La gestion du formulaire dans le contrôleur et son affichage

[sommaire]

1. **Objectifs**

- Gérer la création, la soumission et l'affichage du formulaire dans le contrôleur Symfony, ainsi que son rendu dans la vue Twig associée.

- **Présentation**

  - Le contrôleur joue un rôle clé dans l’architecture MVC (Modèle-Vue-Contrôleur) de Symfony. Il va envoyer à la vue le formulaire et va ensuite traiter les informations recueillies lors de la soumission, du formulaire de l’utilisateur. Découvrons le fonctionnement du contrôleur, avec quelques exemples de sa gestion d’un formulaire, ainsi que les différentes façons d’utiliser la vue du formulaire dans un fichier Twig.

- **createFormBuilder**

  - La méthode **`createFormBuilder()`** fournie par AbstractController permet de créer un formulaire directement dans le contrôleur, sans avoir besoin de définir un Type de formulaire séparément. Cela peut être pratique pour des cas simples où la structure du formulaire est simple et ne nécessite pas une définition complexe.

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\HttpFoundation\Request;

class ExampleController extends AbstractController
{
    public function createFormAction(Request $request)
    {
        // Création d'un formulaire avec createFormBuilder()
        $form = $this->createFormBuilder()
            ->add('nom', TextType::class)
```

apcpedagogie.com

```
            ->add('email', TextType::class)
            ->add('save', SubmitType::class, ['label' => 'Enregistrer'])
            ->getForm();

        // Traitement du formulaire soumis
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            // Traiter les données du formulaire ici
            $formData = $form->getData();
            // ...

            // Redirection ou rendu d'une vue
            return $this->redirectToRoute('route_name');
        }

        // Rendu du formulaire dans une vue
        return $this->render('example/form.html.twig', [
            'form' => $form->createView(),
        ]);
    }
}
```

- Dans cet exemple, `createFormBuilder()` est utilisé pour créer un formulaire avec deux champs (nom et email) et un bouton de soumission. Une fois le formulaire soumis, les données sont récupérées et peuvent être traitées selon les besoins de l'application. Ensuite, le contrôleur peut effectuer une redirection ou rendre une vue pour afficher le résultat du traitement.

Formulaire de rendu

- Pour afficher le formulaire dans votre application, vous pouvez utiliser la méthode createForm() en passant en argument le type de formulaire que vous avez défini. Assurez-vous d'importer le type de formulaire requis pour pouvoir l'utiliser correctement. Voici comment vous pouvez le faire :

```
use App\Form\CommentType; // Assurez-vous d'importer le type de formulaire nécessaire

class ExampleController extends AbstractController
{
    public function renderFormAction(Request $request)
    {
```

apcpedagogie.com

```
        // Création du formulaire en utilisant createForm() et en passant le type de
formulaire en argument
        $form = $this->createForm(CommentType::class);

        // Rendu du formulaire dans une vue
        return $this->render('example/form.html.twig', [
            'form' => $form->createView(),
        ]);
    }
}
```

- Dans cet exemple, CommentType::class est passé en argument à la méthode `createForm()` pour créer le formulaire. Assurez-vous d'avoir importé correctement **CommentType** dans votre contrôleur pour pouvoir l'utiliser.
- Une fois le formulaire créé, vous pouvez le passer à votre vue pour qu'il soit rendu correctement.
- Assurez-vous également d'avoir défini le template Twig correspondant pour rendre le formulaire dans la vue **example/form.html.twig**.

- ## La vue

  - Pour afficher un formulaire dans une vue Twig, vous pouvez utiliser plusieurs méthodes, en fonction de vos besoins de personnalisation et de rendu.
  - Voici quelques exemples avec leurs explications :

1. **Exemple 1 :**

```
{% block body %}
   {{ form(form) }}

{% endblock %}
```

- Dans cet exemple, la fonction form() de Symfony est utilisée pour rendre les balises HTML

- **Exemple 2 :**

apcpedagogie.com

```
{% block body %}
   {{ form_start(form) }}
       {{ form_label(form.title) }}
       {{ form_widget(form.title) }}
       {{ form_label(form.content) }}
       {{ form_widget(form.content) }}

   {{ form_end(form) }}
{% endblock %}
```

- Ici, le formulaire est rendu de manière plus détaillée. On commence le formulaire avec form_start() et on le termine avec form_end(). Pour chaque champ, on utilise form_label() pour afficher le label, et form_widget() pour afficher l'input. Enfin, le bouton de soumission est ajouté manuellement.

- **Exemple 3 :**

