SYMFONY 23/02/2021

1 AJOUT DE L'UPLOAD D'UN FICHIER DANS UN FORMULAIRE

Lier un fichier à une entité revient en fait à relier le nom du fichier et l'endroit où il se trouve. On ne stocke que rarement un fichier dans la base de données.

Ajouter un attribut image à l'entité Produit :

```
symfony console make:entity
➔ nom de l'entité : Produit
➔ nom de l'attribut : image
➔ type : string
➔ peut être null : oui
```

Vous pouvez vérifier que l'attribut image est bien présent dans l'entité Produit.

Vous pouvez ensuite faire une migration :

```
symfony console make:migration
symfony console doctrine:migration:migrate
```

Puis recharger vos fixtures :

```
symfony console doctrine:fixtures:load
```

On ajoute l'upload d'image dans le formulaire sous la forme d'un champs de type File supplémentaire :

```
->add("imageFile", FileType::class,[
                "label" => "Ajouter une photo à votre produit",
                "required" => false,
                "mapped" => false,

                "constraints" => [
                    new Image(["maxSize" => "2048k"])
                ]
            ])
```

Label : permet d'ajouter une indication dans le formulaire

Required : l'ajout d'une image n'est pas obligatoire lorsqu'on crée un produit

Mapped : ce change de type File n'est pas mappé avec la base de données, on remplira après le champ image lié à la base de données, ceci est un champ ajouté pour gérer le fichier.

Constraints : on ajoute une contrainte : le fichier doit être de type image et doit avoir une taille inférieure à 2Mo

Dans le fichier config/packages/services.yaml on ajoute une variable globale qui indique le répertoire où doivent se trouver les fichiers uploadés :

```yaml
    _defaults:
        autowire: true      # Automatically injects dependencies in your services.
        autoconfigure: true # Automatically registers your services as commands, event subscribers, etc.
        bind:
            $imageDir: '%kernel.project_dir%/public/uploads/produits'
```

Dans le controleur, dans la fonction permettant d'ajouter un produit on injecte dans les paramètres de la fonction la variable $imageDir puis on traite l'upload du fichier :

```php
/**
     * @Route("/addproduit", name="addProduit")
     */
    public function creerProduit(Request $request, SluggerInterface $slugger,string $imageDir): Response
    {
        $produit = new Produit;
        $form = $this->createForm(ProduitType::class, $produit);
        $form->add("submit", SubmitType::class, array('label' => 'Créer'));

        $form->handleRequest($request);
        if($form->isSubmitted() && $form->isValid()){
            if($image = $form["imageFile"]->getData()){
                $filename = pathinfo($image->getClientOriginalName(), PATHINFO_FILENAME);
                $safeFilename = $slugger->slug($filename);
                $newFilename = $safeFilename."-".$image->guessExtension();
                try{
                    $image->move($imageDir, $newFilename);
                }
                catch(FileException $e){
                    var_dump($e);
                }
                $produit->setImage($newFilename);
            }
            $this->em->persist($produit);
            $this->em->flush();
            return $this->redirectToRoute("listeProduits",[]);
        }

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

2 UTILISATION DU BUNDLE (PLUGIN) D'ADMINISTRATION AUTOGENEREE

Le bundle easyadmin permet de générer facilement et rapidement une zone d'administration pour gérer les entités de l'application.

Installer le bundle :

```
symfony composer req "admin:^3"
```

Créer dans src/Controller un répertoire Admin

Créer la page de base de l'admin, le dashboard :

```
symfony console make:admin:dashboard
```

Cela génère un contrôleur DashboardControlleur qui va gérer le comportement de base de l'admin.

On peut dans la fonction configureMenuItem gérer les différents liens et catégories de l'admin :

```php
public function configureMenuItems(): iterable
    {
        yield MenuItem::linktoDashboard('Admin', 'fa fa-home');
        yield MenuItem::linktoRoute("Liste des produits", "fa fa-home", "listeProduits");
        yield MenuItem::linkToCrud("Produits", "fa fa-list", Produit::class);
        yield MenuItem::linkToCrud("Commandes", "fa fa-list", Commande::class);
        yield MenuItem::linkToCrud("Catégories", "fa fa-list", Categorie::class);
        yield MenuItem::linkToCrud("Utilisateurs", "fa fa-list", Utilisateur::class);
        // yield MenuItem::linkToCrud('The Label', 'fas fa-list', EntityClass::class);
    }
```

Ici le premier lien renvoi vers l'admin, le deuxième vers une page du site (avec la route listeProduits) et les autres vers la gestion des entités.

On génère un contrôleur CRUD pour chaque entité que l'on veut gérer :

```
symfony console make:admin:crud
```

Dans le contrôleur CRUD on peut gérer plusieurs choses :

La configuration générale :

```php
public function configureCrud(Crud $crud): Crud{
        return $crud->setEntityLabelInSingular("Produit")
                    ->setEntityLabelInPlural("Produits")
                    ->setPageTitle("index","Liste des %entity_label_plural%")
                    ->setSearchFields(["nom", "prix"])
                    ->setDefaultSort(["id" =>"DESC"])
                    ->setPaginatorPageSize(25);
    }
```

Ici on définit les mots pour représenter le singulier et le pluriel de l'entité, on définit le titre de la page. On définit aussi dans quels champs on fait une recherche et par quel attribut on ordonne l'affichage. On peut aussi définir le nombre d'éléments affichés par page (une pagination est automatiquement créée).

On peut définir un ou plusieurs filtres de recherche :

```
public function configureFilters(Filters $filters): Filters{
        return $filters->add('prix')
                       ->add("stock");
    }
```

Enfin on peut donner des propriétés aux différents attributs à afficher :

```
public function configureFields(string $pageName): iterable
    {
        return [
            IdField::new('id')->onlyOnIndex(),
            TextField::new('nom'),
            TextEditorField::new('description'),
            //NumberField::new("prix"),
            MoneyField::new("prix")
                ->setCurrency("EUR")
                ->setNumDecimals(2)
                ->setStoredAsCents(false),
            IntegerField::new("stock"),
            ImageField::new("image")
                ->setLabel("Image")
                ->setBasePath('uploads/produits')
                ->setUploadDir('public/uploads/produits')
        ];
    }
```