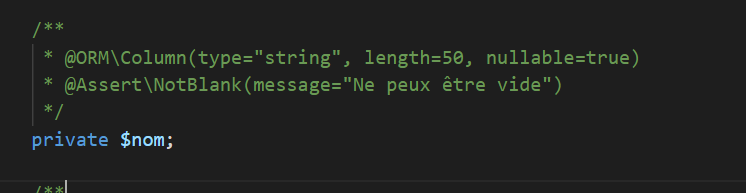
Pour ajouter des contraintes sur les entity qui créer les formulaire on va use un systeme venant de symfony dnas nos entity

use symfony\component\validator\constraints as Assert;

voir et faire un peu de détails : [Validation Constraints Reference (Symfony Docs)](https://symfony.com/doc/current/reference/constraints.html)



pensez a ajouter quelque chose comme

form\_themes: ['bootstrap\_4\_horizontal\_layout.html.twig']

dans la config du bundle twig :)

Pour faire un form de type fichier image etc, on utilisera le bundle : vich/uploader-bundle

créeons un champs dans l’entity qu’il nous faut pour faire le champs image :)

imageName, attention ce package à besoin d’avoir un champs updatedAt de créer

ou je rajoute ceci

    /\*\*

     \* @ORM\Column(type="datetime")

     \*/

    private $updatedAt;

/\*\*

     \* @ORM\PrePersist

     \* @ORM\PreUpdate

     \*/

    public function updateTimestamps()

    {

        $this->setUpdatedAt(new \DateTimeImmutable);

    }

pour mettre a jour ce champ pour chaque modif crée!! attention ici mon champs updatedAt est de type datetime.

Il faut aller configurer le package pour notre projet, -> rdv dans le config/package/vich\_uploader.yaml

On pourra activer les lignes par défaut et ajouter une ligne de namer

    mappings:

       user\_image:

           uri\_prefix: assets/images/user

           upload\_destination: '%kernel.project\_dir%/public/assets/images/user'

           namer: Vich\UploaderBundle\Naming\SmartUniqueNamer

on nomme ici notre mapping, qui s’occupera des images pour nous, on peux en avoir plusieurs !

on lui donne un nom, un préfix pour le liens des images, un dossier ou les mettres et un namer pour créer le nom unique de l’image dans le dossier

Retournons dans l’entity et travaillons notre nouveau champ que doctrine n 'utilisera pas !

Dedans il faudra, dans les annotation, rentrer le nom de notre mapping et la propriété que nous utilisons dans doctrine pour stocker le path de l’image, par defaut c’est imageName

    /\*\*

     \* NOTE: This is not a mapped field of entity metadata, just a simple property.

     \*

     \* @Vich\UploadableField(mapping="user\_image", fileNameProperty="imageName")

     \*

     \* @var File|null

     \*/

    private $imageFile;

    /\*\*

     \* @param File|\Symfony\Component\HttpFoundation\File\UploadedFile|null $imageFile

     \*/

    public function setImageFile(?File $imageFile = null): void

    {

        $this->imageFile = $imageFile;

        if (null !== $imageFile) {

            $this->setUpdatedAt(new \DateTimeImmutable);

        }

    }

    public function getImageFile(): ?File

    {

        return $this->imageFile;

    }

il faudra ajouter ceci au annotation général de l’entity

\* @Vich\Uploadable

et les deux use suivants :

use Symfony\Component\HttpFoundation\File\File;

use Vich\UploaderBundle\Mapping\Annotation as Vich;

Il ne nous reste plus qu’a préparer notre formualrie pour recevoir le champs de vich upload

->add('imageFile', VichImageType::class, [

                'label' => "Image (JPG ou PNG)",

                'required' => false,

                'allow\_delete' => true,

                'delete\_label' => 'Supprimer ?',

                'download\_label' => '...',

                'download\_uri' => true,

                'image\_uri' => true,

                'asset\_helper' => true,

                'attr' => ['class' => "form-control"]

            ])

le label pour l’avant input comme d’hab la class également, le required a false car configurer comme ça dans notre projet normalement.

pour le moment on enleve imagine\_pattern

allow\_delete -> permet de mettre une case a cocher pour delete l’image lors d’une update, ça peremt de ne simplement plus avoir l’image,n (si null is ok dans la db pour le champs !!!)

elle suprime également le fichier du serveur

delete\_label ->permet de metre un texte a coté de la case à cocher de delete

download\_label -> le texte pour lancer le téléchargement de l’image :)

download\_uri -> permet de générer un lien sur l’image pour la téléhcargé a nouveau si l’on veux false pour ne plus pouvoir (ne marche pas tout le temps)

image\_uri -> permet d’afficher lors d’un update, l’image déjà présente

dans notre entity on peux par exemple dire à doctrine de limiter le téléchargement à x mo par exemple

     \* @Assert\Image(maxSize="8M", maxSizeMessage="Trop gros !")

    private $imageFile;

voir la doc pour plus de possibilité : https://symfony.com/doc/current/reference/constraints/Image.html

et son usage pour démarrer : <https://github.com/dustin10/VichUploaderBundle/blob/master/docs/usage.md>

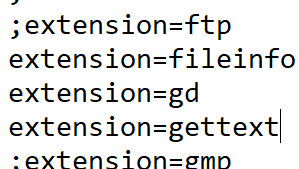
la config pour les image dans le form builder

<https://github.com/dustin10/VichUploaderBundle/blob/master/docs/form/vich_image_type.md>

Pour limite la taille de ‘l’image affichée, on va installer un petit bundle qui nous permettra de faire un tas de choses sur les images affichées.

**symfony composer req liip/imagine-bundle**

il ajoutera cette config pour le bundle, il utilisera par défault la lib php gd pour les images

attention qu’elle n’est peux être pas active par défaut, aller dans les extension de php.ini et décocher la ligne 

redémarer php apache, vider le cache symfony et re-démarrer le serveur



il faudra créer grace a la doc un pattern de filtre pour les images

comme par exemple

    filter\_sets:

        updateThumbnailsUser:

            filters:

                thumbnail:

                    size: [350, 350]

                    mode: outbound

                    allow\_upscale: true

et ensuite retourner dans notre formulaire et rajouter notre ligne “imagine\_pattern” au niveau de l’image !

par exemple :

            ->add('imageFile', VichImageType::class, [

                'label' => "Image (JPG ou PNG)",

                'required' => false,

                'allow\_delete' => true,

                'delete\_label' => 'Supprimer ?',

                'download\_label' => '',

                'download\_uri' => false,

                'image\_uri' => true,

                'asset\_helper' => true,

                'imagine\_pattern' => 'updateThumbnailsUser',

                'attr' => ['class' => "form-control"]

            ])

Pour afficher dans des templates les images que nous avons uploader on peux le faire à l’ancienne avec un path,

mais on peux également utiliser la fonction twig qu’a rajouter le bundle !!

<img src="{{ vich\_uploader\_asset(user, 'imageFile') | imagine\_filter('updateThumbnailsUser') }}" class="card-img-top" alt="...">

ici vich demande l’entity tout simplement, et le nom de la prop au cas ou ça ne marche pas !

on peux également utilser un filtre twig que imagine\_filter nous à fourni ! qui prendra en paramètre le filtre pour adapté l’image, ou alors vous le faite à l’ancienne !

            <img src="/assets/images/user/{{user.imageName}}" class="card-img-top" style="height:150px; width:150px;" alt="...">

ça marche tout aussi bien !

VOIR DEMO UPLOAD\_DEMO