::class, [
                'label' => 'Your idea'
            ])
            ->add('description', TextareaType::class, [
                'label' => 'Please describe it!',
                'required' => false
            ])
            ->add('author', TextType::class, [
                'label' => 'Your username'
            ])
        ;
    }

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

- Dans l'entité Wish, ajouter de la validation sur tous les champs

```
<?php

...
use Symfony\Component\Validator\Constraints as Assert;

/**
 * @ORM\Entity(repositoryClass=WishRepository::class)
 */
class Wish
{
    ...
    /**
     * @ORM\Column(type="string", length=250)
     * @Assert\NotBlank(message="Please provide an idea!")
     * @Assert\Length(
     *      min=5,
     *      max=250,
     *      minMessage="Minimum 5 characters please!",
     *      maxMessage="Maximum 250 characters please!"
```

```
     * )
     */
    private $title;

    /**
     * @ORM\Column(type="text", nullable=true)
     * @Assert\Length(
     *      min=5,
     *      max=5000,
     *      minMessage="Minimum 5 characters please!",
     *      maxMessage="Maximum 5000 characters please!"
     * )
     */
    private $description;

    /**
     * @ORM\Column(type="string", length=50)
     * @Assert\NotBlank(message="Please provide your username!")
     * @Assert\Length(
     *      min=3,
     *      max=50,
     *      minMessage="Minimum 3 characters please!",
     *      maxMessage="Maximum 50 characters please!"
     * )
     * @Assert\Regex(pattern="/^[a-z0-9_-]+$/i", message="Please use only letters, numbers, underscores and dashes!")
     */
    private $author;

    ...
```

- Dans WishController.php :
    - Créer une nouvelle méthode (`create`)
    - Y instancier le formulaire, le traiter et le passer à Twig
    - Si le formulaire est valide :
        - Insérer l'idée dans la base de données avec l'EntityManager
        - Créer un message Flash
        - Rediriger avec `redirectToRoute()`

```
<?php

...
use App\Entity\Wish;
use App\Form\WishType;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\Request;

class WishController extends AbstractController
{
    ...

    /**
     * @Route("/wishes/create", name="wish_create")
     */
    public function create(Request $request, EntityManagerInterface $entityManager): Response
    {
        // notre entité vide
        $wish = new Wish();
        // notre formulaire, associée à l'entité vide
```

```
        $wishForm = $this->createForm(WishType::class, $wish);

        // récupère les données du form et les injecte dans notre $wish
        $wishForm->handleRequest($request);

        // si le formulaire est soumis et valide...
        if ($wishForm->isSubmitted() && $wishForm->isValid()){
            // hydrate les propriétés absentes du formulaires
            $wish->setIsPublished(true);
            $wish->setDateCreated(new \DateTime());

            // sauvegarde en bdd
            $entityManager->persist($wish);
            $entityManager->flush();

            // affiche un message sur la prochaine page
            $this->addFlash('success', 'Idea successfully added!');

            // redirige vers la page de détails de l'idée fraîchement créée
            return $this->redirectToRoute('wish_detail', ['id' => $wish->getId()]);
        }

        // affiche le formulaire
        return $this->render('wish/create.html.twig', [
            'wishForm' => $wishForm->createView()
        ]);
    }
}
```

- Dans create.html.twig, afficher le formulaire en le décomposant le moins possible

```
{% extends 'base.html.twig' %}

{% block body %}
    <h2>Add your wishes!</h2>

    {{ form_start(wishForm, {attr: {novalidate: 'novalidate'}}) }}
    {{ form_widget(wishForm) }}
    <button>Let's go!</button>
    {{ form_end(wishForm) }}
{% endblock %}

{% block title %}Add your wishes | {{ parent() }}{% endblock %}
```

- Modifier le menu et ajouter les messages flash dans base.html.twig :

```
...
    <body>
        {# entête du site #}
        <header>
            <div class="container">
                ...
                <nav class="header-nav">
                    <a href="{{ path('main_home') }}" title="Go back to homepage">Home</a>
                    <a href="{{ path('wish_list') }}" title="All things to do">All wishes</a>
                    <a href="{{ path('wish_create') }}" title="Add your own ideas!">Add yours!</a>
                    <a href="{{ path('main_about_us') }}" title="About us">About us</a>
                </nav>
            </div>
        </header>

        {# contenu principal de chaque page #}
        <main>
            {# affiche les éventuels messages flash #}
```

```
        {% for label, messages in app.flashes %}
            {% for message in messages %}
                <div class="alert alert-{{ label }}">
                    {{ message }}
                </div>
            {% endfor %}
        {% endfor %}
        ...
    </main>

    ...
</body>
```

- Modifier le style dans app.ccs :

```
.alert {
    font-size: 1.5rem;
    font-weight: bold;
    padding: 1rem;
    margin: 1rem;
    border-radius: 5px;
    text-align: center;
}

.alert-success {background-color: #a2a232;}
.alert-error {background-color: #d4550e;}

/* formulaires */
label {
    display: block;
}

input, textarea {
    box-sizing: border-box;
    width: 50%;
    margin-bottom: 1rem;
}

button {
    padding: 0.5rem 0.7rem;
    font-family: inherit;
}
```