```
{% block body %}
   {{ form_start(form) }}
       {{ form_row(form.title) }}
       {{ form_row(form.content) }}

   {{ form_end(form) }}
{% endblock %}
```

- Dans cet exemple, form_row() est utilisé pour rendre chaque champ individuellement. Cette méthode combine les fonctions form_label(), form_widget(), et form_errors(). Elle est utile lorsque vous souhaitez ajouter des options visuelles de façon individuelle à chaque champ.

- **Exemple 4 :**

```
{% block body %}
   {{ form_start(form) }}
       {{ form_row(form.title) }}
       {{ form_row(form.content, {'attr': {'class': 'form-control', 'placeholder':
'Commentaire'}}) }}

   {{ form_end(form) }}
```

apcpedagogie.com

```
{% endblock %}
```

- Dans cet exemple, des attributs sont ajoutés au champ form.content directement dans la vue. Cela peut être utile si vous souhaitez ajouter des options spécifiques à un champ sans avoir à les définir dans le formulaire Type ou le contrôleur.
- Choisissez la méthode qui correspond le mieux à vos besoins et à la structure de votre projet, en gardant à l'esprit les principes de maintenabilité et de bonnes pratiques de développement.

La gestion du formulaire dans le contrôleur et son affichage

- La gestion du formulaire dans le contrôleur et son affichage joue un rôle crucial dans le processus de développement d'une application Symfony.
- Voici une explication détaillée de chaque étape, illustrée par des exemples :

1. **Création du formulaire avec createFormBuilder**

- Dans l'exemple suivant, nous utilisons la méthode createFormBuilder() pour créer un formulaire directement dans le contrôleur :

```
#[Route('/comment')]
class CommentController extends AbstractController
{
    #[Route('/new', name: 'app_comment_new', methods: ['GET', 'POST'])]
    public function new(Request $request): Response
    {
        $comment = new Comment();
        $form = $this->createFormBuilder($comment)
            ->add('title', TextType::class, [
                'attr' => [
                    'class' => 'form-control mt-3 mb-3',
                ],
                'label' => 'Titre du commentaire',
                'required' => true
            ])
            ->add('content', TextareaType::class, [
                'attr' => [
                    'class' => 'form-control mt-3 mb-3',
                ],
                'label' => 'Commentaire',
                'required' => true
```

apcpedagogie.com

```
            ])
            ->getForm();

        // Autres actions...

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

- **Affichage du formulaire dans la vue**
    - Une fois le formulaire créé dans le contrôleur, nous le passons à la vue Twig pour l'afficher. Voici deux façons courantes de le faire :
    - Méthode 1: Utilisation de form(form)

```
{% block body %}
    {{ form(form) }}

{% endblock %}
```

- **Méthode 2: Utilisation de form_start, form_row et form_end**

```
{% block body %}
    {{ form_start(form) }}
        {{ form_row(form.title) }}
        {{ form_row(form.content) }}

    {{ form_end(form) }}
{% endblock %}
```

- Dans ces deux méthodes, le formulaire est affiché dans la vue Twig en utilisant les fonctions form_start, form_row, et form_end, qui génèrent automatiquement les balises HTML nécessaires pour chaque champ du formulaire.

- **Traitement des données du formulaire**
    - Symfony recommande de traiter les données du formulaire dans la même méthode que le rendu. Voici comment compléter la méthode new() du contrôleur pour traiter les données soumises :

apcpedagogie.com

```
#[Route('/comment')]
class CommentController extends AbstractController
{
    #[Route('/new', name: 'app_comment_new', methods: ['GET', 'POST'])]
    public function new(Request $request, EntityManagerInterface $entityManager): Response
    {
        $comment = new Comment();
        $form = $this->createForm(CommentType::class, $comment);

        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $entityManager->persist($comment);
            $entityManager->flush();

            return $this->redirectToRoute('comment_list');
        }

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

- Dans cette méthode, nous utilisons handleRequest() pour traiter les données soumises. Ensuite, nous vérifions si le formulaire a été soumis et s'il est valide. Si c'est le cas, nous persistons les données en base de données et redirigeons l'utilisateur vers une autre page.
- En suivant ces étapes, vous pouvez efficacement gérer la création, l'affichage et le traitement des formulaires dans Symfony.

Sommaire du cours Symfony

Source : https://apcpedagogie.com/la-gestion-du-formulaire-dans-le-controleur-et-son-affichage · téléchargé le 03/11/2026 · © APC Pédagogie · apcpedagogie.com