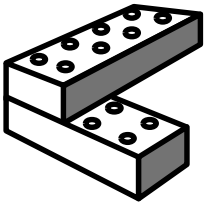

Travaux Pratiques : Enoncés
TP_ReviewPhoto_Partie_10

Objectif : **Consolidation sur le framework Symfony 6**
Création de commentaires

Une fois l'utilisateur connecté, il doit pouvoir commenter une photo à partir de la vue « show.html.twig » du dossier « templates/photo ».

Pour cela, on va ajouter un formulaire qui ne sera visible que pour les utilisateurs identifiés.

Réalisations :

1 : Création du formulaire de commentaire :

On va utiliser le « maker » de Symfony puis on corrigera le fichier obtenu.

- **Entrer la commande**

  php bin/console make:form

- **Donner les informations**

  - **Nom : CommentType**
  - **Classe : Comment**

Ceci va créer le fichier « CommentType.php » dans le dossier « Form ». Il contient la classe de définition de notre formulaire. C'est cette classe qu'il faut modifier car on ne veut pas que l'utilisateur saisisse toutes les propriétés de la classe 'Comment ».

2 : Modification de la méthode « buildForm() » de la classe « CommentType » :

Pour l'instant, le contenu doit ressembler à ça :

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('content')
        ->add('create_at')
        ->add('user')
        ->add('photo')
    ;
}
```

Seule la propriété « content » sera saisie par l'utilisateur. En effet :

- **la propriété « create_at » sera valorisée par la date système**
- **la propriété « user » sera valorisée par l'objet « User » de l'utilisateur connecté**
- **la propriété « photo » sera valorisée par l'objet « Photo » en cours**

code final de la méthode :

```
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('content',TextareaType::class)    // mettre le use correpondant à TexrareaType
//          ->add('create_at')
//          ->add('user')
//          ->add('photo')
        ;
    }
```

L'ajout de la classe « TextareaType » du namespace « Symfony\Component\Form\Extension\Core\Type » permet que le rendu de la balise html « input » soit de type « textarea ».

3 : Modification de la méthode « show » du contrôleur « PhotoController »

C'est bien lors de l'affichage d'une photo que l'on veut, si l'utilisateur est identifié, afficher le formulaire de saisie d'un commentaire.

Pour l'instant, la logique de la méthode est :

Récupération de la photo par l'id se trouvant dans la requête
Affichage de la vue « show.html.twig » en lui passant la photo

On va modifier la logique comme suit :

récupération de la photo par l'id se trouvant dans la requête

Si (l'utilsateur est connecté) Alors
récupération de l'utilisateur connecté
création d'un nouveau commentaire (vide)
valorisation de la propriété « user » par l'utilisateur récupéré
valorisation de la propriété « photo » par la photo récupérée
valorisation de la propriété « create_at » par la date système

Si (c'est la soumission du formulaire et qu'il est valide) Alors
enregistrement du commentaire en base
redirection vers la vue « show.html.twig » avec l'id de la photo
Sinon
affichage de la vue « show.html.twig » en lui passant la photo et le formulaire
Finsi
Sinon
affichage de la vue « show.html.twig » en lui passant la photo
Finsi

Pour savoir si l'utilisateur est identifié ou non, il faut à partir d'un contrôleur appeler la méthode « getUser() » héritée de la classe « App »
Elle renvoi « null » si l'utilisateur est non identifié, dans le cas contraire elle renvoi une instance de la classe « User » dont toutes les propriétés sont valorisées (récupérées par la session, voir la méthode serialize() de la classe).

Le code ressemblera donc à ça :

```
public function show(Photo $photo, EntityManagerInterface $em) {

    //dump($this->getUser());      // affiche null ou l'objet de la classe User

    if ($this->getUser()) {

        $user = $this->getUser();   // récupération du User connecté

        $comment = new \App\Entity\Comment();                 // création d'un nouveau commentaire
        $comment->setUser($user);                             // valorisation de la propriété « user »
        $comment->setPhoto($photo);                           // valorisation de la propriété « photo »
        $comment->setCreate_At(new \DateTimeImmutable());         // valorisation de la propriété « create_at
»

        $form = $this->createForm(\App\Form\CommentType::class, $comment);       // création de l'objet form

        $form->handleRequest($request);   // on valorise les champs avec les valeurs présentes dans la requête

        if ($form->isSubmitted() && $form->isValid()) {       // si c'est la soumission et qu'il est valide
            $em->persist($comment);                           // préparation à l'enregistrement
            $em->flush();                                     // exécution de l'enregistrement en base

            return $this->redirectToRoute('photo.show', ['id' => $photo->get('id')]);      // retour à la vue de la photo
        } else {
            return $this->render('photo/show.html.twig', [       // affichage de la vue photo avec le formulaire
                'photo' => $photo,
                'form' => $form->createView(),
            ]);
        }
    } else {
        return $this->render('photo/show.html.twig', [          // affichage de la vue avec la photo
                'photo' => $photo,
        ]);
    }
}
```

4 : Modifier la vue « show.html.twig » pour afficher le formulaire :

Dans cette vue il va falloir aussi tester si l'utilisateur est connecté ou non car elle reçoit que la photo ou la photo et le commentaire.

On peut le faire par un test twig sur « app.user » qui appelle en fait « app.getApp() » ou faire appel à « is_granted('nom du rôle') que l'on veut tester.

Le code sera donc de modifier la div qui affiche le lien « Mettre un commentaire » :

```
{% if app.user %}
   <a href="#" class="btn btn-primary">Mettre un commentaire</a>
   <div id="commentForm" class="mt-4">
      {{ form_start(form) }}
      <div class="row">
         <div class="col">{{ form_row(form.content) }}</div>
       </div>
      <div class="form-group">
         <button class="btn btn-primary">Envoyer</button>
      </div>
      {{ form_end(form) }}
   </div>
{% endif %}
```

On peut vouloir enlever le label « content » devant le textarea, pour cela modifier la méthode « buildForm() » de la classe « CommentType » comme suit :

```
$builder
   ->add('content',TextareaType::class, ['label' => false])
```

5 : ne pas oublier d'afficher sur les vues « list.html.twig » et « manage.html.twig » le nombre de commentaires déjà fait :

Modifier la vue et là où on avait « commentaires 0 », mettre à la place :

```
commentaires : {{ photo.comments|length }}
```

« photo.comments » fait en fait appel à la méthode « getComments() » de la classe « Photo » qui retourne le tableau des commentaires et le filtre « length » calcul le nombre d'éléments.

6 : tester