Activité 4 : Uploader une image

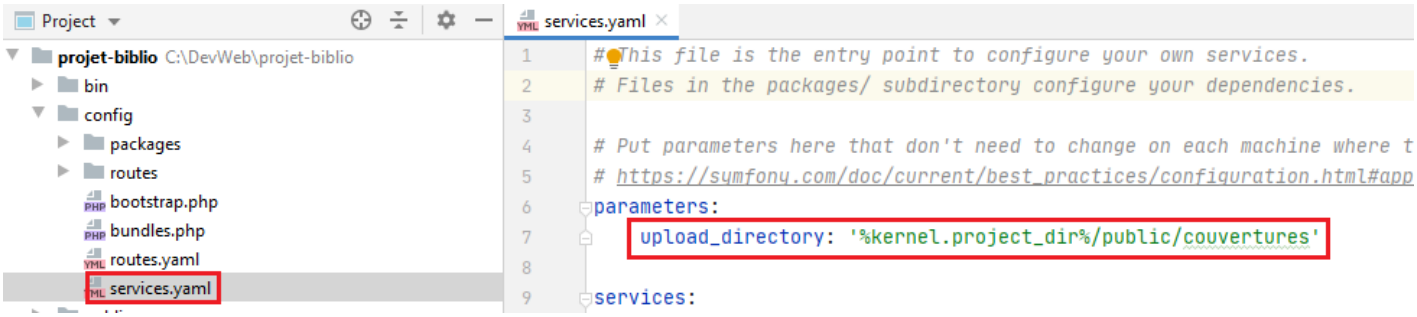
L'objectif de ce TP est d'uploader une image dans une application Symfony. Cette image sera enregistrée dans un dossier sous le dossier public de l'application.

❖ **Création de dossier**

Créer sous le dossier public, le dossier `couvertures` qui contiendra les images uploadés. Placer dans ce dossier l'image `no_image.jpg` qui correspond à une image vide pour les livres qui n'ont pas de couvertures.

❖ **Configuration du chemin d'uplaod**

Editer le fichier de configuration `service.yaml` et ajouter le paramètre correspondant au chemin d'upload `upload_directory` :



❖ **Modifier le fichier LivreType**

Modifier le type du champ image dans le fichier LivreType.php en FileType et rendez le optionnel (non obligatoire) avec data_class null :

```php
class LivreType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('titre')
            ->add('annee')
            ->add('prix')
            ->add('image', FileType::class, array('data_class' => null,
                'required' => false))
```

N'oublier pas d'importer le package de FileType qui est :

```php
use Symfony\Component\Form\Extension\Core\Type\FileType;
```

Dans l'entité `Livre.php`, enlever le type String dans la méthode `setImage` :

```php
    public function setImage($image): self
    {
        $this->image = $image;
```

Enlever le type String devant $image

❖ **Action d'ajout d'un livre**

Dans le contrôleur `LivreController` et précisément dans l'action d'ajout d'un nouveau Livre, ajouter le code des lignes 39 jusqu'à 47 (encadré en rouge):

LivreController.php

```
32      public function new(Request $request): Response
33      {
34          $livre = new Livre();
35          $form = $this->createForm( type: LivreType::class, $livre);
36          $form->handleRequest($request);
37          if ($form->isSubmitted() && $form->isValid())
38          {
39              if($livre->getImage()=="")
40                  $livre->setImage( image: "no_image.jpg");
41              else
42              {
43                  $file = new File($livre->getImage());
44                  $fileName= md5(uniqid()).'.'.$file->guessExtension();
45                  $file->move($this->getParameter( name: 'upload_directory'), $fileName);
46                  $livre->setImage($fileName);
47              }
48              $entityManager = $this->getDoctrine()->getManager();
49              $entityManager->persist($livre);
50              $entityManager->flush();
51
52              return $this->redirectToRoute( route: 'livre_index');
53          }
54          return $this->render( view: 'livre/new.html.twig', [
55              'livre' => $livre,
56              'form' => $form->createView(),
57          ]);
58      }
```

Explication :

- Etant donné que le champ de type Fichier dans le formulaire est optionnel, on commence dans la ligne 39 de tester si ce champ est vide ou pas.
- Dans le cas ou le champ est vide, le nom de l'image de couverture associée par défaut au nouveau livre est `no_image.jpg` (ligne 40).
- Sinon (ligne 41), dans la ligne 43, on crée un objet `$file` de type `File` à partir du nom du fichier uploadé dans le formulaire d'ajout. La classe File doit être importée de :

```
use Symfony\Component\HttpFoundation\File\File;
```

- La ligne 44 génère le nom de l'image `$filename` avec lequel elle sera enregistrée dans le dossier couvertures. C'est une concaténation entre le hachage d'un id unique (pour garantir l'unicité du nom) avec l'extension récupéré de l'image uploadée.

- La ligne 45 déplace le fichier uploadé vers le chemin indiqué dans le paramètre `upload_directory` avec le nom généré par l'instruction précédente.
- La ligne 46 affecte dans l'attribut image du nouveau livre le nom généré `$filename`.

❖ **Action de modification d'un livre**

Dans le contrôleur `LivreController` et précisément dans l'action de modification d'un nouveau Livre, ajouter le code de la ligne 75 et des lignes 79 jusqu'à 90 (encadrés en rouge):

LivreController.php

```
73      public function edit(Request $request, Livre $livre): Response
74      {
75          $name= $livre->getImage();
76          $form = $this->createForm( type: LivreType::class, $livre);
77          $form->handleRequest($request);
78          if ($form->isSubmitted() && $form->isValid()) {
79              if($livre->getImage()=="")
80                  $livre->setImage($name);
81              else
82              {
83                  $file = new File($livre->getImage());
84                  $fileName= md5(uniqid()).'.'.$file->guessExtension();
85                  $file->move($this->getParameter( name: 'upload_directory'), $fileName);
86                  $livre->setImage($fileName);
87                  if($name!="no_image.jpg")
88                      if( file_exists ( filename: "couvertures/".$name))
89                          unlink( filename: "couvertures/".$name ) ;
90              }
91              $this->getDoctrine()->getManager()->flush();
92              return $this->redirectToRoute( route: 'livre_index');
93          }
94          return $this->render( view: 'livre/edit.html.twig', [
95              'livre' => $livre,
96              'form' => $form->createView(),
97          ]);
98      }
```

Explication :

- Dans la ligne 75, on récupère dans `$name` le nom actuel de l'image du livre à modifier.
- Dans le cas ou le champ image est resté vide, le nom de l'image de couverture sera l'ancien nom `$name`. (lignes 79 et 80)
- Sinon (ligne 81), on refait le même traitement de l'upload pendant l'action de l'ajout (lignes de 83 à 86). En modifiant une image on lui génère un nouveau nom.
- Après l'upload d'une nouvelle image, il faut supprimer l'ancienne dans le cas ou elle est différente de `no_image.jpg` (lignes de 87 à 89).