



ADRAR\o/ PÔLE NUMÉRIQUE

- > **INFRASTRUCTURES** SYSTÈMES & RÉSEAUX
- > **CYBERSÉCURITÉ** INFRASTRUCTURES & APPLICATIONS
- > **DEVOPS / SCRIPTING** & AUTOMATISATION
- > **DEVELOPPEMENT** WEB & MOBILE
- > TRANSFORMATION NUMERIQUE DES ENTREPRISES

www.adrar-numerique.com

Nous avons déjà vu les formulaires se construisant à l'aide du form builder.

Nous allons voir comment personnaliser les champs avec des attributs et des types différents:

```
->add('prenom', TextType::class, [  
    'attr' => ['class' => 'form-control'], // on appliquera la classe "form-control" à notre input  
])  
  
->add('permis', FileType::class, [  
    'mapped' => false, // on ignore l'input lors de la lecture/écriture de l'objet  
    'required' => false, // on rend le champs non obligatoire  
    'attr' => [  
        'class' => 'form-file',  
        'accept' => 'image/*' // on décide de n'accepter que les fichiers de type image  
    ],  
    'multiple' => false // on ne rend pas possible la multi-sélection  
])  
  
->add('age', IntegerType::class, [  
    'attr' => ['min' => '18'], // on appliquera un minimum de 18 en valeur à notre input  
])  
)
```

UtilisateurType.php

Pour personnaliser les labels, nous procéderons comme ci-après:

```
{{ form_start(registrationForm) }}  
    {{ form_row(registrationForm.prenom, {  
        label: 'Prénom', {# Cette ligne permet de changer le texte par défaut du label #}  
        label_attr: {  
            class: 'label' {# Cette ligne ajoutera la classe "label" à notre label #}  
        }  
    }) }}  
{{ form_end(registrationForm) }}
```

inscription.html.twig

Vous avez la possibilité de charger le formulaire de cette autre manière plutôt que ligne par ligne

```
{{ form_start(form) }}  
    {{ form_widget(form) }}  
{{ form_end(form) }}
```

inscription.html.twig

Sinon, vous pouvez directement appeler la fonction “form” qui générera les éléments automatiquement par rapport au **BuilderForm**

```
{{ form(registrationForm) }}
```

inscription.html.twig

Nous avons déjà vu les formulaires se construisant à l'aide du form builder.

Ici nous allons voir comment aller encore plus loin dans la confection de ceux-ci.
Par exemple, imaginons que nous voulions avoir accès à une entité B dans un formulaire d'ajout d'une entité A:

```
// Ajoutez ce use en haut de votre fichier
use Symfony\Bridge\Doctrine\Form\Type\EntityType;

// ...
// Puis ajoutez cela à votre FormBuilder en remplaçant les éléments entre crochets
->add('niveau',
    EntityType::class,
    array('class' => 'App\Entity\[MON_ENTITE]',
        'choice_label' => '[MA_COLONNE]',
        'placeholder' => '[Le texte affiché par défaut]')
)
```

AFormType.php

Avec ce code, vous verrez que votre formulaire possède désormais une liste déroulante de votre entité.

NB: n'oubliez pas de changer ce qui est entre crochets

Insertion d'image (1/4)

Symfony

```
// ...
use Symfony\Component\Form\Extension\Core\Type\FileType;
// ...
public function buildForm(FormBuilderInterface $builder, array $options) {
    $builder
        ->add('avatar', FileType::class, [
            'label' => 'Image (tout types)',
            'mapped' => false, // Cette ligne signifie que ce champs ne sera associé à aucune propriété de l'entité
            'required' => false, // Permet de ne pas rendre ce champs obligatoire
            'attr' => ['accept' => 'image/*'] // Accepte tous les fichiers de type image
        ]);
}
```

UtilisateurType.php

Si côté Twig vous utilisez la fonction **form_widget()** vous n'avez rien d'autre à faire de ce côté sinon vous devez ajouter un nouveau **form_row()** pour ce nouvel input.

Insertion d'image (2/4) Symfony

On définit un répertoire avec un nom dans lequel nos fichiers seront envoyés:

```

parameters:
    avatar_directory: '%kernel.project_dir%/public/uploads/avatars'
  
```

config/services.yaml

Ensuite, nous allons traiter notre donnée dans notre contrôleur.

Lors de la vérification de la validité du formulaire, si cette condition est vérifiée nous allons ajouter ce code:

```

// ...
use Symfony\Component\HttpFoundation\Exception\FileException;
use Symfony\Component/String\Slugger\SluggerInterface; // Ajoutez ce use
// ...
#[Route('/utilisateur/ajout', name: 'app_utilisateur_ajout')]
public function ajout(Request $request, SluggerInterface $slugger) { // Ajoutez le slugger en paramètre de votre méthode
    // Ajoutez également EntityManagerInterface $entityManager (plus use) si vous n'utilisez pas les repositories, sinon passez le
    repository concerné en paramètre
    // ...
    if ($form->isSubmitted() && $form->isValid()) {
        $avatarFichier = $form->get('brochure')->getData(); // On récupère les données qui composent l'image
        if ($avatarFichier) { // Si une image a bien été insérée (vu qu'elle n'est pas required, elle peut être vide)
            $originalFilename = pathinfo($avatarFichier->getClientOriginalName(), PATHINFO_FILENAME); // On prend le nom de base du fichier
            // this is needed to safely include the file name as part of the URL
            $safeFilename = $slugger->slug($originalFilename);
            $newFilename = $safeFilename.'-'.uniqid().'.'.$avatarFichier->guessExtension();
  
```

UtilisateurController.php

Insertion d'image (3/4)

Symfony

Ensuite, nous allons traiter notre donnée dans notre contrôleur.

Lors de la vérification de la validité du formulaire, si cette condition est vérifiée nous allons ajouter ce code:

```
// Tente de déplacer le fichier vers le répertoire définit plus tôt
try {
    $avatarFichier->move(
        $this->getParameter('avatar_directory'),
        $newFilename
    );
} catch (FileNotFoundException $e) {
    // Gérer le cas en cas d'exception levée (droits insuffisants, stockage insuffisant, ...)
}
$utilisateur->setImage($newFilename); // On redéfinit l'image de notre objet pour permettre l'enregistrement du bon nom d'image
en BDD
}
$utilisateurRepository->save($utilisateur); // Version avec le UtilisateurRepository
$entityManager->persist($utilisateur); // Version avec l'EntityManager
$entityManager->flush(); // Version avec l'EntityManager
return $this->redirectToRoute('app_utilisateur_profil');
}
return $this->render('utilisateur/ajout.html.twig', [
    'form' => $form,
]);
}
```

UtilisateurController.php

Insertion d'image (4/4)

Symfony

Enfin, pour afficher votre image, vous devrez à nouveau vous servir de la fonction **asset()** avec le lien du dossier et la concaténation du message

```
{# ... #}                                     utilisateur/moncompte.html.twig  
  
{# ... #}
```